SANDIA REPORT
SAND2020-11536 - Unlimited Release

Printed October 20, 2020

Sandia National Laboratories

SIERRA Low Mach Module: Fuego User Manual – Version 4.58

SIERRA Thermal/Fluid Development Team

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods

ABSTRACT

The SIERRA Low Mach Module: Fuego, henceforth referred to as Fuego, is the key element of the ASC fire environment simulation project. The fire environment simulation project is directed at characterizing both open large-scale pool fires and building enclosure fires. Fuego represents the turbulent, buoyantly-driven incompressible flow, heat transfer, mass transfer, combustion, soot, and absorption coefficient model portion of the simulation software. Using MPMD coupling, Scefire and Nalu handle the participating-media thermal radiation mechanics. This project is an integral part of the SIERRA multi-mechanics software development project. Fuego depends heavily upon the core architecture developments provided by SIERRA for massively parallel computing, solution adaptivity, and mechanics coupling on unstructured grids.

ACKNOWLEDGMENTS

The following Sandia staff have been involved with the Fuego project, either in code development, math model definitions, or requirements definition: Colin Aro, Amalia Black, Alex Brown, Shawn Burns, Bob Cochran, Stefan Domino, Lindsay Erickson, Greg Evans, David Glaze, Lou Gritzo, John Hewson, Bill Houf, Robert Knaus, Adrian Kopacz, Mario Martinez, Chris Moen, Elijah Newren, Vern Nicolette, Flint Pierce, Philip Sakievich, James Sutherland, Warren Tauber, Jeremy Templeton, Sheldon Tieszen, Tyler Voskuilen, and Greg Wagner.

Contents

Acknowledgments 4

Nomenclature 11

English Character Symbols . . . 11
Greek Character Symbols . . . 13
Superscript Character Symbols . . . 14
Subscript Character Symbols . . . 14
Dimensionless Groups . . . 15

1 Introduction 17

1.1 Abnormal Thermal Environments . . . 17
1.2 Production Capabilities . . . 18
1.3 Document Organization . . . 21

2 Math Models 22

2.1 Laminar Flow Equations . . . 22
2.2 Radiation Transport Equation . . . 28
2.3 Turbulence Modeling Overview . . . 30
2.4 Turbulent Flow Equations, Favre-Averaged . . . 33
2.5 Turbulence Closure Models . . . 38
2.6 Wall Boundary Conditions for Turbulence Models . . . 49
2.7 Inlet Conditions for Turbulence Quantities . . . 58
2.8 EDC Turbulent Combustion Model . . . 59
2.9 Laminar Flamelet Turbulent Combustion Model . . . 60
2.10 Turbulent Reacting Mixing Models . . . 79
2.11 Volume of Fluid Model . . . 80
2.12 One-Dimensional Composite Fire Boundary Condition . . . 86
2.13 Fire Suppression Model . . . 96
2.14 Soot Generation Model for Multicomponent Combustion . . . 98
2.15 Virtual Thermocouple Model . . . 99
2.16 Units and Unit Conversions . . . 101
2.17 Particle Wall Interaction Models . . . 102

3 Numerics 106

3.1 Flow Solver . . . 107
3.2 Convection Operator . . . 107

3.3 Segregated Solution Procedure . . . 109
3.4 Conjugate Heat Transfer . . . 111

4 MPMD Radiation 112
4.1 Input Deck Conversion . . . 112
4.2 Spectral Radiation Transport . . . 120
4.3 Particle Radiation Terms . . . 122
4.4 Beam (Directed Flux) . . . 123

5 User Subroutines 125

6 SIERRA Abstractions 129
6.1 Framework Procedure . . . 129
6.2 Framework Region . . . 129

7 Domain 132
7.1 Sierra . . . 132
7.2 Aztec Equation Solver . . . 139
7.3 Definition For Function . . . 148
7.4 Values . . . 157
7.5 Finite Element Model . . . 158
7.6 Parameters For Block . . . 162
7.7 Global Constants . . . 173
7.8 Output Scheduler . . . 175
7.9 Property Specification For Fuego Material . . . 179
7.10 Tabular Property Library . . . 194
7.11 Property Specification For Material . . . 199
7.12 Trilinos Equation Solver . . . 200
7.13 Teuchos Parameter Block . . . 209
7.14 Tpetra Equation Solver . . . 211
7.15 Klu2 Solver . . . 213
7.16 Bicgstab Solver . . . 214
7.17 Gmres Solver . . . 215
7.18 Umfpack Solver . . . 217
7.19 Expert Belos Solver . . . 217
7.20 Superlu Solver . . . 218
7.21 Cg Solver . . . 218
7.22 Preset Solver . . . 220
7.23 Dd-Ilu Preconditioner . . . 221
7.24 Jacobi Preconditioner . . . 222
7.25 Dd-Ilut Preconditioner . . . 223
7.26 Muelu Preconditioner . . . 224
7.27 Sgs Preconditioner . . . 225
7.28 Expert Ifpack2 Preconditioner . . . 225

8 Procedure **227**
8.1 Fuego Procedure . . . 227
8.2 Apub Transfer . . . 230
8.3 Transfer . . . 233

9 Time Control **244**
9.1 Time Control . . . 244
9.2 Time Stepping Block . . . 245
9.3 Parameters For Fuego Region . . . 246

10 Solution Control **250**
10.1 Solution Control Description . . . 250
10.2 Initialize . . . 251
10.3 Parameters For . . . 252
10.4 Parameters For Fuego Region . . . 256
10.5 System . . . 259
10.6 Adaptivity . . . 264
10.7 Sequential . . . 267
10.8 Transient . . . 272
10.9 Nonlinear . . . 276
10.10 Subcycle . . . 280

11 Fuego Region **284**
11.1 Fuego Region . . . 284
11.2 Averaging . . . 303
11.3 Heartbeat . . . 308
11.4 History Output . . . 318
11.5 Restart Data . . . 326
11.6 Results Output . . . 336
11.7 Solid Object . . . 349
11.8 Virtual Thermocouple Model On Block . . . 350
11.9 Solution Options . . . 353
11.10 Buoyancy Model Specification . . . 378
11.11 Edc Model Specification . . . 380
11.12 Multiphase Model Specification . . . 386
11.13 Radiation Transport Equation Model Specification . . . 387
11.14 Time Integration Specification . . . 390
11.15 Turbulence Model Specification . . . 392
11.16 Acoustic Transfer Output . . . 400
11.17 Point Source . . . 403
11.18 Mesh Motion Specification . . . 405
11.19 Rigid Body Motion Model . . . 407
11.20 Vof Model Specification . . . 414

12 Particle Region **418**

12.1 Particle Region . . . 418
12.2 Create Initial Particle Distribution . . . 428
12.3 Create Particles From File . . . 430
12.4 Fluid Variable Specification . . . 435
12.5 Heartbeat . . . 438
12.6 History Output . . . 448
12.7 Insert Particle . . . 456
12.8 Particle Average . . . 462
12.9 Particle Breakup Model . . . 463
12.10 Particle Definition . . . 465
12.11 Particle Filled Shape . . . 469
12.12 Particle Filled Tree . . . 476
12.13 Crown Info . . . 479
12.14 Trunk Info . . . 483
12.15 Particle Inflow Boundary Condition On Surface . . . 486
12.16 Particle Interface . . . 490
12.17 Particle Evaporation . . . 492
12.18 Particle Reaction . . . 493
12.19 Particle Material . . . 494
12.20 Particle Open Boundary Condition On Surface . . . 495
12.21 Particle Spray . . . 495
12.22 Particle Tabular Output . . . 507
12.23 Particle Wall Boundary Condition On Surface . . . 508
12.24 Restart Data . . . 511
12.25 Results Output . . . 521

13 Average Region **535**
13.1 Average Region . . . 535
13.2 History Output . . . 536
13.3 Restart Data . . . 544
13.4 Results Output . . . 554
13.5 Solution Options . . . 567

14 Input/Output Region **593**
14.1 Input_Output Region . . . 593
14.2 Results Output . . . 597

15 ODE Solver Specifications **611**
15.1 ODE Solver Parameters . . . 611

16 Chemistry Descriptions **614**
16.1 Reaction Block Overview . . . 614
16.2 Reaction Units . . . 614
16.3 Reaction Inputs . . . 615

17 Fuego Initial Conditions **625**
17.1 Initial Condition Block . . . 625

18 Fuego Boundary Conditions **634**
18.1 Fixed Boundary Condition On Surface . . . 634
18.2 Heat Flux Boundary Condition On Surface . . . 647
18.3 Inflow Boundary Condition On Surface . . . 651
18.4 Mass Flux Boundary Condition On Surface . . . 667
18.5 Open Boundary Condition On Surface . . . 689
18.6 Symmetry Boundary Condition On Surface . . . 709
18.7 Wall Boundary Condition On Surface . . . 723
18.8 Wall Mass Inject Boundary Condition On Surface . . . 741
18.9 Post Process On Surface . . . 759
18.10 Composite Boundary Condition On Surface . . . 760
18.11 Composite Interface Boundary Condition On Surface . . . 779
18.12 Nonconformal Boundary Condition On Surface . . . 796

19 String Functions **801**
19.1 Expression Evaluation . . . 801

20 Platform Instructions **809**

21 Fuego Output **810**
21.1 LOG File . . . 810

22 Frequently Asked Questions **813**

Bibliography **823**

Index **829**

List of Figures

2.2-1 Ordinate Direction Definition, $\mathbf{s} = \sin\theta_{zn}\sin\theta_{az}\mathbf{i} + \cos\theta_{zn}\mathbf{j} + \sin\theta_{zn}\cos\theta_{az}\mathbf{k}$ . . . 29
2.8-1 Model geometry for Magnussen's Eddy Dissipation Concept. The control volume is comprised of two zones; the properties of each zone are assumed to be adequately represented by a single set of values (i.e., lumped or perfectly stirred). The mass exchange between the zones is controlled by turbulent mixing. . . . 60
2.12-1 Representative mesh layout for 1-D composite fire boundary condition . . . 88

3.0-1 Control Volume is Centered about Finite-Element Node 107
3.0-2 Sub Control Volumes Within a Hexahedral Element 108

6.2-1 Fuego Procedure Class 130
6.2-2 Fuego Region Class 131

List of Tables

2.5-1 Constant parameters for $k - \epsilon$ turbulence models. 49
2.5-2 Constant parameters for $k - \omega$ turbulence models. 50
2.5-3 Constant parameters for LES turbulence models. 50
2.5-4 Constant parameters for miscellaneous turbulence models. Default values may be changed using the $k - \epsilon$ model parameters input. 51
2.9-1 Chemical mechanism data packaged into the standalone Spitfire executable 76

5.0-1 User Subroutine Argument Lists 127
5.0-2 User Subroutine Signature Type 128

NOMENCLATURE

Einstein notation is used extensively throughout this report to imply summation over repeated indices, primarily for multiple directions in integral equations. Indices are also used to denote chemical species in a gas mixture. When dealing with notation for chemical species, Einstein notation is not implied. When summation over chemical species is required, we will use a summation operator.

ENGLISH CHARACTER SYMBOLS

C_p	mixture specific heat at constant pressure
D	mass diffusion coefficient
D_i	mixture-averaged mass diffusion coefficient for species i
D_{ij}	mass diffusion coefficient between species i and j in a mixture
E	law of the wall parameter, turbulence model
f_c	mass fraction of "excess" carbon in a given species (over what may for CO_2 from the available oxygen in the species)
G	scalar radiative flux
g	magnitude of the gravity vector
g_i	component of the gravity vector in the x_i direction
h	mixture enthalpy
h	fuel pool depth
$\mathrm{j}_{i,g}$	mass diffusion flux vector for species g in the x_i direction
h_{f_g}	fuel heat of vaporization
K	number of chemical species in a mixture
k	mixture thermal conductivity
k	turbulent kinetic energy
L	length scale
L	integral scale with respect to turbulence

l	characteristic length scale of the products
$\dot{m}$	mass flow rate
M	mass
N	concentration of soot particles per volume
n	concentration of radical nuclei per volume
n_i	unit normal vector component in the x_i direction
p	pressure
P_{th}	thermodynamic pressure
q_i	heat flux vector component in the x_i direction
$\dot{R}$	soot/radical-nuclei particle production/consumption rate per volume in a cell
R	universal gas constant
$\dot{r}$	species mass production/consumption rate per unit volume in cell
r_i	position vector
s_i	unit direction vector for radiation transport
S	ratio of air mass fraction to fuel mass fraction
S_ϕ	source term for scalar variable ϕ
t	time
T	temperature
u_i	velocity component in the x_i direction
u	velocity component in the x-direction
u^τ	friction velocity, turbulence model
$u_\parallel$	velocity parallel to the wall, turbulence model
u^+	dimensionless velocity, turbulence model
v	velocity component in the y-direction
V	volume of the computation cell (control volume)
w	velocity component in the z-direction
W	mixture molecular weight
x_i	Cartesian coordinate direction
X_s	mole fraction of species s
Y_s	mass fraction of species s

y^+ dimensionless distance from wall, turbulence model

GREEK CHARACTER SYMBOLS

α absorptivity

β concentration of radical nuclei per mixture mass

ξ_c mole fraction of carbon available to produce soot

χ weighting function for the reacting portion of the fine structure

Δ scalar difference

δ_{ij} identity matrix

ϵ total normal emissivity

ϵ dissipation of turbulent kinetic energy

θ spherical direction angle for radiation transport

ϕ generic scalar quantity

Φ equivalence ratio

γ volume fraction of turbulent fine structures

γ coefficient of surface tension

η Kolmogorov dissipative turbulent length scale

κ emittance

κ thermal conductivity

κ von Karman constant, turbulence model

λ Taylor turbulent length scale

μ viscosity

ν kinematic viscosity

ρ mixture density

ρ reflectivity

σ Stefan-Boltzmann constant

σ_{ij} deviatoric plus pressure stress tensor

τ characteristic time scale

τ transmissivity

τ_{ij}	viscous stress tensor
υ	Kolmogorov dissipative turbulent velocity scale
ζ	stoichiometric coefficient

SUPERSCRIPT CHARACTER SYMBOLS

n	iteration or time step number
r	indicial notation for reaction number
$'$	fluctuating quantity with respect to time average
$''$	fluctuating quantity with respect to Favre average
$\hat{}$	normalize by stoichiometric values
$\cdot$	time rate of change of a variable
$\tilde{}$	Favre-averaged quantity
$*$	value for the turbulent fine structure in a cell
$\circ$	value for the surrounding structure in a cell
$\bar{}$	time-averaged quantity

SUBSCRIPT CHARACTER SYMBOLS

air	property associated with air
az	azimuthal angle
cell	property associated with a control volume
co	stoichiometric reaction with CO and H_2 products
co2	stoichiometric reaction with CO_2 and H_2O products, also a property associated with CO_2
D	property associated with diluents
flame	property associated with flame zone
fuel	property associated with fuel
g	indicial notation for gas-phase chemical species
h2o	property associated with H_2O
i	indicial notation for component of a vector or tensor
inc	incident quantity

j	indicial notation for component of a vector or tensor
k	indicial notation for chemical species
min	minimum limiting value
mix	mixture property
n	number of hydrogen atoms in the fuel molecule
n2	property associated with N_2
oxy	property associated with O_2
p	number of nitrogen atoms in the fuel molecule
prod	property associated with products
q	number of oxygen atoms in the fuel molecule
rad	property associated with radiation
reac	associated with a specific chemical reaction (??)
res	fine structure residence
soot	property associated with soot
stoich	stoichiometric composition
surr	property associated with the surroundings
t	turbulent quantity
w	wall value
zn	zenith angle

DIMENSIONLESS GROUPS

Pr	Prandtl number, the ratio of viscous and thermal diffusivities
Re	Reynolds number, the ratio of inertial and viscous forces
Sc	Schmidt number, the ratio of viscous and mass diffusivities

1. INTRODUCTION

The SIERRA Low Mach Module: Fuego, henceforth referred to as Fuego, is the key element of the ASC fire environment simulation project. The fire environment simulation project is directed at characterizing both open large-scale pool fires and building enclosure fires. Fuego represents the turbulent, buoyantly-driven incompressible flow, heat transfer, mass transfer, combustion, soot, and absorption coefficient model portion of the simulation software. Using MPMD coupling, Scefire and Nalu handle the participating-media thermal radiation mechanics. This project is an integral part of the SIERRA multi-mechanics software development project. Fuego depends heavily upon the core architecture developments provided by SIERRA for massively parallel computing, solution adaptivity, and mechanics coupling on unstructured grids.

1.1. ABNORMAL THERMAL ENVIRONMENTS

Fuego is part of a suite of numerical simulation tools used to address abnormal thermal environments for nuclear weapon systems [1]. From manufacture to disassembly, a weapon will see three types of environments: normal, hostile, and abnormal. Abnormal environments result from natural phenomena, such as fires, floods, tornadoes, earthquakes, lightning strikes, meteor strikes, etc., and human phenomena, generally classified as “accidents". In general, these phenomena can present thermal, mechanical, and electrical hazards to a weapon system. Nuclear weapon systems must respond to these abnormal environments in a deterministically safe manner.

Fire phenomena in the context of the abnormal thermal environment weapons response issue is part of a three stage process leading from an accident to the system response. For certain scenarios, these stages are uncoupled and may be sequential in time; in others, the stages are tightly coupled and concurrent in time.

The first stage is the initial accident or environmental scenario that is defined typically through probabilistic studies such as historic data involving accident frequencies of a given type, ignition probabilities, etc. These are used to define scenarios for deterministic simulation tools that determine the state of integrity of the weapon system and the distribution of fuel. The weapon integrity is determined by the mechanical, transient-dynamic environment it sees during an accident. For accident scenario description, Fuego is intended to handle the distribution of liquid fuels, although initial implementation will be somewhat limited due to the very broad possibilities (e.g., fuel pools, spills, sprays, porous flows) and complexity involved in two-phase flow.

The second stage is the actual buoyant, turbulent, reacting, flow that is the source of the thermal hazard for the weapon system. Fuego and MPMD-coupled Nalu/Scefire are the primary tools that describe the fire phenomenology that links an accident description to thermal radiation and convection on a weapon

system. Fire involves a very complex, coupled set of physical phenomena over a very broad range of time and length scales. The key features are the turbulent, buoyant flows involving combustion of the fuel and air, and the formation of soot which results in participating media radiation (Nalu/Scefire), and a range of convection heat transfer conditions from free to forced convection (Fuego).

The third stage is the weapon thermal response. As with the fire itself, the response of the warhead to a fire is described by very complex, coupled set of physical phenomena. Simulation will require the coupling of several, separate effects codes for a complete description. Heat from the fire is conducted into the weapon and transmitted by surface-surface radiation. Materials such as foams decompose and result in pressurization. Conduction across engineered joints is pressure dependent as is the decomposition process. Materials such as aluminum can potentially melt and relocate. Energetic materials can decompose and react. Within this environment the engineered fail-safes in the weapon electrical system must operate with high reliability to ensure nuclear safety.

Because of the number of physical phenomena involved from the accident scenario to the weapon response for abnormal thermal environments, and the very disparate time and length scales over which these phenomena occur, it is necessary to have high-performance, massively-parallel, computers to even consider addressing a problem of this scale and complexity. Further, the key to integrating this suite of tools is flexibility of coupling and a common database architecture. Thus it is intended that all the simulation requirements identified above will ride on a common software architecture (SIERRA) with broad coupling flexibilities.

The principal value of the suite of numerical simulation tools is not the description of the accident to response process, but the ability to evaluate prevention and mitigation design strategies. Preventative strategies are primarily applied via administrative controls. Examples include design and maintenance to minimize fuel levels, separation of fuels from air and ignition sources, and/or weapons separate from the combination. Mitigation strategies include suppression (either manually through fire-fighters or by automated fire suppression equipment), design of thermally activated fail-safes, and containment design. In general, multiple barriers exist between fire and health consequences to the general public for nuclear weapons.

1.2. PRODUCTION CAPABILITIES

The core use for SIERRA/Fuego is the abnormal thermal simulation described previously. However, there are additional beta capabilities present that are in various states of development. To help guide the new user towards the most well-established capabilities, we have generated a high level list of production and beta capabilities in SIERRA/Fuego. This list is not meant to be exhaustive, but should let you know when you are getting into experimental capabilities.

1.2.1. Production

- Laminar Flow (Mach < 0.3)
- Turbulent Flow (Mach < 0.3)

  - K-epsilon (RANS)
  - Low Re K-epsilon (RANS)
  - K-omega (RANS)
  - Low Re K-omega (RANS)
  - V2-F (RANS)
  - SST (RANS)
  - Low Re SST (RANS)
  - KSGS (LES)
  - Smagorinsky (LES)
  - Dynamic Smagorinsky LES model (DSMAG)
  - Subgrid scale kinetic energy one-equation turbulence model (KSGS)
- Restart
- EDC
- Particles
- Radiation (Nalu or Scefire)
- Soot
- Coupling (with Aria and Nalu or Scefire)
- Second order time integration
- Mixture Fraction Approach/Flamelets
  -

Warning: Not all combinations with other BCs and physics are equally reliable

1.2.2. Beta

Beta Capability:

- Volume of fluid (in development)
- Dynamic particle parallel rebalance (new capability)
- Particle resuspension (new capability)
- Wildfire particles
- EDC suppression
- 1D composite boundary condition

- Multiple-momentum multiple-phase simulations
- Kinetic energy conservation
- Presto-Salinas coupling for aero-acoustic modeling
- Conservative transfers
- MPMD coupling with SceFire or Nalu (in development)
- Tpetra solver interface (in development)
- Discontinuous Galerkin interfaces
- Dynamic subgrid scale kinetic energy one-equation turbulence model (DKSGS)

1.2.3. Known Issues

Warning:

- The physical applicability of the flow equations in Fuego begin to break down at Mach numbers greater than 0.3 (as a general rule)
- Lagrangian particle simulations at high volume fractions (greater than about 10%) will start to become invalid

1.3. DOCUMENT ORGANIZATION

This document serves as a users manual for using the SIERRA/Fuego code. First, a brief discussion of the governing transport equations, math models, and numerical approach are presented, Chapter 2 and Chapter 3. Next, a brief outline of user subroutine support is presented, Chapter 5. Preceding a detailed description of supported command lines, Chapters 7-8, is a description of the major abstractions involved within SIERRA, Chapter 6. Lastly, pointers to existing instructions for running on a variety of supported platforms and locations of sample tutorial simulations are presented in Chapter 20.

Detailed discussions related to the math models and numerical methods are relegated to the SIERRA/Fuego 1.0 Theoretical Manual.

The Einstein notation of repeated indices is used extensively throughout this document. The only exception is for equations involving chemical species where an explicit summation operator is used to imply summation over all chemical species.

2. MATH MODELS

Fire simulation requires the solution of variable property, high Grashof number, turbulent, low Mach number flow including the effects of species and soot transport, radiation, and buoyancy. Conservation laws include mass of the mixture, momentum, mass of the individual species, and energy. Length scales vary from molecular to convection dominated. For purposes of discussion, length scales are also categorized by the method of resolution.

The transport equations used to describe fire physics are based on two sets of approximations to the fundamental equations of fluid dynamics. Fast acoustic time scales are removed from the equations using low Mach number asymptotics, described in detail in the SIERRA/Fuego Theory Manual. Turbulent transport at high Grashof numbers is modeled using a Reynolds averaging approach, described in Section 2.3.1.

In what follows, we note that unless specifically stated otherwise all units in the equations and submodel expressions are cgs. For a more extensive treatment of units and unit conversions in Fuego, please see the "Units and Unit Conversions" section in the User's Manual. The numerical methods we use to solve the transport equations are of the finite volume class. Therefore, we generally write the transport equations in the integral form.

2.1. LAMINAR FLOW EQUATIONS

Laminar transport equations are not used for fire problems, but they are important for other classes of problems such as manufacturing. The low Mach number approximation is assumed.

2.1.1. Conservation of Mass

The mass conservation equation of a mixture of gases is given by

$$\int \frac{\partial \rho}{\partial t} \mathrm{d}V + \int \rho u_j n_j \mathrm{d}S = 0, \tag{2.1}$$

where u_j is the mass average velocity of the mixture [2].

2.1.2. Conservation of Momentum

The conservation of momentum equations are given by

$$\int \frac{\partial \rho u_i}{\partial t} \mathrm{d}V + \int \rho u_i u_j n_j \mathrm{d}S + \int P n_i \mathrm{d}S = \int \tau_{ij} n_j \mathrm{d}S + \int \left(\rho - \rho_\circ\right) g_i \mathrm{d}V, \tag{2.2}$$

where the viscous stress tensor is

$$\tau_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \frac{\partial u_k}{\partial x_k} \delta_{ij}. \tag{2.3}$$

The pressure, P, in the momentum equations deserves a special note as this quantity can represent either the dynamic, i.e., the second term in the Mach number expansion in the case of the low Mach number assumption, or the static pressure in the case of formally compressibility. In either case, as shown above the hydrostatic pressure gradient has been removed which gives rise to the far-field density, $\rho_\circ$, in the buoyancy body force. Optionally, we allow for the following sets of buoyancy models:

1) a Boussinesq buoyancy approximation where the density difference is approximated as

$$\left(\rho - \rho_\circ\right) \approx -\frac{\rho_\circ}{T_\circ} \left(T - T_\circ\right), \tag{2.4}$$

2) a standard buoyant model in which case the pressure above does include the hydrostatic pressure and the buoyancy right-hand-side source term is,

$$\rho g_i, \tag{2.5}$$

3) A Boussinesq approximation for a binary mixture in which case the right-hand-side contribution is:

$$\rho M W^{ref} \left(\frac{1}{MW_1} - \frac{1}{MW_2} \right) \left[Y_1 - Y^{ref} \right] g_i. \tag{2.6}$$

The user is referred to the Fuego user manual for exact line commands for each of these buoyancy options.

Note that zero pressure is almost always a convenient initial condition for a low Mach fluid flow. However, in cases without buoyancy, it can be anything, as the value only defines the additive constant for the pressure solve. However, one must ensure that the value matches for both initial and boundary condition specifications.

For buoyant flow, specifying zero pressure is convenient in tandem with the "differential" buoyancy option. This buoyancy term subtracts off the hydrostatic contribution such that the source term is written as

$$\rho \left(\rho - \rho_{ref}\right) \tag{2.7}$$

One can see that using this term along with a zero pressure initial condition allows one to avoid specifying initial and boundary conditions as the hydrostatic pressure, i.e., as a function of height.

2.1.3. Conservation of Energy

The conservation of energy equation in terms of enthalpy (including a source term due to radiation absorption and emission) is

$$\begin{aligned}\int \frac{\partial \rho h}{\partial t} \mathrm{d}V + \int \rho h u_j n_j \mathrm{d}S &= -\int q_j n_j \mathrm{d}S - \int \frac{\partial q_i^r}{\partial x_i} \mathrm{d}V \\ &+ \int \left(\frac{\partial P}{\partial t} + u_j \frac{\partial P}{\partial x_j} \right) \mathrm{d}V + \int \tau_{ij} \frac{\partial u_i}{\partial x_j} \mathrm{d}V, \end{aligned} \tag{2.8}$$

where the energy diffusion flux vector is given by

$$q_j = -\kappa \frac{\partial T}{\partial x_j} + \sum_{k=1}^{K} \rho h_k Y_k \hat{u}_{j,k}, \tag{2.9}$$

and $\hat{u}_{j,k}$ is the diffusion velocity of species k in the j direction. The last two terms of Equation 2.8 are only active when formal compressibility (in an acoustic sense) are important (see the Fuego user manual for the appropriate command lines to activate the low speed compressible and high speed compressible form in Fuego).

For a low Mach number flow, the time derivative of the pressure appearing above is substituted by the thermodynamic reference pressure, P_{th}, that can only be nonzero in a closed volume with energy addition or subtraction. However, the low Mach number approximation mandates that the thermodynamic pressure is always spatially uniform.

The enthalpy of the mixture, h, is a mass-average of the component enthalpies, h_k, given by

$$h = \sum_{k=1}^{K} h_k Y_k. \tag{2.10}$$

The energy diffusion flux vector includes a scaled gradient of temperature whereas the independent field to be solved in Equation 2.8 is enthalpy. The form of the gradient of temperature is derived by first taking the gradient of Equation 2.10 and using the chain rule,

$$\frac{\partial h}{\partial x_j} = \sum_{k=1}^{K} Y_k \frac{\partial h_k}{\partial x_j} + \sum_{k=1}^{K} h_k \frac{\partial Y_k}{\partial x_j}. \tag{2.11}$$

Given the thermodynamic definition of specific heat, the above equation is given by,

$$\frac{\partial h}{\partial x_j} = \sum_{k=1}^{K} Y_k C_{p_k} \frac{\partial T}{\partial x_j} + \sum_{k=1}^{K} h_k \frac{\partial Y_k}{\partial x_j} \tag{2.12}$$

$$= C_p \frac{\partial T}{\partial x_j} + \sum_{k=1}^{K} h_k \frac{\partial Y_k}{\partial x_j}. \tag{2.13}$$

This equation is rearranged,

$$\frac{\partial T}{\partial x_j} = \frac{1}{C_p}\left(\frac{\partial h}{\partial x_j} - \sum_{k=1}^{K} h_k \frac{\partial Y_k}{\partial x_j}\right), \tag{2.14}$$

and substituted into the energy diffusion flux vector to obtain,

$$q_j = -\frac{\kappa}{C_p}\left(\frac{\partial h}{\partial x_j} - \sum_{k=1}^{K} h_k \frac{\partial Y_k}{\partial x_j}\right) + \sum_{k=1}^{K} \rho h_k Y_k \hat{u}_{j,k}. \tag{2.15}$$

Commonly, the last two terms in the above equation can be canceled when a simple diffusion model is assumed (see Section 2.1.4, Equation 2.22) in the limit where the ratio of thermal and mass diffusion is equal (unity Lewis number, or equivalently speaking the Prandtl number equals the Schmidt number, i.e.,

$$Le^{unity} = \frac{Sc}{Pr} = \frac{\alpha}{D} = 1. \tag{2.16}$$

For completeness, the thermal diffusivity, Prandtl and Schmidt number are defined by,

$$\alpha = \frac{\kappa}{\rho C_p}, \tag{2.17}$$

$$Pr = \frac{C_p \mu}{\kappa} = \frac{\mu}{\rho \alpha}. \tag{2.18}$$

and

$$Sc = \frac{\mu}{\rho D_{ab}}. \tag{2.19}$$

An alternate laminar enthalpy equation is currently supported (although expected to be deprecated - very soon) is,

$$\begin{aligned} \int \frac{\partial \rho T}{\partial t} \mathrm{d}V + \int \rho T u_j n_j \mathrm{d}S &= \int \frac{\kappa}{C_p} \frac{\partial T}{\partial x_j} n_j \mathrm{d}S + \int \frac{\kappa}{C_p^2} \frac{\partial T}{\partial x_j} \frac{\partial C_p}{\partial x_j} \mathrm{d}V \\ &- \int \frac{1}{C_p} \frac{\partial q_i^r}{\partial x_i} \mathrm{d}V + \int \frac{1}{C_p} \frac{\partial p}{\partial t} \mathrm{d}V. \end{aligned} \tag{2.20}$$

2.1.4. Conservation of Species

The mass conservation equation for species k in a mixture of K gas phase species is

$$\int \frac{\partial \rho Y_k}{\partial t} \mathrm{d}V + \int \rho Y_k u_j n_j \mathrm{d}S = -\int \rho \hat{u}_{j,k} Y_k n_j \mathrm{d}S + \int \dot{\omega}_k \mathrm{d}V, \tag{2.21}$$

where $\dot{\omega}_k$ is the mass generation rate of species k per unit volume by homogeneous chemical reactions. We allow several approximations for the diffusion velocity, $\hat{u}_{j,k}$, derived in the Fuego theory manual. The simplest form is Fickian diffusion with the same value of mass diffusivity for all species,

$$\hat{u}_{i,k} = -D\frac{1}{Y_k}\frac{\partial Y_k}{\partial x_i}. \tag{2.22}$$

This form is used for the Reynolds-averaged form of the equations for turbulent flow. A more accurate approximation uses a mixture-averaged diffusion coefficient, $\bar{D}_k$, for each species diffusion velocity,

$$\hat{u}_{i,k} = -\bar{D}_k\frac{1}{X_k}\frac{\partial X_k}{\partial x_i} = -\bar{D}_k\left(\frac{1}{Y_k}\frac{\partial Y_k}{\partial x_i} + \frac{1}{W}\frac{\partial W}{\partial x_i}\right). \tag{2.23}$$

2.1.5. Conservation of Momentum, Axisymmetric with Swirl

Axisymmetric flows, with or without swirl, are described by two-dimensional equations in cylindrical coordinates. All azimuthal derivatives are zero (i.e., $\partial/\partial\theta = 0$). The axial coordinate is x, the radial coordinate is r, and the azimuthal coordinate is θ. The radius is retained in the equations and the purpose will become more clear in the discussion of the discrete integral form. The axial velocity is u, the radial velocity is v, and the azimuthal velocity is w.

Axial-Momentum:

$$\frac{\partial \rho u r}{\partial t} + \frac{\partial}{\partial x}\left(\rho u^2 r\right) + \frac{\partial}{\partial r}\left(\rho u v r\right) + r\frac{\partial p}{\partial x} = \frac{\partial}{\partial x}\left(r\tau_{xx}\right) + \frac{\partial}{\partial r}\left(r\tau_{xr}\right) + \rho r g_x \tag{2.24}$$

Radial-Momentum:

$$\frac{\partial \rho v r}{\partial t} + \frac{\partial}{\partial x}\left(\rho u v r\right) + \frac{\partial}{\partial r}\left(\rho v^2 r\right) + r\frac{\partial p}{\partial r} - \rho w^2 = \frac{\partial}{\partial x}\left(r\tau_{rx}\right) + \frac{\partial}{\partial r}\left(r\tau_{rr}\right) - \tau_{\theta\theta} + \rho r g_r \tag{2.25}$$

Azimuthal-Momentum:

$$\frac{\partial \rho w r}{\partial t} + \frac{\partial}{\partial x}\left(\rho u w r\right) + \frac{\partial}{\partial r}\left(\rho v w r\right) + \rho v w = \frac{\partial}{\partial x}\left(r\tau_{\theta x}\right) + \frac{1}{r}\frac{\partial}{\partial r}\left(r^2\tau_{\theta r}\right) \tag{2.26}$$

The viscous stress terms for the cylindrical equations are

$$\tau_{xx} = \mu\left[2\frac{\partial u}{\partial x} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial r} + \frac{v}{r}\right)\right] \tag{2.27}$$

$$\tau_{rx} = \mu\left[\frac{\partial v}{\partial x} + \frac{\partial u}{\partial r}\right] \tag{2.28}$$

$$\tau_{rr} = \mu\left[2\frac{\partial v}{\partial r} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial r} + \frac{v}{r}\right)\right] \tag{2.29}$$

$$\tau_{\theta\theta} = \mu\left[2\frac{v}{r} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial r} + \frac{v}{r}\right)\right] \tag{2.30}$$

$$\tau_{r\theta} = \mu r\frac{\partial}{\partial r}\left(\frac{w}{r}\right) \tag{2.31}$$

$$\tau_{x\theta} = \mu\frac{\partial w}{\partial x} \tag{2.32}$$

The azimuthal equation can be simplified by relating the swirl velocity to the angular velocity, $w = r\omega$. The momentum equation, written in terms of the angular velocity, is

$$\frac{\partial \rho\omega r}{\partial t} + \frac{\partial}{\partial x}\left(\rho u\omega r\right) + \frac{\partial}{\partial r}\left(\rho v\omega r\right) + 2\rho v\omega = \frac{\partial}{\partial x}\left(r\mu\frac{\partial \omega}{\partial x}\right) + \frac{\partial}{\partial r}\left(r\mu\frac{\partial \omega}{\partial r}\right) + 2\mu\frac{\partial \omega}{\partial r}. \quad (2.33)$$

The production term that is used in the turbulence model is

$$\Phi = 2\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial r}\right)^2 + \left(\frac{v}{r}\right)^2\right] + \left(\frac{\partial u}{\partial r} + \frac{\partial v}{\partial x}\right)^2 - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial r} + \frac{v}{r}\right)^2. \quad (2.34)$$

2.1.6. Laminar Flow Boundary Conditions

The laminar flow math models require boundary conditions for velocity, pressure, temperature and enthalpy variables, and mixture composition.

2.1.6.1. Inflow

There are three types of inflow boundary conditions. For velocity-specified inflow, Dirichlet conditions are applied to velocities in the momentum equations, temperature in the energy equation, and mass fractions in the species equations. The mass flow rate at the boundary is specified for the continuity equation. The pressure floats to a consistent value. Alternatively, a control volume balance is retained at the boundary nodes and the convection fluxes are specified.

For pressure-specified inflow, the outflow boundary condition is applied with the added condition that the flow must enter the domain normal to the mesh boundary. Transport equations are solved for the momentum, energy and species equations.

2.1.6.2. Outflow

The pressure is specified at integration points on the outflow boundary. The specified pressure is used in the surface integration procedure for approximation nodal gradients. The pressure gradients are used to construct an interpolation for the mass flow rate at the boundary. Transport equations are solved for the momentum, energy and species equations. Upwind extrapolation is used for the scalars if the flow is leaving the domain. The boundary values of velocity and specified far-field values of scalars are used if the flow is entering the domain.

2.1.6.3. Wall

It is assumed that there is no mass flow through the wall. The velocity is specified as a Dirichlet boundary condition in the momentum equations. The temperature is specified as a Dirichlet boundary condition in the energy if the wall is isothermal. We currently do not support heterogeneous chemical reactions at a surface, so there should be no boundary condition applied to the mass fractions.

2.1.6.4. Symmetry Plane

There is no mass flow rate through the symmetry plane and there is no transport of scalar variables. The normal stress (pressure and viscous) at the symmetry plane is applied in the momentum equations.

2.2. RADIATION TRANSPORT EQUATION

For applications involving PMR, both the radiative heat flux and the divergence of the radiative heat flux are needed. The radiative heat flux vector provides the radiative flux to the boundary of the heat conduction region. The flux divergence provides one of the principal volumetric heat sources in the turbulent combustion region for fire applications.

2.2.1. Boltzmann Transport Equation

The spatial variation of the radiative intensity corresponding to a given direction and at a given wavelength within a radiatively participating material, $I(s)$, is governed by the Boltzmann transport equation. In general, the Boltzmann equation represents a balance between absorption, emission, out-scattering, and in-scattering of radiation at a point. For combustion applications, however, the steady form of the Boltzmann equation is appropriate since the transient term only becomes important on nanosecond time scales which is orders of magnitude shorter than the fastest chemical reaction [3].

Experimental data shows that the radiative properties for heavily sooting, fuel-rich hydrocarbon diffusion flames (10^{-4}% to 10^{-6}% soot by volume) are dominated by the soot phase and to a lesser extent by the gas phase (Modest [4], pg. 425). Since soot emits and absorbs radiation in a relatively constant spectrum, it is common to ignore wavelength effects when modeling radiative transport in these environments. Additionally, scattering from soot particles commonly generated by hydrocarbon flames is several orders of magnitude smaller that the absorption effect and may be neglected [3]. With these assumptions in mind, the appropriate form of the Boltzmann radiative transport equation for heavily sooting hydrocarbon diffusion flames is

$$s_i \frac{\partial}{\partial x_i} I(s) + \mu_a I(s) = \frac{\mu_a \sigma T^4}{\pi}, \tag{2.35}$$

where μ_a is the absorption coefficient, $I(s)$ is the intensity along the direction s_i, and T is the temperature.

The flux divergence (on the right hand side of Equation 2.8) may be written as a difference between the radiative emission and mean incident radiation at a point,

$$\frac{\partial q_i^r}{\partial x_i} = \mu_a \left[4\sigma T^4 - G \right], \tag{2.36}$$

where G is the scalar flux. The quantity, $G/4\pi$, is often referred to as the mean incident intensity [5].

The scalar flux and radiative flux vector represent angular moments of the directional radiative intensity at a point [4],

$$G = \int_0^{2\pi} \int_0^{\pi} I\left(s\right) \sin\theta_{zn} d\theta_{zn} d\theta_{az}, \tag{2.37}$$

$$q_i^r = \int_0^{2\pi} \int_0^{\pi} I\left(s\right) s_i \sin\theta_{zn} d\theta_{zn} d\theta_{az}, \tag{2.38}$$

where θ_{zn} and θ_{az} are the zenith and azimuthal angles respectively as shown in Figure 2.2-1.

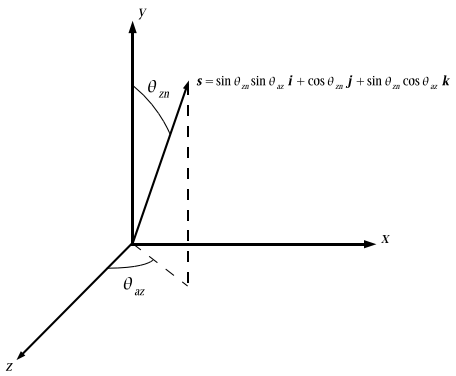


Figure 2.2-1.. Ordinate Direction Definition,
$\mathbf{s} = \sin\theta_{zn} \sin\theta_{az} \mathbf{i} + \cos\theta_{zn} \mathbf{j} + \sin\theta_{zn} \cos\theta_{az} \mathbf{k}$

2.2.2. Radiation Intensity Boundary Condition

The radiation intensity must be defined at all portions of the boundary along which $s_i n_i < 0$, where n_i is the outward directed unit normal vector at the surface. The intensity is applied as a Dirichlet condition which must be determined from the surface properties and temperature. The diffuse surface assumption provides reasonable accuracy for many engineering combustion applications. The intensity leaving a diffuse surface in all directions is given by

$$I\left(s\right) = \frac{1}{\pi} \left[\tau\sigma T_\infty^4 + \epsilon\sigma T_w^4 + \left(1 - \epsilon - \tau\right) q_j^{r,inc} n_j\right], \tag{2.39}$$

where ϵ is the total normal emissivity of the surface, τ is the transmissivity of the surface, T_w is the temperature of the boundary, T_∞ is the environmental temperature and $q_j^{r,inc}$ is the incident radiation, or irradiation for direction j. Recall that the relationship given by Kirchhoff's Law that relates emissivity, transmissivity and reflectivity, ρ, is

$$\rho + \tau + \epsilon = 1. \tag{2.40}$$

where it is implied that $\alpha = \epsilon$.

2.3. TURBULENCE MODELING OVERVIEW

Turbulent reacting flows involve a very large range of length and time scales, requiring massive computational resources to directly resolve all of the physical processes for even the most simple problem. To be able to solve complex problems of interest in a reasonable amount of time, modeling approximations must be made. A filtered form of the time-dependent Navier-Stokes, energy, and species mass conservation equations presented in Section 2.1 are used, and closure models are applied to the new terms that arise due to the filtering operation. Temporal filtering is used in the Reynolds-Averaged Navier-Stokes (RANS) method, and spatial filtering is used in the Large Eddy Simulation (LES) method. The form of the models are dependent on the type of filtering performed, and will be discussed for both the RANS and LES approaches in the following sections.

The length scales between the smallest control volume dimension and the largest mesh dimension are defined as being "resolved", and the transport equations are used to solve the physics in this range. The effects of the resolved turbulent scales may be modeled for RANS closures or they may be directly solved for LES closures. Turbulence length scales can extend down many orders of magnitude beyond the smallest finite volume dimension to the Kolmogorov scales, and these subgrid scales must be modeled in either closure approach.

The output of the closure models is expressed as a source term in the conservation equations for the mean flow and as effective properties in the radiative transport equation. Hence, the output of the closure models can be interpreted as being cell-averaged values for the control volume for the appropriate time scale. For the RANS formulation used here, the time scale is long relative to the turbulence time scales (i.e., long time average). For LES, the time scale is the local advection time. For the current suite of models, the momentum closure model is of the lumped-parameter type; that is, it assumes homogeneity of the subgrid turbulence. The remaining closures, species and energy, are of the zone-model type; that is, they assume heterogeneity of the species and energy subgrid. Two zones (one combusting, one not) are used in the current zone models.

For length scales above the length scale of the mesh, the physics is modified via boundary and initial conditions. Momentum boundary conditions include specified velocity (wind, and mass sources), or constant pressure (inflow/outflow). Species boundary conditions include a mass source for the fuel (pool model). Thermal boundary conditions include flux and temperature conditions. The following sections provide details of the math models for conservation laws and fire physics models used in SIERRA/Fuego.

2.3.1. RANS Temporal Filtering

In many typical engineering applications, only time averages of physical quantities are of interest. Often, details of the turbulent fluctuations are of little concern. RANS formulations address this need by solving a temporally-filtered form of the transport equations, directly yielding the time-averaged variables of interest. For this reason, RANS approaches represent a relatively low-cost solution method at the expense of additional modeling complexity.

An independent variable ϕ can be temporally filtered to obtain its mean $\bar{\phi}$ with the mathematical form (Tennekes and Lumley [6])

$$\overline{\phi(\boldsymbol{x})} = \lim_{\tau\to\infty} \frac{1}{\tau} \int_{t_o}^{t_o+\tau} \phi(\boldsymbol{x}, t)\, \mathrm{d}t. \tag{2.41}$$

The original variable can be represented as the sum of its mean and fluctuating component, $\phi = \bar{\phi} + \phi'$, with the properties that $\bar{\bar{\phi}} = \bar{\phi}$ and $\overline{\phi'} = 0$. This is called the Reynolds decomposition of a variable.

In combustion problems, the overall exothermic process can result in large localized temperature increases and a correspondingly large density decrease in open systems where the molecular weight change from reactants to products is small. Allowing for turbulent fluctuations of density, the above temporal averaging procedure gives rise to additional terms involving time averages of products of density and other variable (e.g., velocity) fluctuations. An alternative approach to applying the Reynolds decomposition strictly to all independent variables is to consider a mass-weighted decomposition known as Favre averaging (Libby and Williams [7], p. 15; Kuo [8], p. 419). This simplifies all of the transport equations and eases modeling. A Favre-averaged variable $\tilde{\phi}$ is defined in terms of Reynolds averages as

$$\tilde{\phi} \equiv \frac{\overline{\rho\phi}}{\bar{\rho}}. \tag{2.42}$$

A variable can then be decomposed into its Favre-mean and fluctuating component as

$$\phi = \tilde{\phi} + \phi'', \tag{2.43}$$

where $\overline{\rho\phi''} = 0$. Note that $\overline{\phi''} \neq 0$. The relation between time averaged and Favre-averaged quantities is

$$\tilde{\phi} = \bar{\phi}\left(1 + \frac{\overline{\rho'\phi'}}{\bar{\rho}\bar{\phi}}\right). \tag{2.44}$$

Favre averaging is used for all turbulent transport equations solved in SIERRA/Fuego.

For the RANS formulation used here, the laminar conservation equations of Section 2.1 are first temporally filtered, revealing additional terms that can be simplified by substituting the Favre decomposition, resulting in the Favre-filtered equations that will be presented in Section 2.4. This procedure results in new terms in the equations that consist of time averages of products of fluctuating quantities, called Reynolds stresses. These moments must be modeled to close the system of equations.

The length of the time filter is typically much larger than the time scales of a turbulent flow, meaning that all time scales from the largest turbulence scale down to the minimum Kolmogorov scale are represented by these Reynolds stresses. In a strict sense, there can be no time dependence of a mean (time-averaged) quantity. However, if there are variations in mean quantities that occur on time intervals long compared to the averaging interval, then the transient terms for the mean quantities may be justified and required. For this reason, unsteady RANS simulations are possible with the present formulation. The available RANS turbulence closure models are discussed in Section 2.5.

2.3.2. LES Spatial Filtering

Unlike the RANS approach which models most or all of the turbulent fluctuations, LES directly solves for all resolved turbulent length scales and only models the smallest scales below the grid size. In this way, a majority of the problem-dependent, energy-containing turbulent structure is directly solved in a model-free fashion. The subgrid scales are closer to being isotropic than the resolved scales, and they generally act to dissipate turbulent kinetic energy cascaded down from the larger scales in momentum-driven turbulent flows. Modeling of these small scales is generally more straightforward than RANS approaches, and overall solutions are usually more tolerant to LES modeling errors because the subgrid scales comprise such a small portion of the overall turbulent structure. While LES is generally accepted to be much more accurate than RANS approaches for complex turbulent flows, it is also significantly more expensive than equivalent RANS simulations due to the finer grid resolution required. Additionally, since LES results in a full unsteady solution, the simulation must be run for a long time to gather any desired time-averaged statistics. The trade-off between accuracy and cost must be weighed before choosing one method over the other.

The separation of turbulent length scales required for LES is obtained by using a spatial filter rather than the RANS temporal filter. This filter has the mathematical form

$$\overline{\phi(\boldsymbol{x},t)} \equiv \int_{-\infty}^{+\infty} \phi(\boldsymbol{x}',t) G(\boldsymbol{x}'-\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}', \tag{2.45}$$

which is a convolution integral over physical space $\boldsymbol{x}$ with the spatially-varying filter function G. The filter function has the normalization property $\int_{-\infty}^{+\infty} G(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 1$, and it has a characteristic length scale Δ so that it filters out turbulent length scales smaller than this size. In the present formulation, a simple "box filter" is used for the filter function,

$$G(\boldsymbol{x}'-\boldsymbol{x}) = \begin{cases} 1/V & : \quad (\boldsymbol{x}'-\boldsymbol{x}) \in \mathcal{V} \\ 0 & : \quad \text{otherwise} \end{cases}, \tag{2.46}$$

where V is the volume of control volume $\mathcal{V}$ whose central node is located at $\boldsymbol{x}$. This is essentially an unweighted average over the control volume. The length scale of this filter is approximated by $\Delta = V^{\frac{1}{3}}$. This is typically called the grid filter, as it filters out scales smaller than the computational grid size.

Similar to the RANS temporal filter, a variable can be represented in terms of its filtered and subgrid fluctuating components as

$$\phi = \bar{\phi} + \phi'. \tag{2.47}$$

For most forms of the filter function $G(\boldsymbol{x})$, repeated applications of the grid filter to a variable do not yield the same result. In other words, $\bar{\bar{\phi}} \neq \bar{\phi}$ and therefore $\overline{\phi'} \neq 0$, unlike with the RANS temporal averages.

As with the RANS formulation, modeling is much simplified in the presence of large density variations if a Favre-filtered approach is used. A Favre-filtered variable $\tilde{\phi}$ is defined as

$$\tilde{\phi} \equiv \frac{\overline{\rho\phi}}{\bar{\rho}} \tag{2.48}$$

and a variable can be decomposed in terms of its Favre-filtered and subgrid fluctuating component as

$$\phi = \tilde{\phi} + \phi''. \tag{2.49}$$

Again, note that the useful identities for the Favre-filtered RANS variables do not apply, so that $\overline{\tilde{\phi}} \neq \tilde{\phi}$ and $\overline{\phi''} \neq 0$. The Favre-filtered approach is used for all LES models in SIERRA/Fuego.

2.4. TURBULENT FLOW EQUATIONS, FAVRE-AVERAGED

The Favre-averaged turbulent transport equations are derived from the laminar equations of Section 2.1 by passing the equations through either the RANS temporal filter of Equation 2.41 or the LES spatial filter of Equation 2.45. The mathematical form of the equations are essentially identical between the two filtering methods, so only a single set of equations will be presented. Care should be taken to interpret the filters as either temporal or spatial, depending on the closure models selected. While it is the Favre-averaged form of the equations that are solved, a comparison of the simple Reynolds-averaged and the Favre-averaged form is given in the Fuego theory manual.

The approach most commonly used in turbulence modeling is called the Boussinesq eddy viscosity approximation, which relates the turbulent stress tensor to the filtered strain rate tensor through a modeled turbulent eddy viscosity. This general modeling approach has shown remarkable success for a broad range of problems (Wilcox [9]), and is the approach used in SIERRA/Fuego. A similar approach is used for scalar transport, where the scalar flux vector is related to scalar gradients through a modeled diffusion coefficient.

The following subsections describe the turbulent transport equations expressed in terms of a turbulent eddy viscosity or turbulent diffusion coefficient through the Boussinesq approximation. The treatment of these coefficients is dependent upon which of the many closure models are selected, and will be described in Section 2.5.

2.4.1. Conservation of Mass

The integral form of the Favre-filtered continuity equation used for turbulent transport is

$$\int \frac{\partial \bar{\rho}}{\partial t} \mathrm{d}V + \int \bar{\rho} \tilde{u}_j n_j \mathrm{d}S = 0. \tag{2.50}$$

This equation is in closed form, and no additional modeling is required.

2.4.2. Conservation of Momentum

The integral form of the Favre-filtered momentum equations used for turbulent transport are

$$\int \frac{\partial \bar{\rho}\tilde{u}_i}{\partial t}\mathrm{d}V + \int \bar{\rho}\tilde{u}_i\tilde{u}_j n_j \mathrm{d}S + \int \bar{p} n_i \mathrm{d}S = \int \bar{\tau}_{ij} n_j \mathrm{d}S + \int \tau_{u_i u_j} n_j \mathrm{d}S + \int (\bar{\rho} - \rho_\circ)\, g_i \mathrm{d}V, \quad (2.51)$$

where the turbulent stress $\tau_{u_i u_j}$ is defined as

$$\tau_{u_i u_j} \equiv -\bar{\rho}(\widetilde{u_i u_j} - \tilde{u}_i\tilde{u}_j). \quad (2.52)$$

2.4.2.1. RANS Modeling

For RANS simulations, $\tau_{u_i u_j}$ represents the Reynolds stress tensor and can be reduced to the form $\tau_{u_i u_j} = -\overline{\rho u_i'' u_j''}$ by substitution of the Favre decomposition $u_i \equiv \tilde{u}_i + u_i''$ of each variable and simplifying. The deviatoric (trace-free) part of the stress tensor is defined as

$$\begin{aligned} \tau^D_{u_i u_j} &\equiv \tau_{u_i u_j} - \frac{1}{3}\tau_{u_k u_k}\delta_{ij} \\ &= \tau_{u_i u_j} + \frac{2}{3}\bar{\rho}\tilde{k}\delta_{ij} \end{aligned} \quad (2.53)$$

where the turbulent kinetic energy is defined as $\tilde{k} \equiv \frac{1}{2}\widetilde{u_k'' u_k''}$. The deviatoric part of the Reynolds stress tensor is modeled by the Boussinesq approximation which relates the Reynolds stresses to the filtered strain rate tensor through a modeled turbulent viscosity μ_t, resulting in

$$\begin{aligned} \tau^D_{u_i u_j} &= \mu_t\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) - \frac{2}{3}\mu_t\frac{\partial \tilde{u}_k}{\partial x_k}\delta_{ij} \\ &= 2\mu_t\left(\tilde{S}_{ij} - \frac{1}{3}\tilde{S}_{kk}\delta_{ij}\right), \end{aligned} \quad (2.54)$$

where the filtered strain rate tensor is defined by

$$\tilde{S}_{ij} \equiv \frac{1}{2}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right). \quad (2.55)$$

Substituting this into Equation 2.53 yields the modeled form of the full Reynolds stress tensor (Kuo [8], p. 445)

$$\tau_{u_i u_j} = 2\mu_t\left(\tilde{S}_{ij} - \frac{1}{3}\tilde{S}_{kk}\delta_{ij}\right) - \frac{2}{3}\bar{\rho}\tilde{k}\delta_{ij}. \quad (2.56)$$

The Favre-filtered momentum equations then become

$$\begin{aligned} &\int \frac{\partial \bar{\rho}\tilde{u}_i}{\partial t}\mathrm{d}V + \int \bar{\rho}\tilde{u}_i\tilde{u}_j n_j \mathrm{d}S + \int \left(\bar{p} + \frac{2}{3}\bar{\rho}\tilde{k}\right) n_i \mathrm{d}S = \\ &\int 2(\mu + \mu_t)\left(\tilde{S}_{ij} - \frac{1}{3}\tilde{S}_{kk}\delta_{ij}\right) n_j \mathrm{d}S + \int (\bar{\rho} - \rho_\circ)\, g_i \mathrm{d}V, \end{aligned} \quad (2.57)$$

where RANS closure models for the turbulent viscosity μ_t are presented in Section 2.5.

2.4.2.2. LES Modeling

For LES, $\tau_{u_i u_j}$ in Equation 2.51 represents the subgrid stress tensor. The deviatoric part of the subgrid stress tensor is defined as

$$\begin{aligned} \tau^D_{u_i u_j} &\equiv \tau_{u_i u_j} - \frac{1}{3}\tau_{u_k u_k}\delta_{ij} \\ &= \tau_{u_i u_j} + \frac{2}{3}\bar{\rho}q^2\delta_{ij}, \end{aligned} \tag{2.58}$$

where the subgrid turbulent kinetic energy is defined as $q^2 \equiv \frac{1}{2}(\widetilde{u_k u_k} - \tilde{u_k}\tilde{u_k})$. The deviatoric part of the subgrid stress tensor is then modeled similar to RANS closures as (Moin, et al. [10])

$$\tau^D_{u_i u_j} = 2\mu_t\left(\tilde{S}_{ij} - \frac{1}{3}\tilde{S}_{kk}\delta_{ij}\right). \tag{2.59}$$

Substituting this into Equation 2.58 yields the modeled form of the full subgrid stress tensor

$$\tau_{u_i u_j} = 2\mu_t\left(\tilde{S}_{ij} - \frac{1}{3}\tilde{S}_{kk}\delta_{ij}\right) - \frac{2}{3}\bar{\rho}q^2\delta_{ij}. \tag{2.60}$$

For low Mach-number flows, a vast majority of the turbulent kinetic energy is contained at resolved scales (Erlebacher, et al. [11]). For this reason, the subgrid turbulent kinetic energy q^2 will not be directly treated and will instead be included in the pressure as an additional normal stress. The Favre-filtered momentum equations then become

$$\begin{aligned} &\int \frac{\partial\bar{\rho}\tilde{u}_i}{\partial t}\mathrm{d}V + \int \bar{\rho}\tilde{u}_i\tilde{u}_j n_j \mathrm{d}S + \int\left(\bar{p} + \frac{2}{3}\bar{\rho}q^2\right)n_i\mathrm{d}S = \\ &\quad \int 2(\mu+\mu_t)\left(\tilde{S}_{ij} - \frac{1}{3}\tilde{S}_{kk}\delta_{ij}\right)n_j\mathrm{d}S + \int(\bar{\rho}-\rho_\circ)\,g_i\mathrm{d}V, \end{aligned} \tag{2.61}$$

where LES closure models for the subgrid turbulent eddy viscosity μ_t are presented in Section 2.5.

2.4.3. Conservation of Energy

The integral form of the Favre-filtered energy equation used for turbulent transport is

$$\begin{aligned} \int \frac{\partial\bar{\rho}\tilde{h}}{\partial t}\mathrm{d}V + \int \bar{\rho}\tilde{h}\tilde{u}_j n_j \mathrm{d}S &= -\int \bar{q}_j n_j \mathrm{d}S - \int \tau_{hu_j} n_j \mathrm{d}S - \int \frac{\partial\bar{q}^r_i}{\partial x_i}\mathrm{d}V \\ &\quad + \int\left(\frac{\partial P}{\partial t} + \tilde{u}_j\frac{\partial P}{\partial x_j}\right)\mathrm{d}V + \int \overline{\tau_{ij}\frac{\partial u_i}{\partial x_j}}\mathrm{d}V. \end{aligned} \tag{2.62}$$

The simple Fickian diffusion velocity approximation, Equation 2.22, is assumed, so that the mean diffusive heat flux vector $\bar{q}_j$ is

$$\bar{q}_j = -\overline{\left[\frac{\mu}{\mathrm{Pr}}\frac{\partial h}{\partial x_j} - \frac{\mu}{\mathrm{Pr}}\sum_{k=1}^{K} h_k\frac{\partial Y_k}{\partial x_j}\right]} - \overline{\frac{\mu}{\mathrm{Sc}}\sum_{k=1}^{K} h_k\frac{\partial Y_k}{\partial x_j}}. \tag{2.63}$$

If Sc = Pr, i.e., unity Lewis number (Le = 1), then the diffusive heat flux vector simplifies to $\bar{q}_j = -\frac{\mu}{\mathrm{Pr}}\frac{\partial \tilde{h}}{\partial x_j}$. The viscous dissipation term is closed by

$$\begin{aligned}\overline{\tau_{ij}\frac{\partial u_i}{\partial x_j}} &= \left((\mu+\mu_t)\left(\frac{\partial \tilde{u}_i}{\partial x_j}+\frac{\partial \tilde{u}_j}{\partial x_i}\right)-\frac{2}{3}\left(\bar{\rho}\tilde{k}+\mu_t\frac{\partial \tilde{u}_k}{\partial x_k}\right)\delta_{ij}\right)\frac{\partial \tilde{u}_i}{\partial x_j} \\ &= \left[2\mu\tilde{S}_{ij}+2\mu_t\left(\tilde{S}_{ij}-\frac{1}{3}\tilde{S}_{kk}\delta_{ij}\right)-\frac{2}{3}\bar{\rho}\tilde{k}\delta_{ij}\right]\frac{\partial \tilde{u}_i}{\partial x_j}. \end{aligned} \tag{2.64}$$

The turbulent diffusive flux vector τ_{hu_j} in Equation 2.62 is defined as

$$\tau_{hu_j} \equiv \bar{\rho}\left(\widetilde{hu_j}-\tilde{h}\tilde{u}_j\right). \tag{2.65}$$

For RANS simulations, τ_{hu_j} represents the turbulent energy diffusive flux vector and is simplified to the form $\tau_{hu_j} = \overline{\rho h'' u_j''}$ by substitution of the Favre decomposition of each variable. It is then modeled by

$$\tau_{hu_j} = \overline{\rho h'' u_j''} = -\frac{\mu_t}{\mathrm{Pr}_t}\frac{\partial \tilde{h}}{\partial x_j}, \tag{2.66}$$

where Pr_t is the turbulent Prandtl number and μ_t is the modeled turbulent eddy viscosity from momentum closure. For LES, τ_{hu_j} represents the subgrid turbulent energy diffusive flux vector, and is modeled in the same way as

$$\tau_{hu_j} = -\frac{\mu_t}{\mathrm{Pr}_t}\frac{\partial \tilde{h}}{\partial x_j}, \tag{2.67}$$

where Pr_t is the subgrid turbulent Prandtl number and μ_t is the modeled subgrid turbulent eddy viscosity from momentum closure.

The resulting filtered and modeled turbulent energy equation for both RANS and LES is given in Libby and Williams [7], p. 25, as

$$\begin{aligned}\int\frac{\partial \bar{\rho}\tilde{h}}{\partial t}\mathrm{d}V+\int\bar{\rho}\tilde{h}\tilde{u}_j n_j\mathrm{d}S &= \int\left(\frac{\mu}{\mathrm{Pr}}+\frac{\mu_t}{\mathrm{Pr}_t}\right)\frac{\partial \tilde{h}}{\partial x_j}n_j\mathrm{d}S-\int\frac{\partial \bar{q}_i^r}{\partial x_i}\mathrm{d}V \\ &+ \int\left(\frac{\partial P}{\partial t}+\tilde{u}_j\frac{\partial P}{\partial x_j}\right)\mathrm{d}V+\int\overline{\tau_{ij}\frac{\partial u_j}{\partial x_j}}\mathrm{d}V. \end{aligned} \tag{2.68}$$

This equation is also given in Gran et al. [12] (without the transient and radiation source terms and the additional term for laminar transport). The turbulent Prandtl number must have the same value as the turbulent Schmidt number for species transport to maintain unity Lewis number.

2.4.4. Conservation of Species

The integral form of the Favre-filtered species equation used for turbulent transport is

$$\int\frac{\partial \bar{\rho}\tilde{Y}_k}{\partial t}\mathrm{d}V+\int\bar{\rho}\tilde{Y}_k\tilde{u}_j n_j\mathrm{d}S = -\int\tau_{Y_k u_j}n_j\mathrm{d}S-\int\overline{\rho Y_k\hat{u}_{j,k}}n_j\mathrm{d}S+\int\overline{\dot{\omega}_k}\mathrm{d}V, \tag{2.69}$$

where the form of diffusion velocities (see Equation 2.22) assumes the Fickian approximation with a constant value of diffusion velocity for consistency with the turbulent form of the energy equation, Equation 2.62.

The turbulent diffusive flux vector $\tau_{Y_k u_j}$ is defined as

$$\tau_{Y_k u_j} \equiv \bar{\rho}\left(\widetilde{Y_k u_j} - \tilde{Y}_k \tilde{u}_j\right). \tag{2.70}$$

For RANS simulations, $\tau_{Y_k u_j}$ represents the turbulent species diffusive flux vector and is simplified to the form $\tau_{Y_k u_j} = \overline{\rho Y_k'' u_j''}$ by substitution of the Favre decomposition of each variable. It is then modeled as

$$\tau_{Y_k u_j} = \overline{\rho Y_k'' u_i''} = -\frac{\mu_t}{\mathrm{Sc}_t}\frac{\partial \tilde{Y}_k}{\partial x_i}, \tag{2.71}$$

where Sc_t is the turbulent Schmidt number for all species and μ_t is the modeled turbulent eddy viscosity from momentum closure. For LES, $\tau_{Y_k u_j}$ represents the subgrid turbulent species diffusive flux vector, and is modeled identically as

$$\tau_{Y_k u_j} = -\frac{\mu_t}{\mathrm{Sc}_t}\frac{\partial \tilde{Y}_k}{\partial x_i}, \tag{2.72}$$

where Sc_t is the subgrid turbulent Schmidt number for all species and μ_t is the subgrid modeled turbulent eddy viscosity from momentum closure.

The Favre-filtered and modeled turbulent species transport equation for both RANS and LES then becomes (Gran et al. [12])

$$\int \frac{\partial \bar{\rho}\tilde{Y}_k}{\partial t}\mathrm{d}V + \int \bar{\rho}\tilde{Y}_k\tilde{u}_j n_j \mathrm{d}S = \int \left(\frac{\mu}{\mathrm{Sc}} + \frac{\mu_t}{\mathrm{Sc}_t}\right)\frac{\partial \tilde{Y}_k}{\partial x_j} n_j \mathrm{d}S + \int \overline{\dot{\omega}}_k \mathrm{d}V. \tag{2.73}$$

If transporting both energy and species equations, the laminar Prandtl number must be equal to the laminar Schmidt number and the turbulent Prandtl number must be equal to the turbulent Schmidt number to maintain unity Lewis number. Although there is a species conservation equation for each species in a mixture of K species, only $K-1$ species equations need to be solved since the mass fractions sum to unity and

$$\tilde{Y}_k = 1 - \sum_{j\neq k}^{K} \tilde{Y}_j. \tag{2.74}$$

2.4.5. Radiation Transport

The Favre-averaged energy equation, Equation 2.69, requires the time-averaged radiative flux divergence. From Equation 2.36, the time-averaged radiative flux divergence is given by

$$\overline{\frac{\partial q_i^r}{\partial x_i}} = 4\sigma\overline{\mu_a T^4} - \overline{\mu_a G}. \tag{2.75}$$

For optically thin turbulent eddies, which is the case for many combustion applications, fluctuations in the absorption coefficient and the scalar flux are weakly correlated [3] so Equation 2.75 may be simplified to

$$\overline{\frac{\partial q_i^r}{\partial x_i}} = 4\sigma\overline{\mu_a T^4} - \bar{\mu}_a \bar{G}. \tag{2.76}$$

The time averaged scalar flux is obtained from the time averaged Boltzmann radiative transport equation

$$s_i \frac{\partial}{\partial x_i}\overline{I\left(s\right)} + \bar{\mu}_a \overline{I\left(s\right)} = \frac{\overline{\mu_a \sigma T^4}}{\pi}, \tag{2.77}$$

where the correlation between the turbulent fluctuations in the absorption coefficient and the intensity is assumed small to simplify the absorption term.

Both Equation 2.76 and Equation 2.77 include the time averaged emission term, $\overline{\alpha T^4}$, which may significantly increase the radiative emission from a turbulent flame above what would be estimated from the mean temperature and absorption coefficient values. The details of the closure used for this term are discussed in the turbulent combustion model section.

2.5. TURBULENCE CLOSURE MODELS

The Favre-filtered turbulent flow equations of the previous section have been modeled in terms of μ_t, the turbulent eddy viscosity for RANS simulations and the subgrid turbulent eddy viscosity for LES. Evaluation of this eddy viscosity is dependent upon the closure model selected. All models supported by SIERRA/Fuego are described below.

2.5.1. Standard k-ϵ RANS Model

The standard k-ϵ closure model is a two-equation type of model, where transport equations for the turbulent kinetic energy and the turbulent dissipation rate are solved to obtain length-scale and time-scale estimates for the local turbulence field, to be used for modeling the turbulent eddy viscosity μ_t. The turbulent kinetic energy, k, and the dissipation rate of turbulent kinetic energy, ϵ, are given by (Gran et al. [12])

$$\int \frac{\partial \bar{\rho} k}{\partial t}\mathrm{d}V + \int \bar{\rho} k \tilde{u}_j n_j \mathrm{d}S = \int \frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j} n_j \mathrm{d}S + \int \left(P_k - \bar{\rho}\epsilon\right)\mathrm{d}V \tag{2.78}$$

$$\int \frac{\partial \bar{\rho} \epsilon}{\partial t}\mathrm{d}V + \int \bar{\rho} \epsilon \tilde{u}_j n_j \mathrm{d}S = \int \frac{\mu_t}{\sigma_\epsilon}\frac{\partial \epsilon}{\partial x_j} n_j \mathrm{d}S + \int \frac{\epsilon}{k}\left(C_{\epsilon 1} P_k - C_{\epsilon 2}\bar{\rho}\epsilon\right)\mathrm{d}V, \tag{2.79}$$

respectively, where the turbulence production rate, P_k, is defined as

$$P_k \equiv -\overline{\rho u_i'' u_j''} \frac{\partial \tilde{u}_i}{\partial x_j}, \tag{2.80}$$

and is modeled using the same Boussinesq approximation as in Equation 2.56,

$$\begin{aligned} P_k &= \mu_t \left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right) \frac{\partial \tilde{u}_i}{\partial x_j} - \frac{2}{3} \left(\bar{\rho} k + \mu_t \frac{\partial \tilde{u}_k}{\partial x_k} \right) \frac{\partial \tilde{u}_m}{\partial x_m} \\ &= \left[2\mu_t \left(\tilde{S}_{ij} - \frac{1}{3} \tilde{S}_{kk} \delta_{ij} \right) - \frac{2}{3} \bar{\rho} \tilde{k} \delta_{ij} \right] \frac{\partial \tilde{u}_i}{\partial \tilde{x}_j}. \end{aligned} \tag{2.81}$$

The turbulent eddy viscosity is then given by the Prandtl-Kolmogorov relationship,

$$\mu_t = \mathrm{C}_\mu \bar{\rho} k \tau. \tag{2.82}$$

where $\tau = min(\frac{k}{\epsilon}, dt_f)$. The filter time, dt_f is provided by the temporally filtered Navier Stokes model (Tieszen et al. [13]). The parameters $C_{\epsilon 1}$, $C_{\epsilon 2}$, σ_k, and σ_ϵ are adjustable constants.

Frequently, although not formally justified in high Reynolds flows, the diffusion coefficient for the turbulent kinetic energy and turbulence dissipation, Equations 2.78 and 2.79, may include the molecular viscosity. This option is supported within Fuego by entering the following command line in the Fuego region block, `include molecular viscosity in k-e diffusion term`.

2.5.2. Low Reynolds Number k-ϵ RANS Model

In the case of the low Reynolds number turbulent flows, the standard k-ϵ transport equations can be modified to contain additional damping functions to improve their accuracy. The low Reynolds number model of Launder and Sharma [14] are used here, which modify the turbulent kinetic energy equation, Equation 2.78, to includes an additional right-hand-side source term

$$S_k^{lr} = -2\mu \left(\frac{\partial k}{\partial x_j} \right)^2 \tag{2.83}$$

and the dissipation rate equation to include the non-isotropic dissipation source term

$$S_\epsilon^{lr} = -2\nu\nu^T \left(\frac{\partial^2 \tilde{u}_i}{\partial x_k \partial x_j} \right)^2. \tag{2.84}$$

The constants in the dissipation rate equation are modified by damping coefficients, $C_{\epsilon_1} = f_1 C_{\epsilon_1}$ and $C_{\epsilon_2} = C_{\epsilon_2} f_2$, where f_1 is unity and $f_2 = 1 - 0.3e^{-R_t^2}$.

The eddy viscosity is then given by

$$\mu_t = \mathrm{C}_\mu \bar{\rho} f_\mu k \tau. \tag{2.85}$$

Wall functions for momentum and turbulence quantities are not used with this model.

2.5.3. RNG k-ϵ RANS Model

The RNG k-ϵ model was derived using a rigorous statistical decomposition of the velocity field called renormalization group (RNG) theory. This model has several significant benefits over the standard k-ϵ model, including improved accuracy for rapidly strained flows, swirling flows, and low Reynolds number flows, without additional modifications. Additionally, values for the model constants are derived analytically rather than being evaluated empirically. Papageorgakis and Assanis [15] describe the version of the RNG k-ϵ model as implemented here.

The same turbulent kinetic energy equation as in the standard k-ϵ model, Equation 2.78, is used for the RNG k-ϵ equation. The turbulent kinetic energy dissipation rate equation is the same as Equation 2.79, with the addition of a single source term on the right-hand-side of the equation,

$$S_\epsilon^{\mathrm{RNG}} = -\frac{C_\mu \eta^3 (1 - \eta/\eta_o)}{1 + \beta\eta^3} \frac{\epsilon^2}{k}, \tag{2.86}$$

where C_μ, β, and η_o are model constants, and

$$\eta = (2\tilde{S}_{ij}\tilde{S}_{ij})^{\frac{1}{2}} \frac{k}{\epsilon}. \tag{2.87}$$

As with the standard k-ϵ model, the turbulent eddy viscosity is then given by the Prandtl-Kolmogorov relationship,

$$\mu_t = C_\mu \bar{\rho} k \tau. \tag{2.88}$$

2.5.4. v^2-f RANS Model

Durbin [16] introduced a method for handling the wall region without using either wall functions or damping functions. In his method a fine grid is required near the wall (e.g., the first grid point is typically within one dimensionless unit of distance from the wall where the coordinate normal to the wall is nondimensionalized with the inner scale for a turbulent boundary layer, $y^+ = y u_\tau / \nu < 1$ at the first grid point, where u_τ is the friction velocity, $\sqrt{\tau_w/\bar{\rho}}$). The model employs two transport equations in addition to slightly modified k and ϵ equations to account for the nonhomogeneous region near the wall. The eddy viscosity is formulated using the component of turbulent kinetic energy normal to the wall for velocity scaling (instead of using $\sqrt{k}$ as in the standard k-ϵ model).

The turbulent kinetic energy, k, is given by Equation 2.78 while the dissipation rate of turbulent kinetic energy, ϵ, is given by

$$\int \frac{\partial \bar{\rho}\epsilon}{\partial t} \mathrm{d}V + \int \bar{\rho}\epsilon \tilde{u}_j n_j \mathrm{d}S = \int \frac{\mu_t}{\sigma_\epsilon} \frac{\partial \epsilon}{\partial x_j} n_j \mathrm{d}S + \int \frac{1}{T} \left(C'_{\epsilon 1} P_k - C_{\epsilon 2} \bar{\rho}\epsilon \right) \mathrm{d}V. \tag{2.89}$$

The time scale, T, is the usual time scale k/ϵ, away from the wall region; however, near the wall, if k/ϵ becomes smaller than the Kolmogorov time scale $\sqrt{\nu/\epsilon}$, then the latter is used for T. This is formally stated by

$$T = \min \left[T_1, \frac{\alpha}{2\sqrt{3}} \frac{k}{\overline{v^2} C_\mu \sqrt{\tilde{S}^2}} \right] \tag{2.90}$$

$$T_1 = \max\left[\frac{k}{\epsilon}, 6\sqrt{\frac{\nu}{\epsilon}}\right], \tag{2.91}$$

where

$$\tilde{S}^2 = \tilde{S}_{ij}\tilde{S}_{ij} = \frac{1}{4}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right)\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) \tag{2.92}$$

and the modified constant, C'_{ϵ_1}, is given by

$$C'_{\epsilon_1} = C_{\epsilon_1}\left(1 + 0.045\sqrt{k/\overline{v^2}}\right). \tag{2.93}$$

The model includes a transport equation for $\overline{v^2}$,

$$\frac{\partial \bar{\rho}\overline{v^2}}{\partial t} + \frac{\partial \bar{\rho}\tilde{u}_j\overline{v^2}}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \mu_t\right)\frac{\partial \overline{v^2}}{\partial x_j}\right] + \bar{\rho}kf - \frac{\bar{\rho}N\overline{v^2}}{T_1}. \tag{2.94}$$

An elliptic relaxation model equation is formulated to solve for the variable f in the above equation. The purpose of the elliptic relaxation model is to account for nonlocal effects such as wall blocking; the equation is given by

$$f - L^2\frac{\partial}{\partial x_j}\left(\frac{\partial f}{\partial x_j}\right) = C_1\frac{\left(2/3 - \overline{v^2}/k\right)}{T_1} + C_2 2\nu_t\frac{\tilde{S}^2}{k} + (N-1)\frac{\overline{v^2}/k}{T_1}. \tag{2.95}$$

Finally, the turbulent eddy viscosity is given by

$$\mu_t = C_\mu\bar{\rho}\overline{v^2}\tau. \tag{2.96}$$

2.5.5. $k-\omega$ RANS Model

The $k-\omega$ turbulence model and its variants are similar in structure to the $k-\epsilon$ models. However, instead of computing the turbulent dissipation rate directly, the $k-\omega$ model models the transport the reciprocal of a turbulent timescale referred to as the turbulent frequency. This quantity, ω, can be related to the turbulent dissipation by

$$\epsilon = \beta^* k\omega. \tag{2.97}$$

The the transport equations are given by the 2006 model, (Wilcox [**?**]),

$$\int \frac{\partial \bar{\rho}k}{\partial t}\mathrm{d}V + \int \bar{\rho}k\tilde{u}_j n_j \mathrm{d}S = \int (\mu + \sigma_k\frac{\bar{\rho}k}{\omega})\frac{\partial k}{\partial x_j}n_j \mathrm{d}V + \int \left(P_k^\omega - \beta^*\bar{\rho}k\omega\right)\mathrm{d}V, \tag{2.98}$$

$$\int \frac{\partial \bar{\rho}\omega}{\partial t}\mathrm{d}V + \int \bar{\rho}\omega\tilde{u}_j n_j \mathrm{d}S = \int (\mu + \sigma_\omega\frac{\bar{\rho}k}{\omega})\frac{\partial \omega}{\partial x_j}n_j \mathrm{d}V + \int \left(\gamma\frac{\omega}{k}P_k^\omega - \beta\bar{\rho}\omega^2 + \frac{\bar{\rho}\sigma_d}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}\right)\mathrm{d}V. \tag{2.99}$$

The user is to note the above standard for writing the effective diffusive flux coefficient. The model also has a number of adjustable parameters: $\beta_0 = 0.0708$, $\beta^* = 0.09$, $\gamma = \frac{13}{25}$, $C_{lim} = \frac{7}{8}$, $\sigma_k = 0.6$, and $\sigma_\omega = 0.5$. The constant β is given by,

$$\beta = \beta_0 f_\beta \tag{2.100}$$

where

$$f_\beta = \frac{1 + 85\chi_\omega}{1 + 100\chi_\omega} \tag{2.101}$$

The value of χ_ω is as follows:

$$\chi_\omega = |\frac{\Omega_{ij}\Omega_{jk}S_{ki}}{(\beta^*\omega)^3}| \tag{2.102}$$

The production term is the same as in $k - \epsilon$. Typically limiters are used to prevent it from exceeding the dissipation rate by too large an amount. Although the 2006 description does not speak of production limiters, other sources that use the 2006 model do, i.e.

$$P_k^\omega = \max\left(P_k, 10\bar{\rho}k\omega\right). \tag{2.103}$$

The value of 10 is expected to be a user specified quantity (see input file manual for more details). In general, this term is defaulted to a very high number.

The eddy viscosity is

$$\mu_T = \bar{\rho}\frac{k}{\hat{\omega}}. \tag{2.104}$$

where $\hat{\omega}$ is,

$$\hat{\omega} = \max(\omega, C_{lim}\sqrt{\frac{2S_{ij}S_{ij}}{\beta^*}}). \tag{2.105}$$

2.5.6. Shear Stress Transport (SST)

It has been observed that standard 1998 $k - \omega$ models display a strong sensitivity to the free stream value of ω. To remedy, this, an alternative set of transport equations have been used that are based on smoothly blending the $k - \omega$ model near a wall with $k - \epsilon$ away from the wall (see Mentor [**?**]). Because of the relationship between ω and ϵ, the transport equations for turbulent kinetic energy and dissipation can be transformed into equations involving k and ω. Aside from constants, the transport equation for k is unchanged. However, an additional cross-diffusion term is present in the ω equation. Blending is introduced by using smoothing which is a function of the distance from the wall, $F(y)$. The transport equations for the Mentor 2003 model ([**?**]) are provided by the following:

$$\int \frac{\partial \bar{\rho}k}{\partial t}\mathrm{d}V + \int \bar{\rho}k\tilde{u}_j n_j \mathrm{d}S = \int (\mu + \hat{\sigma}_k\mu_t)\frac{\partial k}{\partial x_j}n_j + \int \left(P_k^\omega - \beta^*\bar{\rho}k\omega\right)\mathrm{d}V, \tag{2.106}$$

$$\int \frac{\partial \bar{\rho}\omega}{\partial t}\mathrm{d}V + \int \bar{\rho}\omega\tilde{u}_j n_j \mathrm{d}S = \int (\mu + \hat{\sigma}_\omega\mu_t)\frac{\partial \omega}{\partial x_j}n_j + \int 2(1-F)\frac{\bar{\rho}\sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}\mathrm{d}V + \int \left(\frac{\hat{\gamma}}{\nu_t}P_k^\omega - \hat{\beta}\bar{\rho}\omega^2\right)\mathrm{d}V. \tag{2.107}$$

The model coefficients, $\hat{\sigma}_k$, $\hat{\sigma}_\omega$, $\hat{\gamma}$ and $\hat{\beta}$ must also be blended, which is represented by

$$\hat{\phi} = F\phi_1 + (1 - F)\phi_2. \tag{2.108}$$

where $\sigma_{k1} = 0.85$, $\sigma_{k2} = 1.0$, $\sigma_{\omega 1} = 0.5$, $\sigma_{\omega 2} = 0.856$, $\gamma_1 = \frac{5}{9}$, $\gamma_2 = 0.44$, $\beta_1 = 0.075$ and $\beta_2 = 0.0828$.

The blending function is given by

$$F = \tanh(arg_1^4), \tag{2.109}$$

where

$$arg_1 = \min\left(\max\left(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\mu}{\bar{\rho}y^2\omega}\right), \frac{4\bar{\rho}\sigma_{\omega 2}k}{CD_{k\omega}y^2}\right). \tag{2.110}$$

The final parameter is

$$CD_{k\omega} = \max\left(2\bar{\rho}\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-10}\right). \tag{2.111}$$

In the 2003 SST model description, the production term is expected to be limited:

$$P_k^\omega = \max\left(P_k, 10\bar{\rho}k\omega\right). \tag{2.112}$$

The value of 10 is expected to be a user specified quantity (see input file manual for more details). In general, this term is defaulted to a very high number.

An important component of the SST model is the different expression used for the eddy viscosity,

$$\mu_t = \frac{a_1\bar{\rho}k}{\max\left(a_1\omega, SF_2\right)}, \tag{2.113}$$

where F_2 is another blending function given by

$$F_2 = \tanh(arg_2^2). \tag{2.114}$$

The final parameter is

$$arg_2 = \max\left(\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\mu}{\bar{\rho}\omega y^2}\right). \tag{2.115}$$

2.5.7. Standard Smagorinsky LES Model

The standard Smagorinsky LES closure model approximates the subgrid turbulent eddy viscosity using a mixing length-type model, where the LES grid filter size Δ provides a natural length scale. The subgrid eddy viscosity is modeled simply as (Smagorinsky [17])

$$\mu_t = \rho\left(C_s\Delta\right)^2|\tilde{S}|, \tag{2.116}$$

where the strain rate tensor magnitude is defined as $|\tilde{S}| \equiv (2\tilde{S}_{ij}\tilde{S}_{ij})^{\frac{1}{2}}$. The constant coefficient C_s typically varies between 0.1 and 0.24 and should be carefully tuned to match the problem being solved (Rogallo and Moin [18]). It is assigned a value of 0.17 here.

Although this model is desirable due to its simplicity and efficiency, care should be taken in its application. It is known to predict subgrid turbulent eddy viscosity proportional to the shear rate in the flow, independent of the local turbulence intensity. Non-zero subgrid turbulent eddy viscosity is even predicted in completely laminar regions of the flow, sometimes even preventing a natural transition to turbulence. Therefore, this model should only be used when this behavior will not adversely affect results.

2.5.8. Dynamic Smagorinsky LES Model

As mentioned in the previous section, the standard Smagorinsky model requires careful tuning of the constant model coefficient for the particular problem being simulated, and it is often overly-dissipative due to its inability to adapt to the local turbulent environment. Germano et al. [19] developed an improvement over the standard Smagorinsky model, where the coefficient C_s is dynamically calculated based on the local turbulence field. A generalization of this method for variable-density flow is used here (Moin et al. [10]).

Similar to the standard Smagorinsky LES closure model, the subgrid eddy viscosity is modeled by the mixing length approximation

$$\mu_t = C_R \bar{\rho} \Delta^2 |\tilde{S}|, \tag{2.117}$$

where the strain rate tensor magnitude is defined as $|\tilde{S}| \equiv (2\tilde{S}_{ij}\tilde{S}_{ij})^{\frac{1}{2}}$. The coefficient C_R is dynamically evaluated by taking advantage of scale similarity in the inertial range of the turbulence spectrum, near the minimum resolved scales. This is done by introducing a "test filter" which is identical to the grid filter defined in Equation 2.45 except for having a larger filter size denoted by $\hat{\Delta}$. The test filter of variable ϕ is denoted by $\hat{\phi}$.

The previously-defined subgrid stress tensor can be rewritten as

$$\begin{aligned} \tau_{u_i u_j} &\equiv -(\bar{\rho}\widetilde{u_i u_j} - \bar{\rho}\tilde{u}_i\tilde{u}_j) \\ &= -\left(\overline{\rho u_i u_j} - \frac{\overline{\rho u_i}\,\overline{\rho u_j}}{\bar{\rho}}\right) \end{aligned} \tag{2.118}$$

and an analogous larger-scale "subtest" stress $T_{u_i u_j}$ can be analogously defined as

$$T_{u_i u_j} \equiv -\left(\widehat{\overline{\rho u_i u_j}} - \frac{\widehat{\overline{\rho u_i}}\,\widehat{\overline{\rho u_j}}}{\hat{\bar{\rho}}}\right), \tag{2.119}$$

where the $\hat{\bar{()}}$ notation denotes resolved quantities that have been passed through the test filter. These two stresses can be related to each other through the algebraic identity of Germano [20],

$$L_{u_i u_j} \equiv T_{u_i u_j} - \widehat{\tau_{u_i u_j}} \tag{2.120}$$

$$= -\left(\widehat{\bar{\rho}\tilde{u}_i\tilde{u}_j} - \frac{\widehat{\bar{\rho}\tilde{u}_i}\,\widehat{\bar{\rho}\tilde{u}_j}}{\hat{\bar{\rho}}}\right). \tag{2.121}$$

Note that the right-hand side of Equation 2.121 is completely computable in terms of resolved quantities.

By modeling the two stresses in Equation 2.120 and equating them to Equation 2.121, the model coefficient C_R can be dynamically evaluated. The subtest stress is modeled analogously to the subgrid stress, as

$$\tau_{u_i u_j} \approx 2C_R\bar{\rho}\Delta^2 |\tilde{S}| \left(\tilde{S}_{ij} - \frac{1}{3}\tilde{S}_{kk}\delta_{ij}\right) \tag{2.122}$$

$$T_{u_i u_j} \approx 2C_R\hat{\bar{\rho}}\hat{\Delta}^2 |\hat{\tilde{S}}| \left(\widehat{\tilde{S}_{ij}} - \frac{1}{3}\widehat{\tilde{S}_{kk}}\delta_{ij}\right), \tag{2.123}$$

where C_R is assumed to be the same at both scales. The test-filtered strain rate tensor is defined similar to $|\tilde{S}|$ as

$$|\hat{\tilde{S}}| \equiv \left(2\widehat{\tilde{S}_{ij}}\widehat{\tilde{S}_{ij}}\right)^{\frac{1}{2}}. \tag{2.124}$$

Notice that when the modeled forms of $\tau_{u_i u_j}$ and $T_{u_i u_j}$ are substituted into Equation 2.120, C_R appears inside a test filtering operation. Formally solving this system of equations for C_R requires the expensive proposition of solving an additional set of coupled integro-differential equations (Ghosal et al. [21]). Alternatively, it is common practice to remove C_R from the test filter with the assumption that it is varying slowly over distances on the order of the test filter size. This greatly simplifies calculations, although it yields a system of overdetermined equations for this single constant. The square of the error involved in this approximation is $Q = (L_{ij} - C_R M_{ij})^2$, where

$$L_{u_i u_j} = -\left(\widehat{\bar{\rho}\tilde{u}_i\tilde{u}_j} - \frac{\widehat{\bar{\rho}\tilde{u}_i}\,\widehat{\bar{\rho}\tilde{u}_j}}{\hat{\bar{\rho}}}\right) \tag{2.125}$$

$$M_{u_i u_j} = 2\hat{\bar{\rho}}\hat{\Delta}^2|\hat{\tilde{S}}|\left(\widehat{\tilde{S}_{ij}} - \frac{1}{3}\widehat{\tilde{S}_{kk}}\delta_{ij}\right) - \overbrace{2\bar{\rho}\Delta^2 |\tilde{S}| \left(\tilde{S}_{ij} - \frac{1}{3}\tilde{S}_{kk}\delta_{ij}\right)}. \tag{2.126}$$

Minimizing this error in a least-squares fashion yields an expression for the modeled Smagorinsky coefficient (Lilly [22]),

$$C_R = \frac{L_{u_i u_j} M_{u_i u_j}}{M_{u_i u_j} M_{u_i u_j}}, \tag{2.127}$$

that can be used directly in Equation 2.117 for the subgrid turbulent eddy viscosity.

Due to the above simplifications, the model constant C_R can sometimes fluctuate wildly to both large positive and negative values. These fluctuations can possibly lead to numerical instability, so they must be controlled. A common solution, and one that is taken here, is to pass the numerator and denominator of Equation 2.127 through the test filter, yielding

$$C_R = \frac{\overbrace{L_{u_i u_j} M_{u_i u_j}}}{\overbrace{M_{u_i u_j} M_{u_i u_j}}}. \tag{2.128}$$

This can be crudely justified by recognizing that C_R was already assumed to vary slowly over distances equal to the test filter size, so that this filtering operation is simply enforcing that assumption.

This form of the dynamic Smagorinsky closure model allows energy backscatter, which is an intermittent transfer of turbulent kinetic energy from small scales to larger scales rather than the typical

cascade from large to small scales. While backscatter can occur in real turbulent flows, the predicted negative eddy viscosities of the dynamic Smagorinsky model are more often attributable to model errors than to a real physical backscatter process. This can easily destabilize a simulation, so negative eddy viscosity is disallowed in the present formulation.

The only free parameter in the dynamic Smagorinsky closure model is the ratio between the test and grid filter sizes, $\alpha = \hat{\Delta}/\Delta$. Solutions are fairly insensitive to the choice of α, although values of around $\alpha = 2$ are usually considered optimal (Germano et al. [19]). This ratio is dictated by the box filter formulation used in Fuego and the mesh topology selected by the user. The test filter volume for a particular CVFEM node is defined as the volume of all surrounding finite elements that contain that node. (See Chapter 3 for more information about the CVFEM formulation.) On uniform hexahedral and uniform quadrilateral meshes, the test filter ratio will have a value of 2.0. The ratio will be around 1.59 for uniform tetrahedral meshes and around 1.73 for uniform triangular meshes, which are still reasonable values.

2.5.9. Subgrid-Scale Kinetic Energy One-Equation LES Model

The subgrid scale kinetic energy one-equation turbulence model, or $Ksgs$ model, represents a simple LES closure model. The transport equation for subgrid turbulent kinetic energy is given by

$$\int \frac{\partial \bar{\rho} k^{\text{sgs}}}{\partial t} \mathrm{d}V + \int \bar{\rho} k^{\text{sgs}} \tilde{u}_j n_j \mathrm{d}S = \int \frac{\mu_t}{\sigma_k} \frac{\partial k^{\text{sgs}}}{\partial x_j} n_j \mathrm{d}S + \int \left(P_k^{\text{sgs}} - D_k^{\text{sgs}} \right) \mathrm{d}V. \tag{2.129}$$

The production of subgrid turbulent kinetic energy, P_k^{sgs}, is modeled by Equation 2.81 while the dissipation of turbulent kinetic energy, D_k^{sgs}, is given by

$$D_k^{\text{sgs}} = C_\epsilon \bar{\rho} \frac{{k^{\text{sgs}}}^{\frac{3}{2}}}{\Delta}, \tag{2.130}$$

where the grid filter length, Δ, is given in terms of the grid cell volume by

$$\Delta = V^{\frac{1}{3}}. \tag{2.131}$$

The subgrid turbulent eddy viscosity is then provided by

$$\mu_t = C_{\mu_\epsilon} \Delta {k^{\text{sgs}}}^{\frac{1}{2}}, \tag{2.132}$$

where the values of C_ϵ and C_{μ_ϵ} are 0.845 and 0.0856, respectively.

Beta Capability:

2.5.10. Dynamic Subgrid-Scale Kinetic Energy One-Equation LES Model

Similar to the dynamic Smagorinsky model in Section 2.5.8, a dynamic version is developed for the subgrid kinetic energy model. The standard version with fixed coefficients over-predicts turbulent viscosity while the dynamic version is known to offer a better predictability. In Fuego, C_ϵ and C_{μ_ϵ} are calculated dynamically which is considered to be a standard approach for the dynamic $Ksgs$ model [**?**]. The concept of "test filter" is identical to that of the dynamic Smagorinsky model in Section 2.5.8. Subgrid-scale kinetic energy for grid-filter and test-filter levels are

$$k^{\text{sgs}} = \frac{1}{2}\left(\widetilde{u_k u_k} - \tilde{u}_k \tilde{u}_k\right), \tag{2.133}$$

$$k^{\text{test}} = \frac{1}{2}\left(\widehat{\tilde{u}_k \tilde{u}_k} - \hat{\tilde{u}}_k \hat{\tilde{u}}_k\right). \tag{2.134}$$

Exact form of the dissipation D_k^{sgs} in Equation 2.129 is

$$D_k^{\text{sgs}} = 2\bar{\mu}\left[\widetilde{S_{ij}^* D_{ij}^*} - \tilde{S}_{ij}^* \tilde{D}_{ij}^*\right] \tag{2.135}$$

where $S_{ij}^* = S_{ij} - \frac{1}{3} S_{kk}\delta_{ij}$, $D_{ij}^* = D_{ij} - \frac{1}{3} D_{kk}\delta_{ij}$, and $D_{ij} = \partial u_i / \partial x_j$. Meanwhile, k^{test} dissipates by both molecular and turbulent viscosities of the grid-filtered level since the quantity is fully resolved in the test-filter level [**?**].

$$D_k^{\text{test}} = 2(\bar{\mu} + \mu_t)\left[\widehat{\tilde{S}_{ij}^* \tilde{D}_{ij}^*} - \widehat{\tilde{S}_{ij}^*}\,\widehat{\tilde{D}_{ij}^*}\right] \tag{2.136}$$

Using scale similarity, Equation 2.130 applies to the test-filter level as

$$D_k^{\text{test}} = C_\epsilon \hat{\bar{\rho}} \frac{k^{\text{test}\,\frac{3}{2}}}{\hat{\Delta}}, \tag{2.137}$$

and therefore, C_ϵ is calculated by

$$C_\epsilon = \frac{2(\bar{\mu} + \mu_t)\left[\widehat{\tilde{S}_{ij}^* \tilde{D}_{ij}^*} - \widehat{\tilde{S}_{ij}^*}\,\widehat{\tilde{D}_{ij}^*}\right]}{\hat{\bar{\rho}} k^{\text{test}\,\frac{3}{2}} / \hat{\Delta}} \tag{2.138}$$

The other coefficient, C_{μ_ϵ}, is computed similarly to the Equation 2.127 as

$$C_{\mu_\epsilon} = \frac{L_{u_i u_j} M_{u_i u_j}}{M_{u_i u_j} M_{u_i u_j}}, \tag{2.139}$$

where $L_{u_i u_j}$ is defined identically to Equation 2.125 and $M_{u_i u_j}$ is simplified by

$$M_{u_i u_j} = 2\hat{\bar{\rho}}\hat{\Delta} k^{\text{test}\,\frac{1}{2}} \widehat{\tilde{S}_{ij}^*}. \tag{2.140}$$

Note that dynamic subgrid kinetic energy model does not require an additional filtering as in Equation 2.128.

2.5.11. Buoyancy Models for the Production Rate

There are three supported models that augment the production of turbulent kinetic energy via buoyancy contributions, buoyant vorticity generation [23], Rodi's [24], and de Ris' [**?**] buoyancy term.

The buoyant vorticity generation model has been developed and validated by Sandia National Laboratories group 9132 for use in large scale buoyant plumes. The model attempts to augment the production of turbulent kinetic energy by adding a source term, G_B to both the turbulent kinetic energy and dissipation rate equation that is related to the baroclinic torque,

$$G_B = \frac{C_{bvg}(\mu + \mu_t)||\frac{\partial \rho}{\partial x_j} X \frac{\partial P}{\partial x_j}||}{\rho^2}. \tag{2.141}$$

Please refer to the Fuego theory manual for a more detailed derivation of the model.

The buoyancy model of Rodi is given by

$$G_B = \beta \frac{\mu_t}{Pr_t} \frac{\partial T}{\partial x_j} g_j. \tag{2.142}$$

De Ris' buoyancy model offers two versions - flaming and non-flaming.

$$C_{deris}(\rho_\infty - \rho_f)\vec{g}k^{0.5} \qquad \text{flaming} \tag{2.143}$$

$$C_{deris}\Delta k^{0.5}(|\nabla \rho \times \vec{g}| - \nabla \rho \cdot \vec{g}) \qquad \text{non-flaming} \tag{2.144}$$

Default value of the user-defined coefficient C_{deris} is 0.01. Note that ambient and flame density, rather than local density, matters on the flaming version.

In each model, derivatives are evaluated at the subcontrol volume center while the property values are lumped.

The right hand side of the turbulent kinetic energy equation for all model is $rhs+ = \int G_B dV$. For the dissipation rate equation, the source term is $rhs+ = \int C_{\epsilon 3} \frac{1}{T} G_B dV$ for the buoyant vorticity generation model while it is $rhs+ = \int C'_{\epsilon 1} C_{\epsilon 4} \frac{1}{T} G_B dV$ otherwise. Recall that the inverse time scale is determined by the turbulence model of choice, i.e., $\frac{\epsilon}{k}$ for the standard $k - \epsilon$ model and provided in Equation 2.90 for the v^2-f model.

Note that the use of the buoyancy models hass not been evaluated with the v^2-f model.

2.5.12. Turbulence closure model constants

For each of the afore-mentioned turbulence closure models, there are several constant coefficients which may be modified by the user in the input deck. Tables 2.5-1, 2.5-2, 2.5-3, and 2.5-4 list these parameters, their mapping to input deck names, and default values. Each of these default values may be modified by the user by specifying the respective `Turbulence Model Parameter` line in the `Global Constants` block under the Sierra domain.

Table 2.5-1.. Constant parameters for $k-\epsilon$ turbulence models.

Turbulence Model	Symbol	User Input Name	Default Value
Standard $k-\epsilon$	C_μ	Cmu	0.09
	$C_{\epsilon 1}$	Ceps_1	1.44
	$C_{\epsilon 2}$	Ceps_2	1.92
	C_χ	Cchi	2.0
	σ_k	Sigma_K	1.0
	σ_ϵ	Sigma_E	1.3
Low Reynolds $k-\epsilon$	C_μ	Cmu	0.09
	$C_{\epsilon 1}$	Ceps_1	1.44
	$C_{\epsilon 2}$	Ceps_2	1.92
	σ_k	Sigma_K	1.0
	σ_ϵ	Sigma_E	1.3
	A_μ	Amu	3.4
RNG $k-\epsilon$	C_μ	Cmu	0.0837
	$C_{\epsilon 1}$	Ceps_1	1.42
	$C_{\epsilon 2}$	Ceps_2	1.68
	σ_k	Sigma_K	0.7194
	σ_ϵ	Sigma_E	0.7194
v^2-f	C_μ	Cmu	0.22
	$C_{\epsilon 1}$	Ceps_1	1.4
	$C_{\epsilon 2}$	Ceps_2	1.9
	σ_k	Sigma_K	1.0
	σ_ϵ	Sigma_E	1.0
	C_1	CF_1	0.4
	C_2	CF_2	0.3
	α	Alpha	0.6
	C_T	Nseg	6.0
	C_L	CL	0.23
	C_η	Ceta	70.0

2.6. WALL BOUNDARY CONDITIONS FOR TURBULENCE MODELS

2.6.1. Resolution of Boundary Layer; Momentum

The wall velocity boundary condition is the typical no-slip boundary; a specified value is expected.

Table 2.5-2.. Constant parameters for $k-\omega$ turbulence models.

Turbulence Model	Symbol	User Input Name	Default Value
$k-\omega$	β_0	`Beta_Zero`	0.0708
	β^*	`Beta_Star`	0.09
	σ_k	`Sigma_K`	$3/5$
	σ_ω	`Sigma_W`	0.5
	γ	`Gamma`	$13/25$
	C_{lim}	`Clim`	$7/8$
SST	A_1	`A_One`	0.31
	β_1	`Beta_One`	0.075
	β_2	`Beta_Two`	0.0828
	β^*	`Beta_Star`	0.09
	γ_1	`Gamma_One`	$5/9$
	γ_2	`Gamma_Two`	0.44
	σ_{k1}	`Sigma_K_One`	0.85
	σ_{k2}	`Sigma_K_Two`	1.0
	$\sigma_{\omega1}$	`Sigma_W_One`	0.5
	$\sigma_{\omega2}$	`Sigma_W_Two`	0.856

Table 2.5-3.. Constant parameters for LES turbulence models.

Turbulence Model	Symbol	User Input Name	Default Value
One-equation	C_v	`Cv`	0.5
	C_ϵ	`Ceps`	0.845
	C_{μ_ϵ}	`Cmueps`	0.0856
Standard Smagorinsky	C_v	`Cv`	0.5
	C_s	`Cs`	0.17
Dynamic Smagorinsky	C_s	`Cs`	0.17

2.6.2. Resolution of Boundary Layer; Turbulence Quantities

The resolution of the boundary layer is expected when the low Reynolds number or v^2-f model is in use.

For the v^2-f model, the wall turbulent kinetic energy and normal fluctuating stress component are each zero while the dissipation rate is given by

$$\epsilon_w = 2\nu \frac{\partial k^{1/2}{}^2}{\partial x_j} \,. \tag{2.145}$$

For the low Reynolds number, the wall turbulent kinetic energy is again zero while the dissipation rate, here considered to be the isotropic dissipation rate, is given as zero.

Table 2.5-4.. Constant parameters for miscellaneous turbulence models. Default values may be changed using the $k-\epsilon$ model parameters input.

Model	Symbol	User Input Name	Default Value
Buoyant vorticity generation	C_{BVG}	`Cbvg`	0.35
	$C_{\epsilon 3}$	`Ceps_3`	0.0
Rodi's source term	$C_{\epsilon 4}$	`C_eps4`	0.0
EDC laminar limit	$C_{\gamma,lam}$	`Cgammalam`	2.0
	$C_{\tau,lam}$	`Ctaulam`	0.02
	$C_{lam,trans}$	`Clamtrans`	40.0

2.6.3. Resolution of Boundary Layer; Enthalpy

The wall value of enthalpy is computed based on the specified temperature and either reference or local mass fractions. In the case of a heat flux boundary condition, the wall node value is computed based on the control volume balance.

2.6.4. Wall Functions for Turbulent Flow Boundary Conditions

Resolution of the near-wall turbulent boundary layer can require extensive mesh points. Adjacent to the wall exists an extremely thin viscous sublayer where these forces dominate and are relatively insensitive to free stream parameters. Following the viscous sublayer is a buffer layer, the so-called "log-layer" and, ultimately, the turbulent core. The Van Driest hypothesis of turbulent flow near solid boundaries can be used to derive the appropriate form of this log-law zone. In general, the use of wall functions eliminates the need to resolve the near wall layers by prescribing the wall shear stress and resulting force based on the law of the wall (Launder and Spalding [25]).

The primary assumptions of the law of the wall are

- local equilibrium of turbulent kinetic energy production and dissipation,
- constant shear stress within the log-law region,
- Couette flow (pure shear flow).

2.6.5. Wall Functions; Momentum

The wall shear stress enters the discretization of the momentum equations by the term

$$\int \tau_{ij} n_j dS = -F_{wi}. \tag{2.146}$$

Wall functions are used to prescribe the value of the wall shear stress rather than resolving the boundary layer within the near-wall domain. The fundamental momentum law of the wall formulation, assuming fully-developed turbulent flow near a no-slip wall, can be written as (Launder and Spalding [25])

$$u^+ = \frac{u_\parallel}{u_\tau} = \frac{1}{\kappa} \ln\left(E y^+\right), \tag{2.147}$$

where u^+ is defined by the the near-wall parallel velocity, $u_\parallel$, normalized by the wall friction velocity, u_τ. The wall friction velocity is related to the turbulent kinetic energy by

$$u_\tau = C_\mu^{1/4} k^{1/2}. \tag{2.148}$$

by assuming that the production and dissipation of turbulence is in local equilibrium. Moreover, y^+ is defined as the normalized perpendicular distance from the point in question to the wall,

$$y^+ = \frac{\rho Y_p}{\mu} \left(\frac{\tau_w}{\rho}\right)^{1/2} = \frac{\rho Y_p u_\tau}{\mu} \tag{2.149}$$

The classical law of the wall is as follows:

$$u^+ = \frac{1}{\kappa} \ln(y^+) + C \tag{2.150}$$

where κ is the von Karman constant and C is the dimensionless integration constant that varies based on authorship and surface roughness. The above expression can be re-written as

$$u^+ = \frac{1}{\kappa} \ln(y^+) + \frac{1}{\kappa} \ln(\exp(\kappa C)) \tag{2.151}$$

or

$$u^+ = \frac{1}{\kappa} \left(\ln(y^+) + \ln(\exp(\kappa C))\right) \tag{2.152}$$

$$= \frac{1}{\kappa} \ln(E y^+) \tag{2.153}$$

where E is referred to in the text as the dimensionless wall roughness parameter and is described by

$$E = \exp(\kappa C) \tag{2.154}$$

In Fuego, κ is set to the value of 0.42 while the value of E is set to 9.8 for smooth walls[1]. The viscous sublayer is assumed to extend to a value of $y^+ = 11.63$.

The wall shear stress, τ_w, can be expressed as

$$\tau_w = \rho u_\tau^2 = \rho u_\tau \frac{u_\parallel}{u^+} = \frac{\rho \kappa u_\tau}{\ln\left(E y^+\right)} u_\parallel = \lambda_w u_\parallel, \tag{2.155}$$

where λ_w is simply the grouping of the factors from the law of the wall. For values of y^+ less than 11.63, the wall shear stress is given by

$$\tau_w = \mu \frac{u_\parallel}{Y_p}. \tag{2.156}$$

[1] White [26] suggests values of $\kappa = 0.41$ and $E = 7.768$.

The force imparted by the wall, for the i_{th} component of velocity, can be written as

$$F_{w,i} = -\lambda_w A_w u_{i\|}, \tag{2.157}$$

where A_w is the total area over which the shear stress acts.

The use of a general, non-orthogonal mesh adds a slight complexity to specifying the force imparted on the fluid by the wall. As shown in Equation 2.157, the velocity component parallel to the wall must be determined. Use of the unit normal vector, n_j, provides an easy way to determine the parallel velocity component by the following standard vector projection,

$$\Pi_{ij} = \left[\delta_{ij} - n_i n_j\right]. \tag{2.158}$$

Carrying out the projection of a general velocity, which is not necessarily parallel to the wall, yields the velocity vector parallel to the wall,

$$u_{i\|} = \Pi_{ij} u_j = u_i \left(1 - {n_i}^2\right) - \sum_{j=1;j\neq j}^{n} u_j n_i n_j. \tag{2.159}$$

Note that the component that acts on the particular i^{th} component of velocity,

$$-\lambda_w A_w \left(1 - n_i n_i\right) u_{i\|}, \tag{2.160}$$

provides a form that can be potentially treated implicitly; i.e., in a way to augment the diagonal dominance of the central coefficient of the i^{th} component of velocity. The use of residual form adds a slight complexity to this implicit formulation only in that appropriate right-hand-side source terms must be added.

2.6.6. Wall Functions; Turbulent Kinetic Energy

The near wall turbulent kinetic energy can be obtained by two different procedures. The most common approach is to solve a transport equation for the near wall value of turbulent kinetic energy with a modified production and dissipation term on the right hand side of the turbulent kinetic energy equation, Equation 2.78. As will be shown below, the form of the near wall production and dissipation term are determined based on equilibrium arguments, i.e., $P_k = \rho\epsilon$.

Another common approach is to assign the value of turbulent kinetic energy that strictly results in the equality $P_k = \rho\epsilon$. In this formulation, it is assumed that the convection and diffusive flux is zero across the control volume.

Both procedures, which formally do not address the role of buoyancy production, begin with the determination of the near wall value of the production of turbulent kinetic energy. The turbulent kinetic energy production term is consistent with the law of the wall formulation and can be expressed as

$$P_{kw} = \tau_w \frac{\partial u_\|}{\partial y}. \tag{2.161}$$

The parallel velocity, $u_{\parallel}$, can be related to the wall shear stress by

$$\tau_w \frac{u^+}{y^+} = \mu \frac{u_{\parallel}}{Y_p}. \tag{2.162}$$

Taking the derivative of both sides of Equation 2.162, and substituting this relationship into Equation 2.161 yields,

$$P_{kw} = \frac{\tau_w^2}{\mu} \frac{\partial u^+}{\partial y^+}. \tag{2.163}$$

Applying the derivative of the law of the wall formulation, Equation 2.147, provides the functional form of $\partial u^+ / \partial y^+$,

$$\frac{\partial u^+}{\partial y^+} = \frac{\partial}{\partial y^+} \left[\frac{1}{\kappa} \ln \left(E y^+ \right) \right] = \frac{1}{\kappa y^+}. \tag{2.164}$$

Substituting Equation 2.164 within Equation 2.163 yields a commonly used form of the near wall production term,

$$P_{kw} = \frac{{\tau_w}^2}{\rho \kappa u_\tau Y_p}. \tag{2.165}$$

Assuming local equilibrium, $P_k = \rho \epsilon$, and using Equation 2.165 and Equation 2.148 provides the form of the near wall turbulence dissipation,

$$\epsilon = \frac{u_\tau^3}{\kappa Y_p} = \frac{C_\mu^{3/4} k^{3/2}}{\kappa Y_p}, \tag{2.166}$$

while the form of the wall shear stress is given by,

$$\tau_w = \rho C_\mu^{1/2} k \tag{2.167}$$

Under the above assumptions, the near wall value for turbulent kinetic energy, in the absence of convection, diffusion, or accumulation is given by,

$$k = \frac{u_\tau^2}{C_\mu^{1/2}}. \tag{2.168}$$

If the second method (Dirichlet condition on near wall turbulent kinetic energy) is to be used, the value of the wall friction velocity, u_τ, can be obtained in an iterative manner (Sondak and Pletcher [27]) by use of Equation 2.147. This method has been used and shown to be satisfactory (Elkaim [28]) and strictly enforces the assumptions of the law of the wall that have already been outlined.

In the method that elects to solve a near wall turbulent kinetic energy transport equation, the production and dissipation terms in the turbulent kinetic energy transport equation are [potentially] given by Equation 2.165 and

$$-\rho \epsilon = -\rho \frac{C_\mu^{3/4} k^{3/2}}{\kappa Y_p}, \tag{2.169}$$

Unfortunately, there does not seem to be one universal description of the near wall turbulent kinetic energy production term and dissipation term, Equation 2.165 and 2.169, respectively. For example, in

the law of the wall formulation, given by Launder and Spalding [29], the near wall production term is given by,

$$P_{kw} = \tau_w \frac{u_{\parallel}}{y_p}. \tag{2.170}$$

In this formulation, the wall shear stress is given by the law of the wall formulation, Equation 2.155, providing the value of y^+ is greater than 11.63 (otherwise, it is given by the laminar shear stress, Equation 2.156). The dissipation term, $-\rho\epsilon$ is given by

$$-\rho\epsilon = -\rho \frac{C_\mu^{3/4} k^{3/2}}{\kappa Y_p} \ln E y^+. \tag{2.171}$$

Note that in the absence of convection, diffusion or accumulation, the above two forms of the near wall production and dissipation source terms revert to Equation 2.168. Therefore, if the modeled flow is consistent with the law of the wall formulations, all methods should yield similar limiting behavior. Under conditions of non equilibrium, i.e., a separated flow, or values of y^+ within the viscous sublayer, some models may perform better. However, it is important to note that if the flow to be simulated includes separation and reattachment, or the computation mesh is such that y^+ is within the viscous sublayer, the law of the wall formulation can provide non sensical results.

In Fuego, there are currently two general supported methods from which to choose when applying the near wall turbulent kinetic energy boundary condition. The first method, which can be activated by the command line `omit near wall turbulent ke transport equation`, is the form of Equation 2.168 that enforces a Dirichlet condition. The second method is to solve a full control volume balance for the near wall turbulent kinetic, with convection and diffusion terms, with a modified production and dissipation term given by either

- Equations 2.165 and 2.166.
- Equations 2.170 and 2.171

The use of Equations 2.165 and 2.166 can be activated by the command line (within the wall bc block) `use equilibrium production model` which is based on the ability to express the wall shear stress consistent with the assumptions of full equilibrium between production and dissipation, Equation 2.167. In all cases that do not set a Dirichlet condition for the turbulent kinetic energy, the assembled buoyancy source terms are not removed.

2.6.7. k-ω SST Wall Functions; Turbulent Kinetic Energy

When a Dirichlet condition is not set for turbulent kinetic energy, the approach in modifying the near wall production and dissipation terms is followed.

In this approach, the equation for k is solved near the wall to remove the assumptions of log layer flow one level. However, we invoke the log layer assumption to write,

$$P_k = \frac{\tau_w^2}{\rho \kappa u_\tau Y_p}. \tag{2.172}$$

Balancing production and dissipation in the $k-\omega$ model allows us to write,

$$P_k = \rho \frac{u_\tau^3}{\kappa Y_p} = \rho \frac{(\beta')^{3/4} k^{3/2}}{\kappa Y_p}. \tag{2.173}$$

The dissipation rate is also modified accordingly such that the production equality with dissipation is retained. An alternative method is to use the approximation of of Launder and Spaulding which prescribes production as,

$$P_k = \tau_w \frac{u_{||}}{Y_p}. \tag{2.174}$$

In practice, this formulation seems to be less stable since the production and dissipation terms are now in near-equilibrium.

2.6.8. Wall Functions; Turbulence Dissipation Transport

Consistently within the literature, the near wall turbulence dissipation is assigned the Dirichlet value given by Equation 2.166. Frequently, this expression is lagged by one subiteration in an effort to maintain consistency between the Dirichlet wall condition and the freezing of the ϵ/k ratio of the turbulence dissipation equation, Equation 2.79.

2.6.9. Wall Functions; Turbulent Frequency Transport

2.6.9.1. Low Reynolds Number Treatment

The low Reynolds approach for $k-\omega$ uses a sequence of Dirichlet conditions similar to what is used for $k-\epsilon$. However, unlike the latter, $k-\omega$ requires no extra damping terms near the wall. When the wall is resolved, exact Dirichlet conditions are known for both the velocity and k:

$$\vec{u} = 0, \qquad k = 0. \tag{2.175}$$

A Dirichlet condition is also used on ω. While the $k-\epsilon$ model is rendered less stable because k appears in this boundary condition, the ω equation depends only on the near-wall grid spacing. The boundary condition is

$$\omega = \frac{6\nu}{\beta y^2}, \tag{2.176}$$

which is valid for $y^+ < 3$. Above, β depends on the model type. If SST is in use, $\beta = \beta_1$ while if the Wilcox model is in use, $\beta = \beta_0$.

2.6.9.2. High Reynolds Number Treatment

The high Reynolds approach is also quite similar to the $k-\epsilon$ model except ω is handled differently.

2.6.9.3. Automatic Wall Functions

Because ω has analytic solutions in both the log layer and viscous sub-layer, an automatic treatment is developed that blends those two solutions to provide Dirichlet conditions for all y. Let ω_h be the high Reynolds number formulation and ω_l be the low Reynolds version. Then the Dirichlet condition on ω is

$$\omega = \omega_l \sqrt{1 + \left(\frac{\omega_h}{\omega_l}\right)^2}. \tag{2.177}$$

However, u_τ for the high Reynolds ω value is computed based on the parallel velocity: The velocity equation is augmented by a traction force based on the friction velocity u_τ. This quantity may be solved for iteratively using the law of the wall. A Dirichlet condition is also used for k, assuming it is in the log region, which is similar to the $k - \epsilon$ model:

$$k = \frac{u_\tau^2}{\sqrt{\beta^*}}. \tag{2.178}$$

In the case of ω, an analytic expression is known in the log layer:

$$\omega = \frac{u_\tau}{\sqrt{\beta^*}\kappa y}, \tag{2.179}$$

which is independent of k. Note that some implementations use a predefined constant instead of $\sqrt{\beta'}$, although the standard values are consistent with these expressions. Because all these expressions require y to be in the log layer, they should absolutely not be used unless it can be guaranteed that $y^+ > 10$, and $y^+ > 25$ is preferable.

$$u_\tau = \sqrt{\nu \left| \frac{u_{||}}{y} \right|}. \tag{2.180}$$

The automatic wall function approach is obtained by removing the "omit near wall turbulent ke equation" line command and activating either the SST or KW turbulence models.

2.6.10. Wall Functions; Enthalpy Transport

For non-adiabatic boundaries, heat loss to the wall must be considered. The use of the Reynolds analogy provides a functional form of the energy transport similar to the that of the logarithmic law-of-the-wall momentum formulation. The thermal boundary layer is modeled either as a linear profile ($y^+ < 11.63$) where the thermal boundary layer is dominated by conduction or a logarithmic profile where the effects of turbulence dominate over thermal conduction, Versteeg and Malalasekera [30].

The law-of-the-wall used in Fuego has the following form,

$$q_w = \frac{\rho \left(h_w - h_p\right) u_\tau}{T^+}, \tag{2.181}$$

where

$$T^+ = \sigma_T \left[u^+ + P\right]. \tag{2.182}$$

The role of T^+ is to account for the fact that the thickness of the thermal conduction layer is [practically] of a different size than that of the viscous sublayer (momentum).

In the above equation, P is the universal "P function" (Jayatilleke [31]) and can be expressed as a function of the molecular and turbulent Prandtl number,

$$P = 9.24\left[\left(\frac{\sigma}{\sigma_T}\right)^{0.75} - 1\right]\left(1 + 0.28exp\left[-0.007\frac{\sigma}{\sigma_T}\right]\right), \tag{2.183}$$

where σ_T and σ represent the turbulent and molecular Prandtl number, respectively.

Therefore, it is seen that the so-called "P function" is the parameter that functionally changes the thickness of the thermal conduction layer from that of the viscous sublayer. For example, if one were to model a high-Prandtl number fluid such as common vegetable oil, one would note that the thickness of the viscous sublayer is far greater than that of the thermal sublayer. However, for low-Prandtl number fluids, the opposite is true. The subsequent value of T^+ ensures this functionality.

In the case of a user defined heat flux at a wall boundary condition, the full quantity is assembled as a right-hand-side source term. As a post processing step, Equation 2.181 (in temperature form) is rearranged to provide the wall temperature. In practice, the heat flux boundary condition block is to be defined on an already defined wall boundary condition block (without temperature specification). In this manner, multiple boundary conditions are "painted" on a particular sideset.

2.6.11. Wall Functions; Scalar Transport

Wall functions for use in a convective diffusive problem, e.g., diffusional transport of fuel (through multicomponent evaporation) from a jet fuel pool, are not currently supported.

2.7. INLET CONDITIONS FOR TURBULENCE QUANTITIES

2.7.1. Turbulent Kinetic Energy

The inlet turbulent kinetic energy must be specified for any simulation that involves a velocity-specified inlet. If actual values of the inlet turbulent kinetic energy are not available, then a suitable value based on basic definitions is used. In general, the kinetic energy associated with the turbulent flow is defined by,

$$k = \frac{1}{2}\left(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}\right). \tag{2.184}$$

The turbulence intensity T_i, is related to the kinetic energy by,

$$T_i = \frac{\left(\frac{2}{3}k\right)^{1/2}}{U_{ref}}. \tag{2.185}$$

Rearranging Equation 2.185 for the turbulent kinetic energy yields a working form for the specification of inlet turbulent kinetic energy based on a reference velocity, U_{ref},

$$k = \frac{3}{2}\left(U_{ref} T_{in}\right)^2 . \tag{2.186}$$

The value of U_{ref} can typically be taken to be the magnitude of the velocity.

2.7.2. Turbulence Dissipation Rate

As with the turbulent kinetic energy inlet condition for specified velocity, the inlet value of the turbulence dissipation rate must also be specified. If values are known, for instance based on experimental data, then the available data should be used. Otherwise, the following assumed form of the turbulence dissipation rate is used,

$$\epsilon = C_{\mu}^{3/4} \frac{k^{3/2}}{l}, \tag{2.187}$$

where $l = 0.07L$; L represents a characteristic length scale of an inlet eddy and k represents the inlet turbulent kinetic energy as determined above.

2.8. EDC TURBULENT COMBUSTION MODEL

The combustion submodel is Magnussen's Eddy Dissipation Concept (EDC) and development details can be found in Magnussen, et al. [32], Magnussen [33], Byggstyøl and Magnussen [34], Magnussen [35], Lilleheie, et al. [36], and Gran and Magnussen [37].

2.8.1. Model Characteristics

The underlying assumption in the EDC model is that combustion in turbulent flows is controlled by turbulent mixing. The combustion model is an algebraic zone-type model and is influenced by local cell (control volume) values only. The model derivation assumes that the minimum cell dimension is large relative to the thickness of a flame (reaction zone) structure. This thickness varies with strain-rate, but the cell size should not be less than a few millimeters. The equations are not valid for laminar or near-laminar flow, but are based on fully developed turbulence arguments. The turbulent combustion model uses information from three sources: 1) thermochemistry, 2) species and state information from the cell values, and 3) turbulence kinetic energy and dissipation. From these data, the model creates source/sink terms for species equations and the energy equation (via radiative transport).

The model function is to provide an integral effect of combustion processes occurring within the control volume for the duration of a time-step. In this manner, reaction zone structures are not resolved, but the aggregate effect of turbulent combustion is modeled. To model the integral effect, two homogeneous zones are defined within each control volume for which there is combustion, as shown in Figure 2.8-1. The zones are termed the reaction zone (fine structures) and the surrounding zone. The size and mass exchange rate between these zones are influenced by the local turbulence properties and

are the principal means by which turbulent fluctuations are accounted for within the model. The assumption that each zone is homogeneous is equivalent to assuming that the mixing within each zone is instantaneous. Since combustion occurs within (but is not limited to) the reaction zone, the assumptions for combustion correspond to those for a perfectly stirred reactor (PSR). Slower reactions can also occur in the surroundings, in which case, the assumptions for reaction in the surroundings are also consistent with PSR assumptions.

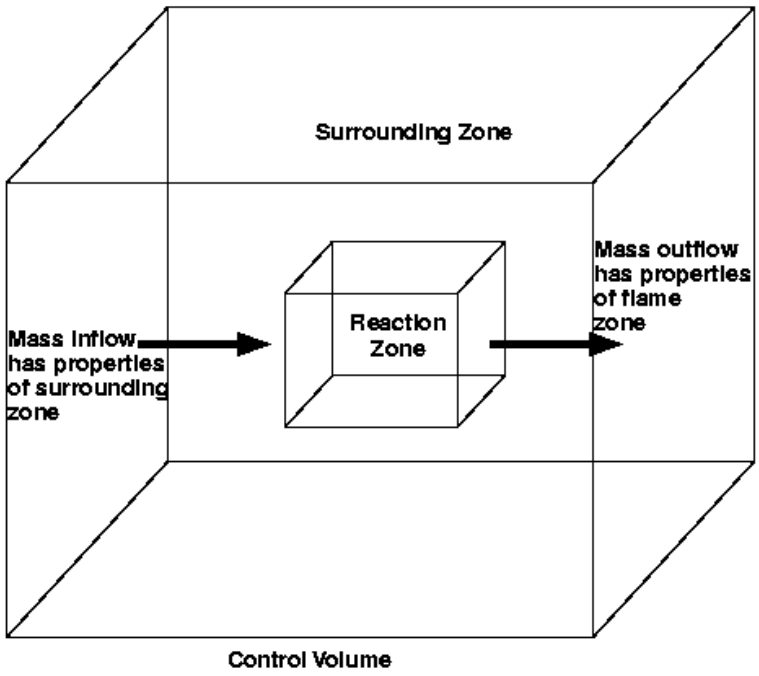


Figure 2.8-1.. Model geometry for Magnussen's Eddy Dissipation Concept. The control volume is comprised of two zones; the properties of each zone are assumed to be adequately represented by a single set of values (i.e., lumped or perfectly stirred). The mass exchange between the zones is controlled by turbulent mixing.

Please see the Fuego theory manual for more details.

2.9. LAMINAR FLAMELET TURBULENT COMBUSTION MODEL

Laminar flamelet models for non-premixed turbulent combustion treat turbulent flames as an ensemble of laminar diffusion flames. [38] Nonequilibrium chemistry effects may be included in the model by

accounting for localized fluid strain, resulting in what is classically called the Strained Laminar flamelet Model (SLFM). Nonadiabatic effects may also be included by accounting for losses to the surroundings in the ensemble of flamelets.

The fundamental assumption is that the chemical time scales of the important reactions are fast enough that they occur only in a thin layer around stoichiometry, thinner (ideally) than the smallest scales of the turbulence. Defining a small quantity $\epsilon = \ell_{\text{reaction zone}}/\ell_{\text{mixing layer}} \ll 1$, we can examine the governing equations in that thin region using a multiscale asymptotic expansion as

$$Y_i = Y_i(\zeta, \tau, x, t) + \epsilon Y_i^1(\zeta, \tau, x, t) + \ldots, \qquad \zeta = \frac{Z(x,t) - Z_{\text{st}}}{\epsilon} \text{ and } \tau = t/\epsilon^2. \tag{2.188}$$

Collecting the dominant terms, making some standard simplifications, and assuming that the chemical reaction scales as ϵ^{-2}, the state of the gas depends on the flow scale Z and $\chi = 2D|\nabla Z|^2$:

$$\begin{aligned}
&\rho\frac{\partial Y_i}{\partial t} - \frac{\rho\chi}{2}\frac{1}{\text{Le}_i}\frac{\partial^2 Y_i}{\partial Z^2} - \dot{\omega}_i(\vec{\Phi}) = 0 \\
&\rho\frac{\partial T}{\partial t} - \frac{\rho\chi}{2}\left(\frac{\partial^2 T}{\partial Z^2} + \frac{1}{c_p}\frac{\partial c_p}{\partial Z}\frac{\partial T}{\partial Z}\right) - \dot{\omega}_T(\vec{\Phi}) = 0 \\
&\text{and } T(Z=0,t) = T_{\text{ox.}}, T(Z=1,t) = T_{\text{fuel}}, Y_i(Z=0,t) = Y_{\text{i,ox.}}, Y_i(Z=1,t) = Y_{\text{i,fuel}}, \\
&\text{with } \rho = \rho(\vec{\Phi}) \text{ and } c_p = c_p(\vec{\Phi}),
\end{aligned} \tag{2.189}$$

where $\vec{\Phi}$ is the state vector $\vec{\Phi} = (P_{\text{th}}, T, Y_0, Y_1, \ldots, Y_N)$. The approximation allows us to resolve the chemical scales in the phase space of the mixture fraction instead of on a three-dimensional grid, granting dramatic computational savings. If we make the additional assumption that the chemistry is quasi-steady on the scale of the flow, then the chemical structure in mixture fraction space can be pre-computed offline from the simulation for a range of flow parameters χ and tabulated (using `fuego_tabular_props`). During the flow simulation, the solution of the flamelet simulation can be queried to determine required flow properties, e.g. $\rho = \rho(Z, \chi)$. Note that the flamelet formulation in Eq. 2.189 is specifically for a "two-stream" problem, with constant Lewis numbers, where the boundary and initial conditions of the simulation can be completely described by linear combinations of two constant state vectors. Additional "streams" and boundary heat losses will require additional transport equations to be solved.

This section summarizes the basic formulation and implementation details of both the adiabatic and nonadiabatic flamelet model and SLF model, including both the property table generation procedure in `fuego_tabular_props` and the usage of the property table in `fuego` to evaluate turbulent filtered quantities of interest for both adiabatic and nonadiabatic configurations.

2.9.1. Adiabatic Property Table Generation

2.9.1.1. Laminar Flamelet Generation

Unstrained flamelet libraries, where nonequilibrium chemistry effects may be neglected with respect to fluid strain rates, can be generated directly with the `fuego_tabular_props` application. These

libraries should be used either in laminar flow or in turbulent flow where the turbulence/chemistry interactions may be neglected.

Equilibrium chemistry, Burke-Schumann chemistry, or nonreacting flow scenarios are supported in configurations where there are two or more streams that may be mixed and potentially reacted. The stream composition is parameterized by the mixture fraction vector Z_m, where each of the M component represents the fraction of mass that originated in that stream, where there are N streams and $M = N - 1$. The mixture fraction for the final stream may be evaluated as $Z_N = 1 - \sum_{m=1}^{M} Z_m$.

The resulting flamelet data can then be assembled into a sequence of multi-dimensional tables of dependent variable ϕ as a function of the mixture fraction vector, $\phi(Z_m)$, and can be used directly for laminar simulations. Adding turbulence interactions, nonequilibrium effects, and nonadiabatic effects will increase the dimensionality of this lookup table and require additional processing. See the following sections for more information.

2.9.1.2. Strained Laminar Flamelet Libraries

Strained laminar flamelet data may be generated for use in Fuego with the Spitfire code. This data is two-dimensional in nature, determined by the mixture fraction and a reference scalar dissipation rate χ_o at a reference mixture fraction Z_o. The instantaneous laminar scalar dissipation rate is defined as

$$\chi = 2D \frac{\partial Z}{\partial x_i} \frac{\partial Z}{\partial x_i}, \tag{2.190}$$

with D being the molecular mass diffusion coefficient. The reference value χ_o is arbitrary, although typical choices include the stoichiometric value $\chi_{\mathrm{st}} = \chi(Z_{\mathrm{st}})$ or the maximum value $\chi_{\mathrm{max}} = \chi(Z = 0.5)$. Stoichiometric values are used in Fuego. After generating flamelet data with Spitfire, the `fuego_tabular_props` application can be used to assemble it into a sequence of multi-dimensional tables of dependent variable ϕ as a function of the mixture fraction vector and reference scalar dissipation rate, $\phi(Z_m, \chi_o)$. These tables may be then be used to evaluate properties in Fuego. Please see section 2.9.6 for details regarding the generation of flamelet libraries with Spitfire.

2.9.1.3. Turbulent Averaging

In turbulent simulations, a filtered form of the governing equations are solved to reduce the resolution requirements to an affordable level. Temporal filtering is used in Reynolds Averaged Navier-Stokes (RANS) models and spatial filtering is used in Large Eddy Simulation (LES) models. Both types of filtering are represented with the notation $\bar{\phi}$, and are handled similarly in the present work. Density-weighted, or Favre filtering greatly simplifies the treatment of variable-density flow. A Favre-filtered quantity is represented by $\tilde{\phi} \equiv \overline{\rho\phi}/\bar{\rho}$. Please see the `fuego` theory manual for further details.

For use in turbulent simulations, a Favre-filtered version of the variables in the property table must be calculated. This is performed by convoluting the property variable with the joint PDF of the

independent variable sub-filter fluctuations, and is mathematically expressed as

$$\tilde{\phi}(\tilde{Z}_m, \widetilde{Z''^2}, \tilde{\chi}) = \int_0^\infty \int_0^1 \phi(Z_m, \chi_o)\; p_{Z\chi}(Z_i, \chi; \tilde{Z}_m, \widetilde{Z''^2}, \tilde{\chi})\, \mathrm{d}Z_i\, \mathrm{d}\chi, \tag{2.191}$$

where $p_{Z\chi}(Z_i, \chi; \tilde{Z}_m, \widetilde{Z''^2}, \tilde{\chi})$ is the joint PDF of sub-filter fluctuations of the dependent variable ϕ in Z_i-χ space, parameterized by the filtered mixture fractions $\tilde{Z}_m$ and the variance $\widetilde{Z''^2}$ of mixture fraction component Z_i, and the filtered scalar dissipation rate $\tilde{\chi}$. The reference scalar dissipation rate has the functionality $\chi_o(\tilde{Z}_i, \widetilde{Z''^2}, \tilde{\chi})$, which will be discussed in the following section. Variance of only a single component of mixture fraction, Z_i, is considered at present for simplicity, although extensions to include additional components are possible. Statistical independence will be assumed between Z_i and χ fluctuations, so that

$$\tilde{\phi}(\tilde{Z}_m, \widetilde{Z''^2}, \tilde{\chi}) = \int_0^\infty \int_0^1 \phi(Z_m, \chi_o)\; p_Z(Z_i; \tilde{Z}_m, \widetilde{Z''^2})\, p_\chi(\chi; \tilde{\chi})\, \mathrm{d}Z_i\, \mathrm{d}\chi. \tag{2.192}$$

For the present work, $p_Z(Z_i; \tilde{Z}_m, \widetilde{Z''^2})$ will be modeled as either a beta PDF or a clipped Gaussian PDF and $p_\chi(\chi; \tilde{\chi})$ will be modeled as the delta function $\delta(\chi - \tilde{\chi})$.

2.9.1.4. Property Table Implementation

The convolution integral in Equation 2.192 would be prohibitively expensive to evaluate each time a value for $\tilde{\phi}$ is needed by a turbulent reacting simulation. Therefore, this integral will be pre-calculated so that each property table query will only involve an interpolation from a table of values.

Storing the final $\tilde{\phi}(\tilde{Z}_m, \widetilde{Z''^2}, \tilde{\chi})$ values directly is undesirable since the range of possible $\tilde{\chi}$ values for each flamelet is different, resulting in a non-orthogonal table. Instead, the values $\tilde{\phi}_T(\tilde{Z}_m, \widetilde{Z''^2}, \chi_o)$ are stored in an orthogonal table that is indexed by $\tilde{Z}_m$, $\widetilde{Z''^2}$, and $\chi_o(\tilde{Z}_i, \widetilde{Z''^2}, \tilde{\chi})$. These tabulated values are calculated by

$$\tilde{\phi}_T(\tilde{Z}_m, \widetilde{Z''^2}, \chi_o) = \int_0^1 \phi(Z_m, \chi_o)\; p_Z(Z_i; \tilde{Z}_m, \widetilde{Z''^2})\, \mathrm{d}Z_i. \tag{2.193}$$

The reference scalar dissipation rate χ_o needed for lookup in the table for $\tilde{\phi}_T(\tilde{Z}_m, \widetilde{Z''^2}, \chi_o)$ can be evaluated from the local filtered scalar dissipation rate $\tilde{\chi}$ through laminar flamelet theory. The instantaneous scalar dissipation rate χ can be approximated by

$$\begin{aligned} \chi &= \chi_{\max} \exp\left(-2[\mathrm{erfc}^{-1}(2Z)]^2\right) \\ &= \chi_{\max} F_\chi(Z), \end{aligned} \tag{2.194}$$

where $\chi_{\max}$ is the maximum scalar dissipation rate found in the counterflow diffusion flame, which occurs at the stagnation point where $Z = 0.5$. (Note that this expression has not yet been extended to multiple mixture fractions, so that this treatment is only applicable for two-stream problems.) The

value of χ at any reference location in the flamelet can be similarly approximated, so that $\chi_o = \chi_{\max} F_\chi(Z_o)$. Combining these models by equating the unknown $\chi_{\max}$ yields a closed-form expression linking the scalar dissipation rate at any location to the reference value on the flamelet with the same characteristic $\chi_{\max}$,

$$\chi = \chi_o \frac{F_\chi(Z)}{F_\chi(Z_o)}. \tag{2.195}$$

Applying the filtering operation in Equation 2.192 to both sides of Equation 2.195 for a single-mixture fraction configuration yields

$$\tilde{\chi} = \int_0^\infty \int_0^1 \chi_o \frac{F_\chi(Z)}{F_\chi(Z_o)} \, p_Z(Z; \tilde{Z}, \widetilde{Z''^2}) \, p_\chi(\chi; \tilde{\chi}) \, \mathrm{d}Z \, \mathrm{d}\chi \tag{2.196}$$

$$= \frac{\chi_o}{F_\chi(Z_o)} \int_0^\infty p_\chi(\chi; \tilde{\chi}) \, \mathrm{d}\chi \int_0^1 F_\chi(Z) \, p_Z(Z; \tilde{Z}, \widetilde{Z''^2}) \, \mathrm{d}Z \tag{2.197}$$

$$= \frac{\chi_o}{F_\chi(Z_o)} \int_0^1 F_\chi(Z) \, p_Z(Z; \tilde{Z}, \widetilde{Z''^2}) \, \mathrm{d}Z, \tag{2.198}$$

so that the filtered reference scalar dissipation rate can be calculated from the filtered quantities provided by the turbulent flame simulation as

$$\chi_o(\tilde{Z}, \widetilde{Z''^2}, \tilde{\chi}) = \frac{\tilde{\chi} \, F_\chi(Z_o)}{\int_0^1 F_\chi(Z) \, p_Z(Z; \tilde{Z}, \widetilde{Z''^2}) \, \mathrm{d}Z,}. \tag{2.199}$$

To decrease computational cost, the integral in the denominator can be interpolated from pre-calculated values in a two-dimensional table as a function of $\tilde{Z}$ and $\tilde{Z''^2}$.

To summarize, the turbulent reacting simulation will query the property table for the variable $\tilde{\phi}(\tilde{Z}_m, \widetilde{Z''^2}, \tilde{\chi})$. Internally, Equation 2.199 will be used to calculate χ_o as a function of the provided filtered independent variables. This value will then be used along with the provided independent variables to interpolate a value for $\tilde{\phi}_T(\tilde{Z}_m, \widetilde{Z''^2}, \chi_o)$ from the stored table that was pre-calculated with Equation 2.193. This interpolated value will then be returned to the main simulation as the requested value for $\tilde{\phi}(\tilde{Z}_m, \widetilde{Z''^2}, \tilde{\chi})$.

If turbulence/chemistry interactions are to be neglected in the simulation, the delta function $\delta(Z - \tilde{Z})$ may be used for $p_Z(Z; \tilde{Z})$ in Equation 2.199 so that the reference scalar dissipation rate can be computed simply as

$$\chi_o(\tilde{Z}, \tilde{\chi}) = \frac{\tilde{\chi} \, F(Z_o)}{F(\tilde{Z})}. \tag{2.200}$$

Once the multidimensional property table has been generated, it can be imported into `fuego` and queried for the dependent variables as a function of the independent variables $\tilde{Z}_m$, $\widetilde{Z''^2}$, and $\tilde{\chi}$. Models are required for each of these independent variables used by the flamelet property table. Sections 2.9.3–2.9.5 present models for each of these quantities for each of the supported turbulence closure models.

2.9.2. Nonadiabatic Property Table Generation

When including the effects of radiative or convective heat losses in a flamelet simulation, additional parameterizations beyond those in the previous section are required. These are the "conserved enthalpy", h^* and heat loss parameter γ, where the heat loss parameter is defined as $\gamma = h - h^*$. The conserved enthalpy is identical to the traditional enthalpy except that its transport equation omits all source terms (typically due to radiative losses).

This formulation is used as a way to parameterize losses in a manner that is consistent with the opposed diffusion flame burner simulations used to generate the flamelet libraries. In these burner simulations, the inflowing pure stream states are fixed and cannot experience any heat losses; Losses only occur in the interior of the burner, and are represented by γ variation. A range of inflowing pure stream states may also be computed, and are parameterized through h^* variation. In this way, the full range of possible states may be tabulated and retrieved in a fire simulation through values of h and h^*, which are both straightforward to compute.

For turbulent simulations, the Favre-filtered property variable $\tilde{\phi}$ is evaluated as

$$\begin{aligned}\tilde{\phi}(\tilde{Z}_m, \widetilde{Z''^2}, \tilde{\chi}, \tilde{\gamma}, \tilde{h}^*) \;=\; & \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{0}^{\infty}\int_{0}^{1} \phi(Z_m, \chi_o, \gamma_o, h_o^*)\; p_{Z\chi\gamma h^*}(Z_m, \chi, \gamma, h^*; \\ & \tilde{Z}_m, \widetilde{Z''^2}, \tilde{\chi}, \tilde{\gamma}, \tilde{h}^*)\, \mathrm{d}Z_m\, \mathrm{d}\chi\, \mathrm{d}\gamma\, \mathrm{d}h^*, \end{aligned} \tag{2.201}$$

where γ_o and h_o^* are reference values of the heat loss parameter and the conserved enthalpy, respectively, to be defined in the following sections. Statistical independence will be assumed between fluctuations of each Z_m component, χ, γ, and h^*, so that

$$\begin{aligned}\tilde{\phi}(\tilde{Z}_m, \widetilde{Z''^2}, \tilde{\chi}, \tilde{\gamma}, \tilde{h}^*) \;=\; & \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{0}^{\infty}\int_{0}^{1}\int_{0}^{1} \phi(Z_m, \chi_o, \gamma_o, h_o^*)\; p_{Z_i}(Z_i; \tilde{Z}_i, \widetilde{Z''^2})\, p_{Z_m}(Z_m; \tilde{Z}_m) \\ & p_\chi(\chi; \tilde{\chi})\, p_\gamma(\gamma; \tilde{\gamma})\, p_{h^*}(h^*; \tilde{h}^*)\, \mathrm{d}Z_i\, \mathrm{d}Z_{m\neq i}\, \mathrm{d}\chi\, \mathrm{d}\gamma\, \mathrm{d}h^*. \end{aligned} \tag{2.202}$$

For the present work, $p_{Z_i}(Z_i; \tilde{Z}_i, \widetilde{Z''^2})$ will be modeled as either a beta PDF or a clipped Gaussian PDF, and $p_{Z_m}(Z_m; \tilde{Z}_m)$, $p_\chi(\chi; \tilde{\chi})$, $p_\gamma(\gamma; \tilde{\gamma})$, and $p_{h^*}(h^*; \tilde{h}^*)$ will be modeled as the delta functions $\delta(Z_m - \tilde{Z}_m)$, $\delta(\gamma - \tilde{\gamma})$, and $\delta(\chi - \tilde{\chi})$, $\delta(\gamma - \tilde{\gamma})$, and $\delta(h^* - \tilde{h}^*)$, respectively.

The convolution integral in Equation 2.202 would be prohibitively expensive to evaluate each time a value for $\tilde{\phi}$ is needed by a turbulent reacting simulation. Therefore, this integral will be pre-calculated so that each property table query will only involve an interpolation from a table of values.

Storing the final $\tilde{\phi}(\tilde{Z}_m, \widetilde{Z''^2}, \tilde{\chi}, \tilde{\gamma}, \tilde{h^*})$ values directly is undesirable since the range of possible $\tilde{\chi}$, $\tilde{\gamma}$, and $\tilde{h}^*$ values for each flamelet is different, resulting in a non-orthogonal table. Instead, the values $\tilde{\phi}_T(\tilde{Z}_m, \widetilde{Z''^2}, \chi_o, \gamma_o, h_o^*)$ are stored in an orthogonal table that is indexed by $\tilde{Z}_m$, $\widetilde{Z''^2}$, $\chi_o(\tilde{Z}_i, \widetilde{Z''^2}, \tilde{\chi})$,

$\gamma_o(\tilde{Z}_i, \widetilde{Z''^2}, \tilde{\gamma})$, and $h_o^*(\tilde{Z}_i, \tilde{h}^*)$. These tabulated values are calculated by

$$\tilde{\phi}_T(\tilde{Z}_m, \widetilde{Z''^2}, \chi_o, \gamma_o, h_o^*) = \int_0^1 \int_0^1 \phi(Z_m, \chi_o, \gamma_o, h_o^*)\; p_{Z_i}(Z_i; \tilde{Z}_i, \widetilde{Z''^2})\, p_{Z_m}(Z_m; \tilde{Z}_m)\, \mathrm{d}Z_i\, \mathrm{d}Z_{m\neq i}. \tag{2.203}$$

The required reference values of γ_o and h_o^* are described in the following sections.

2.9.2.1. Boundary heat loss

The addition of a temperature boundary condition on the wall requires a modification of the flamelet formulation of Eq. 2.189. The equation for a normalized temperature variable, $\theta = T/T_{\text{ox.}} - 1$, is

$$\partial_t(\rho\theta) + \nabla\cdot(\rho\mathbf{u}\theta) - \nabla\cdot(\lambda\nabla\theta) = L_D\theta = \dot{\omega}_T(\theta, \vec{Y}) \text{ in } \Omega \tag{2.204}$$

$$\theta(x \in \partial\Omega_{\text{fuel}}) = \theta_{\text{fuel}},\ \theta(x \in \partial\Omega_{\text{ox}}) = 0,\ \text{and } \theta(x \in \partial\Omega_{\text{wall}}) = \theta_{\text{wall}}, \tag{2.205}$$

which now has an extra boundary term $\theta(x \in \partial\Omega_{\text{wall}}) = \theta_{\text{wall}}$. The extra boundary condition remains when we apply the flamelet approximation, leaving $\theta(Z = Z_{\text{wall}}) = \theta_{\text{wall}}$. The value of the mixture fraction at the wall, however, is undecided: we only know that $\nabla Z \cdot n = 0$. During the simulation, the value of mixture fraction directly evaluated at the wall can be determined dynamically and the value of temperature can be computed. Away from the wall, however, one in principle would need to follow the ζ coordinate from the flamelet transformation until it intersects the wall. However, given that $\nabla Z \cdot n = 0$, the gradient trajectory in principle is tangential to the wall. Although the flamelet equation itself is well-posed, the asymptotic derivation of the flamelet model in the very near region to a nonadiabatic wall and the equations need to modified in some fashion to account.

Flamelets can readily be described when they are adiabatic; in the limit of unity Lewis numbers and adiabatic systems the enthalpy is a linear function of the mixture fraction. The existence of radiative transport and wall heat transfer introduces deviations from this linear relationship between h and Z. Heat losses at the predominant boundary temperature are a common scenario. Defining a reference 'boundary temperature' at $T_{\text{fr}}(Z) = (1-Z)T_{\text{ox}} + ZT_{\text{fuel}}$, then this case a simplified flamelet temperature equation with heat losses could be written as

$$\rho\partial_t T - \frac{1}{2}\rho\chi\partial_Z^2 T = \dot{\omega}_T(\vec{\Phi}) - H_T(T - T_{\text{fr}}) \tag{2.206}$$

where H_T represents a heat transfer coefficient that will be further discussed below. This gives a heat loss term that is linear in T. Alternately, the heat loss can be written specifically for radiative-style losses, $\dot{q}_{\text{losses}} = \sigma(\vec{\Phi})(T^4 - T_{\text{fr}}^4)$. Regardless, with heat loss expressed in terms of T_{fr} the flamelet enthalpy is no longer linear in Z but instead takes on a roughly inverted triangular form with an extrema at the peak temperature, roughly Z_{st}. This has led us to express the difference between the adiabatic enthalpy, defined as $h_c = h_{\text{ox}}(1-Z) + h_{\text{fuel}}Z$, and the actual flamelet computed enthalpy, h, as $\gamma = h - h_c$. The introduction of γ is done strictly as an expedient for generation of flamelet libraries. By assuming a triangular form (or any particular assumed form) we can stretch the table entries into a square format by tabulating as a function of the stoichiometric value of γ_{st}. This does require the use of the assumed form for γ for converting from the local Z value of $\gamma = h(Z) - h_c(Z)$ to γ_{st}. Comparison with DNS

and unsteady flamelets for laminar flames shows good agreement with this type of enthalpy defect model for radiation in unity Lewis number flames (which is an appropriate assumption for turbulent, hydrocarbon fires) [**?**]. We make an assumption of path independence for the solution of at particular integrated heat loss, but in reality the solution will depend somewhat on the value of H_T and the form of the added heat loss term.

The compensation for boundary heat loss can be extended to a full range of temperature (below $T_{\text{fr.}}$) in the flamelet libraries by not only including an integrated heat loss rate from the flamelets, but also a translation of the flamelet in enthalpy space. This translation is simply denoted $h^\star$, where the conserved enthalpy line is shifted as $h_c = h_{\text{ox}}(1-Z) + h_{\text{fuel}}Z + h^\star$. This allows a ful description of wall boundary heat loss. Having two heat loss parameterizations, however, makes the lookup procedure non-unique, requiring a method for deciding which point on $(\gamma, h^\star)$ to use for the flamelet lookup. We prefer γ and use γ as much as possible. When γ is insufficient, which would is the case for overly cold or hot walls (wall temperatures outside of the range of temperatures spanned by the solutions of Eq. 2.206). At the wall boundaries, the conserved enthaply is defined to not be affected by heat loss while the true enthalpy is, providing γ at the wall.

2.9.2.2. Property Table Heat Loss Parameterization

For nonadiabatic flamelet library generation and tabulation, a functional form for the heat loss parameter γ in terms of reference quantities is required, similar in concept to the form of χ in Equation 2.194. The value of γ must be zero in each of the pure streams, and should have a maximum value near the stoichiometric flame sheet since this quantity typically represents radiative losses to the environment. A piecewise linear functional form is selected for simplicity. For a single mixture fraction, this form is simply

$$\gamma = \gamma_o \, F_\gamma(Z, Z_o), \tag{2.207}$$

where γ_o is a reference heat loss at reference state Z_o (selected to be the stoichiometric condition Z_{st}) and the nondimensional function $F_\gamma(Z, Z_o)$ is defined as

$$F_\gamma(Z, Z_o) = \begin{cases} \frac{Z}{Z_o} & : \quad Z \leq Z_o \\ \frac{1-Z}{1-Z_o} & : \quad Z > Z_o \end{cases} . \tag{2.208}$$

For multiple mixture fractions, γ is calculated by

$$\gamma = \gamma_o \, F_\gamma(Z_m, Z_{o,km}, \gamma_{o,k}^{\max}), \tag{2.209}$$

where γ_o is the maximum-magnitude reference heat loss in the vector $\gamma_{o,k}^{\max}$, which contains the reference heat loss parameters corresponding to maximum thermal losses for the K stoichiometric mixture fractions that can be defined between stream pairs, $Z_{o,km}$. The multiple stoichiometric mixture fractions are necessary because a single unique stoichiometric mixture fraction does not exist when using multiple mixture fractions.

The functional form for F_γ is quite complex for multiple mixture fractions, and will only be described briefly here. In general, for a three-stream problem, there are two independent mixture fractions and the realizable mixture fraction space is the triangle where the two mixture fractions sum to a value less than

or equal to unity. The value of F_γ must be zero at the "corners" of this space, where the coordinates are $(0,0)$, $(0,1)$, and $(1,0)$. The multiple stoichiometric mixture fractions between stream pairs will define points along the boundaries of this realizable mixture fraction space that represent local maxima in the heat loss distribution along that boundary. Straight lines may be used to connect these points in mixture fraction space, forming a "ridge" in the multidimensional F_γ distribution. When definable, a linear fit is used between this ridge and a corner where γ is zero. When not uniquely definable, linear fits are used between the ridge and the adjacent boundary value along rays extended from the opposite corner of the state space. Note that the values $\gamma_{o,k}$ are required for the calculation of F_γ so that the final function may be normalized to a unity maximum value with appropriate relative scaling between the boundary heat loss values. Note that no more than a three-stream configuration is currently supported by `fuego_tabular_props`.

Applying the filtering operation in Equation 2.202 to both sides of Equation 2.209 yields

$$\begin{aligned}
\tilde{\gamma} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{0}^{\infty}\int_{0}^{1}\int_{0}^{1} \gamma_o\, F_\gamma(Z_m, Z_{o,km}, \gamma_{o,k}^{\max})\, p_{Z_i}(Z_i; \tilde{Z}_i, \widetilde{Z''^2})\, p_{Z_m}(Z_m; \tilde{Z}_m) \\
&\quad p_\chi(\chi; \tilde{\chi})\, p_\gamma(\gamma; \tilde{\gamma})\, p_{h^*}(h^*; \tilde{h}^*)\, \mathrm{d}Z_i\, \mathrm{d}Z_{m\neq i}\, \mathrm{d}\chi\, \mathrm{d}\gamma\, \mathrm{d}h^* \qquad (2.210) \\
&= \gamma_o \int_{-\infty}^{\infty} p_{h^*}(h^*; \tilde{h}^*)\, \mathrm{d}h^* \int_{-\infty}^{\infty} p_\gamma(\gamma; \tilde{\gamma})\, \mathrm{d}\gamma \int_{0}^{\infty} p_\chi(\chi; \tilde{\chi})\, \mathrm{d}\chi \\
&\quad \int_{0}^{1}\int_{0}^{1} F_\gamma(Z_m, Z_{o,km}, \gamma_{o,k}^{\max})\, p_{Z_i}(Z_i; \tilde{Z}_i, \widetilde{Z''^2})\, p_{Z_m}(Z_m; \tilde{Z}_m)\, \mathrm{d}Z_i\, \mathrm{d}Z_{m\neq i} \qquad (2.211) \\
&= \gamma_o \int_{0}^{1}\int_{0}^{1} F_\gamma(Z_m, Z_{o,km}, \gamma_{o,k}^{\max})\, p_{Z_i}(Z_i; \tilde{Z}_m, \widetilde{Z''^2})\, p_{Z_m}(Z_m; \tilde{Z}_m)\, \mathrm{d}Z_i\, \mathrm{d}Z_{m\neq i}, \qquad (2.212)
\end{aligned}$$

so that the filtered heat loss parameter can be calculated from the filtered quantities provided by the turbulent flame simulation as

$$\gamma_o(\tilde{Z}_m, \widetilde{Z''^2}, \tilde{\gamma}) = \frac{\tilde{\gamma}}{\int_0^1\int_0^1 F_\gamma(Z_m, Z_{o,km}, \gamma_{o,k}^{\max})\, p_{Z_i}(Z_i; \tilde{Z}_i, \widetilde{Z''^2})\, p_{Z_m}(Z_m; \tilde{Z}_m)\, \mathrm{d}Z_i\, \mathrm{d}Z_{m\neq i}}. \qquad (2.213)$$

To decrease computational cost, the integral in the denominator can be interpolated from pre-calculated values in a multi-dimensional table as a function of $\tilde{Z}_m$ and $\widetilde{Z''^2}$. Equation 2.213 can be used during a simulation to convert filtered independent variables to the reference heat loss parameter required to perform table lookups to retrieve $\tilde{\phi}_T$.

If turbulence/chemistry interactions are to be neglected in the simulation, the delta function $\delta(Z_i - \tilde{Z}_i)$ may be used for $p_{Z_i}(Z_i; \tilde{Z}_i, \widetilde{Z''^2})$ in Equation 2.213 so that the reference heat loss parameter can be computed simply as

$$\gamma_o(\tilde{Z}_m, \tilde{\gamma}) = \frac{\tilde{\gamma}}{F_\gamma(\tilde{Z}_m, Z_{o,km}, \gamma_{o,k}^{\max})}. \qquad (2.214)$$

2.9.2.3. Property Table Conserved Enthalpy Parameterization

For nonadiabatic flamelet library generation and tabulation, a functional form for the conserved enthalpy h^* in terms of reference quantities is required. The value of h^* should vary linearly within the range provided for each of the pure streams as a function of a reference heat loss parameter h_o^*, with an appropriate stream-weighted blending for all other compositions.

The stream-weighted mixture properties are computed with an augmented mixture fraction vector Z'_n in terms of Z_m,

$$Z'_n = \left[Z_1, Z_2, \ldots, Z_M, 1 - \sum_{m=1}^{M} Z_m \right], \tag{2.215}$$

where the last component is simply the last implied mixture fraction to recover a unity sum. A reference augmented mixture fraction is defined as the centroid of the realizable mixture fraction space with each component being identical and equal to

$$Z'_{o,n} = \frac{1}{N}. \tag{2.216}$$

From these definitions, minimum and maximum reference conserved enthalpy values may be computed as

$$h^*_{o,\min} = \sum_{n=1}^{N} h^*_{\text{stream,min},n} \, Z'_{o,n} \tag{2.217}$$

$$h^*_{o,\max} = \sum_{n=1}^{N} h^*_{\text{stream,max},n} \, Z'_{o,n}, \tag{2.218}$$

where $h^*_{\text{stream,min},n}$ and $h^*_{\text{stream,max},n}$ are vectors of the minimum and maximum conserved enthalpy in pure stream n, respectively. The conserved enthalpy can then be modeled as

$$h^* = h^*_{\min,Z} + \left(h^*_o - h^*_{o,\min} \right) a_Z, \tag{2.219}$$

where the mixture-weighted minimum conserved enthalpy is

$$h^*_{\min,Z} = \sum_{n=1}^{N} h^*_{\text{stream,min},n} \, Z'_n \tag{2.220}$$

and the mixture-weighted stream variation proportionality constant is

$$a_Z = \sum_{n=1}^{N} \left(\frac{h^*_{\text{stream,max},n} - h^*_{\text{stream,min},n}}{h^*_{o,\max} - h^*_{o,\min}} \right) Z'_n. \tag{2.221}$$

Applying the filtering operation in Equation 2.202 to both sides of Equation 2.219 yields

$$\tilde{h}^* = \tilde{h}^*_{\min,Z} + \left(h^*_o - h^*_{o,\min} \right) \tilde{a}_Z, \tag{2.222}$$

where the two mixture-weighted quantities are now expressed in terms of the augmented filtered mixture fraction as

$$\tilde{h}^*_{\text{min},Z} = \sum_{n=1}^{N} h^*_{\text{stream,min},n}\, \tilde{Z}'_n \tag{2.223}$$

and

$$\tilde{a}_Z = \sum_{n=1}^{N} \frac{h^*_{\text{stream,max},n} - h^*_{\text{stream,min},n}}{h^*_{o,\max} - h^*_{o,\min}}\, \tilde{Z}'_n. \tag{2.224}$$

This allows the reference conserved enthalpy to be expressed in terms of the filtered quantities provided by the turbulent flame simulation as

$$h^*_o(\tilde{Z}_m, \tilde{h}^*) = h^*_{o,\min} + \frac{\tilde{h}^* - \tilde{h}^*_{\text{min},Z}}{\tilde{a}_Z}. \tag{2.225}$$

2.9.3. Filtered Scalar Dissipation Rate

2.9.3.1. RANS Model

For RANS turbulence closure models the instantaneous laminar scalar dissipation rate given in Equation 2.190 can be Favre-filtered and expanded to the form

$$\bar{\rho}\tilde{\chi} = 2\overline{\rho D \frac{\partial Z}{\partial x_i}\frac{\partial Z}{\partial x_i}} \tag{2.226}$$

$$= 2\overline{\rho D}\frac{\partial \tilde{Z}}{\partial x_i}\frac{\partial \tilde{Z}}{\partial x_i} + 4\overline{\rho D \frac{\partial Z''}{\partial x_i}\frac{\partial \tilde{Z}}{\partial x_i}} + 2\overline{\rho D \frac{\partial Z''}{\partial x_i}\frac{\partial Z''}{\partial x_i}}. \tag{2.227}$$

The middle term on the RHS is neglected for constant density flow [39]. The first term is referred to as the mean scalar dissipation rate

$$\bar{\rho}\tilde{\chi}_m = 2\overline{\rho D}\frac{\partial \tilde{Z}}{\partial x_i}\frac{\partial \tilde{Z}}{\partial x_i} \tag{2.228}$$

and the third term is the perturbation scalar dissipation rate $\bar{\rho}\tilde{\chi}_p$. This term can be modeled as

$$\bar{\rho}\tilde{\chi}_p = 2\overline{\rho D \frac{\partial Z''}{\partial x_i}\frac{\partial Z''}{\partial x_i}} \tag{2.229}$$

$$\approx C_\chi \bar{\rho}\frac{\epsilon}{k}\widetilde{Z''^2} \tag{2.230}$$

for RANS-based turbulence closures where $\frac{\epsilon}{k}$ provides an inverse turbulence time scale, $\widetilde{Z''^2}$ is the scalar variance that will be modeled in Section 2.9.5, and C_χ is a model constant that typically has a value of 2.0. [38]

Expressing the molecular mass diffusivity as $\bar{\rho}D = \mu/\text{Sc}$, where μ is the molecular viscosity and Sc is the Schmidt number, the modeled total filtered scalar dissipation rate for RANS closures is

$$\tilde{\chi} = \tilde{\chi}_m + \tilde{\chi}_p \tag{2.231}$$

$$\approx \frac{2}{\bar{\rho}}\frac{\mu}{\text{Sc}}\frac{\partial \tilde{Z}}{\partial x_i}\frac{\partial \tilde{Z}}{\partial x_i} + C_\chi \frac{\epsilon}{k}\widetilde{Z''^2}. \tag{2.232}$$

2.9.3.2. LES Model

For LES closures Equation 2.231 also applies, so that the total filtered scalar dissipation rate is the sum of the mean and the perturbation scalar dissipation rates. The mean scalar dissipation rate is expressed identically to RANS closures as

$$\tilde{\chi}_m = \frac{2}{\bar{\rho}} \frac{\mu}{\mathrm{Sc}} \frac{\partial \tilde{Z}}{\partial x_i} \frac{\partial \tilde{Z}}{\partial x_i}. \tag{2.233}$$

The perturbation scalar dissipation rate $\tilde{\chi}_p$ represents the sub-filter dissipation of scalar variance, and can be modeled by assuming that sub-filter dissipation is in local equilibrium with sub-filter production, and that the sub-filter production can be modeled with a gradient transport assumption as [40]

$$\bar{\rho}\tilde{\chi}_p = 2\overline{\rho D \frac{\partial Z''}{\partial x_i} \frac{\partial Z''}{\partial x_i}} = -2\overline{\rho u_i'' Z''} \frac{\partial \tilde{Z}}{\partial x_i} \tag{2.234}$$

$$\approx 2 \frac{\mu_t}{\mathrm{Sc}_t} \frac{\partial \tilde{Z}}{\partial x_i} \frac{\partial \tilde{Z}}{\partial x_i}, \tag{2.235}$$

where μ_t is the modeled turbulent eddy viscosity and Sc_t is the turbulent Schmidt number.

This results in the final modeled form for the filtered total scalar dissipation rate for LES closures,

$$\tilde{\chi} = \tilde{\chi}_m + \tilde{\chi}_p \tag{2.236}$$

$$\approx \frac{2}{\bar{\rho}} \left(\frac{\mu}{\mathrm{Sc}} + \frac{\mu_t}{\mathrm{Sc}_t} \right) \frac{\partial \tilde{Z}}{\partial x_i} \frac{\partial \tilde{Z}}{\partial x_i}. \tag{2.237}$$

2.9.4. Filtered Mixture Fraction

The primary quantity used to identify the chemical state in Flamelet closure models is the mixture fraction, Z. While there are many different definitions of the mixture fraction that have subtle variations that attempt to capture effects like differential diffusion, they can all be interpreted as a local mass fraction of the chemical elements that originated in the fuel stream. [41] The mixture fraction is a conserved scalar that varies between 0 in the oxidizer stream and 1 in the fuel stream, and is transported in laminar flow by the equation

$$\frac{\partial \rho Z}{\partial t} + \frac{\partial \rho u_i Z}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\rho D \frac{\partial Z}{\partial x_i} \right), \tag{2.238}$$

where D is an effective molecular mass diffusivity.

Applying either temporal Favre filtering for RANS-based treatments or spatial Favre filtering for LES-based treatments yields

$$\frac{\partial \bar{\rho}\tilde{Z}}{\partial t} + \frac{\partial \bar{\rho}\tilde{u}_i \tilde{Z}}{\partial x_i} = -\tau_{Zu_j} + \frac{\partial}{\partial x_i} \left(\bar{\rho} D \frac{\partial \tilde{Z}}{\partial x_i} \right), \tag{2.239}$$

where sub-filter correlations have been neglected in the molecular diffusive flux vector [42] and the turbulent diffusive flux vector is defined as

$$\tau_{Zu_j} \equiv \bar{\rho}\left(\widetilde{Zu_i} - \tilde{Z}\tilde{u}_i\right). \tag{2.240}$$

Similar to species transport, this sub-filter correlation is modeled in both RANS and LES closures with the gradient transport approximation

$$\tau_{Zu_j} \approx -\bar{\rho}D_t\frac{\partial Z}{\partial x_i}, \tag{2.241}$$

where D_t is the turbulent mass diffusivity, modeled as $\bar{\rho}D_t = \mu_t/\mathrm{Sc}_t$ where μ_t is the modeled turbulent viscosity from momentum transport and Sc_t is the turbulent Schmidt number. Please see the Fuego theory manual for further details. The molecular mass diffusivity is then expressed similarly as $\bar{\rho}D = \mu/\mathrm{Sc}$ so that the final modeled form of the filtered mixture fraction transport equation is

$$\frac{\partial\bar{\rho}\tilde{Z}}{\partial t} + \frac{\partial\bar{\rho}\tilde{u}_i\tilde{Z}}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\left(\frac{\mu}{\mathrm{Sc}} + \frac{\mu_t}{\mathrm{Sc}_t}\right)\frac{\partial\tilde{Z}}{\partial x_i}\right]. \tag{2.242}$$

In integral form as used in Fuego, the mixture fraction transport equation is

$$\int\frac{\partial\bar{\rho}\tilde{Z}}{\partial t}\,dV + \int\bar{\rho}\tilde{u}_i\tilde{Z}n_i\,dS = \int\left(\frac{\mu}{\mathrm{Sc}} + \frac{\mu_t}{\mathrm{Sc}_t}\right)\frac{\partial\tilde{Z}}{\partial x_i}n_i\,dS. \tag{2.243}$$

2.9.5. Filtered Scalar Variance

2.9.5.1. RANS Model

For RANS-based turbulence closures, a transport equation is solved for the filtered scalar variance, $\widetilde{Z''^2}$. This equation can be derived by subtracting Equation 2.239 multiplied by $\tilde{Z}$ from the filter of the multiple of Equation 2.238 and Z, yielding

$$\begin{aligned}\frac{\partial\bar{\rho}\widetilde{Z''^2}}{\partial t} + \frac{\partial}{\partial x_i}\left(\bar{\rho}\tilde{u}_i\widetilde{Z''^2}\right) &= -\frac{\partial}{\partial x_i}\left(\overline{\rho u_i''Z''^2}\right) + \frac{\partial}{\partial x_i}\left(\overline{\rho D\frac{\partial Z''^2}{\partial x_i}}\right) + \overline{2Z''^2\frac{\partial}{\partial x_i}\left(\rho D\frac{\partial\tilde{Z}}{\partial x_i}\right)} \\ &\quad -2\overline{\rho u_i''Z''}\frac{\partial\tilde{Z}}{\partial x_i} - \overline{2\rho D\frac{\partial Z''}{\partial x_i}\frac{\partial Z''}{\partial x_i}},\end{aligned} \tag{2.244}$$

where the filtered mixture fraction variance is defined as $\widetilde{Z''^2} \equiv \widetilde{Z^2} - \tilde{Z}^2$.

All five terms on the RHS of Equation 2.244 require closure models. The first term represents turbulent transport of mixture fraction variance, and is modeled by a gradient-transport assumption as

$$-\overline{\rho u_i''Z''^2} \approx \frac{\mu_t}{\mathrm{Sc}_t}\frac{\partial\widetilde{Z''^2}}{\partial x_i}. \tag{2.245}$$

The second and third terms on the RHS of Equation 2.244 taken together represent molecular diffusion of mixture fraction variance, and is typically neglected with respect to turbulent transport for sufficiently high Reynolds numbers. Its effects are included here with another gradient-transport assumption of the form

$$\frac{\partial}{\partial x_i}\left(\overline{\rho D\frac{\partial Z''^2}{\partial x_i}}\right)+\overline{2Z''^2\frac{\partial}{\partial x_i}\left(\rho D\frac{\partial \tilde{Z}}{\partial x_i}\right)}\approx\frac{\partial}{\partial x_i}\left(\frac{\mu}{\mathrm{Sc}}\frac{\partial\widetilde{Z''^2}}{\partial x_i}\right). \tag{2.246}$$

The fourth and fifth terms on the RHS of Equation 2.244 represent production and dissipation of mixture fraction variance, respectively. The production term is similarly modeled with a gradient transport assumption as

$$-2\overline{\rho u_i'' Z''}\frac{\partial\tilde{Z}}{\partial x_i}\approx 2\frac{\mu_t}{\mathrm{Sc}_t}\frac{\partial\tilde{Z}}{\partial x_i}\frac{\partial\tilde{Z}}{\partial x_i}. \tag{2.247}$$

The mixture fraction variance dissipation rate term is equal to the perturbation scalar dissipation rate,

$$\overline{2\rho D\frac{\partial Z''}{\partial x_i}\frac{\partial Z''}{\partial x_i}}=\bar{\rho}\tilde{\chi}_p, \tag{2.248}$$

previously defined in Equation 2.229 and modeled in Equation 2.230. An identical treatment of this term is used here.

The final modeled form of the filtered scalar variance transport equation for RANS turbulence closure models is

$$\frac{\partial\bar{\rho}\widetilde{Z''^2}}{\partial t}+\frac{\partial}{\partial x_i}\left(\bar{\rho}\tilde{u}_i\widetilde{Z''^2}\right)=\frac{\partial}{\partial x_i}\left[\left(\frac{\mu}{\mathrm{Sc}}+\frac{\mu_t}{\mathrm{Sc}_t}\right)\frac{\partial\widetilde{Z''^2}}{\partial x_i}\right]+2\frac{\mu_t}{\mathrm{Sc}_t}\frac{\partial\tilde{Z}}{\partial x_i}\frac{\partial\tilde{Z}}{\partial x_i}-\bar{\rho}\tilde{\chi}_p. \tag{2.249}$$

2.9.5.2. LES Model

For LES turbulence closures, the filtered scalar variance $\widetilde{Z''^2}$ can be modeled with the scaling law [40]

$$\bar{\rho}\widetilde{Z''^2}\approx C_V\bar{\rho}\Delta^2\frac{\partial\tilde{Z}}{\partial x_i}\frac{\partial\tilde{Z}}{\partial x_i}, \tag{2.250}$$

where Δ is a length scale corresponding to the grid filter size and C_V is a model coefficient. For the k^{sgs} closure and the non-dynamic Smagorinsky closure, C_V has a fixed value of 0.5. For the dynamic Smagorinsky LES closure, C_V can be dynamically calculated based on the local instantaneous flowfield.

To dynamically evaluate the filtered scalar variance model coefficient, begin by defining the grid filter-scale correlation

$$\begin{aligned}\tau_{Z''^2} &\equiv \bar{\rho}\widetilde{Z''^2} && (2.251)\\ &= \bar{\rho}\widetilde{Z^2}-\bar{\rho}\tilde{Z}^2 && (2.252)\\ &= \overline{\rho Z^2}-\frac{\left(\overline{\rho Z}\right)^2}{\bar{\rho}}. && (2.253)\end{aligned}$$

Similarly, define an equivalent correlation at a larger test-filter scale

$$T_{Z''^2} \equiv \widehat{\overline{\rho Z^2}} - \frac{\left(\widehat{\overline{\rho Z}}\right)^2}{\hat{\bar{\rho}}}. \tag{2.254}$$

Now, define the quantity $L_{Z''^2}$ as a combination of these two correlations which reduces to an expression that can be evaluated in closed form,

$$\begin{aligned} L_{Z''^2} &\equiv T_{Z''^2} - \widehat{\tau_{Z''^2}} && (2.255)\\ &= \widehat{\bar{\rho}\tilde{Z}^2} - \frac{\left(\widehat{\bar{\rho}\tilde{Z}}\right)^2}{\hat{\bar{\rho}}}. && (2.256) \end{aligned}$$

By modeling the two correlations in Equation 2.255 and equating them to Equation 2.256, the model coefficient C_V can be dynamically evaluated. The correlations at the two filter scales are modeled analogously as

$$\begin{aligned} \tau_{Z''^2} &\approx C_V \bar{\rho}\Delta^2 \left(\frac{\partial \tilde{Z}}{\partial x_i}\right)^2 && (2.257)\\ T_{Z''^2} &\approx C_V \hat{\bar{\rho}}\hat{\Delta}^2 \left[\frac{\partial}{\partial x_i}\left(\frac{\widehat{\bar{\rho}\tilde{Z}}}{\hat{\bar{\rho}}}\right)\right]^2, && (2.258) \end{aligned}$$

where $\hat{\Delta}$ is the characteristic test filter length scale and C_V is assumed to be the same at both scales.

Notice that when the modeled forms of $\tau_{Z''^2}$ and $T_{Z''^2}$ are inserted into Equation 2.255, C_V appears inside a test filtering operation. Formally solving this system of equations for C_V requires the expensive solution of an additional set of coupled integro-differential equations [43]. Alternatively, it is common practice to remove C_V from the test filter with the assumption that it is varying slowly over distances on the order of the test filter size. This greatly simplifies calculations, although it can result in non-physical oscillations in the modeled value for C_V. The square of the error involved in this approximation is $Q = (L_{Z''^2} - C_V M_{Z''^2})^2$, where

$$\begin{aligned} L_{Z''^2} &= \widehat{\bar{\rho}\tilde{Z}^2} - \frac{\left(\widehat{\bar{\rho}\tilde{Z}}\right)^2}{\hat{\bar{\rho}}} && (2.259)\\ M_{Z''^2} &= \hat{\bar{\rho}}\hat{\Delta}^2 \left[\frac{\partial}{\partial x_i}\left(\frac{\widehat{\bar{\rho}\tilde{Z}}}{\hat{\bar{\rho}}}\right)\right]^2 - \widehat{\bar{\rho}\Delta^2 \left(\frac{\partial \tilde{Z}}{\partial x_i}\right)^2}. && (2.260) \end{aligned}$$

Minimizing this error in a least-squares fashion with respect to C_V yields an expression for the modeled coefficient,

$$C_V = \frac{L_{Z''^2} M_{Z''^2}}{M_{Z''^2} M_{Z''^2}}, \tag{2.261}$$

that can be used directly in Equation 2.250 for the filtered scalar variance.

Due to the above simplifications, the model coefficient C_V can sometimes fluctuate wildly, possibly leading to numerical instabilities. A common solution to control these oscillations, and the one that is taken here, is to pass the numerator and denominator of Equation 2.261 through a test filter, yielding

$$C_V = \frac{\widehat{\overline{L_{Z''^2} M_{Z''^2}}}}{\widehat{\overline{M_{Z''^2} M_{Z''^2}}}}. \tag{2.262}$$

This can be crudely justified by recognizing that C_V was already assumed to vary slowly over distances equal to the test filter size, so that this filtering operation is simply enforcing that assumption.

2.9.6. Generating Flamelet Libraries with Spitfire

2.9.6.1. Introduction

Spitfire is a Python-C++ code used to solve low-dimensional combustion problems as a preprocessing step in building tabulated chemistry models for turbulent, sooting fires. In tabulated chemistry models we constrain the thermochemical state space to a manifold over a small number of reaction variables which are often conserved scalars. There are two essential steps in building a tabulated chemistry model. First, solve representative problems such as perfectly stirred reactors or nonpremixed flamelet models, often performing continuation over parameters such as residence times, scalar dissipation rates, or heat loss parameters. Second, use the solution data to build a property reconstruction model with either structured interpolation tables or unstructured regression models (*e.g.*, artificial neural networks). Currently Spitfire solves nonpremixed flamelet equations over parameters of interest and forms the data into a structured table, after which the Tabular Props application is used to build multidimensional B-spline representations of properties which may be queried during a simulation by Fuego.

2.9.6.2. User Interfaces

There are two ways for a user to build a flamelet library with Spitfire. The simplest interface, likely sufficient for most users, works through a simple YAML-style input file, which can be passed to generate any of a set of 'standard' tables. Section 2.9.6.3 shows several examples of the YAML interface.

The second interface, recommended for developers and advanced users, involves the use of Python. At the cost of a little bit of Python programming, this interface offers extensibility, generality, and more precise control than the input file interface. For instance, one can define arbitrarily-complex post processors for tabulated properties, which would enable straightforward parameter studies in soot modeling, for instance. Another examplary use of the Python interface is to generate nonadiabatic flamelet libraries with quasisteady heat loss (a core Spitfire regression test, actually) instead of the transient heat loss allowed by the input file interface. The conda package manager may be used to facilitate installation of Spitfire's external Python and C++ dependencies, as described in the `SpitfireCore` project in the `Spitfire` repository group at `https://cee-gitlab.sandia.gov/spitfirecodes`. Documentation and examples of the Python interface may be found there as well.

2.9.6.3. Using the YAML Interface

On the CEE LAN the YAML interface to Spitfire has been sandboxed into a standalone executable. This removes the need for users to maintain a Python interpreter with Spitfire installed or manage runtime dependencies with paths and environment variables. This reduces the required input down to a chemical mechanism data file in Cantera XML format and an input file for Spitfire. A flamelet library may be generated with the following commands. First, load the `sierra` module to make the Tabular Props application available. Next, run the standalone `generate` executable on the `input.yaml` input file. The `-n 8` argument tells Spitfire to use eight cores wherever possible. The `-t` flag tells Spitfire to run Tabular Props on its output, meaning that after Tabular Props completes a library file will be ready for immediate use with Fuego. Next we discuss the required chemical mechanism data file and show some examples, spanning a simple mixing library to a nonadiabatic, turbulent strained laminar flamelet with soot source terms and radiation sources.

```
$ module load sierra
$ /usr/netpub/mahanse/spitfire_table/generate -i input.yaml -n 8 -t
```

Before showing some examples, we note the "parse-only" mode, which only parses the YAML input file and dumps a comprehensive version with all defaults included. This is helpful when learning how to specify new parameters or examining defaults that are implicitly set.

```
$ /usr/netpub/mahanse/spitfire_table/generate -i input.yaml -p
```

2.9.6.4. Chemical Mechanism Data

Spitfire uses the Cantera code to manage the specification of thermodynamic (species heat capacities), transport (viscosity model parameters) and kinetic rate parameters. This data must be provided in Cantera's XML format. A number of chemical mechanism files are provided in the Spitfire code repositories, and the standalone executable provides several useful mechanism data files listed in Table 2.9-1. Many mechanisms of interest for fire scenarios are available online from various sources, and Cantera provides thorough online resources for building mechanism files from scratch if desired.

Table 2.9-1.. Chemical mechanism data packaged into the standalone Spitfire executable

XML	Fuel	Reference
`heptane_LiuEtAl.xml`	heptane	Liu et al. (2004)
`ethylene_USC.xml`	ethylene	Wang et al. (2007) (USC-mech)
`ethylene_LuoEtAl.xml`	ethylene	Luo et al. (2012) (reduced USC-mech)
`methane_GRI30.xml`	methane	GRI-3.0

2.9.6.5. Simple Mixing Library

The following input is as simple as it gets, which may be used for a mixture of air and n-heptane that is not reacting (*e.g.*, cold flow). First, the `chemistry-type` of the mixture fraction model is specified. This must be one of `unreacted` (pure mixing), `Burke-Schumann` (idealized combustion), `equilibrium` (infinitely fast chemistry), or `SLFM` (strained laminar flamelet library). Next, the Cantera mechanism data file is specified (using the pre-packaged n-heptane mechanism). Following that, the mass fractions and temperature of the oxidizer and fuel streams are given. The mass fractions may be specified in any acceptable format for Cantera, but a comma-separated list as shown here is often the most convenient option.

```
mixture-fraction-model:
  chemistry-type: unreacted
  cantera:
    xml: heptane_LiuEtAl.xml
  oxidizer:
    temperature: 300
    mass-fractions: 'N2:0.767,O2:0.233'
  fuel:
    temperature: 300
    mass-fractions: 'NXC7H16:1'
```

This input will produce a flamelet library with the following properties: temperature, viscosity, density, and isobaric heat capacity of the mixture. By default, Spitfire will write output to the `spitfire_output` directory (set with `output-directory`), all intermediate data will be saved (set with `save-intermediate-data`), and the table for Fuego will be named `table_for_fuego.h5` (set with `fuego-table-file-name`). If defaults are unclear or you want to know what else could be specified, you can run Spitfire in parse-only mode to see a comprehensive summary of the expanded input file with all defaults and options. Another option is to generate a table and check the output directory for the file named `complete_spitfire_input.yaml`, which contains the same information as the parse-only mode produces.

2.9.6.6. Simple Equilibrium Library

There are a number of default parameters left unspecified in this first example. Among the most important are the `pressure` (defaults to 101325 Pa), `heat-loss-type` (defaults to `adiabatic`), and `turbulence-type` (defaults to `laminar`). For instance, to generate the data for an equilibrium chemistry model at two atmospheres, one could change the `chemistry-type` and directly set the pressure. In addition to the properties that are always tabulated, it may be interesting to visualize mass fractions of certain species in the Fuego simulation. They may be added as shown below.

For equilibrium chemistry it may be important to set details of the grid in the mixture fraction dimension. In the adjacent example the number of grid points is set to 96 (overwriting the default of 32) and details of the automatic grid clustering are specified. Spitfire uses a grid clustered around the stoichiometric mixture fraction by default (leave `grid-cluster-point` unspecified or specify `stoichiometric` to accomplish this). The default value of the `grid-cluster-intensity` is 4, and increasing this number leads to a more tightly clustered grid. In cases where a uniform grid is preferred, simply set the `grid-type` to `uniform`.

```
mixture-fraction-model:
  chemistry-type: equilibrium
  pressure: 202650
  cantera:
    xml: heptane_LiuEtAl.xml
  oxidizer:
    temperature: 300
    mass-fractions: 'N2:0.767,O2:0.233'
  fuel:
    temperature: 485
    mass-fractions: 'NXC7H16:1'
  tabulated-quantities:
    mass-fractions:
      - OH
      - C2H2
```

```
mixture-fraction-model:
  chemistry-type: equilibrium
  pressure: 202650
  cantera:
    xml: heptane_LiuEtAl.xml
  oxidizer:
    temperature: 300
    mass-fractions: 'N2:0.767,O2:0.233'
  fuel:
    temperature: 485
    mass-fractions: 'NXC7H16:1'
  tabulated-quantities:
    mass-fractions:
      - OH
      - C2H2
  mixture-fraction:
    grid-points: 96
    grid-type: clustered
    grid-cluster-intensity: 6
    grid-cluster-point: 0.1
```

2.9.6.7. Laminar, Adiabatic SLFM Library

To generate a strained laminar flamelet model, the only required change from the simpler examples is that the chemistry type be set to SLFM. However, we often prefer to specify the number and distribution of grid points in the mixture fraction and dissipation rate dimensions. For more information, run Spitfire in parse-only mode with -p to see an expanded input file.

```
mixture-fraction-model:
    chemistry-type: SLFM
    heat-loss-type: adiabatic
    turbulence-type: laminar
    cantera:
        xml: heptane_LiuEtAl.xml
    oxidizer:
        temperature: 300
        mass-fractions: 'N2:0.767,O2:0.233'
    fuel:
        temperature: 485
        mass-fractions: 'NXC7H16:1'
    mixture-fraction:
        grid-points: 96
    scalar-dissipation-rate:
        grid-points: 36
```

2.9.6.8. Turbulent, Nonadiabatic SLFM Library

In this final demonstration we incorporate heat loss, radiation and soot source terms, and a presumed PDF model for turbulence-chemistry interaction. For more information on parameter settings, run Spitfire in parse-only mode with -p to see an expanded input file.

```
mixture-fraction-model:
    chemistry-type: SLFM
    heat-loss-type: nonadiabatic-defect
    turbulence-type: turbulent
    presumed-PDF:
        PDF-type: ClipGauss
        integrator-type: GaussKronrod
        scaled-scalar-variance:
            grid-points: 32
        mean-mixture-fraction:
            grid-points: 32
        mean-scalar-dissipation-rate:
            grid-points: 32
        mean-stoichiometric-enthalpy-defect:
            grid-points: 32
        minimize-convolution-time: False
    cantera:
        xml: heptane_LiuEtAl.xml
    oxidizer:
        temperature: 300
        mass-fractions: 'N2:0.767,O2:0.233'
    fuel:
        temperature: 485
        mass-fractions: 'NXC7H16:1'
    mixture-fraction:
        grid-points: 96
    scalar-dissipation-rate:
        grid-points: 36
        min: 1.e-3
        max: 1.e2
        reference: stoichiometric
    stoichiometric-enthalpy-defect:
        grid-points: 36
    tabulated-quantities:
        include-radiation: True
        soot-model: Aksit-Moss
```

2.10. TURBULENT REACTING MIXING MODELS

In a reacting flow it is sometimes necessary to limit the finite-rate chemistry reaction when it occurs in regions of high turbulence. In this case, turbulent mixing of the reactants limits the finite-rate (laminar) chemical reaction, the time scale of which can be calculated as

$$t_{\mathrm{c}} = N_{\mathrm{R}} \frac{\sum_{i=1}^{N_{\mathrm{S}}} Y_i}{\sum_{i=1}^{N_{\mathrm{S}}} \frac{\mathrm{dY_i}}{\mathrm{dt}}} \tag{2.263}$$

where N_{R} is the number of reactions, N_{S} is the number of species in the reaction, and Y_i represents the concentration of species i. The denominator in Equation 2.263 is the production rate of species i.

2.10.1. Modified EDC Model (PARENTE)

One such model, named here as the PARENTE model, is adopted from [44] which derives explicit dependencies between the EDC model coefficients and the turbulent Reynolds ($\mathrm{Re_t}$) and Damköhler (Da) dimensionless numbers. This model is based on EDC and is used to modifty the chemistry reactions prescribed in the gas phase. The PARENTE turbulent reacting mixing model can can be used

in addition to or in lieu of the built-in EDC turbulent combustion model found in Section 2.8. The current model is described as

$$\mathrm{Da} = \max\left(\min\left(\frac{1}{t_\mathrm{c}}\sqrt{\frac{\nu}{\epsilon}}, 10\right), 1\times10^{-10}\right) \tag{2.264}$$

$$\mathrm{Re_t} = \frac{k^2}{\nu\epsilon} \tag{2.265}$$

$$C_\tau = \min\left(\frac{C_1}{\sqrt{\mathrm{Da}(\mathrm{Re_t}+1)}}, 2.1377\right) \tag{2.266}$$

$$C_\gamma = \max\left(\min\left(C_2\sqrt{\mathrm{Da}(\mathrm{Re_t}+1)}, 5.0\right), 0.4082\right) \tag{2.267}$$

$$\gamma_\mathrm{L} = C_\gamma\left(\frac{\nu\epsilon}{k^2}\right)^{\frac{1}{4}} \tag{2.268}$$

$$\tau^* = C_\tau\sqrt{\frac{\nu}{\epsilon}} \tag{2.269}$$

where k represents the turbulent kinetic energy, ϵ is the turbulent dissipation and ν is the kinematic viscosity of the gas. The constants C_1 and C_2 can be specified by the user, but default to $C_1 = 0.05774$ and $C_2 = 0.5$. The modified reaction time scale, τ^* is used to solve the reaction equation from time t^n to $t^{n+\tau^*}$. Following this calculation, the reaction source terms for species (S^*_{Yi}) and enthalpy (S^*_H) can be computed. These computed sources are further scaled via the multiplication of a coefficient κ which is calculated as

$$\kappa = \begin{cases} 1, & \text{if } \gamma_\mathrm{L} \geq 1 \\ \max\left(\min\left(\frac{\gamma_\mathrm{L}^{e_1}}{1-\gamma_\mathrm{L}^{e_2}}, 1\right), 0\right), & \text{otherwise} \end{cases}$$

where $e_1 = e_2 = 3.0$ in our implementation of the model. Following the evaluation of κ, the final source terms for species and enthalpy can be calculated as

$$S_{Y_i} = \kappa S^*_{Y_i} \tag{2.270}$$

$$S_H = \kappa S^*_H \tag{2.271}$$

respectively.

2.11. VOLUME OF FLUID MODEL

Beta Capability: The volume-of-fluid capability is a beta feature. The numerics for laminar isothermal flow are well tested, but advanced features such as non-isothermal reacting and evaporating simulations are more experimental.

The volume-of-fluid (VOF) capability allows simulation of two-phase systems. The phases are tracked with a conserved scalar, α, which is the volume fraction of fluid in a given control volume. This scalar is 1 in the liquid, 0 in the gas, and between 0 and 1 in the transition region.

There are a number of numerical challenges associated with multi-phase modeling using a diffuse interface. Many of these challenges exist regardless of whether one uses a level set or a volume-of-fluid approach, or some hybrid of the two.

2.11.1. Governing Equation

The basic advection equation for the volume fraction (α) in VOF equations is

$$\frac{\partial \alpha}{\partial t} + \vec{u} \cdot \nabla \alpha = S_\alpha \tag{2.272}$$

Expanding the conservative form of the convection term with the chain rule as

$$\nabla \cdot (\vec{u}\alpha) = \vec{u} \cdot \nabla \alpha + \alpha \left(\nabla \cdot \vec{u} \right) \tag{2.273}$$

we can express the governing equation in corrected conservative form as

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\vec{u}\alpha) = \alpha \left(\nabla \cdot \vec{u} \right) + S_\alpha \tag{2.274}$$

where the right hand side term is 0 for an incompressible flow with no phase change. The standard advection operator is typically too diffusive for practical use, so one of two approaches are typically used:

1. A geometric advection operator [45, 46, 47, 48]
2. An additional interface compression term [49, 50]

Several varieties of geometric advection operators have been proposed in the literature [45, 46, 47, 48], many of which are only applicable for 2D quadrilateral meshes. Those that can be extended to higher dimensions often do so at considerable complexity. For a more detailed discussion of the different options, see [51]. The form implemented here uses the interface compression approach to modify the governing equation to be

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\vec{u}\alpha) + \nabla \cdot (\vec{u_c}\alpha(1-\alpha)) = \alpha \left(\nabla \cdot \vec{u} \right) + S_\alpha \tag{2.275}$$

where $\vec{u_c}$ is the compressive velocity. A common form for this is

$$\vec{u_c} = C_\alpha |\vec{u}| \vec{n} \tag{2.276}$$

where C_α is a constant and $\vec{n}$ is the normal vector at the phase interface.

When phase change is present, the source term S_α is non-zero. The volume source of liquid due to evaporation is

$$S_\alpha = -\frac{\dot{m}_{evap}}{\rho_L} \tag{2.277}$$

The evaporation term is calculated using the method described by Hardt and Wondra [52] which converts an evaporative mass flux to a volume source using the interphase area, A, and the mass flux predicted by the evaporation model, m''_{evap}, as

$$\dot{m}_{evap} = m''_{evap} A \tag{2.278}$$

$$A = W|\nabla\alpha|\frac{\int |\nabla\alpha| dV}{\int W|\nabla\alpha| dV} \tag{2.279}$$

$$W = \alpha \tag{2.280}$$

One variation of this that we explored is $W = (1-\alpha)^2$, which is well suited for a diffuse interface approach. We find that the form by Hardt and Wondra is sometimes destabilizing to the diffuse interface and leads to an artificially high surface regression rate. For a very simple 1D problem or a sharp interface method, the $W = (1-\alpha)^2$ weighting scheme places all the evaporation outside the liquid and does not correctly predict the surface regression rate. However, it can be better for a diffuse interface representation since evaporation acts as a stabilizing liquid sink to counteract interface diffusion. When using a sharp interface method or a 1D problem where there is no interface diffusion, the weighting of Hardt and Wondra ($W = \alpha$) should be used.

Different evaporation models can be used to define m''_{evap}. The model used by Hardt and Wondra uses deviation from the saturation temperature to define the rate as

$$m''_{evap} = K(T - T_{sat}) \tag{2.281}$$

$$K = \frac{2\beta}{2-\beta}\frac{h_{fg}}{\sqrt{2\pi R}}\frac{\rho_g}{T_{sat}^{1.5}} \tag{2.282}$$

This form is only physically applicable to either fully saturated gasses, or interface temperatures above the boiling point since it does not take into account the gas composition. As such, this model only allows positive evaporation rates (so no condensation) in order to produce realistic behavior during heat-up and away from the hot regions of the simulation. When using a multi-species gas, the more complete Hertz-Knudsen-Langmuir model can be used, where the rate is driven by the difference between the partial pressure of the vapor and the saturation pressure. This model will be added to the available evaporation models once the multi-species gas treatment is complete.

2.11.2. Interface Reconstruction

When the phase interface is represented by a volume-of-fluid variable, numerical diffusion in the normal advection operator causes the interface (cells with volumes of fluid between 0 and 1) to become wider over time. Although the total volume of liquid is conserved, the volume inside the $\alpha = 0.5$ isosurface may change as a result, and the effect of the higher density liquid will be felt well outside the correct liquid interface location. Techniques to counter-act this diffusion in volume-of-fluid include using compressive advection operators in the VOF equation, using a separate anti-diffusive sharpening step, using a geometric reconstruction of the interface to define the advection, and using sharpening and liquid relocation schemes.

With a level set approach, the interface location is defined by the $\phi = 0$ contour. By definition, the interface does not become diffuse, but there are no guarantees of global conservation of liquid either. Techniques to address this typically involve doing a volume-conserving redistancing operation to the level set field. Such operations can preserve the total volume of liquid, but may not preserve the local shape of the liquid-gas interface.

In both VOF and level set approaches, calculation of the interface curvature is required to determine surface tension forces. This is done by taking the divergence of the interface normal vector ($\vec{n}$).

$$\kappa = -\nabla \cdot \vec{n} \tag{2.283}$$

It should be immediately apparent that κ will contain not only discretization error from the divergence operator, but also discretization error from the calculation of the normal vector, which is itself a gradient of some other quantity (call it α_s here, it could be the level set field, the volume of fluid field, or a smoothed version of either).

$$\vec{n} = \frac{\nabla \alpha_s}{|\nabla \alpha_s| + \epsilon} \tag{2.284}$$

The simple approach to use $\alpha_s = \alpha$ results in considerable noise in the interface normal vector calculation, particularly in regions of high curvature relative to mesh sizes. In the coupled level set VOF approach (CLSVOF), α_s is the companion level set distance function that is either advected along with the volume fraction or redistanced from the volume fraction contour at 0.5 (so $\alpha_s = \phi$).

In Fuego we have implemented both a CLSVOF approach and a diffusive approach, where α_s is a diffusively smoothed version of the volume fraction field, α, using a user-specified Fourier number (Fo) and number of iterations.

$$\alpha_s^0 = \alpha \tag{2.285}$$

$$\alpha_s^{n+1} = \alpha_s^n + Fo\Delta x^2 \nabla^2 \alpha_s^n \tag{2.286}$$

In addition, recall that the analytical curvature for a spherical drop is a function of radius: $\kappa = 2/r$. This means that if κ is calculated in cells that span an interface of finite, non-zero width, there will also be a spatial variation of κ across that interface. This means that although in reality the surface tension force is applied sharply at the interface, in a diffuse interface scheme it will be applied over a finite interface thickness of several cells.

This noise and spatial variation in the curvature translates directly into noise in the surface tension force and is the sole cause of the so-called "parasitic currents" that plague diffuse interface methods. Many authors have demonstrated that when using a uniform, analytically prescribed curvature the parasitic currents in the classical static drop test drop to machine precision levels if the forces are otherwise properly balanced.

There are various techniques for improving the curvature calculation. Calculation of the normal vector when the VOF field is sharp is very noisy, so VOF schemes often use a smoothed field or higher order least squares technique. Coupled level set VOF (CLSVOF) carries a level set field along to use for calculating this normal vector on a smooth field (the level set field).

Finally, once the interface has been advected and sharpened, and the curvature has been properly calculated, the surface tension force must be integrated into the momentum and continuity equations in such a manner as to not induce spurious velocities. This is known as the "balanced force" technique, and involves including a surface tension force vector consistently with the pressure gradient.

2.11.3. Continuity Equation

In order to handle the high density ratios in multiphase simulations, the form of the continuity equation is modified by dividing both sides of the equation by density. Interpolation of nodal quantities to subcontrol surfaces is an important consideration to note. Quantities interpolated to faces are indicated with an f subscript in the following equations, while nodal quantities are given no subscript.

The regular form of the pressure Poisson equation in discrete form is

$$\sum_f \left(\tau_f \nabla p_f \cdot A_f\right) = \sum_f \left(\left(\rho \vec{u}\right)_f \cdot A_f + \left(\tau \nabla p\right)_f \cdot A_f\right) \tag{2.287}$$

which includes the projected nodal gradient of pressure ∇p. In the prior form, the projected nodal gradient ∇p was calculated using the divergence theorem as

$$\nabla p = \frac{1}{V} \sum_f p_f A_f \tag{2.288}$$

For the VOF implementation we use the method described by Francois et al. [53] to calculate a projected nodal density-scaled force term (F) based on calculation of forces at faces, which includes both pressure gradient and surface tension forces. The balanced force term is

$$\vec{F} = \nabla p - \kappa \sigma \nabla \alpha \tag{2.289}$$

and the projection from subcontrol faces to nodes is done using the method by Francois as

$$F_j = \frac{\sum_f \left(\nabla p_{f,j} - (\sigma\kappa)_f \nabla \alpha_{f,j}\right)/\rho_f}{\sum_f |A_{f,j}|} \tag{2.290}$$

This term is updated after the solution to the pressure equation, when updated values for pressure and volume-of-fluid have both been calculated in order to keep them synchronized. This nodal force term is included as a source term in the momentum equation (negative, scaled by volume, and multiplied by nodal density) and can be added to the 4th order pressure projection scheme in discrete form as

$$\sum_f \left(\left(\frac{\tau}{\rho}\right)_f \nabla p_f \cdot A_f \right) = \sum_f \left(\vec{u}_f \cdot A_f + \left(\tau \vec{F}\right)_f \cdot A_f + \left(\frac{\tau}{\rho}\right)_f (\sigma\kappa)_f \nabla \alpha_f \cdot A_f \right) \tag{2.291}$$

where $\vec{F}$ is the projected nodal density-weighted balanced force, κ is the curvature, and σ is the surface tension.

The advantage of this projection scheme over a simpler approach where one might define the nodal force term (as done by Lin et al. [54]) as

$$F = \frac{1}{V}\left(\sum_f p_f A_f - \sigma\kappa \sum_f \alpha_f A_f \right) \tag{2.292}$$

is that in such a scheme the evaluation of curvature (κ) occurs at a different location than the gradient of the volume-of-fluid term ($\nabla\alpha$). In the presence of gradients in κ–which are guaranteed to occur in real problems–this introduces a force imbalance that can cause significant spurious non-physical currents in the fluid. Spatial inaccuracies in curvature are the primary cause of spurious currents in VOF simulations – and a baseline test for any balanced force implementation is to see how a static drop with a prescribed (not calculated) value for curvature behaves. In a properly balanced force implementation the resulting parasitic currents should be basically zero (on the order of machine precision). However, a weakness of relying solely on a prescribed curvature test is that it fails to show the force imbalance that arises in the projected nodal force calculation due to interpolation of the curvature. The form published by Francois et al. [53] and implemented in Sierra/Fuego not only recovers machine-precision parasitic currents with the prescribed curvature test, but also keeps them acceptably low when using calculated curvatures.

In addition, when there is phase change there is an additional source term on the right hand side of this equation, which is

$$S = \dot{m}_{evap}\left(\frac{1}{\rho_g} - \frac{1}{\rho_L}\right) \tag{2.293}$$

2.11.4. Property Evaluation

When using VOF, you are required to provide three material models. There is one model for the liquid, one for the gas, and one top level model. The top level model simply specifies the names for the materials to use for gas and liquid, as well as specifying properties like surface tension. Properties needed for flow calculations are combined from the liquid and gas properties using the following rules

A simple averaging is used for non-specific quantities,

$$\phi = \alpha\phi_L + (1-\alpha)\phi_g \tag{2.294}$$

where ϕ is thermal conductivity (k), density (ρ), viscosity (μ), absorption (a), or density-pressure-derivative ($\frac{\partial\rho}{\partial p}$).

For specific quantities, such as specific heat (C_p) and enthalpy (h), a density-weighted average is used

$$\phi = \frac{\alpha\rho_L\phi_L + (1-\alpha)\rho_g\phi_g}{\rho} \tag{2.295}$$

For properties that are only relevant for the gas phase, the property value is simply copied from the gas phase

$$\phi = \phi_g \tag{2.296}$$

Examples include species enthalpy, mass diffusivity, and molecular weight.

Beta Capability:

2.12. ONE-DIMENSIONAL COMPOSITE FIRE BOUNDARY CONDITION

2.12.1. Conceptual Overview

Fuego includes a boundary condition that is capable of modeling the thermal decomposition and outgassing of a thin sheet of porous material at the boundary surface, initially intended to simulate the combustion of a sheet of carbon fiber composite material. Variation through the material thickness is assumed to be locally one-dimensional. The actual implementation is quite flexible, allowing the simulation of the thermal response of essentially any finite-thickness material that can optionally undergo a user-specified chemical decomposition mechanism.

Figure 2.12-1 illustrates a two-dimensional representation of the virtual mesh used for this 1D composite fire boundary condition. One layer of elements above the boundary is shown, within which Fuego performs its normal fluid solve using the control volume finite element CVFEM method. The CVFEM sub-control volumes are demarcated with dashed lines. An equal-order interpolation methodology is used, so that all solution variables are stored at the element vertices.

For this boundary condition, a series of independent one-dimensional virtual domains exist behind each CVFEM surface node, and each virtual 1D domain has a cross-sectional area that matches the group of CVFEM boundary sub-control surfaces that contain the single "parent" surface node. A classical cell-centered finite volume methodology is used for the 1D virtual domains, where the discretization, storage, and numerical solutions all occur within the boundary condition implementation and only interact with the main CVFEM flow solution through fluxes and solution variables at the exposed surface.

Each 1D domain is assumed to have a fixed geometry that is filled with a simple porous material that is allowed to react chemically to form gaseous species. Since the overall volume of each element is fixed, the porosity of each volume is assumed to increase as species are converted from solid to gas. It is assumed that the gaseous species within the pores of the solid phase are of secondary concern, and as such no discrete transport equation is solved for them. The approximation is instead made that all gases generated within the porous material appear instantaneously at the surface of the material as a flux into the main fluid solution. It would be straightforward to solve additional transport equations for fluid flow within the porous material if that level of fidelity were to become necessary, as in the case of oxidative reactions where oxygen must diffuse through the exposed surface into the porous material before reactions may occur.

2.12.2. Model Formulation

2.12.2.1. Transport Equations

Within the solid phase of the porous material, one-dimensional transport equations for continuity, chemical species, and energy are solved in the form:

$$\frac{\partial \bar{\rho}}{\partial t} = \dot{\omega}_c''' \tag{2.297}$$

$$\frac{\partial \bar{\rho} Y_k}{\partial t} = \dot{\omega}_k''' \tag{2.298}$$

$$\bar{\rho}\bar{c}\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\bar{k}\frac{\partial T}{\partial x}\right) + \dot{q}''', \tag{2.299}$$

where $\bar{\rho}$, $\bar{c}$, and $\bar{k}$ are the mixture-averaged bulk density, specific heat, and thermal conductivity, respectively, Y_k is the mass fraction of chemical species k, T is the temperature of the solid phase, $\dot{q}'''$ is the volumetric heat generation rate due to chemical reactions, $\dot{\omega}_k'''$ is the volumetric mass generation rate of chemical species k, and $\dot{\omega}_c'''$ is the overall mass generation rate computed as $\dot{\omega}_c''' = \sum_k \dot{\omega}_k'''$.

2.12.2.2. Material Models

The composite material used for this boundary condition is assumed to be of a fixed volume, *i.e.* there is no structural deformation allowed. The bulk density of the multi-species solid mixture is assumed to be

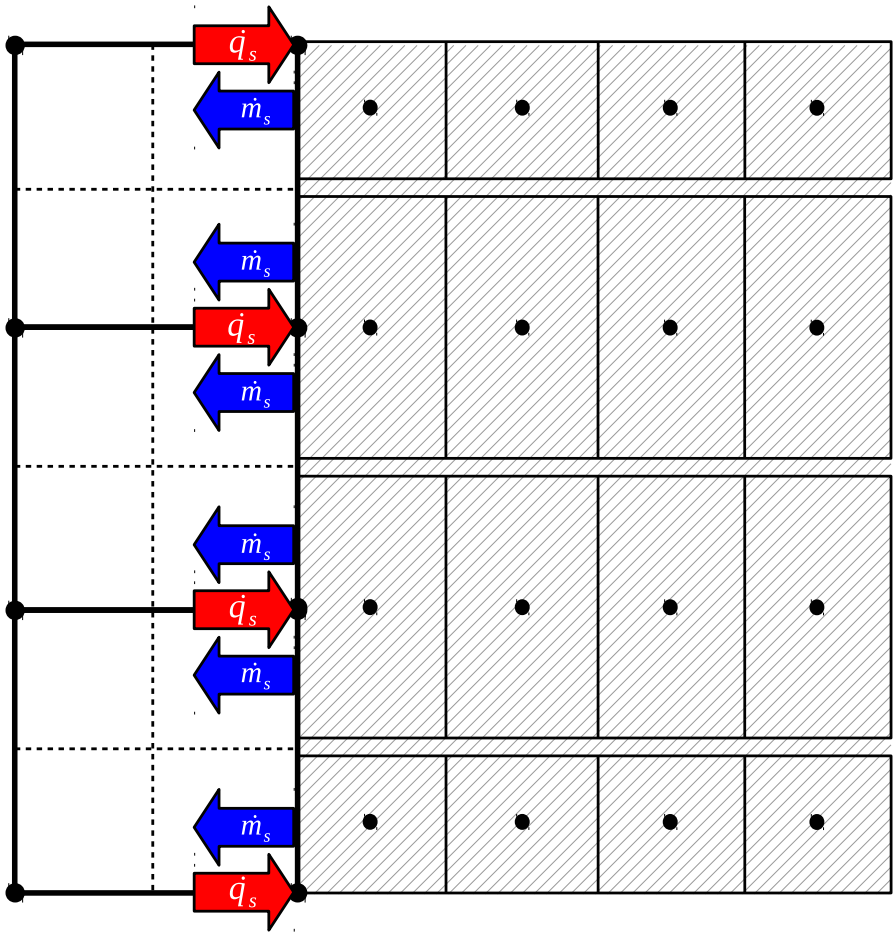


Figure 2.12-1.. Representative mesh layout for 1-D composite fire boundary condition

a function of the density of each component species in their native porous state, as

$$\bar{\rho} = \left(\sum_k \frac{Y_k}{\rho_k} \right)^{-1}, \tag{2.300}$$

where ρ_k is the porous density of species k, provided as a material model by the user. This model for the mixture bulk density is only used to compute the initial bulk density field, which is subsequently solved directly from Equation 2.297.

The porosity of the mixture is assumed to follow the model

$$\bar{\psi} = \sum_k X_k \psi_k, \tag{2.301}$$

where X_k is the volume fraction of species k,

$$X_k = \bar{\rho} \frac{Y_k}{\rho_k}, \tag{2.302}$$

and ψ_k is the porosity of pure species k, modeled as

$$\psi_k = 1 - \frac{\rho_i}{\rho_{s0,k}}, \tag{2.303}$$

where $\rho_{s0,k}$ is the density of the solid (non-porous) species k at a reference temperature. Note that the porosity does not appear explicitly in any of the transport equations or subsequent material models, so that it is never computed as part of the boundary condition solution. It would only appear in transport equations for the gaseous species occupying the pores of the solid skeleton, if this level of detail were ever to be added to this model.

In their most detailed form, the bulk thermal conductivity and specific heat are evaluated as a volume average and mass average of the individual species properties, respectively, as

$$\bar{k} = \sum_k X_k k_k \tag{2.304}$$

$$\bar{c} = \sum_k Y_k c_k, \tag{2.305}$$

although a species-independent model for the overall bulk property may be used if the individual species properties are not known.

The last quantities that require a model are the volumetric species mass production rates, $\dot{\omega}'''$, and the volumetric heat production rate, $\dot{q}'''$. These quantities can be provided by the user in two different ways. The traditional approach is to supply them using standard material property evaluations as a part of the material model definition. These are arbitrary functions that themselves may be dependent on any of the solution variables or other material properties. If a nonreacting material is desired, then these terms may be simply modeled as zero.

The second way of supplying these quantities is by including a chemistry description block in the material model, which allows the user to specify multiple reactions and variable composition gas production.

2.12.2.3. Boundary Conditions

The exposed surface of the composite material interacts thermally with the environment through several mechanisms, including convective heat transfer and both radiation absorption and emission. These external fluxes must balance the conduction inside the composite material at the surface, as

$$\begin{aligned} \dot{q}''' &= \dot{q}'''_{\text{conv}} + \dot{q}'''_{\text{rad}} \\ &= \dot{q}'''_{\text{conv}} + \epsilon \left(\sigma T_1^4 - \dot{q}'''_{\text{irr}} \right) \end{aligned} \tag{2.306}$$

where $\dot{q}'''_{\text{conv}}$ is the convective flux imposed on the surface by the external laminar or turbulent boundary condition treatment, T_1 is the temperature solution from the first control volume in the composite material used to model the gray emission, and $\dot{q}'''_{\text{irr}}$ is the external radiative flux incident on the surface.

On the back-side of the virtual composite material, optional convective and radiative heat transfer to a quiescent environment is modeled as

$$\begin{aligned} \dot{q}'''_{\mathrm{b}} &= \dot{q}'''_{\mathrm{b,conv}} + \dot{q}'''_{\mathrm{b,rad}} \\ &= h_c \left(T_N - T_{\mathrm{ref}}\right) + \sigma \epsilon_b \left(T_N^4 - T_{\mathrm{ref}}^4\right) \end{aligned} \tag{2.307}$$

where h_c is a user-specified convection coefficient, ϵ_b is a user-specified back-side emissivity, T_{ref} is the modeled ambient environment temperature, and T_N is the temperature of the solution node closest to the back-side surface, assumed to be equal to the back-side surface temperature itself.

Please see the Fuego theory manual for details on the numerical implementation of this boundary condition.

2.12.3. Model Usage

This model is used like any other Fuego boundary condition, by filling in a `composite boundary condition` block in the input deck for each target sideset. A typical example would be:

```
BEGIN composite boundary condition ON SURFACE surface_1
  composite material = composite

  composite nodes = 10
  composite min_nonlinear_iterations = 1
  composite max_nonlinear_iterations = 10
  composite residual_tolerance = 1.e-4

  composite thickness = 0.002  # 2 mm

  activate backside convection
  activate backside radiation
  composite backside_convection_coefficient = 10.0
  composite backside_reference_temperature = 300.

  composite initial_temperature = 300.
  composite initial_mass_fraction Carbon = 0.6
  composite initial_mass_fraction Epoxy  = 0.4
  composite initial_mass_fraction Char   = 0.0

  # Composition of outgassed mixture
  mass_fraction CH4 = 1.0
  mass_fraction CO  = 0.0
  mass_fraction CO2 = 0.0
  mass_fraction O2  = 0.0
  mass_fraction H2O = 0.0
  mass_fraction H2  = 0.0
  mass_fraction N2  = 0.0
  turbulent_kinetic_energy = 2.5e-5
  soot_mass_fraction = 0.0
  soot_nuclei_mass_fraction = 0.0
END
```

When using a chemistry description to define the composite, the nonlinear iteration controlls and tolerance options are ignored, and the mass fractions on the boundary should not be provided. They will be calculated as time-varying quantities that depend on the composition of the gas produced.

The line commands for this boundary condition are fairly verbose and self-explanatory, although the salient details will still be described here. In this example, there are 10 nodes (10 control volumes) to be spread evenly through the composite thickness of $2\,mm$ (assuming that the simulation is configured to use the MKS unit system). The nonlinear iteration and residual tolerance commands define the iterative implicit solution behavior, where the solution will be deemed converged when the residual tolerance is met everywhere or when the maximum number of nonlinear iterations is reached, whichever occurs first.

The material to be used for the composite must be specified by name (to be described below), and the initial temperature and mass fractions for species consistent with the material definition are specified. Here, the initial conditions are provided as constant values, although spatially-varying values through the use of a function or user subroutine may also be provided. Note that, even when applying a spatially-varying initial condition, the initial state will always be uniform through the thickness of the composite at every location.

The optional back-side convective and radiative heat transfer modes are activated in this example, with a convection coefficient and ambient reference temperature provided. Note that the same surface emissivity model, defined in the material property block, is used for both the front and back of the composite material. If a model is used for emissivity that is a function of the solution temperature, mass fractions, or other material properties, then a value computed independently from the local solution state is used for both the front and back surfaces. If neither the convective nor the radiative heat transfer models are activated, then the back surface is assumed to be adiabatic.

Finally, the composition and properties of any outgassed mixture must be provided. The outgassed species must be defined from the set of species supported by the material model used for the main flow solution adjacent to the boundary condition, although the specific mass fractions are completely arbitrary. When a mass loss is predicted for the solid phase of the porous material, the equivalent mass of this gas mixture will be produced at the surface.

Below is an example material property definition block for a composite material:

```
BEGIN PROPERTY SPECIFICATION FOR FUEGO MATERIAL composite
  REFERENCE temperature = 300.
  REFERENCE mass_fraction Carbon = 0.6
  REFERENCE mass_fraction Epoxy  = 0.4
  REFERENCE mass_fraction Char   = 0.0
  # Note: Keep above species order in sync with production rate

  species_density Carbon = 1779.  # kg/m^3
  species_density Epoxy  = 1300.  # kg/m^3
  species_density Char   = 650.   # kg/m^3

  SUBROUTINE FOR density = density_inverse_mass_average
  INPUT VARIABLES FOR density_inverse_mass_average = species_density mass_fraction

  # Target model: k = 0.023 * T(C)^0.46  (fig. 3 of DOT/FAA/AR-07/57)
  # functional form: k = a + b * ((T-To)/Tref)^r
  SUBROUTINE FOR thermal_conductivity = thermal_conductivity_T_power
  INPUT VARIABLES FOR thermal_conductivity_T_power = temperature
  REAL DATA FOR thermal_conductivity_T_power = 0.0 0.023 273.15 1.0 0.46  # a b To Tref r

  # Target model: c = 750 + 4.1 * T(C)  (fig. 4 of DOT/FAA/AR-07/57)
  # functional form: c = a + b * ((T-To)/Tref)^r
  SUBROUTINE FOR specific_heat = specific_heat_T_power
  INPUT VARIABLES FOR specific_heat_T_power = temperature
```

```
  REAL DATA FOR specific_heat_T_power = 750 4.1 273.15 1.0 1.0  # a b To Tref r

  emissivity = 0.95

  # Volumetric species mass production rates for reaction:
  #  Epoxy --> nu * Char + (1-nu) * Pyrolysate
  #  (Note: Specify index (Fortran style) for reactant and product species below)
  SUBROUTINE FOR species_production_rate = spec_rate_2spec_1step
  INPUT VARIABLES FOR spec_rate_2spec_1step = temperature density species_density mass_fraction
  REAL DATA FOR spec_rate_2spec_1step = 3.33e15 2.2614e5 8.314 2 3 #  A1     E1       R      idxR  idxP
                                                                   # [1/s] [J/mol] [J/mol-k]

  # Set heat production rate to zero, for a neutral reaction
  SUBROUTINE FOR heat_production_rate = heat_rate_1step
  INPUT VARIABLES FOR heat_rate_1step = species_production_rate
  REAL DATA FOR heat_rate_1step = 0.0 2  # H1  idxR
END
```

One thing of note is that this material makes heavy use of Fuego's user subroutine functionality although function and constant properties are also options, where applicable. Completely arbitrary user-defined models are possible by providing your own user subroutines, although a standard set of essential models are already built into the code and are demonstrated here.

The first thing in this material block is the reference property specification, which is just used in this context to provide diagnostic material property output at code startup and to set how many chemical species there are, what their names are, and their ordering.

The first property specified here is the `species_density`, ρ_k, which is then used by the model for `density`, $\bar{\rho}$ to compute the bulk density initial condition. Note that this property uses the `density_inverse_mass_average` model, which is one of the built-in user subroutines that implements Equation 2.300.

The next two properties specified in this example are the bulk thermal conductivity, $\bar{k}$, and the bulk specific heat, $\bar{c}$. Both of these properties use a built-in temperature power-law model that is intended to give a great deal of flexibility, and is defined as

$$\bar{k} = a + b\left(\frac{T - T_o}{T_{\text{ref}}}\right)^r \tag{2.308}$$

$$\bar{c} = a + b\left(\frac{T - T_o}{T_{\text{ref}}}\right)^r, \tag{2.309}$$

where a, b, T_o, T_{ref}, and r are arbitrary constants that are passed in as constant parameters to the model.

Note that, if species-dependent properties are known for either the thermal conductivity or specific heat, then the built-in models for Equations 2.304 and 2.305 may be used in conjunction with species-specific models for thermal conductivity and specific heat. An example of what this would look like for constant species-specific properties (although arbitrarily-complex models for these may be used as well) is:

```
species_thermal_conductivity Carbon = 0.45  # Made-up values
species_thermal_conductivity Epoxy  = 0.10
species_thermal_conductivity Char   = 0.05

SUBROUTINE FOR thermal_conductivity = thermal_conductivity_mixture_average
INPUT VARIABLES FOR thermal_conductivity_mixture_average = species_thermal_conductivity mass_fraction
```

for thermal conductivity, and:

```
species_specific_heat Carbon = 3000.  # Made-up values
species_specific_heat Epoxy  = 500.
species_specific_heat Char   = 300.

SUBROUTINE FOR specific_heat = specific_heat_mixture_average
INPUT VARIABLES FOR specific_heat_mixture_average = species_specific_heat mass_fraction
```

for specific heat. The emissivity is specified next in this example as a constant value, although arbitrarily-complex functions are also possible.

The last properties defined in this example are the volumetric species production rate, $\dot{\omega}_k'''$, and the volumetric heat production rate, $\dot{q}'''$. These may also be arbitrarily-complex user-defined subroutines, although they are shown here using simple built-in subroutines. The species production rate model implements a simple one-step reaction where one solid species decomposes into another solid species plus a gas mixture, as

$$R \rightarrow \nu P + (1-\nu)\mathrm{Gas}, \tag{2.310}$$

where R is the reactant species, P is the product species, Gas is the gas mixture defined in the boundary condition block, and ν is defined in terms of the species densities as $\nu = \rho_P/\rho_R$. A simple Arrhenius rate expression is used for this reaction, of the form

$$r_1 = \bar{\rho} Y_R A_1 \exp\left(\frac{-E_1}{R_u T}\right), \tag{2.311}$$

where A_1, E_1, and R_u are the pre-exponential factor, the activation energy, and the universal gas constant, which are all parameters passed to this model in the input deck. The remaining parameters passed in are the index of the reactant and product species from the list of species defined in the material definition block. The first species has index 1, and they count upward sequentially. This model then yields species reaction rates of

$$\dot{\omega}_R''' = -r_1 \tag{2.312}$$

$$\dot{\omega}_P''' = \nu r_1. \tag{2.313}$$

The volumetric heat production rate model shown here is a simple function of the production (destruction) rate of the reactant species, modeled as

$$\dot{q}''' = \dot{\omega}_R''' H_1, \tag{2.314}$$

where H_1 is the heat release parameter provided to the model by the user.

Alternately, the same mechanism defined in a block using a chemistry description would be

```
BEGIN PROPERTY SPECIFICATION FOR FUEGO MATERIAL composite
  REFERENCE temperature = 300.
  REFERENCE mass_fraction Carbon = 0.6
  REFERENCE mass_fraction Epoxy  = 0.4
  REFERENCE mass_fraction Char   = 0.0
  # Note: Keep above species order in sync with production rate

  species_density Carbon = 1779.  # kg/m^3
```

```
  species_density Epoxy  = 1300.  # kg/m^3
  species_density Char   = 650.   # kg/m^3

  SUBROUTINE FOR density = density_inverse_mass_average
  INPUT VARIABLES FOR density_inverse_mass_average = species_density mass_fraction

  # Target model: k = 0.023 * T(C)^0.46  (fig. 3 of DOT/FAA/AR-07/57)
  # functional form: k = a + b * ((T-To)/Tref)^r
  SUBROUTINE FOR thermal_conductivity = thermal_conductivity_T_power
  INPUT VARIABLES FOR thermal_conductivity_T_power = temperature
  REAL DATA FOR thermal_conductivity_T_power = 0.0 0.023 273.15 1.0 0.46  # a b To Tref r

  # Target model: c = 750 + 4.1 * T(C)  (fig. 4 of DOT/FAA/AR-07/57)
  # functional form: c = a + b * ((T-To)/Tref)^r
  SUBROUTINE FOR specific_heat = specific_heat_T_power
  INPUT VARIABLES FOR specific_heat_T_power = temperature
  REAL DATA FOR specific_heat_T_power = 750 4.1 273.15 1.0 1.0  # a b To Tref r

  emissivity = 0.95

  Begin ODE solver parameters
    ODE Solver = LSODE
    Absolute Tolerance = 1e-6
    Relative Tolerance = 1e-3
  End

  Begin Chemistry Description MyChem
    Begin Reaction R1
      Reaction is Epoxy -> 0.5Char + 0.5CH4
      Rate Function = Arrhenius A = 3.33e15  Ea = 2.2614e5 R = 8.314
      Concentration Function = Standard  mu = Automatic
      Heat of Reaction = 0.0
    End
  End
END
```

where the single reaction is defined in the reaction block within MyChem. Adding additional reactions in this form is straightforward and requires no subroutine modifications. Since this approach uses a coupled ODE solve in the boundary instead of a segregated solve, an additional block defining the ODE solver parameters can be added. If this is omitted, the default solver will be used (CVODE with a 5^{th} order BDF implicit scheme).

Chemistry descriptions can also be used to define surface reactions. This is a new beta feature. These reactions only occur on the front interface of the chemistry composite, and can currently be used to describe deposition/condensation mechanisms. In order to use surface reactions two chemistry blocks must be defined within the composite material block: one for the bulk reactions (i.e. volumetric reactions) and another for the surface reactions. Users should indicate which is the volumetric reaction block with the line command "Bulk Chemistry Description." Modifying the above example to add a surface reaction yeilds,

```
BEGIN PROPERTY SPECIFICATION FOR FUEGO MATERIAL composite
  REFERENCE temperature = 300.
  REFERENCE mass_fraction Carbon = 0.6
  REFERENCE mass_fraction Epoxy  = 0.4
  REFERENCE mass_fraction Char   = 0.0
  # Note: Keep above species order in sync with production rate

  species_density Carbon = 1779.  # kg/m^3
  species_density Epoxy  = 1300.  # kg/m^3
  species_density Char   = 650.   # kg/m^3

  SUBROUTINE FOR density = density_inverse_mass_average
```

```
  INPUT VARIABLES FOR density_inverse_mass_average = species_density mass_fraction

  # Target model: k = 0.023 * T(C)^0.46  (fig. 3 of DOT/FAA/AR-07/57)
  # functional form: k = a + b * ((T-To)/Tref)^r
  SUBROUTINE FOR thermal_conductivity = thermal_conductivity_T_power
  INPUT VARIABLES FOR thermal_conductivity_T_power = temperature
  REAL DATA FOR thermal_conductivity_T_power = 0.0 0.023 273.15 1.0 0.46  # a b To Tref r

  # Target model: c = 750 + 4.1 * T(C)  (fig. 4 of DOT/FAA/AR-07/57)
  # functional form: c = a + b * ((T-To)/Tref)^r
  SUBROUTINE FOR specific_heat = specific_heat_T_power
  INPUT VARIABLES FOR specific_heat_T_power = temperature
  REAL DATA FOR specific_heat_T_power = 750 4.1 273.15 1.0 1.0  # a b To Tref r

  emissivity = 0.95

  Begin ODE solver parameters
    ODE Solver = LSODE
    Absolute Tolerance = 1e-6
    Relative Tolerance = 1e-3
  End

  Begin Chemistry Description MyChem
    Begin Reaction R1
      Reaction is Epoxy -> 0.5Char + 0.5CH4
      Rate Function = Arrhenius A = 3.33e15  Ea = 2.2614e5 R = 8.314
      Concentration Function = Standard  mu = Automatic
      Heat of Reaction = 0.0
    End
  End

  Begin Chemistry Description MySurfRx
    Begin Reaction R2
      Reaction is Char + O2 -> 2.0CO
      Rate Function = Arrhenius A = 2.0e13
      Concentration Function = Standard mu = Atomic
      Heat of Reaction = 0.0
    End
  End

  Bulk Chemistry Description is MyChem

END
```

Please note if one only desires to have surface reactions a chemistry description should still be added for the bulk, but it can be empty.

Additional functionality has been added to allow the composite thickness to change based on the surface reactions. Currently this is a beta feature and only deposition/condensation type of reactions are supported. This is accomplished by allowing users to specify a max thickness when defining the composite BC. In this case the discretization's cell size is defined by the number of nodes and the max thickness. An example using this feature is included below.

```
BEGIN composite boundary condition ON SURFACE surface_1
  composite material = composite

  composite nodes = 10
  composite min_nonlinear_iterations = 1
  composite max_nonlinear_iterations = 10
  composite residual_tolerance = 1.e-4

  composite thickness = 0.002  # 2 mm
  composite max thickness = 0.0035
```

```
  composite initial_temperature = 300.
  composite initial_mass_fraction Carbon = 0.6
  composite initial_mass_fraction Epoxy  = 0.4
  composite initial_mass_fraction Char   = 0.0

  # Composition of outgassed mixture
  mass_fraction CH4 = 1.0
  mass_fraction CO  = 0.0
  mass_fraction CO2 = 0.0
  mass_fraction O2  = 0.0
  mass_fraction H2O = 0.0
  mass_fraction H2  = 0.0
  mass_fraction N2  = 0.0
  turbulent_kinetic_energy = 2.5e-5
  soot_mass_fraction = 0.0
  soot_nuclei_mass_fraction = 0.0
END
```

In this example the composite thickness becomes the initial condition, and the first 6 cells have material in them. The 6th cell is where the interface resides and it is partially filled. The other 4 cells do not have any material in them, but could potentially be filled with material as the solution proceeds. This is accomplished by allowing the interface cell to overfill during a given ODE solve, and then remapping a portion of the cell's contents to the neighboring cells before starting the next fluid time step. These additional features have increased the stiffness of the ODE equations and so explicit solvers are strongly discouraged.

The options for the ODE solver block can be found in the ODE Solver Specifications chapter. A description of the syntax for entering reactions can be found in the Chemistry Descriptions chapter.

Beta Capability:

2.13. FIRE SUPPRESSION MODEL

The Fuego EDC (Eddy Dissipation Concept) combustion model has been augmented to include a fire suppression mechanism derived from the work of Zhang and Soteriou (*An Analytical Model for the Determination of the Cup-Burner Minimum Extinguishing Concentration of Inert Fire Suppression Agents*, Proceedings of the Combustion Institute, 33, pp. 2505-2513, **2011**). In this model, suppression occurs under the assumptions of a PSR (perfectly stirred reactor). The PSR describes an ideal situation under which reactants are assumed to be mixed completely (not mixing rate limited) and combustive reaction state is a function of the reaction rate and residence time in the region through the Damkohler number, Da, where:

$$Da = \frac{t_{reaction}}{t_{residence}} \tag{2.315}$$

Consistent with this formulation, we utilize a single-step combustion reaction mechanism with the following stoichiometric form:

$$C_{n_C}H_{n_H}O_{n_O}N_{n_N}[Fuel] + (0.25n_H + n_C - 0.5n_O)O_2 -> (n_C)CO_2 + (0.5n_H)H_2O + (0.5n_N)N_2 \quad (2.316)$$

where n_C, n_H, n_O, n_N are the number of carbon, hydrogen, oxygen, and nitrogen atoms, respectively, per molecule of fuel species. The current Fuego EDC implementation allows for only a single fuel species.

In contrast to this single-step form, the Fuego EDC model, makes uses of a two-step reaction mechanism:

Reaction 1:

$$C_{n_C}H_{n_H}O_{n_O}N_{n_N}[Fuel] + (0.5n_C - 0.5n_O)O_2 -> (n_C)CO + (0.5n_H)H_2 + (0.5n_N)N_2 \quad (2.317)$$

Reaction 2:

$$(n_C)CO + (n_H/2)H_2 + (n_C/2 + n_H/4)O_2 -> (n_C)CO_2 + (n_H/2)H_2O \quad (2.318)$$

where CO and H_2 are intermediate products of combustion. In using the current suppression model, we do not alter the actual two-step reaction chemistry used within Fuego, but make use of the one-step form only for the determination of the parameters needed to calculate the mean extinction concentration (MEC) of the suppressant of interest and corresponding extinction time scales associated with suppression consistent with Zhang's theory. Thus, we do not alter the results of non-suppressive combustion simulations.

Either fuel or oxygen (not both) can be in stoichiometric excess. Excess species as well as participators like N_2 or species already present in the simulation volume like H_2O, H_2, CO and CO_2 are taken into account via their effect on the overall gas enthalpy (heat capacity) which affects the adiabatic flame temperature and results in changes to the suppression outcome.

In this model, the adiabatic flame temperature (T_{ad}) and inlet temperature (T_i) are determined from current species mass fractions as well as what fraction of the species are in stoichiometric proportion for the forward (T_{ad}) and reverse (T_i) reactions. In addition to these two temperatures, the reaction activation temperature (T_a) is needed, where the single-step combustion reaction rate follows the Arrhenius form:

$$\dot{\omega}_F = A_f \rho^{n_F + n_O} Y_F^{n_F} Y_O^{n_O} exp(-T_a/T) \quad (2.319)$$

Here, A_f is the reaction rate prefactor, ρ is the gas density, Y_F and Y_O are the fuel and O_2 mass fractions, n_F and n_O are the order of the reaction with respect to fuel and O_2, respectively, and T is the current temperature.

Finally, the extinction temperature for the flame (T_{ex}) is determined via:

$$1 - \frac{T_a(T_{ex} - T_i)}{T_{ex}^2} + (n_F + n_O)\frac{1 - T_i/T_{ex}}{1 - T_{ex}/T_{ad}} = 0 \tag{2.320}$$

This equation is transcendental, so we use a Newton solver to arrive at the value within a small tolerance (10^{-6}). From the extinction temperature, the residence time at extinction is calculated in Zhang as:

$$t_{res,ex} = \frac{1}{A_f \phi^{n_O}} (\frac{\Delta h_r \, R}{C_p p M_{mix}})^{n_F+n_O-1} \frac{T_{ex} - T_i}{(T_{ad} - T_{ex})^{n_F+n_O}} T_{ex}^{n_F+n_O-1} exp(T_a/T_{ex}) \tag{2.321}$$

where ϕ is the stoichiometric mass fraction ratio of oxygen to fuel, C_p is the isobaric heat capacity of the gas, R is the ideal gas constant, p is the gas pressure, and M_{mix} is the molar mass of the gas mixture. An alternative form of this equation, and the one implemented in Fuego is:

$$t_{res,ex} = \frac{1}{A_f \phi^{n_O}} (\frac{T_{ad} - T_i}{\rho Y_{stoich,Fuel}})^{n_F+n_O-1} \frac{T_{ex} - T_i}{(T_{ad} - T_{ex})^{n_F+n_O}} exp(T_a/T_{ex}) \tag{2.322}$$

where $Y_{stoich,Fuel}$ is the stoichiometric mass fraction of fuel in the combustion reaction, and ρ is the gas density. $t_{res,ex}$ is compared to the actual residence time via the critical Damkohler number. If the ratio of the residence time to the residence time at extinction is less than the critical Damkohler number, extinction occurs and $\gamma\chi$ is set to 0 locally, suppressing the combustion reaction. Otherwise, $\gamma\chi$ is unaffected, and combustion proceeds normally. By default, the Da_{crit} takes on a value of 0, so that suppression is not enabled.

2.14. SOOT GENERATION MODEL FOR MULTICOMPONENT COMBUSTION

Soot is an important contributor to radiative exchange within a fire and between a fire and its surroundings. Soot production, destruction and transport at flame scales are still active areas of research, with important chemical/physical processes not understood from a fundamental physics point of view. Basically, soot particles are carbon-rich solid particles generated in regions of excess pyrolyzate, such as on the rich side of a diffusion flame. Unagglomerated soot particles have characteristic dimensions in the range 0.01–0.05 μm (Zukoski [55]).

The main purpose of the soot model is for the calculation of the absorption coefficient in the radiant energy transfer equation. For the current implementation we employ the soot model implemented in the KAMELEON code because it has been used for large turbulent fire calculations with participating media radiation. The model is discussed in Magnussen et al. [32] and Magnussen and Hjertager [56] It is a two-step formulation, first described by Tesner et al. [57]. The model for generation and combustion of soot can be summarized by three principal steps: 1) particle nucleation, where the first solid soot particles (often called radical nuclei) are created as a result of fuel oxidation and pyrolysis, 2) particle growth, whereby the soot particle size increases due to the addition of material which is primarily carbon (10–20% mole fraction hydrogen) through a series of reactions and coagulation, 3) particle oxidation, where soot particles are burned. Additional information is provided in the overview by Haynes and Wagner [58].

Since the soot model is primarily directed at closing emission/absorption terms in the radiative transfer equation, engineering approximations are made with respect to its inclusion in the Navier Stokes equations. Specifically, heats of reaction associated with formation and destruction are not accounted for in the heat balance, and the mass concentrations of soot and radical nuclei are not included in the species mass balances; they are treated as tracers. The model has a significant amount of empiricism associated with it, necessitated by the extreme length scale range of soot processes, its complexity, and the degree to which many processes have yet to be quantified from a first principles perspective. The model choice can be considered to be a pragmatic one based on its prior use in fire calculations.

The basic approach to the subgrid modeling of the elementary mechanisms, suitable for use in a numerical model are presented in detail within the Fuego theory manual.

2.15. VIRTUAL THERMOCOUPLE MODEL

The purpose of the virtual thermocouple model as implemented in Fuego is to approximate the temperature that would be obtained from a thermocouple, given the results of a CFD simulation.

Neglecting conduction through the thermocouple, the governing equations describing heat transfer to the thermocouple are written as

$$\rho c_p V_{tc} \frac{\partial T}{\partial t} = \nabla \cdot \mathbf{q}_r - h A_{tc} \left(T - T_\infty \right), \tag{2.323}$$

or alternatively,

$$\int \rho c_p \frac{\partial T}{\partial t} \, dV = \oint \mathbf{q}_r \cdot \mathbf{n} \, dA - \oint h \left(T - T_\infty \right) dA, \tag{2.324}$$

where $\mathbf{q}_r$ is the radiative heat flux vector, h is the turbulent heat transfer coefficient, and T_∞ is the surrounding gas temperature. In equation (2.324), the integrals are evaluated over the surface of the thermocouple.

2.15.1. Convective Heat Flux

The heat transfer coefficient is given in terms of the Nusselt number, the gas phase thermal conductivity, and the pertinent thermocouple length scale, ℓ, as

$$Nu = \frac{h\ell}{\lambda} \tag{2.325}$$

Correlations for Nusselt Number The Nusselt number is given as a function of the Reynolds and Prandtl numbers by the following correlation, obtained from Incropera & Dewitt (1996) for a cylinder in cross-flow:

$$Nu = 0.3 + \frac{0.62 Re^{1/2} Pr^{1/3}}{\left[1 + (0.4/Pr^{2/3})\right]^{1/4}} \left[1 + \left(\frac{Re}{282,000} \right)^{5/8} \right]^{4/5}, \tag{2.326}$$

with the Reynolds number and Prandtl number given by

$$Re = \frac{u_g \ell \rho_g}{\mu_g}, \qquad Pr = \frac{c_{p_g} \mu_g}{\lambda_g}. \tag{2.327}$$

The subscript g emphasizes that these properties are evaluated in the gas phase. The length scale, ℓ, is the thermocouple diameter. The velocity, u_g, is the component of the gas velocity perpendicular to the thermocouple.

2.15.2. Radiative Heat Flux

Given the incident spectral radiation intensity field, $I = \int_0^\infty I_\lambda \, d\lambda$, the total radiative heat flux in direction $\mathbf{s}$ may be obtained as

$$\mathbf{q}_r = \int_0^{2\pi} \int_0^{\pi} I(\theta, \phi) \, \mathbf{s} \cos\theta \sin\theta \; d\theta \; d\phi, \tag{2.328}$$

where $\mathbf{s}$ is the directional vector,

$$\mathbf{s} = \sin\theta \sin\phi \mathbf{i} + \cos\theta \mathbf{j} + \sin\theta \cos\phi \mathbf{k}, \tag{2.329}$$

as depicted in figure 2.2-1. However, we are interested in only the heat flux incident on the faces of the control volume which contain the thermocouple. The incident heat flux on any surface, q_r^I, may be obtained as

$$q_r^I = \int_0^{2\pi} \int_0^{\pi} I(\theta, \phi) \, F(\theta, \phi) \cos\theta \sin\theta \; d\theta \; d\phi, \tag{2.330}$$

where

$$F(\theta, \phi) \equiv \max\left(0, -\mathbf{n} \cdot \mathbf{s}\right), \tag{2.331}$$

and $\mathbf{n}$ is the outward-pointing unit-vector normal for the given surface.

Assuming that the thermocouple emits radiation according to the Plank distribution, the net radiative heat flux at any point on the thermocouple may be written as

$$q_r = \alpha_c \, q_r^I - \varepsilon_c \sigma T^4, \tag{2.332}$$

where α_c and ε_c are, respectively, the absorptivity and emissivity of the thermocouple, and q_r^I is given by (2.330).

2.15.2.1. Caveat for Fuego Region implementation

When using the new MPMD coupling capability for the PMR solution instead of a Syrinx region the full intensity distribution is not available to Fuego to use in the radiative heat flux calculation for the virtual thermocouple model. Only the first two moments of the intensity are available: the scalar flux G and radiative flux vector $\mathbf{q}_r$. As a result in this implementation we assume a simplified form of the intensity distribution:

$$I(s) = I_0 + I_{\text{directed}} \delta(s - s_{\text{rad flux vector}}) \tag{2.333}$$

where δ is the Dirac delta function. Given this assumption and the definitions of G and $\mathbf{q}_r$ we arrive at:

$$I_{\text{directed}} = \|\mathbf{q}\| \tag{2.334}$$

$$I_0 = \frac{G - \|\mathbf{q}\|}{4\pi} \tag{2.335}$$

2.15.3. Working Assumptions

There are several working assumptions:

1. The thermocouple does not affect the flow field in any way (i.e. one-way coupling to the model). This implies the next assumption:
2. The thermocouple exists entirely within a single computational cell. Ideally, the thermocouple dimensions should be small relative to the computational mesh. Otherwise, the assumption that the thermocouple does not affect the flow field is invalid.
3. Negligible attenuation of the irradiation between the edge of the computational cell and the thermocouple surface,
4. Negligible conduction along the thermocouple.
5. Spectral emission from the thermocouple is assumed to follow the Plank distribution, i.e. the total emissive power is proportional to σT^4.
6. All thermocouple properties are homogeneous and constant (do not vary with space, or time).
7. The emissivity and absorptivity of the thermocouple are equal, $\varepsilon_c = \alpha_c$.
8. The heat transfer coefficient and convective temperature, T_∞ are homogeneous over the thermocouple.
9. The thermal conductivity of the thermocouple is sufficiently large that conduction through the thermocouple is fast relative to the convective and radiative time scales. This implies that the thermocouple is at a single, uniform temperature.

Several of these assumptions stem from the assumption that the thermocouple exists within a single computational cell. For a more detailed treatment, details of the thermocouple geometry must be specified to a level where meshing the thermocouple itself may be required.

2.16. UNITS AND UNIT CONVERSIONS

Units in a Fuego simulation are handled differently depending on whether a third-party library (TPL), like Cantera, is utilized or not.

If a TPL, especially Cantera, is used, then units are assumed to be in terms of the CGS units system, with the reference pressure in material property specified in units of atmospheres. Other pressures such as in boundary and initial conditions are in CGS unit. If it is desired to utilize some other unit system,

then the user must specify proper scaling factors for length, mass, time, and temperature. These commands can be found in Chapter 11.

In the case where a TPL is not utilized, then all units must be specified in the user's unit system. Only exception is the reference pressure in material property which is specified in units of atmospheres - for a more thorough explanation, please see Section 2.1.2. As in many other application codes, care must always be taken to ensure that all units are consistent in order to attain correct results.

Note that some quantities, like turbulent kinetic energy, are clipped to a minimum bound. The value of this bound is dimensional, and is adjusted based on the specified unit system of the problem. It is recommended that you always explicitly specify the unit system so that the clipping can be done appropriately.

2.17. PARTICLE WALL INTERACTION MODELS

Particle-wall interactions are specified in Fuego using a reflection model and an adhesion model. When a particle impacts a wall, the reflection model is called first to determine if it reflects or shatters. If not, then the adhesion model is called to determine if it sticks, resuspends, or is simply removed from the simulation and accumulated on the wall in a scalar field. The various models available are described in the following sections.

2.17.1. Reflection Models

2.17.1.1. No Reflection Model

The no reflection model does no calculations and simply moves on to the adhesion model for every impact. It is specified by:

```
REFLECTION MODEL = NONE
```

2.17.1.2. Rebound Model

The rebound model causes the incoming particle to rebound at an angle based on its incident angle. It never calls through to the adhesion model. It is specified by:

```
REFLECTION MODEL = REBOUND
```

2.17.1.3. Yoon Rebound Model

The Yoon rebound model [59], uses a conditional rebound for liquid particles, and always rebounds for solid particles. The particle rebounds if both of the following conditions are true.

$$\frac{We}{\sqrt{Re}} > \frac{3}{4}(1 - \cos(\theta)) \tag{2.336}$$

$$We > We_{crit} \tag{2.337}$$

where θ is the contact angle with the surface and We_{crit} is a user-specified critical Weber number much less than 1 (the default value is 0.01). If you do not provide a contact angle, only the Weber number criteria is used.

It can be specified in the following ways, where the behavior with omitted values is as-described above and the contact angle is in degrees:

```
REFLECTION MODEL = REBOUND_YOON  contact_angle = 45  We_crit = 0.001
REFLECTION MODEL = REBOUND_YOON  contact_angle = 90
REFLECTION MODEL = REBOUND_YOON  We_crit = 0.001
REFLECTION MODEL = REBOUND_YOON
```

2.17.1.4. Brown Shatter Model

The Brown shatter model [60] uses the Mundo [61] condition for shattering for liquid particles, where particles that satisfy the following condition are shattered.

$$\sqrt{We}Re^{0.25} > K_{crit} \tag{2.338}$$

Solid particles impacting a wall with a shatter model are simply rebounded. If a shatter does not occur for a liquid particle, the Yoon Rebound model is used to determine whether the particle rebounds, or whether the adhesion model should be called. If a shatter does occur, it is described following the methods described by Brown et al. [60]. The number of splashed particles (N_{ps}) and the mass fraction splashed (f_s) are calculated by

$$N_{ps} = -92 + 57\log_{10}(We) \tag{2.339}$$

$$f_s = \frac{We}{We + 10^6} \tag{2.340}$$

where the number splashed (N_{ps}) is limited by the MAX_FINGERS argument that can be provided in the input file (the default value is 20). The angle of rotation of the splashed particle trajectory around

the surface normal (ϕ) is dictated by the following distribution function, where A and B are user-provided constants and R is the ratio of the incoming particle tangential velocity to its total velocity.

$$f(\phi) = 1 - AR\sin^2(\phi) - BR\sin^2\left(\frac{\phi}{2}\right) \tag{2.341}$$

The entire 2π circle is discretized into N_{ps} segments of equal probability based on this distribution function, giving one shattered particle per segment. The trajectory of the particle within that segment is determined from a uniform random distribution.

The shattered particle elevation angle is selected between the specified minimum and maximum graze angles using a uniform random distribution.

The shattered particle velocity is the incoming particle velocity multiplied by a random value selected from a Chi distribution with $k = 2$, which has a mean of approximately 1.25.

The radius of the splashed particles is determined using a mass balance with N_{ps} and f_s.

The only required argument for the model is K_crit. The other optional arguments, with their defaults, are: A (0.297), B (0.723), max_fingers (20), min_graze_angle (1), and max_graze_angle (15). The arguments for contact_angle and We_crit are passed through to the Yoon rebound model. Note that although this example is displayed on two lines, the entire input must be on a single line in the input file.

```
REFLECTION MODEL = SHATTER_BROWN  K_crit = 55  A = 0.2  B = 0.8  contact_angle = 45
          We_crit = 0.001 max_fingers = 30 min_graze_angle = 1 max_graze_angle = 45
```

2.17.2. Adhesion Models

2.17.2.1. No Adhesion Model

The no adhesion model simply deletes the particle and accumulates it in the scalar accumulation variables on the surface. This reproduces the behavior of both the old STICK and PASS_THROUGH interaction types. It is specified using:

```
ADHESION MODEL = NONE
```

2.17.2.2. Stick Model

The stick model is similar to the no adhesion model, except the particle is not deleted from the simulation, but remains on the surface and is marked as stuck. It is specified using:

```
ADHESION MODEL = STICK
```

2.17.2.3. Wichner Resuspension Model

The Wichner resuspension model described in SAND2015-6119 is similar to the stick model in that the particle is stuck to the surface, but it can be removed from the surface if the force balance is favorable to do so. The Wichner resuspension model uses a lift force and adhesion force calculated from

$$F_{lift} = \alpha A_p \tau_w \tag{2.342}$$

$$F_{adh} = 10^{-9} \frac{r_p}{\epsilon} \tag{2.343}$$

where α is a user-provided constant, ϵ is the wall roughness, r_p is the particle radius, A_p is the particle cross-sectional area, and τ_w is the fluid shear stress on the wall. If a standard deviation for ϵ is provided (epsilon_std), then ϵ is sampled from a normal distribution clipped to always be positive. If a gravity vector is specified in the input file, the buoyant force on the particle is also included in this force balance.

If the net force on the particle is away from the surface, it is moved one radius plus the surface roughness from the surface in the normal direction and is reintroduced to the fluid element with zero initial velocity.

```
ADHESION MODEL = RESUSPEND_WICHNER  alpha = 5 epsilon = 1e-5
    epsilon_std = 1e-6
```

2.17.2.4. Generic Resuspension Model

The Generic resuspension model is a superset of the Wichner model described previously. Instead of hard-coding the form of the lift and adhesion forces, they are specified by the user in the input file. The specified forces can be functions of: wall shear (tauw), fluid density (rhoF), surface roughness (eps), particle radius (rP), particle cross-sectional area (AP), and fluid viscosity (muF). You can also specify a functional form for the reinsertion height, which is a function of the same variables as the other functions. If this is omitted, the default behavior is to reintroduce it one radius plus the surface roughness like the Wichner model.

```
ADHESION MODEL = RESUSPEND_GENERIC epsilon = 1e-5
    lift_force = 5*AP*tauw  adhesion_force = 1e9*rP/eps
ADHESION MODEL = RESUSPEND_GENERIC epsilon = 1e-5
    lift_force = 5*AP*tauw  adhesion_force = 1e9*rP/eps
    reinsertion_height = eps + rP
```

3. NUMERICS

We surveyed commercial codes that provide turbulent combustion capabilities and discovered that most of those codes are based on finite volume methods. Between commercial evidence and our own experiences, we came to the conclusion that finite volume methods would provide a robust and stable means of solving the fire math models. Our selection of finite volume methods is constrained by the current implementation of software architecture in the SIERRA Frameworks. The mesh must be unstructured with flow variables located at the element vertices. The domain boundary is coincident with element faces. The discrete equations are assembled on an element-by-element procedure using the SIERRA workset approach for cache-use efficiency. The finite volume approach that we implement is based on the control-volume finite-element method.

Control-volume finite-element methods (CVFEM) are a class of numerical methods for solving the Navier-Stokes equations of fluid mechanics. Although the methods are applicable to the most general case of a compressible flow, they are most commonly applied to incompressible flows. This text is a discussion of the control-volume finite-element methods appropriate for numerical solutions to the low-Mach number Navier-Stokes equations with heat and mass transfer—the equations used to describe physical applications such a combustion or chemical vapor deposition.

The CVFEM's are a combination of desirable features from both the finite-element method (FEM) and the finite-volume method (FVM), though the CVFEM is truly a finite-volume method. The CVFEM differed from other FVM's at its inception in that the CVFEM used non-staggered, unstructured meshes like a FEM. Concepts from the finite-element method include: 1) the finite-element data structure and the associated shape functions or interpolation functions, 2) integral equations assembled on an element-by-element basis, an efficient process for cache-based computer architectures, and 3) unstructured meshes with arbitrary connectivity (this is not particular to FEM's, but certainly more common). Reviews for the finite-element method are given by Zienkiewicz and Taylor [62, 63], Tezduyar [64], and Gresho [65]. Concepts from the finite volume method include: 1) physically-based integral formulation constructed from physically-based interpolation functions, 2) conservation properties at the control-volume level, and 3) both a convecting and convected velocity field to avoid pressure-velocity decoupling. Some comprehensive reviews for the finite-volume method are given by Patankar [66], Shyy [67], and Ferziger and Peric [68].

The standard mesh configuration for vertex-centered CVFEM's has all flow variables collocated at the grid points, also called nodes. The nodes are the vertices of the finite-elements, as shown in Figure 3.0-1. The finite-volumes, also called control volumes, are centered about the nodes. Each element contains a set of sub-faces that define control-volume surfaces. The sub-faces consist of the segments or surfaces that bisect the element faces.

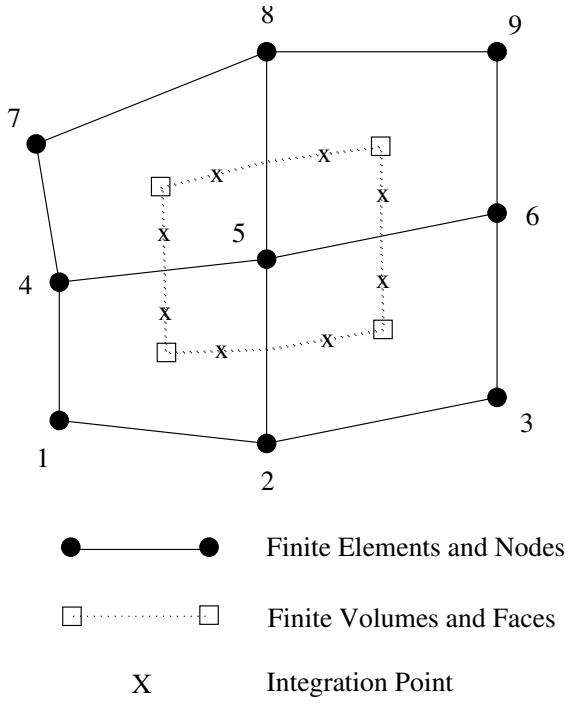


Figure 3.0-1.. Control Volume is Centered about Finite-Element Node

A three dimensional hexahedral element is shown in Figure 3.0-2. SIERRA/Fuego 1.0 supports heterogeneous topologies, i.e., hex, tet, wedge, pyramids, however, only hexahedral elements have been formally verified.

3.1. FLOW SOLVER

The core flow solver is based on a segregated, projection method approach. The projection method is used to compute the pressure field which is consistent with a velocity field that satisfies continuity. A pressure-smoothing approach similar to the Rhie/Chow scheme [69] is used to prevent pressure decoupling on the collocated mesh. An upwind method is used to interpolate convected values to control volume faces. Detailed descriptions of these methods are in the Theoretical document.

3.2. CONVECTION OPERATOR

We currently support several upwind interpolations for convection. The upwind methods are blended with a centered scheme that becomes dominant below a specified cell-Peclet number.

The first scheme is a simple first-order scheme that considers the two nodes adjacent to a control volume face and extrapolates from the node in the upwind direction.

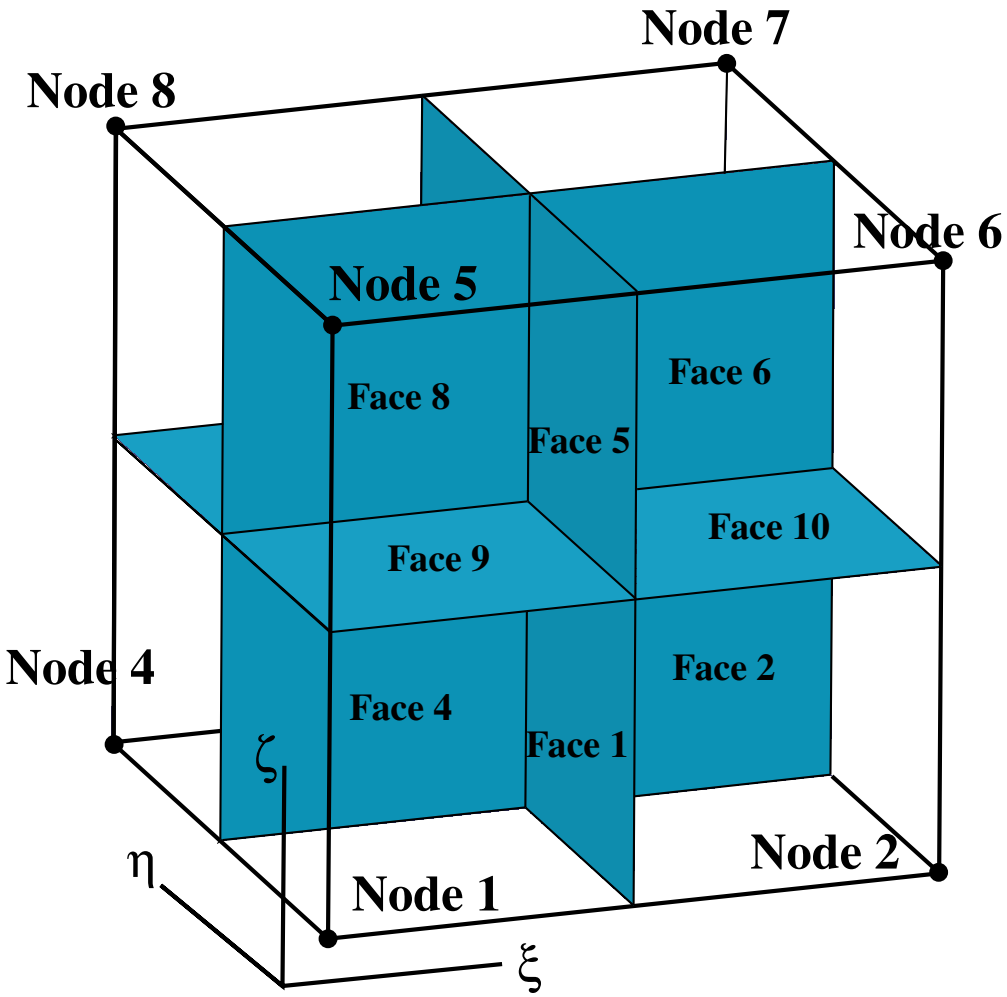


Figure 3.0-2.. Sub Control Volumes Within a Hexahedral Element

The upwinded value is blended with a centered value.

Modified linear profile skew upwinding is a simplification to the skew upwinding approach in the FIELDS scheme [70, 71]. We omit the physical advection correction terms. Integration point values at control volume subfaces are interpolated from upwind intersection points on the element face. In the original skew upwind scheme, the intersection point could either be interior subface or element faces. The interpolation coefficients were computed by inverting a matrix relation between integration point values and nodal values. The linear profile skew upwinding does not use interior subface intersections – only element face intersections.

The MUSCL approach (see Chap. 21 of Hirsch [72]) for higher order upwinding is adapted to unstructured meshes. The upwind interpolation is constructed along each edge of an element. The interpolation makes use of the two end nodes of the edge and the centered gradient constructed at the two end nodes. The MUSCL approach constructs an interpolation in one dimension from four (or more) uniformly distributed nodal values. Limiter functions are introduced to prevent numerical

oscillations from occurring.

3.3. SEGREGATED SOLUTION PROCEDURE

The time integration method is a two-level, backward Euler scheme, requiring data at two time states. The discrete form of the nonlinear equations is

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = F\left(\phi^{n+1}, \phi^n\right). \tag{3.1}$$

Sub-iteration is required within the time step to satisfy the nonlinearities. Over one sub-iteration, the nonlinear equations are solved in a segregated manner. Each segregated equation set is linearized and solved as a linear problem. During the nonlinear iteration process, a temporary variable may be introduced to differentiate the old guess at the state (n+1) from the new guess at the state (n+1). A temporary variable (*) is introduced to hold the new estimate of the state (n+1). The temporary variable is typically only used in describing the algorithm. Functionally, the (*) variables and (n+1) variables are usually represented by the same array within the code. The only time a temporary variable would be used in the code is if the momentum equations were segregated or if the species diffusion velocities were not pre-computed.

Within the transport equations, the convection terms are linearized by freezing the mass flux (density * velocity * area).

A linear solve is performed for each equation set within a nonlinear sub-iteration. There is a solver object associated with each equation set within the SIERRA framework. The solver object contains the matrix connectivity and manages the assembly of the matrix components. There will be ten solver objects for the full turbulent combustion mechanics (the species equations all use the same solver object). There will also be ten repeated sets of connectivity information.

The ordering of the segregated equations during one nonlinear iteration is given in the following list. Reduced equation sets for simplified mechanics maintain the same relative ordering.

1. evaluate material properties using the most recent estimate of temperature and composition
2. evaluate turbulent eddy viscosity if turbulent
3. evaluate combustion model species production rates
4. evaluate soot model production rates
5. evaluate gas and soot absorptivity for radiation model
6. solve x-momentum equation, store new predicted x-velocity until all momentum equations have been evaluated
7. solve y-momentum equation, store new predicted y-velocity until all momentum equations have been evaluated

8. solve z-momentum equation, store new predicted z-velocity until all momentum equations have been evaluated
9. update predicted velocities
10. solve continuity equation using predicted velocities, update new pressure
11. update new mass fluxes at all control volume sub-faces, including boundaries, and use in subsequent transport equations
12. perform the velocity projection and correct all nodal velocities
13. assemble turbulence friction velocity
14. solve the turbulent kinetic energy equation if turbulent, store turbulent kinetic energy until turbulence dissipation equation is solved so that the production and dissipation source terms can be properly linearized
15. solve the turbulence dissipation equation if turbulent, update the turbulent kinetic energy and turbulence dissipation
16. solve the enthalpy equation
    - laminar: solve for temperature
    - turbulent: solve for enthalpy
17. solve each species equation, do not update species mass fractions until all species equations have been solved
18. solve the soot equation, store soot mass fraction until soot nuclei equation is solved
19. solve the soot nuclei equation, update soot mass fraction and soot nuclei mass fraction
20. compute Nth species mass fraction using summation rule
21. update temperature or enthalpy at new time level
    - laminar: compute enthalpy
    - turbulent: extract temperature
22. extract temperature from enthalpy if laminar
23. compute new density and time derivative of density

This procedure is repeated within a time step until the desired level of nonlinear equation convergence is achieved.

3.4. CONJUGATE HEAT TRANSFER

A conjugate heat transfer problem is one in which conductive heat transfer in a solid region is coupled to the convective heat transfer in a neighboring fluid. In its most general form, the coupling at the boundary is governed by the conservation of energy, such that heat flux out of the solid is equal to heat flux into the fluid:

$$\mathbf{q}_s \cdot \mathbf{n} = \mathbf{q}_f \cdot \mathbf{n} \tag{3.2}$$

where $\mathbf{q}_s$ and $\mathbf{q}_f$ are the heat flux in the solid and fluid, respectively, and $\mathbf{n}$ is the surface normal directed into the solid and out of the fluid. For more details of the numerical implementation, please see the theory document.

4. MPMD RADIATION

This chapter outlines the use of Multiple-Program-Multiple-Data (MPMD) coupling to perform radiation solves with two different codes: Scefire and Nalu. MPMD coupling allows the radiation solve to occur on an entirely different set of processors using an executable other than Fuego.

4.1. INPUT DECK CONVERSION

The Conversion of a Fuego/Syrinx input deck to a Fuego MPMD input deck is straightforward. The basic steps to convert a Syrinx case to a Fuego-Nalu case are:

1. Delete the Syrinx region, PMR material, PMR solver, and PMR mesh (finite element) blocks from the Fuego input deck.
2. Delete all transfer blocks to or from the PMR region and all calls to them from solution control (see Section 4.1.1).
3. Add the `USE MPMD RADIATION` command to the Fuego input deck in the Fuego region block to activate the MPMD coupling.
4. Add the `SKIP STEPS FOR PMR` command in the Fuego procedure or time blocks if you want to run the PMR solve less frequently than every time step.
5. Specify emissivity, transmissivity, radiation environment temperature, and radiation boundary temperature in the Fuego boundary condition blocks. If any of these are omitted defaults will be used. Default conditions are emissivity = 1, transmissivity = 0, and boundary temperature equal to the local or wall temperature depending on the boundary type.
6. Copy the template Nalu input file from Section 4.1.2 and update the name of the mesh file.
7. (Optional) Update the quadrature and numerical scheme in the Nalu input deck per your problem requirements.
8. (Optional) If using a particle region, transfer absorption and radiation sources from the particle region to the Fuego region, and transfer scalar flux from the Fuego region to the particle region (see Section 4.3).
9. Run your case using an MPMD launch command (see Section 4.1.4).

4.1.1. Transfers

In prior Fuego-Syrinx simulations the user would specifically call out which fields to transfer between Fuego and Syrinx in the input file, or call out a pre-defined transfer that would automatically define fields to transfer. With MPMD PMR the transfers are always determined automatically, so nothing is required in the input deck with regards to transfers to or from the PMR code.

The Fuego to PMR transfers are:

- Radiative source (rad_source, R_s). This is calculated by Fuego either from the temperature field or combustion model (EDC or mixture fraction) and is sent on all volume nodes. For the non-reacting laminar case, this term is

$$R_s = \frac{\sigma}{\pi}\alpha T^4 \tag{4.1}$$

  and for reacting cases it is either tabulated or otherwise modified to include sub-grid effects.

- Absorption coefficient (absorption, α). Calculated using the current property model and sent on all volume nodes.

- Emissivity. Calculated in Fuego and sent on all surface nodes.

- Transmissivity. Calculated in Fuego and sent on all surface nodes.

- Boundary Source. Calculated in Fuego using the emissivity (ε), transmissivity (τ), radiation boundary temperature (T_b), radiation environment temperature (T_e), and band fractions (f_e, f_b) if doing a spectral calculation. This is sent on all surface nodes.

$$S_{BC} = \sigma\left(\varepsilon T_b^4 f_b + \tau T_e^4 f_e\right) \tag{4.2}$$

- Boundary Beam Source. Calculated in Fuego on all surface nodes only if the beam model is in use.

The PMR to Fuego transfers are:

- Scalar flux (scalar_flux, G). Calculated by the PMR code and sent back on all volume nodes, this is used to apply the radiative source term to the enthalpy equation.

$$S_h = V_{scv}\left(\alpha G - 4\pi R_s\right) \tag{4.3}$$

  This source term is linearized using the laminar form of R_s to give

$$S_{h,lhs} = V_{scv}\frac{16\alpha\sigma T^3}{C_p} \tag{4.4}$$

- Radiative Flux (radiative_flux). Calculated by the PMR code this flux vector is sent back on all volume nodes and is primarily used for post-processing.

- Incident Flux (incident_flux). The incident flux on all surfaces is calculated by Nalu and sent back to Fuego. Scefire does not send this field, so when coupling with Scefire Fuego calculates this term using the radiative flux vector on all surfaces.

4.1.2. Nalu Input Deck

The corresponding Nalu input deck for Fuego/Nalu MPMD calculations is shown in the following sections. The Nalu input file is YAML formatted, so spaces and indentation matter.

4.1.2.1. Header

The file header defines a simulation name (`sim1` here) and names a time integrator block to use (`ti1` here, must match the name used in the following time integrator section).

Since the PMR solve is steady state, no time stepping parameters are required for the time integrator block. All that is required is to list the two realms in use for an MPMD PMR problem.

The `linear_solvers` block defines the list of linear solvers to use. Nalu PMR only solves one equation system so there need only be one solver specified here.

In general, nothing in this section should need to be modified unless you want to adjust the linear solver settings.

```
Simulations:
  - name: sim1
    time_integrator: ti1

Time_Integrators:
  - StandardTimeIntegrator:
      name: ti1

      realms:
        - pmrRealm
        - MPMDRealm

linear_solvers:

  - name: solve_scalar
    type: tpetra
    method: gmres
    preconditioner: sgs
    tolerance: 1e-5
    max_iterations: 100
    kspace: 100
```

```
    output_level: 0
    write_matrix_files: no
```

4.1.2.2. Transfers

The transfers section declares that we will be doing an MPMD PMR transfer. This section should not be modified - the transfers are all pre-defined.

```
transfers:

  - name: xfer_fluid_pmr
    realm_pair: [MPMDRealm, pmrRealm]
    coupling_physics: mpmd_pmr

  - name: xfer_fluid_pmr2
    realm_pair: [pmrRealm, MPMDRealm]
    coupling_physics: mpmd_pmr
```

4.1.2.3. Realms

The Realms section (similar to a Region in a Sierra input deck) lists the two Realms needed for a PMR problem. The first is an MPMD realm, which requires no parameters. The second is the main PMR realm.

Like Fuego, Nalu uses automatic decomposition of the mesh using RIB, so the specification of a decomposition method is not required. If desired, `automatic_decomposition_type: rcb` can be added after the `mesh:` line.

The most common parameters to change in the PMR realm are:

- Name of the mesh file.
- Numerical scheme (Edge-Upwind, Edge-SUCV, or Element-SUCV).
- PMR quadrature rule (`quadrature_type`) and order (`quadrature_order`). Valid options for quadrature type are `Thurgood` (default), `LevelSymmetric`, `PNTN`, and `UserDefined`. If the `quadrature_type` command is omitted, Thurgood is used.
- Number of nonlinear iterations. If the problem has non-zero reflectivity on a surface (typically from non-unity emissivity) then multiple nonlinear iterations are required to reach convergence. Note that you may see `max_iterations` defined in `RadiativeTransport` in some cases. The proper one to change is at the `equation_systems` level, set to 3 in the example below. The one in `RadiativeTransport` can be omitted and will default to 1.

```
realms:

  - name: MPMDRealm
    mesh: MPMDVirtual
    type: MPMDRealmType

  - name: pmrRealm
    mesh: ./100cmCube.g
    use_edges: yes

    boundary_conditions:
    - wall_boundary_condition: walls
      target_name: all_surfaces

    material_properties:
      target_name: all_blocks

    equation_systems:
      name: theEqSys
      max_iterations: 3

      solver_system_specification:
        intensity: solve_scalar

      systems:
        - RadiativeTransport:
            name: myRTE
            convergence_tolerance: 1.e-8
            quadrature_order: 6
            quadrature_type: Thurgood
            activate_upwind: no
            deactivate_sucv: no

    output:
      output_data_base_name: pmr.e
      output_frequency: 1
      output_node_set: no
      output_variables:
       - absorption_coefficient
       - intensity_bc
       - scalar_flux
       - radiative_heat_flux
       - radiation_source
       - irradiation
       - emissivity
```

```
    - transmissivity
```

Setting the numerical scheme required modifying a few parameters together. Examples of these settings are:

- Edge-Upwind

```
use_edges: yes
activate_upwind: yes
deactivate_sucv: yes
```

- Edge-SUCV

```
use_edges: yes
activate_upwind: no
deactivate_sucv: no
```

- Element-SUCV

```
use_edges: no
activate_upwind: no
deactivate_sucv: no
```

You should not use any combinations of `use_edges`, `activate_upwind`, and `deactivate_sucv` not shown in the list above.

Nalu supports the Thurgood quadrature (identical to Tn from Syrinx), which has $8N^2$ ordinate directions, N being the order of the quadrature; level-symmetric quadrature (Lathrop Carlson) orders 2,4,6,8,12, and 16; and $PN\text{-}TN$ (for any even order) which has $N(N+2)$ ordinate directions. The default quadrature is Thurgood.

Nalu requires the user to specify the name of the sidesets to be used in the `boundary_conditions` specifications. For most cases the `all_surfaces` alias can be used. In the case where it cannot be used (e.g. a mesh with internal sidesets), the wall BC sidesets can be supplied as a list (e.g. `[surface_1, surface_2, surface_3]`)

Nalu has no concept of temperatures, since Fuego transfers only radiative intensities. Although there is are temperature fields defined in Nalu they will not have values applied in them and there is no need to output them.

Fuego MPMD simulations will terminate Nalu when Fuego is finished executing, so there is no need to specify an end time or any other end criteria.

4.1.3. SceFire Input Deck

Scefire input decks have the following structure:

```
<?xml version="1.0" encoding="utf-8"?>
<SCEPTRE_Input>
  <Mesh_File>100cmCube.g</Mesh_File>
  <Output_Prefix>radResults.e</Output_Prefix>
  <Output_Format>Simple</Output_Format>
  <Output_Options>
    <Verbosity>high</Verbosity>
  </Output_Options>
  <XS_File>thermal-fromShawn.xslib</XS_File>
  <Sn_Options>
    <Sn_Order>6</Sn_Order>
    <Angular_Quadrature_Type>Level_Symmetric</Angular_Quadrature_Type>
    <!--Angular_Quadrature_Type>Lebedev</Angular_Quadrature_Type-->
  </Sn_Options>
  <Enable_User_Defined_Solvers>true</Enable_User_Defined_Solvers>
  <Solvers>
    <Solver name="1stOrder">
      <Solver_Form>First_Order</Solver_Form>
      <Error_Control_Options>
        <Maximum_Number_Iterations>20</Maximum_Number_Iterations>
        <Convergence_Tolerance>1.e-8</Convergence_Tolerance>
      </Error_Control_Options>
    </Solver>
  </Solvers>
  <Solver_Assignment explicit="true">
    <Solver_By_Group>
      <Group index="1">1stOrder</Group>
    </Solver_By_Group>
  </Solver_Assignment>
</SCEPTRE_Input>
```

The Scefire input deck requires very little user input. The first required component is the `Mesh_File`, here specified as "100cmCube.g". The Scefire mesh does not need to be identical to the Fuego fluid mesh file, but currently Scefire uses a closest node interpolation of data, so that limitation should be taken into consideration. Different decompositions (including numbers of processors) for the Scefire mesh relative to the Fuego Fluid mesh can effectively be used. The name of the output file (radResults.e) is a required entry, though most users will only use output from the Fuego exodus files. Using the "Simple" output format suppresses the Scefire output.

The `Angular_Quadrature_Type` is set here to `Level_Symmetric`, though Lebedev is also available in Scefire. The user also selects the order of the `Angulare_Quadrature` (`Sn_Order`), here set to 6.

`Level_Symmetric` quadratures, as in Syrinx, have $N(N+2)$ ordinate directions, where N is the `Sn_Order`. The `First_Order` solver should be left unchanged for all current Fuego/Scefire MPMD simulations, though Maximum Iterations and Convergence tolerances can be changed at the user?s discretion. The rest of the Scefire input deck should remain unchanged. Note that the user does not need to specify any boundary conditions in the Scefire input deck.

4.1.4. Running MPMD Jobs

The command to run in MPMD mode is different from what was used to run Fuego-Syrinx cases. To run Fuego-Syrinx cases you simply ran a parallel job of Fuego using something like

```
mpirun -np 100 fuego -i fire.i
```

In this case one executable would use 100 cores, so both Fuego and Syrinx were taking turns using the same CPU resources. With MPMD runs you are launching two separate MPI jobs with two different codes that can communicate. An example MPMD launch command would look like

```
mpirun -np 100 fuego -i fire.i : -np 100 nalu -i pmr.i
```

However, there is no requirement any more that the two codes use the same number of cores, so depending on the mesh and computational costs you may choose a different allocation per code. For example, if your PMR solve is very expensive you may allocate more cores to Nalu than Fuego:

```
mpirun -np 50 fuego -i fire.i : -np 150 nalu -i pmr.i
```

Special care must be taken when submitting MPMD jobs on the HPCs or any queued environment. By default, the two MPMD codes cannot share cores so to launch the case above on an HPC you would need to request an allocation 200 cores. This is unnecessarily wasteful though since Nalu would not be using its 150 cores while Fuego runs, and Fuego would not be using its 50 cores while Nalu runs. To get around this, you must enable oversubscription. Since Fuego and Nalu run sequentially (never executing at the same time) you can allow them to share resources. To get an allocation of 100 cores and use them all for both codes you must add additional mpi flags:

```
mpiexec --oversubscribe --bind-to core:overload-allowed -np 100  \
  fuego -i fire.i : --bind-to core:overload-allowed -np 100 nalu -i pmr.i
```

Keep in mind that the specific command to use can be platform dependent. A more complete example submission script on an HPC may look like

```
#!/bin/bash

#SBATCH --nodes=10
#SBATCH --time=48:00:00
#SBATCH --account=xxxxxxxx
#SBATCH --job-name=fire
#SBATCH --partition=batch

nodes=$SLURM_JOB_NUM_NODES
cores=36

module load sierra
export OMPI_MCA_rmaps_base_oversubscribe=1
mpiexec --oversubscribe                                          \
  --bind-to core:overload-allowed                                \
  --npernode $cores --n $(($cores*$nodes)) fuego -i fire.i :     \
  --bind-to core:overload-allowed                                \
  --npernode $cores --n $(($cores*$nodes)) nalu -i pmr.i
```

Contact `sierra-help@sandia.gov` if you need more help or encounter issues running MPMD jobs.

4.2. SPECTRAL RADIATION TRANSPORT

Fuego now supports spectral radiation calculations through MPMD coupling with either Scefire or Nalu (not Syrinx). Spectral radiation transport is instantiated in an MPMD Fuego input deck through a spectral model section similar to the following:

```
BEGIN RAD TRANSPORT SPECTRAL MODEL SPECIFICATION spectralRadModel
  Spectral Band Model = LINEAR
  Spectral Subband Model = AVERAGE
  Minimum Frequency = 1.0e13
  Maximum Frequency = 2.0e13
  Number Spectral Bands = 2
END RAD TRANSPORT SPECTRAL MODEL SPECIFICATION
```

The spectral model utilized in Fuego is based on the concept of spectral bands, which means that the user must describe frequency bands over which radiation absorptivities are averaged. In this context, a richer set of data is available to the analyst, including radiation transport terms (banded absorptivity, scalar flux, and radiative flux) than is available through the standard gray (nonbanded) radiation transport description that has been available to Fuego analysts for many years.

Spectral transport makes use of spectral data files that the user must provide. Files are named as:

`Species-data.dat`

Where "Species" is the name of the chemical species whose spectral transport properties are being described. Each species present (all species for EDC simulations) or those output through the use of output variables in mixture fraction simulations must have all necessary spectral data files available in the working directory when a spectral transport calculation is performed. Spectral data files have the following structure:

$$
\begin{array}{ccc}
T_1 & F_{1,1} & A_{1,1} \\
\cdots & & \\
T_1 & F_{1,N_1} & A_{1,N_1} \\
\cdots & & \\
T_M & F_{M,1} & A_{M,1} \\
\cdots & & \\
T_M & F_{M,N_M} & A_{M,N_M}
\end{array}
$$

For each temperature T_x of the M temperatures specified in the spectral data files, there are N_X frequencies ($F_{x,1}$ through F_{x,N_X}) specified with corresponding absorptivity values ($A_{x,1}$ through A_{x,N_x}). Temperatures within these files are required to be in non-decreasing order and frequencies must be ordered in increasing value within a specific temperature. Absorptivities are expected to be in units of $cm^2/molecule$, which is a number density weighted absorptivity. Unit conversion within the Fuego input deck is required if the desired units of the simulation are not cgs. The ideal gas law and knowledge of the mass fraction of each species is used to calculate the number density.

For the spectral radiation model specified here, the name of the particular spectral model is given as spectralRadModel, but could be any valid string. For this case, a `LINEAR` "Spectral Band Model" is used, where the set of spectral bands, in this case 2 as specified by "Number of Spectral Bands" are equally divided along the linear spectral frequency range with "Minimum Frequency = 1.0e13" Hz and "Maximum Frequency = 2.0e13" Hz. One could alternatively choose a `LOGARITHMIC` Spectral Band Model where bands are divided up equally in logarithmic frequency space.

In this case, we have chosen an `AVERAGE` "Spectral Subband Model", which indicates that averaging of absorptivities for frequencies within a spectral band will follow standard (unweighted) averaging over the particular spectral band. A `PLANCK_AVERAGE` Spectral Subband Model is also available where integrations are weighted by their Planck black-body spectrum contribution. `PLANCK_AVERAGE_WITH_REFERENCE_TEMPERATURE` is a final choice that a user can select where the band contributions are weighted by the Planck black body-spectrum at a specified user reference temperature. If this final option is used, a user must also specify a reference temperature, which can be done through: Planck Subband Model Reference Temperature = 2000.0 Here the Reference temperature is set to 2000.0 K, but could be any nonzero temperature.

Please also note that if running a spectral case with Nalu multiple non-linear iterations should be specified. This is currently necessary to converge the system even if the equation system is linear.

Beta Capability:

4.2.1. Spectral properties on surfaces

Fuego can now perform spectral radiation calculations with banded emissivities defined on surfaces. This is a beta feature, and still requires additional testing. A command similar to the one shown below is used to define banded emissivities on surfaces:

```
BEGIN WALL BOUNDARY CONDITION ON SURFACE SURFACE_1
  INTERFACE BOUNDARY
  EMISSIVITY SPECTRAL FILE NAME = SURFACE_1_SPECTRAL.DATA
  TRANSPARENT BAND EMISSIVITY = 1.0
END WALL BOUNDARY CONDITION ON SURFACE SURFACE_1
```

Similar to the spectral radiation model described in Section 4.2, a spectral data file containing the emissivity bands must be provided for each surface (provided by the "EMISSIVITY SPECTRAL FILE NAME =" command in the above). These files have a structure similar to the species spectral files:

$$
\begin{array}{lll}
T_1 & F_{1,1} & \varepsilon_{1,1} \\
\cdots & & \\
T_1 & F_{1,N_1} & \varepsilon_{1,N_1} \\
\cdots & & \\
T_M & F_{M,1} & \varepsilon_{M,1} \\
\cdots & & \\
T_M & F_{M,N_M} & \varepsilon_{M,N_M}
\end{array}
$$

Similar to the species spectral files, each temperature T_x of the M temperatures specified in the spectral file have N_X frequencies defined, each with a corresponding emissivty value ($\varepsilon_{X,1}$ through ε_{X,N_x}. Temperatures within these files must be defined in a non-decreasing order, with frequencies within each temperature group ordered in increasing value.

In addition to defining a spectral emissivity file, an additional emissivity for the transparent band can be defined as a general expression; this value is defaulted to 1. In the above example, the transparent band emissivity is defined as a constant (of 1).

4.3. PARTICLE RADIATION TERMS

Particle Radiation contributions are also still available in Fuego simulations with Lagrangian particle types that have radiation contributions (heated, CPD, evaporating, general chemistry, wildfire). Previously, the user would specify transfers of necessary thermal/radiation fields between Fuego and Syrinx, which would resemble the following:

```
BEGIN TRANSFER pmr_to_particle
    COPY VOLUME NODES FROM pmr_region TO particle_region
    SEND BLOCK block_1 TO block_1
    SEND FIELD scalar_flux STATE none TO scalar_flux STATE new
END   TRANSFER pmr_to_particle

BEGIN TRANSFER particle_to_pmr
    COPY VOLUME NODES FROM particle_region to pmr_region
    send field particle_absorption_coeff state new to particle_absorption_coeff state no
    SEND FIELD particle_rte_source STATE  new TO particle_rte_source STATE none
END TRANSFER particle_to_pmr
```

For MPMD Radiation transport simulations involving Lagrangian particles, the user now removes the pmr(Syrinx) to particle transfers and instead adds these to the already-present set of fluid to particle transfers as is seen in the following:

```
BEGIN TRANSFER fluid_to_particle
    COPY VOLUME NODES FROM fluid_region TO particle_region
    SEND BLOCK block_1 TO block_1
    ...
    SEND FIELD scalar_flux STATE none TO scalar_flux STATE new
END   TRANSFER fluid_to_particle

BEGIN TRANSFER particle_to_fluid
    COPY VOLUME NODES FROM particle_region TO fluid_region
    SEND BLOCK block_1 TO block_1
    ...
    send field particle_absorption_coeff state new to particle_absorption_coeff state no
    send field particle_rte_source state new to particle_rte_source state none
END   TRANSFER particle_to_fluid
```

4.4. BEAM (DIRECTED FLUX)

Fuego users are able to now specify beam or directed flux boundary conditions for use in MPMD simulations with PMR. A beam boundary condition is specified in the Fuego input deck in the following manner:

```
#CONVERGING/DIVERGING BEAM
BEGIN BEAM RADIATION BOUNDARY SPECIFICATION
     BEAM RADIATION BOUNDARY MODEL = FOCUSED
     FOCAL POINT = FX FY FZ
     FOCAL POWER = FP
```

```
     FOCUSED BEAM TYPE = DIVERGING (OR CONVERGING)
     CONSTRAIN TO SURFACES = sideA sideB ...
     BEAM TEMPERATURE = BT
 END BEAM RADIATION BOUNDARY SPECIFICATION
```

In the above case, a focused beam boundary `BEAM RADIATION BOUNDARY MODEL = FOCUSED` is defined where the `FOCAL POINT` of the beam, either sourcing from or arriving at this point are (`FX, FY, FZ`). The user specifies whether the beam diverges from this point or converges to this focal point. For the `DIVERGING` beam, the focal point needs to lie outside the simulation domain, whereas for the `CONVERGING` case, the focal point could lie in or outside the domain. The focal power is specified as `FP` which is the integrated power of the beam assuming the intensity is spherically isotropic. For a converging case, the power should be properly scaled to respect the actual solid angle over which the beam converges to the focal point. For instance if the converging beam actually occupies a solid angle of π, the `FOCAL POWER` should be scaled up by a factor of $4 = 4\pi/\pi$ since if the beam were incident from all solid angles, the total beam power would increase by this factor. Beams can be constrained to arrive through the domain through only a fixed set of external surfaces if so desired as indicated through `CONSTRAIN TO SURFACES = sideA sideB...`. The user also specifies the `BEAM TEMPERATURE` for cases using spectral radiation transport, since the beam power must then be parceled out among the defined spectral bands.

Another "directed" or beam-type boundary condition is available, where the user specifies not a focused beam but rather a plane wave as is seen below:

```
#PLANE WAVE
BEGIN BEAM RADIATION BOUNDARY SPECIFICATION
     BEAM RADIATION BOUNDARY MODEL = PLANE_WAVE
     PLANE WAVE BEAM DIRECTION = WX WY WZ
     PLANE WAVE INTENSITY = PWI
     CONSTRAIN TO SURFACES = sideA sideB ...
     BEAM TEMPERATURE = 3000.0
END BEAM RADIATION BOUNDARY SPECIFICATION
```

In this case, the `BEAM RADIATION BOUNDARY MODEL = PLANE_WAVE`, indicates a plane-wave type source is to be used. The user also specifies the direction for the plane wave as `PLANE WAVE BEAM DIRECTION = WX WY WZ`. The direction must be non-zero. The intensity of the plane-wave in units of Power per area is defined through `PLANE WAVE INTENSITY = PWI`. The ability to constrain the beam to only certain external surfaces is identical to that for the focused beam above as is the beam temperature.

One should keep in mind that for both beam types (focused and plane wave), beam vectors at external surfaces are only calculated for nodes on surfaces where the inner (dot) product of outwardly directed area vector and the beam flux vector is less than 0, indicating that a user specified beam must enter rather than exit the simulation domain. Of course, given the physics internal to the domain of simulation, a radiative flux vector can exit the simulation, but we disallow the possibility of a user specifying an outwardly directed beam that does not result from internal physics.

5. USER SUBROUTINES

The FUEGO code allows subroutines to be defined and used to set quantities in the code such and boundary conditions (inflow profiles), transport terms, and initial conditions. The only restriction on the subroutine being called is its signature or parameter list. Defined signatures are listed in Table 5.0-1

The subroutine must be dealt with in parsing, initialization, and workset areas of the code. In every case, the relevant parsing callbacks are in the "register_commands" member function associated with the input class.

With respect to initialization, the boundary condition classes are the most involved. Here, each mechanics instance has its own user subroutine and associated user constants. When the parsing triggers a subroutine callback, the appropriate flag is set and the subroutine name is stored. The handler then stores the subroutine name and any constants associated with it in the mesh part's parameter data. There is no other initialization to be done for subroutines, although it is important to guard any function (load curve) calls (or function initialization) with a check on the "type" of condition being set (constant, load curve, or subroutine).

In the workset portion of the boundary condition codes, the variable in question (associated with a subroutine) will need to be calculated. At this stage, the variable "type" within the code gets checked; if it's a subroutine type, the code takes its name from the instance parameter data. At that point, the subroutine pointer itself can be retrieved from the framework registry, its signature checked, and a call made to load the relevant boundary data.

In the transport equations, things are a bit less complicated: the region control data contains the appropriate subroutine associations. However, it could have one for each transport variable. Parsing callbacks are again defined in the "register_commands" member function. In this case, the parsing callback looks at the variable indicated by the parsing (e.g. a source term for pressure) and creates an association within the region control data for that particular variable and the given subroutine name. As with the boundary condition implementation, there is precious little to do to initialize a user subroutine. The only task the callback has is to register the subroutine name in the region's parameter data for later use.

In the workset portion of the code, the source term subroutines are called immediately after the FORTRAN element routine. If the appropriate flags are set, the subroutine name is retrieved from the region's control data, the subroutine pointer is retrieved from the framework registrar, and the signature checked. At this stage, the subroutine can be called. The subroutine takes from the user source terms for the right hand side and the diagonal entries on the left hand side of the discretized algebraic equations.

Like the source term implementations, initial condition user subroutines are less complicated than the boundary condition implementations. Again, the parsing callbacks are triggered in the

“register_commands" member function and again, there is little to do in initialization but store the subroutine name in the mesh part’s parameter data. In the “workset" part of the code (or that part that corresponds to a workset algorithm), the existence of a subroutine association is checked. From there, the code can retrieve the subroutine pointer from the framework registry, check its signature, and call it. In the initial condition class, the call is made by overloading the set_nodal_variable member function. The subroutine pointer is passed in as a parameter and the appropriate nodal variable is set with the subroutine a single node at a time.

The building of user subs changed between versions 2.3 and 2.4 of Fuego. Specifically, one will need to provide the Fmwk_reg_user_subs.F file. An example Fmwk_reg_user_subs.F file is given here:

```
      subroutine Fmwk_reg_user_subs()

      implicit none

      external apub_ftx3_sub
      external apub_ftx3spec_sub
      external afgo_fmmsfgo_sub

      external z_pulse_InflowBC
      external x_pulse_InflowBC
      external speciesInflowBC
      external speciesIC
      external scalarVolume

      call fmwkusersub(apub_ftx3_sub,    z_pulse_InflowBC,'z_pulse_InflowBC')
      call fmwkusersub(apub_ftx3_sub,    x_pulse_InflowBC,'x_pulse_InflowBC')
      call fmwkusersub(apub_ftx3spec_sub,speciesInflowBC, 'speciesInflowBC')
      call fmwkusersub(apub_ftx3spec_sub,speciesIC,       'speciesIC')
      call fmwkusersub(afgo_fmmsfgo_sub, scalarVolume,    'scalarVolume')

      end
```

This Fmwk_reg_user_subs.F file registers user subs that require additional f77 files in the root directory with names: z_pulse_InflowBC.F, x_pulse_InflowBC.F, speciesInflowBC.F, speciesIC.F, and scalarVolume.F. The first three external declarations in Fmwk_reg_user_subs.F are based on the values given in Table 5.0-1. These user subs are specified in the code as, e.g.

```
user subroutine file = speciesIC.F $top of input after ‘begin sierra fuego‘
subroutine for mass fraction = speciesIC
```

Formerly, the sierra script was confused by our new user sub line syntax, both in its inability to “find” a user sub line command (formerly, it was expecting to find “`subroutine =`”) and in its ability to recognize the specific user sub signature, e.g., `apub_ftx3_sub` vs. `apub_ftx3spec_sub`. Now, the user will provide the Fmwk_reg_user_subs.F file in the base directory, along with the *.F files, and use the following command:

```
  sierra --make fuego -i myInput.i
```

This command will build the code with the user subs and create the UserSubsProject as before. Details of how to link in and build user subroutines are provided in the "how to" descriptions of running on various platforms, see Chapter 20 for path.

The signatures associated with each use in the Fuego code are listed in Table 5.0-2. In general, only a slight augmentation to the input file is required, e.g.,

```
BEGIN SIERRA FUEGO

  user subroutine file = z_pulse_BC.F

  BEGIN FUEGO PROCEDURE fuego_procedure

    BEGIN FUEGO REGION fuego_region

      begin inflow boundary condition on surface surface_1
        subroutine for z-velocity = z_pulse_BC
        x-velocity = 0.0
        y-velocity = 0.0
      end   inflow boundary condition on surface surface_1

    END   FUEGO REGION fuego_region

  END   FUEGO PROCEDURE fuego_procedure

END   SIERRA FUEGO
```

Table 5.0-1.. User Subroutine Argument Lists

afgo_prop_sub	int*, real*, int*, int*, int*, real*, int*, real*
Apub_ftx3_sub	int*, real*, real*, real*, int*, int*, real*, int*
Apub_ftx3spec_sub	int*, real*, real*, int*, real*, int*, int*, int*, real*
Afgo_fmmsfgo_sub	int*, int*, int*, real*, real*, real*, real*, real*, real*, real*, real*, real*, real*

Table 5.0-2.. User Subroutine Signature Type

Property Evaluation	afgo_prop_sub
Heat BC	Apub_ftx3_sub
Convection BC	Apub_ftx3_sub
Radiation BC	Apub_ftx3_sub
Fixed BC	Apub_ftx3_sub, Apub_ftx3spec_sub
Inflow BC	Apub_ftx3_sub, Apub_ftx3spec_sub
Outflow BC	Apub_ftx3_sub
Wall BC	Apub_ftx3_sub, Apub_ftx3spec_sub
Initial Conditions	Apub_ftx3_sub, Apub_ftx3spec_sub
Source Terms	Afgo_fmmsfgo_sub

6. SIERRA ABSTRACTIONS

Before the formal command lines and descriptions required to prepare an input file are presented, it is imperative that the user understands the general design structure of the SIERRA/Frameworks.

The SIERRA Frameworks [73] provide a hierarchy for describing a mechanics code or a multi-mechanics code. At the top level is the *domain* which contains all the support infrastructure for the code. Within the *domain* is the *procedure* which manages time integration and the exchange of data between any multi-mechanics components. Within the *procedure* can be multiple *regions*. A *region* contains a description of some particular physics. Within a *region* is a collection of *mechanics* which can either be the math models that describe the physics or part of a solution algorithm.

The bulk of the Fuego code exists at the region level and below. The region-level design philosophy for the Fuego code is based on a core continuity/momentum transport capability with a configurable set of transport math models.

Therefore, a particular input file will include an owning procedure, i.e., SIERRA/Fuego and a collection of regions over which the procedure will iterate.

6.1. FRAMEWORK PROCEDURE

The procedure code manages the time integration and the exchange of data between regions. The procedure object is based on the Afgo_Procedure class (see Figure 6.2-1), which is derived from a Fmwk_Procedure class. The Afgo_Procedure class contains additional timing information. The procedure supports a fixed time step and a variable time step that is set according to a fixed CFL number. The time step control information is defined in the time-control input block. Within the time-control block are multiple time-step blocks. The time step block defines valid time step control parameters for a period of time. The time-step blocks must be contiguous in time. The Fuego code does not use the time-control block TERMINATION time for a stopping criterion. The start and stop times are defined by commands within the procedure.

6.2. FRAMEWORK REGION

The region code manages the nonlinear solve of the nonlinear equations describing a sub-mechanics (see Figure 6.2-2). The matrix assembly, linear-solve, and scatter operations are handled entirely by the Fmwk_LinearSolver class and the finite element interface (FEI). Each equation set in the fire mechanics

has an associated linear_solver. The linear_solver is told, in the parser registration phase, which workset algorithm to use to build matrix contributions.

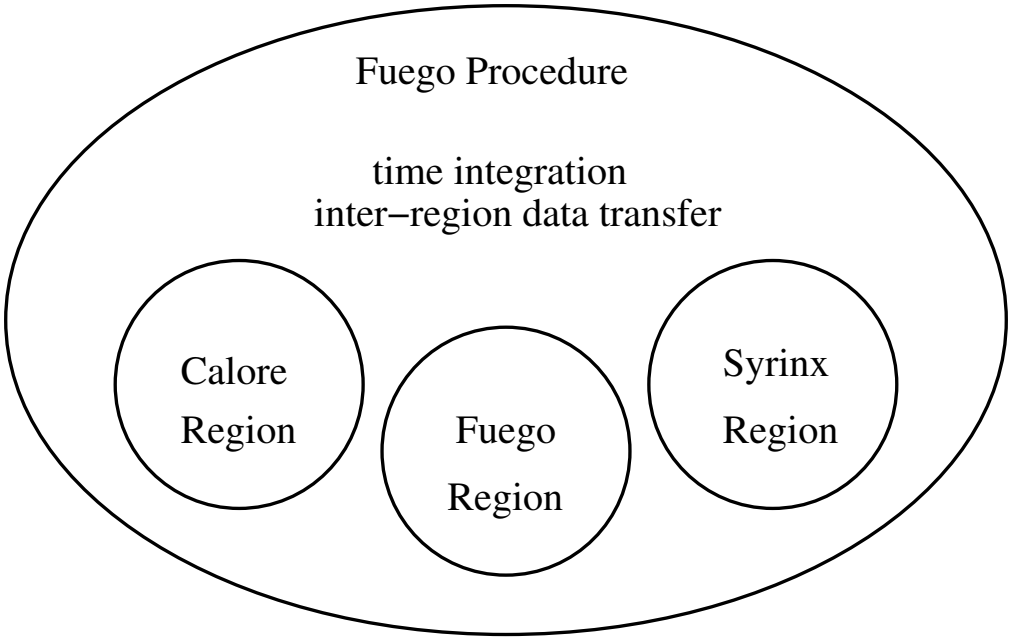


Figure 6.2-1.. Fuego Procedure Class

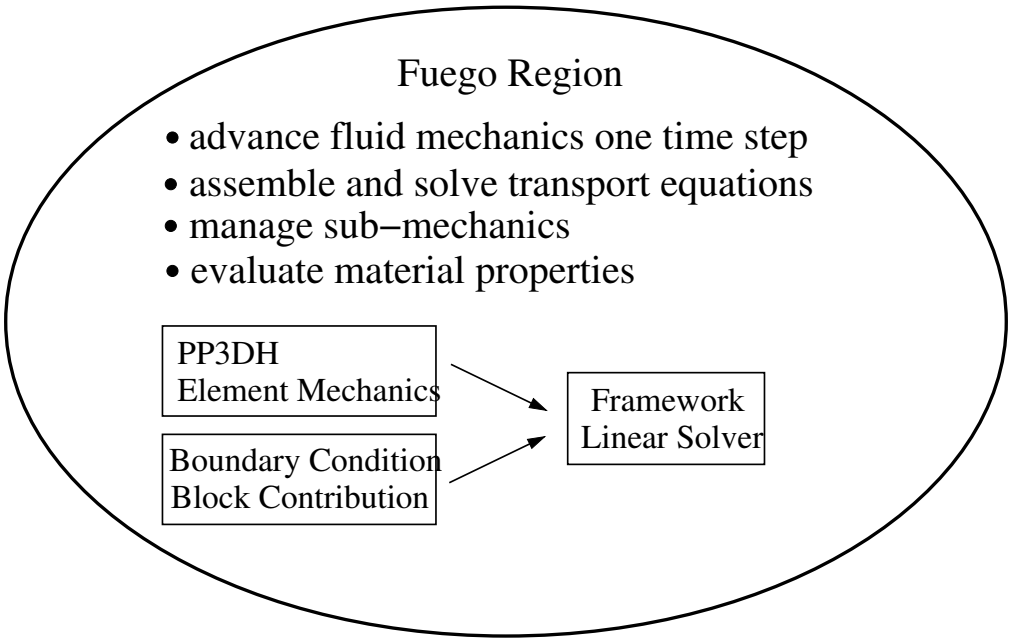


Figure 6.2-2.. Fuego Region Class

7. DOMAIN

The following commands are found in the Sierra scope of the input file.

7.1. SIERRA

Scope:

Begin Sierra *JobIdentifier*

Define Axis *AxisName* With Point *PointName Option DirectName*

Define Direction *DirectName* With Vector *Components*$_1$ *Components*$_2$ *Components*$_3$

Define Point *PointName* With Coordinates *Coordinates*$_1$ *Coordinates*$_2$ *Coordinates*$_3$

Developer Command: Run To Run Repeatably *Option1 Option2*

Load User Plugin File *File Name* [Using Function *Function*]

Maximum Warning Count {=|are|is} *Max_messages*

Restart {=|are|is} *Option*

Restart Time {=|are|is} *Time*

Serialized Io Group Size {=|are|is} *Group-size*

Test Error Messages To File *File_name* And Die On First *types...*

Title

User Subroutine File {=|are|is} *File_name*

Diagnostic Stream *File_name* [*Indent-streambuf-flags*$_1$ *[Indent-streambuf-flags*$_2$*]*]

Enable Timer *Timer-enumeration...*

Log *Log-control-name* Every *Interval*

Print Timer Information Every *Procedure-step-interval* Steps *Checkpointed*

Begin Aztec Equation Solver *Solver Name*

```
    End

    Begin Definition For Function FunctionName

    End

    Begin Finite Element Model Label

    End

    Begin Fuego Procedure ProcedureName

    End

    Begin Global Constants empty

    End

    Begin Material MaterialName

    End

    Begin Output Scheduler Label

    End

    Begin Property Specification For Fuego Material MaterialName

    End

    Begin Property Specification For Material MaterialName

    End

    Begin Tpetra Equation Solver Solver Name

    End

    Begin Trilinos Equation Solver SolverName

    End

End
```

Summary Delimits a domain to contain the procedures.

Description The "Begin Sierra jobid" and "End Sierra jobid" block contains the input commands for the analysis run. The jobid is an arbitrary string identifying the analysis.

7.1.1. Define Axis

Scope: Sierra

`Define Axis` *`AxisName`* `With Point` *`PointName Option DirectName`*

Parameter	Value	Default
AxisName	`string`	-
PointName	`string`	-
Option	`{direction|point}`	-
DirectName	`string`	-

Summary Defines a named axis in terms of a spatial point and direction vector, or from one spatial point to another.

7.1.2. Define Direction

Scope: Sierra

`Define Direction` *`DirectName`* `With Vector` *`Components`*$_1$ *`Components`*$_2$ *`Components`*$_3$

Parameter	Value	Default
DirectName	`string`	-
Components	`real_1 real_2 real_3`	-

Summary Defines a named spatial direction in terms of vector components.

7.1.3. Define Point

Scope: Sierra

`Define Point` *`PointName`* `With Coordinates` *`Coordinates`*$_1$ *`Coordinates`*$_2$ *`Coordinates`*$_3$

Parameter	Value	Default
PointName	`string`	-
Coordinates	`real_1 real_2 real_3`	-

Summary Defines a named point in space in terms of its coordinates.

7.1.4. Developer Command: Run To Run Repeatably

Scope: Sierra

`Developer Command: Run To Run Repeatably` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{off|on}`	-

Summary This option primarly effects multithreaded executation. ON = Use algorithms that are potentially non-deterministic with regards to roundoff, but optimal in terms of speed (E.g. run the model twice, you might not get the same result) OFF = Use algorithms that are fully deterministic, but possibly slower (E.g. run the model twice, you get the same results) Note, even when set to on determinacy is only guaranteed when running with identical initial conditions (same number of processors, same decomposition, same number of threads, same machine, same build options, etc.)

7.1.5. Load User Plugin File

Scope: Sierra

`Load User Plugin File` *`File Name`* `[ Using Function` *`Function`* `]`

Parameter	Value	Default
File Name	`string`	-

Summary This line command names the source file and registration function for C++ user plugins, subroutines and functions.

7.1.6. Maximum Warning Count

Scope: Sierra

`Maximum Warning Count {=|are|is}` *`Max_messages`*

Parameter	Value	Default
Max_messages	`integer`	-

Summary Sets the maximum number of warnings before the execution is terminated.

7.1.7. Restart

Scope: Sierra

`Restart {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`{auto|automatic}`	-

Summary Specify automatic restart file read.

Description **NOTE:** This command must be placed at the Sierra scope of the input file.

Specify that the analysis should be restarted from the last common time on all restart databases for each Region in the analysis. In addition to this line command, each Region in the analysis (strictly, only the region(s) that will be restarted) must have a restart block specifying the database to read the restart state data.

By default, use of this command will not cause output files (e.g., results, history, heartbeat, restart) to be overwritten. Instead output files will be written with the same basename and the suffix `-s000*`. Common visualization packages are written to handle this file organization gracefully in order for the user to view all results seamlessly.

7.1.8. Restart Time

Scope: Sierra

`Restart Time {=|are|is}` *`Time`*

Parameter	Value	Default
Time	`real`	-

Summary Specify restart file read at a specified time.

Description **NOTE:** This command must be placed at the Sierra scope of the input file.

Specify the time that the analysis will be restarted. In addition to this line command, each Region in the analysis (strictly, only the region(s) that will be restarted) must have a restart block specifying the database to read the restart state data. The restart 'time' must be greater than zero and less than or equal to the termination time.

By default, use of this command will cause previous output files (e.g., results, history, heartbeat, restart) to be overwritten. If this command is chosen, the onus is placed on the user to ensure that previous output files are not overwritten.

7.1.9. Serialized Io Group Size

Scope: Sierra

`Serialized Io Group Size {=|are|is}` *`Group-size`*

Parameter	Value	Default
Group-size	`integer`	-

Summary Specifies the number of processors which can concurrently perform I/O. Specifying zero disables serialization.

7.1.10. Test Error Messages To File

Scope: Sierra

`Test Error Messages To File` *`File_name`* `And Die On First` *`types...`*

Parameter	Value	Default
File_name	`string`	-
types	`{error|message|warning}`	-

Summary Write an error message to the specified file and then die

7.1.11. Title

Scope: Sierra

Summary User-defined title for identifying the analysis. The title continues to the end of the line (including continuation lines)

7.1.12. User Subroutine File

Scope: Sierra

`User Subroutine File {=|are|is}` *`File_name`*

Parameter	Value	Default
File_name	`string`	-

Summary This line command is only for the script that runs the application code. It needs to know where to find the *.F file that contains all the user subroutines that are referenced in the input file. Although a C++ handler is provided, the apps do not do anything with this command, only the script that runs the app. Note that the scope of this command is domain!

7.1.13. Diagnostic Stream

Scope: Sierra

`Diagnostic Stream` *`File_name`* `[` *`Indent-streambuf-flags`*$_1$ `[` *`Indent-streambuf-flags`*$_2$ `]` `]`

Parameter	Value	Default
File_name	`string`	-

Summary File path to write diagnostic messages to.

7.1.14. Enable Timer

Scope: Sierra

`Enable Timer` *`Timer-enumeration...`*

Parameter	Value	Default
Timer-enumeration	`{adaptivity` | `algorithm` | `all` | `app1` | `app2` | `app3` | `app4` | `contact` | `cpu` | `domain` | `hms` | `material` | `mechanics` | `none` | `procedure` | `profile1` | `profile2` | `profile3` | `profile4` | `recovery` | `region` | `search` | `seconds` | `solver` | `transfer` | `wall}`	-

Summary Enables runtime performance metrics data collecting.

Description The metrics cpu time, wall time, io count, msg count and flop count which are collected within several classes can be selectively enabled by metric class.

CPU and wall time metrics are enabled on all platforms. The remaining metrics are enabled on some platforms based on data availability.

Metrics: cpu Display CPU times wall Display wall times

Metric classes: all Enable all metrics none Disable all timers

adaptivity Enable metrics on adaptivity algorithm Enable metrics on algorithms app1 Enable app defined metrics contact Enable metrics on contact domain Enable metrics on the domain material Enable metrics on materials mechanics Enable metrics on mechanics procedure Enable metrics on procedures profile1 Enable app defined profiling metrics region Enable metrics on regions search Enable metrics on searches solver Enable metrics on solvers transfer Enable metrics on user functions

Output formats: table Format output as a table xml Format output as an XML file

7.1.15. Log

Scope: Sierra

`Log` *`Log-control-name`* `Every` *`Interval`*

Parameter	Value	Default
Log-control-name	`string`	-
Interval	`integer`	-

Summary Sets the maximum number of warnings before the execution is terminated.

7.1.16. Print Timer Information Every

Scope: Sierra

`Print Timer Information Every` *`Procedure-step-interval`* `Steps` *`Checkpointed`*

Parameter	Value	Default
Procedure-step-interval	`integer`	-
Checkpointed	`{accumulated|checkpointed}`	-

Summary Specifies the procedure step count interval to print timer information

7.2. AZTEC EQUATION SOLVER

Scope: Sierra

```
Begin Aztec Equation Solver Solver Name
    Bc Enforcement {=|are|is} BcEnforcement
    Debug Output Level {=|are|is} Level
    Debug Output Path {=|are|is} DebugOutput
    Determine Sharing {=|are|is} FeiDetermineSharing
    Fei Output Level {=|are|is} FeiOutputLevels
    Ilu Fill {=|are|is} ilu_fill_level
    Ilu Omega {=|are|is} Value
    Ilu Threshold {=|are|is} Threashold
    Matrix Format {=|are|is} MatrixFormat
    Matrix Reduction {=|are|is} AztecReductionType
    Matrix Scaling {=|are|is} MatrixScaling
    Maximum Iterations {=|are|is} max_iterations
```

```
    Num Levels {=|are|is} Num_levels

    Orthog Method {=|are|is} OrthogMethod

    Param-Int Parameter_name Value integer_value

    Param-Real Parameter_name Value Real_value

    Param-String Parameter_name Value String_value

    Polynomial Order {=|are|is} polynomial_order

    Preconditioner Subdomain Overlap {=|are|is} Overlap

    Preconditioning Method {=|are|is} AztecPrecondMethods

    Preconditioning Steps {=|are|is} preconditioning_steps

    Residual Norm Scaling {=|are|is} AztecResidualNormScaling

    Residual Norm Tolerance {=|are|is} residual_norm_tolerance

    Restart Iterations {=|are|is} restart_iterations

    Shared Ownership Rule {=|are|is} SharedOwnershipRule

    Solution Method {=|are|is} AztecSolverMethods

End
```

Summary A set of solver parameters for Aztec equation solver.

7.2.1. Bc Enforcement

Scope: Aztec Equation Solver

`Bc Enforcement {=|are|is} BcEnforcement`

Parameter	Value	Default
BcEnforcement	{exact|exact_no_column_mod|remove|solver|solver_no_column_mod}	-

Summary Controls the way Dirichlet BCs are enforced.

Description Valid values for this line-command are contained in the BcEnforcement enum.

7.2.2. Debug Output Level

Scope: Aztec Equation Solver

`Debug Output Level {=|are|is}` *`Level`*

Parameter	Value	Default
Level	`integer`	-

Summary Output level for debugging. Generally o means no solver screen output, and higher values of this parameter correspond to more screen output.

7.2.3. Debug Output Path

Scope: Aztec Equation Solver

`Debug Output Path {=|are|is}` *`DebugOutput`*

Parameter	Value	Default
DebugOutput	`string`	-

Summary Specify path where debug-logs and other debug output files will be placed.

7.2.4. Determine Sharing

Scope: Aztec Equation Solver

`Determine Sharing {=|are|is}` *`FeiDetermineSharing`*

Parameter	Value	Default
FeiDetermineSharing	`{fei|sierra}`	-

Summary Whether FEI should determine sharing internally or receive the info from Sierra.

Description By default the Eqns::LinearSystem class tells the fei layer which nodes are shared and which processors share them. If this option is set to on, then the fei layer internally determines the sharing info. Requires the fei layer to perform extra communication during the intialize phase.

7.2.5. Fei Output Level

Scope: Aztec Equation Solver

`Fei Output Level {=|are|is}` *`FeiOutputLevels`*

Parameter	Value	Default
FeiOutputLevels	`{all|brief_logs|full_logs|matrix_files|none|stats}`	-

Summary Control the amount of output produced by FEI.

Description Output level controls the amount of debugging information (screen output, matrix/vector files and log files) produced by FEI. The FeiOutputLevels enum contains the valid values for this line-command. The location of files can be controlled by the 'debug output path' line-command.

7.2.6. Ilu Fill

Scope: Aztec Equation Solver

`Ilu Fill {=|are|is}` *`ilu_fill_level`*

Parameter	Value	Default
ilu_fill_level	`integer`	-

Summary Fill-in parameter for incomplete factorizations.

7.2.7. Ilu Omega

Scope: Aztec Equation Solver

`Ilu Omega {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Omega parameter, dd-rilu uses ILU(k,w), w==omega

7.2.8. Ilu Threshold

Scope: Aztec Equation Solver

`Ilu Threshold {=|are|is}` *`Threashold`*

Parameter	Value	Default
Threashold	`real`	-

Summary Threshold parameter for incomplete factorizations.

7.2.9. Matrix Format

Scope: Aztec Equation Solver

`Matrix Format {=|are|is}` *`MatrixFormat`*

Parameter	Value	Default
MatrixFormat	`{csr|msr|vbr}`	-

Summary Storage format for the matrix.

Description This parameter only applies to Trilinos and Aztec.

7.2.10. Matrix Reduction

Scope: Aztec Equation Solver

`Matrix Reduction {=|are|is}` *`AztecReductionType`*

Parameter	Value	Default
AztecReductionType	`{fei-remove-slaves}`	-

Summary Remove constraint equations from matrix.

Description Dependent degrees of freedom in constraints are projected out of the equation space, yielding a linear system with smaller dimension, and retaining positive definiteness (whereas the alternative, an augmented matrix arising from the lagrange multiplier formulation, is indefinite).

7.2.11. Matrix Scaling

Scope: Aztec Equation Solver

`Matrix Scaling {=|are|is}` *`MatrixScaling`*

Parameter	Value	Default
MatrixScaling	`{block-jacobi|jacobi|none|row-sum|symmetric-diagonal|symmetric-row-sum|user supplied}`	-

Summary Scaling to be applied to the matrix.

Description Matrix scaling is a pre-solve operation, typically modifying the matrix in place, whereas preconditioning is performed at each iteration during the solve and doesn't modify the actual matrix.

7.2.12. Maximum Iterations

Scope: Aztec Equation Solver

`Maximum Iterations {=|are|is}` *`max_iterations`*

Parameter	Value	Default
max_iterations	`integer`	-

Summary Maximum number of solution method iterations.

7.2.13. Num Levels

Scope: Aztec Equation Solver

`Num Levels {=|are|is}` *`Num_levels`*

Parameter	Value	Default
Num_levels	`integer`	-

Summary Number of levels for multi-level/multi-grid solvers.

7.2.14. Orthog Method

Scope: Aztec Equation Solver

`Orthog Method {=|are|is}` *`OrthogMethod`*

Parameter	Value	Default
OrthogMethod	`{classical|modified}`	-

Summary User can choose orthogonalization method used by Aztec GMRES.

Description Valid values for this line-command are contained in the OrthogMethod enum.

7.2.15. Param-Int

Scope: Aztec Equation Solver

`Param-Int` *`Parameter_name`* `Value` *`integer_value`*

Parameter	Value	Default
Parameter_name	`"string"`	-
integer_value	`integer`	-

Summary String-Key/Integer-Value pair to be passed to solver.

Description Syntax: 'PARAM-INT "some-name" VALUE 123' Primarily used for passing values to the ML package. ML has a large number of parameters which haven't been individually incorporated into our parameter-parsing system. Can also be used to pass parameters to Aztec, e.g.:

PARAM-REAL "AZ_ilut_fill" VALUE 3

7.2.16. Param-Real

Scope: Aztec Equation Solver

`Param-Real` *`Parameter_name`* `Value` *`Real_value`*

Parameter	Value	Default
Parameter_name	`"string"`	-
Real_value	`real`	-

Summary String-Key/Real-Value pair to be passed to solver.

Description Syntax: 'PARAM-REAL "some-name" VALUE 1.23' Primarily used for passing values to the ML package. ML has a large number of parameters which haven't been individually incorporated into our parameter-parsing system. Can also be used to pass parameters to Aztec, e.g.:

PARAM-REAL "AZ_drop" VALUE 1.e-8

7.2.17. Param-String

Scope: Aztec Equation Solver

`Param-String` *`Parameter_name`* `Value` *`String_value`*

Parameter	Value	Default
Parameter_name	`"string"`	-
String_value	`"string"`	-

Summary Key/Value string-pair to be passed to solver.

Description Syntax: 'PARAM-STRING "some-name" VALUE "some-value"' (Primarily used for passing values to the ML package. ML has a large number of parameters which haven't been individually incorporated into our parameter-parsing system.)

7.2.18. Polynomial Order

Scope: Aztec Equation Solver

`Polynomial Order {=|are|is}` *`polynomial_order`*

Parameter	Value	Default
polynomial_order	integer	-

Summary Polynomial order of preconditioning method.

7.2.19. Preconditioner Subdomain Overlap

Scope: Aztec Equation Solver

`Preconditioner Subdomain Overlap {=|are|is}` *`Overlap`*

Parameter	Value	Default
Overlap	integer	-

Summary Ovrlap of Schwarz subdomains (eg, 0,1 or 2).

7.2.20. Preconditioning Method

Scope: Aztec Equation Solver

`Preconditioning Method {=|are|is}` *`AztecPrecondMethods`*

Parameter	Value	Default
AztecPrecondMethods	{dd-bilu | dd-icc | dd-ilu | dd-ilut | dd-lu | dd-rilu | jacobi | least-squares | neumann | none | symmetric-gauss-seidel}	-

Summary Selection of the equation preconditioning method.

Description Valid values for this are contained in the AztecPrecondMethods enum.

7.2.21. Preconditioning Steps

Scope: Aztec Equation Solver

`Preconditioning Steps {=|are|is}` *`preconditioning_steps`*

Parameter	Value	Default
preconditioning_steps	integer	-

Summary Number of Jacobi, Gauss-Seidel, or other preconditioning methods' applications per iteration.

7.2.22. Residual Norm Scaling

Scope: Aztec Equation Solver

`Residual Norm Scaling {=|are|is}` *`AztecResidualNormScaling`*

Parameter	Value	Default
AztecResidualNormScaling	`{anorm|none|r0|rhs}`	-

Summary Scaling method for the residual norm.

Description Determines the residual expression used in convergence checks and printing.

7.2.23. Residual Norm Tolerance

Scope: Aztec Equation Solver

`Residual Norm Tolerance {=|are|is}` *`residual_norm_tolerance`*

Parameter	Value	Default
residual_norm_tolerance	`real`	-

Summary Iterative solution method relative residual convergence tolerance.

7.2.24. Restart Iterations

Scope: Aztec Equation Solver

`Restart Iterations {=|are|is}` *`restart_iterations`*

Parameter	Value	Default
restart_iterations	`integer`	-

Summary Number of iterations between GMRES restarts.

7.2.25. Shared Ownership Rule

Scope: Aztec Equation Solver

`Shared Ownership Rule {=|are|is}` *`SharedOwnershipRule`*

Parameter	Value	Default

SharedOwnershipRule	`{low-numbered-proc|`
	`proc-with-local-elem|`
	`sierra_specifies}`
		-

Summary Controls the way owning processors are chosen for shared nodes in the FEI.

Description 'low-numbered-proc' is the default.

'proc-with-local-elem' is another valid value.

'sierra_specifies' is the other valid value

7.2.26. Solution Method

Scope: Aztec Equation Solver

`Solution Method {=|are|is}` *`AztecSolverMethods`*

Parameter	Value	Default
AztecSolverMethods	`{bicgstab|cg|cgs|gmres|lu|tfqmr}`	-

Summary Selection of the linear-system solution algorithm.

Description The AztecSolverMethods enum contains valid values for this line-command. They are mostly iterative methods (Krylov subspace algorithms), except for 'lu' which is a serial direct solver provided by the 'Y12M' library.

7.3. DEFINITION FOR FUNCTION

Scope: Sierra

`Begin Definition For Function` *`FunctionName`*

`Abscissa {=|are|is}` *`Name...`*

`Abscissa Offset {=|are|is}` *`Abscissa_offset`*

`Abscissa Scale {=|are|is}` *`Abscissa_scale`*

`At Discontinuity Evaluate To` *`Option`*

`Column Titles` *$Titles_1$* *$Titles_2$...*

`Data File =` *`filename`* `[ X From Column` *`xcol`* `Y From Column` *`ycol`* `]`

`Debug {=|are|is}` *`Option`*

`Differentiate Expression {=|are|is}` *`Expr`*

`Evaluate Expression {=|are|is}` *`Expr`*

`Evaluate From` *`x0`* `To` *`x1`* `By` *`Dx`*

`Expression Variable:` *`Expr = VarType value_var_name...`*

`Expression Variable:` *`Expr`*

`Ordinate {=|are|is}` *`Name...`*

```
    Ordinate Offset {=|are|is} Ordinate_offset
    Ordinate Scale {=|are|is} Ordinate_scale
    Scale By x
    Type {=|are|is} Type
    X Offset {=|are|is} X_offset
    X Scale {=|are|is} X_scale
    Y Offset {=|are|is} Y_offset
    Y Scale {=|are|is} Y_scale
    Begin Expressions empty
    End

    Begin Values empty
    End

End
```

Summary Defines a function in terms of its type and values.

7.3.1. Abscissa

Scope: Definition For Function

```
Abscissa {=|are|is} Name...
```

Parameter	Value	Default
Name	`string...`	-

Summary Specifies a string identifier for the independent variable. Optionally specify a scale and/or offset value which transforms the abscissa values into scaled_abscissa = scale * (abscissa + abscissa_offset).

7.3.2. Abscissa Offset

Scope: Definition For Function

```
Abscissa Offset {=|are|is} Abscissa_offset
```

Parameter	Value	Default
Abscissa_offset	`real`	-

Summary Alias for X OFFSET

7.3.3. Abscissa Scale

Scope: Definition For Function

`Abscissa Scale {=|are|is}` *`Abscissa_scale`*

Parameter	Value	Default
Abscissa_scale	`real`	-

Summary Alias for X SCALE

7.3.4. At Discontinuity Evaluate To

Scope: Definition For Function

`At Discontinuity Evaluate To` *`Option`*

Parameter	Value	Default
Option	`{left|right}`	-

Summary Control the behavior of a piecewise constant function when evaluated at a discontinuity (plus or minus a small tolerance). The default behavior is to take the value to the right of the discontinuity. If "Left" is specified, the value to the left of the discontinuity is taken instead.

7.3.5. Column Titles

Scope: Definition For Function

`Column Titles` $Titles_1$ $Titles_2$`...`

Parameter	Value	Default
Titles	`string_1 string_2...`	-

Summary Name the columns (and also defined the expected number of columns) for Multicolumn Piecewise Linear tabular data.

7.3.6. Data File

Scope: Definition For Function

`Data File =` *`filename`* `[ X From Column` *`xcol`* `Y From Column` *`ycol`* `]`

Parameter	Value	Default
filename	`string`	-

Summary Function will read tabular data from an input file. Compatible with the piecewise linear function type. File must be of form like:

——————————————————— # EXAMPLE FILE 1.099 1191 1.101 221 5.9011 133.1
———————————————————

Lines headed by a # are considered comments and will be ignored. Data itself must by in tabular columns separated by whitespace or commas.

7.3.7. Debug

Scope: Definition For Function

`Debug {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`{off|on}`	-

Summary Prints functions to the log file.

7.3.8. Differentiate Expression

Scope: Definition For Function

`Differentiate Expression {=|are|is}` *`Expr`*

Parameter	Value	Default
Expr	`(expression)`	-

Summary Specifies the expression of derivative of evaluation expression.

7.3.9. Evaluate Expression

Scope: Definition For Function

`Evaluate Expression {=|are|is}` *`Expr`*

Parameter	Value	Default
Expr	`(expression)`	-

Summary Specifies the expression to evaluate.

Description This will greatly help with manufactured solutions, and be useful for other purposes as well.

This first implementation goes like this:

```
begin definition for function pressure
type is analytic
evaluate expression is "x <= 0.0 ?  0.0 :  (x < 0.5 ?
x*200.0 :  (x < 1.0 ?  (x - 0.5) *50.0 + 100.00 :
150.0));"
# type is piecewise linear
# begin values
# 0.0 0.0
# 0.5 100.0
# 1.0 150.0
# end values
end definition for function pressure
```

Also, notice that semicolon at the end. Be sure to put it there for now. You can actually provide multiple expressions to be evaluated, each terminated with a semicolon. This will be handy when multi-dependent variable come into the fold.

The following functions are currently implemented.

Operators All C-language operators are supported, e.g. $+ - */ \| \| ? :$ etc

Parens ()

Math Functions

abs(x) absolute value of x

mod(x, y) modulus of x|y

ipart(x) integer part of x

fpart(x) fractional part of x

min(x0, x1, ...) minimum value of xn

max(x0, x1, ...) maximum value of xn

Power functions

pow(x, y) x to the y power

sqrt(x) square root of x

Trig functions

sin(x) sine of x

sinh(x) hyperbolic sine of x

asin(x) arcsine of x

cos(x) cosine of x

cosh(x) hyperbolic cosine of x

acos(x) arccosine of x

tan(x) tangent of x

tanh(x) hyperbolic tangent of x

atan(x) arctangent of x

atan2(y, x) arctangent of y/x, signs of x and y determine quadrant (see atan2 man page)

Logarithm functions

log(x) natural logarithm of x

ln(x) natural logarithm of x

exp(x) e to the x power

logn(x, y) the y base logarithm of x

Rounding functions

ceil(x) smallest integral value not less than x

floor(x) largest integral value not greater than x

Random functions

rand(x) random number between 0.0 and 1.0, not including 1.0

srand(x) seeds the random number generator

Conversion routines

deg(x) converts radians to degrees

rad(x) converts degrees to radians

recttopolr(x, y) maginitude of vector x, y

recttopola(x, y) angle of vector x, y

poltorectx(r, theta) x coordinate of angle theta at distance r

poltorecty(r, theta) y coordinate of angle theta at distance r

7.3.10. Evaluate From

Scope: Definition For Function

`Evaluate From` *`x0`* `To` *`x1`* `By` *`Dx`*

Parameter	Value	Default
x0	`real`	-
x1	`real`	-
Dx	`real`	-

Summary Specifies the range and evaluation interval.

7.3.11. Expression Variable:

Scope: Definition For Function

`Expression Variable:` *`Expr = VarType value_var_name...`*

Parameter	Value	Default
Expr	`string`	-
VarType	`{element | element_sym_tensor | element_tensor | element_vector | face | global | nodal | nodal_sym_tensor | nodal_tensor | nodal_vector}`	-
value_var_name	`string...`	-

Summary Specifies what the arguments of an expression correspond to. For example:

BEGIN DEFINITION FOR FUNCTION dx_shear TYPE = ANALYTIC EXPRESSION variable: mx = NODAL model_coordinates(x) EXPRESSION variable: my = NODAL model_coordinates(y) EXPRESSION variable: time = GLOBAL time EVALUATE EXPRESSION = "(time/termTime)*(stretchx*(mx - 0.0) + ((my-0.25)/0.5)*stretchxy)" END

Assuming the above expression is being evaluated on nodes the current values for x and y model coordinates would be placed into mx and my and current analysis time placed into time

7.3.12. Expression Variable:

Scope: Definition For Function

`Expression Variable:` *`Expr`*

Parameter	Value	Default
Expr	`string`	-

Summary Specifies what the arguments of an expression exists, but does not define it correspond to. For example:

BEGIN DEFINITION FOR FUNCTION dx_shear TYPE = ANALYTIC EXPRESSION variable: mx EXPRESSION variable: my EXPRESSION variable: time EVALUATE EXPRESSION = "(time/termTime)*(stretchx*(mx - 0.0) + ((my-0.25)/0.5)*stretchxy)" END

Call function must determine what each variable actually is is based off of the string name

7.3.13. Ordinate

Scope: Definition For Function

`Ordinate {=|are|is}` *`Name...`*

Parameter	Value	Default
Name	`string...`	-

Summary Specifies a string identifier for the dependent variable. Optionally specify a scale and/or offset value which transforms the ordinate values into scaled_ordinate = scale * (ordinate + ordinate_offset).

7.3.14. Ordinate Offset

Scope: Definition For Function

`Ordinate Offset {=|are|is}` *`Ordinate_offset`*

Parameter	Value	Default
Ordinate_offset	`real`	-

Summary Alias for Y OFFSET

7.3.15. Ordinate Scale

Scope: Definition For Function

`Ordinate Scale {=|are|is}` *`Ordinate_scale`*

Parameter	Value	Default
Ordinate_scale	`real`	-

Summary Alias for Y SCALE

7.3.16. Scale By

Scope: Definition For Function

`Scale By` *x*

Parameter	Value	Default
x	`real`	-

Summary Specifies a scale factor to be applied.

7.3.17. Type

Scope: Definition For Function

`Type {=|are|is}` *Type*

Parameter	Value	Default
Type	`{analytic|constant|multicolumn piecewise linear|piecewise analytic| piecewise constant|piecewise linear| piecewise multivariate|xtable}`	-

Summary Specifies the type of function.

7.3.18. X Offset

Scope: Definition For Function

`X Offset {=|are|is}` *X_offset*

Parameter	Value	Default
X_offset	`real`	-

Summary Sets an offset for the x-axis

7.3.19. X Scale

Scope: Definition For Function

`X Scale {=|are|is}` *X_scale*

Parameter	Value	Default
X_scale	`real`	-

Summary Sets a scale factor for the x-axis

7.3.20. Y Offset

Scope: Definition For Function

`Y Offset {=|are|is}` *`Y_offset`*

Parameter	Value	Default
Y_offset	`real`	-

Summary Sets an offset for the y-axis

7.3.21. Y Scale

Scope: Definition For Function

`Y Scale {=|are|is}` *`Y_scale`*

Parameter	Value	Default
Y_scale	`real`	-

Summary Sets a scale factor for the y-axis

7.4. VALUES

Scope: Definition For Function

```
Begin Values empty
    Xyvalues...
End
```

Summary Lists the values of the function. The values should be listed one pair per line, independent variable first, with whitespace or comma as a separator.

7.4.1.

Scope: Values

`Xyvalues...`

Parameter	Value	Default
Xyvalues	`real...`	-

Summary For a piecewise linear function, lists an x-y pair for the nth interpolation point.

7.5. FINITE ELEMENT MODEL

Scope: Sierra

```
Begin Finite Element Model Label
    Alias DatabaseName As InternalName
    Component Separator Character Option Separator
    Create GroupType NewSurfaceName Add SurfaceName...
    Coordinate System {=|are|is} CoordinateSystem
    Database Name {=|are|is} StreamName
    Database Type {=|are|is} DatabaseTypes
    Decomposition Method {=|are|is} Method
    Omit Block BlockList...
    Omit Volume VolumeList...
    Time Scale Factor Option Scale
    Use Generic Names
    Use Material MaterialName For VolumeList...
    Begin Parameters For Block Blockname
    End
End
```

Summary Describes the location and type of the input stream used for defining a geometry model for the enclosing region.

7.5.1. Alias

Scope: Finite Element Model

`Alias DatabaseName As InternalName`

Parameter	Value	Default
DatabaseName	string	-
InternalName	string	-

Summary Name the database entity "DatabaseName" as "InternalName"

Description This "InternalName" may then be referenced in the data file in addition to the original name.

7.5.2. Component Separator Character

Scope: Finite Element Model

`Component Separator Character` *`Option Separator`*

Parameter	Value	Default
Option	`{= | is}`	-
Separator	`string`	-

Summary The separator is the single character used to separate the output variable basename (e.g. "stress") from the suffices (e.g. "xx", "yy") when displaying the names of the individual variable components. For example, the default separator is "_", which results in names similar to "stress_xx", "stress_yy", ... "stress_zx". To eliminate the separator, specify an empty string ("") or NONE.

7.5.3. Create

Scope: Finite Element Model

`Create` *`GroupType NewSurfaceName`* `Add` *`SurfaceName...`*

Parameter	Value	Default
GroupType	`{edgeset | elemset | faceset | nodeset | sideset | surface}`	-
NewSurfaceName	`string`	-
SurfaceName	`string...`	-

Summary Create a new set (node, edge, face, element, side/surface) as the union of two or more existing sets. The sets must exist in the mesh database or have been created by a previous CREATE command.

7.5.4. Coordinate System

Scope: Finite Element Model

`Coordinate System {= | are | is}` *`CoordinateSystem`*

Parameter	Value	Default
CoordinateSystem	`{axisymmetric | barycentric | cartesian | cyclidic | cylindrical | polar | quadriplanar | skew | spherical | toroidal | trilinear}`	-

Summary The interpretation of the geometry data stored in this database. Optional. Defaults to Cartesian.

7.5.5. Database Name

Scope: Finite Element Model

`Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The base name of the database containing the output results. If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. If this line is omitted, then a filename will be created from the basename of the input file with a ".g" suffix appended.

7.5.6. Database Type

Scope: Finite Element Model

`Database Type {=|are|is}` *`DatabaseTypes`*

Parameter	Value	Default
DatabaseTypes	`{catalyst|dof|dof_exodus|exodus|exodusii|generated|genesis|parallel_exodus}`	-

Summary The database type/format used for the mesh.

7.5.7. Decomposition Method

Scope: Finite Element Model

`Decomposition Method {=|are|is}` *`Method`*

Parameter	Value	Default
Method	`{block|cyclic|external|geom_kway|hsfc|kway|kway_geom|linear|metis_sfc|random|rcb|rib}`	-

Summary The decomposition algorithm to be used to partition elements to each processor in a parallel run.

7.5.8. Omit Block

Scope: Finite Element Model

`Omit Block` *`BlockList...`*

Parameter	Value	Default
BlockList	`string...`	-

Summary Specifies that the element blocks named in the blockList be omitted from the analysis.

Description If an element block is omitted, then it is illegal to refer to it later in the input file e.g an initial condition may not be specified on an omitted element block. The elements, faces, etc are never created and it is as if the omitted element blocks did not exist in the mesh file. If a surface is completely determined by the omitted element block, then it is illegal to specify boundary conditions on that surface. However, if the surface spans multiple element blocks, boundary conditions may be applied on the portion of the surface supported by the element blocks that are not omitted.

7.5.9. Omit Volume

Scope: Finite Element Model

`Omit Volume` *`VolumeList...`*

Parameter	Value	Default
VolumeList	`string...`	-

Summary Specifies that the volumes named in the volumeList be omitted from the analysis.

Description If a volume is omitted, then it is illegal to refer to it later in the input file e.g an initial condition may not be specified on an omitted volume. The elements, faces, etc are never created and it is as if the omitted volumes did not exist in the mesh file. If a surface is completely determined by the omitted volume, then it is illegal to specify boundary conditions on that surface. However, if the surface spans multiple volumes, boundary conditions may be applied on the portion of the surface supported by the volumes that are not omitted.

7.5.10. Time Scale Factor

Scope: Finite Element Model

`Time Scale Factor` *`Option Scale`*

Parameter	Value	Default
Option	`{=|is}`	-
Scale	`real`	-

Summary The scale factor to be applied to the times on the mesh database. If the scale factor is 20 and the times on the mesh database are 0.1, 0.2, 0.3, then the application will see the mesh times as 2, 4, 6.

7.5.11. Use Generic Names

Scope: Finite Element Model

Summary If this command is present then the name of all blocks and sets in the mesh will be of the form "type_"+id. For example, an element block with id=42 will be named "block_42"; a sideset with id 314 will be named "surface_314". If there are any names in the mesh file, those names will be aliases for the blocks and sets. If this command is not present, then if a name is in the mesh file, it will be used as the name and the generic generated name will be an alias. This is used as a workaround in codes that do not correctly handle named blocks and sets or as a workaround in meshes which contain non-user-specified names.

7.5.12. Use Material

Scope: Finite Element Model

`Use Material` *`MaterialName`* `For` *`VolumeList...`*

Parameter	Value	Default
MaterialName	`string`	-
VolumeList	`string...`	-

Summary Associate the given volumes with the indicated material name.

7.6. PARAMETERS FOR BLOCK

Scope: Finite Element Model

`Begin Parameters For Block` *`Blockname`*

`Active For Procedure` *`ProcedureName`* `During Periods` *`PeriodNames...`*

`Bending Hourglass` *`Option`* `{=|are|is}` *`Hgval`*

`Density Scale Factor {=|are|is}` *`densityScaleFactor`*

`Deposit Specific Internal Energy` *`Edep`* `[ Over Time` *`Tdep`* `Starting At Time` *`Tinit`* `]`

`Effective Moduli Model {=|are|is}` *`Option`*

`Element Numerical Formulation {=|are|is}` *`Option`*

`Energy Iteration Tolerance {=|are|is}` *`Eit`*

Hourglass *Option* {=|are|is} *Hgval*

Inactive For Procedure *ProcedureName* During Periods *PeriodNames...*

Include All Blocks

Inversion Aversion Exponent {=|are|is} *ia_exponent*

Inversion Aversion Stiffness {=|are|is} *ia_stiffness*

Inversion Aversion Transition Jacobian {=|are|is} *transition_jacobian*

Linear Bulk Viscosity {=|are|is} *Lbv*

Material *MatName*

Material = *MatName*

Max Energy Iterations {=|are|is} *Mei*

Membrane Hourglass *Option* {=|are|is} *Hgval*

Minimum Effective Dilatational Moduli Ratio {=|are|is} *minEffectiveModuliRatio*

Minimum Effective Shear Moduli Ratio {=|are|is} *minEffectiveModuliRatio*

Model {=|are|is} *ModelName*

Nonlocal Regularization Kmeans Cell Size {=|are|is} *kmeans_cell_size*

Nonlocal Regularization Kmeans Maximum Iterations {=|are|is} *kmeans_maximum_iterations*

Nonlocal Regularization Kmeans Tolerance {=|are|is} *kmeans_tolerance*

Nonlocal Regularization On *stateVariableName* With Length Scale {=|are|is} *lengthScale* [And Staggering]

Nonlocal Regularization Partitioning Scheme {=|are|is} *PartitioningScheme*

Phase *PhaseLabel* {=|are|is} *MaterialName*

Quadratic Bulk Viscosity {=|are|is} *Qbv*

Remove Block {=|are|is} *ExcludeBlockList...*

Section {=|are|is} *SectionName*

Solid Mechanics Use Model *ModelName*

```
Transverse Shear Hourglass Option {=|are|is} Hgval
End
```

Summary Specifies analysis parameters associated with each element block.

7.6.1. Active For Procedure

Scope: Parameters For Block

```
Active For Procedure ProcedureName During Periods PeriodNames...
```

Parameter	Value	Default
ProcedureName	`string`	-
PeriodNames	`string...`	-

Summary Lists the solution periods during which the given BC, solver, preconditioner, etc. is active. Multiple uses of this line command within a single block will have a cumulative affect.

7.6.2. Bending Hourglass

Scope: Parameters For Block

```
Bending Hourglass Option {=|are|is} Hgval
```

Parameter	Value	Default
Option	`{stiffness|viscosity}`	-
Hgval	`real`	-

Summary Supplies the hourglass stiffness and viscosity parameters for bending deformation in a shell element block.

7.6.3. Density Scale Factor

Scope: Parameters For Block

```
Density Scale Factor {=|are|is} densityScaleFactor
```

Parameter	Value	Default
densityScaleFactor	`real`	-

Summary Specifies a scale factor to apply to the density defined in the material. This value must be greater than zero. The default is 1.0 (no scaling).

7.6.4. Deposit Specific Internal Energy

Scope: Parameters For Block

`Deposit Specific Internal Energy` *`Edep`* `[ Over Time` *`Tdep`* `Starting At Time` *`Tinit`* `]`

Parameter	Value	Default
Edep	`real`	-

Summary Defines the amount of specific (per unit mass) internal energy to be deposited in the material. The energy is deposited over time tdep, beginning at time tinit. The optional parameters tdep and tinit both default to zero, so the energy will be deposited instantaneously at time zero if they are not specified. The deposition is uniform in space, so each element in the block has the same amount edep added to its specific internal energy.

7.6.5. Effective Moduli Model

Scope: Parameters For Block

`Effective Moduli Model {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`{elastic|probed|pronto}`	-

Summary Specifies the method used to determine the effective moduli. This choice can have a significant effect on the resulting hourglassing behavior. The models are: elastic: use the initial elastic moduli pronto: use the old PRONTO method for computing elastic moduli this approach is straight out of PRONTO, PRESTO's predecessor. This is a bounded tangent method. probe: Use a pronto-like method, but pass in a an artificial probe strain rate rather than the actual strain.

7.6.6. Element Numerical Formulation

Scope: Parameters For Block

`Element Numerical Formulation {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`{new|old}`	-

Summary Specifies which element numerical formulation to use for this block.

7.6.7. Energy Iteration Tolerance

Scope: Parameters For Block

`Energy Iteration Tolerance {=|are|is}` *`Eit`*

Parameter	Value	Default
Eit	`real`	-

Summary Specifies the tolerance criteria for exiting the iterative solve of the implicit internal energy update equation. Applicable when using EOS material models with extracted energy updates.

7.6.8. Hourglass

Scope: Parameters For Block

`Hourglass` *`Option`* `{=|are|is}` *`Hgval`*

Parameter	Value	Default
Option	`{exponent|stiffness|transition strain|viscosity}`	-
Hgval	`real`	-

Summary Supplies the hourglass stiffness and viscosity parameters for this element block.

7.6.9. Inactive For Procedure

Scope: Parameters For Block

`Inactive For Procedure` *`ProcedureName`* `During Periods` *`PeriodNames...`*

Parameter	Value	Default
ProcedureName	`string`	-
PeriodNames	`string...`	-

Summary Lists the solution periods during which the given BC, solver, preconditioner, etc. is inactive. Multiple uses of this line command within a single block will have a cumulative affect.

7.6.10. Include All Blocks

Scope: Parameters For Block

Summary Use this parameters definition for all blocks.

When using this option within the FINITE ELEMENT MODEL command block the PARAMETERS FOR BLOCK will not use a Blockname.

7.6.11. Inversion Aversion Exponent

Scope: Parameters For Block

`Inversion Aversion Exponent {=|are|is}` *`ia_exponent`*

Parameter	Value	Default
ia_exponent	`integer`	`5`

Summary Sets the exponent used to compute the smooth approximate nodal jacobian ratio. A higher exponent results in a more-accurate approximation to the ratio. This is only active for uniform gradient elements. Default = 5.

7.6.12. Inversion Aversion Stiffness

Scope: Parameters For Block

`Inversion Aversion Stiffness {=|are|is}` *`ia_stiffness`*

Parameter	Value	Default
ia_stiffness	`real`	`1.e5`

Summary Sets a stiffness parameter for the inversion aversion penalty. This is only active for uniform gradient elements. Default = 1.0e5.

7.6.13. Inversion Aversion Transition Jacobian

Scope: Parameters For Block

`Inversion Aversion Transition Jacobian {=|are|is}` *`transition_jacobian`*

Parameter	Value	Default
transition_jacobian	`real`	`0`

Summary Sets the critical relative nodal Jacobian ratio for inversion aversion. If this value is nonzero, an additional recoverable energy term is added which penalizes further element distortion. This energy is only active for uniform gradient elements.

7.6.14. Linear Bulk Viscosity

Scope: Parameters For Block

`Linear Bulk Viscosity {=|are|is}` *`Lbv`*

Parameter	Value	Default
Lbv	`real`	-

Summary Supplies the linear coefficient for the bulk viscosity computations.

7.6.15. Material

Scope: Parameters For Block

`Material` *`MatName`*

Parameter	Value	Default
MatName	`string`	-

Summary Associates this element block with its material properties.

7.6.16. Material =

Scope: Parameters For Block

`Material =` *`MatName`*

Parameter	Value	Default
MatName	`string`	-

Summary Associates this element block with its material properties.

7.6.17. Max Energy Iterations

Scope: Parameters For Block

`Max Energy Iterations {=|are|is}` *`Mei`*

Parameter	Value	Default
Mei	`integer`	-

Summary Specifies the maximum number of iterations to take in solving the implicit internal energy update equation. Applicable when using EOS material models with extracted energy updates.

7.6.18. Membrane Hourglass

Scope: Parameters For Block

`Membrane Hourglass` *`Option`* `{=|are|is}` *`Hgval`*

Parameter	Value	Default
Option	`{stiffness|viscosity}`	-
Hgval	`real`	-

Summary Supplies the hourglass stiffness and viscosity parameters for membrane deformation in a shell or membrane element block.

7.6.19. Minimum Effective Dilatational Moduli Ratio

Scope: Parameters For Block

`Minimum Effective Dilatational Moduli Ratio {=|are|is}` *`minEffectiveModuliRatio`*

Parameter	Value	Default
minEffectiveModuliRatio	`real`	-

Summary Specifies a minimum effective DILATATIONAL moduli ratio. This value keeps the effective moduli from dropping below minEffectiveModuliRatio*ElasticModulus. This can aid in keeping the corresponding time step and bulk viscosity terms dropping to zero

7.6.20. Minimum Effective Shear Moduli Ratio

Scope: Parameters For Block

`Minimum Effective Shear Moduli Ratio {=|are|is}` *`minEffectiveModuliRatio`*

Parameter	Value	Default
minEffectiveModuliRatio	`real`	-

Summary Specifies a minimum effective SHEAR moduli ratio. This value keeps the effective moduli from dropping below minEffectiveModuliRatio*ElasticModulus. This can aid in keeping the corresponding hourglass stiffness terms dropping to zero

7.6.21. Model

Scope: Parameters For Block

`Model {=|are|is}` *`ModelName`*

Parameter	Value	Default
ModelName	`string`	-

Summary Associates a solid mechanics material model with this element block. The material parameters for this block are specified in the material denoted by the MATERIAL command.

7.6.22. Nonlocal Regularization Kmeans Cell Size

Scope: Parameters For Block

`Nonlocal Regularization Kmeans Cell Size {=|are|is}` *`kmeans_cell_size`*

Parameter	Value	Default
kmeans_cell_size	`real`	-

Summary This line command specifies the cell size used to construct the background grid for the computation of the centroidal Voronoi tesselation for the Kmeans partitioning scheme.

7.6.23. Nonlocal Regularization Kmeans Maximum Iterations

Scope: Parameters For Block

`Nonlocal Regularization Kmeans Maximum Iterations {=|are|is}`
`kmeans_maximum_iterations`

Parameter	Value	Default
kmeans_maximum_iterations	`integer`	-

Summary This line command specifies the maximum number of iterations to perform for Lloyd's algorithm for the computation of the centroidal Voronoi tesselation for the Kmeans partitioning scheme.

7.6.24. Nonlocal Regularization Kmeans Tolerance

Scope: Parameters For Block

`Nonlocal Regularization Kmeans Tolerance {=|are|is}` *`kmeans_tolerance`*

Parameter	Value	Default
kmeans_tolerance	`real`	-

Summary This line command specifies the relative tolerance for Lloyd's algorithm. Iterations continue until the maximum number is reached or the L2 norm of a vector of all the center steps is less or equal than the tolerance times the cell size of the background grid.

7.6.25. Nonlocal Regularization On

Scope: Parameters For Block

`Nonlocal Regularization On` *`stateVariableName`* `With Length Scale {=|are|is}` *`lengthScale`* `[ And Staggering ]`

Parameter	Value	Default
stateVariableName	`string`	-
lengthScale	`real`	-

Summary This line command will cause the mesh to be partitioned into sub domains where each sub domain volume is on the order of $lengthScale^3$ and regularizes the governing PDE by averaging the material state variable stateVariableName over the sub domain.

7.6.26. Nonlocal Regularization Partitioning Scheme

Scope: Parameters For Block

`Nonlocal Regularization Partitioning Scheme {=|are|is}` *`PartitioningScheme`*

Parameter	Value	Default
PartitioningScheme	`{kmeans|metis|zoltan_hypergraph|zoltan_rcb|zoltan_rib}`	-

Summary This line command specifies the type of partitioning algorithm used to perform the domain decomposition for the nonlocal regularization method

7.6.27. Phase

Scope: Parameters For Block

`Phase` *`PhaseLabel`* `{=|are|is}` *`MaterialName`*

Parameter	Value	Default
PhaseLabel	`string`	-
MaterialName	`string`	-

Summary Associate phase PhaseLabel with material Material_Name on this block.

7.6.28. Quadratic Bulk Viscosity

Scope: Parameters For Block

`Quadratic Bulk Viscosity {=|are|is}` *`Qbv`*

Parameter	Value	Default
Qbv	`real`	-

Summary Supplies the quadratic coefficient for the bulk viscosity computations.

7.6.29. Remove Block

Scope: Parameters For Block

`Remove Block {=|are|is}` *`ExcludeBlockList...`*

Parameter	Value	Default
ExcludeBlockList	`string...`	-

Summary List of blocks to exclude.

7.6.30. Section

Scope: Parameters For Block

`Section {=|are|is}` *`SectionName`*

Parameter	Value	Default
SectionName	`string`	-

Summary Specifies the section to use for this element block.

7.6.31. Solid Mechanics Use Model

Scope: Parameters For Block

`Solid Mechanics Use Model` *`ModelName`*

Parameter	Value	Default
ModelName	`string`	-

Summary Associates a solid mechanics material model with this element block. The material parameters for this block are specified in the material denoted by the MATERIAL command.

7.6.32. Transverse Shear Hourglass

Scope: Parameters For Block

`Transverse Shear Hourglass` *`Option`* `{=|are|is}` *`Hgval`*

Parameter	Value	Default
Option	`{stiffness|viscosity}`	-
Hgval	`real`	-

Summary Supplies the hourglass stiffness and viscosity parameters for transverse shear deformation in a shell element block.

7.7. GLOBAL CONSTANTS

Scope: Sierra

```
Begin Global Constants empty
    Gravity Vector {=|are|is} Gravity_1 Gravity_2 Gravity_3
    Ideal Gas Constant {=|are|is} Sigma
    K-E Turbulence Model Parameter Param {=|are|is} Value
    K-W Turbulence Model Parameter Param {=|are|is} Value
    Les Turbulence Model Parameter Param {=|are|is} Value
    Light Speed {=|are|is} LightSpeed
    Planck Constant {=|are|is} PlanckConstant
    Stefan Boltzmann Constant {=|are|is} Sigma
    Turbulence Model Param Number {=|are|is} Value
End
```

Summary Set of universal constants for a simulation.

7.7.1. Gravity Vector

Scope: Global Constants

`Gravity Vector {=|are|is}` *Gravity*$_1$ *Gravity*$_2$ *Gravity*$_3$

Parameter	Value	Default
Gravity	real_1 real_2 real_3	-

Summary Gravity constant in vector form, acceleration components.

7.7.2. Ideal Gas Constant

Scope: Global Constants

`Ideal Gas Constant {=|are|is}` *Sigma*

Parameter	Value	Default
Sigma	real	-

Summary Ideal gas constant. extbfNOTE: Another ideal gas constant value can be specified while using certain code capabilities. This global constants value will be discarded for any other specified ideal gas constant values.

7.7.3. K-E Turbulence Model Parameter

Scope: Global Constants

`K-E Turbulence Model Parameter` *`Param`* `{=|are|is}` *`Value`*

Parameter	Value	Default
Param	`string`	-
Value	`real`	-

Summary $k-\epsilon$ RANS turbulence model parameters.

7.7.4. K-W Turbulence Model Parameter

Scope: Global Constants

`K-W Turbulence Model Parameter` *`Param`* `{=|are|is}` *`Value`*

Parameter	Value	Default
Param	`string`	-
Value	`real`	-

Summary $k-\omega$ RANS turbulence model parameters.

7.7.5. Les Turbulence Model Parameter

Scope: Global Constants

`Les Turbulence Model Parameter` *`Param`* `{=|are|is}` *`Value`*

Parameter	Value	Default
Param	`string`	-
Value	`real`	-

Summary LES turbulence model parameters.

7.7.6. Light Speed

Scope: Global Constants

`Light Speed {=|are|is}` *`LightSpeed`*

Parameter	Value	Default
LightSpeed	`real`	-

Summary Speed of Light. Depending on the units involved in the specific problem by the user, this value will differ.

7.7.7. Planck Constant

Scope: Global Constants

`Planck Constant {=|are|is}` *`PlanckConstant`*

Parameter	Value	Default
PlanckConstant	`real`	-

Summary Planck Constant. Depending on the units involved in the specific problem by the user, this value will differ.

7.7.8. Stefan Boltzmann Constant

Scope: Global Constants

`Stefan Boltzmann Constant {=|are|is}` *`Sigma`*

Parameter	Value	Default
Sigma	`real`	-

Summary Stefan-Boltzmann constant. Depending on the units involved in the specific problem by the user, this value will differ.

7.7.9. Turbulence Model

Scope: Global Constants

`Turbulence Model` *`Param`* `Number {=|are|is}` *`Value`*

Parameter	Value	Default
Param	`string`	-
Value	`real`	-

Summary Turbulence model Schmidt and Prandtl numbers

7.8. OUTPUT SCHEDULER

Scope: Sierra

```
Begin Output Scheduler Label

    Additional Steps {=|are|is} List_of_steps...

    Additional Times {=|are|is} List_of_times...

    At Step n Option {=|are|is} m

    At Time Dt1 Option {=|are|is} Dt2
```

```
    Output On Signal {=|are|is} Signals

    Start Time {=|are|is} Start_time

    Synchronize Output

    Termination Time {=|are|is} Final_time

    Timestep Adjustment Interval {=|are|is} Nsteps

    Use Output Scheduler Timer_name

End
```

Summary Defines an output scheduler at the domain level which can be used by one or more output blocks (restart, results, heartbeat, history> at region scope. If used by multiple output blocks, their output will be synchronized.

7.8.1. Additional Steps

Scope: Output Scheduler

```
Additional Steps {=|are|is} List_of_steps...
```

Parameter	Value	Default
List_of_steps	`integer...`	-

Summary Additional simulation steps when output should occur.

7.8.2. Additional Times

Scope: Output Scheduler

```
Additional Times {=|are|is} List_of_times...
```

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

7.8.3. At Step

Scope: Output Scheduler

`At Step` *`n Option`* `{=|are|is}` *`m`*

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

7.8.4. At Time

Scope: Output Scheduler

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

7.8.5. Output On Signal

Scope: Output Scheduler

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

7.8.6. Start Time

Scope: Output Scheduler

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

7.8.7. Synchronize Output

Scope: Output Scheduler

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

7.8.8. Termination Time

Scope: Output Scheduler

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

7.8.9. Timestep Adjustment Interval

Scope: Output Scheduler

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	integer	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

7.8.10. Use Output Scheduler

Scope: Output Scheduler

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	string	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

7.9. PROPERTY SPECIFICATION FOR FUEGO MATERIAL

Scope: Sierra

`Begin Property Specification For Fuego Material` *`MaterialName`*

`Properties` `{=|are|is}` *`Value`*

`Bulk Chemistry Description {=|are|is}` *`Value`*

`Cantera Xml File {=|are|is}` *`FileName`*

`Chemkin Linking File {=|are|is}` *`FileName`*

`Chemkin Transport File {=|are|is}` *`FileName`*

`Datum Pressure {=|are|is}` *`Value`*

`Do Not Clip Mixture Fraction Variables`

`Function For` *`Properties`* `{=|are|is}` *`FuncName`*

`Function For Mass_Diffusivity` *`SpeciesName`* `{=|are|is}` *`FuncName`*

`Function For Species_Density` *`SpeciesName`* `{=|are|is}` *`FuncName`*

`Function For Species_Emissivity` *`SpeciesName`* `{=|are|is}` *`FuncName`*

Function For Species_Enthalpy *SpeciesName* {=|are|is} *FuncName*

Function For Species_Production_Rate *SpeciesName* {=|are|is} *FuncName*

Function For Species_Specific_Heat *SpeciesName* {=|are|is} *FuncName*

Function For Species_Thermal_Conductivity *SpeciesName* {=|are|is} *FuncName*

Input Variables For *SubName* {=|are|is} *Variable_list...*

Mass_Diffusivity *SpeciesName* {=|are|is} *Value*

Molecular_Weight *SpeciesName* {=|are|is} *Value*

Prandtl_Number {=|are|is} *Value*

Real Data For *SubName* {=|are|is} *Parameter_list...*

Reference *Variable* {=|are|is} *Value*

Reference Mass_Fraction *SpeciesName* {=|are|is} *Value*

Reference Mole_Fraction *SpeciesName* {=|are|is} *Value*

Reference Progress_Variable *ProgressVariableName* With Lower Bound *LowerBound* And Upper Bound *UpperBound*

Set Schmidt Number For Progress Variable *ProgressVariableName* {=|are|is} *Value*

Subroutine For *PropertyName* {=|are|is} *SubName*

Schmidt_Number {=|are|is} *Value*

Species_Density *SpeciesName* {=|are|is} *Value*

Species_Emissivity *SpeciesName* {=|are|is} *Value*

Species_Enthalpy *SpeciesName* {=|are|is} *Value*

Species_Production_Rate *SpeciesName* {=|are|is} *Value*

Species_Specific_Heat *SpeciesName* {=|are|is} *Value*

Species_Thermal_Conductivity *SpeciesName* {=|are|is} *Value*

Turbulent Mixing Model {=|are|is} *TurbulentMixingModel [ModelArgs]...*

Vof Gas Material {=|are|is} *GasMaterial*

Vof Liquid Material {=|are|is} *LiquidMaterial*

Begin Chemistry Description *ChemistryDescName*

```
        End

        Begin Ode Solver Parameters SolverName

        End

        Begin Tabular Property Library LibNameTag

        End

    End
```

Summary Contains the commands to describe the property evaluation scheme for Fuego materials.

7.9.1.

Scope: Property Specification For Fuego Material

Properties {=|are|is} *Value*

Parameter	Value	Default
Properties	{absorption|absorption_coefficient|density|density_pressure_derivative|emissivity|enthalpy|heat_production_rate|interface_absorption|scattering_coefficient|specific_heat|surface_tension|thermal_conductivity|viscosity}	-
Value	real	-

Summary Specify a constant value for the property, in consistent units. For example, viscosity = 1.0e-3

7.9.2. Bulk Chemistry Description

Scope: Property Specification For Fuego Material

Bulk Chemistry Description {=|are|is} *Value*

Parameter	Value	Default
Value	string	-

Summary Chemistry description to apply to the bulk region of the material being defined

Description When multiple chemistry descriptions are included in a material they must apply to different portions of the material i.e. bulk and surface. A bulk description is mandatory in this case and this variable is used to define which description should be referenced. This is only necessary when there are multiple chemistry descriptions for a material.

7.9.3. Cantera Xml File

Scope: Property Specification For Fuego Material

`Cantera Xml File {=|are|is}` *`FileName`*

Parameter	Value	Default
FileName	`string`	-

Summary Name of the Cantera XML file

Description Cantera XML files can be generated from CHEMKIN input files or from *.cti files using Cantera's preprocessing utilities. See Chapter 20 for instructions on how to create a Cantera XML file.

7.9.4. Chemkin Linking File

Scope: Property Specification For Fuego Material

`Chemkin Linking File {=|are|is}` *`FileName`*

Parameter	Value	Default
FileName	`string`	-

Summary specify the file name for the CHEMKIN linking file, in ASCII format

Description NOTE: CHEMKIN is no longer supported for property evaluation. Please use Cantera instead.

7.9.5. Chemkin Transport File

Scope: Property Specification For Fuego Material

`Chemkin Transport File {=|are|is}` *`FileName`*

Parameter	Value	Default
FileName	`string`	-

Summary specify the file name for the CHEMKIN transport linking file, in ASCII format

Description NOTE: CHEMKIN is no longer supported for property evaluation. Please use Cantera instead.

7.9.6. Datum Pressure

Scope: Property Specification For Fuego Material

`Datum Pressure {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Specify the datum value for pressure

Description When running with the acoustic compressibility model, the output pressure can have an arbitrary datum value. If the datum is set equal to the reference (ambient) value, then pressure will be a "gauge" pressure. If the datum is set to zero, then an absolute pressure will result. This is ignored for incompressible flow, where the thermodynamic pressure is always equal to the reference. The datum defaults to the reference pressure for compressible flow.

7.9.7. Do Not Clip Mixture Fraction Variables

Scope: Property Specification For Fuego Material

Summary Turns off clipping for mixture fraction EOS input variables. EOS may still clip the variables

Description Fuego does not clip transported variables, relying on the equation to handle unrealizable values of its input parameters

7.9.8. Function For

Scope: Property Specification For Fuego Material

`Function For` *`Properties`* `{=|are|is}` *`FuncName`*

Parameter	Value	Default
Properties	`{absorption|absorption_coefficient|density|density_pressure_derivative|emissivity|enthalpy|heat_production_rate|interface_absorption|scattering_coefficient|specific_heat|surface_tension|thermal_conductivity|viscosity}`	-
FuncName	`string`	-

Summary Specify a function to use for evaluation of this property.

Description When specifying a function, you should also specify the name of the independent variable (abscissa). If no abscissa is specified, the temperature will be used.

7.9.9. Function For Mass_Diffusivity

Scope: Property Specification For Fuego Material

`Function For Mass_Diffusivity` *`SpeciesName`* `{=|are|is}` *`FuncName`*

Parameter	Value	Default
SpeciesName	`string`	-
FuncName	`string`	-

Summary Specify a function to use for evaluating the molecular mass diffusivity for the given species

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global function applied to all unlisted species. Any species-specific functions, if present, will override the DEFAULT function.

If this is a turbulent simulation with species transport, then the same mass diffusivity function must be used for all species. You may optionally supply just a Schmidt number.

7.9.10. Function For Species_Density

Scope: Property Specification For Fuego Material

`Function For Species_Density` *`SpeciesName`* `{=|are|is}` *`FuncName`*

Parameter	Value	Default
SpeciesName	`string`	-
FuncName	`string`	-

Summary Specify a function to use for evaluating the species enthalpy for the given species

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global function applied to all unlisted species. Any species-specific functions, if present, will override the DEFAULT function.

This command is only applicable when not using an external third-party library (e.g. Cantera) for property evaluation.

7.9.11. Function For Species_Emissivity

Scope: Property Specification For Fuego Material

`Function For Species_Emissivity` *`SpeciesName`* `{=|are|is}` *`FuncName`*

Parameter	Value	Default
SpeciesName	`string`	-
FuncName	`string`	-

Summary Specify a function to use for evaluating the species emissivity

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global function applied to all unlisted species. Any species-specific functions, if present, will override the DEFAULT function.

7.9.12. Function For Species_Enthalpy

Scope: Property Specification For Fuego Material

`Function For Species_Enthalpy` *`SpeciesName`* `{=|are|is}` *`FuncName`*

Parameter	Value	Default
SpeciesName	`string`	-
FuncName	`string`	-

Summary Specify a function to use for evaluating the species enthalpy for the given species

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global function applied to all unlisted species. Any species-specific functions, if present, will override the DEFAULT function.

This command is only applicable when not using an external third-party library (e.g. Cantera) for property evaluation.

7.9.13. Function For Species_Production_Rate

Scope: Property Specification For Fuego Material

`Function For Species_Production_Rate` *`SpeciesName`* `{=|are|is}` *`FuncName`*

Parameter	Value	Default
SpeciesName	`string`	-
FuncName	`string`	-

Summary Specify a function to use for evaluating the species volumetric mass production rate for the given species

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global function applied to all unlisted species. Any species-specific functions, if present, will override the DEFAULT function.

7.9.14. Function For Species_Specific_Heat

Scope: Property Specification For Fuego Material

`Function For Species_Specific_Heat` *`SpeciesName`* `{=|are|is}` *`FuncName`*

Parameter	Value	Default
SpeciesName	`string`	-
FuncName	`string`	-

Summary Specify a function to use for evaluating the species specific heat

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global function applied to all unlisted species. Any species-specific functions, if present, will override the DEFAULT function.

7.9.15. Function For Species_Thermal_Conductivity

Scope: Property Specification For Fuego Material

`Function For Species_Thermal_Conductivity` *`SpeciesName`* `{=|are|is}` *`FuncName`*

Parameter	Value	Default
SpeciesName	`string`	-
FuncName	`string`	-

Summary Specify a function to use for evaluating the species thermal conductivity

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global function applied to all unlisted species. Any species-specific functions, if present, will override the DEFAULT function.

7.9.16. Input Variables For

Scope: Property Specification For Fuego Material

`Input Variables For` *`SubName`* `{=|are|is}` *`Variable_list...`*

Parameter	Value	Default
SubName	`string`	-
Variable_list	`string...`	-

Summary Requests that the given variables be passed to the specified user subroutine as inputs.

7.9.17. Mass_Diffusivity

Scope: Property Specification For Fuego Material

`Mass_Diffusivity` *`SpeciesName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
SpeciesName	`string`	-
Value	`real`	-

Summary Specify the molecular mass diffusivity for the given species

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global value applied to all unlisted species. Any species-specific values, if present, will override the DEFAULT value.

If this is a turbulent simulation with species transport, then the mass diffusivity for all species must be identical. You may optionally supply just a Schmidt number.

7.9.18. Molecular_Weight

Scope: Property Specification For Fuego Material

`Molecular_Weight` *`SpeciesName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
SpeciesName	`string`	-
Value	`real`	-

Summary Specify the molecular weight for the given species

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global value applied to all unlisted species. Any species-specific values, if present, will override the DEFAULT value.

This command is only applicable when not using a material model (e.g. Cantera, Tabular, etc.).

7.9.19. Prandtl_Number

Scope: Property Specification For Fuego Material

`Prandtl_Number {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Specify the Prandtl number (Pr=mu*cp/k) to calculate thermal conductivity as a function of viscosity and specific heat.

Description If transporting both an energy equation and a species equation in turbulent flow, it is mandatory that this be specified and that it be identical to the Schmidt number, to enforce unity Lewis number (Le=Sc/Pr). In all other cases, this is optional.

7.9.20. Real Data For

Scope: Property Specification For Fuego Material

`Real Data For` *`SubName`* `{=|are|is}` *`Parameter_list...`*

Parameter	Value	Default
SubName	`string`	-
Parameter_list	`real...`	-

Summary Specifies a list of parameters that should be passed to the specified user subroutine.

7.9.21. Reference

Scope: Property Specification For Fuego Material

`Reference` *`Variable`* `{=|are|is}` *`Value`*

Parameter	Value	Default
Variable	`string`	-
Value	`real`	-

Summary Specify the reference value for the given variable

Description A reference value must be specified for all input variables required by the particular material model for property evaluation. In the typical case where Cantera is used for property evaluation, then reference properties must be specified for "temperature", "pressure", and "mass_fraction". Other material models may have different dependencies.

These values will be used for reporting property values at the reference state during startup, and for property evaluation at run-time when a transport equation is not being solved for the required variable or when a reference state is needed for calculating "ambient" properties for entrainment purposes.

7.9.22. Reference Mass_Fraction

Scope: Property Specification For Fuego Material

`Reference Mass_Fraction` *`SpeciesName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
SpeciesName	`string`	-
Value	`real`	-

Summary Specify the reference mass fraction for the given species

Description A reference value must be specified for all input variables required by the particular material model for property evaluation. In the typical case where Cantera is used for property evaluation, then reference properties must be specified for "temperature", "pressure", and "mass_fraction". Other material models may have different dependencies.

These values will be used for reporting property values at the reference state during startup, and for property evaluation at run-time when a transport equation is not being solved for the required variable or when a reference state is needed for calculating "ambient" properties for entrainment purposes.

7.9.23. Reference Mole_Fraction

Scope: Property Specification For Fuego Material

`Reference Mole_Fraction` *`SpeciesName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
SpeciesName	`string`	-
Value	`real`	-

Summary Specify the reference mole fraction for the given species

Description A reference value must be specified for all input variables required by the particular material model for property evaluation. In the typical case where Cantera is used for property evaluation, then reference properties must be specified for "temperature", "pressure", and "mass_fraction". Other material models may have different dependencies.

These values will be used for reporting property values at the reference state during startup, and for property evaluation at run-time when a transport equation is not being solved for the required variable or when a reference state is needed for calculating "ambient" properties for entrainment purposes.

7.9.24. Reference Progress_Variable

Scope: Property Specification For Fuego Material

`Reference Progress_Variable` *`ProgressVariableName`* `With Lower Bound` *`LowerBound`* `And Upper Bound` *`UpperBound`*

Parameter	Value	Default
ProgressVariableName	`string`	-
LowerBound	`real`	-
UpperBound	`real`	-

Summary Specify the upper and lower bound of a progress variable, in that order

Description A lower and upper bound must be specified for all progress variables.

7.9.25. Set Schmidt Number For Progress Variable

Scope: Property Specification For Fuego Material

`Set Schmidt Number For Progress Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`real`	-

Summary Specify Schmidt numbers for Progress Variable

Description Specify Schmidt numbers for Progress Variable

7.9.26. Subroutine For

Scope: Property Specification For Fuego Material

`Subroutine For` *`PropertyName`* `{=|are|is}` *`SubName`*

Parameter	Value	Default
PropertyName	`string`	-
SubName	`string`	-

Summary Set the value of the specified variable using a subroutine. If the subroutine should be passed other properties as variables or be given real data, the INPUT VARIABLES FOR and REAL DATA FOR line commands below should be used.

7.9.27. Schmidt_Number

Scope: Property Specification For Fuego Material

`Schmidt_Number {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Specify the Schmidt number (Sc=mu/rho*D) to calculate mass diffusivity as a function of viscosity and density.

Description If transporting both an energy equation and a species equation in turbulent flow, it is mandatory that this be specified and that it be identical to the Prandtl number, to enforce unity Lewis number (Le=Sc/Pr). If transporting turbulent species without an energy equation, then either the Schmidt number or an identical mass diffusivity for all species must be specified. In all other cases, this is optional.

7.9.28. Species_Density

Scope: Property Specification For Fuego Material

`Species_Density` *`SpeciesName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
SpeciesName	`string`	-
Value	`real`	-

Summary Specify the species density for the given species

Description For the SpeciesName parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global value applied to all unlisted species. Any species-specific values, if present, will override the DEFAULT value.

This command is only applicable when not using an external third-party library (e.g. Cantera) for property evaluation.

7.9.29. Species_Emissivity

Scope: Property Specification For Fuego Material

`Species_Emissivity` *`SpeciesName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
SpeciesName	`string`	-
Value	`real`	-

Summary Specify the emissivity for the given species

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global value applied to all unlisted species. Any species-specific values, if present, will override the DEFAULT value.

7.9.30. Species_Enthalpy

Scope: Property Specification For Fuego Material

`Species_Enthalpy` *`SpeciesName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
SpeciesName	`string`	-
Value	`real`	-

Summary Specify the species enthalpy for the given species

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global value applied to all unlisted species. Any species-specific values, if present, will override the DEFAULT value.

This command is only applicable when not using an external third-party library (e.g. Cantera) for property evaluation.

7.9.31. Species_Production_Rate

Scope: Property Specification For Fuego Material

`Species_Production_Rate` *`SpeciesName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
SpeciesName	`string`	-
Value	`real`	-

Summary Specify the species volumetric mass production rate for the given species

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global value applied to all unlisted species. Any species-specific values, if present, will override the DEFAULT value.

7.9.32. Species_Specific_Heat

Scope: Property Specification For Fuego Material

`Species_Specific_Heat` *`SpeciesName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
SpeciesName	`string`	-
Value	`real`	-

Summary Specify the specific heat for the given species

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global value applied to all unlisted species. Any species-specific values, if present, will override the DEFAULT value.

7.9.33. Species_Thermal_Conductivity

Scope: Property Specification For Fuego Material

`Species_Thermal_Conductivity` *`SpeciesName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
SpeciesName	`string`	-
Value	`real`	-

Summary Specify the thermal conductivity for the given species

Description For the *SpeciesName* parameter, you can either specify the actual species name (O_2, N_2, CH_4, etc.) or the word DEFAULT for a global value applied to all unlisted species. Any species-specific values, if present, will override the DEFAULT value.

7.9.34. Turbulent Mixing Model

Scope: Property Specification For Fuego Material

`Turbulent Mixing Model {=|are|is}` *`TurbulentMixingModel [ModelArgs]...`*

Parameter	Value	Default
TurbulentMixingModel	`{none|parente}`	-
ModelArgs	`[string]...`	-

Summary Specify turbulent mixing model to limit reaction chemistry

Description Use turbulent mixing time scale to determine rate of reaction in chemistry model using EDC 2016 Model (Parente et al. Fuel 2016)

7.9.35. Vof Gas Material

Scope: Property Specification For Fuego Material

`Vof Gas Material {=|are|is}` *`GasMaterial`*

Parameter	Value	Default
GasMaterial	`string`	-

Summary Name of the material to use for the gas phase in a VOF simulation

Description Name of the material to use for the gas phase in a VOF simulation.

7.9.36. Vof Liquid Material

Scope: Property Specification For Fuego Material

`Vof Liquid Material {=|are|is}` *`LiquidMaterial`*

Parameter	Value	Default
LiquidMaterial	`string`	-

Summary Name of the material to use for the liquid phase in a VOF simulation

Description Name of the material to use for the liquid phase in a VOF simulation.

7.10. TABULAR PROPERTY LIBRARY

Scope: Property Specification For Fuego Material

```
Begin Tabular Property Library LibNameTag
    Auxiliary Variable AuxVarName Expression {=|are|is} Expr
    Ignition Blending Time {=|are|is} Value
    Ignition Time {=|are|is} Value
    Library Hdf5 File {=|are|is} FileName
    Number Of Logged Input Clipping Events {=|are|is} Number
    Output Library Variable LibraryVariable As OutputName [ On Output
    Block BlockName ]
    Source Term For ProgressVariableName Expression {=|are|is} Expr
    Table Interpolation Method {=|are|is} Value
    Table Units Are In UnitSystem
    Use Approximate Flamelet Enthalpy Reconstruction
    Use Consistently Volume Averaged Properties
    Use Field FuegoField For Library Input LibraryInput
    Use Library Source LibrarySource As SourceName
    Use Library Variable LibraryVariable For Property FuegoProperties
End
```

Summary Begin specification for tabular properties

7.10.1. Auxiliary Variable

Scope: Tabular Property Library

`Auxiliary Variable` *`AuxVarName`* `Expression {=|are|is}` *`Expr`*

Parameter	Value	Default
AuxVarName	`string`	-
Expr	`(expression)`	-

Summary Specifies the expression to evaluate for an auxiliary variable.

Description Specifies the expression to evaluate for an auxiliary variable.

7.10.2. Ignition Blending Time

Scope: Tabular Property Library

`Ignition Blending Time {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	1

Summary Length of time to blend flamelet model from non-ignited to igniting states

Description If an ignition time is specified for the flamelet model, it begins at the specified time and is gradually introduced over a finite blending time (this parameter). By "ignition time" + "blending time" all blending effects have been removed.

7.10.3. Ignition Time

Scope: Tabular Property Library

`Ignition Time {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	0

Summary Time to begin flamelet ignition

Description By default ignition is immediate with the flamelet model. This option lets you delay the ignition until a specified time. Prior to that time, the mixture fraction values used in property evaluation are clamped to 0 or 1.

7.10.4. Library Hdf5 File

Scope: Tabular Property Library

`Library Hdf5 File {=|are|is}` *`FileName`*

Parameter	Value	Default
FileName	`string`	-

Summary Specify the name (with optional path) for the HDF5 property library generated by fuego_tabular_props

7.10.5. Number Of Logged Input Clipping Events

Scope: Tabular Property Library

`Number Of Logged Input Clipping Events {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	-

Summary Number of input clipping events that will be logged and reported

Description Input coordinates for tabular property lookups are always clipped to the valid range specified during table creation. Clipping events are internally ranked by "severity", and only the most severe are reported. Default output is the top 10 events. Set to zero to disable reporting.

7.10.6. Output Library Variable

Scope: Tabular Property Library

`Output Library Variable` *`LibraryVariable`* `As` *`OutputName`* `[ On Output Block` *`BlockName`* `]`

Parameter	Value	Default
LibraryVariable	`string`	-
OutputName	`string`	-

Summary Generates output for the requested library variable.

Description Provide output of the requested property library variable. If the optional output block name is specified, then the variable will only be written to that output block.

7.10.7. Source Term For

Scope: Tabular Property Library

`Source Term For` *`ProgressVariableName`* `Expression {=|are|is}` *`Expr`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Expr	`(expression)`	-

Summary Specifies the expression to evaluate.

Description Specifies the expression to evaluate.

7.10.8. Table Interpolation Method

Scope: Tabular Property Library

`Table Interpolation Method {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`string`	`1`

Summary Which interpolation backend to use (BSPLINE or LAGRANGE)

Description One may either choose a general B-spline formulation or a Lagrange polynomial as the interpolant for table lookups. The B-spline formulation has better guarantees on continuity and smoothness but is is between two to ten times more expensive to evaluate than the Lagrange basis. Lagrange polynomials are likely sufficient for tables with enough resolution for accurate reconstruction of tabulated properties.

7.10.9. Table Units Are In

Scope: Tabular Property Library

`Table Units Are In` *`UnitSystem`*

Parameter	Value	Default
UnitSystem	`{cgs|mks}`	-

Summary Describe the length-mass-time unit system used by the table (CGS or MKS)

Description Shortcut to select meter-kilogram-second (MKS) or centimeter-gram-second unit (CGS) system. This should correspond to the units your table was generated in.

7.10.10. Use Approximate Flamelet Enthalpy Reconstruction

Scope: Tabular Property Library

Summary Use an inexact method of reconstructing the enthalpy defect of nonadiabatic flamelet models.

Description Uses an inexact method of obtaining the stoichiometric enthalpy defect for nonadiabatic strained laminar flamelet models. In cases with strong heat loss this leads to a large, persistent error and should not be used.

7.10.11. Use Consistently Volume Averaged Properties

Scope: Tabular Property Library

Summary Use a wider stencil to evaluate nodal properties, like density

Description Interpolates input fields to subcontrol volume centers to evaluate a volume average, instead of just using the nodal value

7.10.12. Use Field

Scope: Tabular Property Library

`Use Field` *`FuegoField`* `For Library Input` *`LibraryInput`*

Parameter	Value	Default
FuegoField	`string`	-
LibraryInput	`string`	-

Summary Link a Fuego field to the specified property library input variable

7.10.13. Use Library Source

Scope: Tabular Property Library

`Use Library Source` *`LibrarySource`* `As` *`SourceName`*

Parameter	Value	Default
LibrarySource	`string`	-
SourceName	`string`	-

Summary Link property library variable to a tabulated source term

Description Provide a connection between a tabular property library variable and a source term. The library will be used to evaluate the source term.

7.10.14. Use Library Variable

Scope: Tabular Property Library

`Use Library Variable` *`LibraryVariable`* `For Property` *`FuegoProperties`*

Parameter	Value	Default
LibraryVariable	`string`	-
FuegoProperties	`{absorption_coef | conserved_enthalpy | density | density_pressure_derivative | emissivity | enthalpy | heat_production_rate | mass_diffusivity | molecular_weight | scattering_coef | species_density | species_emissivity | species_enthalpy | species_production_rate | species_specific_heat | species_thermal_conductivity | specific_heat | temperature | thermal_conductivity | viscosity}`	-

Summary Link property library variable to a Fuego property

Description Provide a connection between a tabular property library variable and a Fuego property. The library will be used to evaluate the property.

For adiabatic flamelet simulations, the following properties are required: Density and Viscosity.

For non-adiabatic flamelet simulations, the following properties are required: Density, Viscosity, Temperature, Specific_Heat, Enthalpy, and Conserved_Enthalpy. You must also activate both the Enthalpy and Conserved_Enthalpy equations.

7.11. PROPERTY SPECIFICATION FOR MATERIAL

Scope: Sierra

```
Begin Property Specification For Material MaterialName

    Density {=|are|is} Density

    Density Function {=|are|is} FunctionName

End
```

Summary A set of material properties for a named material.

7.11.1. Density

Scope: Property Specification For Material

`Density {=|are|is}` *`Density`*

Parameter	Value	Default
Density	`real`	-

Summary Material or mixture density.

7.11.2. Density Function

Scope: Property Specification For Material

`Density Function {=|are|is}` *`FunctionName`*

Parameter	Value	Default
FunctionName	`string`	-

Summary Material or mixture density function.

7.12. TRILINOS EQUATION SOLVER

Scope: Sierra

```
Begin Trilinos Equation Solver SolverName
    Bc Enforcement {=|are|is} BcEnforcement
    Debug Output Level {=|are|is} Level
    Debug Output Path {=|are|is} DebugOutput
    Determine Sharing {=|are|is} FeiDetermineSharing
    Fei Output Level {=|are|is} FeiOutputLevels
    Ilu Fill {=|are|is} ilu_fill_level
    Ilu Threshold {=|are|is} Threashold
    Matrix Format {=|are|is} MatrixFormat
    Matrix Reduction {=|are|is} AztecReductionType
    Matrix Scaling {=|are|is} MatrixScaling
    Maximum Iterations {=|are|is} max_iterations
    Num Levels {=|are|is} Num_levels
    Param-Bool Parameter_name Value Bool
```

```
    Param-Int Parameter_name Value integer_value
    Param-Real Parameter_name Value Real_value
    Param-String Parameter_name Value String_value
    Polynomial Order {=|are|is} polynomial_order
    Preconditioner Subdomain Overlap {=|are|is} Overlap
    Preconditioning Method {=|are|is} TrilinosPrecondMethods
    Preconditioning Steps {=|are|is} preconditioning_steps
    Residual Norm Scaling {=|are|is} AztecResidualNormScaling
    Residual Norm Tolerance {=|are|is} residual_norm_tolerance
    Restart Iterations {=|are|is} restart_iterations
    Row Ordering {=|are|is} RowOrdering
    Shared Ownership Rule {=|are|is} SharedOwnershipRule
    Solution Method {=|are|is} TrilinosSolverMethods
    Solve Transpose {=|are|is} Option
    Begin Teuchos Parameter Block Teuchos Parameter Block Name
    End
End
```

Summary A set of solver parameters for Trilinos equation solver.

7.12.1. Bc Enforcement

Scope: Trilinos Equation Solver

```
Bc Enforcement {=|are|is} BcEnforcement
```

Parameter	Value	Default
BcEnforcement	{exact|exact_no_column_mod|remove|solver|solver_no_column_mod}	-

Summary Controls the way Dirichlet BCs are enforced.

Description Valid values for this line-command are contained in the BcEnforcement enum.

7.12.2. Debug Output Level

Scope: Trilinos Equation Solver

`Debug Output Level {=|are|is}` *`Level`*

Parameter	Value	Default
Level	`integer`	-

Summary Output level for debugging. Generally o means no solver screen output, and higher values of this parameter correspond to more screen output.

7.12.3. Debug Output Path

Scope: Trilinos Equation Solver

`Debug Output Path {=|are|is}` *`DebugOutput`*

Parameter	Value	Default
DebugOutput	`string`	-

Summary Specify path where debug-logs and other debug output files will be placed.

7.12.4. Determine Sharing

Scope: Trilinos Equation Solver

`Determine Sharing {=|are|is}` *`FeiDetermineSharing`*

Parameter	Value	Default
FeiDetermineSharing	`{fei|sierra}`	-

Summary Whether FEI should determine sharing internally or receive the info from Sierra.

Description By default the Eqns::LinearSystem class tells the fei layer which nodes are shared and which processors share them. If this option is set to on, then the fei layer internally determines the sharing info. Requires the fei layer to perform extra communication during the intialize phase.

7.12.5. Fei Output Level

Scope: Trilinos Equation Solver

`Fei Output Level {=|are|is}` *`FeiOutputLevels`*

Parameter	Value	Default
FeiOutputLevels	`{all|brief_logs|full_logs|matrix_files|none|stats}`	-

Summary Control the amount of output produced by FEI.

Description Output level controls the amount of debugging information (screen output, matrix/vector files and log files) produced by FEI. The FeiOutputLevels enum contains the valid values for this line-command. The location of files can be controlled by the 'debug output path' line-command.

7.12.6. Ilu Fill

Scope: Trilinos Equation Solver

`Ilu Fill {=|are|is}` *`ilu_fill_level`*

Parameter	Value	Default
ilu_fill_level	`integer`	-

Summary Fill-in parameter for incomplete factorizations.

7.12.7. Ilu Threshold

Scope: Trilinos Equation Solver

`Ilu Threshold {=|are|is}` *`Threashold`*

Parameter	Value	Default
Threashold	`real`	-

Summary Threshold parameter for incomplete factorizations.

7.12.8. Matrix Format

Scope: Trilinos Equation Solver

`Matrix Format {=|are|is}` *`MatrixFormat`*

Parameter	Value	Default
MatrixFormat	`{csr|msr|vbr}`	-

Summary Storage format for the matrix.

Description This parameter only applies to Trilinos and Aztec.

7.12.9. Matrix Reduction

Scope: Trilinos Equation Solver

`Matrix Reduction {=|are|is}` *`AztecReductionType`*

Parameter	Value	Default
AztecReductionType	`{fei-remove-slaves}`	-

Summary Remove constraint equations from matrix.

Description Dependent degrees of freedom in constraints are projected out of the equation space, yielding a linear system with smaller dimension, and retaining positive definiteness (whereas the alternative, an augmented matrix arising from the lagrange multiplier formulation, is indefinite).

7.12.10. Matrix Scaling

Scope: Trilinos Equation Solver

`Matrix Scaling {=|are|is}` *`MatrixScaling`*

Parameter	Value	Default
MatrixScaling	`{block-jacobi|jacobi|none|row-sum|symmetric-diagonal|symmetric-row-sum|user supplied}`	-

Summary Scaling to be applied to the matrix.

Description Matrix scaling is a pre-solve operation, typically modifying the matrix in place, whereas preconditioning is performed at each iteration during the solve and doesn't modify the actual matrix.

7.12.11. Maximum Iterations

Scope: Trilinos Equation Solver

`Maximum Iterations {=|are|is}` *`max_iterations`*

Parameter	Value	Default
max_iterations	`integer`	-

Summary Maximum number of solution method iterations.

7.12.12. Num Levels

Scope: Trilinos Equation Solver

`Num Levels {=|are|is}` *`Num_levels`*

Parameter	Value	Default
Num_levels	`integer`	-

Summary Number of levels for multi-level/multi-grid solvers.

7.12.13. Param-Bool

Scope: Trilinos Equation Solver

`Param-Bool` *`Parameter_name`* `Value` *`Bool`*

Parameter	Value	Default
Parameter_name	`"string"`	-
Bool	`{false|true}`	-

Summary String-Key/Boolean-Value pair to be passed to solver.

Description Syntax: 'PARAM-BOOL "some-name" VALUE true | false'

7.12.14. Param-Int

Scope: Trilinos Equation Solver

`Param-Int` *`Parameter_name`* `Value` *`integer_value`*

Parameter	Value	Default
Parameter_name	`"string"`	-
integer_value	`integer`	-

Summary String-Key/Integer-Value pair to be passed to solver.

Description Syntax: 'PARAM-INT "some-name" VALUE 123' Primarily used for passing values to the ML package. ML has a large number of parameters which haven't been individually incorporated into our parameter-parsing system. Can also be used to pass parameters to Aztec, e.g.:

PARAM-REAL "AZ_ilut_fill" VALUE 3

7.12.15. Param-Real

Scope: Trilinos Equation Solver

`Param-Real` *`Parameter_name`* `Value` *`Real_value`*

Parameter	Value	Default
Parameter_name	`"string"`	-
Real_value	`real`	-

Summary String-Key/Real-Value pair to be passed to solver.

Description Syntax: 'PARAM-REAL "some-name" VALUE 1.23' Primarily used for passing values to the ML package. ML has a large number of parameters which haven't been individually incorporated into our parameter-parsing system. Can also be used to pass parameters to Aztec, e.g.:

PARAM-REAL "AZ_drop" VALUE 1.e-8

7.12.16. Param-String

Scope: Trilinos Equation Solver

`Param-String` *`Parameter_name`* `Value` *`String_value`*

Parameter	Value	Default
Parameter_name	`"string"`	-
String_value	`"string"`	-

Summary Key/Value string-pair to be passed to solver.

Description Syntax: 'PARAM-STRING "some-name" VALUE "some-value"' (Primarily used for passing values to the ML package. ML has a large number of parameters which haven't been individually incorporated into our parameter-parsing system.)

7.12.17. Polynomial Order

Scope: Trilinos Equation Solver

`Polynomial Order {=|are|is}` *`polynomial_order`*

Parameter	Value	Default
polynomial_order	`integer`	-

Summary Polynomial order of preconditioning method.

7.12.18. Preconditioner Subdomain Overlap

Scope: Trilinos Equation Solver

`Preconditioner Subdomain Overlap {=|are|is}` *`Overlap`*

Parameter	Value	Default
Overlap	`integer`	-

Summary Ovrlap of Schwarz subdomains (eg, 0,1 or 2).

7.12.19. Preconditioning Method

Scope: Trilinos Equation Solver

`Preconditioning Method {=|are|is}` *`TrilinosPrecondMethods`*

Parameter	Value	Default
TrilinosPrecondMethods	`{dd-bilu|dd-icc|dd-ilu|dd-ilut|dd-lu|dd-rilu|jacobi|least-squares|multilevel|neumann|none}`	-

Summary Selection of the preconditioning method.

Description Valid values for this are contained in the TrilinosPrecondMethods enum.

7.12.20. Preconditioning Steps

Scope: Trilinos Equation Solver

`Preconditioning Steps {=|are|is}` *`preconditioning_steps`*

Parameter	Value	Default
preconditioning_steps	`integer`	-

Summary Number of Jacobi, Gauss-Seidel, or other preconditioning methods' applications per iteration.

7.12.21. Residual Norm Scaling

Scope: Trilinos Equation Solver

`Residual Norm Scaling {=|are|is}` *`AztecResidualNormScaling`*

Parameter	Value	Default
AztecResidualNormScaling	`{anorm|none|r0|rhs}`	-

Summary Scaling method for the residual norm.

Description Determines the residual expression used in convergence checks and printing.

7.12.22. Residual Norm Tolerance

Scope: Trilinos Equation Solver

`Residual Norm Tolerance {=|are|is}` *`residual_norm_tolerance`*

Parameter	Value	Default
residual_norm_tolerance	`real`	-

Summary Iterative solution method relative residual convergence tolerance.

7.12.23. Restart Iterations

Scope: Trilinos Equation Solver

`Restart Iterations {=|are|is}` *`restart_iterations`*

Parameter	Value	Default
restart_iterations	`integer`	-

Summary Number of iterations between GMRES restarts.

7.12.24. Row Ordering

Scope: Trilinos Equation Solver

`Row Ordering {=|are|is}` *`RowOrdering`*

Parameter	Value	Default
RowOrdering	`{rows_with_local_cols_first}`	-

Summary Specify that rows with only local columns should come first in the ordering.

7.12.25. Shared Ownership Rule

Scope: Trilinos Equation Solver

`Shared Ownership Rule {=|are|is}` *`SharedOwnershipRule`*

Parameter	Value	Default

SharedOwnershipRule	`{low-numbered-proc|`
	`proc-with-local-elem|`
	`sierra_specifies}`
		-

Summary Controls the way owning processors are chosen for shared nodes in the FEI.

Description 'low-numbered-proc' is the default.

'proc-with-local-elem' is another valid value.

'sierra_specifies' is the other valid value

7.12.26. Solution Method

Scope: Trilinos Equation Solver

`Solution Method {=|are|is}` *`TrilinosSolverMethods`*

Parameter	Value	Default
TrilinosSolverMethods	`{amesos-klu|amesos-superlu|amesos-umfpack|bicgstab|cg|cgs|gmres|lu|tfqmr}`	-

Summary Selection of the linear-system solution algorithm.

Description The TrilinosSolverMethods enum contains valid values for this line-command. There are several iterative methods (Krylov subspace algorithms), as well as some sparse direct methods available through the Amesos interface package. At this time (Oct'05) not all of the listed direct solvers are available. See the specific entries in the TrilinosSolverMethods enum below for more information.

7.12.27. Solve Transpose

Scope: Trilinos Equation Solver

`Solve Transpose {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`string`	-

Summary Whether to solve for transpose of system matrix.

7.13. TEUCHOS PARAMETER BLOCK

Scope: Trilinos Equation Solver

```
Begin Teuchos Parameter Block Teuchos Parameter Block Name
    Param-Bool Parameter_name Value Bool
    Param-Int Parameter_name Value integer_value
    Param-Real Parameter_name Value Real_value
    Param-String Parameter_name Value String_value
    Param-Sublist Parameter_name Value Block_name
End
```

Summary A block to set a Teuchos parameter list.

Description This allows a general way of passing a string/argument to various packages supported by NOX, eg Ifpack and ML preconditioning packages. See PRECONDITIONING PACKAGE.

7.13.1. Param-Bool

Scope: Teuchos Parameter Block

`Param-Bool` *`Parameter_name`* `Value` *`Bool`*

Parameter	Value	Default
Parameter_name	`"string"`	-
Bool	`{false|true}`	-

Summary String-Key/Boolean-Value pair to be passed to solver.

Description Syntax: 'PARAM-BOOL "some-name" VALUE true | false'

7.13.2. Param-Int

Scope: Teuchos Parameter Block

`Param-Int` *`Parameter_name`* `Value` *`integer_value`*

Parameter	Value	Default
Parameter_name	`"string"`	-
integer_value	`integer`	-

Summary String-Key/Integer-Value pair to be passed to solver.

Description Syntax: 'PARAM-INT "some-name" VALUE 123' Primarily used for passing values to the ML package. ML has a large number of parameters which haven't been individually incorporated into our parameter-parsing system. Can also be used to pass parameters to Aztec, e.g.:

PARAM-REAL "AZ_ilut_fill" VALUE 3

7.13.3. Param-Real

Scope: Teuchos Parameter Block

`Param-Real` *`Parameter_name`* `Value` *`Real_value`*

Parameter	Value	Default
Parameter_name	`"string"`	-
Real_value	`real`	-

Summary String-Key/Real-Value pair to be passed to solver.

Description Syntax: 'PARAM-REAL "some-name" VALUE 1.23' Primarily used for passing values to the ML package. ML has a large number of parameters which haven't been individually incorporated into our parameter-parsing system. Can also be used to pass parameters to Aztec, e.g.:

PARAM-REAL "AZ_drop" VALUE 1.e-8

7.13.4. Param-String

Scope: Teuchos Parameter Block

`Param-String` *`Parameter_name`* `Value` *`String_value`*

Parameter	Value	Default
Parameter_name	`"string"`	-
String_value	`"string"`	-

Summary Key/Value string-pair to be passed to solver.

Description Syntax: 'PARAM-STRING "some-name" VALUE "some-value"' (Primarily used for passing values to the ML package. ML has a large number of parameters which haven't been individually incorporated into our parameter-parsing system.)

7.13.5. Param-Sublist

Scope: Teuchos Parameter Block

`Param-Sublist` *`Parameter_name`* `Value` *`Block_name`*

Parameter	Value	Default
Parameter_name	`"string"`	-
Block_name	`string`	-

Summary String-Key/String-Value pair to designate another Teuchos block as a sublist to this block.

Description Syntax: 'PARAM-SUBLIST "some-name" VALUE teuchos_block_name

7.14. TPETRA EQUATION SOLVER

Scope: Sierra

`Begin Tpetra Equation Solver` *`Solver Name`*

`Bc Enforcement {=|are|is}` *`value`*

`Matrix Output {=|are|is}` *`value`*

`Matrix Output Time Range {=|are|is}` $value_1$ `[` $value_2$ `]`

```
    Matrix Scaling {=|are|is} value

    Begin Bicgstab Solver

    End

    Begin Cg Solver

    End

    Begin Expert Belos Solver

    End

    Begin Gmres Solver

    End

    Begin Klu2 Solver

    End

    Begin Preset Solver

    End

    Begin Superlu Solver

    End

    Begin Umfpack Solver

    End

End
```

Summary Use a Tpetra-based linear solver.

7.14.1. Bc Enforcement

Scope: Tpetra Equation Solver

`Bc Enforcement {=|are|is} value`

Parameter	Value	Default
value	`{exact|exact_no_column_mod|remove|solver|solver_no_column_mod}`	-

Summary Control how Dirichlet BC's are enforced. Fuego only.

7.14.2. Matrix Output

Scope: Tpetra Equation Solver

`Matrix Output {=|are|is}` *value*

Parameter	Value	Default
value	`string`	-

Summary Output the matrix to the specified file name.

7.14.3. Matrix Output Time Range

Scope: Tpetra Equation Solver

`Matrix Output Time Range {=|are|is}` $value_1$ *[* $value_2$ *]*

Parameter	Value	Default
value	`real_1[ real_2]`	-

Summary Range of simulation time to output the matrix to file, default is dump out all matrices per step.

7.14.4. Matrix Scaling

Scope: Tpetra Equation Solver

`Matrix Scaling {=|are|is}` *value*

Parameter	Value	Default
value	`{diagonal|one_norm}`	-

Summary Apply a pre-solve scaling to the matrix.

Description In some problems scaling the matrix prior to the linear solve can help reduce the condition number of the matrix. This should be used in addition to an appropriate preconditioner.

7.15. KLU2 SOLVER

Scope: Tpetra Equation Solver

```
Begin Klu2 Solver
End
```

Summary Use a KLU2 direct solver.

7.16. BICGSTAB SOLVER

Scope: Tpetra Equation Solver

```
Begin Bicgstab Solver
    Convergence Tolerance {=|are|is} value
    Maximum Iterations {=|are|is} value
    Residual Scaling {=|are|is} value
    Begin Dd-Ilu Preconditioner
    End
    Begin Dd-Ilut Preconditioner
    End
    Begin Expert Ifpack2 Preconditioner
    End
    Begin Jacobi Preconditioner
    End
    Begin Muelu Preconditioner
    End
    Begin Sgs Preconditioner
    End
End
```

Summary Use a bicgstab solver.

7.16.1. Convergence Tolerance

Scope: Bicgstab Solver

`Convergence Tolerance {=|are|is} value`

Parameter	Value	Default
value	`real`	`1e-06`

Summary Set the convergence tolerance for the linear solver

7.16.2. Maximum Iterations

Scope: Bicgstab Solver

`Maximum Iterations {=|are|is}` *value*

Parameter	Value	Default
value	integer	300

Summary Set the maximum number of iterations taken by the linear solver

7.16.3. Residual Scaling

Scope: Bicgstab Solver

`Residual Scaling {=|are|is}` *value*

Parameter	Value	Default
value	{none|preconditioned_r0|r0|rhs}	r0

Summary Set the residual scaling for the linear solver

7.17. GMRES SOLVER

Scope: Tpetra Equation Solver

```
Begin Gmres Solver
    Convergence Tolerance {=|are|is} value
    Maximum Iterations {=|are|is} value
    Residual Scaling {=|are|is} value
    Restart Iterations {=|are|is} value
    Begin Dd-Ilu Preconditioner
    End

    Begin Dd-Ilut Preconditioner
    End

    Begin Expert Ifpack2 Preconditioner
    End

    Begin Jacobi Preconditioner
    End

    Begin Muelu Preconditioner
```

```
    End

    Begin Sgs Preconditioner

    End

End
```

Summary Use a GMRES solver.

7.17.1. Convergence Tolerance

Scope: Gmres Solver

```
Convergence Tolerance {=|are|is} value
```

Parameter	Value	Default
value	`real`	`1e-06`

Summary Set the convergence tolerance for the linear solver

7.17.2. Maximum Iterations

Scope: Gmres Solver

```
Maximum Iterations {=|are|is} value
```

Parameter	Value	Default
value	`integer`	`300`

Summary Set the maximum number of iterations taken by the linear solver

7.17.3. Residual Scaling

Scope: Gmres Solver

```
Residual Scaling {=|are|is} value
```

Parameter	Value	Default
value	`{none|preconditioned_r0|r0|rhs}`	`r0`

Summary Set the residual scaling for the linear solver

7.17.4. Restart Iterations

Scope: Gmres Solver

`Restart Iterations {=|are|is}` *`value`*

Parameter	Value	Default
value	`integer`	30

Summary Set the number of iterations between GMRES restarts.

7.18. UMFPACK SOLVER

Scope: Tpetra Equation Solver

```
Begin Umfpack Solver
End
```

Summary Use a UMFPACK direct solver.

7.19. EXPERT BELOS SOLVER

Scope: Tpetra Equation Solver

```
Begin Expert Belos Solver
    Xml File {=|are|is} value
    Begin Dd-Ilu Preconditioner
    End

    Begin Dd-Ilut Preconditioner
    End

    Begin Expert Ifpack2 Preconditioner
    End

    Begin Jacobi Preconditioner
    End

    Begin Muelu Preconditioner
    End

    Begin Sgs Preconditioner
```

```
    End
End
```

Summary Use a Belos solver with parameters specified in an XML file.

7.19.1. Xml File

Scope: Expert Belos Solver

`Xml File {=|are|is}` *value*

Parameter	Value	Default
value	`string`	-

Summary Supply the name of an XML file containing a Belos parameter list.

Description The supplied XML file must contain a ParameterList with a "solution method" string parameter corresponding to the desired type of Belos solver. Any additional parameters that should be passed on to Belos should be part of a nested ParameterList with name="Belos-Tpetra".

7.20. SUPERLU SOLVER

Scope: Tpetra Equation Solver

```
Begin Superlu Solver
End
```

Summary Use a SUPERLU direct solver.

7.21. CG SOLVER

Scope: Tpetra Equation Solver

```
Begin Cg Solver
    Convergence Tolerance {=|are|is} value
    Maximum Iterations {=|are|is} value
    Residual Scaling {=|are|is} value
    Begin Dd-Ilu Preconditioner
    End
```

```
    Begin Dd-Ilut Preconditioner

    End

    Begin Expert Ifpack2 Preconditioner

    End

    Begin Jacobi Preconditioner

    End

    Begin Muelu Preconditioner

    End

    Begin Sgs Preconditioner

    End

End
```

Summary Use a conjugate gradient solver.

7.21.1. Convergence Tolerance

Scope: Cg Solver

`Convergence Tolerance {=|are|is}` *value*

Parameter	Value	Default
value	`real`	`1e-06`

Summary Set the convergence tolerance for the linear solver

7.21.2. Maximum Iterations

Scope: Cg Solver

`Maximum Iterations {=|are|is}` *value*

Parameter	Value	Default
value	`integer`	`300`

Summary Set the maximum number of iterations taken by the linear solver

7.21.3. Residual Scaling

Scope: Cg Solver

```
Residual Scaling {=|are|is} value
```

Parameter	Value	Default
value	`{none|preconditioned_r0|r0|rhs}`	`r0`

Summary Set the residual scaling for the linear solver

7.22. PRESET SOLVER

Scope: Tpetra Equation Solver

```
Begin Preset Solver

    Convergence Tolerance {=|are|is} value

    Maximum Iterations {=|are|is} value

    Residual Scaling {=|are|is} value

    Solver Type {=|are|is} value

End
```

Summary Use a preset solver for a specific linear system type.

Description The preset solver for a given linear system type is intended to provide reasonable behavior for many problems of that type. If this proves to be insufficient, the user must provide more fine-grained control over solver settings. The equivalent command which produces the solver will be output to the log file, and this may be used as a starting point from which changes for robustness may be made (i.e. increasing GMRES subspace size or changing the preconditioner).

7.22.1. Convergence Tolerance

Scope: Preset Solver

```
Convergence Tolerance {=|are|is} value
```

Parameter	Value	Default
value	`real`	`1e-06`

Summary Set the convergence tolerance for the linear solver

7.22.2. Maximum Iterations

Scope: Preset Solver

```
Maximum Iterations {=|are|is} value
```

Parameter	Value	Default
value	integer	300

Summary Set the maximum number of iterations taken by the linear solver

7.22.3. Residual Scaling

Scope: Preset Solver

```
Residual Scaling {=|are|is} value
```

Parameter	Value	Default
value	{none|preconditioned_r0|r0|rhs}	r0

Summary Set the residual scaling for the linear solver

7.22.4. Solver Type

Scope: Preset Solver

```
Solver Type {=|are|is} value
```

Parameter	Value	Default
value	{continuity|multiphysics| scalar_transport|thermal| thermal_bicgstab|thermal_symmetric}	-

Summary Set the preset solver type

7.23. DD-ILU PRECONDITIONER

Scope: Bicgstab Solver

```
Begin Dd-Ilu Preconditioner
    Fill Level {=|are|is} value
    Reordering {=|are|is} value
    Subdomain Overlap Level {=|are|is} value
End
```

Summary Use a domain-decomposition ILU(k) preconditioner.

7.23.1. Fill Level

Scope: Dd-Ilu Preconditioner

`Fill Level {=|are|is}` *`value`*

Parameter	Value	Default
value	`integer`	`0`

Summary Set the level of fill, k, for ILU(k).

7.23.2. Reordering

Scope: Dd-Ilu Preconditioner

`Reordering {=|are|is}` *`value`*

Parameter	Value	Default
value	`{off|on}`	`ON`

Summary Determine whether or not to apply an RCM reordering.

7.23.3. Subdomain Overlap Level

Scope: Dd-Ilu Preconditioner

`Subdomain Overlap Level {=|are|is}` *`value`*

Parameter	Value	Default
value	`integer`	`0`

Summary Set the level of subdomain overlap used in constructing the preconditioner.

7.24. JACOBI PRECONDITIONER

Scope: Bicgstab Solver

```
Begin Jacobi Preconditioner

    Number Of Sweeps {=|are|is} value

End
```

Summary Use a Jacobi preconditioner.

7.24.1. Number Of Sweeps

Scope: Jacobi Preconditioner

`Number Of Sweeps {=|are|is}` *value*

Parameter	Value	Default
value	integer	3

Summary Set the number of relaxation sweeps to perform.

7.25. DD-ILUT PRECONDITIONER

Scope: Bicgstab Solver

```
Begin Dd-Ilut Preconditioner

    Drop Tolerance {=|are|is} value

    Fill Fraction {=|are|is} value

    Reordering {=|are|is} value

    Subdomain Overlap Level {=|are|is} value

End
```

Summary Use a domain-decomposition ILUT preconditioner.

7.25.1. Drop Tolerance

Scope: Dd-Ilut Preconditioner

`Drop Tolerance {=|are|is}` *value*

Parameter	Value	Default
value	real	0

Summary Set the drop tolerance for ilut.

Description Off-diagonal entries of the incomplete factorization smaller than the specified tolerance will be dropped.

7.25.2. Fill Fraction

Scope: Dd-Ilut Preconditioner

`Fill Fraction {=|are|is}` *`value`*

Parameter	Value	Default
value	`real`	1

Summary Set the fill fraction for ilut.

Description This parameter controls the number of nonzero entries that are kept in each row of the incomplete factorization. Larger values will use more memory and runtime, but may result in a more effective preconditioner.

7.25.3. Reordering

Scope: Dd-Ilut Preconditioner

`Reordering {=|are|is}` *`value`*

Parameter	Value	Default
value	`{off|on}`	ON

Summary Determine whether or not to apply an RCM reordering.

7.25.4. Subdomain Overlap Level

Scope: Dd-Ilut Preconditioner

`Subdomain Overlap Level {=|are|is}` *`value`*

Parameter	Value	Default
value	`integer`	0

Summary Set the level of subdomain overlap used in constructing the preconditioner.

7.26. MUELU PRECONDITIONER

Scope: Bicgstab Solver

```
Begin Muelu Preconditioner

    Xml File {=|are|is} value

End
```

Summary Use a MueLu preconditioner.

7.26.1. Xml File

Scope: Muelu Preconditioner

`Xml File {=|are|is}` *value*

Parameter	Value	Default
value	`string`	-

Summary Supply the name of an XML file containing a MueLu parameter list.

7.27. SGS PRECONDITIONER

Scope: Bicgstab Solver

```
Begin Sgs Preconditioner
    Number Of Sweeps {=|are|is} value
End
```

Summary Use a symmetric Gauss-Seidel preconditioner.

7.27.1. Number Of Sweeps

Scope: Sgs Preconditioner

`Number Of Sweeps {=|are|is}` *value*

Parameter	Value	Default
value	`integer`	3

Summary Set the number of relaxation sweeps to perform.

7.28. EXPERT IFPACK2 PRECONDITIONER

Scope: Bicgstab Solver

```
Begin Expert Ifpack2 Preconditioner
    Xml File {=|are|is} value
End
```

Summary Use an Ifpack2 preconditioner with parameters specified in an XML file.

7.28.1. Xml File

Scope: Expert Ifpack2 Preconditioner

`Xml File {=|are|is}` *`value`*

Parameter	Value	Default
value	`string`	-

Summary Supply the name of an XML file containing an Ifpack2 parameter list.

Description The supplied XML file must contain a ParameterList with a "preconditioner type:" string parameter corresponding to the desired type of Ifpack2 preconditioner as well as any additional parameters for that preconditioner.

8. PROCEDURE

The following commands are found in the Procedure scope of the input file.

8.1. FUEGO PROCEDURE

Scope: Sierra

```
Begin Fuego Procedure ProcedureName

    Debug Level {=|are|is} Level

    Disable Conjugate Heat Transfer Symmetry Checks

    Maximum Number Of Nonlinear Iterations {=|are|is} Number

    Skip Steps For Pmr {=|are|is} PMRSkip

    Time Start {=|are|is} StartTime Stop {=|are|is} StopTime Status
    Interval {=|are|is} StatusInterval

    Begin Apub Transfer Transfer_name
    End

    Begin Average Region Regionname
    End

    Begin Fuego Region Regionname
    End

    Begin Input_Output Region Parameter_block_name
    End

    Begin Particle Region Regionname
    End

    Begin Solution Control Description Name
    End

    Begin Time Control
```

```
    End

    Begin Transfer Transfer_name

    End

End
```

Summary Contains the commands needed to execute an analysis in this procedure.

8.1.1. Debug Level

Scope: Fuego Procedure

`Debug Level {=|are|is}` *Level*

Parameter	Value	Default
Level	integer	-

Summary Set the debug level.

Description Need description here.

8.1.2. Disable Conjugate Heat Transfer Symmetry Checks

Scope: Fuego Procedure

Summary Disables conjugate heat transfer symmetry checks.

Description Surface transfers for conjugate heat transfer are symmetric. For example, consider a Dirichlet-Neumann coupling scheme. A transfer from Region A sends the temperature field on surfaces "s1, s2" to be interpolated on surfaces "surf1, surf2" of Region B. If Region B sends the heat flux field only from surface "surf1" to be interpolated on surfaces "s1, s2" of Region A, then this transfer is said to be asymmetric. To be symmetric, Region B needs to send surfaces "surf1, surf2" to surfaces "s1, s2" of Region A. Expert users can bypass this default behavior by activating this flag to ignore checks which would normally throw an error when asymmetric surface transfers are detected. A warning will be printed to the log file.

8.1.3. Maximum Number Of Nonlinear Iterations

Scope: Fuego Procedure

`Maximum Number Of Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	1

Summary Set the number of procedure nonlinear iterations over the defined regions.

Description Need description here.

8.1.4. Skip Steps For Pmr

Scope: Fuego Procedure

`Skip Steps For Pmr {=|are|is}` *`PMRSkip`*

Parameter	Value	Default
PMRSkip	`integer`	1

Summary Controls the frequency of PMR solve.

Description Set the PMR solve interval. Default is 1 (every step). This can also be specified as a time block property. If both are specified, the time block property will be used.

8.1.5. Time Start

Scope: Fuego Procedure

`Time Start {=|are|is}` *`StartTime`* `Stop {=|are|is}` *`StopTime`* `Status Interval {=|are|is}` *`StatusInterval`*

Parameter	Value	Default
StartTime	`real`	-
StopTime	`real`	-
StatusInterval	`integer`	-

Summary Start and stop times for a procedure.

8.2. APUB TRANSFER

Scope: Fuego Procedure

```
Begin Apub Transfer Transfer_name

    Convective Coefficient Node Variable {=|are|is} FName

    Irradiation Node Variable {=|are|is} FName

    Nodes Outside Region Region name

    Reference Temperature Node Variable {=|are|is} FName

    Surface Gap Tolerance {=|are|is} Gap

    Transfer Between Fuego Fluid Region From_region_name Option1
    From_io_entity_name(s) And Fuego Conduction Region To_region_name
    Option2 To_io_entity_name(s) [ Using Option3 ]

    Transfer Between Input Region From_region_name And Balance Region
    To_region_name [ Using Option ]

    Transfer Between Region From_region_name Option1
    From_io_entity_name(s) And Region To_region_name Option2
    To_io_entity_name(s) [ Using Option3 ]

    Transfer Problem Definition {=|are|is} Problem...

End
```

Summary transfer region/mesh information. The variables information will get sorted out by the calling procedure.

8.2.1. Convective Coefficient Node Variable

Scope: Apub Transfer

```
Convective Coefficient Node Variable {=|are|is} FName
```

Parameter	Value	Default
FName	string	-

Summary Name of node variable to use for convective coefficient that is associated with this transfer operation for conjugate heat transfer.

8.2.2. Irradiation Node Variable

Scope: Apub Transfer

`Irradiation Node Variable {=|are|is}` *`FName`*

Parameter	Value	Default
FName	`string`	-

Summary Name of node variable to use for irradiation that is associated with this transfer operation for conduction-radiation transfer.

8.2.3. Nodes Outside Region

Scope: Apub Transfer

`Nodes Outside Region` *`Region name`*

Parameter	Value	Default
Region name	`string`	-

Summary deal with the nodes that fall outside the intersection of two regions for nodal interpolation

8.2.4. Reference Temperature Node Variable

Scope: Apub Transfer

`Reference Temperature Node Variable {=|are|is}` *`FName`*

Parameter	Value	Default
FName	`string`	-

Summary Name of node variable to use for convective coefficient that is associated with this transfer operation for conjugate heat transfer.

8.2.5. Surface Gap Tolerance

Scope: Apub Transfer

`Surface Gap Tolerance {=|are|is}` *`Gap`*

Parameter	Value	Default
Gap	`real`	-

Summary specify the gap tolerance for locating interacting nodes during surface transfers.

8.2.6. Transfer Between Fuego Fluid Region

Scope: Apub Transfer

`Transfer Between Fuego Fluid Region` *`From_region_name Option1 From_io_entity_name(s)`* `And Fuego Conduction Region` *`To_region_name Option2 To_io_entity_name(s)`* `[ Using` *`Option3`* `]`

Parameter	Value	Default
From_region_name	`string`	-
Option1	`{surface|volume}`	-
From_io_entity_name(s)	`string`	-
To_region_name	`string`	-
Option2	`{surface|volume}`	-
To_io_entity_name(s)	`string`	-

Summary transfer from fuego fluids region to fuego heat conduction region

8.2.7. Transfer Between Input Region

Scope: Apub Transfer

`Transfer Between Input Region` *`From_region_name`* `And Balance Region` *`To_region_name`* `[ Using` *`Option`* `]`

Parameter	Value	Default
From_region_name	`string`	-
To_region_name	`string`	-

Summary Transfer mesh from source region region to destination region. The mesh transfer uses the specified load-balancing algorithm. The default transfer operator is the Nodal Copy, and can also be specified by including the optional parameters 'USING COPY'. If 'USING INTERP' is specified, then the (general different-mesh) nodal interpolation transfer is used instead. The latter is intended to be used only for testing the nodal interpolation transfer operator.

8.2.8. Transfer Between Region

Scope: Apub Transfer

`Transfer Between Region` *`From_region_name Option1 From_io_entity_name(s)`* `And Region` *`To_region_name Option2 To_io_entity_name(s)`* `[ Using` *`Option3`* `]`

Parameter	Value	Default
From_region_name	`string`	-
Option1	`{surface|volume}`	-
From_io_entity_name(s)	`string`	-
To_region_name	`string`	-
Option2	`{surface|volume}`	-
To_io_entity_name(s)	`string`	-

Summary transfer from region/block to region/block

8.2.9. Transfer Problem Definition

Scope: Apub Transfer

`Transfer Problem Definition {=|are|is}` *`Problem...`*

Parameter	Value	Default
Problem	`string...`	-

Summary problem type for this transfer

8.3. TRANSFER

Scope: Fuego Procedure

```
Begin Transfer Transfer_name
    Abort If Field Not Defined On Copy Transfer Send Or Receive Object

    Abort If Search Object Outside Of Tolerance
    All Fields
    Copy Option1 Option2 From From_region_name To To_region_name
    Distance Function Is Closest Receive Node To Send Centroid
    Exclude Ghosted
    From Option1 To Option2
    Gauss Point Integration Order {=|are|is} Order
    Geometric Tolerance {=|are|is} Geometric_tolerance
    Inspect With File {=|are|is} File_name Ids {=|are|is} ID_list...

    Interpolate Option1 Option2 From From_region_name To
    To_region_name
```

```
Interpolation Function User_Subroutine

Nodes Outside Region {=|are|is} Option

Parametric Tolerance {=|are|is} Parametric_tolerance

Search Coordinate Field Source_field_name State Option1 To
Destination_field_name State Option2

Search Geometric Tolerance {=|are|is} Geometric_tolerance

Search Surface Gap Tolerance {=|are|is} Surface_gap_tolerance [
Or Less ]

Search Type {=|are|is} [ Option1 Option2 Option3 ]

Select One Receiver For Each Send Object

Select One Unique Receiver For Each Send Object

Send Predefined-transfer Fields

Send Block From_blocks...  To To_blocks...

Send Field Source_field_name State Option1 To
Destination_field_name State Option2 [ Lower Bound Lower_bound
Upper Bound Upper_bound ]

Begin Receive Blocks

End

Begin Send Blocks

End

End
```

Summary transfer region/mesh information. the mechanics/variables information will get sorted out by the calling procedure.

8.3.1. Abort If Field Not Defined On Copy Transfer Send Or Receive Object

Scope: Transfer

Summary For testing purposes only. Normally mesh objects in the send or receive mesh which do not have the specified field defined on them are just ignored. This line command allows the construction of tests in which it is known that every mesh object should have the specified field defined on it and to abort if that field is not found.

8.3.2. Abort If Search Object Outside Of Tolerance

Scope: Transfer

Summary For debugging purposes only. Abort transfer if search object lie outside of specified geometric tolerance.

8.3.3. All Fields

Scope: Transfer

Summary Select all fields for transfer that have same name and state for source and destination regions.

8.3.4. Copy

Scope: Transfer

`Copy` *`Option1 Option2`* `From` *`From_region_name`* `To` *`To_region_name`*

Parameter	Value	Default
Option1	`{surface|volume}`	-
Option2	`{constraints|elements|nodes}`	-
From_region_name	`string`	-
To_region_name	`string`	-

Summary Copy transfer elements, nodes or constraints from one region to another. The copy transfer is very specific in that the sending and receiving mesh parts must have identical global ids for every element to be copied. The copy transfer works by iterating over all the mesh objects in the receiving mesh and using the global id of the receiving mesh object to find a mesh object in the sending mesh with the same global id. The field to transfer is then copied from the sending to receiving objects. There is no interpolation and the actual coordinates of the sending and receiving objects are not used and could be very different. The copy transfer is used in very special cases where the same mesh was read into both the sending and receiving meshes, there was no element death and there was no adaptivity. In this special case, a copy transfer can be much faster than an interpolation transfer.

8.3.5. Distance Function Is Closest Receive Node To Send Centroid

Scope: Transfer

Summary To be used in conjunction with "SELECT ONE UNIQUE RECEIVER FOR EACH SEND OBJECT". This helped in the case where the sending and receiving element blocks did not overlap and an element transfer was using element centroids for the distance computation. The elements were very distorted so that a centroid of a surface element could be far from the surface. It was wanted that the receiving element be the one close to the surface of the block and close to the sending element in the adjacent block. Using the corner nodes was enough since it was a tet mesh with plane faces. In this particular and unusual case this alternative method of matching sending and receiving elements was useful, but it is not expected to be used often or maybe never again.

8.3.6. Exclude Ghosted

Scope: Transfer

Summary exclude ghosted nodes from a copy transfer

8.3.7. From

Scope: Transfer

`From` *`Option1`* `To` *`Option2`*

Parameter	Value	Default
Option1	`{constraints|elements|nodes}`	-
Option2	`{constraints|elements|gauss_points|nodes}`	-

Summary Allows the send/receive mesh objects to be different.

8.3.8. Gauss Point Integration Order

Scope: Transfer

`Gauss Point Integration Order {=|are|is}` *`Order`*

Parameter	Value	Default
Order	`integer`	-

Summary Integration order to use when transferring to Gauss points.

8.3.9. Geometric Tolerance

Scope: Transfer

`Geometric Tolerance {=|are|is}` *`Geometric_tolerance`*

Parameter	Value	Default
Geometric_tolerance	`real`	-

Summary

This is the dimensional tolerance applied during the initial (coarse) search. If specified, all the coarse search boxes are padded by this value. The default behavior is for this to be 1e-9 times the problem bounding box size (diagonal) plus a 10 percent relative expansion per element. If a value is specified for this, it is used as a padding in place of the default with no relative box expansion.

During the interpolation transfer there is a geometric search based on the coordinates of the send and receive objects. As part of this search, an axis aligned bounding box is contracted for each sending object and GEOMETRIC TOLERANCE is used to make this box bigger than just a tight bounding box. Lists of receiving points are then quickly found within these axis aligned boxes.

If all points in the receiving mesh are within at least one box, no additional searching needs to be done and the search algorithm is fast. If there are still points in the receiving mesh that were outside of EVERY box, then a warning message will be issued about an "expensive search for extrapolation" for these points. This 'expensive search" can be very costly if a large number of receiving objects fall into this category and this line command is provided for those special cases.

8.3.10. Inspect With File

Scope: Transfer

`Inspect With File {=|are|is}` *`File_name`* `Ids {=|are|is}` *`ID_list...`*

Parameter	Value	Default
File_name	`string`	-
ID_list	`integer...`	-

Summary

STK Transfer inspection tool that allows user to output the search results for a specified list of entity ids on the receive mesh. The rank of the entities is deduced from the type of transfer being set up.

8.3.11. Interpolate

Scope: Transfer

`Interpolate` *`Option1 Option2`* `From` *`From_region_name`* `To` *`To_region_name`*

Parameter	Value	Default
Option1	`{surface|volume}`	-
Option2	`{constraints|elements|nodes}`	-
From_region_name	`string`	-
To_region_name	`string`	-

Summary Interpolate will transfer elements, nodes or constraints from one mesh to another. The interpolation transfer is very general in that the field values to transfer will be interpolated from the sending to receiving mesh based on the coordinates of the sending and receiving mesh objects.

Many line commands can be used to modify the behavior of the interpolation transfer but the basic algorithm is straightforward. Every mesh object in the receiving mesh is converted into a point. For elements this is the average of the nodal coordinates. An element in the sending mesh containing this point is found. If the field to transfer is nodal, the element shape functions are used to interpolate the nodal field to the receiving point. If the field to transfer is elemental, a bi-linear least squares fit based upon neighboring elements is first performed and then used to define the interpolation of the element field at the receiving point.

8.3.12. Interpolation Function

Scope: Transfer

`Interpolation Function` *`User_Subroutine`*

Parameter	Value	Default
User_Subroutine	`string`	-

Summary Allows an application defined subroutine to be used for the interpolation. Normally the interpolation transfer will determine the best type of interpolation to use: Basis functions for nodal fields and a neighborhood least squares fit for element fields. This line command can be used to override this if needed. It also allows an application to register it's own special interpolation functions that can then be used if the special name it was registered with is known.

8.3.13. Nodes Outside Region

Scope: Transfer

`Nodes Outside Region {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`{extrapolate|ignore|project|truncate}`	-

Summary

This line command defines what to do when a receiving point is outside the scope of the sending mesh.

IGNORE - The receiving mesh object can be ignored and will receive no value. This is almost never a good idea as it can cause mesh objects just outside to have a zero value when the nodes just inside the mesh might have very large values. This can result in a discontinuous receiving field.

EXTRAPOLATE - This is the default behavior. The sending field is extrapolated beyond the bounds of the sending mesh. This can lead to extrapolation error, such as when a large gradient at the surface causes a negative values when only positive values are acceptable. If this happens to the upper and lower bounds that can be placed on the fields to be transferred with the SEND FIELD command.

TRUNCATE - The receiving coordinate is projected back to the surface of the sending mesh to determine a value. This ensures that the receiving value is outside of the field values in the sending mesh.

PROJECT - This option is similar to TRUNCATE in which the receiving coordinate is projected back to the surface of the sending mesh to determine a value. In this case more effort is made to make sure that the projection is normal to the surface in the sending mesh. Sometimes gives a better result than Truncate but is a little more expensive to compute.

If the PROJECT option is used in transferring of surface values, the sending mesh should envelope the receiving mesh. Failure to satisfy this condition will generally result in failure of the transfer.

8.3.14. Parametric Tolerance

Scope: Transfer

`Parametric Tolerance {=|are|is}` *`Parametric_tolerance`*

Parameter	Value	Default
Parametric_tolerance	`real`	-

8.3.15. Search Coordinate Field

Scope: Transfer

`Search Coordinate Field` *`Source_field_name`* `State` *`Option1`* `To` *`Destination_field_name`* `State` *`Option2`*

Parameter	Value	Default
Source_field_name	`string`	-
Option1	`{new|nm1|nm2|nm3|nm4|none|old}`	-
Destination_field_name	`string`	-
Option2	`{new|nm1|nm2|nm3|nm4|none|old}`	-

Summary Normally the interpolation transfers use the default coordinate field to determine geometry information. This line command can be used to specify an alternate field.

8.3.16. Search Geometric Tolerance

Scope: Transfer

`Search Geometric Tolerance {=|are|is}` *`Geometric_tolerance`*

Parameter	Value	Default
Geometric_tolerance	`real`	-

8.3.17. Search Surface Gap Tolerance

Scope: Transfer

`Search Surface Gap Tolerance {=|are|is}` *`Surface_gap_tolerance`* `[ Or Less ]`

Parameter	Value	Default
Surface_gap_tolerance	`real`	-

Summary This is the dimensional tolerance applied during the initial (coarse) search. If specified, all the coarse search boxes are padded by this value. The default behavior is for this to be 1e-9 times the problem bounding box size (diagonal) plus a 10 percent relative expansion per element. If a value is specified for this, it is used as a padding in place of the default with no relative box expansion.

This is a tricky parameter best ignored, let it default to something based on the problem size. During the interpolation transfer there is a geometric search based on the coordinates of the send and receive objects. As part of this search, an axis aligned bounding box is contracted for each sending object and SEARCH GAP TOLERANCE is used to make this box bigger than just a tight bounding box. Lists of receiving points are then quickly found within these axis aligned boxes.

If all points in the receiving mesh are within at least one box, no additional searching needs to be done and the search algorithm is fast. If there are still points in the receiving

mesh that were outside of EVERY box, then a warning message will be issued about an "expensive search for extrapolation" for these points. This 'expensive search" can be very costly if a large number of receiving objects fall into this category and this line command is provided for those special cases.

The OR LESS optional parameter is used when the tolerance must be set to large value for one part of the mesh but much of the mesh needs a much smaller value. In some cases it is necessary for the tolerance to be set to the actual largest surface gap tolerance which may be far too large a gap for the rest of the mesh. Setting OR LESS allows the search tolerance to be reduced in areas of the mesh thus resulting in a faster search.

8.3.18. Search Type

Scope: Transfer

8.3.19. Select One Receiver For Each Send Object

Scope: Transfer

Summary This option will cause each sending object to be used once and only once. This will have the side effect of some receiving objects not getting any value at all. If you use this option, you will also want to set NODES OUTSIDE REGION IGNORE The example which necessitated this option was a case in which there was a delta function defined on an element in the sending mesh. It was desirable that the delta functions be summed into the receiving mesh such that the total value of the sending was conserved. It was better to have only a single element on the receiving side have a non-zero value that was the sum of sending values and not worry about how close the receiving element was to the sending element. A check that this option is working is to use Encore to computer the sum of the values of the sending and receiving fields to make sure the total sum is the same.

8.3.20. Select One Unique Receiver For Each Send Object

Scope: Transfer

Summary An unusual flag to get around an odd problem. Normally each receive object transfers from the nearest sending object so it is almost always the case that a send object will be used multiple times to define a receiving value. This option will cause each sending object to be used only once. This will have the side effect of some receiving objects not getting any value at all. If you use this option, you will also want to set NODES OUTSIDE REGION IGNORE or else the uniqueness will be lost for nodes outside the sending region. The example which necessitated this option was a case in which there was a delta function defined on an element in the sending mesh. It was desirable that the delta function be defined on the receiving mesh for only a single element in the

neighborhood of the sending element. The analysis was more sensitive to the number of delta functions on the receiving side than the location. So it was better to have only a single element on the receiving side have a non-zero value and not worry about how close the receiving element was to the sending element.

8.3.21. Send

Scope: Transfer

`Send` *`Predefined-transfer`* `Fields`

Parameter	Value	Default
Predefined-transfer	`{advection}`	-

Summary Use predefine transfer semantics provided by the specified name.

8.3.22. Send Block

Scope: Transfer

`Send Block` *`From_blocks...`* `To` *`To_blocks...`*

Parameter	Value	Default
From_blocks	`string...`	-
To_blocks	`string...`	-

Summary Add element blocks to a particular same mesh element copy transfer operator.

The copy transfer can have multiple of these lines to define many blocks, but each line sends a single block to a single block: SEND BLOCK block_1 TO block_1 SEND BLOCK block_101 TO block_101

The interpolation transfer can have only a single SEND BLOCK line, but can define many from/to blocks: SEND BLOCK block_3 block_5 block_6 TO block_3 block_5

8.3.23. Send Field

Scope: Transfer

`Send Field` *`Source_field_name`* `State` *`Option1`* `To` *`Destination_field_name`* `State` *`Option2`* `[ Lower Bound` *`Lower_bound`* `Upper Bound` *`Upper_bound`* `]`

Parameter	Value	Default
Source_field_name	`string`	-
Option1	`{new | nm1 | nm2 | nm3 | nm4 | none | old}`	-
Destination_field_name	`string`	-
Option2	`{new | nm1 | nm2 | nm3 | nm4 | none | old}`	-

Summary

Specifies the mapping between source and destination field names. Vector and tensor fields can be subscripted using parenthesis and 1's based or brackets and 0 based. Notes on subscripting: (0) Does not work for COPY transfers, only INTERPOLATION type transfers. (1) If the field name itself actually contains either parenthesis or brackets then we are in trouble and an error is going to be thrown due to a syntax error in index specification. (2) Only a single subscript is allowed so vectors of vectors or higher order tensors can not use double subscripts. But it should be possible to determine the correct offset within the field and pick out the correct value with a little effort. (3) Once subscripted, only a single value will be transferred. It is not possible to transfer multiple values starting at a certain index, instead multiple line commands must be used, as shown above. (4) The indexes can be 0 based with brackets or 1 based when using parenthesis. Although this could be very confusing if mixed within a single line command. (5) Both the from and to fields can be subscripted independently on the same line.

example SEND FIELD velocity TO velocity SEND FIELD temp TO temperature lower bound 0 SEND FIELD x TO y lower bound 10 upper bound 100 SEND FIELD A(2) TO B(3) lower bound 10 upper bound 100 SEND FIELD A[1] TO B[2] lower bound 10 upper bound 100

9. TIME CONTROL

9.1. TIME CONTROL

Scope: Fuego Procedure

```
Begin Time Control

    Termination Time {=|are|is} Tend

    Begin Time Stepping Block BlockName

    End

End
```

Summary Defines the time control for the problem in the procedure scope.

Description This describes the Time Control block which is convenient for single-physics applications. The Solution Control block is described elsewhere, and is appropriate for multiphysics applications.

9.1.1. Termination Time

Scope: Time Control

```
Termination Time {=|are|is} Tend
```

Parameter	Value	Default
Tend	`real`	-

Summary Specifies the ending time for this solution period.

9.2. TIME STEPPING BLOCK

Scope: Time Control

```
Begin Time Stepping Block BlockName

    Number Of Time Steps {=|are|is} Nsteps

    Start Time {=|are|is} Tstart

    Termination Time {=|are|is} Tend

    Time Step {=|are|is} Dt0

    Begin Parameters For Fuego Region RegionName

    End

End
```

Summary Defines the time and time step parameters for a solution period.

9.2.1. Number Of Time Steps

Scope: Time Stepping Block

`Number Of Time Steps {=|are|is} Nsteps`

Parameter	Value	Default
Nsteps	integer	-

Summary Specifies the number of time steps in this solution period.

9.2.2. Start Time

Scope: Time Stepping Block

`Start Time {=|are|is} Tstart`

Parameter	Value	Default
Tstart	real	-

Summary Specifies the starting time for this solution period.

9.2.3. Termination Time

Scope: Time Stepping Block

`Termination Time {=|are|is}` *`Tend`*

Parameter	Value	Default
Tend	`real`	-

Summary Specifies the ending time for this solution period.

9.2.4. Time Step

Scope: Time Stepping Block

`Time Step {=|are|is}` *`Dt0`*

Parameter	Value	Default
Dto	`real`	-

Summary Specifies the time step size for this solution period.

9.3. PARAMETERS FOR FUEGO REGION

Scope: Parameters For

```
Begin Parameters For Fuego Region RegionName
    Cfl Limit {=|are|is} Clfmax
    Iteration At Skip Step {=|are|is} SkipIter
    Maximum Time Step {=|are|is} DtMax
    Minimum Time Step {=|are|is} DtMin
    Skip Step {=|are|is} SkipStep
    Skip Steps For Pmr {=|are|is} PMRSkip
    Time Step Change Factor {=|are|is} Change
    Transient Step Type {=|are|is} FuegoStepType
End
```

Summary Defines region-specific time stepping data.

9.3.1. Cfl Limit

Scope: Parameters For Fuego Region

`Cfl Limit {=|are|is}` *`Clfmax`*

Parameter	Value	Default
Clfmax	`real`	`0.5`

Summary Specifies the maximum value of the CFL number that is used to determine the auto time step.

Description Need a fuller description here.

9.3.2. Iteration At Skip Step

Scope: Parameters For Fuego Region

`Iteration At Skip Step {=|are|is}` *`SkipIter`*

Parameter	Value	Default
SkipIter	`integer`	-

Summary Controls the number of iterations in between a skip step.

Description This is the first implementation of the ability to freeze the Fuego region and allow time advancement without a Fluids/Conduction solve. In the near future, solution control will allow this feature in a more elegant manner.

For now, the iteration count will allow for a given number of iterations when the Afgo regions solve is "active", that is not during a skip step.

9.3.3. Maximum Time Step

Scope: Parameters For Fuego Region

`Maximum Time Step {=|are|is}` *`DtMax`*

Parameter	Value	Default
DtMax	`real`	`1.0e12`

Summary Specifies the maximum allowable time increment for auto timestepping.

Description Need a fuller description here.

9.3.4. Minimum Time Step

Scope: Parameters For Fuego Region

`Minimum Time Step {=|are|is}` *`DtMin`*

Parameter	Value	Default
DtMin	`real`	`0.0`

Summary Specifies the minimum allowable time increment for auto timestepping.

Description Need a fuller description here.

9.3.5. Skip Step

Scope: Parameters For Fuego Region

`Skip Step {=|are|is}` *`SkipStep`*

Parameter	Value	Default
SkipStep	`integer`	-

Summary Controls the frequency of Fuego (Afgo) iteration.

Description This is the first implementation of the ability to freeze the Fuego region and allow time advancement without a Fluids/Conduction solve. In the near future, solution control will allow this feature in a more elegant manner.

9.3.6. Skip Steps For Pmr

Scope: Parameters For Fuego Region

`Skip Steps For Pmr {=|are|is}` *`PMRSkip`*

Parameter	Value	Default
PMRSkip	`integer`	`1`

Summary Controls the frequency of PMR solve.

Description Set the PMR solve interval. Default is 1 (every step).

9.3.7. Time Step Change Factor

Scope: Parameters For Fuego Region

`Time Step Change Factor {=|are|is}` *`Change`*

Parameter	Value	Default
Change	`real`	`10.0`

Summary Specify the factor by which the time step is allowed to increase or decrease between time steps.

Description Need a fuller description here.

9.3.8. Transient Step Type

Scope: Parameters For Fuego Region

`Transient Step Type {=|are|is}` *`FuegoStepType`*

Parameter	Value	Default
FuegoStepType	`{automatic|fixed|invalid time step type}`	`FIXED`

Summary Specifies whether to use fixed time-stepping or automatic time-stepping.

Description Need a fuller description here.

10. SOLUTION CONTROL

10.1. SOLUTION CONTROL DESCRIPTION

Scope: Fuego Procedure

```
Begin Solution Control Description Name

    Use System Name

    Begin Initialize Name

    End

    Begin Parameters For

    End

    Begin System Name

    End

End
```

Summary Contains the commands needed to execute an analysis using the arpeggio procedure that uses Solver Control.

10.1.1. Use System

Scope: Solution Control Description

`Use System` *Name*

Parameter	Value	Default
Name	`string`	-

Summary This set the name of which system to use.

10.2. INITIALIZE

Scope: Solution Control Description

```
Begin Initialize Name

    Advance Name...  [ When When-expression ]

    Event Name...  [ When When-expression ]

    Involve Name

    Transfer Name [ When When-expression ]

End
```

Summary This block wraps a initializer for a given name. The NAME parameter is the name used to define the initialization block. There can be more than one initialize block in the Solver Control Description block. The "use initialize NAME" line commmand controls which one is to be used.

10.2.1. Advance

Scope: Initialize

```
Advance Name...  [ When When-expression ]
```

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that advances the solution. The name is that matches the physics.

10.2.2. Event

Scope: Initialize

```
Event Name...  [ When When-expression ]
```

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that has no time associated with it. It can cause a solution transfer between regions or cause something to print.

10.2.3. Involve

Scope: Initialize

`Involve` *`Name`*

Parameter	Value	Default
Name	`string`	-

Summary Specifiy a physics participant to a coupled problem solved using matrix-free nonlinear.

10.2.4. Transfer

Scope: Initialize

`Transfer` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Transfer line command which executes all transfers defined from the specified region. All transfers with a send region of 'name' will be executed.

10.3. PARAMETERS FOR

Scope: Solution Control Description

```
Begin Parameters For

    Converged When Convergence-expression

    Incremental Number Of Steps {=|are|is} Number

    Initial Deltat {=|are|is} Number

    Number Of Adaptivity Steps {=|are|is} Number

    Number Of Steps {=|are|is} Number

    Reinitialize Transient

    Start Time {=|are|is} Number

    Termination Time {=|are|is} Number

    Time Step Quantum {=|are|is} TimeStepQuantum

    Time Step Style TimeStepStyle...

    Total Change In Time {=|are|is} Number

    Begin Parameters For Fuego Region RegionName
```

```
    End

End
```

Summary A Solver Control PARAMETERS block to set up control data for the SC_type parameter. Inside this block one sets the time step parameters or nonlinear parameters.

10.3.1. Converged When

Scope: Parameters For

`Converged When` *`Convergence-expression`*

Parameter	Value	Default
Convergence-expression	`(expression)`	-

Summary Set the convergence expression.

10.3.2. Incremental Number Of Steps

Scope: Parameters For

`Incremental Number Of Steps {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	-

Summary The incremental number steps to run the time for nonlinear loop. Number of time steps to run after restarting. NUMBER OF STEPS is total number of steps to run

10.3.3. Initial Deltat

Scope: Parameters For

`Initial Deltat {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`real`	-

Summary Assign an initial delta T

10.3.4. Number Of Adaptivity Steps

Scope: Parameters For

`Number Of Adaptivity Steps {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	-

Summary The number steps to run the time or nonlinear loop

10.3.5. Number Of Steps

Scope: Parameters For

`Number Of Steps {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	-

Summary The number steps to run the time for nonlinear loop

10.3.6. Reinitialize Transient

Scope: Parameters For

Summary Reset time and re-initialize regions each step of the adaptivity loop.

10.3.7. Start Time

Scope: Parameters For

`Start Time {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`real`	-

Summary Assign a start time.

10.3.8. Termination Time

Scope: Parameters For

`Termination Time {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`real`	-

Summary Assign a final time to stop

10.3.9. Time Step Quantum

Scope: Parameters For

`Time Step Quantum {=|are|is}` *`TimeStepQuantum`*

Parameter	Value	Default
TimeStepQuantum	`real`	-

Summary Set the time stepping quantum time for SNAP style stepping.

10.3.10. Time Step Style

Scope: Parameters For

`Time Step Style` *`TimeStepStyle...`*

Parameter	Value	Default
TimeStepStyle	`{clip|noclip|nosnap|snap}`	`CLIP NOSNAP`

Summary Set the time stepping style.

When CLIP is specified, the time step size will be clipped at the last step of the transient loop so that it ends at the transient loop's end time. If clip is not specified, the last time is allowed to exceed to the transient loop's end time and the following transient loop will start at the exceeded end time.

When SNAP is specified, the time step is broken down into "quantum" time units. By default this quantum time is 12 orders of magnitude down from the difference between the start and end time for the transient loop. This value can be overridden using the TIME STEP QUANTUM line command. All time values are "snapped" to multiples of the quantum time by rounding to the nearest quantum multiple.

10.3.11. Total Change In Time

Scope: Parameters For

`Total Change In Time {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`real`	-

Summary Use this number and the initial time to compute termination time.

10.4. PARAMETERS FOR FUEGO REGION

Scope: Parameters For

```
Begin Parameters For Fuego Region RegionName

    Cfl Limit {=|are|is} Clfmax

    Iteration At Skip Step {=|are|is} SkipIter

    Maximum Time Step {=|are|is} DtMax

    Minimum Time Step {=|are|is} DtMin

    Skip Step {=|are|is} SkipStep

    Skip Steps For Pmr {=|are|is} PMRSkip

    Time Step Change Factor {=|are|is} Change

    Transient Step Type {=|are|is} FuegoStepType

End
```

Summary Defines region-specific time stepping data.

10.4.1. Cfl Limit

Scope: Parameters For Fuego Region

`Cfl Limit {=|are|is}` *Clfmax*

Parameter	Value	Default
Clfmax	`real`	`0.5`

Summary Specifies the maximum value of the CFL number that is used to determine the auto time step.

Description Need a fuller description here.

10.4.2. Iteration At Skip Step

Scope: Parameters For Fuego Region

`Iteration At Skip Step {=|are|is}` *`SkipIter`*

Parameter	Value	Default
SkipIter	`integer`	-

Summary Controls the number of iterations in between a skip step.

Description This is the first implementation of the ability to freeze the Fuego region and allow time advancement without a Fluids/Conduction solve. In the near future, solution control will allow this feature in a more elegant manner.

For now, the iteration count will allow for a given number of iterations when the Afgo regions solve is "active", that is not during a skip step.

10.4.3. Maximum Time Step

Scope: Parameters For Fuego Region

`Maximum Time Step {=|are|is}` *`DtMax`*

Parameter	Value	Default
DtMax	`real`	`1.0e12`

Summary Specifies the maximum allowable time increment for auto timestepping.

Description Need a fuller description here.

10.4.4. Minimum Time Step

Scope: Parameters For Fuego Region

`Minimum Time Step {=|are|is}` *`DtMin`*

Parameter	Value	Default
DtMin	`real`	`0.0`

Summary Specifies the minimum allowable time increment for auto timestepping.

Description Need a fuller description here.

10.4.5. Skip Step

Scope: Parameters For Fuego Region

`Skip Step {=|are|is}` *`SkipStep`*

Parameter	Value	Default
SkipStep	`integer`	-

Summary Controls the frequency of Fuego (Afgo) iteration.

Description This is the first implementation of the ability to freeze the Fuego region and allow time advancement without a Fluids/Conduction solve. In the near future, solution control will allow this feature in a more elegant manner.

10.4.6. Skip Steps For Pmr

Scope: Parameters For Fuego Region

`Skip Steps For Pmr {=|are|is}` *`PMRSkip`*

Parameter	Value	Default
PMRSkip	`integer`	`1`

Summary Controls the frequency of PMR solve.

Description Set the PMR solve interval. Default is 1 (every step).

10.4.7. Time Step Change Factor

Scope: Parameters For Fuego Region

`Time Step Change Factor {=|are|is}` *`Change`*

Parameter	Value	Default
Change	`real`	`10.0`

Summary Specify the factor by which the time step is allowed to increase or decrease between time steps.

Description Need a fuller description here.

10.4.8. Transient Step Type

Scope: Parameters For Fuego Region

`Transient Step Type {=|are|is}` *`FuegoStepType`*

Parameter	Value	Default
FuegoStepType	{automatic|fixed|invalid time step type}	FIXED

Summary Specifies whether to use fixed time-stepping or automatic time-stepping.

Description Need a fuller description here.

10.5. SYSTEM

Scope: Solution Control Description

`Begin System` *`Name`*

`Adapt` *`Region_name...`* `Using` *`Field_name...`* `[ When` *`When-expression`* `]`

`Compute Indicator On` *`Region_name...`* `Using` *`Indicator_name...`* `[ When` *`When-expression`* `]`

`Event` *`Name...`* `[ When` *`When-expression`* `]`

`Execute Postprocessor Group` *`Group_name...`* `On` *`Region_name...`* `[ When` *`When-expression`* `]`

`Indicatemarkadapt` *`Region_name`* `Using` *`Indicator Marker`* `[ When` *`When-expression`* `]`

`Mark` *`Region_name...`* `Using` *`Marker_name...`* `[ When` *`When-expression`* `]`

`Markadapt` *`Region_name`* `Using` *`Marker`* `[ When` *`When-expression`* `]`

`Output` *`Name`* `[ When` *`When-expression`* `]`

`Simulation Max Global Iterations {=|are|is}` *`Number`*

`Simulation Start Time {=|are|is}` *`Number`*

`Simulation Termination Time {=|are|is}` *`Number`*

`Transfer` *`Name`* `[ When` *`When-expression`* `]`

`Use Initialize` *`Name`*

`Begin Adaptivity` *`Name`*

`End`

```
    Begin Sequential Name

    End

    Begin Transient Name

    End

End
```

Summary This block wraps a solver system for a given name. The NAME parameter is the name used to define the system. There can be more than one system block in the Solver Control Description block. The "use system NAME" line commmand controls which one is to be used.

10.5.1. Adapt

Scope: System

```
Adapt Region_name...  Using Field_name...  [ When When-expression ]
```

Parameter	Value	Default
Region_name	string...	-
Field_name	string...	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.5.2. Compute Indicator On

Scope: System

```
Compute Indicator On Region_name...  Using Indicator_name...  [ When
When-expression ]
```

Parameter	Value	Default
Region_name	string...	-
Indicator_name	string...	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.5.3. Event

Scope: System

`Event` *`Name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that has no time associated with it. It can cause a solution transfer between regions or cause something to print.

10.5.4. Execute Postprocessor Group

Scope: System

`Execute Postprocessor Group` *`Group_name...`* `On` *`Region_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Group_name	`string...`	-
Region_name	`string...`	-

Summary Used within a Solver Control block to cause the group named group_name to be executed on region region_name.

10.5.5. Indicatemarkadapt

Scope: System

`Indicatemarkadapt` *`Region_name`* `Using` *`Indicator Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Indicator	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Compute Indicator On ... Mark ... Adapt ...

10.5.6. Mark

Scope: System

`Mark` *`Region_name...`* `Using` *`Marker_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Marker_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.5.7. Markadapt

Scope: System

`Markadapt` *`Region_name`* `Using` *`Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Mark ... Adapt ...

10.5.8. Output

Scope: System

`Output` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Output line command which execute a perform I/O on the region.

10.5.9. Simulation Max Global Iterations

Scope: System

`Simulation Max Global Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	-

Summary The Total number of Solves.

10.5.10. Simulation Start Time

Scope: System

`Simulation Start Time {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`real`	-

Summary Simulation starting time. (by default 0.0)

10.5.11. Simulation Termination Time

Scope: System

`Simulation Termination Time {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`real`	-

Summary The drop dead time.

10.5.12. Transfer

Scope: System

`Transfer` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Transfer line command which executes all transfers defined from the specified region. All transfers with a send region of 'name' will be executed.

10.5.13. Use Initialize

Scope: System

`Use Initialize` *`Name`*

Parameter	Value	Default
Name	`string`	-

Summary This set the name of which initialization to use.

10.6. ADAPTIVITY

Scope: Sequential

```
Begin Adaptivity Name

    Adapt Region_name... Using Field_name... [ When When-expression
    ]

    Advance Name... [ When When-expression ]

    Compute Indicator On Region_name... Using Indicator_name... [
    When When-expression ]

    Event Name... [ When When-expression ]

    Execute Postprocessor Group Group_name... On Region_name... [
    When When-expression ]

    Indicatemarkadapt Region_name Using Indicator Marker [ When
    When-expression ]

    Mark Region_name... Using Marker_name... [ When When-expression
    ]

    Markadapt Region_name Using Marker [ When When-expression ]

    Transfer Name [ When When-expression ]

    Begin Sequential Name

    End

    Begin Transient Name

    End

End
```

Summary This block is used to wrap an adapt loop.

10.6.1. Adapt

Scope: Adaptivity

`Adapt` *`Region_name...`* `Using` *`Field_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Field_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.6.2. Advance

Scope: Adaptivity

`Advance` *`Name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that advances the solution. The name is that matches the physics.

10.6.3. Compute Indicator On

Scope: Adaptivity

`Compute Indicator On` *`Region_name...`* `Using` *`Indicator_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Indicator_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.6.4. Event

Scope: Adaptivity

`Event` *`Name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that has no time associated with it. It can cause a solution transfer between regions or cause something to print.

10.6.5. Execute Postprocessor Group

Scope: Adaptivity

`Execute Postprocessor Group` *`Group_name...`* `On` *`Region_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Group_name	`string...`	-
Region_name	`string...`	-

Summary Used within a Solver Control block to cause the group named group_name to be executed on region region_name.

10.6.6. Indicatemarkadapt

Scope: Adaptivity

`Indicatemarkadapt` *`Region_name`* `Using` *`Indicator Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Indicator	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Compute Indicator On ... Mark ... Adapt ...

10.6.7. Mark

Scope: Adaptivity

`Mark` *`Region_name...`* `Using` *`Marker_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Marker_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.6.8. Markadapt

Scope: Adaptivity

`Markadapt` *`Region_name`* `Using` *`Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Mark ... Adapt ...

10.6.9. Transfer

Scope: Adaptivity

`Transfer` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Transfer line command which executes all transfers defined from the specified region. All transfers with a send region of 'name' will be executed.

10.7. SEQUENTIAL

Scope: Adaptivity

`Begin Sequential` *`Name`*

`Adapt` *`Region_name...`* `Using` *`Field_name...`* `[ When` *`When-expression`* `]`

`Advance` *`Name...`* `[ When` *`When-expression`* `]`

```
Compute Indicator On Region_name... Using Indicator_name... [
When When-expression ]

Event Name... [ When When-expression ]

Execute Postprocessor Group Group_name... On Region_name... [
When When-expression ]

Indicatemarkadapt Region_name Using Indicator Marker [ When
When-expression ]

Involve Name

Mark Region_name... Using Marker_name... [ When When-expression
]

Markadapt Region_name Using Marker [ When When-expression ]

Output Name [ When When-expression ]

Transfer Name [ When When-expression ]

Begin Adaptivity Name

End


Begin Nonlinear Name

End

End
```

Summary This block is used to wrap a sequential solution. It is used to wrap a sequence of Non-Linear or pseudo time solve step solves.

10.7.1. Adapt

Scope: Sequential

```
Adapt Region_name... Using Field_name... [ When When-expression ]
```

Parameter	Value	Default
Region_name	string...	-
Field_name	string...	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.7.2. Advance

Scope: Sequential

`Advance` *`Name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that advances the solution. The name is that matches the physics.

10.7.3. Compute Indicator On

Scope: Sequential

`Compute Indicator On` *`Region_name...`* `Using` *`Indicator_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Indicator_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.7.4. Event

Scope: Sequential

`Event` *`Name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that has no time associated with it. It can cause a solution transfer between regions or cause something to print.

10.7.5. Execute Postprocessor Group

Scope: Sequential

`Execute Postprocessor Group` *`Group_name...`* `On` *`Region_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Group_name	`string...`	-
Region_name	`string...`	-

Summary Used within a Solver Control block to cause the group named group_name to be executed on region region_name.

10.7.6. Indicatemarkadapt

Scope: Sequential

`Indicatemarkadapt` *`Region_name`* `Using` *`Indicator Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Indicator	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Compute Indicator On ... Mark ... Adapt ...

10.7.7. Involve

Scope: Sequential

`Involve` *`Name`*

Parameter	Value	Default
Name	`string`	-

Summary Specifiy a physics participant to a coupled problem solved using matrix-free nonlinear.

10.7.8. Mark

Scope: Sequential

`Mark` *`Region_name...`* `Using` *`Marker_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Marker_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.7.9. Markadapt

Scope: Sequential

`Markadapt` *`Region_name`* `Using` *`Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Mark ... Adapt ...

10.7.10. Output

Scope: Sequential

`Output` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Output line command which execute a perform I/O on the region.

10.7.11. Transfer

Scope: Sequential

`Transfer` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Transfer line command which executes all transfers defined from the specified region. All transfers with a send region of 'name' will be executed.

10.8. TRANSIENT

Scope: Adaptivity

```
Begin Transient Name

    Adapt Region_name...  Using Field_name...  [ When When-expression
    ]

    Advance Name...  [ When When-expression ]

    Compute Indicator On Region_name...  Using Indicator_name...  [
    When When-expression ]

    Event Name...  [ When When-expression ]

    Execute Postprocessor Group Group_name...  On Region_name...  [
    When When-expression ]

    Indicatemarkadapt Region_name Using Indicator Marker [ When
    When-expression ]

    Involve Name

    Mark Region_name...  Using Marker_name...  [ When When-expression
    ]

    Markadapt Region_name Using Marker [ When When-expression ]

    Output Name [ When When-expression ]

    Transfer Name [ When When-expression ]

    Begin Adaptivity Name

    End


    Begin Nonlinear Name

    End


    Begin Subcycle Name

    End

End
```

Summary This block is used to wrap a time loop.

10.8.1. Adapt

Scope: Transient

`Adapt` *`Region_name...`* `Using` *`Field_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Field_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.8.2. Advance

Scope: Transient

`Advance` *`Name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that advances the solution. The name is that matches the physics.

10.8.3. Compute Indicator On

Scope: Transient

`Compute Indicator On` *`Region_name...`* `Using` *`Indicator_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Indicator_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.8.4. Event

Scope: Transient

`Event` *`Name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that has no time associated with it. It can cause a solution transfer between regions or cause something to print.

10.8.5. Execute Postprocessor Group

Scope: Transient

`Execute Postprocessor Group` *`Group_name...`* `On` *`Region_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Group_name	`string...`	-
Region_name	`string...`	-

Summary Used within a Solver Control block to cause the group named group_name to be executed on region region_name.

10.8.6. Indicatemarkadapt

Scope: Transient

`Indicatemarkadapt` *`Region_name`* `Using` *`Indicator Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Indicator	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Compute Indicator On ... Mark ... Adapt ...

10.8.7. Involve

Scope: Transient

`Involve` *`Name`*

Parameter	Value	Default
Name	`string`	-

Summary Specifiy a physics participant to a coupled problem solved using matrix-free nonlinear.

10.8.8. Mark

Scope: Transient

`Mark` *`Region_name...`* `Using` *`Marker_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Marker_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.8.9. Markadapt

Scope: Transient

`Markadapt` *`Region_name`* `Using` *`Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Mark ... Adapt ...

10.8.10. Output

Scope: Transient

`Output` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Output line command which execute a perform I/O on the region.

10.8.11. Transfer

Scope: Transient

`Transfer` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Transfer line command which executes all transfers defined from the specified region. All transfers with a send region of 'name' will be executed.

10.9. NONLINEAR

Scope: Sequential

```
Begin Nonlinear Name

    Adapt Region_name... Using Field_name... [ When When-expression ]

    Advance Name... [ When When-expression ]

    Compute Indicator On Region_name... Using Indicator_name... [ When When-expression ]

    Event Name... [ When When-expression ]

    Execute Postprocessor Group Group_name... On Region_name... [ When When-expression ]

    Indicatemarkadapt Region_name Using Indicator Marker [ When When-expression ]

    Involve Name

    Mark Region_name... Using Marker_name... [ When When-expression ]

    Markadapt Region_name Using Marker [ When When-expression ]

    Output Name [ When When-expression ]

    Transfer Name [ When When-expression ]

    Begin Subcycle Name

    End

End
```

Summary This block is used to wrap a nonlinear solve loop.

10.9.1. Adapt

Scope: Nonlinear

```
Adapt Region_name... Using Field_name... [ When When-expression ]
```

Parameter	Value	Default
Region_name	string...	-
Field_name	string...	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.9.2. Advance

Scope: Nonlinear

`Advance` *`Name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that advances the solution. The name is that matches the physics.

10.9.3. Compute Indicator On

Scope: Nonlinear

`Compute Indicator On` *`Region_name...`* `Using` *`Indicator_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Indicator_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.9.4. Event

Scope: Nonlinear

`Event` *`Name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that has no time associated with it. It can cause a solution transfer between regions or cause something to print.

10.9.5. Execute Postprocessor Group

Scope: Nonlinear

`Execute Postprocessor Group` *`Group_name...`* `On` *`Region_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Group_name	`string...`	-
Region_name	`string...`	-

Summary Used within a Solver Control block to cause the group named group_name to be executed on region region_name.

10.9.6. Indicatemarkadapt

Scope: Nonlinear

`Indicatemarkadapt` *`Region_name`* `Using` *`Indicator Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Indicator	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Compute Indicator On ... Mark ... Adapt ...

10.9.7. Involve

Scope: Nonlinear

`Involve` *`Name`*

Parameter	Value	Default
Name	`string`	-

Summary Specifiy a physics participant to a coupled problem solved using matrix-free nonlinear.

10.9.8. Mark

Scope: Nonlinear

`Mark` *`Region_name...`* `Using` *`Marker_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Marker_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.9.9. Markadapt

Scope: Nonlinear

`Markadapt` *`Region_name`* `Using` *`Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Mark ... Adapt ...

10.9.10. Output

Scope: Nonlinear

`Output` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Output line command which execute a perform I/O on the region.

10.9.11. Transfer

Scope: Nonlinear

`Transfer` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Transfer line command which executes all transfers defined from the specified region. All transfers with a send region of 'name' will be executed.

10.10. SUBCYCLE

Scope: Nonlinear

```
Begin Subcycle Name

    Adapt Region_name...  Using Field_name...  [ When When-expression
    ]

    Advance Name...  [ When When-expression ]

    Compute Indicator On Region_name...  Using Indicator_name...  [
    When When-expression ]

    Event Name...  [ When When-expression ]

    Execute Postprocessor Group Group_name...  On Region_name...  [
    When When-expression ]

    Indicatemarkadapt Region_name Using Indicator Marker [ When
    When-expression ]

    Involve Name

    Mark Region_name...  Using Marker_name...  [ When When-expression
    ]

    Markadapt Region_name Using Marker [ When When-expression ]

    Output Name [ When When-expression ]

    Transfer Name [ When When-expression ]

End
```

Summary This block is used to wrap a subcycle time loop.

10.10.1. Adapt

Scope: Subcycle

```
Adapt Region_name...  Using Field_name...  [ When When-expression ]
```

Parameter	Value	Default
Region_name	string...	-
Field_name	string...	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.10.2. Advance

Scope: Subcycle

`Advance` *`Name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that advances the solution. The name is that matches the physics.

10.10.3. Compute Indicator On

Scope: Subcycle

`Compute Indicator On` *`Region_name...`* `Using` *`Indicator_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Indicator_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.10.4. Event

Scope: Subcycle

`Event` *`Name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string...`	-

Summary Used within a Solver Control block to indicate a single step that has no time associated with it. It can cause a solution transfer between regions or cause something to print.

10.10.5. Execute Postprocessor Group

Scope: Subcycle

`Execute Postprocessor Group` *`Group_name...`* `On` *`Region_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Group_name	`string...`	-
Region_name	`string...`	-

Summary Used within a Solver Control block to cause the group named group_name to be executed on region region_name.

10.10.6. Indicatemarkadapt

Scope: Subcycle

`Indicatemarkadapt` *`Region_name`* `Using` *`Indicator Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Indicator	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Compute Indicator On ... Mark ... Adapt ...

10.10.7. Involve

Scope: Subcycle

`Involve` *`Name`*

Parameter	Value	Default
Name	`string`	-

Summary Specifiy a physics participant to a coupled problem solved using matrix-free nonlinear.

10.10.8. Mark

Scope: Subcycle

`Mark` *`Region_name...`* `Using` *`Marker_name...`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string...`	-
Marker_name	`string...`	-

Summary Used within a Solver Control block to indicate a mesh adaptment on the specific block should be performed.

10.10.9. Markadapt

Scope: Subcycle

`Markadapt` *`Region_name`* `Using` *`Marker`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Region_name	`string`	-
Marker	`string`	-

Summary Shortcut line command... equivalent to: Mark ... Adapt ...

10.10.10. Output

Scope: Subcycle

`Output` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Output line command which execute a perform I/O on the region.

10.10.11. Transfer

Scope: Subcycle

`Transfer` *`Name`* `[ When` *`When-expression`* `]`

Parameter	Value	Default
Name	`string`	-

Summary A Solver Control Transfer line command which executes all transfers defined from the specified region. All transfers with a send region of 'name' will be executed.

11. FUEGO REGION

The following commands are found in the Fuego Region scope of the input file.

11.1. FUEGO REGION

Scope: Fuego Procedure

Begin Fuego Region *Regionname*

Allow Inverted Elements

Calculate Mesh Quality Metrics

Compute Difference *NormTypes* Of *f1 f2* [Store In *field_name*]

Compute Norm *NormTypes* Of *Function_name* [Store In *field_name*]

Create Edge Field *Field_name* Of Type *Option* And Dimension *Dimension* [On *Part_list...*]

Create Element Field *Field_name* Of Type *Option* And Dimension *Dimension* [On *Part_list...*]

Create Face Field *Field_name* Of Type *Option* And Dimension *Dimension* [On *Part_list...*]

Create Nodal Field *Field_name* Of Type *Option* And Dimension *Dimension* [On *Part_list...*]

Create Quadrature Field *Field_name* Of Type *Option* And Order *Intg_order*

Evaluate Flux Of *Function_name* At *x* [*y z t*]

Evaluate Function *Function_name* At *x* [*y z t*]

Evaluate Gradient Of *Function_name* At *x* [*y z t*]

Evaluate Postprocessor *Postprocessor_name*

Evaluate Time Derivative Of *Function_name* At *x* [*y z t*]

Execute Postprocessor Group *Postprocessor_name*

Find Maximum Residuals

Import Field *Mesh_field_name* As *Option1* Field *Field_name* [Of Type *Option2*]

Indicatemarkadapt Using *Indicator Marker*

Initial Uniform Refinement For *Num* Iterations

Initialize With Continuity Solve

Integer Data For Subroutine *SubName* {=|are|is} *Values...*

Interpolate Function *Option1* Of *Function_name* Into *Option2* Field *Field_name* [On *Option3 Part_list...*]

Markadapt Using *Marker*

Maximum Temperature Allowed From Temperature Extraction {=|are|is} *Value*

Minimum Temperature Allowed From Temperature Extraction {=|are|is} *Value*

Nonlinear Residual Plotfile Path {=|are|is} *Path*

Output Indicator Global Value Of *Indicator_name*

Output Number Of *Option*

Output Number Of Timesteps

Output Postprocessor History Of *Pp_name*

Output Preferred Integration Order Of *Function_name*

Print Values Of *Option* Field *Field_name*

Process Initial Condition

Real Data For Subroutine *SubName* {=|are|is} *Values...*

Region Participates In Time Step Selection When Inactive {=|are|is} *TrueFalse*

Set Function Parameter *Param_name* On *Function_name* To *Value*

Set Length Unit Conversion Factor {=|are|is} *Scale*

Set Mass Unit Conversion Factor {=|are|is} *Scale*

Set Temperature Units {=|are|is} *TempScale*

Set Time Unit Conversion Factor {=|are|is} *Scale*

Set Unit System To *UnitSystem*

Use Mpmd Radiation

Use Reduced Diffusion Stencil

Use Simple Radiation Source With Tref {=|are|is} *Tref* And Emissivity {=|are|is} *Emissivity*

Use Solution Steering With Interval {=|are|is} *Interval*

Use Finite Element Model *ModelName* [Model Coordinates Are *Nodal_variable_name*]

Begin Averaging *OptionsName*

End

Begin Composite Boundary Condition On Surface *Surfacename*

End

Begin Composite Interface Boundary Condition On Surface *Surfacename*

End

Begin Fixed Boundary Condition On Surface *Surfacename*

End

Begin Heartbeat *Label*

End

Begin Heat Flux Boundary Condition On Surface *Surfacename*

End

Begin History Output *Label*

End

Begin Inflow Boundary Condition On Surface *Surfacename*

End

Begin Initial Condition Block *BlockName*

End

Begin Mass Flux Boundary Condition On Surface *Surfacename*

End

Begin Nonconformal Boundary Condition On Surface *Surfacename*

End

Begin Open Boundary Condition On Surface *Surfacename*

```
    End

    Begin Periodic Name
    End

    Begin Periodic Boundary Condition On Surface Surfacename
    End

    Begin Post Process On Surface Surfacename
    End

    Begin Postprocessor Group Group_name
    End

    Begin Restart Data Label
    End

    Begin Results Output Label
    End

    Begin Solid Object Objectname
    End

    Begin Solution Options OptionsName
    End

    Begin Symmetry Boundary Condition On Surface Surfacename
    End

    Begin Virtual Thermocouple Model On Block BlockName
    End

    Begin Wall Boundary Condition On Surface Surfacename
    End

    Begin Wall Mass Inject Boundary Condition On Surface Surfacename
    End

End
```

Summary Contains the commands needed to execute an analysis on this region.

11.1.1. Allow Inverted Elements

Scope: Fuego Region

Summary Don't abort on inverted elements

Description Normal behavior will abort on an inverted element. With this beta option on the element volume will be clipped to epsilon with a warning.

11.1.2. Calculate Mesh Quality Metrics

Scope: Fuego Region

Summary Calculates CVFEM mesh quality metrics

Description Standard finite element quality metrics may not adequately capture mesh quality for CVFEM. Using this will calculate a mesh quality metric that is based on the distance between the dual volume centroid and the node location, normalized to be nominally between 0 and 100. Lower is better here, so a perfect mesh will have a quality metric of 0. Note that boundary elements have a half control volume which makes this calculation less relevant there, so the quality metric is set to 0 on all boundary nodes.

11.1.3. Compute Difference

Scope: Fuego Region

`Compute Difference` *`NormTypes`* `Of` *`f1 f2`* `[ Store In` *`field_name`* `]`

Parameter	Value	Default
NormTypes	`{h1|h1 restriction|l1|l2|linfinity |nodal l1|nodal l2|nodal linfinity| relative h1|relative h1 restriction |relative l1|relative l2|relative linfinity|relative nodal l1|relative nodal l2|relative nodal linfinity}`	-
f1	`string`	-
f2	`string`	-

Summary Computes the error between the two functions using the specified norm.

In the case of relative norms it does norm(f1-f2) / norm(f1) .

11.1.4. Compute Norm

Scope: Fuego Region

`Compute Norm` *`NormTypes`* `Of` *`Function_name`* `[ Store In` *`field_name`* `]`

Parameter	Value	Default
NormTypes	{h1|h1 restriction|l1|l2|linfinity |nodal l1|nodal l2|nodal linfinity| relative h1|relative h1 restriction |relative l1|relative l2|relative linfinity|relative nodal l1|relative nodal l2|relative nodal linfinity}	-
Function_name	string	-

Summary Computes the norm of the function using the specified norm.

11.1.5. Create Edge Field

Scope: Fuego Region

`Create Edge Field` *`Field_name`* `Of Type` *`Option`* `And Dimension` *`Dimension`* `[ On` *`Part_list...`* `]`

Parameter	Value	Default
Field_name	string	-
Option	{asym_tensor_03|complex| full_tensor_22|full_tensor_36|integer |long_integer|matrix_22|matrix_33 |real|sym_tensor_21|sym_tensor_31 |sym_tensor_33|unsigned_integer |unsigned_integer_64|vector_2d| vector_3d}	-
Dimension	integer	-

Summary Creates an edge Field named field_name on the region.

11.1.6. Create Element Field

Scope: Fuego Region

`Create Element Field` *`Field_name`* `Of Type` *`Option`* `And Dimension` *`Dimension`* `[ On` *`Part_list...`* `]`

Parameter	Value	Default
Field_name	`string`	-
Option	`{asym_tensor_03 | complex | full_tensor_22 | full_tensor_36 | integer | long_integer | matrix_22 | matrix_33 | real | sym_tensor_21 | sym_tensor_31 | sym_tensor_33 | unsigned_integer | unsigned_integer_64 | vector_2d | vector_3d}`	-
Dimension	`integer`	-

Summary Creates an Element Field name field_name on the region.

11.1.7. Create Face Field

Scope: Fuego Region

`Create Face Field` *`Field_name`* `Of Type` *`Option`* `And Dimension` *`Dimension`* `[ On` *`Part_list...`* `]`

Parameter	Value	Default
Field_name	`string`	-
Option	`{asym_tensor_03 | complex | full_tensor_22 | full_tensor_36 | integer | long_integer | matrix_22 | matrix_33 | real | sym_tensor_21 | sym_tensor_31 | sym_tensor_33 | unsigned_integer | unsigned_integer_64 | vector_2d | vector_3d}`	-
Dimension	`integer`	-

Summary Creates a face Field named field_name on the region.

11.1.8. Create Nodal Field

Scope: Fuego Region

`Create Nodal Field` *`Field_name`* `Of Type` *`Option`* `And Dimension` *`Dimension`* `[ On` *`Part_list...`* `]`

Parameter	Value	Default
Field_name	`string`	-
Option	`{asym_tensor_03 | complex | full_tensor_22 | full_tensor_36 | integer | long_integer | matrix_22 | matrix_33 | real | sym_tensor_21 | sym_tensor_31 | sym_tensor_33 | unsigned_integer | unsigned_integer_64 | vector_2d | vector_3d}`	-
Dimension	`integer`	-

Summary Creates a Nodal Field name field_name on the region.

11.1.9. Create Quadrature Field

Scope: Fuego Region

`Create Quadrature Field` *`Field_name`* `Of Type` *`Option`* `And Order` *`Intg_order`*

Parameter	Value	Default
Field_name	`string`	-
Option	`{asym_tensor_03 | complex | full_tensor_22 | full_tensor_36 | integer | long_integer | matrix_22 | matrix_33 | real | sym_tensor_21 | sym_tensor_31 | sym_tensor_33 | unsigned_integer | unsigned_integer_64 | vector_2d | vector_3d}`	-
Intg_order	`integer`	-

Summary Creates an Element Field named field_name sized correctly to receive a quadrature transfer of the order specified.

11.1.10. Evaluate Flux Of

Scope: Fuego Region

`Evaluate Flux Of` *`Function_name`* `At` *`x`* `[` *`y z t`* `]`

Parameter	Value	Default
Function_name	`string`	-
x	`real`	-

Summary Evaluates the gradient of the function at the given point.

11.1.11. Evaluate Function

Scope: Fuego Region

`Evaluate Function` *`Function_name`* `At` *`x`* `[` *`y`* *`z`* *`t`* `]`

Parameter	Value	Default
Function_name	string	-
x	real	-

Summary Evaluates the function at the given point.

11.1.12. Evaluate Gradient Of

Scope: Fuego Region

`Evaluate Gradient Of` *`Function_name`* `At` *`x`* `[` *`y`* *`z`* *`t`* `]`

Parameter	Value	Default
Function_name	string	-
x	real	-

Summary Evaluates the gradient of the function at the given point.

11.1.13. Evaluate Postprocessor

Scope: Fuego Region

`Evaluate Postprocessor` *`Postprocessor_name`*

Parameter	Value	Default
Postprocessor_name	string	-

Summary Evaluates the specified processor at the current time.

11.1.14. Evaluate Time Derivative Of

Scope: Fuego Region

`Evaluate Time Derivative Of` *`Function_name`* `At` *`x`* `[` *`y`* *`z`* *`t`* `]`

Parameter	Value	Default
Function_name	string	-
x	real	-

Summary Evaluates the time derivative of the function at the given point.

11.1.15. Execute Postprocessor Group

Scope: Fuego Region

`Execute Postprocessor Group` *`Postprocessor_name`*

Parameter	Value	Default
Postprocessor_name	`string`	-

Summary Executes all of the postprocessors in the specified group.

11.1.16. Find Maximum Residuals

Scope: Fuego Region

Summary Locate maximum nonlinear equation residual values on the mesh.

Description This option will provide the x, y and z location of the maximum nonlinear residual for all equations.

Note that this option may NOT BE USED WITH Prometheus

11.1.17. Import Field

Scope: Fuego Region

`Import Field` *`Mesh_field_name`* `As` *`Option1`* `Field` *`Field_name`* `[ Of Type` *`Option2`* `]`

Parameter	Value	Default
Mesh_field_name	`string`	-
Option1	`{edge|element|face|nodal}`	-
Field_name	`string`	-

Summary Imports a field from the current mesh and copies it to a SIERRA field. The field must be nodal or element. The optional parameter is the time at which to read the field.

11.1.18. Indicatemarkadapt Using

Scope: Fuego Region

`Indicatemarkadapt Using` *`Indicator Marker`*

Parameter	Value	Default
Indicator	`string`	-
Marker	`string`	-

Summary Postprocessor that simulates the solution control IndicateMarkAdapt line command. When this Postprocessor executes the region will be immediately adapted once. To get more complicated execution (for instance more levels of refinement) you must use Solution Control.

11.1.19. Initial Uniform Refinement For

Scope: Fuego Region

`Initial Uniform Refinement For` *`Num`* `Iterations`

Parameter	Value	Default
Num	`integer`	-

Summary Produces initial uniform refinement for num iterations. This also precedes initialization of the Region, including setting initial conditions.

11.1.20. Initialize With Continuity Solve

Scope: Fuego Region

Summary Apply a continuity solve to the initial conditions on the first time step

Description On the very first time step a continuity solve will be applied to the initial conditions before solving momentum. The solution will proceed in the standard order after that. This option allows initial conditions which may not strictly respect the requirements of incompressibility to be corrected by an initial projection step.

11.1.21. Integer Data For Subroutine

Scope: Fuego Region

`Integer Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`integer...`	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

11.1.22. Interpolate Function

Scope: Fuego Region

`Interpolate Function` *`Option1`* `Of` *`Function_name`* `Into` *`Option2`* `Field` *`Field_name`* `[ On` *`Option3 Part_list...`* `]`

Parameter	Value	Default
Option1	{flux | gradient | time derivative | value}	-
Function_name	string	-
Option2	{edge | element | face | nodal | quadrature}	-
Field_name	string	-

Summary Evaluates the specified function at the nodal positions and stores the result in field_name. Optionally, a list of surfaces or volumes can be listed that restrict the interpolation.

11.1.23. Markadapt Using

Scope: Fuego Region

`Markadapt Using` *`Marker`*

Parameter	Value	Default
Marker	string	-

Summary Postprocessor that simulates the solution control MarkAdapt line command. When this Postprocessor executes the region will be immediately adapted once. To get more complicated execution (for instance more levels of refinement) you must use Solution Control.

11.1.24. Maximum Temperature Allowed From Temperature Extraction

Scope: Fuego Region

`Maximum Temperature Allowed From Temperature Extraction {= | are | is}` *`Value`*

Parameter	Value	Default
Value	real	2600.0 K

Summary Enforce a particular maximum temperature that might occur in temperature extraction from enthalpy and composition.

Description This option specifies the maximum temperature to be allowed to be extracted from enthalpy, given the mixture composition. If a temperature is extracted that is greater than this value, a warning will be printed and the temperature will be reset to the previous value.

11.1.25. Minimum Temperature Allowed From Temperature Extraction

Scope: Fuego Region

`Minimum Temperature Allowed From Temperature Extraction {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`250.0 K`

Summary Enforce a particular minimum temperature that might occur in temperature extraction from enthalpy and composition.

Description This option specifies the minimum temperature to be allowed to be extracted from enthalpy, given the mixture composition. If a temperature is extracted that is less than this value, a warning will be printed and the temperature will be reset to the previous value.

11.1.26. Nonlinear Residual Plotfile Path

Scope: Fuego Region

`Nonlinear Residual Plotfile Path {=|are|is}` *`Path`*

Parameter	Value	Default
Path	`string`	-

Summary Specify the path to write the nonlinear residual plot files to for this region. Note that all plotfiles will be tagged by the region name as well as the equation name.

11.1.27. Output Indicator Global Value Of

Scope: Fuego Region

`Output Indicator Global Value Of` *`Indicator_name`*

Parameter	Value	Default
Indicator_name	`string`	-

Summary Outputs the global value of an indicator. This is usually the summation of each element's error contribution.

11.1.28. Output Number Of

Scope: Fuego Region

`Output Number Of` *`Option`*

Parameter	Value	Default
Option	`{edges | elements | faces | nodes}`	-

Summary A postprocessor that just outputs the current number of active nodes or elements.

11.1.29. Output Number Of Timesteps

Scope: Fuego Region

Summary A postprocessor that just outputs the current number of time steps.

11.1.30. Output Postprocessor History Of

Scope: Fuego Region

`Output Postprocessor History Of` *`Pp_name`*

Parameter	Value	Default
Pp_name	`string`	-

Summary Outputs the cumulative PP history values.

11.1.31. Output Preferred Integration Order Of

Scope: Fuego Region

`Output Preferred Integration Order Of` *`Function_name`*

Parameter	Value	Default
Function_name	`string`	-

Summary Outputs the preferred integration order of the function.

11.1.32. Print Values Of

Scope: Fuego Region

`Print Values Of` *`Option`* `Field` *`Field_name`*

Parameter	Value	Default
Option	`{edge|element|face|nodal}`	-
Field_name	`string`	-

Summary Prints the values of a field to the screen.

11.1.33. Process Initial Condition

Scope: Fuego Region

Summary This causes the first time step to be at time 0 when importing a solution... allowing post-processing and transfers of the initial condition.

It should probably only be used when running Encore as a standalone application, as it might throw other applications off.

11.1.34. Real Data For Subroutine

Scope: Fuego Region

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

11.1.35. Region Participates In Time Step Selection When Inactive

Scope: Fuego Region

`Region Participates In Time Step Selection When Inactive {=|are|is}` *`TrueFalse`*

Parameter	Value	Default
TrueFalse	`{false|true}`	-

Summary When region is inactive, it may or may not be able to modify the time step selection

Description Fuego can be run with inactivating the region solve. However, the question of how to negotiate the time step remains. If this line command is true, then the last Fuego time step will be provided. However, if this line command is false, Fuego will not participate in time step selection criteria.

11.1.36. Set Function Parameter

Scope: Fuego Region

`Set Function Parameter` *`Param_name`* `On` *`Function_name`* `To` *`Value`*

Parameter	Value	Default
Param_name	`string`	-
Function_name	`string`	-
Value	`real`	-

Summary Sets the value of "param" on function "function_name" to the specified value.

11.1.37. Set Length Unit Conversion Factor

Scope: Fuego Region

`Set Length Unit Conversion Factor {=|are|is}` *`Scale`*

Parameter	Value	Default
Scale	`real`	-

Summary Specify the conversion factor to convert length units to centimeters

Description Fuego allows arbitrary length units to be specified. The user must provide the conversion factor to convert to centimeters. Users should also ensure complete consistency in their input file and mesh - all units should be in the user-specified system.

Because the default unit system is CGS, use the following convention. If [X] cm = 1 [new length unit], enter "X" for this command. For example, enter 100 if your problem is in meters.

11.1.38. Set Mass Unit Conversion Factor

Scope: Fuego Region

`Set Mass Unit Conversion Factor {=|are|is}` *`Scale`*

Parameter	Value	Default
Scale	`real`	-

Summary Specify the conversion factor to convert mass units to grams

Description Fuego allows arbitrary mass units to be specified. The user must provide the conversion factor to convert to grams. Users should also ensure complete consistency in their input file and mesh - all units should be in the user-specified system.

Because the default unit system is CGS, use the following convention. If [X] g = 1 [new mass unit], enter "X" for this command. For example, enter 1000 if your problem is in kilograms.

11.1.39. Set Temperature Units

Scope: Fuego Region

`Set Temperature Units {=|are|is}` *`TempScale`*

Parameter	Value	Default
TempScale	`{celsius|fahrenheit|kelvin}`	-

Summary Specify the scale for temperature units

Description Fuego allows the user to select from several temperature scales. Users should also ensure complete consistency in their input file and mesh - all units should be in the user-specified system.

11.1.40. Set Time Unit Conversion Factor

Scope: Fuego Region

`Set Time Unit Conversion Factor {=|are|is}` *`Scale`*

Parameter	Value	Default
Scale	`real`	-

Summary Specify the conversion factor to convert time units to seconds

Description Fuego allows arbitrary time units to be specified. The user must provide the conversion factor to convert to seconds. Users should also ensure complete consistency in their input file and mesh - all units should be in the user-specified system.

Because the default unit system is CGS, use the following convention. If [X] s = 1 [new time unit], enter "X" for this command. For example, enter 60 if your problem is in minutes.

11.1.41. Set Unit System To

Scope: Fuego Region

`Set Unit System To` *`UnitSystem`*

Parameter	Value	Default
UnitSystem	`{cgs|mks}`	-

Summary Select a common length-mass-time unit system.

Description Shortcut to select meter-kilogram-second (MKS) or centimeter-gram-second unit (CGS) system. This should correspond to the units your mesh and BCs are in. Using Cantera requires a unit conversion, and CGS units are assumed if you do not specify otherwise.

Explicitly specifying units, either with the "SET UNIT SYSTEM" command or manually with the "SET X UNIT CONVERSION FACTOR" commands also lets Fuego automatically select universal constants like the Stefan-Boltzmann constant, speed of light, and Planck constant (for PMR).

11.1.42. Use Mpmd Radiation

Scope: Fuego Region

Summary Use MPMD radiation coupling - requiring other code to communicate with for PMR through MPI

Description This solution option indicates that Fuego will be run in MPMD mode, communicating with another executable through MPI for PMR (participating media radiation) coupling. Currently, Nalu and Scefire are both capable of coupling to Fuego for PMR calculations, and several examples exist in the regression test library. Radiation properties (including boundary radiation properties emissivity and transmissivity) are communicated from Fuego to the PMR code through MPMD. Quadrature rules, radiation mesh, and radiation solver settings are specified in the input deck for that executable. Fuego and the MPMD coupled PMR code are executed simultaneously as:

mpirun -n nF fuego -i fuego.i : -n nN nalu -i nalu.i (Nalu)

mpirun -n nF fuego -i fuego.i : -n nS scefire scefire.xml (Scefire)

where nF, nN, nS are the integer numbers of processors to be used for Fuego, Nalu, or Scefire, respectively. fuego.i, nalu.i, and scefire.xml are the Fuego, Nalu, and Scefire input decks, respectively.

11.1.43. Use Reduced Diffusion Stencil

Scope: Fuego Region

Summary Use canonical seven point diffusion stencil

Description The standard diffusion operator supported is represented by the twenty seven node canonical hexahedron stencil. This option allows for all diffusion operators to reduce to the canonical seven point stencil (again hexahedron) by shifting integration point evaluation from the sub surface control volume face centroid to the edge centroid.

This option may be beneficial to use on a highly skewed mesh.

11.1.44. Use Simple Radiation Source With Tref

Scope: Fuego Region

`Use Simple Radiation Source With Tref {=|are|is}` *`Tref`* `And Emissivity {=|are|is}` *`Emissivity`*

Parameter	Value	Default
Tref	`real`	-
Emissivity	`real`	-

Summary Use a difference between the T^4 and T^4_{ref} to approximate radiation to the far field.

Description Calculates the scalar_flux term as $4 * \epsilon * \sigma * T^4_{\text{ref}}$ rather than using a PMR solver.

11.1.45. Use Solution Steering With Interval

Scope: Fuego Region

`Use Solution Steering With Interval {=|are|is}` *`Interval`*

Parameter	Value	Default
Interval	`integer`	1

Summary Change parameters interactively

Description Create a solution steering file with parameters that the user can interactively modify during the course of a solution. The file is automatically written during the simulation with the options that are allowed to change. It is read and written again based on the interval set initially in the input file and potentially modified in the steering file. The interval defines how often the file gets read (number of time steps).

11.1.46. Use Finite Element Model

Scope: Fuego Region

`Use Finite Element Model` *`ModelName`* `[ Model Coordinates Are` *`Nodal_variable_name`* `]`

Parameter	Value	Default
ModelName	`string`	-

Summary Associates a predefined finite element model with this region.

11.2. AVERAGING

Scope: Fuego Region

`Begin Averaging` *`OptionsName`*

`Butterworth A Coefficients {=|are|is}` *`Coeffs...`*

`Butterworth B Coefficients {=|are|is}` *`Coeffs...`*

`Butterworth Filtered Field` *`RegisteredField`* `As` *`AverageField`* `[ On Output Block` *`BlockName`* `]`

`Butterworth Time Step {=|are|is}` *`TimeStep`*

`Favre Average Field` *`RegisteredField`* `As` *`AverageField`* `[ On Output Block` *`BlockName`* `]`

`Reset Filter On Restart`

`Reynolds Average Field` *`RegisteredField`* `As` *`AverageField`* `[ On Output Block` *`BlockName`* `]`

`Starting Time {=|are|is}` *`StartingTime`*

`Time Interval Length {=|are|is}` *`IntervalLength`*

`End`

Summary Specify information regarding the Reynolds and Favre averaging. The Reynolds average is the time average of a value:

$$\bar{\phi} = \frac{1}{T} \int \phi(t)\, dt$$

where $\phi(t)$ is the quantity of interest at time t and the integration is over $(0, T)$, the time interval which is averaged.

The Favre average is the ratio of two Reynolds averages:

$$ilde\phi = \frac{\bar{\rho\phi}}{\bar{\rho}}$$

Where ϕ is the quantity of interest and mass is the mass.

11.2.1. Butterworth A Coefficients

Scope: Averaging

`Butterworth A Coefficients {=|are|is}` *`Coeffs...`*

Parameter	Value	Default
Coeffs	`real...`	-

Summary A coefficients for the Butterworth filter

Description The list of A coefficients for the butterworth filter. See the BUTTERWORTH FILTERED FIELD command documentation for details.

11.2.2. Butterworth B Coefficients

Scope: Averaging

`Butterworth B Coefficients {=|are|is}` *`Coeffs...`*

Parameter	Value	Default
Coeffs	`real...`	-

Summary B coefficients for the Butterworth filter

Description The list of B coefficients for the butterworth filter. See the BUTTERWORTH FILTERED FIELD command documentation for details.

11.2.3. Butterworth Filtered Field

Scope: Averaging

`Butterworth Filtered Field` *`RegisteredField`* `As` *`AverageField`* `[ On Output Block` *`BlockName`* `]`

Parameter	Value	Default
RegisteredField	`string`	-
AverageField	`string`	-

Summary Generates Butterworth filter of field.

Description The Butterworth filter operations assume that the data is given at a constant time step. The actual time step may vary during the analysis depending on the time stepping mode. The `BUTTERWORTH TIME STEP` command is used to linearly interpolate the data being produced at a non-constant time step down to some specified constant time step. The interpolation time step must be larger than zero and ideally should be specified to be smaller than the smallest time step with which the computations will iterate.

One way to obtain the filtering coefficients is with MATLAB. The following is an example of defining a third order Butterworth filter with a pass frequency of 100Hz at data interpolated to a time step of $1.0e-5$ seconds. The filter is then used to filter acceleration histories of the nodes to 100Hz. The MATLAB code below will give the desired filtering coefficients. The full 16-digit precision of the coefficients returned by MATLAB should be used. If truncated precision numbers are used, the filters can potentially be unstable.

```
clear;
format long e;
passFrequency = 100;
interp_ts = 1.0e-5;
butterCoeff = 2.0*interp_ts*passFrequency;
[bcoeff,acoeff] = butter(3,butterCoeff);
acoeff
bcoeff
```

The computed filtering coefficients can be used in the averaging block as show below. If the analysis time step always remains above $1.0e-5$ the filter will be valid. If the analysis time step drops below $1.0e-5$ there could be aliasing issues, and a smaller interpolation time step should be specified.

```
BUTTERWORTH A COEFFICIENTS = 1.000000000000000e+00 -2.987433650055
          2.974946132665442e+00 -9.875122361107358e-01
BUTTERWORTH B COEFFICIENTS = 3.081237301416628e-08  9.243711904249
          9.243711904249885e-08  3.081237301416628e-08
BUTTERWORTH TIME STEP = 1e-5
```

Start time and time interval length commands are not used in the Butterworth filter.

11.2.4. Butterworth Time Step

Scope: Averaging

`Butterworth Time Step {=|are|is}` *`TimeStep`*

Parameter	Value	Default
TimeStep	`real`	`0.0`

Summary Time step for butterworth filter.

Description The time step for the butterworth filter. See the BUTTERWORTH FILTERED FIELD command documentation for details.

11.2.5. Favre Average Field

Scope: Averaging

`Favre Average Field` *`RegisteredField`* `As` *`AverageField`* `[ On Output Block` *`BlockName`* `]`

Parameter	Value	Default
RegisteredField	`string`	-
AverageField	`string`	-

Summary Generates Favre average of field.

Description The Favre average is the ratio of two Reynolds averages:

$$\tilde{\phi} = \frac{\bar{\rho\phi}}{\bar{\rho}}$$

Where ϕ is the quantity of interest and ρ is the mass.

The field to be averaged must exist in the model being solved. The averaged field will be created and output on the specified block. It the block is not specified, the average will be defined and output on any nodes that the base field exists.

Since the Reynolds average of the mass and the Reynolds average of the field to be Favre averaged must be computed in order to calculate the ratio, these two extra fields will be created and written to the results file. The Reynolds averaged mass is available as "density_Avg" and the Reynolds average of the mass weighted Favre field, $\bar{\rho\phi}$, will be available as the specified output field name in this line command appended by the string "_Wtd".

11.2.6. Reset Filter On Restart

Scope: Averaging

Summary Reset Favre/Reynolds averages specified in block

Description This option, on restart, will remove history from averages, restarting the average if the current time > start time of the average. This option is not compatible with Butterworth filtering.

11.2.7. Reynolds Average Field

Scope: Averaging

`Reynolds Average Field` *`RegisteredField`* `As` *`AverageField`* `[ On Output Block` *`BlockName`* `]`

Parameter	Value	Default
RegisteredField	`string`	-
AverageField	`string`	-

Summary Generates Reynolds average of field.

Description The Reynolds average is the time average of a value:

$$\bar{\phi} = \frac{1}{T} \int \phi(t)\, dt$$

where $\phi(t)$ is the quantity of interest at time t and the integral is evaluated over $(0, T)$, the time interval which is averaged.

The field to be averaged must exist in the model being solved. The averaged field will be created and output on the specified block. It the block is not specified, the average will be defined and output on any nodes that the base field exists.

Since the Reynolds average of the mass and the Reynolds average of the field to be Favre averaged must be computed in order to calculate the ratio, these two extra fields will be created and written to the results file. The Reynolds averaged mass is available as "density_Avg" and the Reynolds average of the mass weighted Favre field, $\bar{\rho\phi}$, will be available as the specified output field name in the line command appended by the string "_Wtd".

11.2.8. Starting Time

Scope: Averaging

`Starting Time {=|are|is}` *`StartingTime`*

Parameter	Value	Default
StartingTime	`real`	`0.0`

Summary Time for which the averaging starts.

Description If the starting time is specified then the averaging will not start until the starting time is obtained. All data before the starting time will be ignored and the average will be zero. Once the starting time is reached the averaging will proceed as described under the time interval length parameter with intervals over $(T_0 + nT, T_0(n+1)T)$ where T_0 is the starting time.

11.2.9. Time Interval Length

Scope: Averaging

`Time Interval Length {=|are|is}` *`IntervalLength`*

Parameter	Value	Default
IntervalLength	`real`	`REAL_MAX`

Summary — Time interval length over which average is computed.

Description — If the time interval length is specified as T, The Reynolds or Favre averages specified will be determined over intervals of length T. The intervals will be over $(nT, (n+1)T)$ for integers n. At the end of one interval, the running average that is being computed will be zeroed out and the averaging starting all over again. This means that soon after an interval change the output field will contain just the average from the beginning of that time interval to the current time. The result is that at every interval boundary there is liable to be a jump or variation in the running average that will be smoothed out over time.

If the starting time parameter is specified then the averaging will not start until the starting time is obtained. All data before the starting time will be ignored and the average will be zero. Once the starting time is reached the averaging will proceed as described with intervals over $(T_0 + nT, T_0(n + 1rT)$ where T_0 is the starting time.

11.3. HEARTBEAT

Scope: Average Region

```
Begin Heartbeat Label
    Additional Steps {=|are|is} List_of_steps...
    Additional Times {=|are|is} List_of_times...
    Append {=|are|is} Option
    At Step n Option {=|are|is} m
    At Time Dt1 Option {=|are|is} Dt2
    Element [ VariableList... ]
    Exists Option1 Option2
    Face [ VariableList... ]
    Flush Interval {=|are|is} Option
    Format {=|are|is} StreamTypes
    Global [ Variables... ]
    Labels {=|are|is} Option
```

```
    Legend {=|are|is} Option

    Monitor Equals Option

    Nodal [ VariableList...  ]

    Node [ VariableList...  ]

    Nodeset [ VariableList...  ]

    Output On Signal {=|are|is} Signals

    Precision {=|are|is} Precision

    Start Time {=|are|is} Start_time

    Stream Name {=|are|is} OutputFilename

    Synchronize Output

    Termination Time {=|are|is} Final_time

    Timestamp Format

    Timestep Adjustment Interval {=|are|is} Nsteps

    Use Output Scheduler Timer_name

    Variable {=|are|is} Option Variable_list...

End
```

Summary Describes the location and type of the output stream used for outputting the heartbeat information for the enclosing region.

11.3.1. Additional Steps

Scope: Heartbeat

```
Additional Steps {=|are|is} List_of_steps...
```

Parameter	Value	Default
List_of_steps	integer...	-

Summary Additional simulation steps when output should occur.

11.3.2. Additional Times

Scope: Heartbeat

`Additional Times {=|are|is}` *`List_of_times...`*

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

11.3.3. Append

Scope: Heartbeat

`Append {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`{false|off|on|true}`	-

Summary Specifies whether the heartbeat file is appended if it exists. By default, the file is appended if restart is requested and not if restart is not requested. This option does not work for automatic restarts because a new heartbeat file is written with each auto restart.

11.3.4. At Step

Scope: Heartbeat

`At Step` *`n Option`* `{=|are|is}` *`m`*

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

11.3.5. At Time

Scope: Heartbeat

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

11.3.6. Element

Scope: Heartbeat

Summary Define the element variables that should be written to the heartbeat database. The syntax is: "element internal_name at element id as DBname" or "element internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database.

11.3.7. Exists

Scope: Heartbeat

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{abort|append|overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis.

11.3.8. Face

Scope: Heartbeat

Summary Define the face variables that should be written to the heartbeat database. The syntax is: "face internal_name at face id as DBname" or "face internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database.

11.3.9. Flush Interval

Scope: Heartbeat

`Flush Interval {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`integer`	`10`

Summary The minimum time interval (in seconds) at which the heartbeat output will be explicitly flushed to disk. The default is 10 seconds.

11.3.10. Format

Scope: Heartbeat

`Format {=|are|is}` *`StreamTypes`*

Parameter	Value	Default
StreamTypes	`{csv|original|spyhis}`	-

Summary The stream type/format to be used for the output results.The only three options at this time are 'Original' which is the old default Sierra heartbeat format; 'SpyHis' which mimics the CTH Spyhis history output format; and 'CSV'

11.3.11. Global

Scope: Heartbeat

Summary Define the global/reduction variables that should be written to the heartbeat database. The syntax is: "global internal_name as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database.

11.3.12. Labels

Scope: Heartbeat

`Labels {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`{off|on}`	`on`

Summary Specifies whether labels will be displayed or just the value of the variable. Labels will be shown if this line is not present.

11.3.13. Legend

Scope: Heartbeat

`Legend {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`{off|on}`	`on`

Summary Specifies whether a legend will be displayed prior to outputting any variables. The legend will not be shown unless this line is present. The legend shows the names of the variables that will be written to the heartbeat output stream. If the variable has multiple components, then the component count is shown after the variable e.g., velocity(3).

11.3.14. Monitor

Scope: Heartbeat

`Monitor` *`Equals Option`*

Parameter	Value	Default
Equals	`{=|the}`	-
Option	`{history|restart|results}`	-

Summary Specifies whether a line will be written to the heartbeat stream when either the results, history, and/or restart data are output.

11.3.15. Nodal

Scope: Heartbeat

Summary Define the nodal variables that should be written to the heartbeat database. The syntax is: "nodal internal_name at node id as DBname" or "nodal internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database.

11.3.16. Node

Scope: Heartbeat

Summary Define the nodal variables that should be written to the heartbeat database. The syntax is: "node internal_name at node id as DBname" or "node internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database.

11.3.17. Nodeset

Scope: Heartbeat

Summary Define the nodeset variables that should be written to the heartbeat database. The syntax is: "nodeset internal_name at node id as DBname" or "nodeset internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database. This option finds a single value for the internal_name specified without having to specify a nodeset id or name.

11.3.18. Output On Signal

Scope: Heartbeat

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

11.3.19. Precision

Scope: Heartbeat

`Precision {=|are|is}` *`Precision`*

Parameter	Value	Default
Precision	`integer`	`5`

Summary The precision to be used for the output of real variables (default=5).

11.3.20. Start Time

Scope: Heartbeat

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

11.3.21. Stream Name

Scope: Heartbeat

`Stream Name {=|are|is}` *`OutputFilename`*

Parameter	Value	Default
OutputFilename	`string`	-

Summary The filename of where the heartbeat data should be written. If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. In addition, there are several predefined streams that can be specified. The predefined streams are 'cout' or 'stdout' specifies standard output; 'cerr', 'stderr', 'clog', or 'log' specifies standard error; 'output' or 'outputPo' specifies Sierra's standard output which is redirected to the file specified by the '-o' option on the command line. If the file already exists, it is overwritten. If this line is omitted, then a filename will be created from the basename of the input file with a ".hrt" suffix appended.

11.3.22. Synchronize Output

Scope: Heartbeat

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

11.3.23. Termination Time

Scope: Heartbeat

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

11.3.24. Timestamp Format

Scope: Heartbeat

Summary The format to be used for the timestamp. See 'man strftime' for more information.

11.3.25. Timestep Adjustment Interval

Scope: Heartbeat

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	`integer`	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

11.3.26. Use Output Scheduler

Scope: Heartbeat

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	`string`	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

11.3.27. Variable

Scope: Heartbeat

`Variable {=|are|is}` *`Option Variable_list...`*

Parameter	Value	Default
Option	`{edge|element|face|global|nodal|node}`	-
Variable_list	`string...`	-

Summary Define the variables that should be written to the heartbeat output. The user can request that the values of certain variables be output on the heartbeat line. These variables are limited to region and framework control data currently. The syntax is:

```
variable = {entity_type} {internal_name} at
           {entity_type} {entity_id}     as {external_name}
variable = {entity_type} {internal_name} nearest location
           {x,y,z} as {external_name}
```

For global variables, use:

```
variable = global {internal_name} [as {external_name}]
```

Where:

```
entity_type = node, element, face, edge, global
internal_name = Sierra variable name
entity_id = id of the node, element, face, edge that you want
    the specified variable output at.
external_name = name of variable on the database.
```

The names 'timestep', and 'time' can be specified as variables also. They are the current timestep and simulation time. This line can appear multiple times.

11.4. HISTORY OUTPUT

Scope: Average Region

Begin History Output *Label*

Additional Steps {=|are|is} *List_of_steps...*

Additional Times {=|are|is} *List_of_times...*

At Step *n Option* {=|are|is} *m*

At Time *Dt1 Option* {=|are|is} *Dt2*

Database Name {=|are|is} *StreamName*

Database Type {=|are|is} *DatabaseTypes*

Element [*VariableList...*]

Exists *Option1 Option2*

Face [*VariableList...*]

Global [*Variables...*]

Nodal [*VariableList...*]

Node [*VariableList...*]

Nodeset [*VariableList...*]

Output On Signal {=|are|is} *Signals*

Overwrite *Option1 Option2*

Property *PropertyName* {=|are|is} *PropertyValue*

Start Time {=|are|is} *Start_time*

Synchronize Output

Termination Time {=|are|is} *Final_time*

Timestep Adjustment Interval {=|are|is} *Nsteps*

```
        Title
        Use Output Scheduler Timer_name
        Variable {=|are|is} Option Variable_list...
    End
```

Summary Describes the location and type of the output stream used for outputting history for the enclosing region.

11.4.1. Additional Steps

Scope: History Output

```
Additional Steps {=|are|is} List_of_steps...
```

Parameter	Value	Default
List_of_steps	`integer...`	-

Summary Additional simulation steps when output should occur.

11.4.2. Additional Times

Scope: History Output

```
Additional Times {=|are|is} List_of_times...
```

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

11.4.3. At Step

Scope: History Output

```
At Step n Option {=|are|is} m
```

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

11.4.4. At Time

Scope: History Output

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

11.4.5. Database Name

Scope: History Output

`Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The base name of the database containing the output history. If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. If this line is omitted, then a filename will be created from the basename of the input file with a ".h" suffix appended.

11.4.6. Database Type

Scope: History Output

`Database Type {=|are|is}` *`DatabaseTypes`*

Parameter	Value	Default
DatabaseTypes	`{catalyst|dof|dof_exodus|exodus|exodusii|generated|genesis|parallel_exodus}`	-

Summary The database type/format to be used for the output history.

11.4.7. Element

Scope: History Output

Summary Define the element variables that should be written to the history database. The syntax is: "element internal_name at element id as DBname" or "element internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

11.4.8. Exists

Scope: History Output

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{abort|add_suffix|append|overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis. "ADD_SUFFIX" will add a -s???? suffix where the ???? is replaced by a sequential number starting at 0002.

11.4.9. Face

Scope: History Output

Summary Define the face variables that should be written to the history database. The syntax is: "face internal_name at face id as DBname" or "face internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

11.4.10. Global

Scope: History Output

Summary Define the global/reduction variables that should be written to the history database. The syntax is: "global internal_name as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

11.4.11. Nodal

Scope: History Output

Summary Define the nodal variables that should be written to the history database. The syntax is: "nodal internal_name at node id as DBname" or "nodal internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

11.4.12. Node

Scope: History Output

Summary Define the nodal variables that should be written to the history database. The syntax is: "node internal_name at node id as DBname" or "node internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

11.4.13. Nodeset

Scope: History Output

Summary Define the nodeset variables that should be written to the history database. The syntax is: "nodeset internal_name at node id as DBname" or "nodeset internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

11.4.14. Output On Signal

Scope: History Output

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

11.4.15. Overwrite

Scope: History Output

`Overwrite` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{false|no|off|on|true|yes}`	-

Summary (DEPRECATED, Use EXISTS) Specify whether the database should be overwritten if it exists. The default behavior is to overwrite unless this command is specified in the output block and either off, false, or no is specified.

11.4.16. Property

Scope: History Output

`Property` *`PropertyName`* `{=|are|is}` *`PropertyValue`*

Parameter	Value	Default
PropertyName	`string`	-
PropertyValue	`string`	-

Summary Define a database property named "PropertyName" with the value "PropertyValue". If PropertyValue consists of all digits, it will define an integer property. If PropertyValue is "true" or "yes" or "false" or "no", it will define a logical property; otherwise it will define a string property. Suppported properties are typically database dependent; Some history-related properties are: VARIABLE_NAME_CASE = upper|lower MAX_NAME_LENGTH = value (32)

11.4.17. Start Time

Scope: History Output

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

11.4.18. Synchronize Output

Scope: History Output

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

11.4.19. Termination Time

Scope: History Output

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

11.4.20. Timestep Adjustment Interval

Scope: History Output

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	`integer`	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

11.4.21. Title

Scope: History Output

Summary Specify the title to be used for this specific output block.

11.4.22. Use Output Scheduler

Scope: History Output

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	`string`	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

11.4.23. Variable

Scope: History Output

`Variable {=|are|is}` *`Option Variable_list...`*

Parameter	Value	Default
Option	`{edge|element|face|global|nodal|node}`	-
Variable_list	`string...`	-

Summary Define the variables that should be written to the history database. The syntax is: "variable = entity internal_name at entity id as DBname" or "variable = entity internal_name nearest location X, Y, Z as DBname" or "variable = entity internal_name at location X, Y, Z as DBname".

Where entity is 'node', 'element', 'face', or 'edge'; internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

11.5. RESTART DATA

Scope: Average Region

Begin Restart Data *Label*

Additional Steps {=|are|is} *List_of_steps...*

Additional Times {=|are|is} *List_of_times...*

At Step *n Option* {=|are|is} *m*

At Time *Dt1 Option* {=|are|is} *Dt2*

At Wall Time *Dt1 Option* {=|are|is} *Dt2*

Component Separator Character *Option Separator*

Cycle Count {=|are|is} *Count*

Database Name {=|are|is} *StreamName*

Database Type {=|are|is} *DatabaseTypes*

Debug Dump

Decomposition Method {=|are|is} *Method*

Exists *Option1 Option2*

File Cycle Count {=|are|is} *Count*

Input Database Name {=|are|is} *StreamName*

Optional

Output Database Name {=|are|is} *StreamName*

Output On Signal {=|are|is} *Signals*

Overlay Count {=|are|is} *Count*

Overwrite *Option1 Option2*

Property *PropertyName* {=|are|is} *PropertyValue*

Restart {=|are|is} *RestartOption*

Restart Time {=|are|is} *Time*

Start Time {=|are|is} *Start_time*

Synchronize Output

Termination Time {=|are|is} *Final_time*

Timestep Adjustment Interval {=|are|is} *Nsteps*

Use Output Scheduler *Timer_name*

End

Summary Describes the data required to output and input restart data for the enclosing region.

11.5.1. Additional Steps

Scope: Restart Data

`Additional Steps {=|are|is}` *`List_of_steps...`*

Parameter	Value	Default
List_of_steps	`integer...`	-

Summary Additional simulation steps when output should occur.

11.5.2. Additional Times

Scope: Restart Data

`Additional Times {=|are|is}` *`List_of_times...`*

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

11.5.3. At Step

Scope: Restart Data

`At Step` *`n Option`* `{=|are|is}` *`m`*

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

11.5.4. At Time

Scope: Restart Data

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

11.5.5. At Wall Time

Scope: Restart Data

`At Wall Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`string`	-
Option	`{increment|interval}`	-
Dt2	`string`	-

Summary Write a restart file at a specific wall time since the start of the run. Time string format allows s, m, h, d for seconds, minutes, hours, days

11.5.6. Component Separator Character

Scope: Restart Data

`Component Separator Character` *`Option Separator`*

Parameter	Value	Default
Option	`{=|is}`	-
Separator	`string`	-

Summary The separator is the single character used to separate the output variable basename (e.g. "stress") from the suffices (e.g. "xx", "yy") when displaying the names of the individual variable components. For example, the default separator is "_", which results in names similar to "stress_xx", "stress_yy", ... "stress_zx". To eliminate the separator, specify an empty string ("") or NONE.

11.5.7. Cycle Count

Scope: Restart Data

`Cycle Count {=|are|is}` *`Count`*

Parameter	Value	Default
Count	`integer`	-

Summary Specify the number of restart steps which will be written to the restart database before previously written steps are overwritten. For example, if the cycle count is 5 and restart is written every 0.1 seconds, the restart system will write 0.1, 0.2, 0.3, 0.4, 0.5 to the database. It will then overwrite the first step with data from time 0.6, the second with time 0.7. At time 0.8, the database would contain data at times 0.6, 0.7, 0.8, 0.4, 0.5. Note that time will not necessarily be monotonically increasing on a database that specifies the cycle count.

11.5.8. Database Name

Scope: Restart Data

`Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The database containing the input and/or output restart data. If this analysis is being restarted, restart data will be read from this file. If the analysis is writing restart data, the data will be written to this file. It will be overwritten if it exists (after being read if applicable). If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. See also the 'Input Database' and 'Output Database' commands.

11.5.9. Database Type

Scope: Restart Data

`Database Type {=|are|is}` *`DatabaseTypes`*

Parameter	Value	Default
DatabaseTypes	`{catalyst|dof|dof_exodus|exodus|exodusii|generated|genesis|parallel_exodus}`	-

Summary The database type/format used for the restart file.

11.5.10. Debug Dump

Scope: Restart Data

Summary Specify whether the restart system will write the restart data immediately after reading the restart data if the run is restarting. The output data can be compared with the restart input data to determine whether they match.

11.5.11. Decomposition Method

Scope: Restart Data

`Decomposition Method {=|are|is}` *`Method`*

Parameter	Value	Default
Method	`{block|cyclic|external|geom_kway|hsfc|kway|kway_geom|linear|metis_sfc|random|rcb|rib}`	-

Summary The decomposition algorithm to be used to partition elements to each processor in a parallel run.

11.5.12. Exists

Scope: Restart Data

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{abort|add_suffix|append|overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis. "ADD_SUFFIX" will add a suffix to the file name and output to that file.

11.5.13. File Cycle Count

Scope: Restart Data

`File Cycle Count {=|are|is}` *`Count`*

Parameter	Value	Default
Count	`integer`	-

Summary Each restart dump will be written to a separate file suffixed with A,B, ... The count specifies how many separate files are used before the cycle repeats. For example, if "FILE CYCLE COUNT = 3" is specified, the restart dumps would be written to file-A.rs, file-B.rs, file-C.rs, file-A.rs, ... The maximum value for the cycle count is 26.

11.5.14. Input Database Name

Scope: Restart Data

`Input Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The database containing the input restart data. If this analysis is being restarted, restart data will be read from this file. See also the 'Database' and 'Output Database' commands.

11.5.15. Optional

Scope: Restart Data

Summary The database will be read if it exists, but it is not an error if there is no restart database to read for this region during a restarted analysis.

11.5.16. Output Database Name

Scope: Restart Data

`Output Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The database containing the output restart data. If the analysis is writing restart data, the data will be written to this file. It will be overwritten if it exists. See also the 'Database' and 'Input Database' commands.

11.5.17. Output On Signal

Scope: Restart Data

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

11.5.18. Overlay Count

Scope: Restart Data

`Overlay Count {=|are|is}` *`Count`*

Parameter	Value	Default
Count	`integer`	-

Summary Specify the number of restart outputs which will be overlayed on top of the last written step. For example, if restarts are being output every 0.1 seconds and the overlay count is specified as 2, then restart will write times 0.1 to step 1 of the database. It will then write 0.2 and 0.3 also to step 1. It will then increment the database step and write 0.4 to step 2; overlay 0.5 and 0.6 on step 2... At the end of the analysis, assuming it runs to completion, the database would have times 0.3, 0.6, 0.9, ... However, if there were a problem during the analysis, the last step on the database would contain an intermediate step.

11.5.19. Overwrite

Scope: Restart Data

`Overwrite` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{false|no|off|on|true|yes}`	-

Summary (DEPRECATED, Use EXISTS) Specify whether the restart database should be overwritten if it exists. The default behavior is to overwrite unless this command is specified in the restart block and either off, false, or no is specified.

11.5.20. Property

Scope: Restart Data

`Property` *`PropertyName`* `{=|are|is}` *`PropertyValue`*

Parameter	Value	Default
PropertyName	`string`	-
PropertyValue	`string`	-

Summary Define a database property named "PropertyName" with the value "PropertyValue". If PropertyValue consists of all digits, it will define an integer property. If PropertyValue is "true" or "yes" or "false" or "no", it will define a logical property; otherwise it will define a string property. If PropertyName consists of multiple strings, they will be concated together with "_" separating the individual words. Suppported properties are typically database dependent; Current properties are: COMPRESSION_LEVEL = [0..9] COMPRESSION_SHUFFLE = true|false|on|off FILE_TYPE = netcdf4 (forces use of netcdf-4 hdf5-based file) INTEGER_SIZE_DB = 4|8 INTEGER_SIZE_API = 4|8 LOGGING = true|false|on|off MAX_NAME_LENGTH = value

11.5.21. Restart

Scope: Restart Data

`Restart {=|are|is}` *`RestartOption`*

Parameter	Value	Default
RestartOption	`{auto|automatic}`	-

Summary Specify automatic restart file read.

Description **NOTE:** This command must be placed at the Sierra scope of the input file.

Specify that the analysis should be restarted from the last common time on all restart databases for each Region in the analysis. In addition to this line command, each Region in the analysis (strictly, only the region(s) that will be restarted) must have a restart block specifying the database to read the restart state data.

By default, use of this command will not cause output files (e.g., results, history, heartbeat, restart) to be overwritten. Instead output files will be written with the same basename and the suffix `-s000*`. Common visualization packages are written to handle this file organization gracefully in order for the user to view all results seamlessly.

11.5.22. Restart Time

Scope: Restart Data

`Restart Time {=|are|is}` *`Time`*

Parameter	Value	Default
Time	`real`	-

Summary Specify restart file read at a specified time.

Description **NOTE:** This command must be placed at the Sierra scope of the input file.

Specify the time that the analysis will be restarted. In addition to this line command, each Region in the analysis (strictly, only the region(s) that will be restarted) must have a restart block specifying the database to read the restart state data. The restart 'time' must be greater than zero and less than or equal to the termination time.

By default, use of this command will cause previous output files (e.g., results, history, heartbeat, restart) to be overwritten. If this command is chosen, the onus is placed on the user to ensure that previous output files are not overwritten.

11.5.23. Start Time

Scope: Restart Data

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

11.5.24. Synchronize Output

Scope: Restart Data

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where

one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

11.5.25. Termination Time

Scope: Restart Data

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

11.5.26. Timestep Adjustment Interval

Scope: Restart Data

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	`integer`	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

11.5.27. Use Output Scheduler

Scope: Restart Data

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	`string`	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

11.6. RESULTS OUTPUT

Scope: Average Region

```
Begin Results Output Label
    Additional Steps {=|are|is} List_of_steps...
    Additional Times {=|are|is} List_of_times...
    At Step n Option {=|are|is} m
    At Time Dt1 Option {=|are|is} Dt2
    Component Separator Character {=|are|is} Separator
    Database Name {=|are|is} StreamName
    Database Type {=|are|is} DatabaseType
    Edge [ VariableList... ]
    Edge Variables {=|are|is} [ VariableList... ]
    Element [ VariableList... ]
    Element Variables {=|are|is} [ VariableList... ]
    Enable Large Ids
    Exclude {=|are|is} [ ElementBlockList... ]
    Exists Option1 Option2
    Face [ VariableList... ]
    Face Variables {=|are|is} [ VariableList... ]
    Global [ Variables... ]
    Global Variables {=|are|is} [ Variables... ]
    Include {=|are|is} [ ElementBlockList... ]
    Nodal [ VariableList... ]
    Nodal Variables {=|are|is} [ VariableList... ]
    Node [ VariableList... ]
    Node Variables {=|are|is} [ VariableList... ]
    Nodeset [ VariableList... ]
    Nodeset Variables {=|are|is} [ VariableList... ]
    Output Mesh {=|are|is} OutputMesh
    Output On Signal {=|are|is} Signals
```

```
    Overwrite Option1 Option2

    Property PropertyName {=|are|is} PropertyValue

    Sideset [ VariableList...  ]

    Sideset Variables {=|are|is} [ VariableList...  ]

    Start Time {=|are|is} Start_time

    Surface [ VariableList...  ]

    Surface Variables {=|are|is} [ VariableList...  ]

    Synchronize Output

    Termination Time {=|are|is} Final_time

    Timeseries Name {=|are|is} filename

    Timestep Adjustment Interval {=|are|is} Nsteps

    Title

    Use Output Scheduler Timer_name

    Begin Catalyst Label

    End

End
```

Summary Describes the location and type of the output stream used for outputting results for the enclosing region.

11.6.1. Additional Steps

Scope: Results Output

```
Additional Steps {=|are|is} List_of_steps...
```

Parameter	Value	Default
List_of_steps	`integer...`	-

Summary Additional simulation steps when output should occur.

11.6.2. Additional Times

Scope: Results Output

`Additional Times {=|are|is}` *`List_of_times...`*

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

11.6.3. At Step

Scope: Results Output

`At Step` *`n Option`* `{=|are|is}` *`m`*

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

11.6.4. At Time

Scope: Results Output

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

11.6.5. Component Separator Character

Scope: Results Output

`Component Separator Character {=|are|is}` *`Separator`*

Parameter	Value	Default
Separator	`string`	-

Summary The separator is the single character used to separate the output variable basename (e.g. "stress") from the suffices (e.g. "xx", "yy") when displaying the names of the individual variable components. For example, the default separator is "_", which results in names similar to "stress_xx", "stress_yy", ... "stress_zx". To eliminate the separator, specify an empty string ("") or NONE.

11.6.6. Database Name

Scope: Results Output

`Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The base name of the database containing the output results. If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. If this line is omitted, then a filename will be created from the basename of the input file with a ".e" suffix appended.

11.6.7. Database Type

Scope: Results Output

`Database Type {=|are|is}` *`DatabaseType`*

Parameter	Value	Default
DatabaseType	`{catalyst|dof|dof_exodus|exodus|exodusii|generated|genesis|parallel_exodus}`	-

Summary The database type/format to be used for the output results.

11.6.8. Edge

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Edge variables are not supported for all database types.

11.6.9. Edge Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Edge variables are not supported for all database types.

11.6.10. Element

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities"

11.6.11. Element Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same

line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities"

11.6.12. Enable Large Ids

Scope: Results Output

Summary Enable 64 bit entity IDs for output

11.6.13. Exclude

Scope: Results Output

Summary Specify that the results file will only contain a subset of the element blocks in the analysis model. The element_block_list lists only the blocks which will not be output to the results database.

11.6.14. Exists

Scope: Results Output

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{= | is}`	-
Option2	`{abort | add_suffix | append | overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis. "ADD_SUFFIX" will add a -s???? suffix where the ???? is replaced by a sequential number starting at 0002.

11.6.15. Face

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

11.6.16. Face Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

11.6.17. Global

Scope: Results Output

Summary Define the global variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

11.6.18. Global Variables

Scope: Results Output

Summary Define the global variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

11.6.19. Include

Scope: Results Output

Summary Specify that the results file will only contain a subset of the element blocks in the analysis model. The element_block_list lists only the blocks which will be output to the results database.

11.6.20. Nodal

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

11.6.21. Nodal Variables

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

11.6.22. Node

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

11.6.23. Node Variables

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

11.6.24. Nodeset

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Nodeset variables are not supported for all database types.

11.6.25. Nodeset Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Nodeset variables are not supported for all database types.

11.6.26. Output Mesh

Scope: Results Output

`Output Mesh {=|are|is}` *`OutputMesh`*

Parameter	Value	Default
OutputMesh	`{exposed surface|refined|unrefined}`	-

Summary Use this command to turn on "unrefined" as the output mesh. The default behavior is "refined", in which field variables are output on the current mesh, which may have been refined (either uniformly or adaptively) or had its topology altered in some way (e.g., dynamic load balancing) with respect to the original mesh read from the input file. By specifying "Output Mesh = unrefined", all output variables are output only on the original mesh objects read from the input file.

11.6.27. Output On Signal

Scope: Results Output

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

11.6.28. Overwrite

Scope: Results Output

`Overwrite` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{false|no|off|on|true|yes}`	-

Summary (DEPRECATED, Use EXISTS) Specify whether the database should be overwritten if it exists. The default behavior is to overwrite unless this command is specified in the output block and either off, false, or no is specified.

11.6.29. Property

Scope: Results Output

`Property` *`PropertyName`* `{=|are|is}` *`PropertyValue`*

Parameter	Value	Default
PropertyName	`string`	-
PropertyValue	`string`	-

Summary Define a database property named "PropertyName" with the value "PropertyValue". If PropertyValue consists of all digits, it will define an integer property. If PropertyValue is "true" or "yes" or "false" or "no", it will define a logical property; otherwise it will define a string property. Suppported properties are typically database dependent; Current properties are: COMPRESSION_LEVEL = [0..9] (off) COMPRESSION_SHUFFLE = true|false|on|off (off) FILE_TYPE = netcdf4 (forces use of netcdf-4 hdf5-based file) (netcdf3) INTEGER_SIZE_DB = 4|8 (4) INTEGER_SIZE_API = 4|8 (4) REAL_SIZE_DB = 4|8 (8 is default) LOGGING = true|false|on|off (off) MAX_NAME_LENGTH = value (32)

11.6.30. Sideset

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

11.6.31. Sideset Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

11.6.32. Start Time

Scope: Results Output

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

11.6.33. Surface

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

11.6.34. Surface Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

11.6.35. Synchronize Output

Scope: Results Output

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

11.6.36. Termination Time

Scope: Results Output

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

11.6.37. Timeseries Name

Scope: Results Output

`Timeseries Name {=|are|is}` *`filename`*

Parameter	Value	Default
filename	`string`	-

Summary Optionally specify a filename for a timeseries file that outputs the root database filename in the order that they are written. This is useful when running on large numbers of processors with many mesh-mods that cause simple disk operations to hang.

11.6.38. Timestep Adjustment Interval

Scope: Results Output

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	`integer`	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

11.6.39. Title

Scope: Results Output

Summary Specify the title to be used for this specific output block.

11.6.40. Use Output Scheduler

Scope: Results Output

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	`string`	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

11.7. SOLID OBJECT

Scope: Fuego Region

`Begin Solid Object` *`Objectname`*

`Cylinder Base {=|are|is}` *Param2*$_1$ *Param2*$_2$ *Param2*$_3$ `Top {=|are|is}` *Param5*$_1$ *Param5*$_2$ *Param5*$_3$ `Radius {=|are|is}` *Param8*

`Rectangle Center {=|are|is}` *Param2*$_1$ *Param2*$_2$ *Param2*$_3$ `Xyz {=|are|is}` *Param5*$_1$ *Param5*$_2$ *Param5*$_3$

`Sphere Center {=|are|is}` *Param2*$_1$ *Param2*$_2$ *Param2*$_3$ `Radius {=|are|is}` *Param5*

`End`

Summary Contains the commands needed to create a solid object

11.7.1. Cylinder

Scope: Solid Object

`Cylinder Base {=|are|is}` *Param2*$_1$ *Param2*$_2$ *Param2*$_3$ `Top {=|are|is}` *Param5*$_1$ *Param5*$_2$ *Param5*$_3$ `Radius {=|are|is}` *Param8*

Parameter	Value	Default
Param2	`real_1 real_2 real_3`	-
Param5	`real_1 real_2 real_3`	-
Param8	`real`	-

Summary Specify cylinder as interface surface, with given radius and two endpoints of axis p1(base) and p2(top). The length of the cylinder is norm(p2 - p1).

11.7.2. Rectangle

Scope: Solid Object

`Rectangle Center {=|are|is}` *Param2*$_1$ *Param2*$_2$ *Param2*$_3$ `Xyz {=|are|is}` *Param5*$_1$ *Param5*$_2$ *Param5*$_3$

Parameter	Value	Default
Param2	`real_1 real_2 real_3`	-
Param5	`real_1 real_2 real_3`	-

Summary Specify rectangle as interface surface, with given center and x,y,z sizes.

11.7.3. Sphere

Scope: Solid Object

`Sphere Center {=|are|is}` *Param2*$_1$ *Param2*$_2$ *Param2*$_3$ `Radius {=|are|is}` *Param5*

Parameter	Value	Default
Param2	`real_1 real_2 real_3`	-
Param5	`real`	-

Summary Specify sphere as interface surface with given center and radius.

11.8. VIRTUAL THERMOCOUPLE MODEL ON BLOCK

Scope: Fuego Region

```
Begin Virtual Thermocouple Model On Block BlockName
    Compute Steady Solution
    Density {=|are|is} Value
    Diameter {=|are|is} Value
    Emissivity {=|are|is} Value
    Heat Capacity {=|are|is} Value
    Initial Temperature {=|are|is} Value
    Length {=|are|is} Value
    Orientation {=|are|is} Lx Ly [ Lz ]
End
```

Summary Activate the virtual thermocouple model on the requested mesh block

Description The virtual thermocouple model assumes one-way coupling between the fluid and the thermocouple. The thermocouple is assumed to be small enough that it does not influence the fluid behavior in any way.

For simulations without a PMR solve the radiative terms of the thermocouple equation are neglected.

11.8.1. Compute Steady Solution

Scope: Virtual Thermocouple Model On Block

Summary Compute the steady-state temperature for the thermocouple given the current FUEGO/SRYINX solutions. By default, the thermocouple temperature will be updated in a time-accurate manner.

11.8.2. Density

Scope: Virtual Thermocouple Model On Block

`Density {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Specify the density of the thermocouple

11.8.3. Diameter

Scope: Virtual Thermocouple Model On Block

`Diameter {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Specify the diameter of the thermocouple (assumed to be a cylinder)

11.8.4. Emissivity

Scope: Virtual Thermocouple Model On Block

`Emissivity {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Specify the emissivity of the thermocouple

11.8.5. Heat Capacity

Scope: Virtual Thermocouple Model On Block

`Heat Capacity {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Specify the heat capacity of the thermocouple

11.8.6. Initial Temperature

Scope: Virtual Thermocouple Model On Block

`Initial Temperature {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Specify the initial temperature of the thermocouple

11.8.7. Length

Scope: Virtual Thermocouple Model On Block

`Length {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Specify the length of the thermocouple (assumed to be a cylinder)

11.8.8. Orientation

Scope: Virtual Thermocouple Model On Block

`Orientation {=|are|is}` *`Lx Ly [ Lz ]`*

Parameter	Value	Default
Lx	`real`	-
Ly	`real`	-

Summary Specify the spatial orientation vector of the thermocouple

11.9. SOLUTION OPTIONS

Scope: Average Region

Begin Solution Options *OptionsName*

Activate Acoustic Compressibility Algorithm

Activate Equation *Equations*

Activate Full Surface Cvfem Gradient Operator For Muscl Scheme

Activate Lighthill Tensor Postprocessing

Activate Species Enthalpy Calculations

Activate Viscous Dissipation Source Term

Compute Steady Solution Using Pseudo Transient Method

Coordinate System {=|are|is} *CoordSys*

First Order Upwind Factor {=|are|is} *Value* [For Equation *Equations*]

Fix Pressure To *FixedPressure* At A Single Node

Hybrid Upwind Factor {=|are|is} *Value* [For Equation *Equations*]

Hybrid Upwind Method {=|are|is} *HybridMethod* [For Equation *Equations*]

Hybrid Upwind Shift {=|are|is} *Value* [For Equation *Equations*]

Hybrid Upwind Width {=|are|is} *Value* [For Equation *Equations*]

Include Continuity Residual Term [With Diagnostics]

Maximum Number Of Continuity_Momentum Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Energy_Species Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Gas_Solid_Momentum Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Kepsilon Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Komega Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Ksgs Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Mixture Fraction Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Solid Phase Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Soot Nuclei Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Species Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Species_Product Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of V2F Nonlinear Iterations {=|are|is} *Number*

Maximum Wall Time {=|are|is} *WallTime* Hours

Minimum Number Of Nonlinear Iterations {=|are|is} *Number*

Nonlinear Residual Norm Tolerance {=|are|is} *Tolerance* [For Equation *Equations*]

Nonlinear Stabilization Method {=|are|is} *NSOMethod* [For Equation *Equations*]

Omit Density Time Derivative In Continuity Equation [For *OmitSteps* Steps And Blend In Over *BlendSteps* Steps]

Output Nonlinear Residual Field For Equation *Equations* As *ResName* [On Output Block *BlockName*]

Periodic Constant Momentum Body Source Term {=|are|is} $ConstSrc_1$ $ConstSrc_2$ $ConstSrc_3$

Projection Method {=|are|is} *ProjectionMethod* Smoothing [With *ProjectionScaling* Scaling]

Randomize Pressure

Skip Pressure Update If Continuity Solve Fails

Source Term Function {=|are|is} *FuncStr* For Equation *Equation* [*SpeciesName*]

Source Term Subroutine {=|are|is} *Subroutine* For Equation *Equations* [*SpeciesName*]

Stop Simulation If Peak Velocity Exceeds *MaxVel*

Under Relax *Equations* By *Urf* [With Implicit Term]

Under Relax Momentum By *Urf*

Under Relax Pressure By *Urf*

Under Relax Solid_Momentum By *Urf*

Under Relax Temperature_Extraction By *Urf*

Upwind Limiter {=|are|is} *UpwindLimiter* [For Equation *Equations*]

Upwind Method {=|are|is} *UpwindMethod* [For Equation *Equations*]

Use Equation Solver *SolverName* For Equation *Equations*

Use External Continuity Source

Use External Energy Source

Use External Mixture_Fraction Source

Use External Momentum Source

Use External Soot_Mass_Fraction Source

Use External Species Source

Use Lumped Velocity Density Interpolation

Use Radiation Source From External Region [Using Classic Linearization]

Use Skew Symmetric Central Operator [For Equation *Equations*]

Begin Acoustic Transfer Output *DefinitionName*

End

Begin Beam Radiation Boundary Specification *DefinitionName*

End

Begin Buoyancy Model Specification *BuoyModelName*

End

Begin Edc Model Specification *EdcSpecName*

End

Begin Mesh Motion Specification *DefinitionName*

End

Begin Multiphase Model Specification *DefinitionName*

End

Begin Point Source *DefinitionName*

End

```
Begin Rad Transport Spectral Model Specification DefinitionName
End

Begin Radiation Transport Equation Model Specification
RadModelName
End

Begin Time Integration Specification TimeIntSpecName
End

Begin Turbulence Model Specification TurbSpecName
End

Begin Vof Model Specification DefinitionName
End

End
```

Summary Specify information regarding the governing equations to be solved.

11.9.1. Activate Acoustic Compressibility Algorithm

Scope: Solution Options

Summary Variable thermodynamic pressure

Description This option will allow for closed system pressurization either through heat-up or inflow of fluid.

The algorithm will add the substantial derivative of pressure, $\frac{\partial p}{\partial t} + u_j \frac{\partial p}{\partial x_j}$, to the laminar or turbulent enthalpy transport equation and to the laminar temperature transport equation. Additionally, the viscous work term $u_i \frac{\partial \tau_{ij}}{\partial x_j}$ will be added to the turbulent enthalpy equation. An implicit term in the continuity solve is added through the time density derivative. As such, Cantera support is required. The convective terms within the continuity solve are neglected.

Caveats for this model:

1) The Cantera material model evaluator must be used.

2) The initial pressure and any boundary condition pressures must be specified with respect to the datum pressure.

If a zero datum pressure is specified, then all initial and boundary pressures will be in absolute units. If this is a coupled structural simulation, then the surface traction due to

this pressure will need to be counteracted with a load on the "back side" that is equivalent to the ambient pressure in absolute units.

If a non-zero datum pressure is specified, then all initial and boundary pressures will be in relative units with respect to this datum. Pressure can then be thought of as a gauge pressure with respect to the datum. The "back side" load in structural simulations must be set accordingly. (For example, if the datum is set equal to the external ambient pressure, 1 atm, and the initial pressure is set to zero, then the initial surface traction force due to pressure will be zero and no "back side" load due to the ambient pressure should be specified.)

11.9.2. Activate Equation

Scope: Solution Options

`Activate Equation` *`Equations`*

Parameter	Value	Default
Equations	{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}	-

Summary Activate the specified equation.

11.9.3. Activate Full Surface Cvfem Gradient Operator For Muscl Scheme

Scope: Solution Options

Summary Use full stencil for gradient used in MUSCL convection operator

Description The default gradient operator for the MUSCL scheme is the edge-based stencil. This option keeps integration points at the subcontrol surface points.

11.9.4. Activate Lighthill Tensor Postprocessing

Scope: Solution Options

Summary Postprocesses the nodal divergence of the Lighthill tensor

Description This calculates the nodal divergence of the Lighthill tensor, used for acoustic analysis.

11.9.5. Activate Species Enthalpy Calculations

Scope: Solution Options

Summary Enables calculation of species enthalpy

Description This forces the calculation of species enthalpy, needed primarily for coupled Fuego-Aria problems.

11.9.6. Activate Viscous Dissipation Source Term

Scope: Solution Options

Summary Add viscous dissipation source term

Description For low speed viscous dissipation effects, this source term will provide the viscous work source term in the static enthalpy equation. This source term is a subset of the full acoustically compressible source term option, however, the substantial pressure derivative is omitted.

11.9.7. Compute Steady Solution Using Pseudo Transient Method

Scope: Solution Options

Summary Compute a steady-state solution using the pseudo-transient method (time march to steady solution).

Description The solution will march forward in time until either the stopping time is reached or the steady convergence criterion is met. Convergence to steady state is detected when all equations meet their nonlinear residual norm tolerances after the first nonlinear iteration, since this will only occur as the solution stops changing between time steps. The nonlinear residual norm tolerances should be set small enough to prevent false positives. Also make sure the simulation time is set to be fairly large, to prevent a premature end to the simulation before convergence is achieved.

If you are using solution control, then you also need to test for a region parameter to stop the simulation. In your `PARAMETERS FOR TRANSIENT` solution control block, add the line (assuming your Fuego region is called `fuego_region`):

```
CONVERGED WHEN "fuego_region.REGION_STEADY_STATE == 1"
```

11.9.8. Coordinate System

Scope: Solution Options

`Coordinate System {=|are|is}` *`CoordSys`*

Parameter	Value	Default
CoordSys	`{2d|3d|xaxi|yaxi}`	`3D`

Summary Specify the coordinate system.

11.9.9. First Order Upwind Factor

Scope: Solution Options

`First Order Upwind Factor {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
Value	`"string"`	`1.0`

Summary First-order upwind factor, o extless x extgreater 1.

Description This value specifies the explicit upwind blending between pure upwind and the chosen convection operator, e.g., UPW*(firstOrderUpwind) + (1-firstOrderUpwind)*(blendedUpwindCentral).

where UPW is the pure first order upwind value and blendedUpwindCentral is a blend between the selected upwind method and central difference operator based on the local cell Peclet number (see Hybrid Upwind Factor line command). The value can be a time dependent string function.

Values for individual equation sets may be set using optional token. Using both (in either order):

- FIRST ORDER UPWIND FACTOR = String
- FIRST ORDER UPWIND FACTOR = String FOR EQUATION Equations

Will result in the particular equation set to specified value while all others set to general value.

11.9.10. Fix Pressure To

Scope: Solution Options

`Fix Pressure To` *`FixedPressure`* `At A Single Node`

Parameter	Value	Default
FixedPressure	`real`	`0.0`

Summary Sets a dirichlet for pressure at a single arbitrary node. This is required for a well posed pressure equation if none of the boundaries specify pressure (e.g. open).

11.9.11. Hybrid Upwind Factor

Scope: Solution Options

`Hybrid Upwind Factor {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
Value	`"string"`	`1.0`

Summary Hybrid upwinding factor.

Description The upwind schemes are blended with a centered scheme. The HYBRID UPWIND FACTOR is a multiplier against the cell Peclet number used in the switching scheme (see First Order Upwind Factor line command).

- A HYBRID UPWIND FACTOR = 0.0 results in all centered.
- A HYBRID UPWIND FACTOR = 1.0 results in default hybrid.
- A HYBRID UPWIND FACTOR » 1.0 results in all upwind.

Values for individual equation sets may be set using optional token. Using both (in either order):

- HYBRID UPWIND FACTOR = String
- HYBRID UPWIND FACTOR = String FOR EQUATION Equations

Will result in the particular equation set to specified value while all others set to general value. The value can be specified as a time dependent string function or a constant.

11.9.12. Hybrid Upwind Method

Scope: Solution Options

`Hybrid Upwind Method {=|are|is}` *`HybridMethod`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
HybridMethod	`{blending|tanh}`	`BLENDING`

Summary Hybriding method between central and upwind using Peclet number

Description BLENDING and TANH approaches are currently supported.

Function χ determines the ratio between user-chosen upwind (χ) and central (1-χ) operators. The need for an upwind operator is affected by cell-Peclet number.

BLENDING uses HYBRID UPWIND FACTOR (ζ) and the function is, $\chi = \frac{(\zeta Pe)^2}{5+(\zeta Pe)^2}$.

TAHH follows hyperbolic tangent profile between χ and Pe. It uses shift p (HYBRID UPWIND SHIFT) and width w (HYBRID UPWIND WIDTH) parameters as, $\chi = \frac{1}{2}[1 + tanh(\frac{Pe-p}{w})]$. Tanh is centered ($\chi = 0.5$) when Peclet number is at the shifting factor p. Width determines how fast χ changes with Peclet number as follows:

- χ =0.5 at Pe=p.
- χ =0.8808 and 0.1192 at Pe=p+w and p-w.
- χ =0.9820 and 0.0180 at Pe=p+2w and p-2w.
- χ =0.9975 and 0.0025 at Pe=p+3w and p-3w.
- χ =0.9997 and 0.0003 at Pe=p+4w and p-4w.

TANH allows users to effectively remove upwind contribution for lower Pe. In the other extreme, one can enforce user-chosen upwind at all Pe if p < 0.0 and w « 1.0 (ex> p=-1.0, w=1e-10).

11.9.13. Hybrid Upwind Shift

Scope: Solution Options

`Hybrid Upwind Shift {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
Value	`"string"`	`0.0`

Summary Shifting factor for TANH hybrid approach. Can be specified as a time dependent string function or a constant.

Description (see HYBRID UPWIND METHOD description)

11.9.14. Hybrid Upwind Width

Scope: Solution Options

`Hybrid Upwind Width {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
Value	`"string"`	`1.0`

Summary Width factor for TANH hybrid approach

Description Minimum value for this parameter is 1e-10. (see HYBRID UPWIND METHOD description) Can be specified as a time dependent string function or a constant value.

11.9.15. Include Continuity Residual Term

Scope: Solution Options

Summary Include the continuity residual term in transport equations

Description Continuity is not exactly satisfied during the momentum solve for variable density flows since the mass flux is lagged while the density is updated with new properties. Including the continuity error in momentum can keep the momentum prediction better behaved. The residual should be on the order of the linear solver tolerance for other equations, but including the term can also make the other solves more robust to a bad continuity solve.

This term is always included when using VOF or a deforming mesh.

11.9.16. Maximum Number Of Continuity_Momentum Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Continuity_Momentum Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the momentum/continuity solve.

11.9.17. Maximum Number Of Energy_Species Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Energy_Species Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the energy-species grouping.

11.9.18. Maximum Number Of Gas_Solid_Momentum Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Gas_Solid_Momentum Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the gas/solid momentum sets of equations.

11.9.19. Maximum Number Of Kepsilon Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Kepsilon Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the k-epsilon turbulence model equations grouping.

11.9.20. Maximum Number Of Komega Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Komega Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the k-omega turbulence model equations grouping.

11.9.21. Maximum Number Of Ksgs Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Ksgs Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the ksgs turbulence model equations grouping.

11.9.22. Maximum Number Of Mixture Fraction Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Mixture Fraction Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the mixture fraction equations grouping.

11.9.23. Maximum Number Of Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the time step of the Fuego region.

11.9.24. Maximum Number Of Solid Phase Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Solid Phase Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the solid momentum/continuity sets of equations.

11.9.25. Maximum Number Of Soot Nuclei Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Soot Nuclei Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the soot nuclei equations grouping.

11.9.26. Maximum Number Of Species Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Species Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations for the species equations. If the EDC product transport feature is active, then the SPECIES_PRODUCT nonlinear iteration count should be set instead.

11.9.27. Maximum Number Of Species_Product Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Species_Product Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the Species/EDC_Product grouping. This is only used if the EDC model is active and the EDC product transport feature is being used.

11.9.28. Maximum Number Of V2F Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of V2F Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the v2f turbulence model equations grouping.

11.9.29. Maximum Wall Time

Scope: Solution Options

`Maximum Wall Time {=|are|is}` *`WallTime`* `Hours`

Parameter	Value	Default
WallTime	`real`	`Infinite`

Summary Specify a maximum wall time to let the simulation end gracefully and output before slurm kills it.

11.9.30. Minimum Number Of Nonlinear Iterations

Scope: Solution Options

`Minimum Number Of Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Minimum number of nonlinear iterations to take within the time step of the Fuego region.

11.9.31. Nonlinear Residual Norm Tolerance

Scope: Solution Options

`Nonlinear Residual Norm Tolerance {=|are|is}` *`Tolerance`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
Tolerance	`real`	`1.0e-15`

Summary Nonlinear convergence tolerance within a time step in the Fuego region.

Values for individual equation sets may be set using the optional token. Using both (in either order):

```
NONLINEAR RESIDUAL NORM TOLERANCE = {Real}
NONLINEAR RESIDUAL NORM TOLERANCE = {Real} FOR EQUATION {Equations}
```

Will result in the particular equation set to specified value while all others set to general value.

11.9.32. Nonlinear Stabilization Method

Scope: Solution Options

`Nonlinear Stabilization Method {=|are|is}` *`NSOMethod`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
NSOMethod	`{commutation_error|none|pointwise_residual_error}`	`NO_NSO`

Summary Specify a artificial viscosity stabilization; default is NONE.

Description Values for individual equation sets may be set using optional token. Using both (in either order):

- NSO METHOD = NSOMethod
- NSO METHOD = NSOMethod FOR EQUATION Equations

Will result in the particular equation set to specified value while all others set to general value.

11.9.33. Omit Density Time Derivative In Continuity Equation

Scope: Solution Options

Summary Remove density time derivative in continuity equation

Description Remove the density time derivative from the continuity equation. This feature is required for closed boundary flows with accumulation.

The optional arguments let you omit it for a certain number of timesteps at the start of the simulation, then gradually include it over a number of steps.

11.9.34. Output Nonlinear Residual Field For Equation

Scope: Solution Options

`Output Nonlinear Residual Field For Equation` *`Equations`* `As` *`ResName`* `[ On Output Block` *`BlockName`* `]`

Parameter	Value	Default
Equations	{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}	-
ResName	string	-

Summary Generates output of nonlinear residuals for the requested equation.

Description Provide nonlinear residual for output for specified equation. If the optional output block name is specified, then the residual will only be written to that output block.

11.9.35. Periodic Constant Momentum Body Source Term

Scope: Solution Options

Periodic Constant Momentum Body Source Term {=|are|is} *ConstSrc*$_1$ *ConstSrc*$_2$ *ConstSrc*$_3$

Parameter	Value	Default
ConstSrc	real_1 real_2 real_3	-

Summary Add constant body force due to periodic config

Description For periodic BCS, commonly a constant body force is applied to drive the flow. This line command allows one to provide a constant body force in three dimensions. If more complex sources are needed, the user sub source term procedure is required.

11.9.36. Projection Method

Scope: Solution Options

Projection Method {=|are|is} *ProjectionMethod* Smoothing [With *ProjectionScaling* Scaling]

Parameter	Value	Default
ProjectionMethod	{filtered | fourth_order | second_order | stabilized | zeroth_order}	-

Summary Specify choice of projection method.

Description The smoothing choice may include zeroth, second, or fourth order. No smoothing (zeroth) may allow pressure-velocity decoupling.

The scaling term may be specified. This scaling term is related to the factorization approximation to the inverse of the momentum matrix.

Time step scaling may show results that are sensitive to the chosen simulation dt at coarse meshes. This error should vanish as the pressure field approached a linear shape, or refinement is performed. Note that characteristic scaling also has the same error, however, its manifestation is less obvious.

The stabilized option uses a fourth order smoothing term and characteristic scaling along with an additional dt stabilizing term.

In general, the stabilized and "fourth order smoothing" timestep scaling allows for larger time steps. Characteristic scaling seems to limit CFL to below unity, presumably due to stability loss during nodal projection, i.e., splitting error is (I-tau*A)G(deltaP).

"Momentum Scaling", uses the diagonal of the momentum equation as the scaling term. While the leading order term with this method will be similar to the timestep scaling scheme, it can sometimes offer better stability since it also includes effects from the other terms in the momentum equation.

11.9.37. Randomize Pressure

Scope: Solution Options

Summary Set a random pressure field for initial guess

Description Randomize the initial guess to the linear solve for pressure. The randomization is imposed after the nonlinear residual is computed.

11.9.38. Skip Pressure Update If Continuity Solve Fails

Scope: Solution Options

Summary Do not update the pressure field or mdot if the continuity solve fails

Description If the continuity solve fails the resulting pressure delta may be large or non-physical. Activating this option skips the pressure update and mdot update when the solver fails. Repeated solver failures should be watched for in the log file.

This is a beta feature.

11.9.39. Source Term Function

Scope: Solution Options

`Source Term Function {=|are|is}` *`FuncStr`* `For Equation` *`Equation`* `[` *`SpeciesName`* `]`

Parameter	Value	Default
FuncStr	`"string"`	-
Equation	`{conserved_enthalpy|continuity|edc_product|enthalpy|mixture_fraction|nuclei|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot|species|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_momentum|x_solid_momentum|y_momentum|y_solid_momentum|z_momentum|z_solid_momentum}`	-

---

Summary

Source term string function to use for the given equation. Registered variables with aliases include time (t), spatial coordinates (x,y,z), velocity (u,v,w), density (rho), and pressure (p). Additionally, any valid global variable or nodal variable can be used with its full name. Vector variables, like mass fraction, must be indexed numerically (e.g. "mass_fraction[3]")

The function string must be enclosed in quotes if it has spaces or commas. For example: Source Term Function for x_momentum = "min(1, 0.1*t)"

11.9.40. Source Term Subroutine

Scope: Solution Options

`Source Term Subroutine {=|are|is}` *`Subroutine`* `For Equation` *`Equations`* `[` *`SpeciesName`* `]`

Parameter	Value	Default
Subroutine	`string`	-
Equations	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

Summary Source term user subroutine for the given equation. This is often useful in verification studies where one wishes to use a manufactured solution and must provide source terms for various governing equations.

11.9.41. Stop Simulation If Peak Velocity Exceeds

Scope: Solution Options

`Stop Simulation If Peak Velocity Exceeds` *`MaxVel`*

Parameter	Value	Default
MaxVel	`real`	`infinity`

Summary Abort the simulation if velocities get too large.

Description By default Fuego will continue time stepping as the simulation diverges and will go until velocities overflow or solvers start returning NaN or Inf.

If you want it to stop sooner than that, you can set a peak velocity magnitude to abort at.

11.9.42. Under Relax

Scope: Solution Options

`Under Relax` *`Equations`* `By` *`Urf`* `[ With Implicit Term ]`

Parameter	Value	Default
Equations	{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}	-
Urf	"string"	1.0

Summary Under relaxation factor for the given equation.

Description Implicit relaxation is applied to the momentum equations. Explicit relaxation is applied to the pressure update. Transport equations are relaxed explicitly unless the "WITH IMPLICI TERM" option is used.

Under-relaxation can be a constant value or a function of time (t). If the function used has spaces or commas, it should be enclosed in quotes. The value will be internally clipped between 1e-6 and 1.

11.9.43. Under Relax Momentum By

Scope: Solution Options

`Under Relax Momentum By` *Urf*

Parameter	Value	Default
Urf	"string"	-

Summary Under relaxation factor for the momentum equations.

Under-relaxation can be a constant value or a function of time (t). If the function used has spaces or commas, it should be enclosed in quotes. The value will be internally clipped between 1e-6 and 1.

11.9.44. Under Relax Pressure By

Scope: Solution Options

`Under Relax Pressure By` *Urf*

Parameter	Value	Default
Urf	`"string"`	-

Summary Under relaxation factor for the pressure. This is equivalent to specifying an URF on continuity, and is provided for backward compatibility.

Under-relaxation can be a constant value or a function of time (t). If the function used has spaces or commas, it should be enclosed in quotes. The value will be internally clipped between 1e-6 and 1.

11.9.45. Under Relax Solid_Momentum By

Scope: Solution Options

`Under Relax Solid_Momentum By` *Urf*

Parameter	Value	Default
Urf	`"string"`	-

Summary Under relaxation factor for the solid-phase momentum equations.

Under-relaxation can be a constant value or a function of time (t). If the function used has spaces or commas, it should be enclosed in quotes. The value will be internally clipped between 1e-6 and 1.

11.9.46. Under Relax Temperature_Extraction By

Scope: Solution Options

`Under Relax Temperature_Extraction By` *Urf*

Parameter	Value	Default
Urf	`"string"`	-

Summary Under relaxation factor for the temperature extraction from enthalpy

Description Relax the temperature computed from the enthalpy. This gives a temperature that is not entirely consistent with the current state (composition and enthalpy), and will destroy time-accuracy unless sufficient Picard loops are taken. However, it may be useful for steady-state computations where species and energy equations are not being coupled strongly or solved accurately.

Under-relaxation can be a constant value or a function of time (t). If the function used has spaces or commas, it should be enclosed in quotes. The value will be internally clipped between 1e-6 and 1.

11.9.47. Upwind Limiter

Scope: Solution Options

`Upwind Limiter {=|are|is}` *`UpwindLimiter`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
UpwindLimiter	{minmod | none | superbee | van_albada | van_leer}	SUPERBEE

Summary Specify a limiter for convection operator; default is SUPERBEE.

Description Limiter functions are valid only for the MUSCL scheme.

Values for individual equation sets may be set using optional token. Using both (in either order):

- UPWIND LIMITER = UpwindLimiter
- UPWIND LIMITER = UpwindLimiter FOR EQUATION Equations

Will result in the particular equation set to specified value while all others set to general value.

Note: Rotational invariance of the code is not expected while using a limiter function.

11.9.48. Upwind Method

Scope: Solution Options

`Upwind Method {=|are|is}` *`UpwindMethod`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
UpwindMethod	{lps | muscl | upw}	LPS

Summary Upwind method for convective terms

Description All methods are hybrid in the sense that a centered scheme is blended based on the local Peclet number.

Values for individual equation sets may be set using optional token. Using both (in either order):

- UPWIND METHOD = UpwindMethod
- UPWIND METHOD = UpwindMethod FOR EQUATION Equations

Will result in the particular equation set to specified value while all others set to general value.

11.9.49. Use Equation Solver

Scope: Solution Options

`Use Equation Solver` *`SolverName`* `For Equation` *`Equations`*

Parameter	Value	Default
SolverName	`string`	-
Equations	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

---

Summary Link an equation solver to an equation set.

Description For example, if a solver block "scalar" was created using the Aztec package, e.g., BEGIN AZTEC EQUATION SOLVER scalar and the equation set was the u-component of momentum then the line command would be as follows: USE EQUATION SOLVER scalar FOR EQUATION X-Momentum

11.9.50. Use External Continuity Source

Scope: Solution Options

Summary Add external species source term from a transfer

Description Add source terms to the continuity equation from a nodal field transferred to this region, e.g. from a Fuego particle region. The field continuity_source will be added to the RHS of the continuity equation; this fields should have units of rate-of-change of mass per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

11.9.51. Use External Energy Source

Scope: Solution Options

Summary Add external energy source term from a transfer

Description Add source terms to the temperature or enthalpy equations from a nodal field transferred to this region, e.g. from a Fuego particle region. The field energy_source will be added to the RHS of the energy equation; this fields should have units of rate-of-change of energy per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

11.9.52. Use External Mixture_Fraction Source

Scope: Solution Options

Summary Add external species source term from a transfer

Description Add source terms to the mixture fraction equation from a nodal field transferred to this region, e.g. from a Fuego particle region. The field mixture_fraction_source will be added to the RHS of the mixture fraction equation; this fields should have units of rate-of-change of mass per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

11.9.53. Use External Momentum Source

Scope: Solution Options

Summary Add external momentum source terms from a transfer

Description Add source terms to the momentum equations from a nodal field transferred to this region, e.g. from a Fuego particle region. The fields x_momentum_source, y_momentum_source, and z_momentum source will be added to the RHS of the momentum equations; these fields should have units of rate-of-change of momentum per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

11.9.54. Use External Soot_Mass_Fraction Source

Scope: Solution Options

Summary Add external soot source term from a transfer

Description Add source terms to the soot mass fraction equation from a nodal field transferred to this region, e.g. from a Fuego particle region. The field soot_mass_fraction_source will be added to the RHS of the soot mass fraction equation; this fields should have units of rate-of-change of mass per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

11.9.55. Use External Species Source

Scope: Solution Options

Summary Add external species source term from a transfer

Description Add source terms to the species equations from a nodal field transferred to this region, e.g. from a Fuego particle region. The vector field species_source will be added to the RHS of the species equation; this fields should have units of rate-of-change of mass of species i per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

11.9.56. Use Lumped Velocity Density Interpolation

Scope: Solution Options

Summary Interpolate the density-velocity product

Description By default the continuity equation interpolates velocity and density separately to sub-control surfaces. This option interpolates the product of density times velocity instead.

11.9.57. Use Radiation Source From External Region

Scope: Solution Options

Summary Add in a source term from participating-media radiation which comes from another region through a transfer.

The USING CLASSIC LINEARIZATION optional argument is no longer used or needed, and will be removed in a future release.

11.9.58. Use Skew Symmetric Central Operator

Scope: Solution Options

Summary The blended central operator will be skew symmetric, default is **false**.

Description The convection operator is always blended with pure central (see hybrid factor description). For the CVFEM methodology, there is a balance between stability and accuracy. Dotting the momentum equation with velocity and summing yields the kinetic energy equation. If the convection operator is skew symmetric, than this dot product leaves something that is perfectly zero. This means that there can be no generation of kinetic energy and simulations can remain stable.

The full CVFEM stencil (27-pt on a hex mesh) is not skew symmetric. Therefore, in cases where one uses pure central (by specifying a hybrid factor of unity) there can be issues - especially on coarse meshes.

11.10. BUOYANCY MODEL SPECIFICATION

Scope: Solution Options

```
Begin Buoyancy Model Specification BuoyModelName
    Buoyancy Model {=|are|is} BuoyancyModel
    Buoyancy Reference VarName {=|are|is} Value
    Buoyancy Reference Mass_Fraction Species {=|are|is} y0
    Buoyancy Reference Mole_Fraction Species {=|are|is} x0
    Buoyancy Source Term {=|are|is} BuoyancySourceTermType
End
```

Summary Specify buoyancy modeling options.

11.10.1. Buoyancy Model

Scope: Buoyancy Model Specification

`Buoyancy Model {=|are|is}` *BuoyancyModel*

Parameter	Value	Default
BuoyancyModel	{boussinesq|buoyant|concentration|differential|no_buoyancy}	NO_BUOYANCY

Summary Specification of Buoyancy model to be used in momentum equations.

11.10.2. Buoyancy Reference

Scope: Buoyancy Model Specification

`Buoyancy Reference` *`VarName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
VarName	`string`	-
Value	`real`	-

Summary Buoyancy reference value for the given field.

Description If the selected buoyancy model requires material property calculations, then reference values must be specified for all input variables required by the particular material model. In the typical case where Cantera is used for property evaluation, then reference properties must be specified for "temperature", "pressure", and "mass_fraction". Other material models may have different dependencies.

11.10.3. Buoyancy Reference Mass_Fraction

Scope: Buoyancy Model Specification

`Buoyancy Reference Mass_Fraction` *`Species`* `{=|are|is}` *`y0`*

Parameter	Value	Default
Species	`string`	-
yo	`real`	-

Summary Buoyancy reference mass fraction for calculating density.

Description If the selected buoyancy model requires material property calculations, then reference values must be specified for all input variables required by the particular material model. In the typical case where Cantera is used for property evaluation, then reference properties must be specified for "temperature", "pressure", and "mass_fraction". Other material models may have different dependencies.

11.10.4. Buoyancy Reference Mole_Fraction

Scope: Buoyancy Model Specification

`Buoyancy Reference Mole_Fraction` *`Species`* `{=|are|is}` *`x0`*

Parameter	Value	Default
Species	`string`	-
xo	`real`	-

Summary Buoyancy reference mole fraction for calculating density.

Description If the selected buoyancy model requires material property calculations, then reference values must be specified for all input variables required by the particular material model. In the typical case where Cantera is used for property evaluation, then reference properties must be specified for "temperature", "pressure", and "mass_fraction". Other material models may have different dependencies.

11.10.5. Buoyancy Source Term

Scope: Buoyancy Model Specification

`Buoyancy Source Term {=|are|is}` *`BuoyancySourceTermType`*

Parameter	Value	Default
BuoyancySourceTermType	`{consistent|default|lumped}`	`DEFAULT`

Summary Specifies if the buoyancy source term should be lumped, consistent, or the same as the mass matrix.

Description Using a lumped source term diagonalizes the mass matrix contribution to the linear system, and therefore results in a more diagonally-dominant matrix. This under-integration of the source term may result in more monotonic behavior for certain problems.

Using a different source term and mass matrix lumping may introduce undesirable behavior near high gradient regions. Default behavior is to use the same approach for the mass matrix and buoyancy source term.

11.11. EDC MODEL SPECIFICATION

Scope: Solution Options

```
Begin Edc Model Specification EdcSpecName

    Activate Absorption Model

    Activate Co2 Dissociation Model

    Activate Hydrogen Dissociation Model

    Activate Separate Co Irreversible Oxidation Pathway

    Characteristic Length Scale For Absorption Coefficient
    Determination {=|are|is} CharacteristicLength

    Critical Damkohler Number For Suppression {=|are|is} dacrit

    Edc Fuel Name {=|are|is} Fuel

    Edc Ignition Threshold Temperature {=|are|is} IgnTemp
```

```
    Edc Ignition Time {=|are|is} Igntime

    Edc Minimum Product Fraction {=|are|is} Prmin

    Edc Reaction Time Scale {=|are|is} Tchem

    Include Edc Laminar Limit Model

    Minimum Soot Production Temperature {=|are|is} Tsootmin

    Omit Near Wall Combustion

    Use Edc Product Transport

    Use Explicit Treatment Of Edc Source Terms

    Use Pure Oxygen For Oxidizer Mixture

    Use Sintef Soot Model

End
```

Summary Specify EDC combustion model options.

11.11.1. Activate Absorption Model

Scope: Edc Model Specification

Summary Calculate absorption coefficient for use in conjunction with radiation calculations. This will also ensure that the rte source term includes the convolution over gamma chi. If a rte block is active, you may also elect to compute the rte source term by the use of mean temperatures.

11.11.2. Activate Co2 Dissociation Model

Scope: Edc Model Specification

Summary Include effects of CO_2 dissociation into CO and O_2 at high temperatures

Description At high temperatures, the equilibrium between CO_2, CO, and O_2 shifts away from CO_2, which can significantly decrease the flame temperature. Activating this model will add this effect to the standard EDC combustion model.

11.11.3. Activate Hydrogen Dissociation Model

Scope: Edc Model Specification

Summary Include effects of H2 dissociation into H

Description At temperatures greater than about 2000K, the equilibrium between H2 and H will yield non-negligible concentrations of H which can significantly decrease flame temperatures. Activating this model will add this effect to the standard EDC combustion model using the correlations of W.W. Erikson, which are derived from the NASA CEA code [74, 75].

Note that the H species must be included in the Cantera input XML file, and neither H nor H2 should be the "last" species in the list since this species is not independent of the rest (to enforce unity sum) and may be susceptible to more noise than the others. Since temperature and other properties are very sensitive to oscillations in the H and H2 equilibrium, this noise could be problematic.

11.11.4. Activate Separate Co Irreversible Oxidation Pathway

Scope: Edc Model Specification

Summary Add CO oxidation pathway as a separate reaction pathway. This should really be used only in the context of a propellant fire in the presence of hydrogen combustion.

11.11.5. Characteristic Length Scale For Absorption Coefficient Determination

Scope: Edc Model Specification

`Characteristic Length Scale For Absorption Coefficient Determination {=|are|is}` *`CharacteristicLength`*

Parameter	Value	Default
CharacteristicLength	`real`	-

Summary Override the absorption coefficient normally calculated by fuego with this value.

11.11.6. Critical Damkohler Number For Suppression

Scope: Edc Model Specification

`Critical Damkohler Number For Suppression {=|are|is}` *`dacrit`*

Parameter	Value	Default
dacrit	`real`	`0.0`

Summary Critical Damkohler number at which suppression is activated. A value of 0.0 denotes no suppression.

11.11.7. Edc Fuel Name

Scope: Edc Model Specification

`Edc Fuel Name {=|are|is}` *Fuel*

Parameter	Value	Default
Fuel	`string`	-

Summary The name of the EDC fuel species, typically either H2 or a major hydrocarbon

11.11.8. Edc Ignition Threshold Temperature

Scope: Edc Model Specification

`Edc Ignition Threshold Temperature {=|are|is}` *IgnTemp*

Parameter	Value	Default
IgnTemp	`real`	`1000.0 K`

Summary The temperature below which the ignition model is activated

Description If the ignition model is requested (through the IGNITE initial condition keyword), then it will be activated if all temperatures in the corresponding initial condition block are below this temperature.

11.11.9. Edc Ignition Time

Scope: Edc Model Specification

`Edc Ignition Time {=|are|is}` *Igntime*

Parameter	Value	Default
Igntime	`real`	`0.0`

Summary The time at which the EDC combustion model is activated. No combustion will occur before this time.

11.11.10. Edc Minimum Product Fraction

Scope: Edc Model Specification

`Edc Minimum Product Fraction {=|are|is}` *`Prmin`*

Parameter	Value	Default
Prmin	`real`	`1.0e-6`

Summary The minimum product fraction, below which the EDC model will be deactivated.

11.11.11. Edc Reaction Time Scale

Scope: Edc Model Specification

`Edc Reaction Time Scale {=|are|is}` *`Tchem`*

Parameter	Value	Default
Tchem	`real`	`7.0e-5`

Summary Reaction time scale to set extinction

Description Characteristic time scale of the chemical kinetics. Residence time in the fine structure region will be compared to this to determine if extinction will result.

11.11.12. Include Edc Laminar Limit Model

Scope: Edc Model Specification

Summary Turn on the EDC laminar limit model for low-turbulence situations

Description Turn on the EDC laminar limit model. This model requires setting three model constants: CtauLam, CgammaLam, and ClamTrans. The model uses a time scale based on a velocity gradient rather than the turb_ke/turb_diss. This appropriate time scale permits the flame to anchor in laminar regions.

11.11.13. Minimum Soot Production Temperature

Scope: Edc Model Specification

`Minimum Soot Production Temperature {=|are|is}` *`Tsootmin`*

Parameter	Value	Default
Tsootmin	`real`	`300.0 K`

Summary The minimum temperature for which we allow soot to be produced.

11.11.14. Omit Near Wall Combustion

Scope: Edc Model Specification

Summary Disables combustion near the wall nodes by setting the production rate to zero. This prevents the values calculated for the EDC combustion rate to be added to the wall element.

11.11.15. Use Edc Product Transport

Scope: Edc Model Specification

Summary Solve a transport equation for the EDC reaction products.

Description If any of the "product" species (CO_2 or H_2O for a hydrocarbon fuel, or H_2O for hydrogen fuel) are to be injected into the domain either through a boundary condition or initial condition to simulate a diluent stream or ambient concentration, then the EDC model will be unable to differentiate which portion of the product species originated directly from combustion and which did not. This can badly distort the EDC model reaction rate calculation and yield incorrect results.

This option solves a transport equation for EDC products that were generated through combustion rather than evaluating it directly from local mass fractions, eliminating issues with product species injection. (If no product species are being injected, then identical results can be obtained at a lower cost without this option.)

To use this feature, activate this line command and also activate the EDC_Product equation along with the Species equation. Note that no initial or boundary conditions are needed for the EDC_Product equation since it has its own internally-handled special needs. (Any specified initial or boundary conditions will be ignored.)

Also note that a pilot stream will be unable to ignite a flame when using this model. It will be treated as an inert diluent stream, so that the normal ignition model will be required to ignite the flame. This model in its current form should not be used for piloted flames.

11.11.16. Use Explicit Treatment Of Edc Source Terms

Scope: Edc Model Specification

Summary Allow no implicit treatment of EDC combustion source terms

Description The general form of the EDC combustion model for species k is rate*(Yfs_k - Y_k) where rate is a function of gammachi and residence times. This option specifies that the full source term be placed on the right hand side of the species transport equation, e.g., rhs+= rate*(Yfs_k - Yk) as opposed to: rhs+=rate*(Yfs_k - Y_k) and lhs+=rate. The user will recall that the transport equations are solved in residual form, hence the above rhs form.

11.11.17. Use Pure Oxygen For Oxidizer Mixture

Scope: Edc Model Specification

Summary Use pure oxygen as the oxidizer mixture rather than air

Description By default, the EDC model assumes a 3.76 N2:O2 ratio for the oxidizer mixture. This option switches to pure O2 as the oxidizer.

11.11.18. Use Sintef Soot Model

Scope: Edc Model Specification

Summary Use the SINTEF soot model instead of the standard soot model.

11.12. MULTIPHASE MODEL SPECIFICATION

Scope: Solution Options

```
Begin Multiphase Model Specification DefinitionName

    Compute Gas Phase Volume Fraction Using GasVolFracEnum1 [
    GasVolFracEnum2]

    Enable Porous Media Model

    Multiphase MultiPhaseParams {=|are|is} Values

End
```

Summary Specify a number of physical parameters that are to be used for multi-phase flow simulations.

11.12.1. Compute Gas Phase Volume Fraction Using

Scope: Multiphase Model Specification

`Compute Gas Phase Volume Fraction Using` *GasVolFracEnum*$_1$ `[` *GasVolFracEnum*$_2$`]`

Parameter	Value	Default
GasVolFracEnum	{particle|soot}	-

Summary Compute the gas phase volume fraction based on soot and particle volume fraction.

11.12.2. Enable Porous Media Model

Scope: Multiphase Model Specification

Summary Invoke porous media (aka clutter model, darcey flow) model equations; an initial condition for gas volume fraction is expected, however, defaulted to unity.

11.12.3. Multiphase

Scope: Multiphase Model Specification

`Multiphase` *`MultiPhaseParams`* `{=|are|is}` *`Values`*

Parameter	Value	Default
MultiPhaseParams	`{compaction_modulus|density|effective_viscosity_multiplier|maximum_packing_diameter|particle_diameter|reference_elastic_modulus|solid_velocity|soot_density|turbulent_schmidt_number}`	-
Values	`real`	-

Summary Specify parameters for multi-phase simulations.

Description These values are used in solid and gas phase transport equations

11.13. RADIATION TRANSPORT EQUATION MODEL SPECIFICATION

Scope: Solution Options

```
Begin Radiation Transport Equation Model Specification RadModelName
    Absorption Coefficient Model {=|are|is} AbsorpCoeffModel
    Aluminum_Oxide Emittance Model {=|are|is} AluminaEmittModel
    Characteristic Length Scale For Absorption Coefficient
    Determination {=|are|is} CharacteristicLength
    Species Name For AbsorpCoeffSpecies {=|are|is} SpeciesName
    Subgrid Mixing Model {=|are|is} RTESubgridModel
End
```

Summary Specify options for radiation model.

11.13.1. Absorption Coefficient Model

Scope: Radiation Transport Equation Model Specification

`Absorption Coefficient Model {=|are|is}` *`AbsorpCoeffModel`*

Parameter	Value	Default
AbsorpCoeffModel	`{leckner}`	`LECKNER`

Summary Specify absorption coefficient model

Description This option will allow for the specification of an absorption coefficient model. At present, only the Leckner model is supported and is, therefore, the default. Specification of the model requires designation of water and carbon dioxide string names.

11.13.2. Aluminum_Oxide Emittance Model

Scope: Radiation Transport Equation Model Specification

`Aluminum_Oxide Emittance Model {=|are|is}` *`AluminaEmittModel`*

Parameter	Value	Default
AluminaEmittModel	`{brewster|konopka|table}`	`brewster`

Summary Specify alumina emittance model

Description This option allows for the specification of an absorption coefficient model for alumina. Konopka and Table (user-specified) are also available. For the table model, the user must have T-ABS-MODEL.txt file in the current directory with the following format:

0.000094

4

300.0 0.0821

1000.0 0.1064

2320.0 0.0024

10000.0 0.7435

Where:

0.000094 is DTHTWO (Characteristic diameter of Al_2O_3 Units: [cm]). 4 is the number of points in the linear interpolation absorption model (Absorption as function of temperature). The remaining lines are T, ABS points for the model. Linear interpolation is used between the points. For T less than T_0 (lowest temperature for these points), ABS = ABS(T_0). For T greater than T_0 (highest temperature for these points), ABS = ABS(T_0).

Brewster and Konopka options are equivalent to Table option with:

Konopka:

0.000094

4

300 0.0821

1000 0.10639

2320 0.002384

2320.001 0.04 (at $T = 2320$, this represents a discontinuous change in absorption coefficient)

Brewster:

0.000094

4

300 0.0821

1000 0.10639

2320 0.002384

10000 0.74352 ($T = 10000$ is set high enough that particle temperature is not likely to be this high)

11.13.3. Characteristic Length Scale For Absorption Coefficient Determination

Scope: Radiation Transport Equation Model Specification

`Characteristic Length Scale For Absorption Coefficient Determination {=|are|is}` *`CharacteristicLength`*

Parameter	Value	Default
CharacteristicLength	`real`	-

Summary Override the absorption coefficient normally calculated by fuego with this value.

11.13.4. Species Name For

Scope: Radiation Transport Equation Model Specification

`Species Name For` *`AbsorpCoeffSpecies`* `{=|are|is}` *`SpeciesName`*

Parameter	Value	Default
AbsorpCoeffSpecies	`{aluminum_oxide|carbon_dioxide|hydrogen_chloride|soot|water}`	-
SpeciesName	`string`	-

Summary Specify string names that map to water, carbon dioxide, hydrogen chloride or aluminum oxide (AKA alumina)

Description This option will allow for the specification of an absorption coefficient model specification. At present, only the Leckner model is supported for water and carbon dioxide. Hydrogen chloride and aluminum oxide (alumina) provided by tables.

11.13.5. Subgrid Mixing Model

Scope: Radiation Transport Equation Model Specification

`Subgrid Mixing Model {=|are|is}` *`RTESubgridModel`*

Parameter	Value	Default
RTESubgridModel	`{no_subgrid_mixing}`	`no_subgrid_mixing`

Summary Specify subgrid mixing model for RTE source term

Description This option will allow for the specification of subgrid mixing model. At present, there is no model supported other than to ignore fluctuation effects, which is the default.

11.14. TIME INTEGRATION SPECIFICATION

Scope: Solution Options

```
Begin Time Integration Specification TimeIntSpecName

    Activate Bdf2 [ After Step BDF2StartStep ]

    Mass Matrix {=|are|is} MassMatrixType

    Predictor Algorithm {=|are|is} PredictorAlgorithm

    Use Second Order Implicit Time Integration [ With Blending
    Coefficient {=|are|is} BlendCoeff ]

End
```

Summary Specify time integration options. Currently, either Crank-Nicolson through blending or BDF2.

11.14.1. Activate Bdf2

Scope: Time Integration Specification

Summary Activation of the second-order BDF2 time integrator.

Description BDF2 is a A-stable, three state time integrator that has been demonstrated to be well performing in the low-Mach application space. When activated, all PDEs with time terms will use BDF2. The user specification for MASS MATRIX is used to determine the usage of the lumped or consistent form.

Use the optional argument to delay the use of BDF2 for some number of time steps. Backward euler will be used for those time steps instead.

11.14.2. Mass Matrix

Scope: Time Integration Specification

`Mass Matrix {=|are|is}` *`MassMatrixType`*

Parameter	Value	Default
MassMatrixType	`{consistent|lumped}`	`LUMPED`

Summary Specifies if the mass matrix (also known as the capacitance matrix) is to be lumped using row sum rule.

Description Using a lumped mass matrix diagonalizes the mass matrix contribution to the linear system, and therefore results in a more diagonally-dominant matrix. This under-integration of the mass matrix may result in more monotonic behavior for certain problems.

11.14.3. Predictor Algorithm

Scope: Time Integration Specification

`Predictor Algorithm {=|are|is}` *`PredictorAlgorithm`*

Parameter	Value	Default
PredictorAlgorithm	`{adams_bashforth|forward_euler|simple}`	-

Summary Specification of predictor algorithm

Description Three predictor algorithms are supported, i.e.,

- Simple Predictor: $\phi^{n+1} = \phi^n$
- Forward Euler: $\phi^{n+1} = \phi^n + \Delta t \dot{\phi}$
- Adams Bashforth: $\phi^{n+1} = \phi^n + \frac{\Delta t}{2}(3\dot{\phi}^n - \dot{\phi}^{n-1})$.

11.14.4. Use Second Order Implicit Time Integration

Scope: Time Integration Specification

Summary Allow for higher order time integration method

Description This option will allow for second order time integration to be activated within Fuego. The blending coefficient is optional however, defaults to 2, i.e., trapezoid rule. The user may elect to choose a blending coefficient between 1 and 2. values of 1 result in pure backward Euler. All transport equations must share the same integration order.

The current implementation is within the context of a fully implicit solve. Therefore, Picard iteration is required.

11.15. TURBULENCE MODEL SPECIFICATION

Scope: Solution Options

```
Begin Turbulence Model Specification TurbSpecName
    Activate Deris Flaming Buoyancy Source Term
    Activate Deris Nonflaming Buoyancy Source Term
    Activate Rodi Buoyancy Source Term
    Activate Rodi Density Buoyancy Source Term
    Determine Utau Via Nonlinear Law Of The Wall Iteration
    Include Molecular Viscosity In K-E Diffusion Coefficient
    Limit Turbulent Ke Production To Value Times Dissipation
    Omit Low Reynolds Turbulence Dissipation Source Term
    Omit Near Wall Turbulent Ke Transport Equation
    Omit Turbulence Source Terms On Block(S) BlockList...
    Omit Velocity Divergence In Turbulent Production Term
    Time Filter {=|are|is} Value
    Turbulence Model {=|are|is} TurbModel
```

```
    Turbulence Model Parameter TurbulenceModelParams {=|are|is} Value

    Turbulence Postprocessor {=|are|is} TurbPP
    Under Relax Turbulent_Viscosity By Urf
    Use Buoyant Vorticity Generation At Value Seconds
    Use Dynamic Ksgs
    Wall Distance Band Size {=|are|is} BandSize
    Wall Model {=|are|is} WallModel
End
```

Summary Specify turbulence modeling options.

11.15.1. Activate Deris Flaming Buoyancy Source Term

Scope: Turbulence Model Specification

Summary Add de Ris' flaming buoyancy source term to turbulence transport equations.

Description This option adds the buoyancy source term of Rodi to the turbulent kinetic energy equation (for all turbulence models) and to some extent, to the dissipation rate equation. Two versions - flaming and non-flaming - are proposed in the paper. The source term of the flaming version is given by $C_{deris}(\rho_F - \rho)g_j k^{0.5}$ where C_{deris} is a user-defined coefficient. The coefficient is set 0.01 for now without sufficient valication efforts.

Reference: JL de Ris, Procedia Engineering 62(2013)13-27

11.15.2. Activate Deris Nonflaming Buoyancy Source Term

Scope: Turbulence Model Specification

Summary Add de Ris' non-flaming buoyancy source term to turbulence transport equations.

Description This option adds the buoyancy source term of Rodi to the turbulent kinetic energy equation (for all turbulence models) and to some extent, to the dissipation rate equation. Non-flaming version is a combination of BVG (Buoyant Vorticity Generation) and Rodi models. The source term is given by $C_{deris}\Delta k^{0.5}(|\nabla\rho \times g| - \nabla\rho \cdot g)$ where C_{deris} is a user-defined coefficient. The coefficient is set 0.01 for now without sufficient valication efforts.

Reference: JL de Ris, Procedia Engineering 62(2013)13-27

11.15.3. Activate Rodi Buoyancy Source Term

Scope: Turbulence Model Specification

Summary Add Rodi's buoyancy source term to turbulence transport equations.

Description This option adds the buoyancy source term of Rodi to the turbulent kinetic energy equation (for all turbulence models) and in some cases, the dissipation rate equation. The source term is given by, $G_B = \beta \frac{\mu_t}{Pr_t} \frac{\partial T}{\partial x_j} g_j$. For the turbulent kinetic energy, G_B augments the right hand side while for the dissipation rate equation, $C'_{\epsilon 1} C_{\epsilon 4} \frac{1}{T} G_B$ is the augmented right hand side term. The source term is limited to a fraction of the dissipation rate.

This model has not been validated for use in the context of turbulence models other than the standard k-ϵ model.

11.15.4. Activate Rodi Density Buoyancy Source Term

Scope: Turbulence Model Specification

Summary Add Rodi's buoyancy source term to turbulence transport equations, modified to be density based

Description This option adds the buoyancy source term of Rodi to the turbulent kinetic energy equation (for all turbulence models) and in some cases, the dissipation rate equation. The source term is given by, $G_B = \rho_{ref} \frac{\mu_t}{Pr_t} \frac{\partial (1/\rho)}{\partial x_j} g_j$. For the turbulent kinetic energy, G_B augments the right hand side while for the dissipation rate equation, $C'_{\epsilon 1} C_{\epsilon 4} \frac{1}{T} G_B$ is the augmented right hand side term. The source term is limited to a fraction of the dissipation rate.

This model is equivalent to the standard rodi model for single component, thermal plumes.

11.15.5. Determine Utau Via Nonlinear Law Of The Wall Iteration

Scope: Turbulence Model Specification

Summary Use law of the wall nonlinear iteration for utau calculation

Description Elect to perform a nonlinear iteration of the law-of-the-wall formula to calculate utau, the wall friction velocity at the boundary integration points. This value will be used to calculate the wall shear stress when using the standard k-e wall function model. Alternatively, the friction velocity will be calculated from the nodal value of turbulent k.e.

11.15.6. Include Molecular Viscosity In K-E Diffusion Coefficient

Scope: Turbulence Model Specification

Summary Augment diffusion coefficient in turbulence equations via molecular viscosity

Description The standard k-e model does not include the molecular viscosity in the diffusion term. This option adds the molecular viscosity to tvisc/sigma_i. The line command is also appropriate for the Ksgs and v2-f model.

11.15.7. Limit Turbulent Ke Production

Scope: Turbulence Model Specification

`Limit Turbulent Ke Production To` *`Value`* `Times Dissipation`

Parameter	Value	Default
Value	`real`	-

Summary Choose to limit production source terms

Description This option limits the turbulent ke production to a scale factor of dissipation, prod = min(prod, limit*den*en1). In practice, the ratio of production to dissipation is not very high. In some flows, it is useful to specify a value of approximately 1000. The ratio should be checked as part of the analysis to make sure that violation of the physical ratio has not been done. In general, this option is only activated in domain locations where dissipation rate is very small. (Default: 1.0e8)

11.15.8. Omit Low Reynolds Turbulence Dissipation Source Term

Scope: Turbulence Model Specification

Summary Remove low Reynolds source term from dissipation rate equation.

Description This option removes the low Reynolds number source term from the dissipation rate equation given by Launder and Sharma. The form is $2 * \nu * \mu_t * \left(\frac{\partial^2 u_i}{\partial x_k \partial x_j} \frac{\partial^2 u_i}{\partial x_k \partial x_j} \right)$.

This source term is calculated using a two pass shape function evaluation. The first pass calculates the standard derivatives, $\partial u_i / \partial x_j$. The second pass assumes a piecewise constant interpolation of the standard derivatives from the sub control volume to the nodes. These interpolated values are used in the shape function derivative loop to calculate the second derivatives.

The default is to include the source term while the specification of it is placed as a temporary field, i.e., can be changed as restart.

11.15.9. Omit Near Wall Turbulent Ke Transport Equation

Scope: Turbulence Model Specification

Summary Do not construct a transport equation for the near wall turbulent kinetic energy equation.

Description Rather than solving a transport equation for the near wall turbulent transport equation, assign a Dirichlet condition based on local equilibrium argument between production and dissipation that yields kwall = utau*utau/sqrt(Cmu). This formulation neglects convection and diffusion effects for the near wall equation.

11.15.10. Omit Turbulence Source Terms On Block(S)

Scope: Turbulence Model Specification

`Omit Turbulence Source Terms On Block(S)` *`BlockList...`*

Parameter	Value	Default
BlockList	`string...`	-

Summary Omit source terms on these sets of I/O block; specified by name.

Description This line command will result in all turbulence source terms to be zeroed in the respective blocks. We can not yet dial in exact equation sets. Example usage: OMIT TURBULENCE SOURCE TERMS ON BLOCK(S) block_1 block_2.

Note that the source terms for output will still exist and, in general, be non-zero within the skipped block. However, the contribution to the equation system will be omitted.

11.15.11. Omit Velocity Divergence In Turbulent Production Term

Scope: Turbulence Model Specification

Summary Do not include divergence term in turbulence production

Description This option removes the divergence term from the turbulent production of kinetic energy. The default is to include this term.

11.15.12. Time Filter

Scope: Turbulence Model Specification

`Time Filter {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`1.0e32`

Summary Time filter size for Time Filtered Navier-Stokes model

Description Turbulent viscosity is normally calculated based on a time scale given by k/epsilon (for k-epsilon models) or T (v2f model). The TFNS model substitutes the minimum of the normal computed value and the user-specified time filter size in the turbulent viscosity calculation. In general, this filter should be no less than twice the physical time step. A non-fatal warning is issued if this condition is violated. (Default: 1.0e32 to essentially deactivate the model)

11.15.13. Turbulence Model

Scope: Turbulence Model Specification

`Turbulence Model {=|are|is}` *`TurbModel`*

Parameter	Value	Default
TurbModel	`{dsmag|ke|ksgs|kw|lam|lrke|lrkw|lrsst|rng|smag|sst|sstdes|v2f}`	`LAM`

Summary Specify type of turbulence model to be used

Description Fuego provides a broad set of turbulence models which includes both RANS and LES models. For more information regarding specific models, please see the theory documentation. One note of importance is that the SST, SSTDES, and LRSST models require the `min_wall_distance` field, which is a measure of the nearest distance to a wall. Fuego will automatically use Sierra/Krino library to generate this field when using these turbulence models.

11.15.14. Turbulence Model Parameter

Scope: Turbulence Model Specification

`Turbulence Model Parameter` *`TurbulenceModelParams`* `{=|are|is}` *`Value`*

Parameter	Value	Default
TurbulenceModelParams	`{frequency_scaling|kappa|minimum_k|yplus_crit}`	-
Value	`real`	-

Summary Specify turbulence model parameters

11.15.15. Turbulence Postprocessor

Scope: Turbulence Model Specification

`Turbulence Postprocessor {=|are|is}` *`TurbPP`*

Parameter	Value	Default
TurbPP	`{q_criterion|residual_reynolds_stress RESOLVED_REYNOLDS_STRESS |resolved_reynolds_stress}`	

Summary Specify a turbulence postprocessor

Description A set of post processors for turbulence quantities. Options include RESOLVED_REYNOLDS_STRESS, RESIDUAL_REYNOLDS_STRESS, and Q_CRITERION

11.15.16. Under Relax Turbulent_Viscosity By

Scope: Turbulence Model Specification

`Under Relax Turbulent_Viscosity By` *`Urf`*

Parameter	Value	Default
Urf	`real`	-

Summary Under relaxation factor for the turbulent viscosity

11.15.17. Use Buoyant Vorticity Generation

Scope: Turbulence Model Specification

`Use Buoyant Vorticity Generation At` *`Value`* `Seconds`

Parameter	Value	Default
Value	`real`	-

Summary Use the BVG model of Nicollete and Tieszen

Description Invoke the buoyant vorticity generation augmentation to the turbulent kinetic energy production term. Allow for a delay in the activation to avoid potential instabilities. The delay will will be only in assembling the source term to the appropriate partial differential equation. However, the source term will be computed at all times when the model is activated, and will be limited to a fraction of the dissipation rate.

This model has not been validated for use in the context of turbulence models other than the standard k-ϵ model. The model uses the turbulence model parameters C_{bvg} and $C_{\epsilon 3}$.

11.15.18. Use Dynamic Ksgs

Scope: Turbulence Model Specification

Summary Compute model coefficients dynamically for KSGS turbulent model

Description With the dynamic option, two model coefficients (Cmu_eps and Ceps) in the KSGS one-equation LES model are computed dynamically. This line command will be recognized only when KSGS turbulent model is used.

11.15.19. Wall Distance Band Size

Scope: Turbulence Model Specification

`Wall Distance Band Size {=|are|is}` *`BandSize`*

Parameter	Value	Default
BandSize	`real`	-

Summary Band size for the normal distance to the wall calculation. Set to 10x the max face element size if omitted.

11.15.20. Wall Model

Scope: Turbulence Model Specification

`Wall Model {=|are|is}` *`WallModel`*

Parameter	Value	Default
WallModel	`{modeled|resolved}`	-

Summary Specify wall model to be used

Description Options are MODELED and RESOLVED. Current default depends on which turbulence model is in use.

This is a beta feature, not all combinations are valid.

11.16. ACOUSTIC TRANSFER OUTPUT

Scope: Solution Options

```
Begin Acoustic Transfer Output DefinitionName

    At Step n Option {=|are|is} m

    At Time Dt1 Option {=|are|is} Dt2

    Force Search In Model Coordinates

    Input Mesh Name {=|are|is} MeshName

    Output Mesh Name {=|are|is} MeshName

    Overlap Drop Tolerance {=|are|is} DropTol

    Send Block From_blocks...  To To_blocks...

    Source Vector Name {=|are|is} SrcName [ Scaling {=|are|is}
    ScalingValue ]

    Timestep Adjustment Interval {=|are|is} Nsteps

End
```

Summary Specify transfer output of the divergence of the Lighthill tensor to a different mesh for use in acoustic simulations.

11.16.1. At Step

Scope: Acoustic Transfer Output

```
At Step n Option {=|are|is} m
```

Parameter	Value	Default
n	integer	-
Option	{increment|interval}	-
m	integer	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

11.16.2. At Time

Scope: Acoustic Transfer Output

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

11.16.3. Force Search In Model Coordinates

Scope: Acoustic Transfer Output

Summary By default the geometric search will be repeated in the current coordinates at each time step for problems with mesh motion. Adding this line command will force the search to be done once at the start of the problem in the model coordinates.

11.16.4. Input Mesh Name

Scope: Acoustic Transfer Output

`Input Mesh Name {=|are|is}` *`MeshName`*

Parameter	Value	Default
MeshName	`string`	-

Summary Specify the input mesh that will be the destination of the transfer.

11.16.5. Output Mesh Name

Scope: Acoustic Transfer Output

`Output Mesh Name {=|are|is}` *`MeshName`*

Parameter	Value	Default
MeshName	`string`	-

Summary Specify where to output the result of the transfer.

Description The divergence of the Lighthill tensor will be output as divT

11.16.6. Overlap Drop Tolerance

Scope: Acoustic Transfer Output

`Overlap Drop Tolerance {=|are|is}` *`DropTol`*

Parameter	Value	Default
DropTol	`real`	-

Summary Specify an overlap drop tolerance.

Description Specify a volume fraction to ignore overlaps below. This does not affect conservation, if you set a tolerance of 5one acoustic element by 97term would be sent to the single acoustic element. This can reduce the cost and memory use of the acoustic transfer.

11.16.7. Send Block

Scope: Acoustic Transfer Output

`Send Block` *`From_blocks...`* `To` *`To_blocks...`*

Parameter	Value	Default
From_blocks	`string...`	-
To_blocks	`string...`	-

Summary The acoustic transfer will define one transfer operation per SEND BLOCK line, but can define many from/to blocks per line. If there is a block with mesh motion in the send block list it will update that transfer every time step unless you have forced the search to be done in model coordinates. If you have some moving blocks and some fixed blocks it may be faster to split the moving block into its own transfer as long as this is geometrically reasonable.

SEND BLOCK block_3 block_5 block_6 TO block_3 block_5 SEND BLOCK block_7 TO block_1

11.16.8. Source Vector Name

Scope: Acoustic Transfer Output

`Source Vector Name {=|are|is}` *`SrcName`* `[ Scaling {=|are|is}` *`ScalingValue`* `]`

Parameter	Value	Default
SrcName	`string`	-

Summary Specify the noise source vector name.

Description The divergence of the Lighthill tensor will be output as divT. This specifies the name of the source vector to transfer. By default this is "div_lighthill_tensor" but if you are using a filter or averaging you should set the name here to the name of the filtered vector.

11.16.9. Timestep Adjustment Interval

Scope: Acoustic Transfer Output

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	integer	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

11.17. POINT SOURCE

Scope: Solution Options

```
Begin Point Source DefinitionName

    PrimitiveVariable {=|are|is} Value

    Location {=|are|is} LocationType Location1 Location2 Location3 [
    Radius {=|are|is} RadiusValue ]

    Mdot {=|are|is} Mdot

    Mass_Fraction Species {=|are|is} Mass fraction

    Progress_Variable ProgressVariableName {=|are|is} Value

End
```

Summary Specify a point source.

11.17.1.

Scope: Point Source

`PrimitiveVariable` `{=|are|is}` *`Value`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}`	-
Value	`"string"`	-

Summary Assign inflow condition to the specified variable. Specified value can be a constant or a string function of time or global variables. Omitted properties will default to the reference value, and if no reference value is present will default to 0.

11.17.2. Location

Scope: Point Source

`Location {= | are | is}` *`LocationType Location`*$_1$ *`Location`*$_2$ *`Location`*$_3$ `[ Radius {= | are | is}` *`RadiusValue`* `]`

Parameter	Value	Default
LocationType	`{node nearest | sphere at}`	-
Location	`real_1 real_2 real_3`	-

Summary Specify the location of the point source. The source will be applied at the node closest to the specified location. If mesh motion is active, the node will be selected only once at the initial mesh coordinates.

The source location can either be the node nearest to a point, or a sphere where the source is applied evenly on all inclosed nodes. If the sphere does not intersect any nodes, the closest node is used.

11.17.3. Mdot

Scope: Point Source

`Mdot {= | are | is}` *`Mdot`*

Parameter	Value	Default
Mdot	`"string"`	-

Summary Specify the mass flow rate of the point source. Can be a constant or string function of time and global variables. This should be in mass/time units (e.g. kg/s). Positive (inflow) and negative (outflow) values are both allowed.

11.17.4. Mass_Fraction

Scope: Point Source

`Mass_Fraction` *`Species`* `{=|are|is}` *`Mass fraction`*

Parameter	Value	Default
Species	`string`	-
Mass fraction	`"string"`	-

Summary Set the source mass fraction. Specified value can be a constant or a string function of time or global variables. Omitted mass fractions will default to 0.

11.17.5. Progress_Variable

Scope: Point Source

`Progress_Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`"string"`	-

Summary Set the source progress variable. Specified value can be a constant or a string function of time or global variables.

11.18. MESH MOTION SPECIFICATION

Scope: Solution Options

```
Begin Mesh Motion Specification DefinitionName

    Activate Bdf2 For Mesh Velocity

    Mesh Motion Model On BlockName {=|are|is} MeshMotionModel [
    DisplacementArgs...  ]

    Begin Rigid Body Motion Model DefinitionName

    End
```

```
End
```

Summary Specify mesh motion parameters.

11.18.1. Activate Bdf2 For Mesh Velocity

Scope: Mesh Motion Specification

Summary Activation of the second-order BDF2 time integrator for mesh velocity.

Description BDF2 is a A-stable, three state time integrator that has been demonstrated to be well performing in the low-Mach application space. When activated, the mesh velocity calculated from displacements will use BDF2. This is enabled automatically if the overall time integration is set to use BDF2.

11.18.2. Mesh Motion Model On

Scope: Mesh Motion Specification

`Mesh Motion Model On` *`BlockName`* `{=|are|is}` *`MeshMotionModel`* `[` *`DisplacementArgs...`* `]`

Parameter	Value	Default
BlockName	`string`	-
MeshMotionModel	`{fixed_const_velocity|none|rigid_body|rotation|specified|transferred}`	-

Summary Specify the mesh motion model to use.

Description Select the mesh motion model (SPECIFIED, FIXED_CONST_VELOCITY, TRANSFERRED, ROTATION, or NONE) along with any additional model arguments.

The 'SPECIFIED' model takes either a set of specified displacements that are string functions of t, x, y, and z, or a specified set of accelerations that are functions of t and any global variables. In the latter mode, an initial velocity, vo, can also be specified (default: 0).

The 'TRANSFERRED' and 'NONE' options do not take any arguments.

The 'FIXED_CONST_VELOCITY' lets you apply motion to a block without actually moving it in the model. This is typically applied to shells, for example to move the top fluid in a lid-driven cavity flow without actually moving the mesh, or to apply motion on a quarter of a circle without actually moving it.

The 'ROTATION' model takes a time function for the angle (theta) in rotations (radius/2*pi or degrees/360), and an origin and rotation axis (only for 3D). For example, a

rotation of 1 rpm would use 'theta=t/60'. The rotation uses the right hand rule about the specified axis and origin. Rotation can also be specified by providing an angular acceleration term that can be a function of time (t) or global variables. An optional initial angular velocity (omegao) can also be specified (default: 0). Both the angular acceleration and angular velocity are specified in revolutions, so they are multiplied internally by 2*pi to convert to radians.

Any blocks that don't have a mesh motion model specified get no displacement. Some example syntax is shown below. You can use the "all_blocks" keyword to specify all blocks, but it cannot be mixed with per-block specifications.

```
MESH MOTION MODEL ON block_1    = SPECIFIED D = 0.1*t y*z sin(t)
MESH MOTION MODEL ON block_1    = SPECIFIED v0 = 0.1 0 0 a=0 0.1 0
MESH MOTION MODEL ON all_blocks = TRANSFERRED
MESH MOTION MODEL ON block_3    = ROTATION  origin=0 0 0 \$
    axis=1 0 0 theta=t/60
MESH MOTION MODEL ON block_3    = ROTATION  origin=0 0 0 \$
    axis=1 0 0 omega0=1/60 alpha=0
MESH MOTION MODEL ON block_4    = NONE
MESH MOTION MODEL ON block_6    = FIXED_CONST_VELOCITY \$
    Velocity=0.1 0*t "0 + 4*x*y"
```

The mesh motion model also generates global variables of its internal state (position, velocity, and acceleration) which can be used in the functional forms. These global variables are output in units of revolutions for consistency with the input units. For example, to smoothly accelerate up to a constant angular velocity of 10 rev/s, you could use:

```
MESH MOTION MODEL ON block_3 = ROTATION  origin=0 0 0 \$
    axis=1 0 0 alpha=10-block_3_omega
```

11.19. RIGID BODY MOTION MODEL

Scope: Mesh Motion Specification

Begin Rigid Body Motion Model *DefinitionName*

Acceleration {=|are|is} $Acceleration_1$ $Acceleration_2$ $Acceleration_3$

Blocks {=|are|is} *BlockNames...*

Buoyant Object Mass {=|are|is} *ObjectMass...*

Buoyant Object Pitch Moment {=|are|is} *Moment*

Buoyant Object Roll Moment {=|are|is} *Moment*

Buoyant Object Surfaces {=|are|is} *ObjSurfaces...*

```
    Buoyant Object Yaw Moment {=|are|is} Moment

    Initial Velocity {=|are|is} InitialVelocity_1 InitialVelocity_2
    InitialVelocity_3

    Origin {=|are|is} OriginCoords_1 OriginCoords_2 OriginCoords_3

    Pitch Acceleration {=|are|is} PitchAcceleration

    Pitch Initial Velocity {=|are|is} PitchInitialVelocity

    Roll Acceleration {=|are|is} RollAcceleration

    Roll Axis {=|are|is} RollAxis_1 RollAxis_2 RollAxis_3

    Roll Initial Velocity {=|are|is} RollInitialVelocity

    Under Relaxation {=|are|is} UnderRelaxation

    Yaw Acceleration {=|are|is} YawAcceleration

    Yaw Axis {=|are|is} YawAxis_1 YawAxis_2 YawAxis_3

    Yaw Initial Velocity {=|are|is} YawInitialVelocity

End
```

Summary Specify parameters for 6-DOF rigid body motion.

11.19.1. Acceleration

Scope: Rigid Body Motion Model

`Acceleration {=|are|is}` *Acceleration*$_1$ *Acceleration*$_2$ *Acceleration*$_3$

Parameter	Value	Default
Acceleration	"string"_1 "string"_2 "string"_3	-

Summary Specify the translational acceleration

Description Specify the translational acceleration using three quoted string functions. Functions can include time and any global variables as valid variables.

11.19.2. Blocks

Scope: Rigid Body Motion Model

`Blocks {=|are|is}` *`BlockNames...`*

Parameter	Value	Default
BlockNames	`string...`	-

Summary Specify which blocks to apply this model to.

Description Give a list of blocks to apply the model to. The all_blocks command is permitted here too.

11.19.3. Buoyant Object Mass

Scope: Rigid Body Motion Model

`Buoyant Object Mass {=|are|is}` *`ObjectMass...`*

Parameter	Value	Default
ObjectMass	`real...`	-

Summary Specify the buoyant object mass

Description Specify the buoyant object mass. Optionally, you can specify three values for mass to act on the three translation directions. This is primarily useful for supressing motion in a given direction by setting that mass to a very large number. For example, to only allow motion of a 50 kg object in the z-direction you could set the mass to:

```
BUOYANT OBJECT MASS = 1e30 1e30 50
```

11.19.4. Buoyant Object Pitch Moment

Scope: Rigid Body Motion Model

`Buoyant Object Pitch Moment {=|are|is}` *`Moment`*

Parameter	Value	Default
Moment	`real`	-

Summary Specify the buoyant object pitch moment

Description Specify the buoyant object moment of inertia about pitch axis. Set to a very large number to disable pitch.

11.19.5. Buoyant Object Roll Moment

Scope: Rigid Body Motion Model

`Buoyant Object Roll Moment {=|are|is}` *`Moment`*

Parameter	Value	Default
Moment	`real`	-

Summary Specify the buoyant object roll moment

Description Specify the buoyant object moment of inertia about roll axis. Set to a very large number to disable roll.

11.19.6. Buoyant Object Surfaces

Scope: Rigid Body Motion Model

`Buoyant Object Surfaces {=|are|is}` *`ObjSurfaces...`*

Parameter	Value	Default
ObjSurfaces	`string...`	-

Summary Specify surfaces of a buoyant object.

Description Specify surfaces of a buoyant object.

11.19.7. Buoyant Object Yaw Moment

Scope: Rigid Body Motion Model

`Buoyant Object Yaw Moment {=|are|is}` *`Moment`*

Parameter	Value	Default
Moment	`real`	-

Summary Specify the buoyant object yaw moment

Description Specify the buoyant object moment of inertia about yaw axis. Set to a very large number to disable yaw.

11.19.8. Initial Velocity

Scope: Rigid Body Motion Model

`Initial Velocity {=|are|is}` *InitialVelocity*$_1$ *InitialVelocity*$_2$ *InitialVelocity*$_3$

Parameter	Value	Default
InitialVelocity	`"string"_1 "string"_2 "string"_3`	-

Summary Specify the translational initial velocity

Description Specify the translational initial velocity using three quoted string functions. Functions can include time and any global variables as valid variables.

11.19.9. Origin

Scope: Rigid Body Motion Model

`Origin {=|are|is}` *OriginCoords*$_1$ *OriginCoords*$_2$ *OriginCoords*$_3$

Parameter	Value	Default
OriginCoords	`real_1 real_2 real_3`	-

Summary Specify the origin for the motion.

Description The origin coordinates are those around which rotation occurs. This would typically be the center of mass.

11.19.10. Pitch Acceleration

Scope: Rigid Body Motion Model

`Pitch Acceleration {=|are|is}` *PitchAcceleration*

Parameter	Value	Default
PitchAcceleration	`"string"`	-

Summary Specify the pitch acceleration

Description Specify the pitch acceleration using a quoted string function. Functions can include time and any global variables as valid variables. Units are revolutions, not radians.

11.19.11. Pitch Initial Velocity

Scope: Rigid Body Motion Model

`Pitch Initial Velocity {=|are|is}` *`PitchInitialVelocity`*

Parameter	Value	Default
PitchInitialVelocity	`"string"`	-

Summary Specify the pitch initial velocity

Description Specify the pitch initial velocity using a quoted string function. Functions can include time and any global variables as valid variables. Units are revolutions, not radians.

11.19.12. Roll Acceleration

Scope: Rigid Body Motion Model

`Roll Acceleration {=|are|is}` *`RollAcceleration`*

Parameter	Value	Default
RollAcceleration	`"string"`	-

Summary Specify the roll acceleration

Description Specify the roll acceleration using a quoted string function. Functions can include time and any global variables as valid variables. Units are revolutions, not radians.

11.19.13. Roll Axis

Scope: Rigid Body Motion Model

`Roll Axis {=|are|is}` $RollAxis_1$ $RollAxis_2$ $RollAxis_3$

Parameter	Value	Default
RollAxis	`real_1 real_2 real_3`	-

Summary Specify the roll axis for rotation.

Description Rotation is specified using pitch, roll, and yaw.

Roll is rotation about the roll axis. Yaw is rotation about the yaw axis. The pitch axis is defined by the cross product of the roll and yaw axes.

Rotation is applied as roll, then pitch, then yaw. The axes about which these translations happen are fixed.

The roll and yaw axes must be orthogonal. These do not need to be unit vectors, they will be normalized internally.

11.19.14. Roll Initial Velocity

Scope: Rigid Body Motion Model

`Roll Initial Velocity {=|are|is}` *`RollInitialVelocity`*

Parameter	Value	Default
RollInitialVelocity	`"string"`	-

Summary Specify the roll initial velocity

Description Specify the roll initial velocity using a quoted string function. Functions can include time and any global variables as valid variables. Units are revolutions, not radians.

11.19.15. Under Relaxation

Scope: Rigid Body Motion Model

`Under Relaxation {=|are|is}` *`UnderRelaxation`*

Parameter	Value	Default
UnderRelaxation	`real`	`1`

Summary Specify under-relaxation for the equations of motion

Description Specify under-relaxation for the equations of motion

11.19.16. Yaw Acceleration

Scope: Rigid Body Motion Model

`Yaw Acceleration {=|are|is}` *`YawAcceleration`*

Parameter	Value	Default
YawAcceleration	`"string"`	-

Summary Specify the yaw acceleration

Description Specify the yaw acceleration using a quoted string function. Functions can include time and any global variables as valid variables. Units are revolutions, not radians.

11.19.17. Yaw Axis

Scope: Rigid Body Motion Model

`Yaw Axis {=|are|is}` *YawAxis*$_1$ *YawAxis*$_2$ *YawAxis*$_3$

Parameter	Value	Default
YawAxis	`real_1 real_2 real_3`	-

Summary Specify the yaw axis for rotation.

Description Rotation is specified using pitch, roll, and yaw.

Roll is rotation about the roll axis. Yaw is rotation about the yaw axis. The pitch axis is defined by the cross product of the roll and yaw axes.

Rotation is applied as roll, then pitch, then yaw. The axes about which these translations happen are fixed.

The roll and yaw axes must be orthogonal. These do not need to be unit vectors, they will be normalized internally.

11.19.18. Yaw Initial Velocity

Scope: Rigid Body Motion Model

`Yaw Initial Velocity {=|are|is}` *YawInitialVelocity*

Parameter	Value	Default
YawInitialVelocity	`"string"`	-

Summary Specify the yaw initial velocity

Description Specify the yaw initial velocity using a quoted string function. Functions can include time and any global variables as valid variables. Units are revolutions, not radians.

11.20. VOF MODEL SPECIFICATION

Scope: Solution Options

`Begin Vof Model Specification` *DefinitionName*

`Interface Compression Factor {=|are|is}` *VOFInterfaceCompressionFactor*

`Interface Compression Model {=|are|is}` *VOFInterfaceCompressionModel*

`Interface Curvature Algorithm {=|are|is}` *VOFSmootherAlgorithm*

`Interface Location Tolerance {=|are|is}` *InterfaceTol*

```
    Interface Sharpening Model {=|are|is} VOFInterfaceSharpeningModel

    Num Smoother Iterations {=|are|is} NumSmootherIters

    Phase Change Model {=|are|is} VOFPhaseChangeModel [ModelArgs]...

    Post Process Interface Normal

    Smoother Fourier Number {=|are|is} SmootherFo

End
```

Summary Specify a number of physical parameters that are to be used for multi-phase VOF flow simulations.

11.20.1. Interface Compression Factor

Scope: Vof Model Specification

```
Interface Compression Factor {=|are|is} VOFInterfaceCompressionFactor
```

Parameter	Value	Default
VOFInterfaceCompressionFactor	`real`	-

Summary Specify parameters for interface compression model for VOF.

Description Select the VOF interface compression factor (only used in the Constant model)

11.20.2. Interface Compression Model

Scope: Vof Model Specification

```
Interface Compression Model {=|are|is} VOFInterfaceCompressionModel
```

Parameter	Value	Default
VOFInterfaceCompressionModel	`{adaptive|constant|none}`	-

Summary Specify parameters for interface compression model for VOF.

Description Select the VOF interface compression model (Constant, Adaptive, or None)

11.20.3. Interface Curvature Algorithm

Scope: Vof Model Specification

`Interface Curvature Algorithm {=|are|is}` *`VOFSmootherAlgorithm`*

Parameter	Value	Default
VOFSmootherAlgorithm	`{diffusive|level_set|none}`	-

Summary Specify the curvature model to use.

Description Select the VOF curvature model (DIFFUSIVE, LEVEL_SET, or NONE)

11.20.4. Interface Location Tolerance

Scope: Vof Model Specification

`Interface Location Tolerance {=|are|is}` *`InterfaceTol`*

Parameter	Value	Default
InterfaceTol	`real`	-

Summary Specify parameters for interface location for VOF.

Description Select the tolerance used to identify the interface zone

11.20.5. Interface Sharpening Model

Scope: Vof Model Specification

`Interface Sharpening Model {=|are|is}` *`VOFInterfaceSharpeningModel`*

Parameter	Value	Default
VOFInterfaceSharpeningModel	`{non_conserving|none|volume_conserving}`	-

Summary Specify parameters for interface sharpening model for VOF.

Description Select the VOF interface sharpening model (Volume_Conserving or None)

11.20.6. Num Smoother Iterations

Scope: Vof Model Specification

`Num Smoother Iterations {=|are|is}` *`NumSmootherIters`*

Parameter	Value	Default
NumSmootherIters	`integer`	-

Summary Specify parameters for interface smoothing for VOF.

Description Select the number of smoother iterations for the interface normal calculation

11.20.7. Phase Change Model

Scope: Vof Model Specification

`Phase Change Model {=|are|is}` *`VOFPhaseChangeModel [ModelArgs]...`*

Parameter	Value	Default
VOFPhaseChangeModel	`{constant|langmuir|none|thermal}`	-
ModelArgs	`[string]...`	-

Summary Specify the phase change model to use.

Description Select the VOF phase change model (CONSTANT, THERMAL, or NONE)

11.20.8. Post Process Interface Normal

Scope: Vof Model Specification

Summary Post-process the interface normal vector

11.20.9. Smoother Fourier Number

Scope: Vof Model Specification

`Smoother Fourier Number {=|are|is}` *`SmootherFo`*

Parameter	Value	Default
SmootherFo	`real`	-

Summary Specify parameters for interface smoothing for VOF.

Description Select the Fourier number for the interface diffusive smoother

12. PARTICLE REGION

The following commands are found in the Particle Region scope of the input file.

12.1. PARTICLE REGION

Scope: Fuego Procedure

Begin Particle Region *Regionname*

Compute Continuity Source

Compute Energy Source

Compute Mixture_Fraction Source

Compute Momentum Source

Compute Rte Particle Source

Compute Soot_Mass_Fraction Source [With Soot Split {=|are|is} *SootSplit*]

Compute Source Lhs Contribution

Compute Species Source

Disable Particle Transfer Checks

Fluid Region Name {=|are|is} *FluidRegionName*

Fluid Velocity Fluctuation Model {=|are|is} *FluctVelModelType*

Interpolate Fluid Variable *FluidVarName* For Output As *ParticleVarName* [With Vector Size *VecSize*]

Lagrangian Particles

Maximum Particle Subcycles {=|are|is} *SubCycles*

Minimum Particle Subcycles {=|are|is} *SubCycles*

Number Of Fluid Species {=|are|is} *NumSpecies*

Particle Maximum Search Grid Dimension {=|are|is} *Size*

Particle Random Number Generator Seed {=|are|is} *Size*

Particle Rebalance Imbalance Threshold {=|are|is} *Threshold*

Particle Rebalance Step Frequency {=|are|is} *Nsteps*

Particle Reservoir Size {=|are|is} *Size*

Perform Initial Particle Rebalance

Use Finite Element Model *ModelName* [Model Coordinates Are *Nodal_variable_name*]

Begin Create Initial Particle Distribution *DistributionName*
End

Begin Create Particles From File *BlockName*
End

Begin Fluid Variable Specification *SpecificationName*
End

Begin Heartbeat *Label*
End

Begin History Output *Label*
End

Begin Insert Particle *Particlename*
End

Begin Particle Average *AverageName*
End

Begin Particle Breakup Model *ModelName*
End

Begin Particle Definition *Particlename*
End

Begin Particle Filled Shape *FilledShapeName*
End

Begin Particle Filled Tree *FilledTreeName*
End

Begin Particle Inflow Boundary Condition On Surface *Surfacename*

```
    End

    Begin Particle Interaction Model ModelName

    End

    Begin Particle Interface InterfaceName

    End

    Begin Particle Material MaterialName

    End

    Begin Particle Open Boundary Condition On Surface Surfacename

    End

    Begin Particle Spray SprayName

    End

    Begin Particle Stats Output Name

    End

    Begin Particle Tabular Output Name

    End

    Begin Particle Wall Boundary Condition On Surface Surfacename

    End

    Begin Restart Data Label

    End

    Begin Results Output Label

    End

End
```

Summary Contains the commands needed to build and solve a set of Lagrangian particles

12.1.1. Compute Continuity Source

Scope: Particle Region

Summary Compute continuity source to be transferred to fluid region.

Description Compute a continuity source term to be used in the fluid continuity equation (pressure). The source term is accumulated in the nodal field called continuity_source, which has units of mass change per time per volume. This field should be transferred to the fluid region into the corresponding field.

12.1.2. Compute Energy Source

Scope: Particle Region

Summary Compute energy source to be transferred to fluid region.

Description Compute an energy source term to be used in the fluid energy equation. The source term is accumulated in the nodal field called energy_source, which has units of internal energy change per time per volume (the same units as density times dh/dt). This field should be transferred to the fluid region into the corresponding field.

12.1.3. Compute Mixture_Fraction Source

Scope: Particle Region

Summary Compute mixture fraction sources to be transferred to fluid region.

Description Compute a mixture fraction source terms to be used in the fluid species. The source terms are accumulated in two nodal fields called `mixture_fraction_source` and `second_mixture_fraction_source`, which have units of change in mass per time per volume. These fields should be transferred to the fluid region into the corresponding fields. The specific components of the mixture fraction vector for which a source is added depend on the specific Particle Type being used. For example, for the `WILDFIRE_PARTICLE` type the first variable contains the source for water vapor and the second for the volatiles stream.

12.1.4. Compute Momentum Source

Scope: Particle Region

Summary Compute momentum source to be transferred to fluid region.

Description Compute a momentum source term to be used in the fluid momentum equation. The source term is accumulated in the nodal field called momentum_source, which has units of momentum change per time per volume (the same units as density times du/dt). This field should be transferred to the fluid region into the corresponding field.

12.1.5. Compute Rte Particle Source

Scope: Particle Region

Summary Compute radiative source term in particle energy equation.

Description Compute a radiative source term in the particle energy equation. This option will require specification of the `ABSORPTIVITY` in the particle material block. This option will also require the transfer of scalar flux in the transfer from a PMR region to the Lagrangian region.

12.1.6. Compute Soot_Mass_Fraction Source

Scope: Particle Region

Summary Compute soot mass fraction source to be transferred to fluid region.

Description Compute a soot mass fraction source terms to be used in the fluid soot mass fraction transport equation. The source terms are accumulated in the nodal fields called `soot_mass_fraction_source`, which have units of change in mass per time per volume. These fields should be transferred to the fluid region into the corresponding fields.

An optional parameter to this line command specifies the split of volatile production directed to soot. Default for this parameter is 0.01.

12.1.7. Compute Source Lhs Contribution

Scope: Particle Region

Summary Compute LHS matrix contribution from particle source terms to fluid equation.

Description Compute contributions from particle source terms to the diagonal of the LHS matrix of the corresponding fluid equations. This contribution provides added stability in the fluid solve. The LHS contribution is accumulated in nodal fields called x_momentum_source_lhs, y_momentum_source_lhs, and z_momentum_source_lhs for the momentum equations; energy_source_lhs for the energy equation; species_source_lhs for the species equations (not yet implemented); and continuity_source_lhs for the continuity equation (not yet implemented). These fields should be transferred to the fluid region into the corresponding fields.

12.1.8. Compute Species Source

Scope: Particle Region

Summary Compute species source to be transferred to fluid region.

Description Compute a species source term to be used in the fluid species. The source term is accumulated in the nodal field called species_source, which has units of mass of species k change per time per volume. This field should be transferred to the fluid region into the corresponding field.

12.1.9. Disable Particle Transfer Checks

Scope: Particle Region

Summary Disables transfer checking on the particle region.

Description Normally the transfers to the particle region are verified against the particle physics, but this command disables these checks. This is an advanced user option - use with care. Transfer errors are treated as warnings and printed to the log file when this option is active.

12.1.10. Fluid Region Name

Scope: Particle Region

`Fluid Region Name {=|are|is}` *`FluidRegionName`*

Parameter	Value	Default
FluidRegionName	`string`	-

Summary Inform particle region of the name of the fluid region to which it is coupled. This is needed if associations must be made between particle-related and fluid species, e.g. between oxidizers or products of particle reactions that are coupled to fluid species.

12.1.11. Fluid Velocity Fluctuation Model

Scope: Particle Region

`Fluid Velocity Fluctuation Model {=|are|is}` *`FluctVelModelType`*

Parameter	Value	Default
FluctVelModelType	`{continuous_random_walk|discrete_random_walk|none}`	-

Summary Specify the fluid velocity fluctuation type for this particle block.

12.1.12. Interpolate Fluid Variable

Scope: Particle Region

`Interpolate Fluid Variable` *`FluidVarName`* `For Output As` *`ParticleVarName`* `[ With Vector Size` *`VecSize`* `]`

Parameter	Value	Default
FluidVarName	`string`	-
ParticleVarName	`string`	-

Summary Define an interpolation from the fluid to the particles

Description The FluidVarName should be the name of a fluid variable that is on the receiving end of a transfer operation from a Fuego fluid region. The ParticleVarName may be chosen by the user; this same variable should then be listed in the region output block to obtain values in the output file.

12.1.13. Lagrangian Particles

Scope: Particle Region

Summary Allows the introduction of Lagrangian particles into the flow.

12.1.14. Maximum Particle Subcycles

Scope: Particle Region

`Maximum Particle Subcycles {=|are|is}` *`SubCycles`*

Parameter	Value	Default
SubCycles	`integer`	`10`

Summary Set maximum number of subcycles to be taken for each particle at each timestep.

12.1.15. Minimum Particle Subcycles

Scope: Particle Region

`Minimum Particle Subcycles {=|are|is}` *`SubCycles`*

Parameter	Value	Default
SubCycles	`integer`	`1`

Summary Set minimum number of subcycles to be taken for each particle at each timestep.

12.1.16. Number Of Fluid Species

Scope: Particle Region

`Number Of Fluid Species {=|are|is}` *`NumSpecies`*

Parameter	Value	Default
NumSpecies	`integer`	-

Summary Set number of fluid species in the coupled Fuego region (needed for handling mass fractions transferred from Fuego region)

12.1.17. Particle Maximum Search Grid Dimension

Scope: Particle Region

`Particle Maximum Search Grid Dimension {=|are|is}` *`Size`*

Parameter	Value	Default
Size	`integer`	`0`

Summary Specify a maximum dimension of the 2D or 3D grid used for searching for elements that own particles.

Description When particles are initially inserted into the domain, a search is done using an auxiliary structured grid. This parameter
texttPARTICLE MAXIMUM SEARCH GRID DIMENSION allows selection of a maximum size for any one dimension of this grid, i.e. N in an NxNxN grid. This is needed for meshes that are highly graded and can produce a grid that is larger than memory will allow. The default is 0, which means the parameter is ignored. A reasonable value is 50, but is problem dependent. If your job crashes during particle initialization, try setting this parameter.

12.1.18. Particle Random Number Generator Seed

Scope: Particle Region

`Particle Random Number Generator Seed {=|are|is}` *`Size`*

Parameter	Value	Default
Size	`integer`	`0`

Summary Specify a random number generator seed for particle region related rand() calls.

Description Seeds the pseudo-random number generator used by the boost mersene twister random number generator within the particle region with the unsigned integer value. The default seed is set to 12345 + Rank ID for a multiprocessor run (12345 + 0 for a single processor run). This command line will also influence the sequence of values produced by the random number generator outside the particle region. Currently random numbers are used in particle probability distribution functions, particle spray, particle breakup model, particle surface interactions and particle dynamics.

12.1.19. Particle Rebalance Imbalance Threshold

Scope: Particle Region

`Particle Rebalance Imbalance Threshold {=|are|is}` *`Threshold`*

Parameter	Value	Default
Threshold	`real`	`0.5`

Summary Set threshold for the ratio of maximum to minimum particles per processor that triggers dynamic particle rebalance.

Description Set threshold for the ratio of maximum to minimum particles per processor that triggers dynamic particle rebalance. This is the percent imbalance to trigger a rebalance, so an entry of 0.25 will rebalance when the maximum imbalance is 25 percent off from idea. This command works in tandem with the `PARTICLE REBALANCE STEP FREQUENCY` command, and Fuego will test the imbalance threshold and if necessary perform a rebalance only on the steps indicated by the frequency command. If no step frequency is specified, no rebalance will be performed. If the minimum number of particles on a

processor is zero, a value of one is used for the minimum when computing the max/min ratio.

12.1.20. Particle Rebalance Step Frequency

Scope: Particle Region

`Particle Rebalance Step Frequency {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	integer	-

Summary Set frequency (in timesteps) with which dynamic rebalancing of the mesh decomposition of the particle region will be recomputed, when necessary.

Description Set frequency (in timesteps) with which the need for dynamic rebalancing of the mesh decomposition of the particle region will be recomputed. Weighting for the decomposition is based on number of particles per element, so that after redistributing the mesh each processor should include approximately the same number of particles. Note that this command works together with the `PARTICLE REBALANCE IMBALANCE THRESHOLD` command. At the specified frequency, Fuego computes the current level of imbalance (the ratio of maximum to minimum particles per processor). If this imbalance exceeds the specified threshold, a rebalance is performed. If no frequency is specified, the default is to perform no rebalancing.

12.1.21. Particle Reservoir Size

Scope: Particle Region

`Particle Reservoir Size {=|are|is}` *`Size`*

Parameter	Value	Default
Size	integer	-

Summary Keep a reservoir of inactive particles to minimize topology changes and reduce the number of output files.

Description The exodus output file format requires a constant number of mesh objects across all timesteps. In particle simulations, particle mesh objects are created and destroyed whenever they enter or exit the domain; in parallel, particle mesh objects must also be created or destroyed whenever they enter of leave the local mesh partition. This results in a new exodus file being written at each output time step. These files can be concatenated in EnSight or ParaView using filename wildcards, but the number of files can become unwieldy. To alleviate this, Fuego allows a reservoir of inactive particles to be created and stored. At the beginning of the simulation, the reservoir is filled with the number of inactive particles specified by the

texttPARTICLE RESERVOIR SIZE command. When a new particle is introduced to the flow, instead of creating or destroying a mesh object, a particle from the reservoir is activated. Similarly, when a particle passes out of the domain, its mesh object is inactivated and returned to the reservoir. An element variable with field name "active" can be output, equal to 1 for active particles and 0 for inactive; this variable can aid in visualization of only the active particles. If the reservoir shrinks to size zero or grows to twice its specified size, it is reset to the specified size; this results in a change in the number of mesh objects and the need for a new exodus file. However, by choosing a reasonable reservoir size the user can reduce the total number of output files; the ideal size is approximately the maximum number of particles expected on any one processor at any given time throughout the course of the simulation.

12.1.22. Perform Initial Particle Rebalance

Scope: Particle Region

Summary Perform a particle rebalance during initialization

12.1.23. Use Finite Element Model

Scope: Particle Region

```
Use Finite Element Model ModelName [ Model Coordinates Are
Nodal_variable_name ]
```

Parameter	Value	Default
ModelName	string	-

Summary Associates a predefined finite element model with this region.

12.2. CREATE INITIAL PARTICLE DISTRIBUTION

Scope: Particle Region

```
Begin Create Initial Particle Distribution DistributionName
    Diameter {=|are|is} Diameter
    X_Velocity {=|are|is} X_velocity
    Y_Velocity {=|are|is} Y_velocity
    Z_Velocity {=|are|is} Z_velocity
    Particle Definition {=|are|is} Definition
End
```

Summary Defines an initial distribution of particles within the region's volume based on the mesh

This feature is DEPRECATED – it is difficult to ensure that particle creation along the elements of the mesh is applied consistently

12.2.1. Diameter

Scope: Create Initial Particle Distribution

`Diameter {=|are|is}` *`Diameter`*

Parameter	Value	Default
Diameter	`real`	-

Summary Set initial diameter of particle(s)

12.2.2. X_Velocity

Scope: Create Initial Particle Distribution

`X_Velocity {=|are|is}` *`X_velocity`*

Parameter	Value	Default
X_velocity	`real`	-

Summary Set initial x velocity for particle(s).

12.2.3. Y_Velocity

Scope: Create Initial Particle Distribution

`Y_Velocity {=|are|is}` *`Y_velocity`*

Parameter	Value	Default
Y_velocity	`real`	-

Summary Set initial y velocity for particle(s).

12.2.4. Z_Velocity

Scope: Create Initial Particle Distribution

`Z_Velocity {=|are|is}` *`Z_velocity`*

Parameter	Value	Default
Z_velocity	`real`	-

Summary Set initial z velocity for particle(s).

12.2.5. Particle Definition

Scope: Create Initial Particle Distribution

`Particle Definition {=|are|is}` *`Definition`*

Parameter	Value	Default
Definition	`string`	-

Summary Set particle definition to be used for this block

12.3. CREATE PARTICLES FROM FILE

Scope: Particle Region

```
Begin Create Particles From File BlockName
    Creation_Time {=|are|is} CreationTime
    Diameter {=|are|is} Diameter
    Filename {=|are|is} FileName
    Interaction List Filename {=|are|is} InteractionListFileName
    Insertion Time {=|are|is} InsertionTime
    Length_Scale_Factor {=|are|is} LengthScaleFactor
    Mass Represented {=|are|is} Mass_represented
    Mass_Fraction Species {=|are|is} y0
    Number Represented {=|are|is} Num_represented
    Porosity {=|are|is} Porosity
    Time_Scale_Factor {=|are|is} TimeScaleFactor
    Track Particles For Tabular Output
    Temperature {=|are|is} Temperature
    X_Velocity {=|are|is} X_velocity
    Y_Velocity {=|are|is} Y_velocity
    Z_Velocity {=|are|is} Z_velocity
    Particle Definition {=|are|is} Definition
End
```

Summary Block that defines creation of particles using a tab-delimited data file

Description This block defines the creation of a set of particles based on data in a tab-delimited text file. The first line of this file should contain a single integer, giving the number of particles to be created. The rest of the file defines the particle data, one particle per line. There are two accepted formats for the particle file: 7-columns and 10-columns. The following is required in the correct order for the 7-column format: particle positions x, y, and z; particle velocities u, v, and w; and particle diameter d. 3 additional fields are required for the 10-column format, and the fields should be in this order: particle positions x, y, and z; particle velocities u, v, and w; particle diameter d; particle temperature T; particle number represented (parcelling); and particle insertion time.

Note that the particle temperature and number represented, and insertion time are specified as constant for all particles in the file when using the 7-column format.

12.3.1. Creation_Time

Scope: Create Particles From File

`Creation_Time {=|are|is}` *`CreationTime`*

Parameter	Value	Default
CreationTime	`real`	-

Summary Set time at which to insert particles

12.3.2. Diameter

Scope: Create Particles From File

`Diameter {=|are|is}` *`Diameter`*

Parameter	Value	Default
Diameter	`real`	-

Summary Set initial diameter of particle(s)

12.3.3. Filename

Scope: Create Particles From File

`Filename {=|are|is}` *`FileName`*

Parameter	Value	Default
FileName	`string`	-

Summary Name of file containing tab-delimited particle data.

12.3.4. Interaction List Filename

Scope: Create Particles From File

`Interaction List Filename {=|are|is}` *`InteractionListFileName`*

Parameter	Value	Default
InteractionListFileName	`string`	-

Summary Name of file containing interaction list.

12.3.5. Insertion Time

Scope: Create Particles From File

`Insertion Time {=|are|is}` *`InsertionTime`*

Parameter	Value	Default
InsertionTime	`real`	-

Summary Set time at which to insert particle; particle will be inserted at the first time step at or after this value.

12.3.6. Length_Scale_Factor

Scope: Create Particles From File

`Length_Scale_Factor {=|are|is}` *`LengthScaleFactor`*

Parameter	Value	Default
LengthScaleFactor	`real`	-

Summary Multiplicative factor to be applied to all lengths in the input file (e.g. if file data is in meters and simulation is in cm, this factor should be 0.01)

12.3.7. Mass Represented

Scope: Create Particles From File

`Mass Represented {=|are|is}` *`Mass_represented`*

Parameter	Value	Default
Mass_represented	`real`	-

Summary Set mass of physical particles represented by each particle node.

12.3.8. Mass_Fraction

Scope: Create Particles From File

`Mass_Fraction` *`Species`* `{=|are|is}` *`y0`*

Parameter	Value	Default
Species	`string`	-
yo	`real`	-

Summary Specification of mass fraction to be used for computing particle mass initial condition.

12.3.9. Number Represented

Scope: Create Particles From File

`Number Represented {=|are|is}` *`Num_represented`*

Parameter	Value	Default
Num_represented	`integer`	`1`

Summary Set number of physical particles represented by each particle node.

12.3.10. Porosity

Scope: Create Particles From File

`Porosity {=|are|is}` *`Porosity`*

Parameter	Value	Default
Porosity	`real`	-

Summary Set initial porosity for particle(s).

12.3.11. Time_Scale_Factor

Scope: Create Particles From File

`Time_Scale_Factor {=|are|is}` *`TimeScaleFactor`*

Parameter	Value	Default
TimeScaleFactor	`real`	-

Summary Multiplicative factor to be applied to all time values in the input file

12.3.12. Track Particles For Tabular Output

Scope: Create Particles From File

Summary All particles of this description are added to the tabular output

Description Any particle matching this description will be output to the tabular data

12.3.13. Temperature

Scope: Create Particles From File

`Temperature {=|are|is}` *`Temperature`*

Parameter	Value	Default
Temperature	`real`	-

Summary Set initial temperature for particle(s).

12.3.14. X_Velocity

Scope: Create Particles From File

`X_Velocity {=|are|is}` *`X_velocity`*

Parameter	Value	Default
X_velocity	`real`	-

Summary Set initial x velocity for particle(s).

12.3.15. Y_Velocity

Scope: Create Particles From File

`Y_Velocity {=|are|is}` *`Y_velocity`*

Parameter	Value	Default
Y_velocity	`real`	-

Summary Set initial y velocity for particle(s).

12.3.16. Z_Velocity

Scope: Create Particles From File

```
Z_Velocity {=|are|is} Z_velocity
```

Parameter	Value	Default
Z_velocity	real	-

Summary Set initial z velocity for particle(s).

12.3.17. Particle Definition

Scope: Create Particles From File

```
Particle Definition {=|are|is} Definition
```

Parameter	Value	Default
Definition	string	-

Summary Set particle definition to be used for this block

12.4. FLUID VARIABLE SPECIFICATION

Scope: Particle Region

```
Begin Fluid Variable Specification SpecificationName
    Density {=|are|is} Density
    Molecular_Weight {=|are|is} Molec_weight
    Specific_Heat {=|are|is} Specific_heat
    Temperature {=|are|is} Temperature
    Thermal_Conductivity {=|are|is} Thermal_cond
    Turbulent_Dissipation {=|are|is} Turbulent_diss
    Turbulent_Kinetic_Energy {=|are|is} Turbulent_ke
    Viscosity {=|are|is} Viscosity
    X_Velocity {=|are|is} X_velocity
    Y_Velocity {=|are|is} Y_velocity
    Z_Velocity {=|are|is} Z_velocity
End
```

Summary Specify fluid variables that will be used in the absence of transfers from the Fuego fluid region.

12.4.1. Density

Scope: Fluid Variable Specification

`Density {=|are|is}` *`Density`*

Parameter	Value	Default
Density	`real`	-

Summary Set density for fluid.

12.4.2. Molecular_Weight

Scope: Fluid Variable Specification

`Molecular_Weight {=|are|is}` *`Molec_weight`*

Parameter	Value	Default
Molec_weight	`real`	-

Summary Set molecular weight for fluid.

12.4.3. Specific_Heat

Scope: Fluid Variable Specification

`Specific_Heat {=|are|is}` *`Specific_heat`*

Parameter	Value	Default
Specific_heat	`real`	-

Summary Set specific heat for fluid.

12.4.4. Temperature

Scope: Fluid Variable Specification

`Temperature {=|are|is}` *`Temperature`*

Parameter	Value	Default
Temperature	`real`	-

Summary Set temperature for fluid.

12.4.5. Thermal_Conductivity

Scope: Fluid Variable Specification

`Thermal_Conductivity {=|are|is}` *`Thermal_cond`*

Parameter	Value	Default
Thermal_cond	`real`	-

Summary Set thermal conductivity for fluid.

12.4.6. Turbulent_Dissipation

Scope: Fluid Variable Specification

`Turbulent_Dissipation {=|are|is}` *`Turbulent_diss`*

Parameter	Value	Default
Turbulent_diss	`real`	-

Summary Set turbulent dissipation for fluid.

12.4.7. Turbulent_Kinetic_Energy

Scope: Fluid Variable Specification

`Turbulent_Kinetic_Energy {=|are|is}` *`Turbulent_ke`*

Parameter	Value	Default
Turbulent_ke	`real`	-

Summary Set turbulent kinetic energy for fluid.

12.4.8. Viscosity

Scope: Fluid Variable Specification

`Viscosity {=|are|is}` *`Viscosity`*

Parameter	Value	Default
Viscosity	`real`	-

Summary Set viscosity for fluid.

12.4.9. X_Velocity

Scope: Fluid Variable Specification

`X_Velocity {=|are|is}` *`X_velocity`*

Parameter	Value	Default
X_velocity	`real`	-

Summary Set x velocity for fluid.

12.4.10. Y_Velocity

Scope: Fluid Variable Specification

`Y_Velocity {=|are|is}` *`Y_velocity`*

Parameter	Value	Default
Y_velocity	`real`	-

Summary Set y velocity for fluid.

12.4.11. Z_Velocity

Scope: Fluid Variable Specification

`Z_Velocity {=|are|is}` *`Z_velocity`*

Parameter	Value	Default
Z_velocity	`real`	-

Summary Set z velocity for fluid.

12.5. HEARTBEAT

Scope: Average Region

```
Begin Heartbeat Label
    Additional Steps {=|are|is} List_of_steps...
    Additional Times {=|are|is} List_of_times...
    Append {=|are|is} Option
    At Step n Option {=|are|is} m
    At Time Dt1 Option {=|are|is} Dt2
    Element [ VariableList...  ]
```

```
    Exists Option1 Option2

    Face [ VariableList...  ]

    Flush Interval {=|are|is} Option

    Format {=|are|is} StreamTypes

    Global [ Variables...  ]

    Labels {=|are|is} Option

    Legend {=|are|is} Option

    Monitor Equals Option

    Nodal [ VariableList...  ]

    Node [ VariableList...  ]

    Nodeset [ VariableList...  ]

    Output On Signal {=|are|is} Signals

    Precision {=|are|is} Precision

    Start Time {=|are|is} Start_time

    Stream Name {=|are|is} OutputFilename

    Synchronize Output

    Termination Time {=|are|is} Final_time

    Timestamp Format

    Timestep Adjustment Interval {=|are|is} Nsteps

    Use Output Scheduler Timer_name

    Variable {=|are|is} Option Variable_list...

End
```

Summary Describes the location and type of the output stream used for outputting the heartbeat information for the enclosing region.

12.5.1. Additional Steps

Scope: Heartbeat

`Additional Steps {=|are|is}` *`List_of_steps...`*

Parameter	Value	Default
List_of_steps	`integer...`	-

Summary Additional simulation steps when output should occur.

12.5.2. Additional Times

Scope: Heartbeat

`Additional Times {=|are|is}` *`List_of_times...`*

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

12.5.3. Append

Scope: Heartbeat

`Append {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`{false|off|on|true}`	-

Summary Specifies whether the heartbeat file is appended if it exists. By default, the file is appended if restart is requested and not if restart is not requested. This option does not work for automatic restarts because a new heartbeat file is written with each auto restart.

12.5.4. At Step

Scope: Heartbeat

`At Step` *`n Option`* `{=|are|is}` *`m`*

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

12.5.5. At Time

Scope: Heartbeat

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

12.5.6. Element

Scope: Heartbeat

Summary Define the element variables that should be written to the heartbeat database. The syntax is: "element internal_name at element id as DBname" or "element internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database.

12.5.7. Exists

Scope: Heartbeat

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{abort|append|overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis.

12.5.8. Face

Scope: Heartbeat

Summary Define the face variables that should be written to the heartbeat database. The syntax is: "face internal_name at face id as DBname" or "face internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database.

12.5.9. Flush Interval

Scope: Heartbeat

`Flush Interval {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`integer`	`10`

Summary The minimum time interval (in seconds) at which the heartbeat output will be explicitly flushed to disk. The default is 10 seconds.

12.5.10. Format

Scope: Heartbeat

`Format {=|are|is}` *`StreamTypes`*

Parameter	Value	Default
StreamTypes	`{csv|original|spyhis}`	-

Summary The stream type/format to be used for the output results.The only three options at this time are 'Original' which is the old default Sierra heartbeat format; 'SpyHis' which mimics the CTH Spyhis history output format; and 'CSV'

12.5.11. Global

Scope: Heartbeat

Summary Define the global/reduction variables that should be written to the heartbeat database. The syntax is: "global internal_name as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database.

12.5.12. Labels

Scope: Heartbeat

`Labels {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`{off|on}`	`on`

Summary Specifies whether labels will be displayed or just the value of the variable. Labels will be shown if this line is not present.

12.5.13. Legend

Scope: Heartbeat

`Legend {=|are|is}` *`Option`*

Parameter	Value	Default
Option	`{off|on}`	`on`

Summary Specifies whether a legend will be displayed prior to outputting any variables. The legend will not be shown unless this line is present. The legend shows the names of the variables that will be written to the heartbeat output stream. If the variable has multiple components, then the component count is shown after the variable e.g., velocity(3).

12.5.14. Monitor

Scope: Heartbeat

`Monitor` *`Equals Option`*

Parameter	Value	Default
Equals	`{=|the}`	-
Option	`{history|restart|results}`	-

Summary Specifies whether a line will be written to the heartbeat stream when either the results, history, and/or restart data are output.

12.5.15. Nodal

Scope: Heartbeat

Summary Define the nodal variables that should be written to the heartbeat database. The syntax is: "nodal internal_name at node id as DBname" or "nodal internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database.

12.5.16. Node

Scope: Heartbeat

Summary Define the nodal variables that should be written to the heartbeat database. The syntax is: "node internal_name at node id as DBname" or "node internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database.

12.5.17. Nodeset

Scope: Heartbeat

Summary Define the nodeset variables that should be written to the heartbeat database. The syntax is: "nodeset internal_name at node id as DBname" or "nodeset internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the heartbeat database. This option finds a single value for the internal_name specified without having to specify a nodeset id or name.

12.5.18. Output On Signal

Scope: Heartbeat

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

12.5.19. Precision

Scope: Heartbeat

`Precision {=|are|is}` *`Precision`*

Parameter	Value	Default
Precision	`integer`	`5`

Summary The precision to be used for the output of real variables (default=5).

12.5.20. Start Time

Scope: Heartbeat

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

12.5.21. Stream Name

Scope: Heartbeat

`Stream Name {=|are|is}` *`OutputFilename`*

Parameter	Value	Default
OutputFilename	`string`	-

Summary The filename of where the heartbeat data should be written. If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. In addition, there are several predefined streams that can be specified. The predefined streams are 'cout' or 'stdout' specifies standard output; 'cerr', 'stderr', 'clog', or 'log' specifies standard error; 'output' or 'outputPo' specifies Sierra's standard output which is redirected to the file specified by the '-o' option on the command line. If the file already exists, it is overwritten. If this line is omitted, then a filename will be created from the basename of the input file with a ".hrt" suffix appended.

12.5.22. Synchronize Output

Scope: Heartbeat

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

12.5.23. Termination Time

Scope: Heartbeat

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

12.5.24. Timestamp Format

Scope: Heartbeat

Summary The format to be used for the timestamp. See 'man strftime' for more information.

12.5.25. Timestep Adjustment Interval

Scope: Heartbeat

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	`integer`	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

12.5.26. Use Output Scheduler

Scope: Heartbeat

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	`string`	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

12.5.27. Variable

Scope: Heartbeat

`Variable {=|are|is}` *`Option Variable_list...`*

Parameter	Value	Default
Option	`{edge|element|face|global|nodal|node}`	-
Variable_list	`string...`	-

Summary Define the variables that should be written to the heartbeat output. The user can request that the values of certain variables be output on the heartbeat line. These variables are limited to region and framework control data currently. The syntax is:

```
variable = {entity_type} {internal_name} at
           {entity_type} {entity_id}     as {external_name}
variable = {entity_type} {internal_name} nearest location
           {x,y,z} as {external_name}
```

For global variables, use:

```
variable = global {internal_name} [as {external_name}]
```

Where:

```
entity_type = node, element, face, edge, global
internal_name = Sierra variable name
entity_id = id of the node, element, face, edge that you want
    the specified variable output at.
external_name = name of variable on the database.
```

The names 'timestep', and 'time' can be specified as variables also. They are the current timestep and simulation time. This line can appear multiple times.

12.6. HISTORY OUTPUT

Scope: Average Region

Begin History Output *Label*

Additional Steps {=|are|is} *List_of_steps...*

Additional Times {=|are|is} *List_of_times...*

At Step *n Option* {=|are|is} *m*

At Time *Dt1 Option* {=|are|is} *Dt2*

Database Name {=|are|is} *StreamName*

Database Type {=|are|is} *DatabaseTypes*

Element [*VariableList...*]

Exists *Option1 Option2*

Face [*VariableList...*]

Global [*Variables...*]

Nodal [*VariableList...*]

Node [*VariableList...*]

Nodeset [*VariableList...*]

Output On Signal {=|are|is} *Signals*

Overwrite *Option1 Option2*

Property *PropertyName* {=|are|is} *PropertyValue*

Start Time {=|are|is} *Start_time*

Synchronize Output

Termination Time {=|are|is} *Final_time*

Timestep Adjustment Interval {=|are|is} *Nsteps*

```
    Title
    Use Output Scheduler Timer_name
    Variable {=|are|is} Option Variable_list...
End
```

Summary Describes the location and type of the output stream used for outputting history for the enclosing region.

12.6.1. Additional Steps

Scope: History Output

```
Additional Steps {=|are|is} List_of_steps...
```

Parameter	Value	Default
List_of_steps	`integer...`	-

Summary Additional simulation steps when output should occur.

12.6.2. Additional Times

Scope: History Output

```
Additional Times {=|are|is} List_of_times...
```

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

12.6.3. At Step

Scope: History Output

```
At Step n Option {=|are|is} m
```

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

12.6.4. At Time

Scope: History Output

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

12.6.5. Database Name

Scope: History Output

`Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The base name of the database containing the output history. If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. If this line is omitted, then a filename will be created from the basename of the input file with a ".h" suffix appended.

12.6.6. Database Type

Scope: History Output

`Database Type {=|are|is}` *`DatabaseTypes`*

Parameter	Value	Default
DatabaseTypes	`{catalyst|dof|dof_exodus|exodus|exodusii|generated|genesis|parallel_exodus}`	-

Summary The database type/format to be used for the output history.

12.6.7. Element

Scope: History Output

Summary Define the element variables that should be written to the history database. The syntax is: "element internal_name at element id as DBname" or "element internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

12.6.8. Exists

Scope: History Output

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{= | is}`	-
Option2	`{abort | add_suffix | append | overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis. "ADD_SUFFIX" will add a -s???? suffix where the ???? is replaced by a sequential number starting at 0002.

12.6.9. Face

Scope: History Output

Summary Define the face variables that should be written to the history database. The syntax is: "face internal_name at face id as DBname" or "face internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

12.6.10. Global

Scope: History Output

Summary Define the global/reduction variables that should be written to the history database. The syntax is: "global internal_name as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

12.6.11. Nodal

Scope: History Output

Summary Define the nodal variables that should be written to the history database. The syntax is: "nodal internal_name at node id as DBname" or "nodal internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

12.6.12. Node

Scope: History Output

Summary Define the nodal variables that should be written to the history database. The syntax is: "node internal_name at node id as DBname" or "node internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

12.6.13. Nodeset

Scope: History Output

Summary Define the nodeset variables that should be written to the history database. The syntax is: "nodeset internal_name at node id as DBname" or "nodeset internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

12.6.14. Output On Signal

Scope: History Output

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

12.6.15. Overwrite

Scope: History Output

`Overwrite` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{false|no|off|on|true|yes}`	-

Summary (DEPRECATED, Use EXISTS) Specify whether the database should be overwritten if it exists. The default behavior is to overwrite unless this command is specified in the output block and either off, false, or no is specified.

12.6.16. Property

Scope: History Output

`Property` *`PropertyName`* `{=|are|is}` *`PropertyValue`*

Parameter	Value	Default
PropertyName	`string`	-
PropertyValue	`string`	-

Summary Define a database property named "PropertyName" with the value "PropertyValue". If PropertyValue consists of all digits, it will define an integer property. If PropertyValue is "true" or "yes" or "false" or "no", it will define a logical property; otherwise it will define a string property. Suppported properties are typically database dependent; Some history-related properties are: VARIABLE_NAME_CASE = upper|lower MAX_NAME_LENGTH = value (32)

12.6.17. Start Time

Scope: History Output

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

12.6.18. Synchronize Output

Scope: History Output

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

12.6.19. Termination Time

Scope: History Output

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

12.6.20. Timestep Adjustment Interval

Scope: History Output

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	`integer`	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

12.6.21. Title

Scope: History Output

Summary Specify the title to be used for this specific output block.

12.6.22. Use Output Scheduler

Scope: History Output

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	`string`	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

12.6.23. Variable

Scope: History Output

`Variable {=|are|is}` *`Option Variable_list...`*

Parameter	Value	Default
Option	`{edge|element|face|global|nodal|node}`	-
Variable_list	`string...`	-

Summary Define the variables that should be written to the history database. The syntax is: "variable = entity internal_name at entity id as DBname" or "variable = entity internal_name nearest location X, Y, Z as DBname" or "variable = entity internal_name at location X, Y, Z as DBname".

Where entity is 'node', 'element', 'face', or 'edge'; internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

12.7. INSERT PARTICLE

Scope: Particle Region

```
Begin Insert Particle Particlename
    Diameter {=|are|is} Diameter
    Insertion Time {=|are|is} InsertionTime
    Mass Represented {=|are|is} Mass_represented
    Mass_Fraction Species {=|are|is} y0
    Number {=|are|is} Number
    Number Represented {=|are|is} Num_represented
    Porosity {=|are|is} Porosity
    Track Particles For Tabular Output
    Temperature {=|are|is} Temperature
    X {=|are|is} x
    X1 {=|are|is} x
    X2 {=|are|is} x
    X_Velocity {=|are|is} X_velocity
    Y {=|are|is} y
    Y1 {=|are|is} y
    Y2 {=|are|is} y
    Y_Velocity {=|are|is} Y_velocity
    Z {=|are|is} z
    Z1 {=|are|is} z
    Z2 {=|are|is} z
    Z_Velocity {=|are|is} Z_velocity
    Particle Definition {=|are|is} Definition
End
```

Summary Block that defines data for an individual particle to be inserted into the flow.

12.7.1. Diameter

Scope: Insert Particle

`Diameter {=|are|is}` *`Diameter`*

Parameter	Value	Default
Diameter	`real`	-

Summary Set initial diameter of particle(s)

12.7.2. Insertion Time

Scope: Insert Particle

`Insertion Time {=|are|is}` *`InsertionTime`*

Parameter	Value	Default
InsertionTime	`real`	-

Summary Set time at which to insert particle; particle will be inserted at the first time step at or after this value.

12.7.3. Mass Represented

Scope: Insert Particle

`Mass Represented {=|are|is}` *`Mass_represented`*

Parameter	Value	Default
Mass_represented	`real`	-

Summary Set mass of physical particles represented by each particle node.

12.7.4. Mass_Fraction

Scope: Insert Particle

`Mass_Fraction` *`Species`* `{=|are|is}` *`y0`*

Parameter	Value	Default
Species	`string`	-
yo	`real`	-

Summary Specification of mass fraction to be used for computing particle mass initial condition.

12.7.5. Number

Scope: Insert Particle

`Number {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	-

Summary Set number of particle nodes to insert. For a line insertion, this should be greater than 1, and points (x1,y1,z1) and (x2,y2,z2) should be input. This places a line of equally spaced particle nodes between the two points. If no second point is specified, all particles are placed at the same point. Note that if the NUMBER REPRESENTED command is used, the total number of physical particles equals NUMBER * NUMBER_REPRESENTED

12.7.6. Number Represented

Scope: Insert Particle

`Number Represented {=|are|is}` *`Num_represented`*

Parameter	Value	Default
Num_represented	`integer`	1

Summary Set number of physical particles represented by each particle node.

12.7.7. Porosity

Scope: Insert Particle

`Porosity {=|are|is}` *`Porosity`*

Parameter	Value	Default
Porosity	`real`	-

Summary Set initial porosity for particle(s).

12.7.8. Track Particles For Tabular Output

Scope: Insert Particle

Summary All particles of this description are added to the tabular output

Description Any particle matching this description will be output to the tabular data

12.7.9. Temperature

Scope: Insert Particle

`Temperature {=|are|is}` *`Temperature`*

Parameter	Value	Default
Temperature	`real`	-

Summary Set initial temperature for particle(s).

12.7.10. X

Scope: Insert Particle

`X {=|are|is}` *x*

Parameter	Value	Default
x	`real`	-

Summary Set initial x position for particle(s).

12.7.11. X1

Scope: Insert Particle

`X1 {=|are|is}` *x*

Parameter	Value	Default
x	`real`	-

Summary Set x position for point 1 defining a line of particles.

12.7.12. X2

Scope: Insert Particle

`X2 {=|are|is}` *x*

Parameter	Value	Default
x	`real`	-

Summary Set x position for point 2 defining a line of particles.

12.7.13. X_Velocity

Scope: Insert Particle

`X_Velocity {=|are|is}` *`X_velocity`*

Parameter	Value	Default
X_velocity	`real`	-

Summary Set initial x velocity for particle(s).

12.7.14. Y

Scope: Insert Particle

`Y {=|are|is}` *y*

Parameter	Value	Default
y	`real`	-

Summary Set initial y position for particle(s).

12.7.15. Y1

Scope: Insert Particle

`Y1 {=|are|is}` *y*

Parameter	Value	Default
y	`real`	-

Summary Set y position for point 1 defining a line of particles.

12.7.16. Y2

Scope: Insert Particle

`Y2 {=|are|is}` *y*

Parameter	Value	Default
y	`real`	-

Summary Set y position for point 2 defining a line of particles.

12.7.17. Y_Velocity

Scope: Insert Particle

`Y_Velocity {=|are|is}` *`Y_velocity`*

Parameter	Value	Default
Y_velocity	`real`	-

Summary Set initial y velocity for particle(s).

12.7.18. Z

Scope: Insert Particle

`Z {=|are|is}` *`z`*

Parameter	Value	Default
z	`real`	-

Summary Set initial z position for particle(s).

12.7.19. Z1

Scope: Insert Particle

`Z1 {=|are|is}` *`z`*

Parameter	Value	Default
z	`real`	-

Summary Set z position for point 1 defining a line of particles.

12.7.20. Z2

Scope: Insert Particle

`Z2 {=|are|is}` *`z`*

Parameter	Value	Default
z	`real`	-

Summary Set z position for point 2 defining a line of particles.

12.7.21. Z_Velocity

Scope: Insert Particle

`Z_Velocity {=|are|is}` *`Z_velocity`*

Parameter	Value	Default
Z_velocity	`real`	-

Summary Set initial z velocity for particle(s).

12.7.22. Particle Definition

Scope: Insert Particle

`Particle Definition {=|are|is}` *`Definition`*

Parameter	Value	Default
Definition	`string`	-

Summary Set particle definition to be used for this block

12.8. PARTICLE AVERAGE

Scope: Particle Region

```
Begin Particle Average AverageName

    Average Field ParticleField For Output As CellField

    Averaging Type {=|are|is} AverageType

    Time Average Using Period TimePeriod

End
```

Summary Block that defines averaging over particles in each control volume cell.

12.8.1. Average Field

Scope: Particle Average

`Average Field` *`ParticleField`* `For Output As` *`CellField`*

Parameter	Value	Default
ParticleField	`string`	-
CellField	`string`	-

Summary Define averaging operation from a field defined onto the particles into another field available for output on the fluid mesh.

12.8.2. Averaging Type

Scope: Particle Average

`Averaging Type {=|are|is}` *`AverageType`*

Parameter	Value	Default
AverageType	`{mass|number|volume}`	-

Summary Type of particle averaging to perform. Options are NUMBER, MASS, and VOLUME for number-weighted, mass-weighted, and volume weighted means.

12.8.3. Time Average Using Period

Scope: Particle Average

`Time Average Using Period` *`TimePeriod`*

Parameter	Value	Default
TimePeriod	`real`	-

Summary Average particle variables over time as well as volume, over specified time period.

12.9. PARTICLE BREAKUP MODEL

Scope: Particle Region

```
Begin Particle Breakup Model ModelName
    Breakup Model Type {=|are|is} ParticleBreakupModelType
    Diameter Reduction Factor {=|are|is} Factor
    Limit New Particle Splitting
    Maximum Number Of Particles Per Parcel {=|are|is} Number
```

```
        Minimum Mass Per Parcel {=|are|is} Mass

        Particle Breakup Parameter VarName {=|are|is} Value...

    End
```

Summary Block that defines a specific particle breakup model.

12.9.1. Breakup Model Type

Scope: Particle Breakup Model

```
Breakup Model Type {=|are|is} ParticleBreakupModelType
```

Parameter	Value	Default
ParticleBreakupModelType	`{tab_model}`	-

Summary Type of breakup model.

12.9.2. Diameter Reduction Factor

Scope: Particle Breakup Model

```
Diameter Reduction Factor {=|are|is} Factor
```

Parameter	Value	Default
Factor	`real`	-

Summary Limit reduction of particle diameter by the specified value.

12.9.3. Limit New Particle Splitting

Scope: Particle Breakup Model

Summary If a particle has just been created on the previous step, limit breakup to splitting the particle in half.

12.9.4. Maximum Number Of Particles Per Parcel

Scope: Particle Breakup Model

```
Maximum Number Of Particles Per Parcel {=|are|is} Number
```

Parameter	Value	Default
Number	`integer`	-

Summary After splitting, create new parcels if, and only if, number of particles in the parcel exceeds specified maximum. This may be superseded by a specified minimum mass.

12.9.5. Minimum Mass Per Parcel

Scope: Particle Breakup Model

`Minimum Mass Per Parcel {=|are|is}` *`Mass`*

Parameter	Value	Default
Mass	`real`	-

Summary Limit creation of new parcels if parcel mass drops below the specified minimum.

12.9.6. Particle Breakup Parameter

Scope: Particle Breakup Model

`Particle Breakup Parameter` *`VarName`* `{=|are|is}` *`Value...`*

Parameter	Value	Default
VarName	`string`	-
Value	`real...`	-

Summary Parameter for the particle breakup model.

Description The breakup model requires a number of user-defined parameters to be entered. An exception will be thrown if any of these required parameters are missing.

The TAB_MODEL breakup model requires parameters PARTICLE_VISCOSITY and PARTICLE_SURFACE_TENSION.

12.10. PARTICLE DEFINITION

Scope: Particle Region

```
Begin Particle Definition Particlename
    Add Particle Interface InterfaceName
    Add Particle Material MaterialName
    Minimum_Diameter {=|are|is} MinimumDiameter
    Minimum_Mass {=|are|is} MinimumMass
    Particle Breakup Model {=|are|is} ModelName
    Particle Energy Equation {=|are|is} ParticleEnergyType
    Particle Interaction Model {=|are|is} ModelName
    Particle Is Stationary
    Particle Type {=|are|is} ParticleType
```

```
    Particle Velocity Equation {=|are|is} ParticleVelocityType

    Use Particle Species {=|are|is} UseParticleSpecies...

    Use Species Material SpeciesMaterialName For SpeciesName

    Begin General Chemistry ChemistryName

    End

    Begin Ode Solver Parameters SolverName

    End

End
```

Summary Block that defines a particle type, including all of its parameters. At least one of these blocks is necessary in every particle region

12.10.1. Add Particle Interface

Scope: Particle Definition

```
Add Particle Interface InterfaceName
```

Parameter	Value	Default
InterfaceName	string	-

Summary Add a particle interface to the particle definition

12.10.2. Add Particle Material

Scope: Particle Definition

```
Add Particle Material MaterialName
```

Parameter	Value	Default
MaterialName	string	-

Summary Add a material component to the particle definition

12.10.3. Minimum_Diameter

Scope: Particle Definition

`Minimum_Diameter {=|are|is}` *`MinimumDiameter`*

Parameter	Value	Default
MinimumDiameter	`real`	-

Summary Set the minimum diameter before deletion of the particle.

12.10.4. Minimum_Mass

Scope: Particle Definition

`Minimum_Mass {=|are|is}` *`MinimumMass`*

Parameter	Value	Default
MinimumMass	`real`	-

Summary Set the minimum mass before deletion of the particle.

12.10.5. Particle Breakup Model

Scope: Particle Definition

`Particle Breakup Model {=|are|is}` *`ModelName`*

Parameter	Value	Default
ModelName	`string`	-

Summary Set a breakup model on this particle type

12.10.6. Particle Energy Equation

Scope: Particle Definition

`Particle Energy Equation {=|are|is}` *`ParticleEnergyType`*

Parameter	Value	Default

ParticleEnergyType	`{chemically_reactive_energy_type`
	`|constant_particle_energy_type`
	`|heated_evaporating_energy_type`
	`|heated_particle_energy_type|`
	`wild_fire_energy_type}`
		-

Summary Specify the particle energy type for this particle block.

12.10.7. Particle Interaction Model

Scope: Particle Definition

`Particle Interaction Model {=|are|is}` *`ModelName`*

Parameter	Value	Default
ModelName	`string`	-

Summary Set an interaction model on this particle type

12.10.8. Particle Is Stationary

Scope: Particle Definition

Summary Marks the particle as being stationary, ie. the particle remains at the same position for the lifetime of the simulation.

12.10.9. Particle Type

Scope: Particle Definition

`Particle Type {=|are|is}` *`ParticleType`*

Parameter	Value	Default
ParticleType	`{cpd_particle|evaporating_particle|fixed_heated_particle|fixed_particle|generalized_particle|heated_particle|inertial_particle|tracker|wildfire_particle}`	-

Summary Specify the particle type for this particle block.

12.10.10. Particle Velocity Equation

Scope: Particle Definition

`Particle Velocity Equation {=|are|is}` *`ParticleVelocityType`*

Parameter	Value	Default
ParticleVelocityType	`{inertial_particle_velocity|tracker_particle_velocity}`	-

Summary Specify the particle velocity type for this particle block.

12.10.11. Use Particle Species

Scope: Particle Definition

`Use Particle Species {=|are|is}` *`UseParticleSpecies...`*

Parameter	Value	Default
UseParticleSpecies	`string...`	-

Summary Specifies the given particle species present within a particle.

Note that the number of speies in a particle is determined through the list of species given in the input file.

12.10.12. Use Species Material

Scope: Particle Definition

`Use Species Material` *`SpeciesMaterialName`* `For` *`SpeciesName`*

Parameter	Value	Default
SpeciesMaterialName	`string`	-
SpeciesName	`string`	-

Summary Assign a material type to a species specified in USE PARTICLE SPECIES

Description Allows for material types defined in BEGIN PARTICLE MATERIAL to be used as a specific species specified within the PARTICLE DEFINITION block.

12.11. PARTICLE FILLED SHAPE

Scope: Particle Region

```
Begin Particle Filled Shape FilledShapeName
    Diameter {=|are|is} Diameter
    Insertion Time {=|are|is} InsertionTime
    Mass Represented {=|are|is} Mass_represented
    Mass_Fraction Species {=|are|is} y0
    Number Represented {=|are|is} Num_represented
    Particle Diameter Distribution Parameter VarName {=|are|is} Value

    Particle Diameter Distribution Type {=|are|is} DistributionType
    Particle Velocity Distribution Parameter VarName {=|are|is} Value
```

```
    Particle Velocity Distribution Type {=|are|is} DistributionType

    Particle Shape Parameter VarName {=|are|is} Value...

    Particle Shape Type {=|are|is} ShapeType

    Porosity {=|are|is} Porosity

    Track Particles For Tabular Output

    Temperature {=|are|is} Temperature

    X_Velocity {=|are|is} X_velocity

    Y_Velocity {=|are|is} Y_velocity

    Z_Velocity {=|are|is} Z_velocity

    Particle Definition {=|are|is} Definition

End
```

Summary Block that defines data for a particle filled shape.

12.11.1. Diameter

Scope: Particle Filled Shape

```
Diameter {=|are|is} Diameter
```

Parameter	Value	Default
Diameter	real	-

Summary Set initial diameter of particle(s)

12.11.2. Insertion Time

Scope: Particle Filled Shape

```
Insertion Time {=|are|is} InsertionTime
```

Parameter	Value	Default
InsertionTime	real	-

Summary Set time at which to insert particle; particle will be inserted at the first time step at or after this value.

12.11.3. Mass Represented

Scope: Particle Filled Shape

`Mass Represented {=|are|is}` *`Mass_represented`*

Parameter	Value	Default
Mass_represented	`real`	-

Summary Set mass of physical particles represented by each particle node.

12.11.4. Mass_Fraction

Scope: Particle Filled Shape

`Mass_Fraction` *`Species`* `{=|are|is}` *`y0`*

Parameter	Value	Default
Species	`string`	-
yo	`real`	-

Summary Specification of mass fraction to be used for computing particle mass initial condition.

12.11.5. Number Represented

Scope: Particle Filled Shape

`Number Represented {=|are|is}` *`Num_represented`*

Parameter	Value	Default
Num_represented	`integer`	1

Summary Set number of physical particles represented by each particle node.

12.11.6. Particle Diameter Distribution Parameter

Scope: Particle Filled Shape

`Particle Diameter Distribution Parameter` *`VarName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
VarName	`string`	-
Value	`real`	-

Summary Parameter for the probability distribution function for particle diameters.

Description The number and type of parameters needed depends on the type of the probability distribution being used.

```
CONSTANT      distribution requires parameter  VALUE
UNIFORM       distribution requires parameters MIN, MAX
NORMAL        distribution requires parameters MEAN, STDEV
LOGNORMAL     distribution requires parameters LOGMEAN, LOGSTDEV
              where LOGMEAN is the mean of the variable's
      natural logarithm (which is normally
      distributed), and LOGSTDEV is the standard
      deviation of the logarithms
ROSIN_RAMMLER distribution requires parameters X, Q
                  where X is related to the mean and Q
          controls the spread
```

Each parameter required should be entered in a separate command line.

PDF names with "_MASS" appended at the end indicate that the distribution is weighted by particle mass, i.e. that the PDF gives the fraction of total particle mass, rather than particle number, with the given diameter range.

12.11.7. Particle Diameter Distribution Type

Scope: Particle Filled Shape

`Particle Diameter Distribution Type {=|are|is}` *`DistributionType`*

Parameter	Value	Default
DistributionType	{constant|lognormal|lognormal_mass|normal|normal_mass|rosin_rammler|rosin_rammler_mass|uniform|uniform_mass}	CONSTANT

Summary Probability distribution type for particle diameters

12.11.8. Particle Velocity Distribution Parameter

Scope: Particle Filled Shape

`Particle Velocity Distribution Parameter` *`VarName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
VarName	string	-
Value	real	-

Summary Parameters for the probability distribution function for particle velocities.

Description The number and type of parameters needed depends on the type of the probability distribution being used.

```
CONSTANT      distribution requires parameter  VALUE
UNIFORM       distribution requires parameters MIN, MAX
NORMAL        distribution requires parameters MEAN, STDEV
LOGNORMAL     distribution requires parameters LOGMEAN, LOGSTDEV
                where LOGMEAN is the mean of the variable's
natural logarithm (which is normally
distributed), and LOGSTDEV is the standard
deviation of the logarithms
ROSIN_RAMMLER distribution requires parameters X, Q
                        where X is related to the mean and Q
        controls the spread
```

Each parameter required should be entered in a separate command line.

PDF names with "_MASS" appended at the end indicate that the distribution is weighted by particle mass, i.e. that the PDF gives the fraction of total particle mass, rather than particle number, with the given velocity range.

12.11.9. Particle Velocity Distribution Type

Scope: Particle Filled Shape

`Particle Velocity Distribution Type {=|are|is}` *`DistributionType`*

Parameter	Value	Default
DistributionType	{constant|lognormal|lognormal_mass |normal|normal_mass|rosin_rammler |rosin_rammler_mass|uniform| uniform_mass}	CONSTANT

Summary Probability distribution type for particle velocities

12.11.10. Particle Shape Parameter

Scope: Particle Filled Shape

`Particle Shape Parameter` *`VarName`* `{=|are|is}` *`Value...`*

Parameter	Value	Default
VarName	string	-
Value	real...	-

Summary Parameter for the particle filled shape.

Description For each particle filled shape a particle definition and shape type must be specified. Also, particle diameter distribution type and its appropriate parameters must be specified.

A correctly defined ParticleFilledShape must also meet prescribed parameters. The number and type of parameters needed depends on the type of the particle filled shape being employed.

For all particle filled shapes, CENTER and NUMBER_OF_PARTICLES or PARTICLE_NUMBER_DENSITY parameters must be set. Additional parameters that must be set based on the shape are as follows:

```
SPHERE   req. params RADIUS
CYLINDER req. params RADIUS, HEIGHT, NORMAL
BRICK    req. params DIMENSIONS, V1, V2
CONE     req. params RADIUS, HEIGHT, NORMAL

 opt. param *VELOCITY_NORMAL
```

Each parameter required should be entered in a separate command line.

Extra notes:

CYLINDER: For the cylinder, the center point is defined at the center of the circular base. The normal is a vector that points in the direction of the cylinders height.

BRICK: For the brick, the center point is defined at the center of the rectangular base, which lies in the plane of V1 and V2 orthogonal vectors. The normal, constructed from the cross product of V1 and V2, points in the direction of the bricks height. Parameter DIMENSIONS specifies width, length and height (in that order). The width and length is defined along V1 and V2 vectors respectively.

CONE: For the cone, the center point is defined at the center of the circular base. The normal is a vector that points in the direction of the cones height (tip).

*VELOCITY_NORMAL is a parameter required when a Particle Velocity Distribution is specified. If particle velocity distribution is not specified, the magnitude of velocity of all particles within a shape is set to 0.

12.11.11. Particle Shape Type

Scope: Particle Filled Shape

`Particle Shape Type {=|are|is}` *`ShapeType`*

Parameter	Value	Default
ShapeType	`{brick|cone|cylinder|sphere}`	-

Summary Particle filled shape type.

12.11.12. Porosity

Scope: Particle Filled Shape

`Porosity {=|are|is}` *`Porosity`*

Parameter	Value	Default
Porosity	`real`	-

Summary Set initial porosity for particle(s).

12.11.13. Track Particles For Tabular Output

Scope: Particle Filled Shape

Summary All particles of this description are added to the tabular output

Description Any particle matching this description will be output to the tabular data

12.11.14. Temperature

Scope: Particle Filled Shape

`Temperature {=|are|is}` *`Temperature`*

Parameter	Value	Default
Temperature	`real`	-

Summary Set initial temperature for particle(s).

12.11.15. X_Velocity

Scope: Particle Filled Shape

`X_Velocity {=|are|is}` *`X_velocity`*

Parameter	Value	Default
X_velocity	`real`	-

Summary Set initial x velocity for particle(s).

12.11.16. Y_Velocity

Scope: Particle Filled Shape

`Y_Velocity {=|are|is}` *`Y_velocity`*

Parameter	Value	Default
Y_velocity	`real`	-

Summary Set initial y velocity for particle(s).

12.11.17. Z_Velocity

Scope: Particle Filled Shape

`Z_Velocity {=|are|is}` *`Z_velocity`*

Parameter	Value	Default
Z_velocity	`real`	-

Summary Set initial z velocity for particle(s).

12.11.18. Particle Definition

Scope: Particle Filled Shape

`Particle Definition {=|are|is}` *`Definition`*

Parameter	Value	Default
Definition	`string`	-

Summary Set particle definition to be used for this block

12.12. PARTICLE FILLED TREE

Scope: Particle Region

```
Begin Particle Filled Tree FilledTreeName

    Particle Tree Parameter VarName {=|are|is} Value...

    Particle Tree Type {=|are|is} TreeType

    Begin Crown Info Name

    End

    Begin Trunk Info Name

    End
```

```
End
```

Summary Block that defines data for a particle filled tree.

12.12.1. Particle Tree Parameter

Scope: Particle Filled Tree

```
Particle Tree Parameter VarName {=|are|is} Value...
```

Parameter	Value	Default
VarName	`string`	-
Value	`real...`	-

Summary Parameter for the particle filled tree.

Description ParticleFilledTree classes are added to support the following 3D particle filled tree shape categories: DECIDUOUS, CONIFEROUS, SHRUB.

DECIDUOUS tree includes oak, maple, redwood. These types of trees are represented by a sphere, cone and cylinder shapes. Sphere represents the crown, cylinder represents the trunk and the cone represents trunk narrowing inside the crown defined between the bottom surface of the crown (sphere) where its basal radius is that of the trunk and its center.

CONIFEROUS tree includes pine, cedar, spruce, fir. These types of trees are represented by two cones and cylinder shapes. The first cone represents the crown, cylinder represents the trunk and the second cone represents trunk narrowing inside the crown defined between the bottom surface of the crown (cone) where its basal radius is that of the trunk and its tip.

SHRUB, also referred to as a woody plant, includes a broom. They are represented by a sphere and cone shapes. Sphere represents the crown and the cone represents trunk narrowing inside the crown between the bottom surface of the crown (sphere) and its center. Note, a SHRUB is a special case of a DECIDUOUS tree in which the trunk outside the crown is missing.

A correctly defined ParticleFilledTree must meet prescribed parameters. The number and type of parameters needed depends on the type of the particle filled tree being employed.

For all particle filled trees, the following must be set:

- Particle Tree Type
- Particle Crown Definition
- Particle Trunk Definition

For all particle filled trees, the following parameters must also be set:

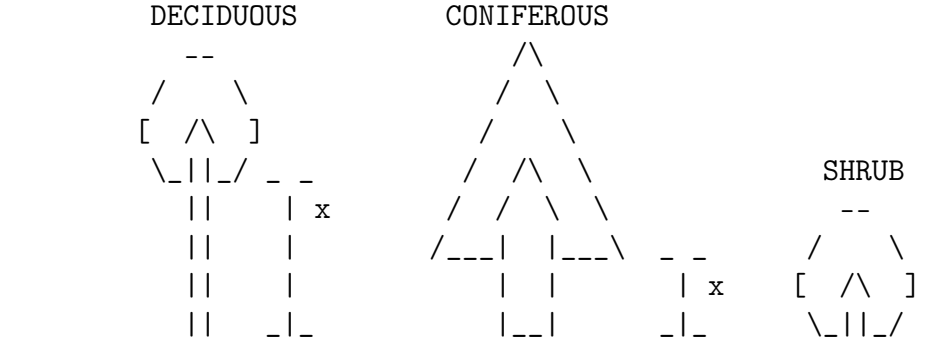
```
CENTER
NORMAL
CROWN_NUMBER_OF_PARTICLES
CROWN_VOLUME_FRACTION
CROWN_PARTICLE_DIAMETER_VARIATION
TRUNK_PARTICLE_DIAMETER_VARIATION
        TRUNK_NUMBER_OF_PARTICLES
        TRUNK_VOLUME_FRACTION
        CROWN_RADIUS
        TRUNK_RADIUS
```

Additional parameters that must be set based on the tree are as follows:

```
DECIDUOUS  req. params CROWN_BASE_HEIGHT
CONIFEROUS req. params TREE_HEIGHT, CROWN_BASE_HEIGHT
```

Each parameter required should be entered in a separate command line.

Extra notes:




```
    CENTER is a center point defined at the base of the trunk.
    NORMAL is a vector perpendicular to the base of the trunk.
    TREE_HEIGHT is the distance defined between the ground and the
top of the tree crown.
    CROWN_BASE_HEIGHT is the distance between the ground and the base
      of the tree crown (or 'x' in the above schematic).
    CROWN_VOLUME_FRACTION, or porosity for a porous medium, is a percent
   number provided between 0 and 1 that describes
   constituent material based on the whole volume.
                          For a spherical fuel element, this value should
                          never be larger than a sixth of pi.
    CROWN_PARTICLE_DIAMETER_VARIATION is a percent number provided between
      0 and 1 that describes particle
      diameter variation. Within the range,
      random particle diameters are selected,
      which are based on the number of
      particles and volume fraction params.
```

```
TRUNK_PARTICLE_DIAMETER_VARIATION (see above)
TRUNK_VOLUME_FRACTION (see above)
```

12.12.2. Particle Tree Type

Scope: Particle Filled Tree

`Particle Tree Type {=|are|is}` *`TreeType`*

Parameter	Value	Default
TreeType	{coniferous|deciduous|shrub}	-

Summary Particle filled tree type. Supported tree types include DECIDUOUS, CONIFEROUS and SHRUB.

12.13. CROWN INFO

Scope: Particle Filled Tree

```
Begin Crown Info Name
    Diameter {=|are|is} Diameter
    Insertion Time {=|are|is} InsertionTime
    Mass Represented {=|are|is} Mass_represented
    Mass_Fraction Species {=|are|is} y0
    Number Represented {=|are|is} Num_represented
    Porosity {=|are|is} Porosity
    Track Particles For Tabular Output
    Temperature {=|are|is} Temperature
    X_Velocity {=|are|is} X_velocity
    Y_Velocity {=|are|is} Y_velocity
    Z_Velocity {=|are|is} Z_velocity
    Particle Definition {=|are|is} Definition
End
```

Summary Block that defines data for a tree crown.

12.13.1. Diameter

Scope: Crown Info

`Diameter {=|are|is}` *`Diameter`*

Parameter	Value	Default
Diameter	`real`	-

Summary Set initial diameter of particle(s)

12.13.2. Insertion Time

Scope: Crown Info

`Insertion Time {=|are|is}` *`InsertionTime`*

Parameter	Value	Default
InsertionTime	`real`	-

Summary Set time at which to insert particle; particle will be inserted at the first time step at or after this value.

12.13.3. Mass Represented

Scope: Crown Info

`Mass Represented {=|are|is}` *`Mass_represented`*

Parameter	Value	Default
Mass_represented	`real`	-

Summary Set mass of physical particles represented by each particle node.

12.13.4. Mass_Fraction

Scope: Crown Info

`Mass_Fraction` *`Species`* `{=|are|is}` *`y0`*

Parameter	Value	Default
Species	`string`	-
yo	`real`	-

Summary Specification of mass fraction to be used for computing particle mass initial condition.

12.13.5. Number Represented

Scope: Crown Info

`Number Represented {=|are|is}` *`Num_represented`*

Parameter	Value	Default
Num_represented	`integer`	`1`

Summary Set number of physical particles represented by each particle node.

12.13.6. Porosity

Scope: Crown Info

`Porosity {=|are|is}` *`Porosity`*

Parameter	Value	Default
Porosity	`real`	-

Summary Set initial porosity for particle(s).

12.13.7. Track Particles For Tabular Output

Scope: Crown Info

Summary All particles of this description are added to the tabular output

Description Any particle matching this description will be output to the tabular data

12.13.8. Temperature

Scope: Crown Info

`Temperature {=|are|is}` *`Temperature`*

Parameter	Value	Default
Temperature	`real`	-

Summary Set initial temperature for particle(s).

12.13.9. X_Velocity

Scope: Crown Info

`X_Velocity {=|are|is}` *`X_velocity`*

Parameter	Value	Default
X_velocity	`real`	-

Summary Set initial x velocity for particle(s).

12.13.10. Y_Velocity

Scope: Crown Info

`Y_Velocity {=|are|is}` *`Y_velocity`*

Parameter	Value	Default
Y_velocity	`real`	-

Summary Set initial y velocity for particle(s).

12.13.11. Z_Velocity

Scope: Crown Info

`Z_Velocity {=|are|is}` *`Z_velocity`*

Parameter	Value	Default
Z_velocity	`real`	-

Summary Set initial z velocity for particle(s).

12.13.12. Particle Definition

Scope: Crown Info

`Particle Definition {=|are|is}` *`Definition`*

Parameter	Value	Default
Definition	`string`	-

Summary Set particle definition to be used for this block

12.14. TRUNK INFO

Scope: Particle Filled Tree

```
Begin Trunk Info Name

    Diameter {=|are|is} Diameter

    Insertion Time {=|are|is} InsertionTime

    Mass Represented {=|are|is} Mass_represented

    Mass_Fraction Species {=|are|is} y0

    Number Represented {=|are|is} Num_represented

    Porosity {=|are|is} Porosity

    Track Particles For Tabular Output

    Temperature {=|are|is} Temperature

    X_Velocity {=|are|is} X_velocity

    Y_Velocity {=|are|is} Y_velocity

    Z_Velocity {=|are|is} Z_velocity

    Particle Definition {=|are|is} Definition

End
```

Summary Block that defines data for a tree trunk.

12.14.1. Diameter

Scope: Trunk Info

```
Diameter {=|are|is} Diameter
```

Parameter	Value	Default
Diameter	real	-

Summary Set initial diameter of particle(s)

12.14.2. Insertion Time

Scope: Trunk Info

`Insertion Time {=|are|is}` *`InsertionTime`*

Parameter	Value	Default
InsertionTime	`real`	-

Summary Set time at which to insert particle; particle will be inserted at the first time step at or after this value.

12.14.3. Mass Represented

Scope: Trunk Info

`Mass Represented {=|are|is}` *`Mass_represented`*

Parameter	Value	Default
Mass_represented	`real`	-

Summary Set mass of physical particles represented by each particle node.

12.14.4. Mass_Fraction

Scope: Trunk Info

`Mass_Fraction` *`Species`* `{=|are|is}` *`y0`*

Parameter	Value	Default
Species	`string`	-
yo	`real`	-

Summary Specification of mass fraction to be used for computing particle mass initial condition.

12.14.5. Number Represented

Scope: Trunk Info

`Number Represented {=|are|is}` *`Num_represented`*

Parameter	Value	Default
Num_represented	`integer`	1

Summary Set number of physical particles represented by each particle node.

12.14.6. Porosity

Scope: Trunk Info

`Porosity {=|are|is}` *`Porosity`*

Parameter	Value	Default
Porosity	`real`	-

Summary Set initial porosity for particle(s).

12.14.7. Track Particles For Tabular Output

Scope: Trunk Info

Summary All particles of this description are added to the tabular output

Description Any particle matching this description will be output to the tabular data

12.14.8. Temperature

Scope: Trunk Info

`Temperature {=|are|is}` *`Temperature`*

Parameter	Value	Default
Temperature	`real`	-

Summary Set initial temperature for particle(s).

12.14.9. X_Velocity

Scope: Trunk Info

`X_Velocity {=|are|is}` *`X_velocity`*

Parameter	Value	Default
X_velocity	`real`	-

Summary Set initial x velocity for particle(s).

12.14.10. Y_Velocity

Scope: Trunk Info

`Y_Velocity {=|are|is}` *`Y_velocity`*

Parameter	Value	Default
Y_velocity	`real`	-

Summary Set initial y velocity for particle(s).

12.14.11. Z_Velocity

Scope: Trunk Info

`Z_Velocity {=|are|is}` *`Z_velocity`*

Parameter	Value	Default
Z_velocity	`real`	-

Summary Set initial z velocity for particle(s).

12.14.12. Particle Definition

Scope: Trunk Info

`Particle Definition {=|are|is}` *`Definition`*

Parameter	Value	Default
Definition	`string`	-

Summary Set particle definition to be used for this block

12.15. PARTICLE INFLOW BOUNDARY CONDITION ON SURFACE

Scope: Particle Region

`Begin Particle Inflow Boundary Condition On Surface` *`Surfacename`*

`Diameter {=|are|is}` *`Diameter`*

`Insertion Time {=|are|is}` *`InsertionTime`*

`Mass Represented {=|are|is}` *`Mass_represented`*

`Mass_Fraction` *`Species`* `{=|are|is}` *`y0`*

`Number Represented {=|are|is}` *`Num_represented`*

```
    Porosity {=|are|is} Porosity

    Track Particles For Tabular Output

    Temperature {=|are|is} Temperature

    X_Velocity {=|are|is} X_velocity

    Y_Velocity {=|are|is} Y_velocity

    Z_Velocity {=|are|is} Z_velocity

    Particle Definition {=|are|is} Definition

End
```

Summary Defines a boundary condition for particle inflow on a named surface of the mesh.

12.15.1. Diameter

Scope: Particle Inflow Boundary Condition On Surface

```
Diameter {=|are|is} Diameter
```

Parameter	Value	Default
Diameter	real	-

Summary Set initial diameter of particle(s)

12.15.2. Insertion Time

Scope: Particle Inflow Boundary Condition On Surface

```
Insertion Time {=|are|is} InsertionTime
```

Parameter	Value	Default
InsertionTime	real	-

Summary Set time at which to insert particle; particle will be inserted at the first time step at or after this value.

12.15.3. Mass Represented

Scope: Particle Inflow Boundary Condition On Surface

`Mass Represented {=|are|is}` *`Mass_represented`*

Parameter	Value	Default
Mass_represented	`real`	-

Summary Set mass of physical particles represented by each particle node.

12.15.4. Mass_Fraction

Scope: Particle Inflow Boundary Condition On Surface

`Mass_Fraction` *`Species`* `{=|are|is}` *`y0`*

Parameter	Value	Default
Species	`string`	-
yo	`real`	-

Summary Specification of mass fraction to be used for computing particle mass initial condition.

12.15.5. Number Represented

Scope: Particle Inflow Boundary Condition On Surface

`Number Represented {=|are|is}` *`Num_represented`*

Parameter	Value	Default
Num_represented	`integer`	1

Summary Set number of physical particles represented by each particle node.

12.15.6. Porosity

Scope: Particle Inflow Boundary Condition On Surface

`Porosity {=|are|is}` *`Porosity`*

Parameter	Value	Default
Porosity	`real`	-

Summary Set initial porosity for particle(s).

12.15.7. Track Particles For Tabular Output

Scope: Particle Inflow Boundary Condition On Surface

Summary All particles of this description are added to the tabular output

Description Any particle matching this description will be output to the tabular data

12.15.8. Temperature

Scope: Particle Inflow Boundary Condition On Surface

`Temperature {=|are|is}` *`Temperature`*

Parameter	Value	Default
Temperature	`real`	-

Summary Set initial temperature for particle(s).

12.15.9. X_Velocity

Scope: Particle Inflow Boundary Condition On Surface

`X_Velocity {=|are|is}` *`X_velocity`*

Parameter	Value	Default
X_velocity	`real`	-

Summary Set initial x velocity for particle(s).

12.15.10. Y_Velocity

Scope: Particle Inflow Boundary Condition On Surface

`Y_Velocity {=|are|is}` *`Y_velocity`*

Parameter	Value	Default
Y_velocity	`real`	-

Summary Set initial y velocity for particle(s).

12.15.11. Z_Velocity

Scope: Particle Inflow Boundary Condition On Surface

`Z_Velocity {=|are|is}` *`Z_velocity`*

Parameter	Value	Default
Z_velocity	`real`	-

Summary Set initial z velocity for particle(s).

12.15.12. Particle Definition

Scope: Particle Inflow Boundary Condition On Surface

`Particle Definition {=|are|is}` *`Definition`*

Parameter	Value	Default
Definition	`string`	-

Summary Set particle definition to be used for this block

12.16. PARTICLE INTERFACE

Scope: Particle Region

```
Begin Particle Interface InterfaceName

     ParticleInterfaceParameter {=|are|is} Value

    Particle Species SpeciesName StoichCoef

    Reaction Rate Parameter Name {=|are|is} RateParam

    Begin Particle Evaporation EvaporationName

    End

    Begin Particle Reaction ReactionName

    End

End
```

Summary Block that defines a specific particle-fluid interaction (e.g. reaction or evaporation)

12.16.1.

Scope: Particle Interface

ParticleInterfaceParameter {=|are|is} *Value*

Parameter	Value	Default
ParticleInterfaceParameter	{critical_porosity| critical_temperature| flame_temperature_for_transport |fuel_molecular_weight |heat_of_combustion| mass_diffusivity|prandtl_number |reference_heat_of_vaporization |reference_pressure| reference_temperature| schmidt_number|thermal_diffusivity |universal_gas_constant}	-
Value	real	-

Summary Constant value for the specified parameter (in consistent units).

12.16.2. Particle Species

Scope: Particle Interface

Particle Species *SpeciesName StoichCoef*

Parameter	Value	Default
SpeciesName	string	-
StoichCoef	real	-

Summary Particle species name and stoichiometric coefficient for this interface.

12.16.3. Reaction Rate Parameter

Scope: Particle Interface

Reaction Rate Parameter *Name* {=|are|is} *RateParam*

Parameter	Value	Default
Name	string	-
RateParam	real	-

Summary Specification of reaction rate parameters, E, A, latent heat.

Description Specification of reaction rate parameters, E, A, latent heat. Note that activation energies should be in per-mole units (e.g. kJ/mol or erg/mol, where energy units are consistent with the rest of the input file)

12.17. PARTICLE EVAPORATION

Scope: Particle Interface

```
Begin Particle Evaporation EvaporationName
     ParticleInterfaceParameter {=|are|is} Value
    Gas Species SpeciesName StoichCoef
End
```

Summary Sub-block that defines a specific particle evaporation. Maximum one per interface.

12.17.1.

Scope: Particle Evaporation

ParticleInterfaceParameter {=|are|is} *Value*

Parameter	Value	Default
ParticleInterfaceParameter	{critical_porosity| critical_temperature| flame_temperature_for_transport |fuel_molecular_weight |heat_of_combustion| mass_diffusivity|prandtl_number |reference_heat_of_vaporization |reference_pressure| reference_temperature| schmidt_number|thermal_diffusivity |universal_gas_constant}	-
Value	real	-

Summary Constant value for the specified parameter (in consistent units).

12.17.2. Gas Species

Scope: Particle Evaporation

Gas Species *SpeciesName StoichCoef*

Parameter	Value	Default
SpeciesName	string	-
StoichCoef	real	-

Summary GAS species name and stoichiometric coefficient for this interface.

12.18. PARTICLE REACTION

Scope: Particle Interface

```
Begin Particle Reaction ReactionName

     ParticleInterfaceParameter {=|are|is} Value

    Gas Species SpeciesName StoichCoef

    Oxidizer {=|are|is} SpeciesName

    Water {=|are|is} SpeciesName

End
```

Summary Sub-block that defines a specific particle reaction.

12.18.1.

Scope: Particle Reaction

ParticleInterfaceParameter {=|are|is} *Value*

Parameter	Value	Default
ParticleInterfaceParameter	{critical_porosity| critical_temperature| flame_temperature_for_transport |fuel_molecular_weight |heat_of_combustion| mass_diffusivity|prandtl_number |reference_heat_of_vaporization |reference_pressure| reference_temperature| schmidt_number|thermal_diffusivity |universal_gas_constant}	-
Value	real	-

Summary Constant value for the specified parameter (in consistent units).

12.18.2. Gas Species

Scope: Particle Reaction

`Gas Species` *`SpeciesName StoichCoef`*

Parameter	Value	Default
SpeciesName	`string`	-
StoichCoef	`real`	-

Summary GAS species name and stoichiometric coefficient for this interface.

12.18.3. Oxidizer

Scope: Particle Reaction

`Oxidizer {=|are|is}` *`SpeciesName`*

Parameter	Value	Default
SpeciesName	`string`	-

Summary Oxidizer species name for this interface (may be more than one).

12.18.4. Water

Scope: Particle Reaction

`Water {=|are|is}` *`SpeciesName`*

Parameter	Value	Default
SpeciesName	`string`	-

Summary Water species name for this interface.

12.19. PARTICLE MATERIAL

Scope: Particle Region

```
Begin Particle Material MaterialName

    ParticleMaterialProperty {=|are|is} EvalFunction...

End
```

Summary Block that defines material property data for a particle component

12.19.1.

Scope: Particle Material

ParticleMaterialProperty {=|are|is} *EvalFunction...*

Parameter	Value	Default
ParticleMaterialProperty	{absorptivity|density |emission_multiplier| emissivity|enthalpy_of_fusion |film_prandtl_number| freezing_temperature|molecular_weight |specific_heat|surface_tension| thermal_conductivity|viscosity}	-
EvalFunction	"string"...	0.0

Summary Constant/temperature dependent value for the specified property (in consistent units). The string entered can be a constant, or a function of time which will be evaluated using the stk expression evaluator. You must use 'T' as the temperature variable in this function.

12.20. PARTICLE OPEN BOUNDARY CONDITION ON SURFACE

Scope: Particle Region

```
Begin Particle Open Boundary Condition On Surface Surfacename
End
```

Summary Defines an open boundary condition and its interaction with particles.

12.21. PARTICLE SPRAY

Scope: Particle Region

```
Begin Particle Spray SprayName
    Arc_Angle {=|are|is} ArcAngle
    Arc_Ref {=|are|is} nRef1 nRef2 nRef3
    Center {=|are|is} Xcenter1 Xcenter2 Xcenter3
    Diameter Number Represented Vector {=|are|is} diamnumrepvec...
    Diameter {=|are|is} Diameter
```

Insertion Time {=|are|is} *InsertionTime*

Length Vector {=|are|is} $LengthVector_1$ $LengthVector_2$ $LengthVector_3$

Length {=|are|is} *Length*

Mass Represented {=|are|is} *Mass_represented*

Mass_Flow_Rate {=|are|is} *Mdot*

Mass_Fraction *Species* {=|are|is} *y0*

Normal Vector {=|are|is} $NormalVector_1$ $NormalVector_2$ $NormalVector_3$

Nozzle Radius {=|are|is} *Radius*

Number Representative Points {=|are|is} *NumRepPoints*

Number Represented Vector {=|are|is} *numrepvec...*

Number Represented {=|are|is} *Num_represented*

Particle Average_Diameter {=|are|is} *Velocity*

Particle Average_Velocity {=|are|is} *Velocity*

Particle Diameter Distribution Parameter *VarName* {=|are|is} *Value*

Particle Diameter Distribution Type {=|are|is} *DistributionType*

Particle Log File {=|are|is} *FieldName*

Particle Velocity Distribution Parameter *VarName* {=|are|is} *Value*

Particle Velocity Distribution Type {=|are|is} *DistributionType*

Porosity {=|are|is} *Porosity*

Spray_Angle {=|are|is} *Angle*

Spray_Angle_End {=|are|is} *Angle*

Spray_Angle_Start {=|are|is} *Angle*

Track Particles For Tabular Output

Temperature {=|are|is} *Temperature*

Width {=|are|is} *Width*

X_Velocity {=|are|is} *X_velocity*

Y_Velocity {=|are|is} *Y_velocity*

```
    Z_Velocity {=|are|is} Z_velocity
    Diameter Cutoff High {=|are|is} diamcutoffhigh
    Diameter Cutoff Low {=|are|is} diamcutofflow
    Particle Definition {=|are|is} Definition
End
```

Summary Block that defines data for a nozzle inflow.

12.21.1. Arc_Angle

Scope: Particle Spray

`Arc_Angle {=|are|is}` *ArcAngle*

Parameter	Value	Default
ArcAngle	`real`	-

Summary Angle (in degrees) for circular wedge shaped spray

12.21.2. Arc_Ref

Scope: Particle Spray

`Arc_Ref {=|are|is}` $nRef_1$ $nRef_2$ $nRef_3$

Parameter	Value	Default
nRef	`real_1 real_2 real_3`	-

Summary Reference vector within spray plane to measure angle from for circular wedge shaped spray

12.21.3. Center

Scope: Particle Spray

`Center {=|are|is}` $Xcenter_1$ $Xcenter_2$ $Xcenter_3$

Parameter	Value	Default
Xcenter	`real_1 real_2 real_3`	-

Summary (x,y,z) coordinates of center of spray nozzle plane

12.21.4. Diameter Number Represented Vector

Scope: Particle Spray

`Diameter Number Represented Vector {=|are|is}` *`diamnumrepvec...`*

Parameter	Value	Default
diamnumrepvec	`real...`	-

Summary Vector of diameters for function relating diameter of particle to number represented for fine control over number represented distributions

12.21.5. Diameter

Scope: Particle Spray

`Diameter {=|are|is}` *`Diameter`*

Parameter	Value	Default
Diameter	`real`	-

Summary Set initial diameter of particle(s)

12.21.6. Insertion Time

Scope: Particle Spray

`Insertion Time {=|are|is}` *`InsertionTime`*

Parameter	Value	Default
InsertionTime	`real`	-

Summary Set time at which to insert particle; particle will be inserted at the first time step at or after this value.

12.21.7. Length Vector

Scope: Particle Spray

`Length Vector {=|are|is}` $LengthVector_1$ $LengthVector_2$ $LengthVector_3$

Parameter	Value	Default
LengthVector	`real_1 real_2 real_3`	-

Summary Components (lx,ly,lz) of vector in the plane of the spray nozzle.

Description This vector is necessary for a rectangular spray nozzle to define the direction in which the LENGTH parameter is measured. This vector must be orthogonal to the spray normal direction. A third vector defining the direction for the WIDTH parameter is computed internally to be orthogonal to both the NORMAL and LENGTH vectors.

12.21.8. Length

Scope: Particle Spray

`Length {=|are|is}` *`Length`*

Parameter	Value	Default
Length	`real`	`0.0`

Summary Length of rectangular spray nozzle.

12.21.9. Mass Represented

Scope: Particle Spray

`Mass Represented {=|are|is}` *`Mass_represented`*

Parameter	Value	Default
Mass_represented	`real`	-

Summary Set mass of physical particles represented by each particle node.

12.21.10. Mass_Flow_Rate

Scope: Particle Spray

`Mass_Flow_Rate {=|are|is}` *`Mdot`*

Parameter	Value	Default
Mdot	`"string"`	`0.0`

Summary Mass flow rate of particles through the spray nozzle. The string entered can be a constant, or a function of time which will be evaluated using the stk expression evaluator. You must use 't' as the time variable in this function.

If the function contains commas or spaces it should be enclosed in quotes (e.g. "max(t,1)")

12.21.11. Mass_Fraction

Scope: Particle Spray

`Mass_Fraction` *`Species`* `{=|are|is}` *`y0`*

Parameter	Value	Default
Species	`string`	-
y0	`real`	-

Summary Specification of mass fraction to be used for computing particle mass initial condition.

12.21.12. Normal Vector

Scope: Particle Spray

`Normal Vector {=|are|is}` *NormalVector*$_1$ *NormalVector*$_2$ *NormalVector*$_3$

Parameter	Value	Default
NormalVector	`real_1 real_2 real_3`	-

Summary Components (nx,ny,nz) of vector normal to spray nozzle plane

12.21.13. Nozzle Radius

Scope: Particle Spray

`Nozzle Radius {=|are|is}` *Radius*

Parameter	Value	Default
Radius	`real`	-

Summary Radius of circular spray nozzle

12.21.14. Number Representative Points

Scope: Particle Spray

`Number Representative Points {=|are|is}` *NumRepPoints*

Parameter	Value	Default
NumRepPoints	`integer`	1000

Summary Number of possible insertion points used to represent this spray.

12.21.15. Number Represented Vector

Scope: Particle Spray

`Number Represented Vector {=|are|is}` *numrepvec...*

Parameter	Value	Default
numrepvec	`real...`	-

Summary Vector of number represented for function relating diameter of particle to number represented for fine control over number represented distribution

12.21.16. Number Represented

Scope: Particle Spray

`Number Represented {=|are|is}` *`Num_represented`*

Parameter	Value	Default
Num_represented	`integer`	`1`

Summary Set number of physical particles represented by each particle node.

12.21.17. Particle Average_Diameter

Scope: Particle Spray

`Particle Average_Diameter {=|are|is}` *`Velocity`*

Parameter	Value	Default
Velocity	`real`	-

Summary Mean diameter of particles entering through spray nozzle.

12.21.18. Particle Average_Velocity

Scope: Particle Spray

`Particle Average_Velocity {=|are|is}` *`Velocity`*

Parameter	Value	Default
Velocity	`real`	`0.0`

Summary Mean velocity of particles entering through spray nozzle

12.21.19. Particle Diameter Distribution Parameter

Scope: Particle Spray

`Particle Diameter Distribution Parameter` *`VarName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
VarName	`string`	-
Value	`real`	-

Summary Parameter for the probability distribution function for particle diameters.

Description The number and type of parameters needed depends on the type of the probability distribution being used.

```
CONSTANT      distribution requires parameter  VALUE
UNIFORM       distribution requires parameters MIN, MAX
NORMAL        distribution requires parameters MEAN, STDEV
LOGNORMAL     distribution requires parameters LOGMEAN, LOGSTDEV
              where LOGMEAN is the mean of the variable's
      natural logarithm (which is normally
      distributed), and LOGSTDEV is the standard
      deviation of the logarithms
ROSIN_RAMMLER distribution requires parameters X, Q
                  where X is related to the mean and Q
          controls the spread
```

Each parameter required should be entered in a separate command line.

PDF names with "_MASS" appended at the end indicate that the distribution is weighted by particle mass, i.e. that the PDF gives the fraction of total particle mass, rather than particle number, with the given diameter range.

12.21.20. Particle Diameter Distribution Type

Scope: Particle Spray

`Particle Diameter Distribution Type {=|are|is}` *`DistributionType`*

Parameter	Value	Default
DistributionType	{constant|lognormal|lognormal_mass|normal|normal_mass|rosin_rammler|rosin_rammler_mass|uniform|uniform_mass}	CONSTANT

Summary Probability distribution type for particle diameters

12.21.21. Particle Log File

Scope: Particle Spray

`Particle Log File {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	string	-

Summary Activates output to a file of a log of inserted particle nodes, including time injected, number represented, particle diameter, and particle mass

12.21.22. Particle Velocity Distribution Parameter

Scope: Particle Spray

`Particle Velocity Distribution Parameter` *`VarName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
VarName	`string`	-
Value	`real`	-

Summary Parameters for the probability distribution function for particle velocities.

Description The number and type of parameters needed depends on the type of the probability distribution being used.

```
CONSTANT      distribution requires parameter  VALUE
UNIFORM       distribution requires parameters MIN, MAX
NORMAL        distribution requires parameters MEAN, STDEV
LOGNORMAL     distribution requires parameters LOGMEAN, LOGSTDEV
                where LOGMEAN is the mean of the variable's
natural logarithm (which is normally
distributed), and LOGSTDEV is the standard
deviation of the logarithms
ROSIN_RAMMLER distribution requires parameters X, Q
                        where X is related to the mean and Q
        controls the spread
```

Each parameter required should be entered in a separate command line.

PDF names with "_MASS" appended at the end indicate that the distribution is weighted by particle mass, i.e. that the PDF gives the fraction of total particle mass, rather than particle number, with the given velocity range.

12.21.23. Particle Velocity Distribution Type

Scope: Particle Spray

`Particle Velocity Distribution Type {=|are|is}` *`DistributionType`*

Parameter	Value	Default
DistributionType	`{constant|lognormal|lognormal_mass|normal|normal_mass|rosin_rammler|rosin_rammler_mass|uniform|uniform_mass}`	CONSTANT

Summary Probability distribution type for particle velocities

12.21.24. Porosity

Scope: Particle Spray

`Porosity {=|are|is}` *`Porosity`*

Parameter	Value	Default
Porosity	`real`	-

Summary Set initial porosity for particle(s).

12.21.25. Spray_Angle

Scope: Particle Spray

`Spray_Angle {=|are|is}` *`Angle`*

Parameter	Value	Default
Angle	`real`	-

Summary Maximum spray angle (degrees)

12.21.26. Spray_Angle_End

Scope: Particle Spray

`Spray_Angle_End {=|are|is}` *`Angle`*

Parameter	Value	Default
Angle	`real`	`0.0`

Summary Maximum spray angle (degrees)

12.21.27. Spray_Angle_Start

Scope: Particle Spray

`Spray_Angle_Start {=|are|is}` *`Angle`*

Parameter	Value	Default
Angle	`real`	`0.0`

Summary Minimum spray angle (degrees); if >0 results in a hollow cone spray.

12.21.28. Track Particles For Tabular Output

Scope: Particle Spray

Summary All particles of this description are added to the tabular output

Description Any particle matching this description will be output to the tabular data

12.21.29. Temperature

Scope: Particle Spray

`Temperature {=|are|is}` *`Temperature`*

Parameter	Value	Default
Temperature	`real`	-

Summary Set initial temperature for particle(s).

12.21.30. Width

Scope: Particle Spray

`Width {=|are|is}` *`Width`*

Parameter	Value	Default
Width	`real`	`0.0`

Summary Width of rectangular spray nozzle.

12.21.31. X_Velocity

Scope: Particle Spray

`X_Velocity {=|are|is}` *`X_velocity`*

Parameter	Value	Default
X_velocity	`real`	-

Summary Set initial x velocity for particle(s).

12.21.32. Y_Velocity

Scope: Particle Spray

`Y_Velocity {=|are|is}` *`Y_velocity`*

Parameter	Value	Default
Y_velocity	`real`	-

Summary Set initial y velocity for particle(s).

12.21.33. Z_Velocity

Scope: Particle Spray

`Z_Velocity {=|are|is}` *`Z_velocity`*

Parameter	Value	Default
Z_velocity	`real`	-

Summary Set initial z velocity for particle(s).

12.21.34. Diameter Cutoff High

Scope: Particle Spray

`Diameter Cutoff High {=|are|is}` *`diamcutoffhigh`*

Parameter	Value	Default
diamcutoffhigh	`real`	`1000000.0`

Summary Upper Cutoff for particle Diameters.

12.21.35. Diameter Cutoff Low

Scope: Particle Spray

`Diameter Cutoff Low {=|are|is}` *`diamcutofflow`*

Parameter	Value	Default
diamcutofflow	`real`	`0.0`

Summary Lower Cutoff for particle Diameters.

12.21.36. Particle Definition

Scope: Particle Spray

`Particle Definition {=|are|is}` *`Definition`*

Parameter	Value	Default
Definition	`string`	-

Summary Set particle definition to be used for this block

12.22. PARTICLE TABULAR OUTPUT

Scope: Particle Region

```
Begin Particle Tabular Output Name

    Filename Prefix {=|are|is} FilenamePrefix

    Output Variable VariableName [ Component Comp ]

End
```

Summary Block that creates and defines options for tabular output of particle data.

Description Particle data is output in tabulated columns, with one file per particle containing the time history of that particle, and one variable per column. Only particles inserted from an insertion source marked with the `TRACK PARTICLES FOR TABULAR OUTPUT` command line are tracked. IDs are assigned automatically using a hash code.

12.22.1. Filename Prefix

Scope: Particle Tabular Output

`Filename Prefix {=|are|is}` *`FilenamePrefix`*

Parameter	Value	Default
FilenamePrefix	`string`	-

Summary Set filename prefix for output files.

Description Filenames will be created for data output files by concatenating the prefix with the particle ID number, followed by the .dat extension, e.g. `particle_data_1.dat`.

12.22.2. Output Variable

Scope: Particle Tabular Output

```
Output Variable VariableName [ Component Comp ]
```

Parameter	Value	Default
VariableName	string	-

Summary Designate a nodal particle variable for output.

12.23. PARTICLE WALL BOUNDARY CONDITION ON SURFACE

Scope: Particle Region

```
Begin Particle Wall Boundary Condition On Surface Surfacename

    Adhesion Model {=|are|is} particle_adhesion_model...

    Particle Surface Interaction Type {=|are|is}
    SurfaceInteractionType

    Reflection Model {=|are|is} particle_reflection_model...

    Particle_K_Pf_Crit {=|are|is} particle_K_PF_crit

    Particle_Contact_Angle {=|are|is} particle_contact_angle

    Particle_Maximum_Number_Fingers {=|are|is}
    particle_maximum_number_fingers

    Particle_Random_Graze_Angle_Maximum {=|are|is}
    particle_random_graze_angle_maximum

    Particle_Random_Graze_Angle_Minimum {=|are|is}
    particle_random_graze_angle_minimum

    Particle_Splash_Parameter_A {=|are|is} particle_splash_parameter_A


    Particle_Splash_Parameter_B {=|are|is} particle_splash_parameter_B


End
```

Summary Defines a wall boundary condition and its interaction with particles.

12.23.1. Adhesion Model

Scope: Particle Wall Boundary Condition On Surface

`Adhesion Model {=|are|is}` *`particle_adhesion_model...`*

Parameter	Value	Default
particle_adhesion_model	`string...`	-

Summary

Specifies the adhesion model to use with the combined surface interaction model. Options are "RESUSPEND_WICHNER", "STICK", or "NONE".

Each of these options is be followed by a list of arguments specific to that model. For details on the available options, refer to the particle section in the Math Models chapter of the user manual.

12.23.2. Particle Surface Interaction Type

Scope: Particle Wall Boundary Condition On Surface

`Particle Surface Interaction Type {=|are|is}` *`SurfaceInteractionType`*

Parameter	Value	Default
SurfaceInteractionType	`{combined|pass_through|rebound|shatter|stick|undefined}`	-

Summary

Type of interaction between particles and surface. Use "COMBINED" to specify separate reflection and adhesion models. All other options are DEPRECATED and are internally replaced with the equivalent "COMBINED" model.

12.23.3. Reflection Model

Scope: Particle Wall Boundary Condition On Surface

`Reflection Model {=|are|is}` *`particle_reflection_model...`*

Parameter	Value	Default
particle_reflection_model	`string...`	-

Summary

Specifies the reflection model to use with the combined surface interaction model. Options are "SHATTER_BROWN", "REBOUND", "REBOUND_YOON", or "NONE".

Each of these options is be followed by a list of arguments specific to that model. For example, the shatter model would be "REFLECTION MODEL = SHATTER_BROWN K_crit = 58 contact_angle = 45".

For details on the other options, refer to the particle section in the Math Models chapter of the user manual.

12.23.4. Particle_K_Pf_Crit

Scope: Particle Wall Boundary Condition On Surface

`Particle_K_Pf_Crit {=|are|is}` *particle_K_PF_crit*

Parameter	Value	Default
particle_K_PF_crit	`real`	-

Summary THIS INPUT FORMAT IS DEPRECATED. USE THE "REFLECTION MODEL" and "ADHESION MODEL" FORMAT.

12.23.5. Particle_Contact_Angle

Scope: Particle Wall Boundary Condition On Surface

`Particle_Contact_Angle {=|are|is}` *particle_contact_angle*

Parameter	Value	Default
particle_contact_angle	`real`	-

Summary THIS INPUT FORMAT IS DEPRECATED. USE THE "REFLECTION MODEL" and "ADHESION MODEL" FORMAT.

12.23.6. Particle_Maximum_Number_Fingers

Scope: Particle Wall Boundary Condition On Surface

`Particle_Maximum_Number_Fingers {=|are|is}`
particle_maximum_number_fingers

Parameter	Value	Default
particle_maximum_number_fingers	`integer`	-

Summary THIS INPUT FORMAT IS DEPRECATED. USE THE "REFLECTION MODEL" and "ADHESION MODEL" FORMAT.

12.23.7. Particle_Random_Graze_Angle_Maximum

Scope: Particle Wall Boundary Condition On Surface

`Particle_Random_Graze_Angle_Maximum {=|are|is}`
particle_random_graze_angle_maximum

Parameter	Value	Default
particle_random_graze_angle_maximum	`real`	-

Summary THIS INPUT FORMAT IS DEPRECATED. USE THE "REFLECTION MODEL" and "ADHESION MODEL" FORMAT.

12.23.8. Particle_Random_Graze_Angle_Minimum

Scope: Particle Wall Boundary Condition On Surface

`Particle_Random_Graze_Angle_Minimum {=|are|is}`
`particle_random_graze_angle_minimum`

Parameter	Value	Default
particle_random_graze_angle_minimum	real	-

Summary THIS INPUT FORMAT IS DEPRECATED. USE THE "REFLECTION MODEL" and "ADHESION MODEL" FORMAT.

12.23.9. Particle_Splash_Parameter_A

Scope: Particle Wall Boundary Condition On Surface

`Particle_Splash_Parameter_A {=|are|is}` *`particle_splash_parameter_A`*

Parameter	Value	Default
particle_splash_parameter_A	`real`	-

Summary THIS INPUT FORMAT IS DEPRECATED. USE THE "REFLECTION MODEL" and "ADHESION MODEL" FORMAT.

12.23.10. Particle_Splash_Parameter_B

Scope: Particle Wall Boundary Condition On Surface

`Particle_Splash_Parameter_B {=|are|is}` *`particle_splash_parameter_B`*

Parameter	Value	Default
particle_splash_parameter_B	`real`	-

Summary THIS INPUT FORMAT IS DEPRECATED. USE THE "REFLECTION MODEL" and "ADHESION MODEL" FORMAT.

12.24. RESTART DATA

Scope: Average Region

```
Begin Restart Data Label

    Additional Steps {=|are|is} List_of_steps...

    Additional Times {=|are|is} List_of_times...

    At Step n Option {=|are|is} m
```

```
    At Time Dt1 Option {=|are|is} Dt2
    At Wall Time Dt1 Option {=|are|is} Dt2
    Component Separator Character Option Separator
    Cycle Count {=|are|is} Count
    Database Name {=|are|is} StreamName
    Database Type {=|are|is} DatabaseTypes
    Debug Dump
    Decomposition Method {=|are|is} Method
    Exists Option1 Option2
    File Cycle Count {=|are|is} Count
    Input Database Name {=|are|is} StreamName
    Optional
    Output Database Name {=|are|is} StreamName
    Output On Signal {=|are|is} Signals
    Overlay Count {=|are|is} Count
    Overwrite Option1 Option2
    Property PropertyName {=|are|is} PropertyValue
    Restart {=|are|is} RestartOption
    Restart Time {=|are|is} Time
    Start Time {=|are|is} Start_time
    Synchronize Output
    Termination Time {=|are|is} Final_time
    Timestep Adjustment Interval {=|are|is} Nsteps
    Use Output Scheduler Timer_name
End
```

Summary Describes the data required to output and input restart data for the enclosing region.

12.24.1. Additional Steps

Scope: Restart Data

`Additional Steps {=|are|is}` *`List_of_steps...`*

Parameter	Value	Default
List_of_steps	`integer...`	-

Summary Additional simulation steps when output should occur.

12.24.2. Additional Times

Scope: Restart Data

`Additional Times {=|are|is}` *`List_of_times...`*

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

12.24.3. At Step

Scope: Restart Data

`At Step` *`n Option`* `{=|are|is}` *`m`*

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

12.24.4. At Time

Scope: Restart Data

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

12.24.5. At Wall Time

Scope: Restart Data

`At Wall Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`string`	-
Option	`{increment|interval}`	-
Dt2	`string`	-

Summary Write a restart file at a specific wall time since the start of the run. Time string format allows s, m, h, d for seconds, minutes, hours, days

12.24.6. Component Separator Character

Scope: Restart Data

`Component Separator Character` *`Option Separator`*

Parameter	Value	Default
Option	`{=|is}`	-
Separator	`string`	-

Summary The separator is the single character used to separate the output variable basename (e.g. "stress") from the suffices (e.g. "xx", "yy") when displaying the names of the individual variable components. For example, the default separator is "_", which results in names similar to "stress_xx", "stress_yy", ... "stress_zx". To eliminate the separator, specify an empty string ("") or NONE.

12.24.7. Cycle Count

Scope: Restart Data

`Cycle Count {=|are|is}` *`Count`*

Parameter	Value	Default
Count	`integer`	-

Summary Specify the number of restart steps which will be written to the restart database before previously written steps are overwritten. For example, if the cycle count is 5 and restart is written every 0.1 seconds, the restart system will write 0.1, 0.2, 0.3, 0.4, 0.5 to the database. It will then overwrite the first step with data from time 0.6, the second with time 0.7. At time 0.8, the database would contain data at times 0.6, 0.7, 0.8, 0.4, 0.5. Note that time will not necessarily be monotonically increasing on a database that specifies the cycle count.

12.24.8. Database Name

Scope: Restart Data

`Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The database containing the input and/or output restart data. If this analysis is being restarted, restart data will be read from this file. If the analysis is writing restart data, the data will be written to this file. It will be overwritten if it exists (after being read if applicable). If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. See also the 'Input Database' and 'Output Database' commands.

12.24.9. Database Type

Scope: Restart Data

`Database Type {=|are|is}` *`DatabaseTypes`*

Parameter	Value	Default
DatabaseTypes	`{catalyst|dof|dof_exodus|exodus|exodusii|generated|genesis|parallel_exodus}`	-

Summary The database type/format used for the restart file.

12.24.10. Debug Dump

Scope: Restart Data

Summary Specify whether the restart system will write the restart data immediately after reading the restart data if the run is restarting. The output data can be compared with the restart input data to determine whether they match.

12.24.11. Decomposition Method

Scope: Restart Data

`Decomposition Method {=|are|is}` *`Method`*

Parameter	Value	Default
Method	`{block|cyclic|external|geom_kway|hsfc|kway|kway_geom|linear|metis_sfc|random|rcb|rib}`	-

Summary The decomposition algorithm to be used to partition elements to each processor in a parallel run.

12.24.12. Exists

Scope: Restart Data

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{abort|add_suffix|append|overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis. "ADD_SUFFIX" will add a suffix to the file name and output to that file.

12.24.13. File Cycle Count

Scope: Restart Data

`File Cycle Count {=|are|is}` *`Count`*

Parameter	Value	Default
Count	`integer`	-

Summary Each restart dump will be written to a separate file suffixed with A,B, ... The count specifies how many separate files are used before the cycle repeats. For example, if "FILE CYCLE COUNT = 3" is specified, the restart dumps would be written to file-A.rs, file-B.rs, file-C.rs, file-A.rs, ... The maximum value for the cycle count is 26.

12.24.14. Input Database Name

Scope: Restart Data

`Input Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The database containing the input restart data. If this analysis is being restarted, restart data will be read from this file. See also the 'Database' and 'Output Database' commands.

12.24.15. Optional

Scope: Restart Data

Summary The database will be read if it exists, but it is not an error if there is no restart database to read for this region during a restarted analysis.

12.24.16. Output Database Name

Scope: Restart Data

`Output Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The database containing the output restart data. If the analysis is writing restart data, the data will be written to this file. It will be overwritten if it exists. See also the 'Database' and 'Input Database' commands.

12.24.17. Output On Signal

Scope: Restart Data

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

12.24.18. Overlay Count

Scope: Restart Data

`Overlay Count {=|are|is}` *`Count`*

Parameter	Value	Default
Count	`integer`	-

Summary

Specify the number of restart outputs which will be overlayed on top of the last written step. For example, if restarts are being output every 0.1 seconds and the overlay count is specified as 2, then restart will write times 0.1 to step 1 of the database. It will then write 0.2 and 0.3 also to step 1. It will then increment the database step and write 0.4 to step 2; overlay 0.5 and 0.6 on step 2... At the end of the analysis, assuming it runs to completion, the database would have times 0.3, 0.6, 0.9, ... However, if there were a problem during the analysis, the last step on the database would contain an intermediate step.

12.24.19. Overwrite

Scope: Restart Data

`Overwrite` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{false|no|off|on|true|yes}`	-

Summary

(DEPRECATED, Use EXISTS) Specify whether the restart database should be overwritten if it exists. The default behavior is to overwrite unless this command is specified in the restart block and either off, false, or no is specified.

12.24.20. Property

Scope: Restart Data

`Property` *`PropertyName`* `{=|are|is}` *`PropertyValue`*

Parameter	Value	Default
PropertyName	`string`	-
PropertyValue	`string`	-

Summary

Define a database property named "PropertyName" with the value "PropertyValue". If PropertyValue consists of all digits, it will define an integer property. If PropertyValue is "true" or "yes" or "false" or "no", it will define a logical property; otherwise it will define a string property. If PropertyName consists of multiple strings, they will be concated together with "_" separating the individual words. Suppported properties are typically database dependent; Current properties are: COMPRESSION_LEVEL = [0..9] COMPRESSION_SHUFFLE = true|false|on|off FILE_TYPE = netcdf4 (forces use of

netcdf-4 hdf5-based file) INTEGER_SIZE_DB = 4|8 INTEGER_SIZE_API = 4|8 LOGGING = true|false|on|off MAX_NAME_LENGTH = value

12.24.21. Restart

Scope: Restart Data

`Restart {=|are|is}` *`RestartOption`*

Parameter	Value	Default
RestartOption	`{auto|automatic}`	-

Summary Specify automatic restart file read.

Description **NOTE:** This command must be placed at the Sierra scope of the input file.

Specify that the analysis should be restarted from the last common time on all restart databases for each Region in the analysis. In addition to this line command, each Region in the analysis (strictly, only the region(s) that will be restarted) must have a restart block specifying the database to read the restart state data.

By default, use of this command will not cause output files (e.g., results, history, heartbeat, restart) to be overwritten. Instead output files will be written with the same basename and the suffix `-s000*`. Common visualization packages are written to handle this file organization gracefully in order for the user to view all results seamlessly.

12.24.22. Restart Time

Scope: Restart Data

`Restart Time {=|are|is}` *`Time`*

Parameter	Value	Default
Time	`real`	-

Summary Specify restart file read at a specified time.

Description **NOTE:** This command must be placed at the Sierra scope of the input file.

Specify the time that the analysis will be restarted. In addition to this line command, each Region in the analysis (strictly, only the region(s) that will be restarted) must have a restart block specifying the database to read the restart state data. The restart 'time' must be greater than zero and less than or equal to the termination time.

By default, use of this command will cause previous output files (e.g., results, history, heartbeat, restart) to be overwritten. If this command is chosen, the onus is placed on the user to ensure that previous output files are not overwritten.

12.24.23. Start Time

Scope: Restart Data

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

12.24.24. Synchronize Output

Scope: Restart Data

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

12.24.25. Termination Time

Scope: Restart Data

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

12.24.26. Timestep Adjustment Interval

Scope: Restart Data

`Timestep Adjustment Interval {=|are|is} Nsteps`

Parameter	Value	Default
Nsteps	integer	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

12.24.27. Use Output Scheduler

Scope: Restart Data

`Use Output Scheduler Timer_name`

Parameter	Value	Default
Timer_name	string	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

12.25. RESULTS OUTPUT

Scope: Average Region

```
Begin Results Output Label
    Additional Steps {=|are|is} List_of_steps...
    Additional Times {=|are|is} List_of_times...
    At Step n Option {=|are|is} m
    At Time Dt1 Option {=|are|is} Dt2
    Component Separator Character {=|are|is} Separator
    Database Name {=|are|is} StreamName
    Database Type {=|are|is} DatabaseType
    Edge [ VariableList... ]
    Edge Variables {=|are|is} [ VariableList... ]
    Element [ VariableList... ]
    Element Variables {=|are|is} [ VariableList... ]
    Enable Large Ids
```

Exclude {=|are|is} [*ElementBlockList...*]

Exists *Option1 Option2*

Face [*VariableList...*]

Face Variables {=|are|is} [*VariableList...*]

Global [*Variables...*]

Global Variables {=|are|is} [*Variables...*]

Include {=|are|is} [*ElementBlockList...*]

Nodal [*VariableList...*]

Nodal Variables {=|are|is} [*VariableList...*]

Node [*VariableList...*]

Node Variables {=|are|is} [*VariableList...*]

Nodeset [*VariableList...*]

Nodeset Variables {=|are|is} [*VariableList...*]

Output Mesh {=|are|is} *OutputMesh*

Output On Signal {=|are|is} *Signals*

Overwrite *Option1 Option2*

Property *PropertyName* {=|are|is} *PropertyValue*

Sideset [*VariableList...*]

Sideset Variables {=|are|is} [*VariableList...*]

Start Time {=|are|is} *Start_time*

Surface [*VariableList...*]

Surface Variables {=|are|is} [*VariableList...*]

Synchronize Output

Termination Time {=|are|is} *Final_time*

Timeseries Name {=|are|is} *filename*

Timestep Adjustment Interval {=|are|is} *Nsteps*

Title

Use Output Scheduler *Timer_name*

Begin Catalyst *Label*

End

```
End
```

Summary Describes the location and type of the output stream used for outputting results for the enclosing region.

12.25.1. Additional Steps

Scope: Results Output

```
Additional Steps {=|are|is} List_of_steps...
```

Parameter	Value	Default
List_of_steps	`integer...`	-

Summary Additional simulation steps when output should occur.

12.25.2. Additional Times

Scope: Results Output

```
Additional Times {=|are|is} List_of_times...
```

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

12.25.3. At Step

Scope: Results Output

```
At Step n Option {=|are|is} m
```

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

12.25.4. At Time

Scope: Results Output

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

12.25.5. Component Separator Character

Scope: Results Output

`Component Separator Character {=|are|is}` *`Separator`*

Parameter	Value	Default
Separator	`string`	-

Summary The separator is the single character used to separate the output variable basename (e.g. "stress") from the suffices (e.g. "xx", "yy") when displaying the names of the individual variable components. For example, the default separator is "_", which results in names similar to "stress_xx", "stress_yy", ... "stress_zx". To eliminate the separator, specify an empty string ("") or NONE.

12.25.6. Database Name

Scope: Results Output

`Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The base name of the database containing the output results. If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. If this line is omitted, then a filename will be created from the basename of the input file with a ".e" suffix appended.

12.25.7. Database Type

Scope: Results Output

`Database Type {=|are|is}` *`DatabaseType`*

Parameter	Value	Default
DatabaseType	`{catalyst|dof|dof_exodus|exodus|exodusii|generated|genesis|parallel_exodus}`	-

Summary The database type/format to be used for the output results.

12.25.8. Edge

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Edge variables are not supported for all database types.

12.25.9. Edge Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Edge variables are not supported for all database types.

12.25.10. Element

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities"

12.25.11. Element Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities"

12.25.12. Enable Large Ids

Scope: Results Output

Summary Enable 64 bit entity IDs for output

12.25.13. Exclude

Scope: Results Output

Summary Specify that the results file will only contain a subset of the element blocks in the analysis model. The element_block_list lists only the blocks which will not be output to the results database.

12.25.14. Exists

Scope: Results Output

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{abort|add_suffix|append|overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis. "ADD_SUFFIX" will add a -s???? suffix where the ???? is replaced by a sequential number starting at 0002.

12.25.15. Face

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

12.25.16. Face Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

12.25.17. Global

Scope: Results Output

Summary Define the global variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

12.25.18. Global Variables

Scope: Results Output

Summary Define the global variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

12.25.19. Include

Scope: Results Output

Summary Specify that the results file will only contain a subset of the element blocks in the analysis model. The element_block_list lists only the blocks which will be output to the results database.

12.25.20. Nodal

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

12.25.21. Nodal Variables

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

12.25.22. Node

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

12.25.23. Node Variables

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

12.25.24. Nodeset

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Nodeset variables are not supported for all database types.

12.25.25. Nodeset Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Nodeset variables are not supported for all database types.

12.25.26. Output Mesh

Scope: Results Output

`Output Mesh {=|are|is}` *`OutputMesh`*

Parameter	Value	Default
OutputMesh	`{exposed surface|refined|unrefined}`	-

Summary Use this command to turn on "unrefined" as the output mesh. The default behavior is "refined", in which field variables are output on the current mesh, which may have been refined (either uniformly or adaptively) or had its topology altered in some way (e.g., dynamic load balancing) with respect to the original mesh read from the input file. By specifying "Output Mesh = unrefined", all output variables are output only on the original mesh objects read from the input file.

12.25.27. Output On Signal

Scope: Results Output

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

12.25.28. Overwrite

Scope: Results Output

`Overwrite` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{false|no|off|on|true|yes}`	-

Summary (DEPRECATED, Use EXISTS) Specify whether the database should be overwritten if it exists. The default behavior is to overwrite unless this command is specified in the output block and either off, false, or no is specified.

12.25.29. Property

Scope: Results Output

`Property` *`PropertyName`* `{=|are|is}` *`PropertyValue`*

Parameter	Value	Default
PropertyName	`string`	-
PropertyValue	`string`	-

Summary Define a database property named "PropertyName" with the value "PropertyValue". If PropertyValue consists of all digits, it will define an integer property. If PropertyValue is "true" or "yes" or "false" or "no", it will define a logical property; otherwise it will define a string property. Suppported properties are typically database dependent; Current properties are: COMPRESSION_LEVEL = [0..9] (off) COMPRESSION_SHUFFLE = true|false|on|off (off) FILE_TYPE = netcdf4 (forces use of netcdf-4 hdf5-based file) (netcdf3) INTEGER_SIZE_DB = 4|8 (4) INTEGER_SIZE_API = 4|8 (4) REAL_SIZE_DB = 4|8 (8 is default) LOGGING = true|false|on|off (off) MAX_NAME_LENGTH = value (32)

12.25.30. Sideset

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

12.25.31. Sideset Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

12.25.32. Start Time

Scope: Results Output

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

12.25.33. Surface

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

12.25.34. Surface Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

12.25.35. Synchronize Output

Scope: Results Output

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

12.25.36. Termination Time

Scope: Results Output

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

12.25.37. Timeseries Name

Scope: Results Output

`Timeseries Name {=|are|is}` *`filename`*

Parameter	Value	Default
filename	`string`	-

Summary Optionally specify a filename for a timeseries file that outputs the root database filename in the order that they are written. This is useful when running on large numbers of processors with many mesh-mods that cause simple disk operations to hang.

12.25.38. Timestep Adjustment Interval

Scope: Results Output

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	`integer`	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

12.25.39. Title

Scope: Results Output

Summary Specify the title to be used for this specific output block.

12.25.40. Use Output Scheduler

Scope: Results Output

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	`string`	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

13. AVERAGE REGION

The following commands are found in the Average Region scope of the input file.

13.1. AVERAGE REGION

Scope: Fuego Procedure

```
Begin Average Region Regionname

    Use Finite Element Model ModelName [ Model Coordinates Are
    Nodal_variable_name ]

    Begin Heartbeat Label

    End

    Begin History Output Label

    End

    Begin Restart Data Label

    End

    Begin Results Output Label

    End

    Begin Solution Options OptionsName

    End

End
```

Summary Contains the commands needed to execute an analysis on this region.

13.1.1. Use Finite Element Model

Scope: Average Region

```
Use Finite Element Model ModelName [ Model Coordinates Are
Nodal_variable_name ]
```

Parameter	Value	Default
ModelName	`string`	-

Summary Associates a predefined finite element model with this region.

13.2. HISTORY OUTPUT

Scope: Average Region

```
Begin History Output Label
    Additional Steps {=|are|is} List_of_steps...
    Additional Times {=|are|is} List_of_times...
    At Step n Option {=|are|is} m
    At Time Dt1 Option {=|are|is} Dt2
    Database Name {=|are|is} StreamName
    Database Type {=|are|is} DatabaseTypes
    Element [ VariableList... ]
    Exists Option1 Option2
    Face [ VariableList... ]
    Global [ Variables... ]
    Nodal [ VariableList... ]
    Node [ VariableList... ]
    Nodeset [ VariableList... ]
    Output On Signal {=|are|is} Signals
    Overwrite Option1 Option2
    Property PropertyName {=|are|is} PropertyValue
    Start Time {=|are|is} Start_time
    Synchronize Output
    Termination Time {=|are|is} Final_time
```

```
        Timestep Adjustment Interval {=|are|is} Nsteps
        Title
        Use Output Scheduler Timer_name
        Variable {=|are|is} Option Variable_list...
    End
```

Summary Describes the location and type of the output stream used for outputting history for the enclosing region.

13.2.1. Additional Steps

Scope: History Output

```
Additional Steps {=|are|is} List_of_steps...
```

Parameter	Value	Default
List_of_steps	integer...	-

Summary Additional simulation steps when output should occur.

13.2.2. Additional Times

Scope: History Output

```
Additional Times {=|are|is} List_of_times...
```

Parameter	Value	Default
List_of_times	real...	-

Summary Additional simulation times when output should occur.

13.2.3. At Step

Scope: History Output

```
At Step n Option {=|are|is} m
```

Parameter	Value	Default
n	integer	-
Option	{increment|interval}	-
m	integer	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

13.2.4. At Time

Scope: History Output

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

13.2.5. Database Name

Scope: History Output

`Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The base name of the database containing the output history. If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. If this line is omitted, then a filename will be created from the basename of the input file with a ".h" suffix appended.

13.2.6. Database Type

Scope: History Output

`Database Type {=|are|is}` *`DatabaseTypes`*

Parameter	Value	Default
DatabaseTypes	`{catalyst|dof|dof_exodus|exodus|exodusii|generated|genesis|parallel_exodus}`	-

Summary The database type/format to be used for the output history.

13.2.7. Element

Scope: History Output

Summary Define the element variables that should be written to the history database. The syntax is: "element internal_name at element id as DBname" or "element internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

13.2.8. Exists

Scope: History Output

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{abort|add_suffix|append|overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis. "ADD_SUFFIX" will add a -s???? suffix where the ???? is replaced by a sequential number starting at 0002.

13.2.9. Face

Scope: History Output

Summary Define the face variables that should be written to the history database. The syntax is: "face internal_name at face id as DBname" or "face internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

13.2.10. Global

Scope: History Output

Summary Define the global/reduction variables that should be written to the history database. The syntax is: "global internal_name as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

13.2.11. Nodal

Scope: History Output

Summary Define the nodal variables that should be written to the history database. The syntax is: "nodal internal_name at node id as DBname" or "nodal internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

13.2.12. Node

Scope: History Output

Summary Define the nodal variables that should be written to the history database. The syntax is: "node internal_name at node id as DBname" or "node internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

13.2.13. Nodeset

Scope: History Output

Summary Define the nodeset variables that should be written to the history database. The syntax is: "nodeset internal_name at node id as DBname" or "nodeset internal_name nearest location X, Y, Z as DBname".

Where internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

13.2.14. Output On Signal

Scope: History Output

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

13.2.15. Overwrite

Scope: History Output

`Overwrite` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{false|no|off|on|true|yes}`	-

Summary (DEPRECATED, Use EXISTS) Specify whether the database should be overwritten if it exists. The default behavior is to overwrite unless this command is specified in the output block and either off, false, or no is specified.

13.2.16. Property

Scope: History Output

`Property` *`PropertyName`* `{=|are|is}` *`PropertyValue`*

Parameter	Value	Default
PropertyName	`string`	-
PropertyValue	`string`	-

Summary Define a database property named "PropertyName" with the value "PropertyValue". If PropertyValue consists of all digits, it will define an integer property. If PropertyValue is "true" or "yes" or "false" or "no", it will define a logical property; otherwise it will define a string property. Suppported properties are typically database dependent; Some history-related properties are: VARIABLE_NAME_CASE = upper|lower MAX_NAME_LENGTH = value (32)

13.2.17. Start Time

Scope: History Output

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

13.2.18. Synchronize Output

Scope: History Output

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

13.2.19. Termination Time

Scope: History Output

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

13.2.20. Timestep Adjustment Interval

Scope: History Output

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	`integer`	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

13.2.21. Title

Scope: History Output

Summary Specify the title to be used for this specific output block.

13.2.22. Use Output Scheduler

Scope: History Output

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	`string`	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

13.2.23. Variable

Scope: History Output

`Variable {=|are|is}` *`Option Variable_list...`*

Parameter	Value	Default
Option	`{edge|element|face|global|nodal|node}`	-
Variable_list	`string...`	-

Summary Define the variables that should be written to the history database. The syntax is: "variable = entity internal_name at entity id as DBname" or "variable = entity internal_name nearest location X, Y, Z as DBname" or "variable = entity internal_name at location X, Y, Z as DBname".

Where entity is 'node', 'element', 'face', or 'edge'; internal_name is the name of the variable in the Sierra application; and DBname is the name as it should appear on the history database.

13.3. RESTART DATA

Scope: Average Region

Begin Restart Data *Label*

Additional Steps {=|are|is} *List_of_steps...*

Additional Times {=|are|is} *List_of_times...*

At Step *n Option* {=|are|is} *m*

At Time *Dt1 Option* {=|are|is} *Dt2*

At Wall Time *Dt1 Option* {=|are|is} *Dt2*

Component Separator Character *Option Separator*

Cycle Count {=|are|is} *Count*

Database Name {=|are|is} *StreamName*

Database Type {=|are|is} *DatabaseTypes*

Debug Dump

Decomposition Method {=|are|is} *Method*

Exists *Option1 Option2*

File Cycle Count {=|are|is} *Count*

Input Database Name {=|are|is} *StreamName*

Optional

Output Database Name {=|are|is} *StreamName*

Output On Signal {=|are|is} *Signals*

Overlay Count {=|are|is} *Count*

Overwrite *Option1 Option2*

Property *PropertyName* {=|are|is} *PropertyValue*

Restart {=|are|is} *RestartOption*

Restart Time {=|are|is} *Time*

Start Time {=|are|is} *Start_time*

Synchronize Output

Termination Time {=|are|is} *Final_time*

Timestep Adjustment Interval {=|are|is} *Nsteps*

Use Output Scheduler *Timer_name*

```
End
```

Summary Describes the data required to output and input restart data for the enclosing region.

13.3.1. Additional Steps

Scope: Restart Data

```
Additional Steps {=|are|is} List_of_steps...
```

Parameter	Value	Default
List_of_steps	`integer...`	-

Summary Additional simulation steps when output should occur.

13.3.2. Additional Times

Scope: Restart Data

```
Additional Times {=|are|is} List_of_times...
```

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

13.3.3. At Step

Scope: Restart Data

```
At Step n Option {=|are|is} m
```

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

13.3.4. At Time

Scope: Restart Data

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

13.3.5. At Wall Time

Scope: Restart Data

`At Wall Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`string`	-
Option	`{increment|interval}`	-
Dt2	`string`	-

Summary Write a restart file at a specific wall time since the start of the run. Time string format allows s, m, h, d for seconds, minutes, hours, days

13.3.6. Component Separator Character

Scope: Restart Data

`Component Separator Character` *`Option Separator`*

Parameter	Value	Default
Option	`{=|is}`	-
Separator	`string`	-

Summary The separator is the single character used to separate the output variable basename (e.g. "stress") from the suffices (e.g. "xx", "yy") when displaying the names of the individual variable components. For example, the default separator is "_", which results in names similar to "stress_xx", "stress_yy", ... "stress_zx". To eliminate the separator, specify an empty string ("") or NONE.

13.3.7. Cycle Count

Scope: Restart Data

`Cycle Count {=|are|is}` *`Count`*

Parameter	Value	Default
Count	`integer`	-

Summary Specify the number of restart steps which will be written to the restart database before previously written steps are overwritten. For example, if the cycle count is 5 and restart is written every 0.1 seconds, the restart system will write 0.1, 0.2, 0.3, 0.4, 0.5 to the database. It will then overwrite the first step with data from time 0.6, the second with time 0.7. At time 0.8, the database would contain data at times 0.6, 0.7, 0.8, 0.4, 0.5. Note that time will not necessarily be monotonically increasing on a database that specifies the cycle count.

13.3.8. Database Name

Scope: Restart Data

`Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The database containing the input and/or output restart data. If this analysis is being restarted, restart data will be read from this file. If the analysis is writing restart data, the data will be written to this file. It will be overwritten if it exists (after being read if applicable). If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. See also the 'Input Database' and 'Output Database' commands.

13.3.9. Database Type

Scope: Restart Data

`Database Type {=|are|is}` *`DatabaseTypes`*

Parameter	Value	Default
DatabaseTypes	`{catalyst|dof|dof_exodus|exodus|exodusii|generated|genesis|parallel_exodus}`	-

Summary The database type/format used for the restart file.

13.3.10. Debug Dump

Scope: Restart Data

Summary Specify whether the restart system will write the restart data immediately after reading the restart data if the run is restarting. The output data can be compared with the restart input data to determine whether they match.

13.3.11. Decomposition Method

Scope: Restart Data

`Decomposition Method {=|are|is}` *`Method`*

Parameter	Value	Default
Method	`{block|cyclic|external|geom_kway|hsfc|kway|kway_geom|linear|metis_sfc|random|rcb|rib}`	-

Summary The decomposition algorithm to be used to partition elements to each processor in a parallel run.

13.3.12. Exists

Scope: Restart Data

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{abort|add_suffix|append|overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis. "ADD_SUFFIX" will add a suffix to the file name and output to that file.

13.3.13. File Cycle Count

Scope: Restart Data

`File Cycle Count {=|are|is}` *`Count`*

Parameter	Value	Default
Count	`integer`	-

Summary Each restart dump will be written to a separate file suffixed with A,B, ... The count specifies how many separate files are used before the cycle repeats. For example, if "FILE CYCLE COUNT = 3" is specified, the restart dumps would be written to file-A.rs, file-B.rs, file-C.rs, file-A.rs, ... The maximum value for the cycle count is 26.

13.3.14. Input Database Name

Scope: Restart Data

`Input Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The database containing the input restart data. If this analysis is being restarted, restart data will be read from this file. See also the 'Database' and 'Output Database' commands.

13.3.15. Optional

Scope: Restart Data

Summary The database will be read if it exists, but it is not an error if there is no restart database to read for this region during a restarted analysis.

13.3.16. Output Database Name

Scope: Restart Data

`Output Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The database containing the output restart data. If the analysis is writing restart data, the data will be written to this file. It will be overwritten if it exists. See also the 'Database' and 'Input Database' commands.

13.3.17. Output On Signal

Scope: Restart Data

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

13.3.18. Overlay Count

Scope: Restart Data

`Overlay Count {=|are|is}` *`Count`*

Parameter	Value	Default
Count	`integer`	-

Summary Specify the number of restart outputs which will be overlayed on top of the last written step. For example, if restarts are being output every 0.1 seconds and the overlay count is specified as 2, then restart will write times 0.1 to step 1 of the database. It will then write 0.2 and 0.3 also to step 1. It will then increment the database step and write 0.4 to step 2; overlay 0.5 and 0.6 on step 2... At the end of the analysis, assuming it runs to completion, the database would have times 0.3, 0.6, 0.9, ... However, if there were a problem during the analysis, the last step on the database would contain an intermediate step.

13.3.19. Overwrite

Scope: Restart Data

`Overwrite` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{false|no|off|on|true|yes}`	-

Summary (DEPRECATED, Use EXISTS) Specify whether the restart database should be overwritten if it exists. The default behavior is to overwrite unless this command is specified in the restart block and either off, false, or no is specified.

13.3.20. Property

Scope: Restart Data

`Property` *`PropertyName`* `{=|are|is}` *`PropertyValue`*

Parameter	Value	Default
PropertyName	`string`	-
PropertyValue	`string`	-

Summary Define a database property named "PropertyName" with the value "PropertyValue". If PropertyValue consists of all digits, it will define an integer property. If PropertyValue is "true" or "yes" or "false" or "no", it will define a logical property; otherwise it will define a string property. If PropertyName consists of multiple strings, they will be concated together with "_" separating the individual words. Suppported properties are typically database dependent; Current properties are: COMPRESSION_LEVEL = [0..9] COMPRESSION_SHUFFLE = true|false|on|off FILE_TYPE = netcdf4 (forces use of netcdf-4 hdf5-based file) INTEGER_SIZE_DB = 4|8 INTEGER_SIZE_API = 4|8 LOGGING = true|false|on|off MAX_NAME_LENGTH = value

13.3.21. Restart

Scope: Restart Data

`Restart {=|are|is}` *`RestartOption`*

Parameter	Value	Default
RestartOption	`{auto|automatic}`	-

Summary Specify automatic restart file read.

Description **NOTE:** This command must be placed at the Sierra scope of the input file.

Specify that the analysis should be restarted from the last common time on all restart databases for each Region in the analysis. In addition to this line command, each Region in the analysis (strictly, only the region(s) that will be restarted) must have a restart block specifying the database to read the restart state data.

By default, use of this command will not cause output files (e.g., results, history, heartbeat, restart) to be overwritten. Instead output files will be written with the same basename and the suffix `-s000*`. Common visualization packages are written to handle this file organization gracefully in order for the user to view all results seamlessly.

13.3.22. Restart Time

Scope: Restart Data

`Restart Time {=|are|is}` *`Time`*

Parameter	Value	Default
Time	`real`	-

Summary Specify restart file read at a specified time.

Description **NOTE:** This command must be placed at the Sierra scope of the input file.

Specify the time that the analysis will be restarted. In addition to this line command, each Region in the analysis (strictly, only the region(s) that will be restarted) must have a restart block specifying the database to read the restart state data. The restart 'time' must be greater than zero and less than or equal to the termination time.

By default, use of this command will cause previous output files (e.g., results, history, heartbeat, restart) to be overwritten. If this command is chosen, the onus is placed on the user to ensure that previous output files are not overwritten.

13.3.23. Start Time

Scope: Restart Data

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

13.3.24. Synchronize Output

Scope: Restart Data

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where

one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

13.3.25. Termination Time

Scope: Restart Data

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

13.3.26. Timestep Adjustment Interval

Scope: Restart Data

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	`integer`	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

13.3.27. Use Output Scheduler

Scope: Restart Data

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	`string`	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

13.4. RESULTS OUTPUT

Scope: Average Region

Begin Results Output *Label*

Additional Steps {=|are|is} *List_of_steps...*

Additional Times {=|are|is} *List_of_times...*

At Step *n Option* {=|are|is} *m*

At Time *Dt1 Option* {=|are|is} *Dt2*

Component Separator Character {=|are|is} *Separator*

Database Name {=|are|is} *StreamName*

Database Type {=|are|is} *DatabaseType*

Edge [*VariableList...*]

Edge Variables {=|are|is} [*VariableList...*]

Element [*VariableList...*]

Element Variables {=|are|is} [*VariableList...*]

Enable Large Ids

Exclude {=|are|is} [*ElementBlockList...*]

Exists *Option1 Option2*

Face [*VariableList...*]

Face Variables {=|are|is} [*VariableList...*]

Global [*Variables...*]

Global Variables {=|are|is} [*Variables...*]

Include {=|are|is} [*ElementBlockList...*]

Nodal [*VariableList...*]

Nodal Variables {=|are|is} [*VariableList...*]

Node [*VariableList...*]

Node Variables {=|are|is} [*VariableList...*]

Nodeset [*VariableList...*]

Nodeset Variables {=|are|is} [*VariableList...*]

Output Mesh {=|are|is} *OutputMesh*

Output On Signal {=|are|is} *Signals*

```
    Overwrite Option1 Option2

    Property PropertyName {=|are|is} PropertyValue

    Sideset [ VariableList...  ]

    Sideset Variables {=|are|is} [ VariableList...  ]

    Start Time {=|are|is} Start_time

    Surface [ VariableList...  ]

    Surface Variables {=|are|is} [ VariableList...  ]

    Synchronize Output

    Termination Time {=|are|is} Final_time

    Timeseries Name {=|are|is} filename

    Timestep Adjustment Interval {=|are|is} Nsteps

    Title

    Use Output Scheduler Timer_name

    Begin Catalyst Label

    End

End
```

Summary Describes the location and type of the output stream used for outputting results for the enclosing region.

13.4.1. Additional Steps

Scope: Results Output

```
Additional Steps {=|are|is} List_of_steps...
```

Parameter	Value	Default
List_of_steps	integer...	-

Summary Additional simulation steps when output should occur.

13.4.2. Additional Times

Scope: Results Output

`Additional Times {=|are|is}` *`List_of_times...`*

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

13.4.3. At Step

Scope: Results Output

`At Step` *`n Option`* `{=|are|is}` *`m`*

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

13.4.4. At Time

Scope: Results Output

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

13.4.5. Component Separator Character

Scope: Results Output

`Component Separator Character {=|are|is}` *`Separator`*

Parameter	Value	Default
Separator	`string`	-

Summary The separator is the single character used to separate the output variable basename (e.g. "stress") from the suffices (e.g. "xx", "yy") when displaying the names of the individual variable components. For example, the default separator is "_", which results in names similar to "stress_xx", "stress_yy", ... "stress_zx". To eliminate the separator, specify an empty string ("") or NONE.

13.4.6. Database Name

Scope: Results Output

`Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The base name of the database containing the output results. If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. If this line is omitted, then a filename will be created from the basename of the input file with a ".e" suffix appended.

13.4.7. Database Type

Scope: Results Output

`Database Type {=|are|is}` *`DatabaseType`*

Parameter	Value	Default
DatabaseType	`{catalyst|dof|dof_exodus|exodus|exodusii|generated|genesis|parallel_exodus}`	-

Summary The database type/format to be used for the output results.

13.4.8. Edge

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Edge variables are not supported for all database types.

13.4.9. Edge Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Edge variables are not supported for all database types.

13.4.10. Element

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities"

13.4.11. Element Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same

line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities"

13.4.12. Enable Large Ids

Scope: Results Output

Summary Enable 64 bit entity IDs for output

13.4.13. Exclude

Scope: Results Output

Summary Specify that the results file will only contain a subset of the element blocks in the analysis model. The element_block_list lists only the blocks which will not be output to the results database.

13.4.14. Exists

Scope: Results Output

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{= | is}`	-
Option2	`{abort | add_suffix | append | overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis. "ADD_SUFFIX" will add a -s???? suffix where the ???? is replaced by a sequential number starting at 0002.

13.4.15. Face

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

13.4.16. Face Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

13.4.17. Global

Scope: Results Output

Summary Define the global variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

13.4.18. Global Variables

Scope: Results Output

Summary Define the global variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

13.4.19. Include

Scope: Results Output

Summary Specify that the results file will only contain a subset of the element blocks in the analysis model. The element_block_list lists only the blocks which will be output to the results database.

13.4.20. Nodal

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

13.4.21. Nodal Variables

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

13.4.22. Node

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

13.4.23. Node Variables

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

13.4.24. Nodeset

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Nodeset variables are not supported for all database types.

13.4.25. Nodeset Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Nodeset variables are not supported for all database types.

13.4.26. Output Mesh

Scope: Results Output

`Output Mesh {=|are|is}` *`OutputMesh`*

Parameter	Value	Default
OutputMesh	`{exposed surface|refined|unrefined}`	-

Summary Use this command to turn on "unrefined" as the output mesh. The default behavior is "refined", in which field variables are output on the current mesh, which may have been refined (either uniformly or adaptively) or had its topology altered in some way (e.g., dynamic load balancing) with respect to the original mesh read from the input file. By specifying "Output Mesh = unrefined", all output variables are output only on the original mesh objects read from the input file.

13.4.27. Output On Signal

Scope: Results Output

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

13.4.28. Overwrite

Scope: Results Output

`Overwrite` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{false|no|off|on|true|yes}`	-

Summary (DEPRECATED, Use EXISTS) Specify whether the database should be overwritten if it exists. The default behavior is to overwrite unless this command is specified in the output block and either off, false, or no is specified.

13.4.29. Property

Scope: Results Output

`Property` *`PropertyName`* `{=|are|is}` *`PropertyValue`*

Parameter	Value	Default
PropertyName	`string`	-
PropertyValue	`string`	-

Summary Define a database property named "PropertyName" with the value "PropertyValue". If PropertyValue consists of all digits, it will define an integer property. If PropertyValue is "true" or "yes" or "false" or "no", it will define a logical property; otherwise it will define a string property. Suppported properties are typically database dependent; Current properties are: COMPRESSION_LEVEL = [0..9] (off) COMPRESSION_SHUFFLE = true|false|on|off (off) FILE_TYPE = netcdf4 (forces use of netcdf-4 hdf5-based file) (netcdf3) INTEGER_SIZE_DB = 4|8 (4) INTEGER_SIZE_API = 4|8 (4) REAL_SIZE_DB = 4|8 (8 is default) LOGGING = true|false|on|off (off) MAX_NAME_LENGTH = value (32)

13.4.30. Sideset

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

13.4.31. Sideset Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

13.4.32. Start Time

Scope: Results Output

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

13.4.33. Surface

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

13.4.34. Surface Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

13.4.35. Synchronize Output

Scope: Results Output

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

13.4.36. Termination Time

Scope: Results Output

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

13.4.37. Timeseries Name

Scope: Results Output

`Timeseries Name {=|are|is}` *`filename`*

Parameter	Value	Default
filename	`string`	-

Summary Optionally specify a filename for a timeseries file that outputs the root database filename in the order that they are written. This is useful when running on large numbers of processors with many mesh-mods that cause simple disk operations to hang.

13.4.38. Timestep Adjustment Interval

Scope: Results Output

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	`integer`	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

13.4.39. Title

Scope: Results Output

Summary Specify the title to be used for this specific output block.

13.4.40. Use Output Scheduler

Scope: Results Output

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	string	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

13.5. SOLUTION OPTIONS

Scope: Average Region

`Begin Solution Options` *`OptionsName`*

`Activate Acoustic Compressibility Algorithm`

`Activate Equation` *`Equations`*

`Activate Full Surface Cvfem Gradient Operator For Muscl Scheme`

`Activate Lighthill Tensor Postprocessing`

`Activate Species Enthalpy Calculations`

`Activate Viscous Dissipation Source Term`

`Compute Steady Solution Using Pseudo Transient Method`

`Coordinate System {=|are|is}` *`CoordSys`*

`First Order Upwind Factor {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

`Fix Pressure To` *`FixedPressure`* `At A Single Node`

`Hybrid Upwind Factor {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

`Hybrid Upwind Method {=|are|is}` *`HybridMethod`* `[ For Equation` *`Equations`* `]`

`Hybrid Upwind Shift {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

`Hybrid Upwind Width {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

`Include Continuity Residual Term [ With Diagnostics ]`

`Maximum Number Of Continuity_Momentum Nonlinear Iterations {=|are|is}` *`Number`*

Maximum Number Of Energy_Species Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Gas_Solid_Momentum Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Kepsilon Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Komega Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Ksgs Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Mixture Fraction Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Solid Phase Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Soot Nuclei Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Species Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of Species_Product Nonlinear Iterations {=|are|is} *Number*

Maximum Number Of V2F Nonlinear Iterations {=|are|is} *Number*

Maximum Wall Time {=|are|is} *WallTime* Hours

Minimum Number Of Nonlinear Iterations {=|are|is} *Number*

Nonlinear Residual Norm Tolerance {=|are|is} *Tolerance* [For Equation *Equations*]

Nonlinear Stabilization Method {=|are|is} *NSOMethod* [For Equation *Equations*]

Omit Density Time Derivative In Continuity Equation [For *OmitSteps* Steps And Blend In Over *BlendSteps* Steps]

Output Nonlinear Residual Field For Equation *Equations* As *ResName* [On Output Block *BlockName*]

Periodic Constant Momentum Body Source Term {=|are|is} $ConstSrc_1$ $ConstSrc_2$ $ConstSrc_3$

Projection Method {=|are|is} *ProjectionMethod* Smoothing [With *ProjectionScaling* Scaling]

Randomize Pressure

Skip Pressure Update If Continuity Solve Fails

Source Term Function {=|are|is} *FuncStr* For Equation *Equation* [*SpeciesName*]

Source Term Subroutine {=|are|is} *Subroutine* For Equation *Equations* [*SpeciesName*]

Stop Simulation If Peak Velocity Exceeds *MaxVel*

Under Relax *Equations* By *Urf* [With Implicit Term]

Under Relax Momentum By *Urf*

Under Relax Pressure By *Urf*

Under Relax Solid_Momentum By *Urf*

Under Relax Temperature_Extraction By *Urf*

Upwind Limiter {=|are|is} *UpwindLimiter* [For Equation *Equations*]

Upwind Method {=|are|is} *UpwindMethod* [For Equation *Equations*]

Use Equation Solver *SolverName* For Equation *Equations*

Use External Continuity Source

Use External Energy Source

Use External Mixture_Fraction Source

Use External Momentum Source

Use External Soot_Mass_Fraction Source

Use External Species Source

Use Lumped Velocity Density Interpolation

Use Radiation Source From External Region [Using Classic Linearization]

Use Skew Symmetric Central Operator [For Equation *Equations*]

Begin Acoustic Transfer Output *DefinitionName*

End

Begin Beam Radiation Boundary Specification *DefinitionName*

End

Begin Buoyancy Model Specification *BuoyModelName*

End

```
    Begin Edc Model Specification EdcSpecName

    End

    Begin Mesh Motion Specification DefinitionName

    End

    Begin Multiphase Model Specification DefinitionName

    End

    Begin Point Source DefinitionName

    End

    Begin Rad Transport Spectral Model Specification DefinitionName

    End

    Begin Radiation Transport Equation Model Specification
    RadModelName

    End

    Begin Time Integration Specification TimeIntSpecName

    End

    Begin Turbulence Model Specification TurbSpecName

    End

    Begin Vof Model Specification DefinitionName

    End

End
```

Summary Specify information regarding the governing equations to be solved.

13.5.1. Activate Acoustic Compressibility Algorithm

Scope: Solution Options

Summary Variable thermodynamic pressure

Description This option will allow for closed system pressurization either through heat-up or inflow of fluid.

The algorithm will add the substantial derivative of pressure, $\frac{\partial p}{\partial t} + u_j \frac{\partial p}{\partial x_j}$, to the laminar or turbulent enthalpy transport equation and to the laminar temperature transport equation. Additionally, the viscous work term $u_i \frac{\partial \tau_{ij}}{\partial x_j}$ will be added to the turbulent enthalpy equation. An implicit term in the continuity solve is added through the time density derivative. As such, Cantera support is required. The convective terms within the continuity solve are neglected.

Caveats for this model:

1) The Cantera material model evaluator must be used.

2) The initial pressure and any boundary condition pressures must be specified with respect to the datum pressure.

If a zero datum pressure is specified, then all initial and boundary pressures will be in absolute units. If this is a coupled structural simulation, then the surface traction due to this pressure will need to be counteracted with a load on the "back side" that is equivalent to the ambient pressure in absolute units.

If a non-zero datum pressure is specified, then all initial and boundary pressures will be in relative units with respect to this datum. Pressure can then be thought of as a gauge pressure with respect to the datum. The "back side" load in structural simulations must be set accordingly. (For example, if the datum is set equal to the external ambient pressure, 1 atm, and the initial pressure is set to zero, then the initial surface traction force due to pressure will be zero and no "back side" load due to the ambient pressure should be specified.)

13.5.2. Activate Equation

Scope: Solution Options

`Activate Equation` *`Equations`*

Parameter	Value	Default
Equations	{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}	-

Summary Activate the specified equation.

13.5.3. Activate Full Surface Cvfem Gradient Operator For Muscl Scheme

Scope: Solution Options

Summary Use full stencil for gradient used in MUSCL convection operator

Description The default gradient operator for the MUSCL scheme is the edge-based stencil. This option keeps integration points at the subcontrol surface points.

13.5.4. Activate Lighthill Tensor Postprocessing

Scope: Solution Options

Summary Postprocesses the nodal divergence of the Lighthill tensor

Description This calculates the nodal divergence of the Lighthill tensor, used for acoustic analysis.

13.5.5. Activate Species Enthalpy Calculations

Scope: Solution Options

Summary Enables calculation of species enthalpy

Description This forces the calculation of species enthalpy, needed primarily for coupled Fuego-Aria problems.

13.5.6. Activate Viscous Dissipation Source Term

Scope: Solution Options

Summary Add viscous dissipation source term

Description For low speed viscous dissipation effects, this source term will provide the viscous work source term in the static enthalpy equation. This source term is a subset of the full acoustically compressible source term option, however, the substantial pressure derivative is omitted.

13.5.7. Compute Steady Solution Using Pseudo Transient Method

Scope: Solution Options

Summary Compute a steady-state solution using the pseudo-transient method (time march to steady solution).

Description The solution will march forward in time until either the stopping time is reached or the steady convergence criterion is met. Convergence to steady state is detected when all equations meet their nonlinear residual norm tolerances after the first nonlinear iteration, since this will only occur as the solution stops changing between time steps. The nonlinear residual norm tolerances should be set small enough to prevent false positives. Also make sure the simulation time is set to be fairly large, to prevent a premature end to the simulation before convergence is achieved.

If you are using solution control, then you also need to test for a region parameter to stop the simulation. In your `PARAMETERS FOR TRANSIENT` solution control block, add the line (assuming your Fuego region is called `fuego_region`):

```
CONVERGED WHEN "fuego_region.REGION_STEADY_STATE == 1"
```

13.5.8. Coordinate System

Scope: Solution Options

`Coordinate System {=|are|is}` *`CoordSys`*

Parameter	Value	Default
CoordSys	`{2d|3d|xaxi|yaxi}`	3D

Summary Specify the coordinate system.

13.5.9. First Order Upwind Factor

Scope: Solution Options

`First Order Upwind Factor {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
Value	`"string"`	`1.0`

Summary First-order upwind factor, 0 extless x extgreater 1.

Description This value specifies the explicit upwind blending between pure upwind and the chosen convection operator, e.g., UPW*(firstOrderUpwind) + (1-firstOrderUpwind)*(blendedUpwindCentral).

where UPW is the pure first order upwind value and blendedUpwindCentral is a blend between the selected upwind method and central difference operator based on the local cell Peclet number (see Hybrid Upwind Factor line command). The value can be a time dependent string function.

Values for individual equation sets may be set using optional token. Using both (in either order):

- FIRST ORDER UPWIND FACTOR = String
- FIRST ORDER UPWIND FACTOR = String FOR EQUATION Equations

Will result in the particular equation set to specified value while all others set to general value.

13.5.10. Fix Pressure To

Scope: Solution Options

`Fix Pressure To` *`FixedPressure`* `At A Single Node`

Parameter	Value	Default
FixedPressure	`real`	`0.0`

Summary Sets a dirichlet for pressure at a single arbitrary node. This is required for a well posed pressure equation if none of the boundaries specify pressure (e.g. open).

13.5.11. Hybrid Upwind Factor

Scope: Solution Options

`Hybrid Upwind Factor {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
Value	`"string"`	`1.0`

Summary Hybrid upwinding factor.

Description The upwind schemes are blended with a centered scheme. The HYBRID UPWIND FACTOR is a multiplier against the cell Peclet number used in the switching scheme (see First Order Upwind Factor line command).

- A HYBRID UPWIND FACTOR = 0.0 results in all centered.
- A HYBRID UPWIND FACTOR = 1.0 results in default hybrid.
- A HYBRID UPWIND FACTOR » 1.0 results in all upwind.

Values for individual equation sets may be set using optional token. Using both (in either order):

- HYBRID UPWIND FACTOR = String
- HYBRID UPWIND FACTOR = String FOR EQUATION Equations

Will result in the particular equation set to specified value while all others set to general value. The value can be specified as a time dependent string function or a constant.

13.5.12. Hybrid Upwind Method

Scope: Solution Options

`Hybrid Upwind Method {=|are|is}` *`HybridMethod`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
HybridMethod	`{blending|tanh}`	`BLENDING`

Summary Hybriding method between central and upwind using Peclet number

Description BLENDING and TANH approaches are currently supported.

Function χ determines the ratio between user-chosen upwind (χ) and central (1-χ) operators. The need for an upwind operator is affected by cell-Peclet number.

BLENDING uses HYBRID UPWIND FACTOR (ζ) and the function is, $\chi = \frac{(\zeta Pe)^2}{5+(\zeta Pe)^2}$.

TAHH follows hyperbolic tangent profile between χ and Pe. It uses shift p (HYBRID UPWIND SHIFT) and width w (HYBRID UPWIND WIDTH) parameters as,

$\chi = \frac{1}{2}[1 + tanh(\frac{Pe-p}{w})]$. Tanh is centered ($\chi = 0.5$) when Peclet number is at the shifting factor p. Width determines how fast χ changes with Peclet number as follows:

- χ =0.5 at Pe=p.
- χ =0.8808 and 0.1192 at Pe=p+w and p-w.
- χ =0.9820 and 0.0180 at Pe=p+2w and p-2w.
- χ =0.9975 and 0.0025 at Pe=p+3w and p-3w.
- χ =0.9997 and 0.0003 at Pe=p+4w and p-4w.

TANH allows users to effectively remove upwind contribution for lower Pe. In the other extreme, one can enforce user-chosen upwind at all Pe if p < 0.0 and w « 1.0 (ex> p=-1.0, w=1e-10).

13.5.13. Hybrid Upwind Shift

Scope: Solution Options

`Hybrid Upwind Shift {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
Value	`"string"`	`0.0`

Summary Shifting factor for TANH hybrid approach. Can be specified as a time dependent string function or a constant.

Description (see HYBRID UPWIND METHOD description)

13.5.14. Hybrid Upwind Width

Scope: Solution Options

`Hybrid Upwind Width {=|are|is}` *`Value`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
Value	`"string"`	`1.0`

Summary Width factor for TANH hybrid approach

Description Minimum value for this parameter is 1e-10. (see HYBRID UPWIND METHOD description) Can be specified as a time dependent string function or a constant value.

13.5.15. Include Continuity Residual Term

Scope: Solution Options

Summary Include the continuity residual term in transport equations

Description Continuity is not exactly satisfied during the momentum solve for variable density flows since the mass flux is lagged while the density is updated with new properties. Including the continuity error in momentum can keep the momentum prediction better behaved. The residual should be on the order of the linear solver tolerance for other equations, but including the term can also make the other solves more robust to a bad continuity solve.

This term is always included when using VOF or a deforming mesh.

13.5.16. Maximum Number Of Continuity_Momentum Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Continuity_Momentum Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the momentum/continuity solve.

13.5.17. Maximum Number Of Energy_Species Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Energy_Species Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the energy-species grouping.

13.5.18. Maximum Number Of Gas_Solid_Momentum Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Gas_Solid_Momentum Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the gas/solid momentum sets of equations.

13.5.19. Maximum Number Of Kepsilon Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Kepsilon Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the k-epsilon turbulence model equations grouping.

13.5.20. Maximum Number Of Komega Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Komega Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the k-omega turbulence model equations grouping.

13.5.21. Maximum Number Of Ksgs Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Ksgs Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the ksgs turbulence model equations grouping.

13.5.22. Maximum Number Of Mixture Fraction Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Mixture Fraction Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the mixture fraction equations grouping.

13.5.23. Maximum Number Of Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the time step of the Fuego region.

13.5.24. Maximum Number Of Solid Phase Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Solid Phase Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the solid momentum/continuity sets of equations.

13.5.25. Maximum Number Of Soot Nuclei Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Soot Nuclei Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	`integer`	`1`

Summary Maximum number of nonlinear iterations to take within the soot nuclei equations grouping.

13.5.26. Maximum Number Of Species Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Species Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	integer	1

Summary Maximum number of nonlinear iterations for the species equations. If the EDC product transport feature is active, then the SPECIES_PRODUCT nonlinear iteration count should be set instead.

13.5.27. Maximum Number Of Species_Product Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of Species_Product Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	integer	1

Summary Maximum number of nonlinear iterations to take within the Species/EDC_Product grouping. This is only used if the EDC model is active and the EDC product transport feature is being used.

13.5.28. Maximum Number Of V2F Nonlinear Iterations

Scope: Solution Options

`Maximum Number Of V2F Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	integer	1

Summary Maximum number of nonlinear iterations to take within the v2f turbulence model equations grouping.

13.5.29. Maximum Wall Time

Scope: Solution Options

`Maximum Wall Time {=|are|is}` *`WallTime`* `Hours`

Parameter	Value	Default
WallTime	real	Infinite

Summary Specify a maximum wall time to let the simulation end gracefully and output before slurm kills it.

13.5.30. Minimum Number Of Nonlinear Iterations

Scope: Solution Options

`Minimum Number Of Nonlinear Iterations {=|are|is}` *`Number`*

Parameter	Value	Default
Number	integer	1

Summary Minimum number of nonlinear iterations to take within the time step of the Fuego region.

13.5.31. Nonlinear Residual Norm Tolerance

Scope: Solution Options

`Nonlinear Residual Norm Tolerance {=|are|is}` *`Tolerance`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
Tolerance	real	1.0e-15

Summary Nonlinear convergence tolerance within a time step in the Fuego region.

Values for individual equation sets may be set using the optional token. Using both (in either order):

```
NONLINEAR RESIDUAL NORM TOLERANCE = {Real}
NONLINEAR RESIDUAL NORM TOLERANCE = {Real} FOR EQUATION {Equations}
```

Will result in the particular equation set to specified value while all others set to general value.

13.5.32. Nonlinear Stabilization Method

Scope: Solution Options

`Nonlinear Stabilization Method {=|are|is}` *`NSOMethod`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
NSOMethod	{commutation_error|none|pointwise_residual_error}	NO_NSO

Summary Specify a artificial viscosity stabilization; default is NONE.

Description Values for individual equation sets may be set using optional token. Using both (in either order):

- NSO METHOD = NSOMethod

- NSO METHOD = NSOMethod FOR EQUATION Equations

Will result in the particular equation set to specified value while all others set to general value.

13.5.33. Omit Density Time Derivative In Continuity Equation

Scope: Solution Options

Summary Remove density time derivative in continuity equation

Description Remove the density time derivative from the continuity equation. This feature is required for closed boundary flows with accumulation.

The optional arguments let you omit it for a certain number of timesteps at the start of the simulation, then gradually include it over a number of steps.

13.5.34. Output Nonlinear Residual Field For Equation

Scope: Solution Options

```
Output Nonlinear Residual Field For Equation Equations As ResName [ On
Output Block BlockName ]
```

Parameter	Value	Default
Equations	{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}	-
ResName	string	-

Summary Generates output of nonlinear residuals for the requested equation.

Description Provide nonlinear residual for output for specified equation. If the optional output block name is specified, then the residual will only be written to that output block.

13.5.35. Periodic Constant Momentum Body Source Term

Scope: Solution Options

`Periodic Constant Momentum Body Source Term {=|are|is}` *ConstSrc*$_1$ *ConstSrc*$_2$ *ConstSrc*$_3$

Parameter	Value	Default
ConstSrc	`real_1 real_2 real_3`	-

Summary Add constant body force due to periodic config

Description For periodic BCS, commonly a constant body force is applied to drive the flow. This line command allows one to provide a constant body force in three dimensions. If more complex sources are needed, the user sub source term procedure is required.

13.5.36. Projection Method

Scope: Solution Options

`Projection Method {=|are|is}` *ProjectionMethod* `Smoothing [ With` *ProjectionScaling* `Scaling ]`

Parameter	Value	Default
ProjectionMethod	`{filtered|fourth_order|second_order|stabilized|zeroth_order}`	-

Summary Specify choice of projection method.

Description The smoothing choice may include zeroth, second, or fourth order. No smoothing (zeroth) may allow pressure-velocity decoupling.

The scaling term may be specified. This scaling term is related to the factorization approximation to the inverse of the momentum matrix.

Time step scaling may show results that are sensitive to the chosen simulation dt at coarse meshes. This error should vanish as the pressure field approached a linear shape, or refinement is performed. Note that characteristic scaling also has the same error, however, its manifestation is less obvious.

The stabilized option uses a fourth order smoothing term and characteristic scaling along with an additional dt stabilizing term.

In general, the stabilized and "fourth order smoothing" timestep scaling allows for larger time steps. Characteristic scaling seems to limit CFL to below unity, presumably due to stability loss during nodal projection, i.e., splitting error is (I-tau*A)G(deltaP).

"Momentum Scaling", uses the diagonal of the momentum equation as the scaling term. While the leading order term with this method will be similar to the timestep scaling scheme, it can sometimes offer better stability since it also includes effects from the other terms in the momentum equation.

13.5.37. Randomize Pressure

Scope: Solution Options

Summary Set a random pressure field for initial guess

Description Randomize the initial guess to the linear solve for pressure. The randomization is imposed after the nonlinear residual is computed.

13.5.38. Skip Pressure Update If Continuity Solve Fails

Scope: Solution Options

Summary Do not update the pressure field or mdot if the continuity solve fails

Description If the continuity solve fails the resulting pressure delta may be large or non-physical. Activating this option skips the pressure update and mdot update when the solver fails. Repeated solver failures should be watched for in the log file.

This is a beta feature.

13.5.39. Source Term Function

Scope: Solution Options

`Source Term Function {=|are|is}` *`FuncStr`* `For Equation` *`Equation`* `[` *`SpeciesName`* `]`

Parameter	Value	Default
FuncStr	`"string"`	-
Equation	`{conserved_enthalpy|continuity|edc_product|enthalpy|mixture_fraction|nuclei|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot|species|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_momentum|x_solid_momentum|y_momentum|y_solid_momentum|z_momentum|z_solid_momentum}`	-

---

Summary Source term string function to use for the given equation. Registered variables with aliases include time (t), spatial coordinates (x,y,z), velocity (u,v,w), density (rho), and pressure (p). Additionally, any valid global variable or nodal variable can be used with its full name. Vector variables, like mass fraction, must be indexed numerically (e.g. "mass_fraction[3]")

The function string must be enclosed in quotes if it has spaces or commas. For example: Source Term Function for x_momentum = "min(1, 0.1*t)"

13.5.40. Source Term Subroutine

Scope: Solution Options

`Source Term Subroutine {=|are|is}` *`Subroutine`* `For Equation` *`Equations`* `[` *`SpeciesName`* `]`

Parameter	Value	Default
Subroutine	`string`	-
Equations	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

---

Summary Source term user subroutine for the given equation. This is often useful in verification studies where one wishes to use a manufactured solution and must provide source terms for various governing equations.

13.5.41. Stop Simulation If Peak Velocity Exceeds

Scope: Solution Options

`Stop Simulation If Peak Velocity Exceeds` *`MaxVel`*

Parameter	Value	Default
MaxVel	`real`	`infinity`

Summary Abort the simulation if velocities get too large.

Description By default Fuego will continue time stepping as the simulation diverges and will go until velocities overflow or solvers start returning NaN or Inf.

If you want it to stop sooner than that, you can set a peak velocity magnitude to abort at.

13.5.42. Under Relax

Scope: Solution Options

`Under Relax` *`Equations`* `By` *`Urf`* `[ With Implicit Term ]`

Parameter	Value	Default
Equations	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-
Urf	`"string"`	`1.0`

Summary Under relaxation factor for the given equation.

Description Implicit relaxation is applied to the momentum equations. Explicit relaxation is applied to the pressure update. Transport equations are relaxed explicitly unless the "WITH IMPLICI TERM" option is used.

Under-relaxation can be a constant value or a function of time (t). If the function used has spaces or commas, it should be enclosed in quotes. The value will be internally clipped between 1e-6 and 1.

13.5.43. Under Relax Momentum By

Scope: Solution Options

`Under Relax Momentum By` *`Urf`*

Parameter	Value	Default
Urf	`"string"`	-

Summary Under relaxation factor for the momentum equations.

Under-relaxation can be a constant value or a function of time (t). If the function used has spaces or commas, it should be enclosed in quotes. The value will be internally clipped between 1e-6 and 1.

13.5.44. Under Relax Pressure By

Scope: Solution Options

`Under Relax Pressure By` *`Urf`*

Parameter	Value	Default
Urf	`"string"`	-

Summary Under relaxation factor for the pressure. This is equivalent to specifying an URF on continuity, and is provided for backward compatibility.

Under-relaxation can be a constant value or a function of time (t). If the function used has spaces or commas, it should be enclosed in quotes. The value will be internally clipped between 1e-6 and 1.

13.5.45. Under Relax Solid_Momentum By

Scope: Solution Options

`Under Relax Solid_Momentum By` *`Urf`*

Parameter	Value	Default
Urf	`"string"`	-

Summary Under relaxation factor for the solid-phase momentum equations.

Under-relaxation can be a constant value or a function of time (t). If the function used has spaces or commas, it should be enclosed in quotes. The value will be internally clipped between 1e-6 and 1.

13.5.46. Under Relax Temperature_Extraction By

Scope: Solution Options

`Under Relax Temperature_Extraction By` *`Urf`*

Parameter	Value	Default
Urf	`"string"`	-

Summary Under relaxation factor for the temperature extraction from enthalpy

Description Relax the temperature computed from the enthalpy. This gives a temperature that is not entirely consistent with the current state (composition and enthalpy), and will destroy time-accuracy unless sufficient Picard loops are taken. However, it may be useful for steady-state computations where species and energy equations are not being coupled strongly or solved accurately.

Under-relaxation can be a constant value or a function of time (t). If the function used has spaces or commas, it should be enclosed in quotes. The value will be internally clipped between 1e-6 and 1.

13.5.47. Upwind Limiter

Scope: Solution Options

`Upwind Limiter {=|are|is}` *`UpwindLimiter`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
UpwindLimiter	{minmod | none | superbee | van_albada | van_leer}	SUPERBEE

Summary Specify a limiter for convection operator; default is SUPERBEE.

Description Limiter functions are valid only for the MUSCL scheme.

Values for individual equation sets may be set using optional token. Using both (in either order):

- UPWIND LIMITER = UpwindLimiter
- UPWIND LIMITER = UpwindLimiter FOR EQUATION Equations

Will result in the particular equation set to specified value while all others set to general value.

Note: Rotational invariance of the code is not expected while using a limiter function.

13.5.48. Upwind Method

Scope: Solution Options

`Upwind Method {=|are|is}` *`UpwindMethod`* `[ For Equation` *`Equations`* `]`

Parameter	Value	Default
UpwindMethod	{lps | muscl | upw}	LPS

Summary Upwind method for convective terms

Description All methods are hybrid in the sense that a centered scheme is blended based on the local Peclet number.

Values for individual equation sets may be set using optional token. Using both (in either order):

- UPWIND METHOD = UpwindMethod
- UPWIND METHOD = UpwindMethod FOR EQUATION Equations

Will result in the particular equation set to specified value while all others set to general value.

13.5.49. Use Equation Solver

Scope: Solution Options

`Use Equation Solver` *`SolverName`* `For Equation` *`Equations`*

Parameter	Value	Default
SolverName	`string`	-
Equations	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

---

Summary Link an equation solver to an equation set.

Description For example, if a solver block "scalar" was created using the Aztec package, e.g., BEGIN AZTEC EQUATION SOLVER scalar and the equation set was the u-component of momentum then the line command would be as follows: USE EQUATION SOLVER scalar FOR EQUATION X-Momentum

13.5.50. Use External Continuity Source

Scope: Solution Options

Summary Add external species source term from a transfer

Description Add source terms to the continuity equation from a nodal field transferred to this region, e.g. from a Fuego particle region. The field continuity_source will be added to the RHS of the continuity equation; this fields should have units of rate-of-change of mass per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

13.5.51. Use External Energy Source

Scope: Solution Options

Summary Add external energy source term from a transfer

Description Add source terms to the temperature or enthalpy equations from a nodal field transferred to this region, e.g. from a Fuego particle region. The field energy_source will be added to the RHS of the energy equation; this fields should have units of rate-of-change of energy per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

13.5.52. Use External Mixture_Fraction Source

Scope: Solution Options

Summary Add external species source term from a transfer

Description Add source terms to the mixture fraction equation from a nodal field transferred to this region, e.g. from a Fuego particle region. The field mixture_fraction_source will be added to the RHS of the mixture fraction equation; this fields should have units of rate-of-change of mass per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

13.5.53. Use External Momentum Source

Scope: Solution Options

Summary Add external momentum source terms from a transfer

Description Add source terms to the momentum equations from a nodal field transferred to this region, e.g. from a Fuego particle region. The fields x_momentum_source, y_momentum_source, and z_momentum source will be added to the RHS of the momentum equations; these fields should have units of rate-of-change of momentum per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

13.5.54. Use External Soot_Mass_Fraction Source

Scope: Solution Options

Summary Add external soot source term from a transfer

Description Add source terms to the soot mass fraction equation from a nodal field transferred to this region, e.g. from a Fuego particle region. The field soot_mass_fraction_source will be added to the RHS of the soot mass fraction equation; this fields should have units of rate-of-change of mass per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

13.5.55. Use External Species Source

Scope: Solution Options

Summary Add external species source term from a transfer

Description Add source terms to the species equations from a nodal field transferred to this region, e.g. from a Fuego particle region. The vector field species_source will be added to the RHS of the species equation; this fields should have units of rate-of-change of mass of species i per volume, so that multiplication by the control volume gives the correct source term. The transfer operation should send to the variables at state "none".

13.5.56. Use Lumped Velocity Density Interpolation

Scope: Solution Options

Summary Interpolate the density-velocity product

Description By default the continuity equation interpolates velocity and density separately to sub-control surfaces. This option interpolates the product of density times velocity instead.

13.5.57. Use Radiation Source From External Region

Scope: Solution Options

Summary Add in a source term from participating-media radiation which comes from another region through a transfer.

The USING CLASSIC LINEARIZATION optional argument is no longer used or needed, and will be removed in a future release.

13.5.58. Use Skew Symmetric Central Operator

Scope: Solution Options

Summary The blended central operator will be skew symmetric, default is **false**.

Description The convection operator is always blended with pure central (see hybrid factor description). For the CVFEM methodology, there is a balance between stability and accuracy. Dotting the momentum equation with velocity and summing yields the kinetic energy equation. If the convection operator is skew symmetric, than this dot product leaves something that is perfectly zero. This means that there can be no generation of kinetic energy and simulations can remain stable.

The full CVFEM stencil (27-pt on a hex mesh) is not skew symmetric. Therefore, in cases where one uses pure central (by specifying a hybrid factor of unity) there can be issues - especially on coarse meshes.

14. INPUT/OUTPUT REGION

The following commands are found in the Input/Output Region scope of the input file.

14.1. INPUT_OUTPUT REGION

Scope: Fuego Procedure

```
Begin Input_Output Region Parameter_block_name

    Create Element Field Field_name Of Type Option And Dimension
    Dimension [ Value {=|are|is} Number... ]

    Create Nodal Field Field_name Of Type Option And Dimension
    Dimension [ Value {=|are|is} Number... ]

    Fixed Time [ {=|are|is} Fixed_time ]

    Offset Time {=|are|is} Period_offset_time

    Periodicity Time {=|are|is} Periodicity_time

    Start Time {=|are|is} Start_time

    Use Finite Element Model ModelName [ Model Coordinates Are
    Nodal_variable_name ]

    Begin Heartbeat Label

    End

    Begin History Output Label

    End

    Begin Restart Data Label

    End

    Begin Results Output Label

    End

End
```

Summary BEGIN INPUT TRANSFER model_name USE FINITE ELEMENT MODEL fred START TIME is 0 OFFSET TIME is 1 PERIODICITY TIME is 10 END INPUT TRANSFER model_name

14.1.1. Create Element Field

Scope: Input_Output Region

`Create Element Field` *`Field_name`* `Of Type` *`Option`* `And Dimension` *`Dimension`* `[` *`Value`* `{=|are|is}` *`Number...`* `]`

Parameter	Value	Default
Field_name	`string`	-
Option	`{asym_tensor_03|complex|full_tensor_22|full_tensor_36|integer|long_integer|matrix_22|matrix_33|real|sym_tensor_21|sym_tensor_31|sym_tensor_33|unsigned_integer|unsigned_integer_64|vector_2d|vector_3d}`	-
Dimension	`integer`	-

Summary Creates a Element Field name field_name on the region.

14.1.2. Create Nodal Field

Scope: Input_Output Region

`Create Nodal Field` *`Field_name`* `Of Type` *`Option`* `And Dimension` *`Dimension`* `[` *`Value`* `{=|are|is}` *`Number...`* `]`

Parameter	Value	Default
Field_name	`string`	-
Option	`{asym_tensor_03|complex|full_tensor_22|full_tensor_36|integer|long_integer|matrix_22|matrix_33|real|sym_tensor_21|sym_tensor_31|sym_tensor_33|unsigned_integer|unsigned_integer_64|vector_2d|vector_3d}`	-
Dimension	`integer`	-

Summary Creates a Nodal Field name field_name on the region.

14.1.3. Fixed Time

Scope: Input_Output Region

Summary The line specifies that the database will be read for a single, fixed time. Specifying the actual time is optional. If the time is not specified, the final time plane in the database will be read.

NOTE: This option take precedence over the periodic specifications given by START TIME, PERIODICITY TIME, and OFFSET TIME.

if FIXED TIME is specified then if FIXED TIME value is given then (eg., FIXED TIME is 1.) DATABASE TIME = FIXED TIME else (eg., FIXED TIME) DATABASE TIME = last time in database else if PERIODICITY TIME greater than 0 then if APPLICATION TIME less than or equal to START TIME then DATABASE TIME = APPLICATION TIME else DATABASE TIME = START TIME + (APPLICATION TIME - START TIME) modulo PERIODICITY TIME else DATABASE TIME = APPLICATION TIME now add OFFSET TIME to the computed DATABASE TIME

14.1.4. Offset Time

Scope: Input_Output Region

`Offset Time {=|are|is}` *`Period_offset_time`*

Parameter	Value	Default
Period_offset_time	`real`	-

Summary This value is added to the application time to determine what database time slice to input. If OFFSET TIME were 15 than at application time 0 database time slice 15 would be read from the file and used for the initial values. At application time 1, database time slice 16 would be read. NOTE: The OFFSET TIME is added in after the START TIME and PERIODICITY TIME are used. The FIXED TIME option take precedence over this option.

if FIXED TIME is specified then if FIXED TIME value is given then (eg., FIXED TIME is 1.) DATABASE TIME = FIXED TIME else (eg., FIXED TIME) DATABASE TIME = last time in database else if PERIODICITY TIME greater than 0 then if APPLICATION TIME less than or equal to START TIME then DATABASE TIME = APPLICATION TIME else DATABASE TIME = START TIME + (APPLICATION TIME - START TIME) modulo PERIODICITY TIME else DATABASE TIME = APPLICATION TIME now add OFFSET TIME to the computed DATABASE TIME

14.1.5. Periodicity Time

Scope: Input_Output Region

`Periodicity Time {=|are|is}` *`Periodicity_time`*

Parameter	Value	Default
Periodicity_time	`real`	-

Summary

START TIME and PERIODICITY TIME taken together give the time frame from the input database to use to initialize the application values. If START TIME is 25 and PERIODICITY TIME is 10, then time slices from 25 to 35 will be used over and over again as the application time runs from 0 to whatever. In general DATABASE TIME is (APPLICATION TIME - START TIME) modulo PERIODICITY TIME after the application time reaches the START TIME.

NOTE: The OFFSET TIME is added in after the START TIME and PERIODICITY TIME are used. The FIXED TIME option take precedence over this option.

if FIXED TIME is specified then if FIXED TIME value is given then (eg., FIXED TIME is 1.) DATABASE TIME = FIXED TIME else (eg., FIXED TIME) DATABASE TIME = last time in database else if PERIODICITY TIME greater than 0 then if APPLICATION TIME less than or equal to START TIME then DATABASE TIME = APPLICATION TIME else DATABASE TIME = START TIME + (APPLICATION TIME - START TIME) modulo PERIODICITY TIME else DATABASE TIME = APPLICATION TIME now add OFFSET TIME to the computed DATABASE TIME

14.1.6. Start Time

Scope: Input_Output Region

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary

The time in which to start applying PERIODICITY TIME. If PERIODICITY TIME is not specified then START TIME is ignored.

NOTES: The OFFSET TIME is added in after the START TIME and PERIODICITY TIME are used. The FIXED TIME option take precedence over this option.

if FIXED TIME is specified then if FIXED TIME value is given then (eg., FIXED TIME is 1.) DATABASE TIME = FIXED TIME else (eg., FIXED TIME) DATABASE TIME = last time in database else if PERIODICITY TIME greater than 0 then if APPLICATION TIME less than or equal to START TIME then DATABASE TIME = APPLICATION TIME else DATABASE TIME = START TIME + (APPLICATION TIME - START TIME) modulo PERIODICITY TIME else DATABASE TIME = APPLICATION TIME now add OFFSET TIME to the computed DATABASE TIME

14.1.7. Use Finite Element Model

Scope: Input_Output Region

`Use Finite Element Model` *`ModelName`* `[ Model Coordinates Are` *`Nodal_variable_name`* `]`

Parameter	Value	Default
ModelName	`string`	-

Summary Associates a predefined finite element model with this region.

14.2. RESULTS OUTPUT

Scope: Average Region

`Begin Results Output` *`Label`*

`Additional Steps {=|are|is}` *`List_of_steps...`*

`Additional Times {=|are|is}` *`List_of_times...`*

`At Step` *`n Option`* `{=|are|is}` *`m`*

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

`Component Separator Character {=|are|is}` *`Separator`*

`Database Name {=|are|is}` *`StreamName`*

`Database Type {=|are|is}` *`DatabaseType`*

`Edge [` *`VariableList...`* `]`

`Edge Variables {=|are|is} [` *`VariableList...`* `]`

`Element [` *`VariableList...`* `]`

`Element Variables {=|are|is} [` *`VariableList...`* `]`

`Enable Large Ids`

`Exclude {=|are|is} [` *`ElementBlockList...`* `]`

`Exists` *`Option1 Option2`*

`Face [` *`VariableList...`* `]`

`Face Variables {=|are|is} [` *`VariableList...`* `]`

`Global [` *`Variables...`* `]`

`Global Variables {=|are|is} [` *`Variables...`* `]`

`Include {=|are|is} [` *`ElementBlockList...`* `]`

```
    Nodal [ VariableList...  ]
    Nodal Variables {=|are|is} [ VariableList...  ]
    Node [ VariableList...  ]
    Node Variables {=|are|is} [ VariableList...  ]
    Nodeset [ VariableList...  ]
    Nodeset Variables {=|are|is} [ VariableList...  ]
    Output Mesh {=|are|is} OutputMesh
    Output On Signal {=|are|is} Signals
    Overwrite Option1 Option2
    Property PropertyName {=|are|is} PropertyValue
    Sideset [ VariableList...  ]
    Sideset Variables {=|are|is} [ VariableList...  ]
    Start Time {=|are|is} Start_time
    Surface [ VariableList...  ]
    Surface Variables {=|are|is} [ VariableList...  ]
    Synchronize Output
    Termination Time {=|are|is} Final_time
    Timeseries Name {=|are|is} filename
    Timestep Adjustment Interval {=|are|is} Nsteps
    Title
    Use Output Scheduler Timer_name
    Begin Catalyst Label
    End
End
```

Summary Describes the location and type of the output stream used for outputting results for the enclosing region.

14.2.1. Additional Steps

Scope: Results Output

`Additional Steps {=|are|is}` *`List_of_steps...`*

Parameter	Value	Default
List_of_steps	`integer...`	-

Summary Additional simulation steps when output should occur.

14.2.2. Additional Times

Scope: Results Output

`Additional Times {=|are|is}` *`List_of_times...`*

Parameter	Value	Default
List_of_times	`real...`	-

Summary Additional simulation times when output should occur.

14.2.3. At Step

Scope: Results Output

`At Step` *`n Option`* `{=|are|is}` *`m`*

Parameter	Value	Default
n	`integer`	-
Option	`{increment|interval}`	-
m	`integer`	-

Summary Specify an output interval in terms of the internal iteration step count. The first step specifies the step count at the beginning of this interval and the second step specifies the output frequency to be used within this interval.

14.2.4. At Time

Scope: Results Output

`At Time` *`Dt1 Option`* `{=|are|is}` *`Dt2`*

Parameter	Value	Default
Dt1	`real`	-
Option	`{increment|interval}`	-
Dt2	`real`	-

Summary Specify an output interval in terms of the internal simulation time. The first time specifies the time at the beginning of this time interval and the second time specifies the output frequency to be used within this interval.

14.2.5. Component Separator Character

Scope: Results Output

`Component Separator Character {=|are|is}` *`Separator`*

Parameter	Value	Default
Separator	`string`	-

Summary The separator is the single character used to separate the output variable basename (e.g. "stress") from the suffices (e.g. "xx", "yy") when displaying the names of the individual variable components. For example, the default separator is "_", which results in names similar to "stress_xx", "stress_yy", ... "stress_zx". To eliminate the separator, specify an empty string ("") or NONE.

14.2.6. Database Name

Scope: Results Output

`Database Name {=|are|is}` *`StreamName`*

Parameter	Value	Default
StreamName	`string`	-

Summary The base name of the database containing the output results. If the filename begins with the '/' character, it is an absolute path; otherwise, the path to the current directory will be prepended to the name. If this line is omitted, then a filename will be created from the basename of the input file with a ".e" suffix appended.

14.2.7. Database Type

Scope: Results Output

`Database Type {=|are|is}` *`DatabaseType`*

Parameter	Value	Default
DatabaseType	`{catalyst|dof|dof_exodus|exodus|exodusii|generated|genesis|parallel_exodus}`	-

Summary The database type/format to be used for the output results.

14.2.8. Edge

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Edge variables are not supported for all database types.

14.2.9. Edge Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Edge variables are not supported for all database types.

14.2.10. Element

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities"

14.2.11. Element Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same

line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities"

14.2.12. Enable Large Ids

Scope: Results Output

Summary Enable 64 bit entity IDs for output

14.2.13. Exclude

Scope: Results Output

Summary Specify that the results file will only contain a subset of the element blocks in the analysis model. The element_block_list lists only the blocks which will not be output to the results database.

14.2.14. Exists

Scope: Results Output

`Exists` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{= | is}`	-
Option2	`{abort | add_suffix | append | overwrite}`	-

Summary Specify the behavior when creating this database and there is an existing file with the same name. The default behavior is "OVERWRITE" which deletes the existing file and creates a new file of the same name. "APPEND" will (if possible) append the new data to the end of the existing file. "ABORT" will print an error message and end the analysis. "ADD_SUFFIX" will add a -s???? suffix where the ???? is replaced by a sequential number starting at 0002.

14.2.15. Face

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

14.2.16. Face Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

14.2.17. Global

Scope: Results Output

Summary Define the global variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

14.2.18. Global Variables

Scope: Results Output

Summary Define the global variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

14.2.19. Include

Scope: Results Output

Summary Specify that the results file will only contain a subset of the element blocks in the analysis model. The element_block_list lists only the blocks which will be output to the results database.

14.2.20. Nodal

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

14.2.21. Nodal Variables

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

14.2.22. Node

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

14.2.23. Node Variables

Scope: Results Output

Summary Define the nodal variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line.

14.2.24. Nodeset

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Nodeset variables are not supported for all database types.

14.2.25. Nodeset Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Nodeset variables are not supported for all database types.

14.2.26. Output Mesh

Scope: Results Output

`Output Mesh {=|are|is}` *`OutputMesh`*

Parameter	Value	Default
OutputMesh	`{exposed surface|refined|unrefined}`	-

Summary Use this command to turn on "unrefined" as the output mesh. The default behavior is "refined", in which field variables are output on the current mesh, which may have been refined (either uniformly or adaptively) or had its topology altered in some way (e.g., dynamic load balancing) with respect to the original mesh read from the input file. By specifying "Output Mesh = unrefined", all output variables are output only on the original mesh objects read from the input file.

14.2.27. Output On Signal

Scope: Results Output

`Output On Signal {=|are|is}` *`Signals`*

Parameter	Value	Default
Signals	`{sigabrt|sigalrm|sigfpe|sighup|sigill|sigint|sigkill|sigpipe|sigquit|sigsegv|sigterm|sigusr1|sigusr2}`	-

Summary When the specified signal is raised, the output stream associated with this block will be output.

14.2.28. Overwrite

Scope: Results Output

`Overwrite` *`Option1 Option2`*

Parameter	Value	Default
Option1	`{=|is}`	-
Option2	`{false|no|off|on|true|yes}`	-

Summary (DEPRECATED, Use EXISTS) Specify whether the database should be overwritten if it exists. The default behavior is to overwrite unless this command is specified in the output block and either off, false, or no is specified.

14.2.29. Property

Scope: Results Output

`Property` *`PropertyName`* `{=|are|is}` *`PropertyValue`*

Parameter	Value	Default
PropertyName	`string`	-
PropertyValue	`string`	-

Summary Define a database property named "PropertyName" with the value "PropertyValue". If PropertyValue consists of all digits, it will define an integer property. If PropertyValue is "true" or "yes" or "false" or "no", it will define a logical property; otherwise it will define a string property. Suppported properties are typically database dependent; Current properties are: COMPRESSION_LEVEL = [0..9] (off) COMPRESSION_SHUFFLE = true|false|on|off (off) FILE_TYPE = netcdf4 (forces use of netcdf-4 hdf5-based file) (netcdf3) INTEGER_SIZE_DB = 4|8 (4) INTEGER_SIZE_API = 4|8 (4) REAL_SIZE_DB = 4|8 (8 is default) LOGGING = true|false|on|off (off) MAX_NAME_LENGTH = value (32)

14.2.30. Sideset

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

14.2.31. Sideset Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

14.2.32. Start Time

Scope: Results Output

`Start Time {=|are|is}` *`Start_time`*

Parameter	Value	Default
Start_time	`real`	-

Summary Specify the time to start outputting results from this output request block. This time overrides all 'at time' and 'at step' specifications.

14.2.33. Surface

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

14.2.34. Surface Variables

Scope: Results Output

Summary Define the variables that should be written to the results database. If "variable" is entered, then its name will be used on the output database. If "variable as db_name" is entered, then "db_name" will be the name used on the database for the internal variable "variable". Multiple "variable" or "variable as db_name" entries are allowed on the same line. The entities that this variable are written to can also be limited or specified with "exclude list_of_entities" or "include list_of_entities". Face variables are not supported for all database types.

14.2.35. Synchronize Output

Scope: Results Output

Summary In an analysis with multiple regions, it is sometimes desirable to synchronize the output of results data between the regions. This can be done by adding the SYNCHRONIZE OUTPUT command line to the results output block. If a results block has this set, then it will write output whenever a previous region writes output. The ordering of regions is based on the order in the input file, algorithmic considerations, or by solution control specifications.

Although the USE OUTPUT SCHEDULER command line can also synchronize output between regions, the SYNCHRONIZE OUTPUT command line will synchronize the output with regions where the output frequency is not under the direct control of the Sierra IO system. Examples of this are typically coupled applications where one or more of the codes are not Sierra-based applications such as Alegra and CTH. A results block with SYNCHRONIZE OUTPUT specified will also synchronize its output with the output of the external code.

The SYNCHRONIZE OUTPUT command can be used with other output scheduling commands such as time-based or step-based output specifications.

14.2.36. Termination Time

Scope: Results Output

`Termination Time {=|are|is}` *`Final_time`*

Parameter	Value	Default
Final_time	`real`	-

Summary Specify the time to stop outputting results from this output request block.

14.2.37. Timeseries Name

Scope: Results Output

`Timeseries Name {=|are|is}` *`filename`*

Parameter	Value	Default
filename	`string`	-

Summary Optionally specify a filename for a timeseries file that outputs the root database filename in the order that they are written. This is useful when running on large numbers of processors with many mesh-mods that cause simple disk operations to hang.

14.2.38. Timestep Adjustment Interval

Scope: Results Output

`Timestep Adjustment Interval {=|are|is}` *`Nsteps`*

Parameter	Value	Default
Nsteps	`integer`	-

Summary Specify the number of steps to 'look ahead' and adjust the timestep to ensure that the specified output times or simulation end time will be hit 'exactly'.

14.2.39. Title

Scope: Results Output

Summary Specify the title to be used for this specific output block.

14.2.40. Use Output Scheduler

Scope: Results Output

`Use Output Scheduler` *`Timer_name`*

Parameter	Value	Default
Timer_name	`string`	-

Summary Associates a predefined output scheduler with this output block (results, restart, heartbeat, or history).

15. ODE SOLVER SPECIFICATIONS

The following commands are used during ODE solver specification in the input file.

15.1. ODE SOLVER PARAMETERS

There are several ODE solvers available that can be selected. The basic set of input arguments for an ODE solver are given below, where the specific solver arguments are described in the following subsections.

```
ODE Solver = [SOLVER ARGS]
Absolute Tolerance = 1e-10
Relative Tolerance = 1e-5
Maximum Substeps = 100000
Minimum Step Size = 1e-15
```

15.1.1. CVODE

The CVODE solver is the most flexible of the options available, and often the fastest. It is the only option that allows you to select the method (BDF or ADAMS), the order (max of 5 for BDF and 12 for ADAMS), and the iteration scheme (FUNCTIONAL or NEWTON). For stiff problems, "BDF 5 NEWTON" will be most robust, but is also the most costly. For non-stiff problems, a lower order Adams method, such as "ADAMS 6 FUNCTIONAL" may be better.

The default is "BDF 5 NEWTON" if you only write "ODE Solver = CVODE". A few valid options are shown below.

```
ODE Solver = CVODE [METHOD] [ORDER] [ITERATION\_SCHEME]
ODE Solver = CVODE ADAMS 6 FUNCTIONAL
ODE Solver = CVODE BDF 3 NEWTON
ODE Solver = CVODE
```

15.1.2. LSODE

The LSODE solver can sometimes provide better performance than CVODE for large, stiff problems. It always uses the BDF 5 NEWTON scheme, so there are no extra options. The minimum time in LSODE is set to machine precision and cannot be set by the user.

```
ODE Solver = LSODE
```

15.1.3. RKEH

The RKEH is the Euler-Heun Method. It is a second-order error-adaptive Runge-Kutta scheme. It is typically the slow for stiff systems, but may be advantageous for slowly changing non-stiff systems.

```
ODE Solver = RKEH
```

15.1.4. RK12

The RK12 is the Fehlberg 1-2 Method. It is a second-order error-adaptive Runge-Kutta scheme. It is typically the slow for stiff systems, but may be advantageous for slowly changing non-stiff systems.

```
ODE Solver = RK12
```

15.1.5. RKBS

The RKBS is the Bogacki-Shampine Method. It is a third-order error-adaptive Runge-Kutta scheme. It is typically the slow for stiff systems, but may be advantageous for slowly changing non-stiff systems,

```
ODE Solver = RKBS
```

15.1.6. RKF45

The RKF45 is the Runge-Kutta-Fehlberg 4-5 Method. It is a fourth-order error-adaptive Runge-Kutta scheme. It is typically the slow for stiff systems, but may be advantageous for slowly changing non-stiff systems,

```
ODE Solver = RKF5
```

15.1.7. CashKarp

The CashKarp is the Cash-Karp Method. It is a fourth-order error-adaptive Runge-Kutta scheme. It is typically the slow for stiff systems, but may be advantageous for slowly changing non-stiff systems.

```
ODE Solver = CashKarp
```

15.1.8. DOPRI5

The DOPRI5 is the Dormand-Prince Method. It is a fourth-order error-adaptive Runge-Kutta scheme. It is typically the slow for stiff systems, but may be advantageous for slowly changing non-stiff systems.

```
ODE Solver = DOPRI5
```

16. CHEMISTRY DESCRIPTIONS

Chemistry description blocks are used in both particle and composite BC reactions. This chapter describes the syntax used for entering the reactions.

16.1. REACTION BLOCK OVERVIEW

The chemistry module in TFTK provides a common input syntax for specifying chemical reaction mechanisms in both Fuego and Aria. The individual reactions will be entered in reaction blocks that are nested within an overall chemistry block defined by the application.

An example mechanism is shown below, with more details about the available options in the following sections.

```
Begin Reaction R1
  Reaction is A -> B
  Rate Function = Arrhenius  A = 0.1  Ea = 350  R = 1
  Concentration Function = Standard  mu = A 1.0
  Heat of Reaction = -1e6
End

Begin Reaction R2
  Reaction is B -> C
  Rate Function = Arrhenius  A = 0.1  Ea = 350  R = 1
  Concentration Function = Standard  mu = Automatic
  Heat of Reaction = 1e6
End
```

16.2. REACTION UNITS

The method for specifying units can vary between Fuego and Aria. In Aria, the user specifies the "Species Variable Name". This defines what values the rate functions use for the Y values. There is no implicit conversion, so if the species variable name is DENSITY, the Y values will be in units of mass per volume. Likewise, if the species variable name is SPECIES, the units of Y will be in moles per volume. While it is technically possible to specify a species variable of mass fraction or species fraction (mole fraction), this should be done with caution since unit management will be completely up to the user.

Given the user-specified units of Y, a reaction rate (ω_j) is calculated for each reaction. The time rate of change of a reactant or product in a reaction is simply the product of ω_j with the stoichiometric coefficient for that species. This means if the rate function is kept constant but all stoichiometric coefficients are scaled by a constant factor, the production and consumption rates will also be scaled.

Similarly, the heat source is the product of the calculated ω_j value and the reaction heat release. The resulting units should be in power per volume (e.g. W/m^3). As before, the user should ensure that the units of ω_j and the reaction heat release are consistently defined to produce the appropriate output unit.

16.3. REACTION INPUTS

The reactions described above use a common reaction form with an Arrhenius rate and a standard concentration dependence. The chemistry module uses a generic reaction form, where the reaction rate is expressed as as a product of four contributing functions

$$\omega_j = f_m(t, T, P, Y) f_r(t, T, P, Y) f_p(P, T) f_c(Y), \tag{16.1}$$

where f_m is the multiplier function, f_r is the rate function, f_p is the pressure function, and f_c is the concentration function. The available options for each of these functions are outlined in the following section. They can be selected in the input file by adding as many reaction blocks in the parent chemistry description block, as

```
Begin Reaction [REACTION NAME]
  Reaction is A + 1.2B -> 3C
  Multiplier Function = [Optional, See below]
  Rate Function = [Required, See below]
  Pressure Function = [Optional, See below]
  Concentration Function = [Optional, See below]
  Temperature Phase = [GAS_PHASE, SOLID_PHASE, NO_MATERIAL_PHASE (default)]
  Pressure Phase = [GAS_PHASE, SOLID_PHASE, NO_MATERIAL_PHASE (default)]
  Heat of Reaction = [Value]
  Origin Phase Heat Fraction = [Value]
End
```

where each reaction in a given general chemistry block must have a unique name. The reaction string is parsed to determine the appropriate stoichiometry for the reaction. The left hand side and right hand side must be separated by either "->" or "=>". There are multiple options for each of the functions, which are described in the following sections.

If the pressure phase argument is omitted but temperature phase is provided, the pressure phase defaults to the same phase as the temperature. By default, all of the reaction heating is applied in the temperature phase (e.g. solid phase). However, if you specify a value for the "Origin Phase Heat Fraction" between 0 and 1 you can divide that heat between phases. The fraction specified controlls how

much energy goes into the origin phase (a.k.a. temperature phase). The remaining energy is split evenly between the other phases that participate in the chemistry model.

If a reversible reaction is desired, the forward and reverse reaction must be specified in two separate reaction blocks. However, there is an option to say 'Reaction is Reverse of [REACTION NAME]' giving the name of the forward reaction block. This will copy over the reaction string with the opposite signs for the stoichiometric coefficients, the rate function, the temperature and pressure phases, and the equal and opposite heat of reaction. If the forward reaction block contains a concentration function that is not Unity, this statement will also create a Standard concentration function for the reverse reaction with automatic coefficients. These default rate and concentration functions can be overwritten by specifying a new function inside the reverse reaction block. An example of how this option might be used for an equilibrium model is shown below.

```
Begin Reaction [FORWARD REACTION NAME]
  Reaction is A -> B
  Rate Function = [Required, See below]
  Concentration Function = [Optional, See below]
  Heat of Reaction = [Value]
End
Begin Reaction [REVERSE REACTION NAME]
  Reaction is Reverse of [FORWARD REACTION NAME]
  Multiplier Function = [Thermodynamic_Equilibrium, Simplified_Equilibrium, See below] [
End
```

A summary of all the reactions parsed from the input is also printed to the log file so you can verify they are evaluating as you have intended. An example of this output is shown below.

```
+-------------------------------------------------------------------------------+
| Reaction Summary                                                              |
+-------------------------------------------------------------------------------+

 Name:                      R1
  Reaction String:          A->B
  Stoich Coeffs:            -1.00000 1.00000
  Temperature Phase:        No Material Phase
  Pressure Phase:           No Material Phase
  Heat of Reaction:         -1.00000e+06
  Multiplier Function:      Unity
  Rate Function:            Arrhenius
      Arguments:            A = 0.100000 Ea = 350.000 beta = 0.00000 R = 1.00000
  Pressure Function:        Unity
  Concentration Function:   Standard
      Arguments:            mu = 1.00000 0.00000
```

16.3.1. Multiplier Functions

16.3.1.1. Unity

The default multiplier function is unity,

$$f_m(t, T, P, Y) = 1 \tag{16.2}$$

and can be used by either omitting the Multiplier Function line or using

```
Multiplier Function = Unity
```

16.3.1.2. Tanh_Time

The tanh_time multiplier function uses a hyperbolic tangent function of time to increase the reaction rate at a specified time.

$$f_m(t, T, P, Y) = \frac{1}{2}\left(1 + \tanh\left(x_{span}(t - x_{ref})\right)\right)\left(f_{end} - f_{start}\right) + f_{start} \tag{16.3}$$

The two inputs for this are x_{ref} and x_{span}. If you do not specify an input for the span, it uses $x_{span} = 1/x_{ref}$, unless x_{ref} was set as 0, in which case an error is thrown. The default start and end points (f_{start} and f_{end}) of the tanh blending function are 0 and 1, respectively. These can be overridden by user-specified start and end values if desired. Example inputs are shown below.

```
Multiplier Function = Tanh_Time x_ref = 54
Multiplier Function = Tanh_Time x_ref = 2.5 span = 0.1
Multiplier Function = Tanh_Time x_ref = 2.5 span = 0.1 start = 1 end = 100
```

16.3.1.3. Tanh_Temperature

The tanh_temperature multiplier function uses a hyperbolic tangent function of temperature to increase the reaction rate at a specified temperature.

$$f_m(t, T, P, Y) = \frac{1}{2}\left(1 + \tanh\left(x_{span}(T - x_{ref})\right)\right)\left(f_{end} - f_{start}\right) + f_{start} \tag{16.4}$$

The arguments for this function as the same format as for the tanh_time function.

```
Multiplier Function = Tanh_Temperature x_ref = 450
Multiplier Function = Tanh_Temperature x_ref = 600 span = 10
Multiplier Function = Tanh_Temperature x_ref = 600 span = 10 start = 100 end = 5
```

16.3.1.4. Thermodynamic_Equilibrium

The thermodynamic_equilibrium multiplier function can be used to calculate the inverse equilibrium constant $\frac{1}{K_c}$ using thermodynamic properties. When multiplied by the forward rate function, this will provide a reverse rate function. The equation for K_c satisfies the Van't Hoff equation and uses Gibbs free energy.

$$f_m(t,T,P,Y) = \exp\left\{-\sum_{i=1}^{n_s}\frac{\nu_i}{RT}(TS_i^o - H_i^o)\right\}\left(\frac{RT}{P_{atm}}\right)^{\sum_{i=1}^{n_s}\nu_i} \tag{16.5}$$

In this equation, P_{atm} is the atmospheric pressure, R is the universal gas constant, n_s is the number of species, ν_i is the stoichiometric coefficient of the ith species, and S_i^o and H_i^o are the standard state molar entropy and molar enthalpy of the ith species respectively.

The two optional inputs for this are atmospheric pressure, P_{atm}, and the universal gas constant, R. The default values for these are 101325 Pa and 8.314 J/mol-K respectively. Heat capacity evaluators must be provided with the species properties in order to use this function (the enthalpy and entropy evaluators will be derived from the provided heat capacity evaluators). Example inputs are shown below.

```
Multiplier Function = Thermodynamic_Equilibrium
Multiplier Function = Thermodynamic_Equilibrium  Patm = 101325 R = 8.314
```

16.3.1.5. Simplified_Equilibrium

The simplified_equilibrium multiplier function can be used to calculate the inverse equilibrium constant $\frac{1}{K_c}$ using a known value of the equilibrium constant (in terms of partial pressures) at a specific temperature and the heat of reaction. When multiplied by the forward rate function, this will provide a reverse rate function. The equation for K_c is the integrated Van't Hoff equation.

$$f_m(t,T,P,Y) = \left(\frac{1}{K_{eq}}\right)\exp\left\{\frac{\Delta H^o}{R}\left(\frac{1}{T}-\frac{1}{T_{eq}}\right)\right\}\left(\frac{RT}{P_{atm}}\right)^{\sum_{i=1}^{n_s}\nu_i} \tag{16.6}$$

In this equation, P_{atm} is the atmospheric pressure, R is the universal gas constant, n_s is the number of species, ν_i is the stoichiometric coefficient of the ith species, K_{eq} is the known equilibrium constant (in terms of partial pressures) at temperature T_{eq}, and ΔH^o is the heat of reaction.

The required input for this is T_{eq}. The optional inputs include K_{eq}, P_{atm}, and R defaulted to 1.0, 101325 Pa, and 8.314 J/mol-K respectively. Either the heat of reaction or the enthalpies of formation must be provided with the reactions in the input file to use this equation accurately. Example inputs are shown below.

```
Multiplier Function = Simplified_Equilibrium  Teq=800
Multiplier Function = Simplified_Equilibrium  Teq=800 Keq=0.6
Multiplier Function = Simplified_Equilibrium  Teq=800 Keq=0.6 Patm=101325 R=8.314
```

16.3.1.6. General

The general multiplier function uses a function string supplied in the input file to evaluate the rate generally. Allowable variables in this string include t (time), T (temperature), P (pressure), and any valid species name from the current model. The value used for a species is in whatever unit system the general chemistry model is using. Powers in this string are denoted using the '^' character. These functions can include multi-argument functions like min and max.

An example input for this option is

```
Multiplier Function = General  Function = 0.5*(1+tanh((T-350)^2))*max(T,300)
```

16.3.2. Rate Functions

16.3.2.1. Unity

The rate function is the only required function, so it has no default like the others. However, you can still select Unity for it if desired, using

```
Rate Function = Unity
```

$$f_r(t, T, P, Y) = 1 \tag{16.7}$$

16.3.2.2. Arrhenius

The Arrhenius rate function is the most common, and uses

$$f_r(t, T, P, Y) = AT^{\beta} \exp\left(\frac{-E_a}{RT}\right) \tag{16.8}$$

The pre-exponential constant, A, is the only required input. Optional inputs include E_a, β, and R. If omitted, the defaults for these are 0 for E_a and β and 8.314 for R. Some example lines to select this rate function is

```
Rate Function = Arrhenius  A = 100 Ea = 20000
Rate Function = Arrhenius  A = 100 Ea = 350 R = 1
Rate Function = Arrhenius  A = 100 beta = 2 Ea = 350 R = 1
```

16.3.2.3. Distributed Arrhenius

The Distributed Arrhenius rate function uses a cumulative normal distribution to vary the activation energy with respect to a selected progress species.

$$f_r(t, T, P, Y) = AT^{\beta} \exp\left(\frac{-(E_a + \sigma\zeta)}{RT}\right) \tag{16.9}$$

$$\zeta = CDF^{-1}(1 - Y_j/Y_0) \tag{16.10}$$

where CDF^{-1} is the clipped inverse cumulative normal distribution function, Y_j is the concentration of the selected progress species, Y_0 is the specified reference concentration, and σ is the activation energy variation. In addition to these inputs, the same input rules as for the Arrhenius reaction apply (only A is required of those again). The name specified for the progress species must be a valid species name.

Some example inputs for this rate type are (note that these are shown on multiple lines to fit in the manual, but must be on the same line in your input file):

```
Rate Function = Distributed_Arrhenius  A = 100 Ea = 20000 sigma = 1000 Y0 = 1.2
        ProgressSpecies = N2
Rate Function = Distributed_Arrhenius  A = 100 beta = 2 Ea = 350 R = 1 sigma = 100
        Y0 = 1.2 ProgressSpecies = N2
```

16.3.2.4. General

The general rate function uses a function string supplied in the input file to evaluate the rate generally. Allowable variables in this string include t (time), T (temperature), P (pressure), and any valid species name from the current model. The value used for a species is in whatever unit system the general chemistry model is using. Powers in this string are denoted using the '^' character. These functions can include multi-argument functions like min and max.

An example input for this option is:

```
Rate Function = General Function = 2.1*exp(-350/T)*N2^3.2*(1-CO2)*max(CO2,0.1)
Rate Function = General Function = (T>300 ? 5.0 : 0.01)*CO2*exp(-350/T)
```

16.3.3. Pressure Functions

16.3.3.1. Unity

The default pressure function is unity,

$$f_p(P, T) = 1 \tag{16.11}$$

and can be used by either omitting the Pressure Function line or using

```
Pressure Function = Unity
```

16.3.3.2. Exponential

The exponential pressure function is evaluated as

$$f_p(P,T) = \left(\frac{P}{P_{ref}}\right)^n, \tag{16.12}$$

where P_{ref} must be positive. An example input for this option is

```
Pressure Function = Exponential  P_ref = 1 n = 1.5
```

16.3.4. Concentration Functions

16.3.4.1. Unity

The default concentration function is unity,

$$f_c(Y) = 1 \tag{16.13}$$

and can be used by either omitting the Concentration Function line or using

```
Concentration Function = Unity
```

16.3.4.2. Standard

The standard concentration function is evaluated as

$$f_c(Y) = \prod_{i=1}^{N_s} Y_i^{\mu_i} \tag{16.14}$$

The values of μ can be specified manually using a list of entries for μ. This is done by entering a list of name-value pairs for each entry with a non-zero μ value. Alternately, you can provide a list of μ values for all species.

Alternately, if your reaction rate only depends on the concentration of the reactants, and the value of μ is the same as the stoichiometric coefficient, then you can have the values set automatically from the reaction stoichiometry by using the keyword "Automatic".

For example, if your reaction is "2A + B -> C" the automatic option would do the same thing as specifying $\mu = A2B1$. If you have repeated species as both reactants and products, only the reaction portion of the coefficient is considered. For example, "A + B -> 2A + C" would result in automatic exponents of $\mu = A1B1$.

```
Concentration Function = Standard  mu = A 2 B 1
Concentration Function = Standard  mu = 2 1 0
Concentration Function = Standard  mu = Automatic
```

16.3.4.3. Prout Tompkins

The Prout Tompkins concentration function can only be used with a reaction that has a single reactant and a single product, and is evaluated as

$$f_c(Y) = \alpha^n \left(1 - \left(1 - 10^{-p}\right) \alpha\right)^m \tag{16.15}$$

where α is the concentration of the reactant. An example input for this model is

```
Concentration Function = Prout_Tompkins  n = 1.2 m = 2.5 p = 9
```

16.3.4.4. Evaporation

The evaporation concentration function is evaluated as

$$f_c(Y) = A\sqrt{\frac{2}{\pi MRT}}\frac{1}{2 - C_c}\left(C_e p_v(T) x_{liq} - C_c p x_{vap}\right) \tag{16.16}$$

$$p_v(T) = p_0 \exp\left(\frac{L}{R}\left(\frac{1}{T_b} - \frac{1}{T}\right)\right) \tag{16.17}$$

where C_c and C_e are the condensation and evaporation coefficients, L is the molar enthalpy of vaporization, T_b is the boiling temperature at p_0, p is the gas phase pressure, A is the surface area, x_{liq} is the mole fraction of the liquid that is evaporating, and x_{vap} is the gas phase mole fraction of the evaporating species. If L is provided in mass units, R must be converted appropriately so their units match. If you provide only C_c or only C_e, then the coefficients are assumed to be the same.

For reactions that do not include evaporation, species that are defined in the material but not involved in any reactions may be omitted from the chemistry definition block. When using evaporation reactions, this is not allowed since it will alter the value of the mole fractions in the equations above. You must also use a species variable name of 'species' when using evaporation since this requires molar concentrations to get mole fractions.

The only rate, pressure, and multiplier functions allowed with an Evaporation concentration function are the Unity functions.

```
Concentration Function = Evaporation  R=8.314  M=18e-3  Cc=0.01  Tb=373  L=4e4
```

16.3.4.5. General Evaporation

This is a more general case of the evaporation function specified above. With general evaporation, different forms of the saturation pressure may be specified through a fugacity functor. The general evaporation function is evaluated as

$$f_c(Y) = A\sqrt{\frac{2}{\pi MRT}}\frac{1}{2-C_c}\left(C_e f_l - C_c f_v\right) \tag{16.18}$$

where f_l and f_v are the fugacity of the liquid and vapor respectively. Choices for the liquid fugacity include:

1. HENRY_CONSTANT: returns $x_{liq}H$ for a specified Henry's law constant H.
2. ANTOINE: returns $x_{liq}10^{\left(A-\frac{B}{C+T}\right)}$ for specified Antoine coefficients A, B, and C.
3. VANT_HOFF_BOILING: returns $x_{liq}P_{atm}\exp\left(\frac{L}{R}\left(\frac{1}{T_b}-\frac{1}{T}\right)\right)$ for required parameters of molar enthalpy of vaporization L and boiling temperature T_b at P_{atm}; and optional parameters of atmospheric pressure P_{atm} and universal gas constant R with default values of 101325 Pa and 8.314 J/mol-K respectively.

Choices for the vapor fugacity include:

1. IDEAL_GAS: returns $x_{vap}P$ (no parameters to specify).

In the equation for $f_c(Y)$, inputs R, M, C_c, and C_e work the same as they would for the Evaporation case above. The rules defined above for using Evaporation also apply to General_Evaporation.

For an example reaction Liq $\rightarrow$ Gas, general evaporation may be specified as

```
Concentration Function = General_Evaporation  R=8.314  M=18e-3  Cc=0.01
Begin Fugacity For Liq
    Type = VANT_HOFF_BOILING
    Parameter Patm = 101325
    Parameter R = 8.314
    Parameter Tb = 373
    Parameter L = 4e4
End
Begin Fugacity For Gas
    Type = IDEAL_GAS
End
```

which would give the same result as the Evaporation example given above. Other options for the liquid fugacity might look like

```
Begin Fugacity For Liq
    Type = ANTOINE
    Parameter A = 8
    Parameter B = 1000
    Parameter C = 200
End


Begin Fugacity For Liq
    Type = HENRY_CONSTANT
    Parameter H = 1
End
```

16.3.4.6. Condensation

The condensation concentration function is evaluated as

$$f_c(Y) = \frac{3DM(f_v - f_l)}{r^2 RT} \tag{16.19}$$

where f_l and f_v are the fugacity of the liquid and vapor respectively. Choices for the fugacity are given in the General Evaporation description above.

In the equation for $f_c(Y)$, D is the diffusion coefficient, M is the molar mass of the condensing species, r is the droplet radius, R is the universal gas constant, and T is temperature. The required parameter for condensation is D, while the optional parameter of R has the default value 8.314 J/mol-K.

Similar rules apply for using Condensation as using General_Evaporation defined above. However, you must use a mass basis for Y when using condensation and provide the molecular weights of all species present in order to calculate the mole fractions in the fugacity functions.

For an example reaction Gas $\rightarrow$ Liq, condensation may be specified as

```
Concentration Function = Condensation  R=8.314  D=1e-4
Begin Fugacity For Gas
    Type = IDEAL_GAS
End
Begin Fugacity For Liq
    Type = VANT_HOFF_BOILING
    Parameter Patm = 101325
    Parameter R = 8.314
    Parameter Tb = 373
    Parameter L = 4e4
End
```

17. FUEGO INITIAL CONDITIONS

17.1. INITIAL CONDITION BLOCK

Scope: Fuego Region

Begin Initial Condition Block *BlockName*

PrimitiveVariable {=|are|is} *Value*

Copy Nodal File Variable *ExodusName* To Region Field *AfgoName* [At Time *InterpTime*]

External Field For *VariableName* {=|are|is} *ExtFieldName* [Of Size *Value*]

Function For *PrimitiveVariable* {=|are|is} *FuncName* In The *Direction* Direction

Function For Mass_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Mole_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Ignite

Inline Function For *PrimitiveVariable* {=|are|is} *FuncStr*

Integer Data For Subroutine *SubName* {=|are|is} *Values...*

Mass_Fraction *Species* {=|are|is} *y0*

Mole_Fraction *Species* {=|are|is} *y0*

Progress_Variable *ProgressVariableName* {=|are|is} *Value*

Real Data For Subroutine *SubName* {=|are|is} *Values...*

Solid Object {=|are|is} *t0*

Subroutine For *PrimitiveVariable* {=|are|is} *SubName*

Subroutine For Mass_Fraction {=|are|is} *Sub*

Subroutine For Mole_Fraction {=|are|is} *Sub*

Update Enthalpy And Density

```
    Vof Shape For PrimitiveVariable {=|are|is} ShapeName ShapeArgs...

    Volume {=|are|is} t0

End
```

Summary Allows the specification of piecewise constant initial conditions on any combination of volumes, surfaces and nodes.

17.1.1.

Scope: Initial Condition Block

PrimitiveVariable {=|are|is} *Value*

Parameter	Value	Default
PrimitiveVariable	{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}	-
Value	"string"	-

Summary Assign initial condition to the specified variable. Specified value can be a constant or a string function of time, space, and global variables.

17.1.2. Copy Nodal File Variable

Scope: Initial Condition Block

```
Copy Nodal File Variable ExodusName To Region Field AfgoName [ At Time
InterpTime ]
```

Parameter	Value	Default
ExodusName	`string`	-
AfgoName	`string`	-

Summary Solid geometry to which this initial condition applies.

17.1.3. External Field For

Scope: Initial Condition Block

`External Field For` *`VariableName`* `{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `]`

Parameter	Value	Default
VariableName	`string`	-
ExtFieldName	`string`	-

Summary Name of the field that is to be transferred in.

17.1.4. Function For

Scope: Initial Condition Block

`Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of function with which to set the initial condition for the given variable in the given direction.

17.1.5. Function For Mass_Fraction

Scope: Initial Condition Block

`Function For Mass_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`*
`Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Names of functions with which to set the initial mass fractions for the given species in the given direction.

17.1.6. Function For Mole_Fraction

Scope: Initial Condition Block

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`*
`Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mole fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

17.1.7. Ignite

Scope: Initial Condition Block

Summary Flag this block for combustion ignition

17.1.8. Inline Function For

Scope: Initial Condition Block

`Inline Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncStr`*

Parameter	Value	Default
PrimitiveVariable	{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}	-
FuncStr	"string"	-

Summary String function of x, y, z, and t to use for the given variable. The function string must be enclosed in quotes if it contains spaces or commas. For example: Inline function for x_velocity = "min(1, 0.1*t)"

17.1.9. Integer Data For Subroutine

Scope: Initial Condition Block

`Integer Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	string	-
Values	integer...	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

17.1.10. Mass_Fraction

Scope: Initial Condition Block

`Mass_Fraction` *`Species`* `{=|are|is}` *`y0`*

Parameter	Value	Default
Species	string	-
yo	"string"	-

Summary Set the initial mass fraction. Specified value can be a constant or a string function of time, space, and global variables.

17.1.11. Mole_Fraction

Scope: Initial Condition Block

`Mole_Fraction` *`Species`* `{=|are|is}` *`y0`*

Parameter	Value	Default
Species	`string`	-
yo	`"string"`	-

Summary Set the initial mole fractions to a constant value.

17.1.12. Progress_Variable

Scope: Initial Condition Block

`Progress_Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`"string"`	-

Summary Set the initial progress variable. Specified value can be a constant or a string function of time, space, and global variables.

17.1.13. Real Data For Subroutine

Scope: Initial Condition Block

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

17.1.14. Solid Object

Scope: Initial Condition Block

`Solid Object {=|are|is}` *`to`*

Parameter	Value	Default
to	`string`	-

Summary Solid geometry to which this initial condition applies.

17.1.15. Subroutine For

Scope: Initial Condition Block

`Subroutine For` *`PrimitiveVariable`* `{=|are|is}` *`SubName`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
SubName	`string`	-

Summary Set the initial condition on the specified variable using a subroutine

17.1.16. Subroutine For Mass_Fraction

Scope: Initial Condition Block

`Subroutine For Mass_Fraction {=|are|is}` *`Sub`*

Parameter	Value	Default
Sub	`string`	-

Summary Set the initial mass fractions using a subroutine.

17.1.17. Subroutine For Mole_Fraction

Scope: Initial Condition Block

`Subroutine For Mole_Fraction {=|are|is}` *`Sub`*

Parameter	Value	Default
Sub	`string`	-

Summary Set the initial mole fractions using a subroutine.

17.1.18. Update Enthalpy And Density

Scope: Initial Condition Block

Summary Flag this block for updating enthalpy after replacing T or Y

17.1.19. Vof Shape For

Scope: Initial Condition Block

`Vof Shape For` *`PrimitiveVariable`* `{=|are|is}` *`ShapeName ShapeArgs...`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
ShapeName	`string`	-
ShapeArgs	`string...`	-

Summary Set the smooth VOF initial conditions using basic shapes. Multiple shapes can be specified, and where they intersect will be a union operation.

17.1.20. Volume

Scope: Initial Condition Block

`Volume {=|are|is}` *to*

Parameter	Value	Default
to	`string`	-

Summary Element Block name to which this initial condition applies.

18. FUEGO BOUNDARY CONDITIONS

This section lists the allowed Fuego boundary conditions and the commands available for each.

18.1. FIXED BOUNDARY CONDITION ON SURFACE

Scope: Fuego Region

Begin Fixed Boundary Condition On Surface *Surfacename*

PrimitiveVariable {=|are|is} *Value*

Emissivity Spectral File Name {=|are|is} *Name*

External Field For *VariableName* [{of} *Species*]{=|are|is} *ExtFieldName* [Of Size *Value* With Multiplier {=|are|is} *Multiplier*]

Emissivity {=|are|is} *value*

Function For *PrimitiveVariable* {=|are|is} *FuncName* [In The *Direction* Direction]

Function For Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Mass_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Mole_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Radiation Boundary Temperature {=|are|is} *functionName*

Function For Radiation Environment Temperature {=|are|is} *functionName*

Function For Transmissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Transparent Band Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

```
    Function For Wall Temperature {=|are|is} functionName [ In The
    Direction Direction ]

    Inline Function For PrimitiveVariable {=|are|is} FuncStr

    Integer Data For Subroutine SubName {=|are|is} Values...

    Mass_Fraction Species {=|are|is} Mass fraction

    Mole_Fraction Species {=|are|is} Mole fraction

    Postprocess FluxType Flux Of equation [ As aliases...  ]

    Progress_Variable ProgressVariableName {=|are|is} Value

    Real Data For Subroutine SubName {=|are|is} Values...

    Radiation Boundary Temperature {=|are|is} value

    Radiation Boundary Temperature Field {=|are|is} FieldName

    Radiation Environment Temperature {=|are|is} value

    Radiation Environment Temperature Field {=|are|is} FieldName

    Subroutine For PrimitiveVariable {=|are|is} Subroutine

    Subroutine For Emissivity {=|are|is} subroutineName

    Subroutine For Mass_Fraction {=|are|is} Subroutine

    Subroutine For Mole_Fraction {=|are|is} Subroutine

    Subroutine For Radiation Boundary Temperature {=|are|is}
    subroutineName

    Subroutine For Radiation Environment Temperature {=|are|is}
    subroutineName

    Subroutine For Transmissivity {=|are|is} subroutineName

    Subroutine For Transparent Band Emissivity {=|are|is}
    subroutineName

    Subroutine For Wall Temperature {=|are|is} subroutineName

    Transparent Band Emissivity {=|are|is} value

    Transmissivity {=|are|is} value

    Wall Temperature {=|are|is} value

End
```

Summary Defines a fixed boundary condition on a named surface of the mesh.

18.1.1.

Scope: Fixed Boundary Condition On Surface

PrimitiveVariable {=|are|is} *Value*

Parameter	Value	Default
PrimitiveVariable	{contact_angle|edc_product |gas_volume_fraction| mixture_fraction|pressure| progress_variable|scalar_variance |second_mixture_fraction |solid_volume_fraction |soot_mass_fraction| soot_nuclei_mass_fraction| temperature|turbulent_dissipation |turbulent_frequency| turbulent_helmholtz_function| turbulent_kinetic_energy|turbulent_v2 |volume_of_fluid|x_solid_velocity |x_velocity|y_solid_velocity| y_velocity|z_solid_velocity| z_velocity}	-
Value	"string"	-

Summary Value for the specified variable (in consistent units). Value can be a constant or a string function of position (x,y,z), time (t), and any global variable.

18.1.2. Emissivity Spectral File Name

Scope: Fixed Boundary Condition On Surface

Emissivity Spectral File Name {=|are|is} *Name*

Parameter	Value	Default
Name	string	-

Summary Specify the file name for defining spectral radiation properties on a surface.

18.1.3. External Field For

Scope: Fixed Boundary Condition On Surface

External Field For *VariableName* [{of} *Species*]{=|are|is} *ExtFieldName* [Of Size *Value* With Multiplier {=|are|is} *Multiplier*]

Parameter	Value	Default
VariableName	`string`	-
Species	`string`	-
ExtFieldName	`string`	-

Summary Name of the field that is to be transferred in. For species equation, specify variable as "mass fraction of" followed by the species name for transfers associated with separated individual species field names.

18.1.4. Emissivity

Scope: Fixed Boundary Condition On Surface

```
Emissivity {=|are|is} value
```

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for emissivity variable.

18.1.5. Function For

Scope: Fixed Boundary Condition On Surface

```
Function For PrimitiveVariable {=|are|is} FuncName [ In The Direction
Direction ]
```

Parameter	Value	Default
PrimitiveVariable	{contact_angle|edc_product |gas_volume_fraction| mixture_fraction|pressure| progress_variable|scalar_variance |second_mixture_fraction |solid_volume_fraction |soot_mass_fraction| soot_nuclei_mass_fraction| temperature|turbulent_dissipation |turbulent_frequency| turbulent_helmholtz_function| turbulent_kinetic_energy|turbulent_v2 |volume_of_fluid|x_solid_velocity |x_velocity|y_solid_velocity| y_velocity|z_solid_velocity| z_velocity}	-
FuncName	`string`	-

Summary Name of function to use for the given variable, in the specified direction.

18.1.6. Function For Emissivity

Scope: Fixed Boundary Condition On Surface

`Function For Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of the function to use for the emissivity in the given direction (default direction = time).

18.1.7. Function For Mass_Fraction

Scope: Fixed Boundary Condition On Surface

`Function For Mass_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.1.8. Function For Mole_Fraction

Scope: Fixed Boundary Condition On Surface

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.1.9. Function For Radiation Boundary Temperature

Scope: Fixed Boundary Condition On Surface

`Function For Radiation Boundary Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_boundary_temperature variable

18.1.10. Function For Radiation Environment Temperature

Scope: Fixed Boundary Condition On Surface

`Function For Radiation Environment Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_environment_temperature variable

18.1.11. Function For Transmissivity

Scope: Fixed Boundary Condition On Surface

`Function For Transmissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transmissivity variable

18.1.12. Function For Transparent Band Emissivity

Scope: Fixed Boundary Condition On Surface

`Function For Transparent Band Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transparent emissivity band for spectral emissivity on surfaces.

18.1.13. Function For Wall Temperature

Scope: Fixed Boundary Condition On Surface

`Function For Wall Temperature {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the wall_temperature variable, in the specified direction.

18.1.14. Inline Function For

Scope: Fixed Boundary Condition On Surface

`Inline Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncStr`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
FuncStr	`"string"`	-

Summary String function of x, y, z, and t to use for the given variable. The function string must be enclosed in quotes if it has spaces or commas. For example: Inline function for x_velocity = "min(1, 0.1*t)"

18.1.15. Integer Data For Subroutine

Scope: Fixed Boundary Condition On Surface

`Integer Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`integer...`	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.1.16. Mass_Fraction

Scope: Fixed Boundary Condition On Surface

`Mass_Fraction` *`Species`* `{=|are|is}` *`Mass fraction`*

Parameter	Value	Default
Species	`string`	-
Mass fraction	`"string"`	-

Summary Value for the mass fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.1.17. Mole_Fraction

Scope: Fixed Boundary Condition On Surface

`Mole_Fraction` *`Species`* `{=|are|is}` *`Mole fraction`*

Parameter	Value	Default
Species	`string`	-
Mole fraction	`"string"`	-

Summary Value for the mole fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.1.18. Postprocess

Scope: Fixed Boundary Condition On Surface

`Postprocess` *`FluxType`* `Flux Of` *`equation`* `[ As` *`aliases...`* `]`

Parameter	Value	Default
FluxType	`string`	-
equation	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

Summary

Enable nodal flux post-processing for a given equation. Flux types can be "total", "advective", or "diffusive". Integrated fluxes will also be output to global variables for post-processing or use in other string functions.

By default the variables will be named based on the equation and sideset they are applied to as bc_FluxType_EquationName_flux_SurfaceName. However, if you want to assign a more compact or descriptive name you can provide it with

```
POSTPROCESS TOTAL FLUX OF Continuity AS mdot1
```

When post-processing enthalpy, the average temperature is also output as a global variable. If you provide only one alias the enthalpy flux uses your alias and the average temperature is automatically named. If you provide two aliases the first is used for enthalpy and the second is used for temperature.

```
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux Tavg
```

When post-processing species, the post-processor is run for each species that is solved for. This means that there is no post-processor run on the last species (which is determined by a fractional balance). If you provide aliases for the species post-processor you should provide one for each post-processed species. For example, in a problem with O_2, CO_2, and N_2 (in that order) you could use:

```
POSTPROCESS TOTAL FLUX OF SPECIES AS mdotO2 mdotCO2
```

18.1.19. Progress_Variable

Scope: Fixed Boundary Condition On Surface

`Progress_Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`real`	-

Summary Constant value for the progress variable of selected scalars.

18.1.20. Real Data For Subroutine

Scope: Fixed Boundary Condition On Surface

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.1.21. Radiation Boundary Temperature

Scope: Fixed Boundary Condition On Surface

`Radiation Boundary Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_boundary_temperature variable.

18.1.22. Radiation Boundary Temperature Field

Scope: Fixed Boundary Condition On Surface

`Radiation Boundary Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation boundary temperature.

18.1.23. Radiation Environment Temperature

Scope: Fixed Boundary Condition On Surface

`Radiation Environment Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_environment_temperature variable.

18.1.24. Radiation Environment Temperature Field

Scope: Fixed Boundary Condition On Surface

`Radiation Environment Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation environment temperature.

18.1.25. Subroutine For

Scope: Fixed Boundary Condition On Surface

`Subroutine For` *`PrimitiveVariable`* `{=|are|is}` *`Subroutine`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
Subroutine	`string`	-

Summary Name of the subroutine to use for this variable.

18.1.26. Subroutine For Emissivity

Scope: Fixed Boundary Condition On Surface

`Subroutine For Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the emissivity variable.

18.1.27. Subroutine For Mass_Fraction

Scope: Fixed Boundary Condition On Surface

`Subroutine For Mass_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mass fraction. ALL species mass fractions must be assigned by this subroutine.

18.1.28. Subroutine For Mole_Fraction

Scope: Fixed Boundary Condition On Surface

`Subroutine For Mole_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mole fractions. ALL species mole fractions must be assigned by this subroutine.

18.1.29. Subroutine For Radiation Boundary Temperature

Scope: Fixed Boundary Condition On Surface

`Subroutine For Radiation Boundary Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_boundary_temperature variable.

18.1.30. Subroutine For Radiation Environment Temperature

Scope: Fixed Boundary Condition On Surface

`Subroutine For Radiation Environment Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_environment_temperature variable.

18.1.31. Subroutine For Transmissivity

Scope: Fixed Boundary Condition On Surface

`Subroutine For Transmissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transmissivity variable.

18.1.32. Subroutine For Transparent Band Emissivity

Scope: Fixed Boundary Condition On Surface

`Subroutine For Transparent Band Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transparent emissivity band for spectral emissivity on surfaces.

18.1.33. Subroutine For Wall Temperature

Scope: Fixed Boundary Condition On Surface

`Subroutine For Wall Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the wall_temperature variable.

18.1.34. Transparent Band Emissivity

Scope: Fixed Boundary Condition On Surface

`Transparent Band Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value to use for the emissivity in the transparent band for spectral surfaces.

18.1.35. Transmissivity

Scope: Fixed Boundary Condition On Surface

`Transmissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for transmissivity variable.

18.1.36. Wall Temperature

Scope: Fixed Boundary Condition On Surface

`Wall Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Value for wall_temperature variable. Can be a constant or a string function of space (x,y,z), time (t) or global variables.

18.2. HEAT FLUX BOUNDARY CONDITION ON SURFACE

Scope: Fuego Region

`Begin Heat Flux Boundary Condition On Surface` *`Surfacename`*

`External Field For` *`VariableName`* `[{of}` *`Species`* `]{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `With Multiplier {=|are|is}` *`Multiplier`* `]`

`Function For Heat_Flux {=|are|is}` *`functionName`* `In The` *`Direction`* `Direction`

```
    Heat_Flux {=|are|is} value

    Integer Data For Subroutine SubName {=|are|is} Values...

    Postprocess FluxType Flux Of equation [ As aliases...  ]

    Real Data For Subroutine SubName {=|are|is} Values...

    Subroutine For Heat_Flux {=|are|is} subroutineName

    Law_Of_Wall_Roughness_Parameter {=|are|is} Law of the Wall
    Roughness parameter

End
```

Summary Defines a heat flux boundary condition for heat transfer on a named surface of the mesh.

Description From Section 2.6.10, "In practice, the heat flux boundary condition block is to be defined on an already defined wall boundary condition block (without temperature specification). In this manner, multiple boundary conditions are "painted" on a particular sideset."

If one desires to post-process heat flux on a surface, one should remember that post-processed heat flux (both in the log file and Exodus output) is the reconstruction from solution variables and any turbulent wall model that may be specified. As such, the post-processed heat flux is dependent on mesh quality and has been shown to converge to the correct value at a zeroth order rate. Thus, caution should be exercised when evaluating post-processed heat flux values.

18.2.1. External Field For

Scope: Heat Flux Boundary Condition On Surface

```
External Field For VariableName [{of} Species ]{=|are|is} ExtFieldName [
Of Size Value With Multiplier {=|are|is} Multiplier ]
```

Parameter	Value	Default
VariableName	`string`	-
Species	`string`	-
ExtFieldName	`string`	-

Summary Name of the field that is to be transferred in. For species equation, specify variable as "mass fraction of" followed by the species name for transfers associated with separated individual species field names.

18.2.2. Function For Heat_Flux

Scope: Heat Flux Boundary Condition On Surface

`Function For Heat_Flux {=|are|is}` *`functionName`* `In The` *`Direction`*
`Direction`

Parameter	Value	Default
functionName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the function to use for the heat flux in the given direction.

18.2.3. Heat_Flux

Scope: Heat Flux Boundary Condition On Surface

`Heat_Flux {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for the heat flux through the surface.

Description This value is positive for heat into the surface.

18.2.4. Integer Data For Subroutine

Scope: Heat Flux Boundary Condition On Surface

`Integer Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`integer...`	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.2.5. Postprocess

Scope: Heat Flux Boundary Condition On Surface

`Postprocess` *`FluxType`* `Flux Of` *`equation`* `[ As` *`aliases...`* `]`

Parameter	Value	Default
FluxType	`string`	-
equation	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

---

Summary

Enable nodal flux post-processing for a given equation. Flux types can be "total", "advective", or "diffusive". Integrated fluxes will also be output to global variables for post-processing or use in other string functions.

By default the variables will be named based on the equation and sideset they are applied to as bc_FluxType_EquationName_flux_SurfaceName. However, if you want to assign a more compact or descriptive name you can provide it with

```
POSTPROCESS TOTAL FLUX OF Continuity AS mdot1
```

When post-processing enthalpy, the average temperature is also output as a global variable. If you provide only one alias the enthalpy flux uses your alias and the average temperature is automatically named. If you provide two aliases the first is used for enthalpy and the second is used for temperature.

```
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux Tavg
```

When post-processing species, the post-processor is run for each species that is solved for. This means that there is no post-processor run on the last species (which is determined by a fractional balance). If you provide aliases for the species post-processor you should provide one for each post-processed species. For example, in a problem with O_2, CO_2, and N_2 (in that order) you could use:

```
POSTPROCESS TOTAL FLUX OF SPECIES AS mdotO2 mdotCO2
```

18.2.6. Real Data For Subroutine

Scope: Heat Flux Boundary Condition On Surface

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.2.7. Subroutine For Heat_Flux

Scope: Heat Flux Boundary Condition On Surface

`Subroutine For Heat_Flux {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the subroutine to use for the heat flux.

18.2.8. Law_Of_Wall_Roughness_Parameter

Scope: Heat Flux Boundary Condition On Surface

`Law_Of_Wall_Roughness_Parameter {=|are|is}` *`Law of the Wall Roughness parameter`*

Parameter	Value	Default
Law of the Wall Roughness parameter	`real`	`9.8`

Summary Specify dimensionless roughness factor to be used in the law-of-the-wall formulation.

18.3. INFLOW BOUNDARY CONDITION ON SURFACE

Scope: Fuego Region

`Begin Inflow Boundary Condition On Surface` *`Surfacename`*

`PrimitiveVariable` `{=|are|is}` *`Value`*

`Emissivity Spectral File Name {=|are|is}` *`Name`*

External Field For *VariableName* [{of} *Species*]{=|are|is} *ExtFieldName* [Of Size *Value* With Multiplier {=|are|is} *Multiplier*]

Emissivity {=|are|is} *value*

Function For *PrimitiveVariable* {=|are|is} *FuncName* [In The *Direction* Direction]

Function For Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Mass_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Mole_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Radiation Boundary Temperature {=|are|is} *functionName*

Function For Radiation Environment Temperature {=|are|is} *functionName*

Function For Transmissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Transparent Band Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Wall Temperature {=|are|is} *functionName* [In The *Direction* Direction]

Inline Function For *PrimitiveVariable* {=|are|is} *FuncStr*

Integer Data For Subroutine *SubName* {=|are|is} *Values...*

Mass_Fraction *Species* {=|are|is} *Mass fraction*

Mole_Fraction *Species* {=|are|is} *Mole fraction*

Omit Diffusion Terms

Post Process Delta Area

Post Process Delta Mass Flow Rate

Post Process Delta Pressure

Post Process Delta Thrust

Postprocess *FluxType* Flux Of *equation* [As *aliases...*]

Project Nodes

Progress_Variable *ProgressVariableName* {=|are|is} *Value*

```
    Real Data For Subroutine SubName {=|are|is} Values...

    Radiation Boundary Temperature {=|are|is} value

    Radiation Boundary Temperature Field {=|are|is} FieldName

    Radiation Environment Temperature {=|are|is} value

    Radiation Environment Temperature Field {=|are|is} FieldName

    Subroutine For PrimitiveVariable {=|are|is} Subroutine

    Subroutine For Emissivity {=|are|is} subroutineName

    Subroutine For Mass_Fraction {=|are|is} Subroutine

    Subroutine For Mole_Fraction {=|are|is} Subroutine

    Subroutine For Radiation Boundary Temperature {=|are|is}
    subroutineName

    Subroutine For Radiation Environment Temperature {=|are|is}
    subroutineName

    Subroutine For Transmissivity {=|are|is} subroutineName

    Subroutine For Transparent Band Emissivity {=|are|is}
    subroutineName

    Subroutine For Wall Temperature {=|are|is} subroutineName

    Transparent Band Emissivity {=|are|is} value

    Transmissivity {=|are|is} value

    Use Fluxes

    Wall Temperature {=|are|is} value

End
```

Summary Defines a inflow boundary condition on a named surface of the mesh.

18.3.1.

Scope: Inflow Boundary Condition On Surface

PrimitiveVariable {=|are|is} *Value*

Parameter	Value	Default
PrimitiveVariable	{contact_angle|edc_product |gas_volume_fraction| mixture_fraction|pressure| progress_variable|scalar_variance |second_mixture_fraction |solid_volume_fraction |soot_mass_fraction| soot_nuclei_mass_fraction| temperature|turbulent_dissipation |turbulent_frequency| turbulent_helmholtz_function| turbulent_kinetic_energy|turbulent_v2 |volume_of_fluid|x_solid_velocity |x_velocity|y_solid_velocity| y_velocity|z_solid_velocity| z_velocity}	-
Value	"string"	-

Summary Value for the specified variable (in consistent units). Value can be a constant or a string function of position (x,y,z), time (t), and any global variable.

18.3.2. Emissivity Spectral File Name

Scope: Inflow Boundary Condition On Surface

Emissivity Spectral File Name {=|are|is} *Name*

Parameter	Value	Default
Name	string	-

Summary Specify the file name for defining spectral radiation properties on a surface.

18.3.3. External Field For

Scope: Inflow Boundary Condition On Surface

External Field For *VariableName* [{of} *Species*]{=|are|is} *ExtFieldName* [Of Size *Value* With Multiplier {=|are|is} *Multiplier*]

Parameter	Value	Default
VariableName	`string`	-
Species	`string`	-
ExtFieldName	`string`	-

Summary Name of the field that is to be transferred in. For species equation, specify variable as "mass fraction of" followed by the species name for transfers associated with separated individual species field names.

18.3.4. Emissivity

Scope: Inflow Boundary Condition On Surface

```
Emissivity {=|are|is} value
```

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for emissivity variable.

18.3.5. Function For

Scope: Inflow Boundary Condition On Surface

```
Function For PrimitiveVariable {=|are|is} FuncName [ In The Direction
Direction ]
```

Parameter	Value	Default
PrimitiveVariable	{contact_angle|edc_product |gas_volume_fraction| mixture_fraction|pressure| progress_variable|scalar_variance |second_mixture_fraction |solid_volume_fraction |soot_mass_fraction| soot_nuclei_mass_fraction| temperature|turbulent_dissipation |turbulent_frequency| turbulent_helmholtz_function| turbulent_kinetic_energy|turbulent_v2 |volume_of_fluid|x_solid_velocity |x_velocity|y_solid_velocity| y_velocity|z_solid_velocity| z_velocity}	-
FuncName	`string`	-

Summary Name of function to use for the given variable, in the specified direction.

18.3.6. Function For Emissivity

Scope: Inflow Boundary Condition On Surface

`Function For Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of the function to use for the emissivity in the given direction (default direction = time).

18.3.7. Function For Mass_Fraction

Scope: Inflow Boundary Condition On Surface

`Function For Mass_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.3.8. Function For Mole_Fraction

Scope: Inflow Boundary Condition On Surface

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.3.9. Function For Radiation Boundary Temperature

Scope: Inflow Boundary Condition On Surface

`Function For Radiation Boundary Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_boundary_temperature variable

18.3.10. Function For Radiation Environment Temperature

Scope: Inflow Boundary Condition On Surface

`Function For Radiation Environment Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_environment_temperature variable

18.3.11. Function For Transmissivity

Scope: Inflow Boundary Condition On Surface

`Function For Transmissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transmissivity variable

18.3.12. Function For Transparent Band Emissivity

Scope: Inflow Boundary Condition On Surface

`Function For Transparent Band Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transparent emissivity band for spectral emissivity on surfaces.

18.3.13. Function For Wall Temperature

Scope: Inflow Boundary Condition On Surface

`Function For Wall Temperature {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the wall_temperature variable, in the specified direction.

18.3.14. Inline Function For

Scope: Inflow Boundary Condition On Surface

`Inline Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncStr`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
FuncStr	`"string"`	-

Summary String function of x, y, z, and t to use for the given variable. The function string must be enclosed in quotes if it has spaces or commas. For example: Inline function for x_velocity = "min(1, 0.1*t)"

18.3.15. Integer Data For Subroutine

Scope: Inflow Boundary Condition On Surface

`Integer Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`integer...`	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.3.16. Mass_Fraction

Scope: Inflow Boundary Condition On Surface

`Mass_Fraction` *`Species`* `{=|are|is}` *`Mass fraction`*

Parameter	Value	Default
Species	`string`	-
Mass fraction	`"string"`	-

Summary Value for the mass fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.3.17. Mole_Fraction

Scope: Inflow Boundary Condition On Surface

`Mole_Fraction` *`Species`* `{=|are|is}` *`Mole fraction`*

Parameter	Value	Default
Species	`string`	-
Mole fraction	`"string"`	-

Summary Value for the mole fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.3.18. Omit Diffusion Terms

Scope: Inflow Boundary Condition On Surface

Summary Deactivate the diffusion terms on an open boundary.

Description It may be preferable to omit the diffusion terms on open open boundaries, but retain them (default) on confined open boundaries such as duct flow. This is only applied to the momentum equation.

18.3.19. Post Process Delta Area

Scope: Inflow Boundary Condition On Surface

Summary Request post processing of delta area on this mesh part.

Description Area change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of area change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaArea(n)` where for area change in the x-direction n=1, area change in the y-direction n=2, and area change in the z-direction n=3.

Note that `MaxDeltaArea(1)` will return the maximum area change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.3.20. Post Process Delta Mass Flow Rate

Scope: Inflow Boundary Condition On Surface

Summary Request post processing of delta mass flow rate on this mesh part.

Description Mass flow rate change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of mass flow rate change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this variable use `region_name.MaxDeltaMassFlowRate(0)`.

Note that `MaxDeltaMassFlowRate(0)` will return the maximum mass flow rate change if the request to post process this value was made on more than one boundary in a given region.

18.3.21. Post Process Delta Pressure

Scope: Inflow Boundary Condition On Surface

Summary Request post processing of delta pressure on this mesh part.

Description Pressure change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of pressure change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaPressure(n)` where for pressure change in the x-direction n=1, pressure change in the y-direction n=2, and pressure change in the z-direction n=3.

Note that `MaxDeltaPressure(1)` will return the maximum pressure change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.3.22. Post Process Delta Thrust

Scope: Inflow Boundary Condition On Surface

Summary Request post processing of delta thrust on this mesh part.

Description Thrust change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of thrust change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaThrust(n)` where for thrust change in the x-direction n=1, thrust change in the y-direction n=2, and thrust change in the z-direction n=3.

Note that `MaxDeltaThrust(1)` will return the maximum thrust change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.3.23. Postprocess

Scope: Inflow Boundary Condition On Surface

`Postprocess` *`FluxType`* `Flux Of` *`equation`* `[ As` *`aliases...`* `]`

Parameter	Value	Default
FluxType	`string`	-
equation	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

---

Summary Enable nodal flux post-processing for a given equation. Flux types can be "total", "advective", or "diffusive". Integrated fluxes will also be output to global variables for post-processing or use in other string functions.

By default the variables will be named based on the equation and sideset they are applied to as bc_FluxType_EquationName_flux_SurfaceName. However, if you want to assign a more compact or descriptive name you can provide it with

```
POSTPROCESS TOTAL FLUX OF Continuity AS mdot1
```

When post-processing enthalpy, the average temperature is also output as a global variable. If you provide only one alias the enthalpy flux uses your alias and the average temperature is automatically named. If you provide two aliases the first is used for enthalpy and the second is used for temperature.

```
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux Tavg
```

When post-processing species, the post-processor is run for each species that is solved for. This means that there is no post-processor run on the last species (which is determined by a fractional balance). If you provide aliases for the species post-processor you should provide one for each post-processed species. For example, in a problem with O_2, CO_2, and N_2 (in that order) you could use:

```
POSTPROCESS TOTAL FLUX OF SPECIES AS mdotO2 mdotCO2
```

18.3.24. Project Nodes

Scope: Inflow Boundary Condition On Surface

Summary Allow for nodal projection on the particular mesh part.

Description The nodal projection of velocity is defaulted to occur on open boundary conditions only. Wall and inflow BCs are defaulted to not be projected, even on flux inflow (i.e., `USE FLUXES`) and turbulent wall nodes. This option flags the particular mesh part to be included in the nodes to be projected.

18.3.25. Progress_Variable

Scope: Inflow Boundary Condition On Surface

`Progress_Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`real`	-

Summary Constant value for the progress variable of selected scalars.

18.3.26. Real Data For Subroutine

Scope: Inflow Boundary Condition On Surface

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.3.27. Radiation Boundary Temperature

Scope: Inflow Boundary Condition On Surface

`Radiation Boundary Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_boundary_temperature variable.

18.3.28. Radiation Boundary Temperature Field

Scope: Inflow Boundary Condition On Surface

`Radiation Boundary Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation boundary temperature.

18.3.29. Radiation Environment Temperature

Scope: Inflow Boundary Condition On Surface

`Radiation Environment Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_environment_temperature variable.

18.3.30. Radiation Environment Temperature Field

Scope: Inflow Boundary Condition On Surface

`Radiation Environment Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation environment temperature.

18.3.31. Subroutine For

Scope: Inflow Boundary Condition On Surface

`Subroutine For` *`PrimitiveVariable`* `{=|are|is}` *`Subroutine`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
Subroutine	`string`	-

Summary Name of the subroutine to use for this variable.

18.3.32. Subroutine For Emissivity

Scope: Inflow Boundary Condition On Surface

`Subroutine For Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the emissivity variable.

18.3.33. Subroutine For Mass_Fraction

Scope: Inflow Boundary Condition On Surface

`Subroutine For Mass_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mass fraction. ALL species mass fractions must be assigned by this subroutine.

18.3.34. Subroutine For Mole_Fraction

Scope: Inflow Boundary Condition On Surface

`Subroutine For Mole_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mole fractions. ALL species mole fractions must be assigned by this subroutine.

18.3.35. Subroutine For Radiation Boundary Temperature

Scope: Inflow Boundary Condition On Surface

`Subroutine For Radiation Boundary Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_boundary_temperature variable.

18.3.36. Subroutine For Radiation Environment Temperature

Scope: Inflow Boundary Condition On Surface

`Subroutine For Radiation Environment Temperature {=|are|is}`
`subroutineName`

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_environment_temperature variable.

18.3.37. Subroutine For Transmissivity

Scope: Inflow Boundary Condition On Surface

`Subroutine For Transmissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transmissivity variable.

18.3.38. Subroutine For Transparent Band Emissivity

Scope: Inflow Boundary Condition On Surface

`Subroutine For Transparent Band Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transparent emissivity band for spectral emissivity on surfaces.

18.3.39. Subroutine For Wall Temperature

Scope: Inflow Boundary Condition On Surface

`Subroutine For Wall Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the wall_temperature variable.

18.3.40. Transparent Band Emissivity

Scope: Inflow Boundary Condition On Surface

`Transparent Band Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value to use for the emissivity in the transparent band for spectral surfaces.

18.3.41. Transmissivity

Scope: Inflow Boundary Condition On Surface

`Transmissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for transmissivity variable.

18.3.42. Use Fluxes

Scope: Inflow Boundary Condition On Surface

Summary Use fluxes at a velocity-specified inflow boundary instead of Dirichlet boundary conditions.

18.3.43. Wall Temperature

Scope: Inflow Boundary Condition On Surface

`Wall Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Value for wall_temperature variable. Can be a constant or a string function of space (x,y,z), time (t) or global variables.

18.4. MASS FLUX BOUNDARY CONDITION ON SURFACE

Scope: Fuego Region

`Begin Mass Flux Boundary Condition On Surface` *`Surfacename`*

`PrimitiveVariable` `{=|are|is}` *`Value`*

`Characteristic_Length {=|are|is}` *`Characteristic length`*

`Emissivity Spectral File Name {=|are|is}` *`Name`*

`External Field For` *`VariableName`* `[{of}` *`Species`* `]{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `With Multiplier {=|are|is}` *`Multiplier`* `]`

`Emissivity {=|are|is}` *`value`*

Enthalpy {=|are|is} *value*

Function For *PrimitiveVariable* {=|are|is} *FuncName* [In The *Direction* Direction]

Function For Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Mass_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Mole_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Radiation Boundary Temperature {=|are|is} *functionName*

Function For Radiation Environment Temperature {=|are|is} *functionName*

Function For Transmissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Transparent Band Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Wall Temperature {=|are|is} *functionName* [In The *Direction* Direction]

Include Condensed Phase Convection

Inline Function For *PrimitiveVariable* {=|are|is} *FuncStr*

Integer Data For Subroutine *SubName* {=|are|is} *Values...*

Mass Flux {=|are|is} *Mass_flux*

Mass_Fraction *Species* {=|are|is} *Mass fraction*

Mole_Fraction *Species* {=|are|is} *Mole fraction*

Porosity Field {=|are|is} *porosity_field_name*

Porous Condensed Temperature Field {=|are|is} *porous_condensed_temp_field_name*

Porous Flux Field For *equation* [{of} *Species*]{=|are|is} *porous_flux_field_name*

Porous Penalty Field For *equation* [{of} *Species*]{=|are|is} *porous_penalty_field_name*

Porous Target Value Field For *equation* [{of} *Species*]{=|are|is} *porous_target_field_name*

Porous Volumetric Heat Transfer Coefficient {=|are|is} *porous_vol_xfer_field_name*

Post Process Delta Area

Post Process Delta Mass Flow Rate

Post Process Delta Pressure

Post Process Delta Thrust

Postprocess *FluxType* Flux Of *equation* [As *aliases...*]

Progress_Variable *ProgressVariableName* {=|are|is} *Value*

Real Data For Subroutine *SubName* {=|are|is} *Values...*

Radiation Boundary Temperature {=|are|is} *value*

Radiation Boundary Temperature Field {=|are|is} *FieldName*

Radiation Environment Temperature {=|are|is} *value*

Radiation Environment Temperature Field {=|are|is} *FieldName*

Specific Surface Area Field {=|are|is} *ssa_field_name*

Subroutine For *PrimitiveVariable* {=|are|is} *Subroutine*

Subroutine For Emissivity {=|are|is} *subroutineName*

Subroutine For Mass_Fraction {=|are|is} *Subroutine*

Subroutine For Mole_Fraction {=|are|is} *Subroutine*

Subroutine For Radiation Boundary Temperature {=|are|is} *subroutineName*

Subroutine For Radiation Environment Temperature {=|are|is} *subroutineName*

Subroutine For Transmissivity {=|are|is} *subroutineName*

Subroutine For Transparent Band Emissivity {=|are|is} *subroutineName*

Subroutine For Wall Temperature {=|are|is} *subroutineName*

Transparent Band Emissivity {=|are|is} *value*

Transmissivity {=|are|is} *value*

Turbulence_Intensity {=|are|is} *Turbulent intensity*

Use Porous Coupling For *equation*

Wall Temperature {=|are|is} *value*

```
End
```

Summary Defines a boundary condition with a specified mass flux on a named surface of the mesh.

Description The user can specify a mass flow rate either into or out of the domain. The mass flux should obey the sign convention: positive for flux OUT OF the domain, negative for flux INTO the domain. The specified quantity is the total flow rate through the surface, in units of mass per time (note that this is not mass flow per area). When flow is out of the domain (positive mass flux), no variables other than the mass flux need to be specified. When flow into the domain (negative mass flux), the user must provide all of the variables needed for an inflow boundary condition, including (for turbulent flow) the quantities "turbulence_intensity" and "characteristic_length". The variables provided by the user are used to compute the state of the incoming fluid (e.g. enthalpy). In addition, by computing the density at the specified state, the velocity at the inlet can be computed (since mass flux is the product of density, velocity and area). This velocity is assumed to be in the direction normal to the inlet surface. Any other velocity specified by the user is ignored.

18.4.1.

Scope: Mass Flux Boundary Condition On Surface

PrimitiveVariable {=|are|is} *Value*

Parameter	Value	Default
PrimitiveVariable	{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}	-
Value	"string"	-

Summary Value for the specified variable (in consistent units). Value can be a constant or a string function of position (x,y,z), time (t), and any global variable.

18.4.2. Characteristic_Length

Scope: Mass Flux Boundary Condition On Surface

`Characteristic_Length {=|are|is}` *`Characteristic length`*

Parameter	Value	Default
Characteristic length	`real`	`1.0`

Summary Specify characteristic length for possible entrainment on open pressure specified boundary.

18.4.3. Emissivity Spectral File Name

Scope: Mass Flux Boundary Condition On Surface

`Emissivity Spectral File Name {=|are|is}` *`Name`*

Parameter	Value	Default
Name	`string`	-

Summary Specify the file name for defining spectral radiation properties on a surface.

18.4.4. External Field For

Scope: Mass Flux Boundary Condition On Surface

`External Field For` *`VariableName`* `[{of}` *`Species`* `]{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `With Multiplier {=|are|is}` *`Multiplier`* `]`

Parameter	Value	Default
VariableName	`string`	-
Species	`string`	-
ExtFieldName	`string`	-

Summary Name of the field that is to be transferred in. For species equation, specify variable as "mass fraction of" followed by the species name for transfers associated with separated individual species field names.

18.4.5. Emissivity

Scope: Mass Flux Boundary Condition On Surface

`Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for emissivity variable.

18.4.6. Enthalpy

Scope: Mass Flux Boundary Condition On Surface

`Enthalpy {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Value for inflowing enthalpy on mass flux BC.

Description Value for inflowing enthalpy on mass flux BC. Normally it is best to specify temperature and let Fuego calculate enthalpy, but when coupling mass BCs together it is more natural and less error prone to specify enthalpy directly. The value can be either a constant or a string function of time and global variables.

18.4.7. Function For

Scope: Mass Flux Boundary Condition On Surface

`Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
FuncName	`string`	-

Summary Name of function to use for the given variable, in the specified direction.

18.4.8. Function For Emissivity

Scope: Mass Flux Boundary Condition On Surface

`Function For Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of the function to use for the emissivity in the given direction (default direction = time).

18.4.9. Function For Mass_Fraction

Scope: Mass Flux Boundary Condition On Surface

`Function For Mass_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.4.10. Function For Mole_Fraction

Scope: Mass Flux Boundary Condition On Surface

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.4.11. Function For Radiation Boundary Temperature

Scope: Mass Flux Boundary Condition On Surface

`Function For Radiation Boundary Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_boundary_temperature variable

18.4.12. Function For Radiation Environment Temperature

Scope: Mass Flux Boundary Condition On Surface

`Function For Radiation Environment Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_environment_temperature variable

18.4.13. Function For Transmissivity

Scope: Mass Flux Boundary Condition On Surface

`Function For Transmissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transmissivity variable

18.4.14. Function For Transparent Band Emissivity

Scope: Mass Flux Boundary Condition On Surface

`Function For Transparent Band Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transparent emissivity band for spectral emissivity on surfaces.

18.4.15. Function For Wall Temperature

Scope: Mass Flux Boundary Condition On Surface

`Function For Wall Temperature {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the wall_temperature variable, in the specified direction.

18.4.16. Include Condensed Phase Convection

Scope: Mass Flux Boundary Condition On Surface

Summary Include condensed phase convection contribution to enthalpy flux. Requires specifying condensed temperature, porosity, and volumetric heat transfer coefficient fields that are transferred from the porous region.

18.4.17. Inline Function For

Scope: Mass Flux Boundary Condition On Surface

`Inline Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncStr`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
FuncStr	`"string"`	-

Summary String function of x, y, z, and t to use for the given variable. The function string must be enclosed in quotes if it has spaces or commas. For example: Inline function for x_velocity = "min(1, 0.1*t)"

18.4.18. Integer Data For Subroutine

Scope: Mass Flux Boundary Condition On Surface

`Integer Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`integer...`	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.4.19. Mass Flux

Scope: Mass Flux Boundary Condition On Surface

`Mass Flux {=|are|is}` *`Mass_flux`*

Parameter	Value	Default
Mass_flux	`"string"`	-

Summary Value for the specified mass flux in consistent units. The value can be a constant or a function of time and global variables.

18.4.20. Mass_Fraction

Scope: Mass Flux Boundary Condition On Surface

`Mass_Fraction` *`Species`* `{=|are|is}` *`Mass fraction`*

Parameter	Value	Default
Species	`string`	-
Mass fraction	`"string"`	-

Summary Value for the mass fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.4.21. Mole_Fraction

Scope: Mass Flux Boundary Condition On Surface

`Mole_Fraction` *`Species`* `{=|are|is}` *`Mole fraction`*

Parameter	Value	Default
Species	`string`	-
Mole fraction	`"string"`	-

Summary Value for the mole fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.4.22. Porosity Field

Scope: Mass Flux Boundary Condition On Surface

`Porosity Field {=|are|is}` *`porosity_field_name`*

Parameter	Value	Default
porosity_field_name	`string`	-

Summary Specify name of the external field containing the porosity transferred from the porous region.

18.4.23. Porous Condensed Temperature Field

Scope: Mass Flux Boundary Condition On Surface

`Porous Condensed Temperature Field {=|are|is}`
`porous_condensed_temp_field_name`

Parameter	Value	Default
porous_condensed_temp_field_name	`string`	-

Summary Specify name of the external field containing the condensed phase temperature transferred from the porous region.

18.4.24. Porous Flux Field For

Scope: Mass Flux Boundary Condition On Surface

`Porous Flux Field For` *`equation`* `[{of}` *`Species`* `]{=|are|is}` *`porous_flux_field_name`*

Parameter	Value	Default
equation	`{conserved_enthalpy|continuity|edc_product|enthalpy|mixture_fraction|nuclei|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot|species|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_momentum|x_solid_momentum|y_momentum|y_solid_momentum|z_momentum|z_solid_momentum}`	-
Species	`string`	-
porous_flux_field_name	`string`	-

Summary Specify name of the external field containing the flux transferred from the porous region for the specified equation. For continuity this should be the mass flux (rho*u) field, for enthalpy or species this should be the diffusive flux field.

18.4.25. Porous Penalty Field For

Scope: Mass Flux Boundary Condition On Surface

`Porous Penalty Field For` *`equation`* `[{of}` *`Species`* `]{=|are|is}` *`porous_penalty_field_name`*

Parameter	Value	Default
equation	{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}	-
Species	string	-
porous_penalty_field_name	string	-

Summary Specify name of the external field containing the penalty parameter transferred from the porous region for the specified equation.

18.4.26. Porous Target Value Field For

Scope: Mass Flux Boundary Condition On Surface

Porous Target Value Field For *equation* [{of} *Species*]{=|are|is} *porous_target_field_name*

Parameter	Value	Default
equation	{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}	-
Species	string	-
porous_target_field_name	string	-

Summary Specify name of the external field containing the target value transferred from the porous region for the specified equation. For continuity this should be pressure, enthalpy it should be enthalpy, and species it should be mass fraction.

18.4.27. Porous Volumetric Heat Transfer Coefficient

Scope: Mass Flux Boundary Condition On Surface

`Porous Volumetric Heat Transfer Coefficient {=|are|is}`
`porous_vol_xfer_field_name`

Parameter	Value	Default
porous_vol_xfer_field_name	`string`	-

Summary Specify name of the external field containing the volumetric heat transfer coefficient transferred from the porous region.

18.4.28. Post Process Delta Area

Scope: Mass Flux Boundary Condition On Surface

Summary Request post processing of delta area on this mesh part.

Description Area change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of area change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaArea(n)` where for area change in the x-direction n=1, area change in the y-direction n=2, and area change in the z-direction n=3.

Note that `MaxDeltaArea(1)` will return the maximum area change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.4.29. Post Process Delta Mass Flow Rate

Scope: Mass Flux Boundary Condition On Surface

Summary Request post processing of delta mass flow rate on this mesh part.

Description Mass flow rate change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of mass flow rate change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this variable use `region_name.MaxDeltaMassFlowRate(0)`.

Note that `MaxDeltaMassFlowRate(0)` will return the maximum mass flow rate change if the request to post process this value was made on more than one boundary in a given region.

18.4.30. Post Process Delta Pressure

Scope: Mass Flux Boundary Condition On Surface

Summary Request post processing of delta pressure on this mesh part.

Description Pressure change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of pressure change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaPressure(n)` where for pressure change in the x-direction n=1, pressure change in the y-direction n=2, and pressure change in the z-direction n=3.

Note that `MaxDeltaPressure(1)` will return the maximum pressure change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.4.31. Post Process Delta Thrust

Scope: Mass Flux Boundary Condition On Surface

Summary Request post processing of delta thrust on this mesh part.

Description Thrust change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of thrust change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaThrust(n)` where for thrust change in the x-direction n=1, thrust change in the y-direction n=2, and thrust change in the z-direction n=3.

Note that `MaxDeltaThrust(1)` will return the maximum thrust change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.4.32. Postprocess

Scope: Mass Flux Boundary Condition On Surface

`Postprocess` *`FluxType`* `Flux Of` *`equation`* `[ As` *`aliases...`* `]`

Parameter	Value	Default
FluxType	`string`	-
equation	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

---

Summary

Enable nodal flux post-processing for a given equation. Flux types can be "total", "advective", or "diffusive". Integrated fluxes will also be output to global variables for post-processing or use in other string functions.

By default the variables will be named based on the equation and sideset they are applied to as bc_FluxType_EquationName_flux_SurfaceName. However, if you want to assign a more compact or descriptive name you can provide it with

```
POSTPROCESS TOTAL FLUX OF Continuity AS mdot1
```

When post-processing enthalpy, the average temperature is also output as a global variable. If you provide only one alias the enthalpy flux uses your alias and the average temperature is automatically named. If you provide two aliases the first is used for enthalpy and the second is used for temperature.

```
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux Tavg
```

When post-processing species, the post-processor is run for each species that is solved for. This means that there is no post-processor run on the last species (which is determined by a fractional balance). If you provide aliases for the species post-processor you should provide one for each post-processed species. For example, in a problem with O_2, CO_2, and N_2 (in that order) you could use:

```
POSTPROCESS TOTAL FLUX OF SPECIES AS mdotO2 mdotCO2
```

18.4.33. Progress_Variable

Scope: Mass Flux Boundary Condition On Surface

`Progress_Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`real`	-

Summary Constant value for the progress variable of selected scalars.

18.4.34. Real Data For Subroutine

Scope: Mass Flux Boundary Condition On Surface

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.4.35. Radiation Boundary Temperature

Scope: Mass Flux Boundary Condition On Surface

`Radiation Boundary Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_boundary_temperature variable.

18.4.36. Radiation Boundary Temperature Field

Scope: Mass Flux Boundary Condition On Surface

`Radiation Boundary Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation boundary temperature.

18.4.37. Radiation Environment Temperature

Scope: Mass Flux Boundary Condition On Surface

`Radiation Environment Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_environment_temperature variable.

18.4.38. Radiation Environment Temperature Field

Scope: Mass Flux Boundary Condition On Surface

`Radiation Environment Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation environment temperature.

18.4.39. Specific Surface Area Field

Scope: Mass Flux Boundary Condition On Surface

`Specific Surface Area Field {=|are|is}` *`ssa_field_name`*

Parameter	Value	Default
ssa_field_name	`string`	-

Summary Specify name of the external field containing the specific surface area transferred from the porous region.

18.4.40. Subroutine For

Scope: Mass Flux Boundary Condition On Surface

`Subroutine For` *`PrimitiveVariable`* `{=|are|is}` *`Subroutine`*

Parameter	Value	Default
PrimitiveVariable	{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}	-
Subroutine	string	-

Summary Name of the subroutine to use for this variable.

18.4.41. Subroutine For Emissivity

Scope: Mass Flux Boundary Condition On Surface

`Subroutine For Emissivity {=|are|is}` *subroutineName*

Parameter	Value	Default
subroutineName	string	-

Summary Name of the user subroutine to use for the emissivity variable.

18.4.42. Subroutine For Mass_Fraction

Scope: Mass Flux Boundary Condition On Surface

`Subroutine For Mass_Fraction {=|are|is}` *Subroutine*

Parameter	Value	Default
Subroutine	string	-

Summary Name of the subroutine to use for the mass fraction. ALL species mass fractions must be assigned by this subroutine.

18.4.43. Subroutine For Mole_Fraction

Scope: Mass Flux Boundary Condition On Surface

`Subroutine For Mole_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mole fractions. ALL species mole fractions must be assigned by this subroutine.

18.4.44. Subroutine For Radiation Boundary Temperature

Scope: Mass Flux Boundary Condition On Surface

`Subroutine For Radiation Boundary Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_boundary_temperature variable.

18.4.45. Subroutine For Radiation Environment Temperature

Scope: Mass Flux Boundary Condition On Surface

`Subroutine For Radiation Environment Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_environment_temperature variable.

18.4.46. Subroutine For Transmissivity

Scope: Mass Flux Boundary Condition On Surface

`Subroutine For Transmissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transmissivity variable.

18.4.47. Subroutine For Transparent Band Emissivity

Scope: Mass Flux Boundary Condition On Surface

`Subroutine For Transparent Band Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transparent emissivity band for spectral emissivity on surfaces.

18.4.48. Subroutine For Wall Temperature

Scope: Mass Flux Boundary Condition On Surface

`Subroutine For Wall Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the wall_temperature variable.

18.4.49. Transparent Band Emissivity

Scope: Mass Flux Boundary Condition On Surface

`Transparent Band Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value to use for the emissivity in the transparent band for spectral surfaces.

18.4.50. Transmissivity

Scope: Mass Flux Boundary Condition On Surface

`Transmissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for transmissivity variable.

18.4.51. Turbulence_Intensity

Scope: Mass Flux Boundary Condition On Surface

`Turbulence_Intensity {=|are|is}` *`Turbulent intensity`*

Parameter	Value	Default
Turbulent intensity	`real`	-

Summary Specify turbulent intensity for use in the calculation for the entrainment value of turbulent kinetic energy when there is possible entrainment on open pressure specified boundary.

18.4.52. Use Porous Coupling For

Scope: Mass Flux Boundary Condition On Surface

`Use Porous Coupling For` *`equation`*

Parameter	Value	Default
equation	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

Summary Enable porous fluid coupling algorithm for specified equation.

18.4.53. Wall Temperature

Scope: Mass Flux Boundary Condition On Surface

`Wall Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Value for wall_temperature variable. Can be a constant or a string function of space (x,y,z), time (t) or global variables.

18.5. OPEN BOUNDARY CONDITION ON SURFACE

Scope: Fuego Region

Begin Open Boundary Condition On Surface *Surfacename*

PrimitiveVariable {=|are|is} *Value*

Apply Pressure Penalty [Of *PenaltyValue*]

Characteristic_Length {=|are|is} *Characteristic length*

Emissivity Spectral File Name {=|are|is} *Name*

Entrain Local Values

External Field For *VariableName* [{of} *Species*]{=|are|is} *ExtFieldName* [Of Size *Value* With Multiplier {=|are|is} *Multiplier*]

Emissivity {=|are|is} *value*

Flow Must Exit Domain

Function For *PrimitiveVariable* {=|are|is} *FuncName* [In The *Direction* Direction]

Function For Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Mass_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Mole_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Radiation Boundary Temperature {=|are|is} *functionName*

Function For Radiation Environment Temperature {=|are|is} *functionName*

Function For Transmissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Transparent Band Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Total_Pressure {=|are|is} *Pressure function* In The *Direction* Direction

Function For Wall Temperature {=|are|is} *functionName* [In The *Direction* Direction]

Inline Function For *PrimitiveVariable* {=|are|is} *FuncStr*

Integer Data For Subroutine *SubName* {=|are|is} *Values...*

Momentum Entrainment {=|are|is} *EntrainmentMethod*

Mass_Fraction *Species* {=|are|is} *Mass fraction*

Mole_Fraction *Species* {=|are|is} *Mole fraction*

Omit Diffusion Terms

Post Process Delta Area

Post Process Delta Mass Flow Rate

Post Process Delta Pressure

Post Process Delta Thrust

Postprocess *FluxType* Flux Of *equation* [As *aliases...*]

Progress_Variable *ProgressVariableName* {=|are|is} *Value*

Real Data For Subroutine *SubName* {=|are|is} *Values...*

Radiation Boundary Temperature {=|are|is} *value*

Radiation Boundary Temperature Field {=|are|is} *FieldName*

Radiation Environment Temperature {=|are|is} *value*

Radiation Environment Temperature Field {=|are|is} *FieldName*

Sponge Layer Blending Function {=|are|is} *Value*

Sponge Layer Thickness {=|are|is} *Value*

Sponge Layer Viscosity {=|are|is} *Value*

Sponge Layer Viscosity Type {=|are|is} *Value*

Subroutine For *PrimitiveVariable* {=|are|is} *Subroutine*

Subroutine For Emissivity {=|are|is} *subroutineName*

Subroutine For Mass_Fraction {=|are|is} *Subroutine*

Subroutine For Mole_Fraction {=|are|is} *Subroutine*

Subroutine For Radiation Boundary Temperature {=|are|is} *subroutineName*

Subroutine For Radiation Environment Temperature {=|are|is} *subroutineName*

Subroutine For Transmissivity {=|are|is} *subroutineName*

Subroutine For Transparent Band Emissivity {=|are|is} *subroutineName*

```
    Subroutine For Total_Pressure {=|are|is} Subroutine

    Subroutine For Wall Temperature {=|are|is} subroutineName

    Transparent Band Emissivity {=|are|is} value

    Total_Pressure {=|are|is} Pressure

    Transmissivity {=|are|is} value

    Turbulence_Intensity {=|are|is} Turbulent intensity

    Use Sponge Layer

    Wall Temperature {=|are|is} value


End
```

Summary Defines an open (specified pressure) boundary condition on a named surface of the mesh. Optionally, specify far-field entrainment values for scalars. Any primitive variable line command below is used to specify these far-field entrainment values. One may also specify a total pressure in which case, the dynamic component will be removed for use in the pressure gradient and mass flux routine. Lastly, for turbulent flows, if the turbulence quantities are not set, then one must specify at least a characteristic length or also specify a turbulence intensity (10% assumed). These values are used to determine an entrainment value for turbulence quantities.

18.5.1.

Scope: Open Boundary Condition On Surface

PrimitiveVariable {=|are|is} *Value*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}`	-
Value	`"string"`	-

Summary Value for the specified variable (in consistent units). Value can be a constant or a string function of position (x,y,z), time (t), and any global variable.

18.5.2. Apply Pressure Penalty

Scope: Open Boundary Condition On Surface

Summary Apply a penalty term on the open boundary pressure.

Description Apply a DG-like penalty term to the open boundary pressure. This can improve stability for tets and non-orthogonal meshes on open boundaries.

18.5.3. Characteristic_Length

Scope: Open Boundary Condition On Surface

`Characteristic_Length {=|are|is}` *`Characteristic length`*

Parameter	Value	Default
Characteristic length	`real`	`1.0`

Summary Specify characteristic length for possible entrainment on open pressure specified boundary.

18.5.4. Emissivity Spectral File Name

Scope: Open Boundary Condition On Surface

`Emissivity Spectral File Name {=|are|is}` *`Name`*

Parameter	Value	Default
Name	`string`	-

Summary Specify the file name for defining spectral radiation properties on a surface.

18.5.5. Entrain Local Values

Scope: Open Boundary Condition On Surface

Summary If the flow re-circulates at the open, entrain the local values rather than specified far-field values. This is applied to scalars at the open (not to velocity or pressure) and if activated will entrain the local scalar value from the previous time step rather than the user-specified far-field value.

18.5.6. External Field For

Scope: Open Boundary Condition On Surface

`External Field For` *`VariableName`* `[{of}` *`Species`* `]{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `With Multiplier {=|are|is}` *`Multiplier`* `]`

Parameter	Value	Default
VariableName	`string`	-
Species	`string`	-
ExtFieldName	`string`	-

Summary Name of the field that is to be transferred in. For species equation, specify variable as "mass fraction of" followed by the species name for transfers associated with separated individual species field names.

18.5.7. Emissivity

Scope: Open Boundary Condition On Surface

`Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for emissivity variable.

18.5.8. Flow Must Exit Domain

Scope: Open Boundary Condition On Surface

Summary This command is deprecated and will no longer have any effect. It will be completely removed in a future version of Fuego.

18.5.9. Function For

Scope: Open Boundary Condition On Surface

`Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
PrimitiveVariable	{contact_angle|edc_product |gas_volume_fraction| mixture_fraction|pressure| progress_variable|scalar_variance |second_mixture_fraction |solid_volume_fraction |soot_mass_fraction| soot_nuclei_mass_fraction| temperature|turbulent_dissipation |turbulent_frequency| turbulent_helmholtz_function| turbulent_kinetic_energy|turbulent_v2 |volume_of_fluid|x_solid_velocity |x_velocity|y_solid_velocity| y_velocity|z_solid_velocity| z_velocity}	-
FuncName	string	-

Summary Name of function to use for the given variable, in the specified direction.

18.5.10. Function For Emissivity

Scope: Open Boundary Condition On Surface

`Function For Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	string	-

Summary Name of the function to use for the emissivity in the given direction (default direction = time).

18.5.11. Function For Mass_Fraction

Scope: Open Boundary Condition On Surface

`Function For Mass_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`*
`Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.5.12. Function For Mole_Fraction

Scope: Open Boundary Condition On Surface

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`*
`Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.5.13. Function For Radiation Boundary Temperature

Scope: Open Boundary Condition On Surface

`Function For Radiation Boundary Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_boundary_temperature variable

18.5.14. Function For Radiation Environment Temperature

Scope: Open Boundary Condition On Surface

`Function For Radiation Environment Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_environment_temperature variable

18.5.15. Function For Transmissivity

Scope: Open Boundary Condition On Surface

`Function For Transmissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transmissivity variable

18.5.16. Function For Transparent Band Emissivity

Scope: Open Boundary Condition On Surface

`Function For Transparent Band Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transparent emissivity band for spectral emissivity on surfaces.

18.5.17. Function For Total_Pressure

Scope: Open Boundary Condition On Surface

`Function For Total_Pressure {=|are|is}` *`Pressure function`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
Pressure function	`string`	-
Direction	`{t|x|y|z}`	-

Summary total pressure function name

18.5.18. Function For Wall Temperature

Scope: Open Boundary Condition On Surface

`Function For Wall Temperature {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the wall_temperature variable, in the specified direction.

18.5.19. Inline Function For

Scope: Open Boundary Condition On Surface

`Inline Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncStr`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
FuncStr	`"string"`	-

Summary String function of x, y, z, and t to use for the given variable. The function string must be enclosed in quotes if it has spaces or commas. For example: Inline function for x_velocity = "min(1, 0.1*t)"

18.5.20. Integer Data For Subroutine

Scope: Open Boundary Condition On Surface

`Integer Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`integer...`	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.5.21. Momentum Entrainment

Scope: Open Boundary Condition On Surface

`Momentum Entrainment {=|are|is}` *`EntrainmentMethod`*

Parameter	Value	Default
EntrainmentMethod	`{computed|specified|tangential}`	-

Summary Select how momentum is entrained at the open BC.

Description The available options are COMPUTED, TANGENTIAL, or SPECIFIED (default).

The default momentum entrainment (SPECIFIED) entrains the user-specified velocities at the open when flow enters the domain.

The computed momentum entrainment (COMPUTED) calculates a normal velocity component using the local mass flux and density, and a tangential component from the user specification on the boundary. This can have stability issues at high vorticity, but often gives better behavior for open BCs that are primarily inflowing or entraining.

It may be preferable to omit the normal component of entrained momentum at an open BC that is primarily exiting the domain (TANGENTIAL) or to simply entrain the user-specified velocities (SPECIFIED).

18.5.22. Mass_Fraction

Scope: Open Boundary Condition On Surface

`Mass_Fraction` *`Species`* `{=|are|is}` *`Mass fraction`*

Parameter	Value	Default
Species	`string`	-
Mass fraction	`"string"`	-

Summary Value for the mass fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.5.23. Mole_Fraction

Scope: Open Boundary Condition On Surface

`Mole_Fraction` *`Species`* `{=|are|is}` *`Mole fraction`*

Parameter	Value	Default
Species	`string`	-
Mole fraction	`"string"`	-

Summary Value for the mole fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.5.24. Omit Diffusion Terms

Scope: Open Boundary Condition On Surface

Summary Deactivate the diffusion terms on an open boundary.

Description It may be preferable to omit the diffusion terms on open open boundaries, but retain them (default) on confined open boundaries such as duct flow. This is only applied to the momentum equation.

18.5.25. Post Process Delta Area

Scope: Open Boundary Condition On Surface

Summary Request post processing of delta area on this mesh part.

Description Area change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of area change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaArea(n)` where for area change in the x-direction n=1, area change in the y-direction n=2, and area change in the z-direction n=3.

Note that `MaxDeltaArea(1)` will return the maximum area change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.5.26. Post Process Delta Mass Flow Rate

Scope: Open Boundary Condition On Surface

Summary Request post processing of delta mass flow rate on this mesh part.

Description Mass flow rate change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of mass flow rate change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this variable use `region_name.MaxDeltaMassFlowRate(0)`.

Note that `MaxDeltaMassFlowRate(0)` will return the maximum mass flow rate change if the request to post process this value was made on more than one boundary in a given region.

18.5.27. Post Process Delta Pressure

Scope: Open Boundary Condition On Surface

Summary Request post processing of delta pressure on this mesh part.

Description Pressure change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of pressure change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaPressure(n)` where for pressure change in the x-direction n=1, pressure change in the y-direction n=2, and pressure change in the z-direction n=3.

Note that `MaxDeltaPressure(1)` will return the maximum pressure change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.5.28. Post Process Delta Thrust

Scope: Open Boundary Condition On Surface

Summary Request post processing of delta thrust on this mesh part.

Description Thrust change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of thrust change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaThrust(n)` where for thrust change in the x-direction n=1, thrust change in the y-direction n=2, and thrust change in the z-direction n=3.

Note that `MaxDeltaThrust(1)` will return the maximum thrust change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.5.29. Postprocess

Scope: Open Boundary Condition On Surface

`Postprocess` *`FluxType`* `Flux Of` *`equation`* `[ As` *`aliases...`* `]`

Parameter	Value	Default
FluxType	`string`	-
equation	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

Summary

Enable nodal flux post-processing for a given equation. Flux types can be "total", "advective", or "diffusive". Integrated fluxes will also be output to global variables for post-processing or use in other string functions.

By default the variables will be named based on the equation and sideset they are applied to as bc_FluxType_EquationName_flux_SurfaceName. However, if you want to assign a more compact or descriptive name you can provide it with

```
POSTPROCESS TOTAL FLUX OF Continuity AS mdot1
```

When post-processing enthalpy, the average temperature is also output as a global variable. If you provide only one alias the enthalpy flux uses your alias and the average temperature is automatically named. If you provide two aliases the first is used for enthalpy and the second is used for temperature.

```
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux Tavg
```

When post-processing species, the post-processor is run for each species that is solved for. This means that there is no post-processor run on the last species (which is determined by a fractional balance). If you provide aliases for the species post-processor you should provide one for each post-processed species. For example, in a problem with O_2, CO_2, and N_2 (in that order) you could use:

```
POSTPROCESS TOTAL FLUX OF SPECIES AS mdotO2 mdotCO2
```

18.5.30. Progress_Variable

Scope: Open Boundary Condition On Surface

`Progress_Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`real`	-

Summary Constant value for the progress variable of selected scalars.

18.5.31. Real Data For Subroutine

Scope: Open Boundary Condition On Surface

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.5.32. Radiation Boundary Temperature

Scope: Open Boundary Condition On Surface

`Radiation Boundary Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_boundary_temperature variable.

18.5.33. Radiation Boundary Temperature Field

Scope: Open Boundary Condition On Surface

`Radiation Boundary Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation boundary temperature.

18.5.34. Radiation Environment Temperature

Scope: Open Boundary Condition On Surface

`Radiation Environment Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_environment_temperature variable.

18.5.35. Radiation Environment Temperature Field

Scope: Open Boundary Condition On Surface

`Radiation Environment Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation environment temperature.

18.5.36. Sponge Layer Blending Function

Scope: Open Boundary Condition On Surface

`Sponge Layer Blending Function {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`{heaviside|linear}`	-

Summary Options are Heaviside or Linear. Linear gradually applies the sponge forcing across the layer, and Heaviside applies the full sponge forcing across the entier sponge layer. Default is heaviside.

18.5.37. Sponge Layer Thickness

Scope: Open Boundary Condition On Surface

`Sponge Layer Thickness {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Distance the sponge layer will propogate into the domain normal to the open boundary condition's surface

18.5.38. Sponge Layer Viscosity

Scope: Open Boundary Condition On Surface

`Sponge Layer Viscosity {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	-

Summary Value of the viscosity applied within the sponge layer

18.5.39. Sponge Layer Viscosity Type

Scope: Open Boundary Condition On Surface

`Sponge Layer Viscosity Type {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`{absolute|multiple}`	-

Summary Options are multiplie or absolute. The Multiple option uses the product of the molecular visocisty specified sponge layer viscosity to calcualte the forcing term. The Absolute option just uses the specified sponge layer viscosity for calculating the forcing term. Multiple is the default

18.5.40. Subroutine For

Scope: Open Boundary Condition On Surface

`Subroutine For` *`PrimitiveVariable`* `{=|are|is}` *`Subroutine`*

Parameter	Value	Default
PrimitiveVariable	{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}	-
Subroutine	string	-

Summary Name of the subroutine to use for this variable.

18.5.41. Subroutine For Emissivity

Scope: Open Boundary Condition On Surface

`Subroutine For Emissivity {=|are|is}` *subroutineName*

Parameter	Value	Default
subroutineName	string	-

Summary Name of the user subroutine to use for the emissivity variable.

18.5.42. Subroutine For Mass_Fraction

Scope: Open Boundary Condition On Surface

`Subroutine For Mass_Fraction {=|are|is}` *Subroutine*

Parameter	Value	Default
Subroutine	string	-

Summary Name of the subroutine to use for the mass fraction. ALL species mass fractions must be assigned by this subroutine.

18.5.43. Subroutine For Mole_Fraction

Scope: Open Boundary Condition On Surface

`Subroutine For Mole_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mole fractions. ALL species mole fractions must be assigned by this subroutine.

18.5.44. Subroutine For Radiation Boundary Temperature

Scope: Open Boundary Condition On Surface

`Subroutine For Radiation Boundary Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_boundary_temperature variable.

18.5.45. Subroutine For Radiation Environment Temperature

Scope: Open Boundary Condition On Surface

`Subroutine For Radiation Environment Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_environment_temperature variable.

18.5.46. Subroutine For Transmissivity

Scope: Open Boundary Condition On Surface

`Subroutine For Transmissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transmissivity variable.

18.5.47. Subroutine For Transparent Band Emissivity

Scope: Open Boundary Condition On Surface

`Subroutine For Transparent Band Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transparent emissivity band for spectral emissivity on surfaces.

18.5.48. Subroutine For Total_Pressure

Scope: Open Boundary Condition On Surface

`Subroutine For Total_Pressure {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for pressure.

18.5.49. Subroutine For Wall Temperature

Scope: Open Boundary Condition On Surface

`Subroutine For Wall Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the wall_temperature variable.

18.5.50. Transparent Band Emissivity

Scope: Open Boundary Condition On Surface

`Transparent Band Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value to use for the emissivity in the transparent band for spectral surfaces.

18.5.51. Total_Pressure

Scope: Open Boundary Condition On Surface

`Total_Pressure {=|are|is}` *`Pressure`*

Parameter	Value	Default
Pressure	`real`	-

Summary Constant value for the total pressure.

18.5.52. Transmissivity

Scope: Open Boundary Condition On Surface

`Transmissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for transmissivity variable.

18.5.53. Turbulence_Intensity

Scope: Open Boundary Condition On Surface

`Turbulence_Intensity {=|are|is}` *`Turbulent intensity`*

Parameter	Value	Default
Turbulent intensity	`real`	-

Summary Specify turbulent intensity for use in the calculation for the entrainment value of turbulent kinetic energy when there is possible entrainment on open pressure specified boundary.

18.5.54. Use Sponge Layer

Scope: Open Boundary Condition On Surface

Summary Indicator that a sponge layer should be implemented next to this open boundary condition

18.5.55. Wall Temperature

Scope: Open Boundary Condition On Surface

`Wall Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Value for wall_temperature variable. Can be a constant or a string function of space (x,y,z), time (t) or global variables.

18.6. SYMMETRY BOUNDARY CONDITION ON SURFACE

Scope: Fuego Region

`Begin Symmetry Boundary Condition On Surface` *`Surfacename`*

`PrimitiveVariable` `{=|are|is}` *`Value`*

`Emissivity Spectral File Name {=|are|is}` *`Name`*

`External Field For` *`VariableName`* `[{of}` *`Species`* `]{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `With Multiplier {=|are|is}` *`Multiplier`* `]`

`Emissivity {=|are|is}` *`value`*

`Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

`Function For Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

`Function For Mass_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

`Function For Radiation Boundary Temperature {=|are|is}` *`functionName`*

`Function For Radiation Environment Temperature {=|are|is}` *`functionName`*

`Function For Transmissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Function For Transparent Band Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Wall Temperature {=|are|is} *functionName* [In The *Direction* Direction]

Inline Function For *PrimitiveVariable* {=|are|is} *FuncStr*

Integer Data For Subroutine *SubName* {=|are|is} *Values...*

Mass_Fraction *Species* {=|are|is} *Mass fraction*

Mole_Fraction *Species* {=|are|is} *Mole fraction*

Postprocess *FluxType* Flux Of *equation* [As *aliases...*]

Progress_Variable *ProgressVariableName* {=|are|is} *Value*

Real Data For Subroutine *SubName* {=|are|is} *Values...*

Radiation Boundary Temperature {=|are|is} *value*

Radiation Boundary Temperature Field {=|are|is} *FieldName*

Radiation Environment Temperature {=|are|is} *value*

Radiation Environment Temperature Field {=|are|is} *FieldName*

Subroutine For *PrimitiveVariable* {=|are|is} *Subroutine*

Subroutine For Emissivity {=|are|is} *subroutineName*

Subroutine For Mass_Fraction {=|are|is} *Subroutine*

Subroutine For Mole_Fraction {=|are|is} *Subroutine*

Subroutine For Radiation Boundary Temperature {=|are|is} *subroutineName*

Subroutine For Radiation Environment Temperature {=|are|is} *subroutineName*

Subroutine For Transmissivity {=|are|is} *subroutineName*

Subroutine For Transparent Band Emissivity {=|are|is} *subroutineName*

Subroutine For Wall Temperature {=|are|is} *subroutineName*

Transparent Band Emissivity {=|are|is} *value*

Transmissivity {=|are|is} *value*

Wall Temperature {=|are|is} *value*

End

Summary Defines a symmetry plane boundary condition on a named surface of the mesh.

18.6.1.

Scope: Symmetry Boundary Condition On Surface

PrimitiveVariable {=|are|is} *Value*

Parameter	Value	Default
PrimitiveVariable	{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}	-
Value	"string"	-

Summary Value for the specified variable (in consistent units). Value can be a constant or a string function of position (x,y,z), time (t), and any global variable.

18.6.2. Emissivity Spectral File Name

Scope: Symmetry Boundary Condition On Surface

Emissivity Spectral File Name {=|are|is} *Name*

Parameter	Value	Default
Name	string	-

Summary Specify the file name for defining spectral radiation properties on a surface.

18.6.3. External Field For

Scope: Symmetry Boundary Condition On Surface

`External Field For` *`VariableName`* `[{of}` *`Species`* `]{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `With Multiplier {=|are|is}` *`Multiplier`* `]`

Parameter	Value	Default
VariableName	`string`	-
Species	`string`	-
ExtFieldName	`string`	-

Summary Name of the field that is to be transferred in. For species equation, specify variable as "mass fraction of" followed by the species name for transfers associated with separated individual species field names.

18.6.4. Emissivity

Scope: Symmetry Boundary Condition On Surface

`Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for emissivity variable.

18.6.5. Function For

Scope: Symmetry Boundary Condition On Surface

`Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
PrimitiveVariable	`{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}`	-
FuncName	`string`	-

Summary Name of function to use for the given variable, in the specified direction.

18.6.6. Function For Emissivity

Scope: Symmetry Boundary Condition On Surface

`Function For Emissivity {= | are | is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of the function to use for the emissivity in the given direction (default direction = time).

18.6.7. Function For Mass_Fraction

Scope: Symmetry Boundary Condition On Surface

`Function For Mass_Fraction {= | are | is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t | x | y | z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.6.8. Function For Mole_Fraction

Scope: Symmetry Boundary Condition On Surface

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.6.9. Function For Radiation Boundary Temperature

Scope: Symmetry Boundary Condition On Surface

`Function For Radiation Boundary Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_boundary_temperature variable

18.6.10. Function For Radiation Environment Temperature

Scope: Symmetry Boundary Condition On Surface

`Function For Radiation Environment Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_environment_temperature variable

18.6.11. Function For Transmissivity

Scope: Symmetry Boundary Condition On Surface

`Function For Transmissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transmissivity variable

18.6.12. Function For Transparent Band Emissivity

Scope: Symmetry Boundary Condition On Surface

`Function For Transparent Band Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transparent emissivity band for spectral emissivity on surfaces.

18.6.13. Function For Wall Temperature

Scope: Symmetry Boundary Condition On Surface

`Function For Wall Temperature {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the wall_temperature variable, in the specified direction.

18.6.14. Inline Function For

Scope: Symmetry Boundary Condition On Surface

`Inline Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncStr`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}`	-
FuncStr	`"string"`	-

Summary String function of x, y, z, and t to use for the given variable. The function string must be enclosed in quotes if it has spaces or commas. For example: Inline function for x_velocity = "min(1, 0.1*t)"

18.6.15. Integer Data For Subroutine

Scope: Symmetry Boundary Condition On Surface

`Integer Data For Subroutine` *`SubName`* `{= | are | is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`integer...`	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.6.16. Mass_Fraction

Scope: Symmetry Boundary Condition On Surface

`Mass_Fraction` *`Species`* `{= | are | is}` *`Mass fraction`*

Parameter	Value	Default
Species	`string`	-
Mass fraction	`"string"`	-

Summary Value for the mass fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.6.17. Mole_Fraction

Scope: Symmetry Boundary Condition On Surface

`Mole_Fraction` *`Species`* `{=|are|is}` *`Mole fraction`*

Parameter	Value	Default
Species	`string`	-
Mole fraction	`"string"`	-

Summary Value for the mole fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.6.18. Postprocess

Scope: Symmetry Boundary Condition On Surface

`Postprocess` *`FluxType`* `Flux Of` *`equation`* `[ As` *`aliases...`* `]`

Parameter	Value	Default
FluxType	`string`	-
equation	`{conserved_enthalpy|continuity|edc_product|enthalpy|mixture_fraction|nuclei|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot|species|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_momentum|x_solid_momentum|y_momentum|y_solid_momentum|z_momentum|z_solid_momentum}`	-

Summary Enable nodal flux post-processing for a given equation. Flux types can be "total", "advective", or "diffusive". Integrated fluxes will also be output to global variables for post-processing or use in other string functions.

By default the variables will be named based on the equation and sideset they are applied to as bc_FluxType_EquationName_flux_SurfaceName. However, if you want to assign a more compact or descriptive name you can provide it with

```
POSTPROCESS TOTAL FLUX OF Continuity AS mdot1
```

When post-processing enthalpy, the average temperature is also output as a global variable. If you provide only one alias the enthalpy flux uses your alias and the average temperature is automatically named. If you provide two aliases the first is used for enthalpy and the second is used for temperature.

```
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux Tavg
```

When post-processing species, the post-processor is run for each species that is solved for. This means that there is no post-processor run on the last species (which is determined by a fractional balance). If you provide aliases for the species post-processor you should provide one for each post-processed species. For example, in a problem with O_2, CO_2, and N_2 (in that order) you could use:

```
POSTPROCESS TOTAL FLUX OF SPECIES AS mdotO2 mdotCO2
```

18.6.19. Progress_Variable

Scope: Symmetry Boundary Condition On Surface

`Progress_Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`real`	-

Summary Constant value for the progress variable of selected scalars.

18.6.20. Real Data For Subroutine

Scope: Symmetry Boundary Condition On Surface

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.6.21. Radiation Boundary Temperature

Scope: Symmetry Boundary Condition On Surface

`Radiation Boundary Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_boundary_temperature variable.

18.6.22. Radiation Boundary Temperature Field

Scope: Symmetry Boundary Condition On Surface

`Radiation Boundary Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation boundary temperature.

18.6.23. Radiation Environment Temperature

Scope: Symmetry Boundary Condition On Surface

`Radiation Environment Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_environment_temperature variable.

18.6.24. Radiation Environment Temperature Field

Scope: Symmetry Boundary Condition On Surface

`Radiation Environment Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation environment temperature.

18.6.25. Subroutine For

Scope: Symmetry Boundary Condition On Surface

`Subroutine For` *`PrimitiveVariable`* `{=|are|is}` *`Subroutine`*

Parameter	Value	Default
PrimitiveVariable	{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}	-
Subroutine	string	-

Summary Name of the subroutine to use for this variable.

18.6.26. Subroutine For Emissivity

Scope: Symmetry Boundary Condition On Surface

`Subroutine For Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	string	-

Summary Name of the user subroutine to use for the emissivity variable.

18.6.27. Subroutine For Mass_Fraction

Scope: Symmetry Boundary Condition On Surface

`Subroutine For Mass_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	string	-

Summary Name of the subroutine to use for the mass fraction. ALL species mass fractions must be assigned by this subroutine.

18.6.28. Subroutine For Mole_Fraction

Scope: Symmetry Boundary Condition On Surface

`Subroutine For Mole_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mole fractions. ALL species mole fractions must be assigned by this subroutine.

18.6.29. Subroutine For Radiation Boundary Temperature

Scope: Symmetry Boundary Condition On Surface

`Subroutine For Radiation Boundary Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_boundary_temperature variable.

18.6.30. Subroutine For Radiation Environment Temperature

Scope: Symmetry Boundary Condition On Surface

`Subroutine For Radiation Environment Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_environment_temperature variable.

18.6.31. Subroutine For Transmissivity

Scope: Symmetry Boundary Condition On Surface

`Subroutine For Transmissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transmissivity variable.

18.6.32. Subroutine For Transparent Band Emissivity

Scope: Symmetry Boundary Condition On Surface

`Subroutine For Transparent Band Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transparent emissivity band for spectral emissivity on surfaces.

18.6.33. Subroutine For Wall Temperature

Scope: Symmetry Boundary Condition On Surface

`Subroutine For Wall Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the wall_temperature variable.

18.6.34. Transparent Band Emissivity

Scope: Symmetry Boundary Condition On Surface

`Transparent Band Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value to use for the emissivity in the transparent band for spectral surfaces.

18.6.35. Transmissivity

Scope: Symmetry Boundary Condition On Surface

`Transmissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for transmissivity variable.

18.6.36. Wall Temperature

Scope: Symmetry Boundary Condition On Surface

`Wall Temperature {=|are|is}` *value*

Parameter	Value	Default
value	"string"	-

Summary Value for wall_temperature variable. Can be a constant or a string function of space (x,y,z), time (t) or global variables.

18.7. WALL BOUNDARY CONDITION ON SURFACE

Scope: Fuego Region

`Begin Wall Boundary Condition On Surface` *Surfacename*

PrimitiveVariable `{=|are|is}` *Value*

`Activate Surface Flux Balance Temperature Algorithm [ With` $BacksideParamFlux_1$ `[` $BacksideParamFlux_2$ `] ]`

`Backside` *BacksideParam* `{=|are|is}` *BackVal*

`Emissivity Spectral File Name {=|are|is}` *Name*

`Enforce Zero Flux For Turbulence Wall Bc`

`External Field For` *VariableName* `[{of}` *Species* `]{=|are|is}` *ExtFieldName* `[ Of Size` *Value* `With Multiplier {=|are|is}` *Multiplier* `]`

`Emissivity {=|are|is}` *value*

`Function For` *PrimitiveVariable* `{=|are|is}` *FuncName* `[ In The` *Direction* `Direction ]`

`Function For Emissivity {=|are|is}` *functionName* `[ In The` *Direction* `Direction ]`

`Function For Mass_Fraction {=|are|is}` *FuncName* `In The` *Direction* `Direction`

`Function For Mole_Fraction {=|are|is}` *FuncName* `In The` *Direction* `Direction`

`Function For Radiation Boundary Temperature {=|are|is}` *functionName*

`Function For Radiation Environment Temperature {=|are|is}` *functionName*

Function For Transmissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Transparent Band Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Wall Temperature {=|are|is} *functionName* [In The *Direction* Direction]

Inline Function For *PrimitiveVariable* {=|are|is} *FuncStr*

Integer Data For Subroutine *SubName* {=|are|is} *Values...*

Interface Boundary

Mass_Fraction *Species* {=|are|is} *Mass fraction*

Mole_Fraction *Species* {=|are|is} *Mole fraction*

Post Process Delta Area

Post Process Delta Heat Flux

Post Process Delta Pressure

Post Process Delta Viscous Stress

Post Process Yplus

Postprocess *FluxType* Flux Of *equation* [As *aliases...*]

Project Nodes

Progress_Variable *ProgressVariableName* {=|are|is} *Value*

Real Data For Subroutine *SubName* {=|are|is} *Values...*

Radiation Boundary Temperature {=|are|is} *value*

Radiation Boundary Temperature Field {=|are|is} *FieldName*

Radiation Environment Temperature {=|are|is} *value*

Radiation Environment Temperature Field {=|are|is} *FieldName*

Subroutine For *PrimitiveVariable* {=|are|is} *Subroutine*

Subroutine For Emissivity {=|are|is} *subroutineName*

Subroutine For Mass_Fraction {=|are|is} *Subroutine*

Subroutine For Mole_Fraction {=|are|is} *Subroutine*

Subroutine For Radiation Boundary Temperature {=|are|is} *subroutineName*

Subroutine For Radiation Environment Temperature {=|are|is} *subroutineName*

```
    Subroutine For Transmissivity {=|are|is} subroutineName

    Subroutine For Transparent Band Emissivity {=|are|is}
    subroutineName

    Subroutine For Wall Temperature {=|are|is} subroutineName

    Transparent Band Emissivity {=|are|is} value

    Transmissivity {=|are|is} value

    Use Equilibrium Production Model

    Velocity Is Relative To Mesh

    Wall Temperature {=|are|is} value

    Law_Of_Wall_Roughness_Parameter {=|are|is} Law of the Wall
    Roughness parameter

End
```

Summary Defines a wall boundary condition on a named surface of the mesh.

18.7.1.

Scope: Wall Boundary Condition On Surface

PrimitiveVariable {=|are|is} *Value*

Parameter	Value	Default
PrimitiveVariable	{contact_angle|edc_product |gas_volume_fraction| mixture_fraction|pressure| progress_variable|scalar_variance |second_mixture_fraction |solid_volume_fraction |soot_mass_fraction| soot_nuclei_mass_fraction| temperature|turbulent_dissipation |turbulent_frequency| turbulent_helmholtz_function| turbulent_kinetic_energy|turbulent_v2 |volume_of_fluid|x_solid_velocity |x_velocity|y_solid_velocity| y_velocity|z_solid_velocity| z_velocity}	-
Value	"string"	-

Summary Value for the specified variable (in consistent units). Value can be a constant or a string function of position (x,y,z), time (t), and any global variable.

18.7.2. Activate Surface Flux Balance Temperature Algorithm

Scope: Wall Boundary Condition On Surface

Summary Compute a surface balance equation for nodal temperature.

Description To be done

18.7.3. Backside

Scope: Wall Boundary Condition On Surface

`Backside` *`BacksideParam`* `{=|are|is}` *`BackVal`*

Parameter	Value	Default
BacksideParam	`{dx|h|k|too}`	-
BackVal	`real`	-

Summary Specify h, T, k and dx.

Description To be done

18.7.4. Emissivity Spectral File Name

Scope: Wall Boundary Condition On Surface

`Emissivity Spectral File Name {=|are|is}` *`Name`*

Parameter	Value	Default
Name	`string`	-

Summary Specify the file name for defining spectral radiation properties on a surface.

18.7.5. Enforce Zero Flux For Turbulence Wall Bc

Scope: Wall Boundary Condition On Surface

Summary Enforce no operation for turbulence wall bc.

Description All turbulence equations receive a no operation (zero flux) while momentum receives a Dirichlet. Enthalpy equation receives a Dirichlet if a temperature is specified.

This command is only intended to be used for boundaries of a block where turbulent source terms are being skipped. It is deprecated, and this is now determined automatically from the mesh.

18.7.6. External Field For

Scope: Wall Boundary Condition On Surface

`External Field For` *`VariableName`* `[{of}` *`Species`* `]{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `With Multiplier {=|are|is}` *`Multiplier`* `]`

Parameter	Value	Default
VariableName	`string`	-
Species	`string`	-
ExtFieldName	`string`	-

Summary Name of the field that is to be transferred in. For species equation, specify variable as "mass fraction of" followed by the species name for transfers associated with separated individual species field names.

18.7.7. Emissivity

Scope: Wall Boundary Condition On Surface

`Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for emissivity variable.

18.7.8. Function For

Scope: Wall Boundary Condition On Surface

`Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
PrimitiveVariable	`{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}`	-
FuncName	`string`	-

Summary Name of function to use for the given variable, in the specified direction.

18.7.9. Function For Emissivity

Scope: Wall Boundary Condition On Surface

`Function For Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of the function to use for the emissivity in the given direction (default direction = time).

18.7.10. Function For Mass_Fraction

Scope: Wall Boundary Condition On Surface

`Function For Mass_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t | x | y | z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.7.11. Function For Mole_Fraction

Scope: Wall Boundary Condition On Surface

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.7.12. Function For Radiation Boundary Temperature

Scope: Wall Boundary Condition On Surface

`Function For Radiation Boundary Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_boundary_temperature variable

18.7.13. Function For Radiation Environment Temperature

Scope: Wall Boundary Condition On Surface

`Function For Radiation Environment Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_environment_temperature variable

18.7.14. Function For Transmissivity

Scope: Wall Boundary Condition On Surface

`Function For Transmissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transmissivity variable

18.7.15. Function For Transparent Band Emissivity

Scope: Wall Boundary Condition On Surface

`Function For Transparent Band Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transparent emissivity band for spectral emissivity on surfaces.

18.7.16. Function For Wall Temperature

Scope: Wall Boundary Condition On Surface

`Function For Wall Temperature {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the wall_temperature variable, in the specified direction.

18.7.17. Inline Function For

Scope: Wall Boundary Condition On Surface

`Inline Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncStr`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}`	-
FuncStr	`"string"`	-

Summary String function of x, y, z, and t to use for the given variable. The function string must be enclosed in quotes if it has spaces or commas. For example: Inline function for x_velocity = "min(1, 0.1*t)"

18.7.18. Integer Data For Subroutine

Scope: Wall Boundary Condition On Surface

`Integer Data For Subroutine` *`SubName`* `{= | are | is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`integer...`	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.7.19. Interface Boundary

Scope: Wall Boundary Condition On Surface

Summary Mark this fluids boundary as an interface between two physical regions. Data will be interpolated across this boundary.

18.7.20. Mass_Fraction

Scope: Wall Boundary Condition On Surface

`Mass_Fraction` *`Species`* `{=|are|is}` *`Mass fraction`*

Parameter	Value	Default
Species	`string`	-
Mass fraction	`"string"`	-

Summary Value for the mass fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.7.21. Mole_Fraction

Scope: Wall Boundary Condition On Surface

`Mole_Fraction` *`Species`* `{=|are|is}` *`Mole fraction`*

Parameter	Value	Default
Species	`string`	-
Mole fraction	`"string"`	-

Summary Value for the mole fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.7.22. Post Process Delta Area

Scope: Wall Boundary Condition On Surface

Summary Request post processing of delta area on this mesh part.

Description Area change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of area change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaArea(n)` where for area change in the x-direction n=1, area change in the y-direction n=2, and area change in the z-direction n=3.

Note that `MaxDeltaArea(1)` will return the maximum area change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.7.23. Post Process Delta Heat Flux

Scope: Wall Boundary Condition On Surface

Summary Request post processing of delta heat flux on this mesh part.

Description Heat flux change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of heat flux change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this variable use `region_name.MaxDeltaHeatFlux(0)`

Note that `MaxDeltaHeatFlux(0)` will return the maximum heat flux change if the request to post process this value was made on more than one boundary in a given region.

18.7.24. Post Process Delta Pressure

Scope: Wall Boundary Condition On Surface

Summary Request post processing of delta pressure on this mesh part.

Description Pressure change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of pressure change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaPressure(n)` where for pressure change in the x-direction n=1, pressure change in the y-direction n=2, and pressure change in the z-direction n=3.

Note that `MaxDeltaPressure(1)` will return the maximum pressure change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.7.25. Post Process Delta Viscous Stress

Scope: Wall Boundary Condition On Surface

Summary Request post processing of delta viscous stress on this mesh part.

Description Viscous stress change can be post-processed on this mesh part for laminar or turbulent flows. This option calculates the absolute value of viscous stress change after each iteration of a time step, writes it to the log and stores the last value for a given step. The value can be accessed by solution control and used as a criterion in the evaluator. To access this value use `region_name.MaxDeltaViscousStress(n)` where for viscous stress change in the x-direction n=1, viscous stress change in the y-direction n=2, and viscous stress change in the z-direction n=3.

Note that `MaxDeltaViscousStress(1)` will return the maximum viscous stress change in x-direction if the request to post process this value was made on more than one boundary in a given region.

18.7.26. Post Process Yplus

Scope: Wall Boundary Condition On Surface

Summary Request post processing of yplus on this mesh part.

Description The normalized distance to the wall (y^+) can be post-processed on any wall (laminar or turbulent). Yplus is calculated in a manner consistent with how it is calculated in the solution procedure. Max/min y^+ values are provided as output at every wall. This option computes and stores the entire y^+ field along the wall. Since this option computes y^+ using the most recent values for all solution variables, the max/min written to the log may differ from the max/min of this field. However, these will become identical at convergence.

Note that this line command does not automatically output this field to a results file. For that, one must use the standard output line command for a nodal field using the string name, `yplus`.

18.7.27. Postprocess

Scope: Wall Boundary Condition On Surface

`Postprocess` *`FluxType`* `Flux Of` *`equation`* `[ As` *`aliases...`* `]`

Parameter	Value	Default
FluxType	`string`	-
equation	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

---

Summary Enable nodal flux post-processing for a given equation. Flux types can be "total", "advective", or "diffusive". Integrated fluxes will also be output to global variables for post-processing or use in other string functions.

By default the variables will be named based on the equation and sideset they are applied to as bc_FluxType_EquationName_flux_SurfaceName. However, if you want to assign a more compact or descriptive name you can provide it with

```
POSTPROCESS TOTAL FLUX OF Continuity AS mdot1
```

When post-processing enthalpy, the average temperature is also output as a global variable. If you provide only one alias the enthalpy flux uses your alias and the average temperature is automatically named. If you provide two aliases the first is used for enthalpy and the second is used for temperature.

```
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux Tavg
```

When post-processing species, the post-processor is run for each species that is solved for. This means that there is no post-processor run on the last species (which is determined by a fractional balance). If you provide aliases for the species post-processor you should provide one for each post-processed species. For example, in a problem with O_2, CO_2, and N_2 (in that order) you could use:

```
POSTPROCESS TOTAL FLUX OF SPECIES AS mdotO2 mdotCO2
```

18.7.28. Project Nodes

Scope: Wall Boundary Condition On Surface

Summary Allow for nodal projection on the particular mesh part.

Description The nodal projection of velocity is defaulted to occur on open boundary conditions only. Wall and inflow BCs are defaulted to not be projected, even on flux inflow (i.e., `USE FLUXES`) and turbulent wall nodes. This option flags the particular mesh part to be included in the nodes to be projected.

18.7.29. Progress_Variable

Scope: Wall Boundary Condition On Surface

`Progress_Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`real`	-

Summary Constant value for the progress variable of selected scalars.

18.7.30. Real Data For Subroutine

Scope: Wall Boundary Condition On Surface

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.7.31. Radiation Boundary Temperature

Scope: Wall Boundary Condition On Surface

`Radiation Boundary Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_boundary_temperature variable.

18.7.32. Radiation Boundary Temperature Field

Scope: Wall Boundary Condition On Surface

`Radiation Boundary Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation boundary temperature.

18.7.33. Radiation Environment Temperature

Scope: Wall Boundary Condition On Surface

`Radiation Environment Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_environment_temperature variable.

18.7.34. Radiation Environment Temperature Field

Scope: Wall Boundary Condition On Surface

`Radiation Environment Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation environment temperature.

18.7.35. Subroutine For

Scope: Wall Boundary Condition On Surface

`Subroutine For` *`PrimitiveVariable`* `{=|are|is}` *`Subroutine`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
Subroutine	`string`	-

Summary Name of the subroutine to use for this variable.

18.7.36. Subroutine For Emissivity

Scope: Wall Boundary Condition On Surface

`Subroutine For Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the emissivity variable.

18.7.37. Subroutine For Mass_Fraction

Scope: Wall Boundary Condition On Surface

`Subroutine For Mass_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mass fraction. ALL species mass fractions must be assigned by this subroutine.

18.7.38. Subroutine For Mole_Fraction

Scope: Wall Boundary Condition On Surface

`Subroutine For Mole_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mole fractions. ALL species mole fractions must be assigned by this subroutine.

18.7.39. Subroutine For Radiation Boundary Temperature

Scope: Wall Boundary Condition On Surface

`Subroutine For Radiation Boundary Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_boundary_temperature variable.

18.7.40. Subroutine For Radiation Environment Temperature

Scope: Wall Boundary Condition On Surface

`Subroutine For Radiation Environment Temperature {=|are|is}`
`subroutineName`

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_environment_temperature variable.

18.7.41. Subroutine For Transmissivity

Scope: Wall Boundary Condition On Surface

`Subroutine For Transmissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transmissivity variable.

18.7.42. Subroutine For Transparent Band Emissivity

Scope: Wall Boundary Condition On Surface

`Subroutine For Transparent Band Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transparent emissivity band for spectral emissivity on surfaces.

18.7.43. Subroutine For Wall Temperature

Scope: Wall Boundary Condition On Surface

`Subroutine For Wall Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the wall_temperature variable.

18.7.44. Transparent Band Emissivity

Scope: Wall Boundary Condition On Surface

`Transparent Band Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value to use for the emissivity in the transparent band for spectral surfaces.

18.7.45. Transmissivity

Scope: Wall Boundary Condition On Surface

`Transmissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for transmissivity variable.

18.7.46. Use Equilibrium Production Model

Scope: Wall Boundary Condition On Surface

Summary Change near-wall turbulence production.

Description This option specifies that the near wall turb_ke production term is given by:

$$\frac{{\tau_w}^2}{\kappa \sqrt[4]{C_\mu} \rho y_p \sqrt{k}}$$

. Modifications are made for the KW and SST model. This model is suggested when using the standard KW and SST model when a Dirichlet condition is not placed on the wall node of turbulence kinetic energy.

18.7.47. Velocity Is Relative To Mesh

Scope: Wall Boundary Condition On Surface

Summary Indicates that the specified velocity is relative to the mesh motion. Applicable only for wall BCs.

18.7.48. Wall Temperature

Scope: Wall Boundary Condition On Surface

`Wall Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Value for wall_temperature variable. Can be a constant or a string function of space (x,y,z), time (t) or global variables.

18.7.49. Law_Of_Wall_Roughness_Parameter

Scope: Wall Boundary Condition On Surface

`Law_Of_Wall_Roughness_Parameter {=|are|is}` *`Law of the Wall Roughness parameter`*

Parameter	Value	Default
Law of the Wall Roughness parameter	`real`	`9.8`

Summary Specify dimensionless roughness factor to be used in the law-of-the-wall formulation.

18.8. WALL MASS INJECT BOUNDARY CONDITION ON SURFACE

Scope: Fuego Region

`Begin Wall Mass Inject Boundary Condition On Surface` *`Surfacename`*

`PrimitiveVariable` `{=|are|is}` *`Value`*

`Emissivity Spectral File Name {=|are|is}` *`Name`*

`External Field For` *`VariableName`* `[{of}` *`Species`* `]{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `With Multiplier {=|are|is}` *`Multiplier`* `]`

`Emissivity {=|are|is}` *`value`*

`Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

`Function For Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

`Function For Injected_Temperature {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

`Function For Mass Flux {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

`Function For Mass_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

`Function For Radiation Boundary Temperature {=|are|is}` *`functionName`*

Function For Radiation Environment Temperature {=|are|is} *functionName*

Function For Transmissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Transparent Band Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Wall Temperature {=|are|is} *functionName* [In The *Direction* Direction]

Injected_Temperature {=|are|is} *temperatureValue*

Inline Function For *PrimitiveVariable* {=|are|is} *FuncStr*

Inline Function For Injected_Temperature {=|are|is} *temperatureFunction*

Inline Function For Mass Flux {=|are|is} *fluxFunction*

Integer Data For Subroutine *SubName* {=|are|is} *Values...*

Interface Boundary

Mass Flux {=|are|is} *fluxValue*

Mass_Fraction *Species* {=|are|is} *Mass fraction*

Mole_Fraction *Species* {=|are|is} *Mole fraction*

Pool *PoolBC* {=|are|is} *The pool values*

Post Process Yplus

Postprocess *FluxType* Flux Of *equation* [As *aliases...*]

Project Nodes

Progress_Variable *ProgressVariableName* {=|are|is} *Value*

Real Data For Subroutine *SubName* {=|are|is} *Values...*

Radiation Boundary Temperature {=|are|is} *value*

Radiation Boundary Temperature Field {=|are|is} *FieldName*

Radiation Environment Temperature {=|are|is} *value*

Radiation Environment Temperature Field {=|are|is} *FieldName*

Subroutine For *PrimitiveVariable* {=|are|is} *Subroutine*

Subroutine For Emissivity {=|are|is} *subroutineName*

Subroutine For Mass_Fraction {=|are|is} *Subroutine*

Subroutine For Mole_Fraction {=|are|is} *Subroutine*

```
    Subroutine For Radiation Boundary Temperature {=|are|is}
    subroutineName

    Subroutine For Radiation Environment Temperature {=|are|is}
    subroutineName

    Subroutine For Transmissivity {=|are|is} subroutineName

    Subroutine For Transparent Band Emissivity {=|are|is}
    subroutineName

    Subroutine For Wall Temperature {=|are|is} subroutineName

    Transparent Band Emissivity {=|are|is} value

    Transmissivity {=|are|is} value

    Use Equilibrium Production Model

    Wall Temperature {=|are|is} value

    Law_Of_Wall_Roughness_Parameter {=|are|is} Law of the Wall
    Roughness parameter

End
```

Summary Defines a wall with mass injection boundary condition on a named surface of the mesh. This boundary condition will use all flux conditions for all fields other than turbulent k.e.

18.8.1.

Scope: Wall Mass Inject Boundary Condition On Surface

```
PrimitiveVariable {=|are|is} Value
```

Parameter	Value	Default
PrimitiveVariable	{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}	-
Value	"string"	-

Summary Value for the specified variable (in consistent units). Value can be a constant or a string function of position (x,y,z), time (t), and any global variable.

18.8.2. Emissivity Spectral File Name

Scope: Wall Mass Inject Boundary Condition On Surface

```
Emissivity Spectral File Name {=|are|is} Name
```

Parameter	Value	Default
Name	string	-

Summary Specify the file name for defining spectral radiation properties on a surface.

18.8.3. External Field For

Scope: Wall Mass Inject Boundary Condition On Surface

```
External Field For VariableName [{of} Species ]{=|are|is} ExtFieldName [
Of Size Value With Multiplier {=|are|is} Multiplier ]
```

Parameter	Value	Default
VariableName	string	-
Species	string	-
ExtFieldName	string	-

Summary Name of the field that is to be transferred in. For species equation, specify variable as "mass fraction of" followed by the species name for transfers associated with separated individual species field names.

18.8.4. Emissivity

Scope: Wall Mass Inject Boundary Condition On Surface

`Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for emissivity variable.

18.8.5. Function For

Scope: Wall Mass Inject Boundary Condition On Surface

`Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
FuncName	`string`	-

Summary Name of function to use for the given variable, in the specified direction.

18.8.6. Function For Emissivity

Scope: Wall Mass Inject Boundary Condition On Surface

`Function For Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of the function to use for the emissivity in the given direction (default direction = time).

18.8.7. Function For Injected_Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Function For Injected_Temperature {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the injected temperature, in the specified direction.

18.8.8. Function For Mass Flux

Scope: Wall Mass Inject Boundary Condition On Surface

`Function For Mass Flux {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for mass flux, in the specified direction.

18.8.9. Function For Mass_Fraction

Scope: Wall Mass Inject Boundary Condition On Surface

`Function For Mass_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.8.10. Function For Mole_Fraction

Scope: Wall Mass Inject Boundary Condition On Surface

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.8.11. Function For Radiation Boundary Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Function For Radiation Boundary Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_boundary_temperature variable

18.8.12. Function For Radiation Environment Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Function For Radiation Environment Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_environment_temperature variable

18.8.13. Function For Transmissivity

Scope: Wall Mass Inject Boundary Condition On Surface

`Function For Transmissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transmissivity variable

18.8.14. Function For Transparent Band Emissivity

Scope: Wall Mass Inject Boundary Condition On Surface

`Function For Transparent Band Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transparent emissivity band for spectral emissivity on surfaces.

18.8.15. Function For Wall Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Function For Wall Temperature {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the wall_temperature variable, in the specified direction.

18.8.16. Injected_Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Injected_Temperature {=|are|is}` *`temperatureValue`*

Parameter	Value	Default
temperatureValue	`"string"`	-

Summary Injected temperature of mass injection wall BC with specified flux. Value can be a consant or a string function of time, space, and global variables.

18.8.17. Inline Function For

Scope: Wall Mass Inject Boundary Condition On Surface

`Inline Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncStr`*

Parameter	Value	Default
PrimitiveVariable	{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}	-
FuncStr	"string"	-

Summary String function of x, y, z, and t to use for the given variable. The function string must be enclosed in quotes if it has spaces or commas. For example: Inline function for x_velocity = "min(1, 0.1*t)"

18.8.18. Inline Function For Injected_Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Inline Function For Injected_Temperature {=|are|is}` *`temperatureFunction`*

Parameter	Value	Default
temperatureFunction	"string"	-

Summary Injected temperature inline function of time and space. The function string must be enclosed in quotes if it has spaces or commas.

18.8.19. Inline Function For Mass Flux

Scope: Wall Mass Inject Boundary Condition On Surface

`Inline Function For Mass Flux {=|are|is}` *`fluxFunction`*

Parameter	Value	Default
fluxFunction	"string"	-

Summary Fixed mass injection rate (mass/area-time) inline function of time and space. The function string must be enclosed in quotes if it has spaces or commas.

18.8.20. Integer Data For Subroutine

Scope: Wall Mass Inject Boundary Condition On Surface

`Integer Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`integer...`	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.8.21. Interface Boundary

Scope: Wall Mass Inject Boundary Condition On Surface

Summary Mark this fluids boundary as an interface between two physical regions. Data will be interpolated across this boundary.

18.8.22. Mass Flux

Scope: Wall Mass Inject Boundary Condition On Surface

`Mass Flux {=|are|is}` *`fluxValue`*

Parameter	Value	Default
fluxValue	`"string"`	-

Summary Mass injection rate (mass/area-time). Value can be a constant or a string function of time, space, and global variables.

18.8.23. Mass_Fraction

Scope: Wall Mass Inject Boundary Condition On Surface

`Mass_Fraction` *`Species`* `{=|are|is}` *`Mass fraction`*

Parameter	Value	Default
Species	`string`	-
Mass fraction	`"string"`	-

Summary Value for the mass fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.8.24. Mole_Fraction

Scope: Wall Mass Inject Boundary Condition On Surface

`Mole_Fraction` *`Species`* `{=|are|is}` *`Mole fraction`*

Parameter	Value	Default
Species	`string`	-
Mole fraction	`"string"`	-

Summary Value for the mole fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.8.25. Pool

Scope: Wall Mass Inject Boundary Condition On Surface

`Pool` *`PoolBC`* `{=|are|is}` *`The pool values`*

Parameter	Value	Default
PoolBC	`{absorption_coefficient|boil_flash_temperature_difference|density|evaporation_temperature|heat_of_evaporation|ignition_time|initial_height|initial_temperature|minimum_mass_injection_rate|percent_sensible|specific_heat}`	-
The pool values	`real`	-

Summary Specify parameters for pool evaporation model.

18.8.26. Post Process Yplus

Scope: Wall Mass Inject Boundary Condition On Surface

Summary Request post processing of yplus on this mesh part.

Description The normalized distance to the wall (y^+) can be post-processed on any wall (laminar or turbulent). Yplus is calculated in a manner consistent with how it is calculated in the solution procedure. Max/min y^+ values are provided as output at every wall. This option computes and stores the entire y^+ field along the wall. Since this option computes y^+ using the most recent values for all solution variables, the max/min written to the log may differ from the max/min of this field. However, these will become identical at convergence.

Note that this line command does not automatically output this field to a results file. For that, one must use the standard output line command for a nodal field using the string name, `yplus`.

18.8.27. Postprocess

Scope: Wall Mass Inject Boundary Condition On Surface

`Postprocess` *`FluxType`* `Flux Of` *`equation`* `[ As` *`aliases...`* `]`

Parameter	Value	Default
FluxType	`string`	-
equation	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

---

Summary

Enable nodal flux post-processing for a given equation. Flux types can be "total", "advective", or "diffusive". Integrated fluxes will also be output to global variables for post-processing or use in other string functions.

By default the variables will be named based on the equation and sideset they are applied to as bc_FluxType_EquationName_flux_SurfaceName. However, if you want to assign a more compact or descriptive name you can provide it with

```
POSTPROCESS TOTAL FLUX OF Continuity AS mdot1
```

When post-processing enthalpy, the average temperature is also output as a global variable. If you provide only one alias the enthalpy flux uses your alias and the average temperature is automatically named. If you provide two aliases the first is used for enthalpy and the second is used for temperature.

```
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux Tavg
```

When post-processing species, the post-processor is run for each species that is solved for. This means that there is no post-processor run on the last species (which is determined by a fractional balance). If you provide aliases for the species post-processor you should provide one for each post-processed species. For example, in a problem with O_2, CO_2, and N_2 (in that order) you could use:

```
POSTPROCESS TOTAL FLUX OF SPECIES AS mdotO2 mdotCO2
```

18.8.28. Project Nodes

Scope: Wall Mass Inject Boundary Condition On Surface

Summary Allow for nodal projection on the particular mesh part.

Description The nodal projection of velocity is defaulted to occur on open boundary conditions only. Wall and inflow BCs are defaulted to not be projected, even on flux inflow (i.e., `USE FLUXES`) and turbulent wall nodes. This option flags the particular mesh part to be included in the nodes to be projected.

18.8.29. Progress_Variable

Scope: Wall Mass Inject Boundary Condition On Surface

`Progress_Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`real`	-

Summary Constant value for the progress variable of selected scalars.

18.8.30. Real Data For Subroutine

Scope: Wall Mass Inject Boundary Condition On Surface

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.8.31. Radiation Boundary Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Radiation Boundary Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_boundary_temperature variable.

18.8.32. Radiation Boundary Temperature Field

Scope: Wall Mass Inject Boundary Condition On Surface

`Radiation Boundary Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation boundary temperature.

18.8.33. Radiation Environment Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Radiation Environment Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_environment_temperature variable.

18.8.34. Radiation Environment Temperature Field

Scope: Wall Mass Inject Boundary Condition On Surface

`Radiation Environment Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation environment temperature.

18.8.35. Subroutine For

Scope: Wall Mass Inject Boundary Condition On Surface

`Subroutine For` *`PrimitiveVariable`* `{=|are|is}` *`Subroutine`*

Parameter	Value	Default
PrimitiveVariable	{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}	-
Subroutine	string	-

Summary Name of the subroutine to use for this variable.

18.8.36. Subroutine For Emissivity

Scope: Wall Mass Inject Boundary Condition On Surface

`Subroutine For Emissivity {=|are|is}` *subroutineName*

Parameter	Value	Default
subroutineName	string	-

Summary Name of the user subroutine to use for the emissivity variable.

18.8.37. Subroutine For Mass_Fraction

Scope: Wall Mass Inject Boundary Condition On Surface

`Subroutine For Mass_Fraction {=|are|is}` *Subroutine*

Parameter	Value	Default
Subroutine	string	-

Summary Name of the subroutine to use for the mass fraction. ALL species mass fractions must be assigned by this subroutine.

18.8.38. Subroutine For Mole_Fraction

Scope: Wall Mass Inject Boundary Condition On Surface

`Subroutine For Mole_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mole fractions. ALL species mole fractions must be assigned by this subroutine.

18.8.39. Subroutine For Radiation Boundary Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Subroutine For Radiation Boundary Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_boundary_temperature variable.

18.8.40. Subroutine For Radiation Environment Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Subroutine For Radiation Environment Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_environment_temperature variable.

18.8.41. Subroutine For Transmissivity

Scope: Wall Mass Inject Boundary Condition On Surface

`Subroutine For Transmissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transmissivity variable.

18.8.42. Subroutine For Transparent Band Emissivity

Scope: Wall Mass Inject Boundary Condition On Surface

`Subroutine For Transparent Band Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transparent emissivity band for spectral emissivity on surfaces.

18.8.43. Subroutine For Wall Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Subroutine For Wall Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the wall_temperature variable.

18.8.44. Transparent Band Emissivity

Scope: Wall Mass Inject Boundary Condition On Surface

`Transparent Band Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value to use for the emissivity in the transparent band for spectral surfaces.

18.8.45. Transmissivity

Scope: Wall Mass Inject Boundary Condition On Surface

`Transmissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for transmissivity variable.

18.8.46. Use Equilibrium Production Model

Scope: Wall Mass Inject Boundary Condition On Surface

Summary Change near-wall turbulence production.

Description This option specifies that the near wall turb_ke production term is given by:

$$\frac{{\tau_w}^2}{\kappa \sqrt[4]{C_\mu} \rho y_p \sqrt{k}}$$

. Modifications are made for the KW and SST model. This model is suggested when using the standard KW and SST model when a Dirichlet condition is not placed on the wall node of turbulence kinetic energy.

18.8.47. Wall Temperature

Scope: Wall Mass Inject Boundary Condition On Surface

`Wall Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Value for wall_temperature variable. Can be a constant or a string function of space (x,y,z), time (t) or global variables.

18.8.48. Law_Of_Wall_Roughness_Parameter

Scope: Wall Mass Inject Boundary Condition On Surface

`Law_Of_Wall_Roughness_Parameter {=|are|is}` *`Law of the Wall Roughness parameter`*

Parameter	Value	Default
Law of the Wall Roughness parameter	`real`	`9.8`

Summary Specify dimensionless roughness factor to be used in the law-of-the-wall formulation.

18.9. POST PROCESS ON SURFACE

Scope: Fuego Region

```
Begin Post Process On Surface Surfacename

    Post Processing File Name {=|are|is} Name

    Reference Temperature {=|are|is} Temperature

End
```

Summary Defines a wall boundary condition on a named surface of the mesh.

18.9.1. Post Processing File Name

Scope: Post Process On Surface

`Post Processing File Name {=|are|is} Name`

Parameter	Value	Default
Name	`string`	-

Summary Specify the base file name for post-processing surface data.

18.9.2. Reference Temperature

Scope: Post Process On Surface

`Reference Temperature {=|are|is} Temperature`

Parameter	Value	Default
Temperature	`real`	-

Summary Constant value for the reference temperature in consistent units.

18.10. COMPOSITE BOUNDARY CONDITION ON SURFACE

Scope: Fuego Region

Begin Composite Boundary Condition On Surface *Surfacename*

PrimitiveVariable {=|are|is} *Value*

Activate Backside Convection

Activate Backside Radiation

Activate Particle Deposition

Composite *CompositeVariable* {=|are|is} *value*

Composite Initial_Mass_Fraction *Species* {=|are|is} *value*

Composite Material {=|are|is} *material_name*

Emissivity Spectral File Name {=|are|is} *Name*

External Field For *VariableName* [{of} *Species*]{=|are|is} *ExtFieldName* [Of Size *Value* With Multiplier {=|are|is} *Multiplier*]

External Field For Composite *CompositeVariable* {=|are|is} *ExtFieldName*

Emissivity {=|are|is} *value*

Function For *PrimitiveVariable* {=|are|is} *FuncName* [In The *Direction* Direction]

Function For Composite *CompositeVariable* {=|are|is} *functionName* [In The *Direction* Direction]

Function For Composite Initial_Mass_Fraction {=|are|is} *funcName* In The *Direction* Direction

Function For Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Mass_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Mole_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Radiation Boundary Temperature {=|are|is} *functionName*

Function For Radiation Environment Temperature {=|are|is} *functionName*

Function For Transmissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Transparent Band Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Wall Temperature {=|are|is} *functionName* [In The *Direction* Direction]

Inline Function For *PrimitiveVariable* {=|are|is} *FuncStr*

Integer Data For Subroutine *SubName* {=|are|is} *Values...*

Interface Boundary

Mass_Fraction *Species* {=|are|is} *Mass fraction*

Mole_Fraction *Species* {=|are|is} *Mole fraction*

Particle Species {=|are|is} *species_name...*

Post Process Yplus

Postprocess *FluxType* Flux Of *equation* [As *aliases...*]

Project Nodes

Progress_Variable *ProgressVariableName* {=|are|is} *Value*

Real Data For Subroutine *SubName* {=|are|is} *Values...*

Radiation Boundary Temperature {=|are|is} *value*

Radiation Boundary Temperature Field {=|are|is} *FieldName*

Radiation Environment Temperature {=|are|is} *value*

Radiation Environment Temperature Field {=|are|is} *FieldName*

Subroutine For *PrimitiveVariable* {=|are|is} *Subroutine*

Subroutine For Composite *CompositeVariable* {=|are|is} *subroutineName*

Subroutine For Emissivity {=|are|is} *subroutineName*

Subroutine For Mass_Fraction {=|are|is} *Subroutine*

Subroutine For Mole_Fraction {=|are|is} *Subroutine*

Subroutine For Radiation Boundary Temperature {=|are|is} *subroutineName*

```
    Subroutine For Radiation Environment Temperature {=|are|is}
    subroutineName

    Subroutine For Transmissivity {=|are|is} subroutineName

    Subroutine For Transparent Band Emissivity {=|are|is}
    subroutineName

    Subroutine For Wall Temperature {=|are|is} subroutineName

    Transparent Band Emissivity {=|are|is} value

    Transmissivity {=|are|is} value

    Use Equilibrium Production Model

    Wall Temperature {=|are|is} value

    Law_Of_Wall_Roughness_Parameter {=|are|is} Law of the Wall
    Roughness parameter


End
```

Summary Defines a thermally-decomposing composite material wall boundary condition that outgasses flammable species when heated sufficiently.

18.10.1.

Scope: Composite Boundary Condition On Surface

```
PrimitiveVariable {=|are|is} Value
```

Parameter	Value	Default
PrimitiveVariable	{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}	-
Value	"string"	-

Summary Value for the specified variable (in consistent units). Value can be a constant or a string function of position (x,y,z), time (t), and any global variable.

18.10.2. Activate Backside Convection

Scope: Composite Boundary Condition On Surface

Summary Turn on model for convective heat transfer on the back side of the composite material. A convection coefficient and reference temperature must also be specified.

18.10.3. Activate Backside Radiation

Scope: Composite Boundary Condition On Surface

Summary Turn on model for radiative heat transfer on the back side of the composite material. A reference temperature must also be specified.

18.10.4. Activate Particle Deposition

Scope: Composite Boundary Condition On Surface

Summary Turn on deposition of material from the accumulation of particles on the composite material. A species name for the particles must also be supplied. Currently only one species is supported.

18.10.5. Composite

Scope: Composite Boundary Condition On Surface

`Composite` *`CompositeVariable`* `{=|are|is}` *`value`*

Parameter	Value	Default
CompositeVariable	`{backside_convection_coefficient | backside_reference_temperature | initial_temperature | max_nonlinear_iterations | max_thickness | min_nonlinear_iterations | nodes | residual_tolerance | subcycle_safety_factor | thickness}`	-
value	`real`	-

Summary Parameters for composite fire model.

18.10.6. Composite Initial_Mass_Fraction

Scope: Composite Boundary Condition On Surface

`Composite Initial_Mass_Fraction` *`Species`* `{=|are|is}` *`value`*

Parameter	Value	Default
Species	`string`	-
value	`"string"`	-

Summary Constant value for the initial mass fraction of selected species in the composite.

18.10.7. Composite Material

Scope: Composite Boundary Condition On Surface

`Composite Material {=|are|is}` *`material_name`*

Parameter	Value	Default
material_name	`string`	-

Summary Name of the Fuego material to use for the composite BC.

18.10.8. Emissivity Spectral File Name

Scope: Composite Boundary Condition On Surface

`Emissivity Spectral File Name {=|are|is}` *`Name`*

Parameter	Value	Default
Name	`string`	-

Summary Specify the file name for defining spectral radiation properties on a surface.

18.10.9. External Field For

Scope: Composite Boundary Condition On Surface

`External Field For` *`VariableName`* `[{of}` *`Species`* `]{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `With Multiplier {=|are|is}` *`Multiplier`* `]`

Parameter	Value	Default
VariableName	`string`	-
Species	`string`	-
ExtFieldName	`string`	-

Summary Name of the field that is to be transferred in. For species equation, specify variable as "mass fraction of" followed by the species name for transfers associated with separated individual species field names.

18.10.10. External Field For Composite

Scope: Composite Boundary Condition On Surface

`External Field For Composite` *`CompositeVariable`* `{=|are|is}` *`ExtFieldName`*

Parameter	Value	Default
CompositeVariable	`{backside_convection_coefficient |backside_reference_temperature |initial_temperature| max_nonlinear_iterations |max_thickness| min_nonlinear_iterations| nodes|residual_tolerance| subcycle_safety_factor|thickness}`	-
ExtFieldName	`string`	-

Summary Name of an external field to use for composite variable.

18.10.11. Emissivity

Scope: Composite Boundary Condition On Surface

`Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for emissivity variable.

18.10.12. Function For

Scope: Composite Boundary Condition On Surface

`Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
FuncName	`string`	-

Summary Name of function to use for the given variable, in the specified direction.

18.10.13. Function For Composite

Scope: Composite Boundary Condition On Surface

`Function For Composite` *`CompositeVariable`* `{=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
CompositeVariable	{backside_convection_coefficient | backside_reference_temperature | initial_temperature | max_nonlinear_iterations | max_thickness | min_nonlinear_iterations | nodes | residual_tolerance | subcycle_safety_factor | thickness}	-
functionName	string	-

Summary Name of function to use for the given composite fire variable, in the specified direction.

18.10.14. Function For Composite Initial_Mass_Fraction

Scope: Composite Boundary Condition On Surface

`Function For Composite Initial_Mass_Fraction {=|are|is}` *`funcName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
funcName	string	-
Direction	{t | x | y | z}	-

Summary Name of the mass fraction function to use for all species in the composite in a given direction. Note that this must be a Multicolumn function

18.10.15. Function For Emissivity

Scope: Composite Boundary Condition On Surface

`Function For Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	string	-

Summary Name of the function to use for the emissivity in the given direction (default direction = time).

18.10.16. Function For Mass_Fraction

Scope: Composite Boundary Condition On Surface

`Function For Mass_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.10.17. Function For Mole_Fraction

Scope: Composite Boundary Condition On Surface

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.10.18. Function For Radiation Boundary Temperature

Scope: Composite Boundary Condition On Surface

`Function For Radiation Boundary Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_boundary_temperature variable

18.10.19. Function For Radiation Environment Temperature

Scope: Composite Boundary Condition On Surface

`Function For Radiation Environment Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_environment_temperature variable

18.10.20. Function For Transmissivity

Scope: Composite Boundary Condition On Surface

`Function For Transmissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transmissivity variable

18.10.21. Function For Transparent Band Emissivity

Scope: Composite Boundary Condition On Surface

`Function For Transparent Band Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transparent emissivity band for spectral emissivity on surfaces.

18.10.22. Function For Wall Temperature

Scope: Composite Boundary Condition On Surface

`Function For Wall Temperature {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the wall_temperature variable, in the specified direction.

18.10.23. Inline Function For

Scope: Composite Boundary Condition On Surface

`Inline Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncStr`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}`	-
FuncStr	`"string"`	-

Summary String function of x, y, z, and t to use for the given variable. The function string must be enclosed in quotes if it has spaces or commas. For example: Inline function for x_velocity = "min(1, 0.1*t)"

18.10.24. Integer Data For Subroutine

Scope: Composite Boundary Condition On Surface

`Integer Data For Subroutine` *`SubName`* `{= | are | is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`integer...`	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.10.25. Interface Boundary

Scope: Composite Boundary Condition On Surface

Summary Mark this fluids boundary as an interface between two physical regions. Data will be interpolated across this boundary.

18.10.26. Mass_Fraction

Scope: Composite Boundary Condition On Surface

`Mass_Fraction` *`Species`* `{=|are|is}` *`Mass fraction`*

Parameter	Value	Default
Species	`string`	-
Mass fraction	`"string"`	-

Summary Value for the mass fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.10.27. Mole_Fraction

Scope: Composite Boundary Condition On Surface

`Mole_Fraction` *`Species`* `{=|are|is}` *`Mole fraction`*

Parameter	Value	Default
Species	`string`	-
Mole fraction	`"string"`	-

Summary Value for the mole fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.10.28. Particle Species

Scope: Composite Boundary Condition On Surface

`Particle Species {=|are|is}` *`species_name...`*

Parameter	Value	Default
species_name	`string...`	-

Summary Names of the particle species that can accumulate on the composite BC (this must be the complete list of species in the particle definition, listed in the same order as defined in the particle definition).

18.10.29. Post Process Yplus

Scope: Composite Boundary Condition On Surface

Summary Request post processing of yplus on this mesh part.

Description The normalized distance to the wall (y^+) can be post-processed on any wall (laminar or turbulent). Yplus is calculated in a manner consistent with how it is calculated in the solution procedure. Max/min y^+ values are provided as output at every wall. This option computes and stores the entire y^+ field along the wall. Since this option computes y^+ using the most recent values for all solution variables, the max/min written to the log may differ from the max/min of this field. However, these will become identical at convergence.

Note that this line command does not automatically output this field to a results file. For that, one must use the standard output line command for a nodal field using the string name, `yplus`.

18.10.30. Postprocess

Scope: Composite Boundary Condition On Surface

`Postprocess` *`FluxType`* `Flux Of` *`equation`* `[ As` *`aliases...`* `]`

Parameter	Value	Default
FluxType	`string`	-
equation	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

---

Summary Enable nodal flux post-processing for a given equation. Flux types can be "total", "advective", or "diffusive". Integrated fluxes will also be output to global variables for post-processing or use in other string functions.

By default the variables will be named based on the equation and sideset they are applied to as bc_FluxType_EquationName_flux_SurfaceName. However, if you want to assign a more compact or descriptive name you can provide it with

```
POSTPROCESS TOTAL FLUX OF Continuity AS mdot1
```

When post-processing enthalpy, the average temperature is also output as a global variable. If you provide only one alias the enthalpy flux uses your alias and the average temperature is automatically named. If you provide two aliases the first is used for enthalpy and the second is used for temperature.

```
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux Tavg
```

When post-processing species, the post-processor is run for each species that is solved for. This means that there is no post-processor run on the last species (which is determined by a fractional balance). If you provide aliases for the species post-processor you should provide one for each post-processed species. For example, in a problem with O_2, CO_2, and N_2 (in that order) you could use:

```
POSTPROCESS TOTAL FLUX OF SPECIES AS mdotO2 mdotCO2
```

18.10.31. Project Nodes

Scope: Composite Boundary Condition On Surface

Summary Allow for nodal projection on the particular mesh part.

Description The nodal projection of velocity is defaulted to occur on open boundary conditions only. Wall and inflow BCs are defaulted to not be projected, even on flux inflow (i.e., `USE FLUXES`) and turbulent wall nodes. This option flags the particular mesh part to be included in the nodes to be projected.

18.10.32. Progress_Variable

Scope: Composite Boundary Condition On Surface

`Progress_Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`real`	-

Summary Constant value for the progress variable of selected scalars.

18.10.33. Real Data For Subroutine

Scope: Composite Boundary Condition On Surface

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.10.34. Radiation Boundary Temperature

Scope: Composite Boundary Condition On Surface

`Radiation Boundary Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_boundary_temperature variable.

18.10.35. Radiation Boundary Temperature Field

Scope: Composite Boundary Condition On Surface

`Radiation Boundary Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation boundary temperature.

18.10.36. Radiation Environment Temperature

Scope: Composite Boundary Condition On Surface

`Radiation Environment Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_environment_temperature variable.

18.10.37. Radiation Environment Temperature Field

Scope: Composite Boundary Condition On Surface

`Radiation Environment Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation environment temperature.

18.10.38. Subroutine For

Scope: Composite Boundary Condition On Surface

`Subroutine For` *`PrimitiveVariable`* `{=|are|is}` *`Subroutine`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle|edc_product|gas_volume_fraction|mixture_fraction|pressure|progress_variable|scalar_variance|second_mixture_fraction|solid_volume_fraction|soot_mass_fraction|soot_nuclei_mass_fraction|temperature|turbulent_dissipation|turbulent_frequency|turbulent_helmholtz_function|turbulent_kinetic_energy|turbulent_v2|volume_of_fluid|x_solid_velocity|x_velocity|y_solid_velocity|y_velocity|z_solid_velocity|z_velocity}`	-
Subroutine	`string`	-

Summary Name of the subroutine to use for this variable.

18.10.39. Subroutine For Composite

Scope: Composite Boundary Condition On Surface

`Subroutine For Composite` *`CompositeVariable`* `{=|are|is}` *`subroutineName`*

Parameter	Value	Default
CompositeVariable	{backside_convection_coefficient | backside_reference_temperature | initial_temperature | max_nonlinear_iterations | max_thickness | min_nonlinear_iterations | nodes | residual_tolerance | subcycle_safety_factor | thickness}	-
subroutineName	string	-

Summary Name of the user subroutine to use for the given composite fire variable.

18.10.40. Subroutine For Emissivity

Scope: Composite Boundary Condition On Surface

```
Subroutine For Emissivity {=|are|is} subroutineName
```

Parameter	Value	Default
subroutineName	string	-

Summary Name of the user subroutine to use for the emissivity variable.

18.10.41. Subroutine For Mass_Fraction

Scope: Composite Boundary Condition On Surface

```
Subroutine For Mass_Fraction {=|are|is} Subroutine
```

Parameter	Value	Default
Subroutine	string	-

Summary Name of the subroutine to use for the mass fraction. ALL species mass fractions must be assigned by this subroutine.

18.10.42. Subroutine For Mole_Fraction

Scope: Composite Boundary Condition On Surface

```
Subroutine For Mole_Fraction {=|are|is} Subroutine
```

Parameter	Value	Default
Subroutine	string	-

Summary Name of the subroutine to use for the mole fractions. ALL species mole fractions must be assigned by this subroutine.

18.10.43. Subroutine For Radiation Boundary Temperature

Scope: Composite Boundary Condition On Surface

`Subroutine For Radiation Boundary Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_boundary_temperature variable.

18.10.44. Subroutine For Radiation Environment Temperature

Scope: Composite Boundary Condition On Surface

`Subroutine For Radiation Environment Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_environment_temperature variable.

18.10.45. Subroutine For Transmissivity

Scope: Composite Boundary Condition On Surface

`Subroutine For Transmissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transmissivity variable.

18.10.46. Subroutine For Transparent Band Emissivity

Scope: Composite Boundary Condition On Surface

`Subroutine For Transparent Band Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transparent emissivity band for spectral emissivity on surfaces.

18.10.47. Subroutine For Wall Temperature

Scope: Composite Boundary Condition On Surface

`Subroutine For Wall Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the wall_temperature variable.

18.10.48. Transparent Band Emissivity

Scope: Composite Boundary Condition On Surface

`Transparent Band Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value to use for the emissivity in the transparent band for spectral surfaces.

18.10.49. Transmissivity

Scope: Composite Boundary Condition On Surface

`Transmissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for transmissivity variable.

18.10.50. Use Equilibrium Production Model

Scope: Composite Boundary Condition On Surface

Summary Change near-wall turbulence production.

Description This option specifies that the near wall turb_ke production term is given by:

$$\frac{{\tau_w}^2}{\kappa \sqrt[4]{C_\mu} \rho y_p \sqrt{k}}$$

. Modifications are made for the KW and SST model. This model is suggested when using the standard KW and SST model when a Dirichlet condition is not placed on the wall node of turbulence kinetic energy.

18.10.51. Wall Temperature

Scope: Composite Boundary Condition On Surface

`Wall Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Value for wall_temperature variable. Can be a constant or a string function of space (x,y,z), time (t) or global variables.

18.10.52. Law_Of_Wall_Roughness_Parameter

Scope: Composite Boundary Condition On Surface

`Law_Of_Wall_Roughness_Parameter {=|are|is}` *`Law of the Wall Roughness parameter`*

Parameter	Value	Default
Law of the Wall Roughness parameter	`real`	`9.8`

Summary Specify dimensionless roughness factor to be used in the law-of-the-wall formulation.

18.11. COMPOSITE INTERFACE BOUNDARY CONDITION ON SURFACE

Scope: Fuego Region

`Begin Composite Interface Boundary Condition On Surface` *`Surfacename`*

`PrimitiveVariable` `{=|are|is}` *`Value`*

`Composite Initial_Mass_Fraction` *`Species`* `{=|are|is}` *`value`*

`Emissivity Spectral File Name {=|are|is}` *`Name`*

`External Field For` *`VariableName`* `[{of}` *`Species`* `]{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `With Multiplier {=|are|is}` *`Multiplier`* `]`

`Emissivity {=|are|is}` *`value`*

`Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

`Function For Composite Initial_Mass_Fraction {=|are|is}` *`funcName`* `In The` *`Direction`* `Direction`

Function For Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Mass_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Mole_Fraction {=|are|is} *FuncName* In The *Direction* Direction

Function For Radiation Boundary Temperature {=|are|is} *functionName*

Function For Radiation Environment Temperature {=|are|is} *functionName*

Function For Transmissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Transparent Band Emissivity {=|are|is} *functionName* [In The *Direction* Direction]

Function For Wall Temperature {=|are|is} *functionName* [In The *Direction* Direction]

Inline Function For *PrimitiveVariable* {=|are|is} *FuncStr*

Integer Data For Subroutine *SubName* {=|are|is} *Values...*

Interface Boundary

Mass_Fraction *Species* {=|are|is} *Mass fraction*

Mole_Fraction *Species* {=|are|is} *Mole fraction*

Post Process Yplus

Postprocess *FluxType* Flux Of *equation* [As *aliases...*]

Project Nodes

Progress_Variable *ProgressVariableName* {=|are|is} *Value*

Real Data For Subroutine *SubName* {=|are|is} *Values...*

Radiation Boundary Temperature {=|are|is} *value*

Radiation Boundary Temperature Field {=|are|is} *FieldName*

Radiation Environment Temperature {=|are|is} *value*

Radiation Environment Temperature Field {=|are|is} *FieldName*

Subroutine For *PrimitiveVariable* {=|are|is} *Subroutine*

Subroutine For Composite Initial_Mass_Fraction {=|are|is} *subName*

```
    Subroutine For Emissivity {=|are|is} subroutineName

    Subroutine For Mass_Fraction {=|are|is} Subroutine

    Subroutine For Mole_Fraction {=|are|is} Subroutine

    Subroutine For Radiation Boundary Temperature {=|are|is}
    subroutineName

    Subroutine For Radiation Environment Temperature {=|are|is}
    subroutineName

    Subroutine For Transmissivity {=|are|is} subroutineName

    Subroutine For Transparent Band Emissivity {=|are|is}
    subroutineName

    Subroutine For Wall Temperature {=|are|is} subroutineName

    Transparent Band Emissivity {=|are|is} value

    Transmissivity {=|are|is} value

    Use Equilibrium Production Model

    Wall Temperature {=|are|is} value

    Law_Of_Wall_Roughness_Parameter {=|are|is} Law of the Wall
    Roughness parameter

End
```

Summary Defines a wall interface boundary condition that outgasses flammable species when heated sufficiently. This boundary condition may be treated as a porous composite material wall where the heat flux, mass transfer and pressure are applied from a coupled region via a transfer. This boundary enforces Dirichlet-Robin coupling.

18.11.1.

Scope: Composite Interface Boundary Condition On Surface

```
PrimitiveVariable {=|are|is} Value
```

Parameter	Value	Default
PrimitiveVariable	{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}	-
Value	"string"	-

Summary Value for the specified variable (in consistent units). Value can be a constant or a string function of position (x,y,z), time (t), and any global variable.

18.11.2. Composite Initial_Mass_Fraction

Scope: Composite Interface Boundary Condition On Surface

`Composite Initial_Mass_Fraction` *`Species`* `{=|are|is}` *`value`*

Parameter	Value	Default
Species	string	-
value	"string"	-

Summary Constant value for the initial mass fraction of selected species in the composite.

18.11.3. Emissivity Spectral File Name

Scope: Composite Interface Boundary Condition On Surface

`Emissivity Spectral File Name {=|are|is}` *`Name`*

Parameter	Value	Default
Name	string	-

Summary Specify the file name for defining spectral radiation properties on a surface.

18.11.4. External Field For

Scope: Composite Interface Boundary Condition On Surface

`External Field For` *`VariableName`* `[{of}` *`Species`* `]{=|are|is}` *`ExtFieldName`* `[ Of Size` *`Value`* `With Multiplier {=|are|is}` *`Multiplier`* `]`

Parameter	Value	Default
VariableName	`string`	-
Species	`string`	-
ExtFieldName	`string`	-

Summary Name of the field that is to be transferred in. For species equation, specify variable as "mass fraction of" followed by the species name for transfers associated with separated individual species field names.

18.11.5. Emissivity

Scope: Composite Interface Boundary Condition On Surface

`Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for emissivity variable.

18.11.6. Function For

Scope: Composite Interface Boundary Condition On Surface

`Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
PrimitiveVariable	{contact_angle|edc_product |gas_volume_fraction| mixture_fraction|pressure| progress_variable|scalar_variance |second_mixture_fraction |solid_volume_fraction |soot_mass_fraction| soot_nuclei_mass_fraction| temperature|turbulent_dissipation |turbulent_frequency| turbulent_helmholtz_function| turbulent_kinetic_energy|turbulent_v2 |volume_of_fluid|x_solid_velocity |x_velocity|y_solid_velocity| y_velocity|z_solid_velocity| z_velocity}	-
FuncName	string	-

Summary Name of function to use for the given variable, in the specified direction.

18.11.7. Function For Composite Initial_Mass_Fraction

Scope: Composite Interface Boundary Condition On Surface

`Function For Composite Initial_Mass_Fraction {=|are|is}` *`funcName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
funcName	string	-
Direction	{t|x|y|z}	-

Summary Name of the mass fraction function to use for all species in the composite in a given direction. Note that this must be a Multicolumn function

18.11.8. Function For Emissivity

Scope: Composite Interface Boundary Condition On Surface

`Function For Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	string	-

Summary Name of the function to use for the emissivity in the given direction (default direction = time).

18.11.9. Function For Mass_Fraction

Scope: Composite Interface Boundary Condition On Surface

`Function For Mass_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.11.10. Function For Mole_Fraction

Scope: Composite Interface Boundary Condition On Surface

`Function For Mole_Fraction {=|are|is}` *`FuncName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
FuncName	`string`	-
Direction	`{t|x|y|z}`	-

Summary Name of the mass fraction function to use for all species in the given direction. Note that this must be a Multicolumn function

18.11.11. Function For Radiation Boundary Temperature

Scope: Composite Interface Boundary Condition On Surface

`Function For Radiation Boundary Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_boundary_temperature variable

18.11.12. Function For Radiation Environment Temperature

Scope: Composite Interface Boundary Condition On Surface

`Function For Radiation Environment Temperature {=|are|is}` *`functionName`*

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the radiation_environment_temperature variable

18.11.13. Function For Transmissivity

Scope: Composite Interface Boundary Condition On Surface

`Function For Transmissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transmissivity variable

18.11.14. Function For Transparent Band Emissivity

Scope: Composite Interface Boundary Condition On Surface

`Function For Transparent Band Emissivity {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the transparent emissivity band for spectral emissivity on surfaces.

18.11.15. Function For Wall Temperature

Scope: Composite Interface Boundary Condition On Surface

`Function For Wall Temperature {=|are|is}` *`functionName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
functionName	`string`	-

Summary Name of function to use for the wall_temperature variable, in the specified direction.

18.11.16. Inline Function For

Scope: Composite Interface Boundary Condition On Surface

`Inline Function For` *`PrimitiveVariable`* `{=|are|is}` *`FuncStr`*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}`	-
FuncStr	`"string"`	-

Summary String function of x, y, z, and t to use for the given variable. The function string must be enclosed in quotes if it has spaces or commas. For example: Inline function for x_velocity = "min(1, 0.1*t)"

18.11.17. Integer Data For Subroutine

Scope: Composite Interface Boundary Condition On Surface

`Integer Data For Subroutine` *`SubName`* `{= | are | is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`integer...`	-

Summary List of integer data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.11.18. Interface Boundary

Scope: Composite Interface Boundary Condition On Surface

Summary Mark this fluids boundary as an interface between two physical regions. Data will be interpolated across this boundary.

18.11.19. Mass_Fraction

Scope: Composite Interface Boundary Condition On Surface

`Mass_Fraction` *`Species`* `{=|are|is}` *`Mass fraction`*

Parameter	Value	Default
Species	`string`	-
Mass fraction	`"string"`	-

Summary Value for the mass fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.11.20. Mole_Fraction

Scope: Composite Interface Boundary Condition On Surface

`Mole_Fraction` *`Species`* `{=|are|is}` *`Mole fraction`*

Parameter	Value	Default
Species	`string`	-
Mole fraction	`"string"`	-

Summary Value for the mole fraction of selected species. Can be a constant value or a string function of space (x,y,z), time (t), and any global variable.

18.11.21. Post Process Yplus

Scope: Composite Interface Boundary Condition On Surface

Summary Request post processing of yplus on this mesh part.

Description The normalized distance to the wall (y^+) can be post-processed on any wall (laminar or turbulent). Yplus is calculated in a manner consistent with how it is calculated in the solution procedure. Max/min y^+ values are provided as output at every wall. This option computes and stores the entire y^+ field along the wall. Since this option computes y^+ using the most recent values for all solution variables, the max/min written to the log may differ from the max/min of this field. However, these will become identical at convergence.

Note that this line command does not automatically output this field to a results file. For that, one must use the standard output line command for a nodal field using the string name, `yplus`.

18.11.22. Postprocess

Scope: Composite Interface Boundary Condition On Surface

`Postprocess` *`FluxType`* `Flux Of` *`equation`* `[ As` *`aliases...`* `]`

Parameter	Value	Default
FluxType	`string`	-
equation	`{conserved_enthalpy | continuity | edc_product | enthalpy | mixture_fraction | nuclei | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot | species | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_momentum | x_solid_momentum | y_momentum | y_solid_momentum | z_momentum | z_solid_momentum}`	-

---

Summary

Enable nodal flux post-processing for a given equation. Flux types can be "total", "advective", or "diffusive". Integrated fluxes will also be output to global variables for post-processing or use in other string functions.

By default the variables will be named based on the equation and sideset they are applied to as bc_FluxType_EquationName_flux_SurfaceName. However, if you want to assign a more compact or descriptive name you can provide it with

```
POSTPROCESS TOTAL FLUX OF Continuity AS mdot1
```

When post-processing enthalpy, the average temperature is also output as a global variable. If you provide only one alias the enthalpy flux uses your alias and the average temperature is automatically named. If you provide two aliases the first is used for enthalpy and the second is used for temperature.

```
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux
POSTPROCESS TOTAL FLUX OF ENTHALPY AS hFlux Tavg
```

When post-processing species, the post-processor is run for each species that is solved for. This means that there is no post-processor run on the last species (which is determined by a fractional balance). If you provide aliases for the species post-processor you should provide one for each post-processed species. For example, in a problem with O_2, CO_2, and N_2 (in that order) you could use:

```
POSTPROCESS TOTAL FLUX OF SPECIES AS mdotO2 mdotCO2
```

18.11.23. Project Nodes

Scope: Composite Interface Boundary Condition On Surface

Summary Allow for nodal projection on the particular mesh part.

Description The nodal projection of velocity is defaulted to occur on open boundary conditions only. Wall and inflow BCs are defaulted to not be projected, even on flux inflow (i.e., `USE FLUXES`) and turbulent wall nodes. This option flags the particular mesh part to be included in the nodes to be projected.

18.11.24. Progress_Variable

Scope: Composite Interface Boundary Condition On Surface

`Progress_Variable` *`ProgressVariableName`* `{=|are|is}` *`Value`*

Parameter	Value	Default
ProgressVariableName	`string`	-
Value	`real`	-

Summary Constant value for the progress variable of selected scalars.

18.11.25. Real Data For Subroutine

Scope: Composite Interface Boundary Condition On Surface

`Real Data For Subroutine` *`SubName`* `{=|are|is}` *`Values...`*

Parameter	Value	Default
SubName	`string`	-
Values	`real...`	-

Summary List of real data values to be passed down in to the user subroutine. These values may be changed by the user subroutine.

18.11.26. Radiation Boundary Temperature

Scope: Composite Interface Boundary Condition On Surface

`Radiation Boundary Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_boundary_temperature variable.

18.11.27. Radiation Boundary Temperature Field

Scope: Composite Interface Boundary Condition On Surface

`Radiation Boundary Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation boundary temperature.

18.11.28. Radiation Environment Temperature

Scope: Composite Interface Boundary Condition On Surface

`Radiation Environment Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for radiation_environment_temperature variable.

18.11.29. Radiation Environment Temperature Field

Scope: Composite Interface Boundary Condition On Surface

`Radiation Environment Temperature Field {=|are|is}` *`FieldName`*

Parameter	Value	Default
FieldName	`string`	-

Summary Name of the field to use for the radiation environment temperature.

18.11.30. Subroutine For

Scope: Composite Interface Boundary Condition On Surface

`Subroutine For` *`PrimitiveVariable`* `{=|are|is}` *`Subroutine`*

Parameter	Value	Default
PrimitiveVariable	{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}	-
Subroutine	string	-

Summary Name of the subroutine to use for this variable.

18.11.31. Subroutine For Composite Initial_Mass_Fraction

Scope: Composite Interface Boundary Condition On Surface

`Subroutine For Composite Initial_Mass_Fraction {=|are|is}` *subName*

Parameter	Value	Default
subName	string	-

Summary Name of the subroutine to use for the mass fraction. ALL species mass fractions must be assigned by this subroutine.

18.11.32. Subroutine For Emissivity

Scope: Composite Interface Boundary Condition On Surface

`Subroutine For Emissivity {=|are|is}` *subroutineName*

Parameter	Value	Default
subroutineName	string	-

Summary Name of the user subroutine to use for the emissivity variable.

18.11.33. Subroutine For Mass_Fraction

Scope: Composite Interface Boundary Condition On Surface

`Subroutine For Mass_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mass fraction. ALL species mass fractions must be assigned by this subroutine.

18.11.34. Subroutine For Mole_Fraction

Scope: Composite Interface Boundary Condition On Surface

`Subroutine For Mole_Fraction {=|are|is}` *`Subroutine`*

Parameter	Value	Default
Subroutine	`string`	-

Summary Name of the subroutine to use for the mole fractions. ALL species mole fractions must be assigned by this subroutine.

18.11.35. Subroutine For Radiation Boundary Temperature

Scope: Composite Interface Boundary Condition On Surface

`Subroutine For Radiation Boundary Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_boundary_temperature variable.

18.11.36. Subroutine For Radiation Environment Temperature

Scope: Composite Interface Boundary Condition On Surface

`Subroutine For Radiation Environment Temperature {=|are|is}`
`subroutineName`

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the radiation_environment_temperature variable.

18.11.37. Subroutine For Transmissivity

Scope: Composite Interface Boundary Condition On Surface

`Subroutine For Transmissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transmissivity variable.

18.11.38. Subroutine For Transparent Band Emissivity

Scope: Composite Interface Boundary Condition On Surface

`Subroutine For Transparent Band Emissivity {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the transparent emissivity band for spectral emissivity on surfaces.

18.11.39. Subroutine For Wall Temperature

Scope: Composite Interface Boundary Condition On Surface

`Subroutine For Wall Temperature {=|are|is}` *`subroutineName`*

Parameter	Value	Default
subroutineName	`string`	-

Summary Name of the user subroutine to use for the wall_temperature variable.

18.11.40. Transparent Band Emissivity

Scope: Composite Interface Boundary Condition On Surface

`Transparent Band Emissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value to use for the emissivity in the transparent band for spectral surfaces.

18.11.41. Transmissivity

Scope: Composite Interface Boundary Condition On Surface

`Transmissivity {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Constant value for transmissivity variable.

18.11.42. Use Equilibrium Production Model

Scope: Composite Interface Boundary Condition On Surface

Summary Change near-wall turbulence production.

Description This option specifies that the near wall turb_ke production term is given by:

$$\frac{{\tau_w}^2}{\kappa \sqrt[4]{C_\mu} \rho y_p \sqrt{k}}$$

. Modifications are made for the KW and SST model. This model is suggested when using the standard KW and SST model when a Dirichlet condition is not placed on the wall node of turbulence kinetic energy.

18.11.43. Wall Temperature

Scope: Composite Interface Boundary Condition On Surface

`Wall Temperature {=|are|is}` *`value`*

Parameter	Value	Default
value	`"string"`	-

Summary Value for wall_temperature variable. Can be a constant or a string function of space (x,y,z), time (t) or global variables.

18.11.44. Law_Of_Wall_Roughness_Parameter

Scope: Composite Interface Boundary Condition On Surface

`Law_Of_Wall_Roughness_Parameter {=|are|is}` *`Law of the Wall Roughness parameter`*

Parameter	Value	Default
Law of the Wall Roughness parameter	`real`	`9.8`

Summary Specify dimensionless roughness factor to be used in the law-of-the-wall formulation.

18.12. NONCONFORMAL BOUNDARY CONDITION ON SURFACE

Scope: Fuego Region

```
Begin Nonconformal Boundary Condition On Surface Surfacename

    PrimitiveVariable {=|are|is} Value

    Enforce Zero Flux For Turbulence Wall Bc

    Normal Treatment {=|are|is} NormalMethod

    Omit Pressure Stabilization Term

    Opposing Sides {=|are|is} SideNames...

    Post Process Yplus

    Pressure Penalty Multiplier {=|are|is} PressurePenaltyMult

    Search Expansion Factor {=|are|is} SearchExpansion

    Search Method {=|are|is} SearchMethod

    Search Tolerance {=|are|is} SearchTol

    Use Equilibrium Production Model

    Law_Of_Wall_Roughness_Parameter {=|are|is} Law of the Wall
    Roughness parameter

End
```

Summary Defines a non-conformal boundary condition on a named surface of the mesh.

18.12.1.

Scope: Nonconformal Boundary Condition On Surface

PrimitiveVariable {=|are|is} *Value*

Parameter	Value	Default
PrimitiveVariable	`{contact_angle | edc_product | gas_volume_fraction | mixture_fraction | pressure | progress_variable | scalar_variance | second_mixture_fraction | solid_volume_fraction | soot_mass_fraction | soot_nuclei_mass_fraction | temperature | turbulent_dissipation | turbulent_frequency | turbulent_helmholtz_function | turbulent_kinetic_energy | turbulent_v2 | volume_of_fluid | x_solid_velocity | x_velocity | y_solid_velocity | y_velocity | z_solid_velocity | z_velocity}`	-
Value	`"string"`	-

Summary Value for the specified variable (in consistent units). Value can be a constant or a string function of position (x,y,z), time (t), and any global variable.

18.12.2. Enforce Zero Flux For Turbulence Wall Bc

Scope: Nonconformal Boundary Condition On Surface

Summary Enforce no operation for turbulence wall bc.

Description All turbulence equations receive a no operation (zero flux) while momentum receives a Dirichlet. Enthalpy equation receives a Dirichlet if a temperature is specified.

This command is only intended to be used for boundaries of a block where turbulent source terms are being skipped. It is deprecated, and this is now determined automatically from the mesh.

18.12.3. Normal Treatment

Scope: Nonconformal Boundary Condition On Surface

`Normal Treatment {= | are | is}` *`NormalMethod`*

Parameter	Value	Default
NormalMethod	`{keep_both | use_average | use_current}`	-

Summary Select which search method to use. Options are USE_CURRENT (default), USE_AVERAGE, and KEEP_BOTH. You can specify this in just one non-conformal BC and it will apply to all. Specifying different treatments in different non-conformal BCs is currently not supported and will generate an error.

In general, USE_CURRENT will provide the cleanest behavior, with the least amount of numerical noise at the interface. For very complex, moving and deforming geometries you may sometimes gain stability with the USE_AVERAGE treatment. KEEP_BOTH is generally not recommended, and is included for diagnostic purposes only.

18.12.4. Omit Pressure Stabilization Term

Scope: Nonconformal Boundary Condition On Surface

Summary Omits the pressure stabilization term at the non-conformal interface. This option can improve stability, particularly for difficult meshes, but may adversely affect convergence rates.

18.12.5. Opposing Sides

Scope: Nonconformal Boundary Condition On Surface

`Opposing Sides {=|are|is}` *`SideNames...`*

Parameter	Value	Default
SideNames	`string...`	-

Summary List of opposing sides for non-conformal BC specification.

18.12.6. Post Process Yplus

Scope: Nonconformal Boundary Condition On Surface

Summary Request post processing of yplus on this mesh part.

Description The normalized distance to the wall (y^+) can be post-processed on any wall (laminar or turbulent). Yplus is calculated in a manner consistent with how it is calculated in the solution procedure. Max/min y^+ values are provided as output at every wall. This option computes and stores the entire y^+ field along the wall. Since this option computes y^+ using the most recent values for all solution variables, the max/min written to the log may differ from the max/min of this field. However, these will become identical at convergence.

Note that this line command does not automatically output this field to a results file. For that, one must use the standard output line command for a nodal field using the string name, `yplus`.

18.12.7. Pressure Penalty Multiplier

Scope: Nonconformal Boundary Condition On Surface

`Pressure Penalty Multiplier {=|are|is}` *`PressurePenaltyMult`*

Parameter	Value	Default
PressurePenaltyMult	`real`	-

Summary Multiplier to increase the penalty factor on discontinuous pressures at the non-conformal boundaries. Default = 1.0.

18.12.8. Search Expansion Factor

Scope: Nonconformal Boundary Condition On Surface

`Search Expansion Factor {=|are|is}` *`SearchExpansion`*

Parameter	Value	Default
SearchExpansion	`real`	-

Summary Expansion factor for the search tolerance in terms of face size. Setting a value larger than 0 will expand the stencil so the linear system has to be re-initialized fewer times.

Warning: This feature is under development and may incur a significant performance penalty at this time.

18.12.9. Search Method

Scope: Nonconformal Boundary Condition On Surface

`Search Method {=|are|is}` *`SearchMethod`*

Parameter	Value	Default
SearchMethod	`{boost_rtree|kdtree}`	-

Summary DEPRECATED! Select which search method to use. Options are KDTREE (default), KDTREE, and KDTREE. Choosing BOOST_RTREE is deprecated and has no effect.

18.12.10. Search Tolerance

Scope: Nonconformal Boundary Condition On Surface

`Search Tolerance {=|are|is}` *`SearchTol`*

Parameter	Value	Default
SearchTol	`real`	-

Summary Search tolerance for the non-conformal side matching search. This is in real distance units and should be set larger than the expected gaps between sidesets due to faceting or geometry mismatches. If omitted, Fuego calculates this automatically as a tenth of the square root of the largest face area on the non-conformal BC.

18.12.11. Use Equilibrium Production Model

Scope: Nonconformal Boundary Condition On Surface

Summary Change near-wall turbulence production.

Description This option specifies that the near wall turb_ke production term is given by:

$$\frac{{\tau_w}^2}{\kappa \sqrt[4]{C_\mu} \rho y_p \sqrt{k}}$$

. Modifications are made for the KW and SST model. This model is suggested when using the standard KW and SST model when a Dirichlet condition is not placed on the wall node of turbulence kinetic energy.

18.12.12. Law_Of_Wall_Roughness_Parameter

Scope: Nonconformal Boundary Condition On Surface

`Law_Of_Wall_Roughness_Parameter {=|are|is}` *`Law of the Wall Roughness parameter`*

Parameter	Value	Default
Law of the Wall Roughness parameter	`real`	`9.8`

Summary Specify dimensionless roughness factor to be used in the law-of-the-wall formulation.

19. STRING FUNCTIONS

Fuego uses string functions in several different places (initial conditions, boundary conditions, source terms, mesh motion). This chapter describes valid functions and syntax for those functions.

19.1. EXPRESSION EVALUATION

String function evaluation is handled using the STK expression evaluator. The specific set of valid variables that can be used in the function depends on where it is being used, but often includes things like time (t) and spatial coordinates (x,y,z). In some cases, global variables are also registered as valid variables.

Variable names are case sensitive in the evaluator, however the calling app may remove this by converting the expression to all upper case or lower case in the code.

In addition to the registered variables, there are a large number of standard functions and operations available in the evaluator. The following section shows a number of different operators with examples and descriptions.

19.1.1. Powers

Powers can be specified either using the `pow` function or the standard `a^b` notation.

```
3^2
pow(3,2)
```

19.1.2. Min and Max Functions

There are min and max functions that take 2 to 4 comma-separated inputs.

```
min(1,2)
min(1,2,3,4)
max(1,2)
max(1,2,3,4)
```

19.1.3. Inline Variables

You can define variables inline in the function string, although doing this with aprepro in your input file is often more straightforward.

```
a=1;b=2;a+b+4
```

19.1.4. Ramps and Pulses

There are a number of pre-defined ramp and step functions.

```
cycloidal_ramp(t,ts,te)
haversine_pulse(t,ts,te)
cosine_ramp(t)
cosine_ramp(t,te)
cosine_ramp(t,ts,te)
sign(x)
unit_step(t,ts,te)
point2d(x,y,r,w)
point3d(x,y,z,r,w)
```

19.1.4.1. Cosine Ramp

The `cosine_ramp` function provides a smooth ramp from 0 to 1. It can be used with 1, 2, or 3 inputs. The form with one input uses a start and end time of 0 and 1. The form with two inputs specifies the end time, but uses a start time of 0. Note that `cos_ramp` is also a valid function, and is equivalent to `cosine_ramp`.

$$\text{cosine_ramp}(t, t_s, t_e) = \begin{cases} 0, & t \leq t_s \\ \frac{1}{2}\left(1 - \cos\left(\frac{\pi(t-t_s)}{t_e-t_s}\right)\right), & t_s < t < t_e \\ 1, & t \geq t_e \end{cases} \tag{19.1}$$

19.1.4.2. Cycloidal Ramp

The `cycloidal_ramp` function is another smooth ramp function from 0 to 1. Unlike the `cosine_ramp` function, it requires all three arguments.

$$\text{cycloidal_ramp}(t, t_s, t_e) = \begin{cases} 0, & t \leq t_s \\ \frac{t-t_s}{t_e-t_s} - \frac{1}{2\pi}\sin\left(\frac{2\pi(t-t_s}{t_e-t_s}\right), & t_s < t < t_e \\ 1, & t \geq t_e \end{cases} \tag{19.2}$$

19.1.4.3. Haversine Pulse

The `haversine_pulse` function is a smooth sinusoidal finite width pulse, defined by

$$\text{haversine_pulse}(t, t_s, t_e) = \begin{cases} 0, & t \leq t_s \\ \sin\left(\frac{\pi(t-t_s)}{t_e-t_s}\right)^2, & t_s < t < t_e \\ 0, & t \geq t_e \end{cases} \tag{19.3}$$

19.1.4.4. Sign

The `sign` function returns the sign of its single argument

$$\text{sign}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \tag{19.4}$$

19.1.4.5. Unit Step

The `unit_step` function is a sharp square step function

$$\text{unit_step}(t, t_s, t_e) = \begin{cases} 0, & t \leq t_s \\ 1, & t_s < t < t_e \\ 0, & t \geq t_e \end{cases} \tag{19.5}$$

19.1.4.6. 2D Point

The `point2d` function provides a 2D point mask for applying a function in a certain spatial window. It is equivalent to a longer call to `cosine_ramp` where

$$\text{point2d}(x, y, r, w) = 1 - \text{cosine_ramp}(\sqrt{x^2 + y^2}, r - w/2, r + w/2) \tag{19.6}$$

19.1.4.7. 3D Point

The `point3d` function provides a 3D point mask for applying a function in a certain spatial window. It is equivalent to a longer call to `cosine_ramp` where

$$\text{point3d}(x, y, z, r, w) = 1 - \text{cosine_ramp}(\sqrt{x^2 + y^2 + z^2}, r - w/2, r + w/2) \tag{19.7}$$

19.1.5. Basic Math

There are standard mathematical functions. Most of these require no explanation, although it is worth noting that `log` is the natural log (equivalent to `ln` not `log10`).

```
exp(x)
ln(x)
log(x)
log10(x)
pow(a,b)
sqrt(x)
erfc(x)
erf(x)
acos(x)
asin(x)
asinh(x)
atan(x)
atan2(y,x)
atanh(x)
cos(x)
cosh(x)
acosh(x)
sin(x)
sinh(x)
tan(x)
tanh(x)
```

19.1.6. Rounding Functions

There are a variety of rounding and numerical manipulation routines available.

```
ceil(x)
floor(x)
abs(x)
mod(x,y)
ipart(x)
fpart(x)
```

You can round up with `ceil` and round down with `floor`. You can separate a number into its integer part (`ipart`) and floating point part (`fpart`). You can compute the modulus with `mod` and get the absolute value with `abs`.

19.1.7. Polar Coordinate and Angle Helpers

When working with polar and rectangular coordinates there are helper functions for converting coordinate systems as well as for converting degrees to radians and back.

```
poltorectx(r,theta)
poltorecty(r,theta)
deg(r)
rad(d)
recttopola(x,y)
recttopolr(x,y)
```

The conversion functions are:

$$\text{poltorectx}(r, \theta) = r \cos(\theta) \tag{19.8}$$

$$\text{poltorecty}(r, \theta) = r \sin(\theta) \tag{19.9}$$

$$\text{recttopola}(x, y) = \text{atan2}(y, x) \tag{19.10}$$

$$\text{recttopolr}(x, y) = \sqrt{x^2 + y^2} \tag{19.11}$$

The angle returned by `recttopola` is adjusted to be between 0 and 2π.

19.1.8. Distributions and Random Sampling

There are several functions available for generating pseudo-random output with different distributions. When using random output in a string function, be careful using it in places where it can affect nonlinear convergence.

For example, setting an initial condition using a `random` distribution is acceptable since it is only evaluated once. However, using that to define a material property (for example, thermal conductivity) or boundary condition would result in that property varying not just in time and space, but also from one nonlinear iteration to the next, which would general keep a Newton solver from converging. The `ts_random` and `ts_normal` functions are designed to help with this problem.

```
weibull_pdf(x,shape,scale)
normal_pdf(x,mu,sigma)
gamma_pdf(x,shape,scale)
log_uniform_pdf(x,xmin,xmax)
exponential_pdf(x,beta)
random()
random(seed)
time()
ts_random(t,x,y,z)
ts_normal(t,x,y,z,mu,sigma,minR,maxR)
```

19.1.8.1. Weibull Distribution

The `weibull_pdf` function returns a Weibull distribution, where λ is the scale parameter and k is the shape parameter.

$$\text{weibull_pdf}(x, k, \lambda) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{19.12}$$

19.1.8.2. Normal Distribution

The `normal_pdf` function returns a normal distribution, where μ is the mean and σ is the standard deviation.

$$\text{normal_pdf}(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{19.13}$$

19.1.8.3. Gamma Distribution

The `gamma_pdf` function returns a gamma distribution, where θ is the scale parameter and k is the shape parameter.

$$\text{gamma_pdf}(x, k, \theta) = \frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-x/\theta} \tag{19.14}$$

19.1.8.4. Log Uniform Distribution

The `log_uniform_pdf` function returns a log-uniform distribution

$$\text{log_uniform_pdf}(x, x_0, x_1) = \frac{1}{(\ln(x_1) - \ln(x_0))x} \tag{19.15}$$

19.1.8.5. Exponential Distribution

The `exponential_pdf` function returns a one-parameter exponential distribution

$$\text{exponential_pdf}(x, \beta) = \frac{1}{\beta} e^{-x/\beta} \tag{19.16}$$

19.1.8.6. Random Values

There are several functions for generating pseudo-random numbers. These can be used as inputs to any of the distributions, or as standalone values to get a uniform distribution.

`random()` calls a platform-indepent fast pseudo-random number algorithm. It is seeded with a static constant, so the first call to it will always produce the same value. The value returned will be between 0 and 1.

`random(seed)` calls the same algorithm as `random()` but sets the seed first then returns a number between 0 and 1. The seed in this case can be any floating point number.

`time()` returns the current time integer. This can used used to provide a random number seed to make the distributions non-deterministic from run to run.

`ts_random(t,x,y,z)` returns a uniformly distributed random number from 0 to 1 that is unique in time and space. This means that repeated evaluations of this function at the same point in time and space will produce the same output. This is useful for applying randomness on boundary conditions, for example, where the value should not change during nonlinear iterations at a given time.

`ts_normal(t,x,y,z,mu,sigma,minR,maxR)` returns a clipped normally distributed random number with a mean of `mu` and standard deviation of `sigma`. The output is clipped to be between `minR` and `maxR` and, like `ts_random`, the output is deterministic in time and space.

19.1.9. Boolean and Ternary Logic

In addition to all these functions, you can make more complicated functions using ternary statements and boolean logic.

You can use ternary operators for simple conditional assignment. The basic structure of a ternary operator is `boolean ? value_if_true : value_if_false` For example, the following if-else code

```
if x > 2.2:
  return 1.2*y
else
  return 0.1*y
```

can be converted to a ternary operator that returns the same thing

```
(x > 2.2 ? 1.2 : 0.1) * y
```

You can also use boolean operations in the string functions. By default, booleans that evaluate to true are treated as 1 and booleans that are false are 0. The prior example can be expressed using boolean statements as

```
(x>2.2)*1.2*y + (x<=2.2)*0.1*y
((x>2.2)*1.1 + 0.1)*y
```

These operations can be combined (multiplication of boolean statements is equivalent to a logical `and`). To replicate the `unit_step` function described earlier you can use either ternary or boolean operators. The following three functions all produce the same result—a step of height 5 between 1 and 2.

```
5*unit_step(x,1,2)
((x>1 && x<2) ? 5 : 0)
5*(x>1)*(x<2)
```

20. PLATFORM INSTRUCTIONS

Detailed instructions for running any SIERRA application code on the available local and high-performance computing platforms may be found on Sandia's Computational Simulation website, which is located at the following URL:

`http://compsim.sandia.gov/compsim/index.html`

Access to this website is currently limited to Sandia users, but future plans include granting access to those who hold SIERRA licenses.

A set of canonical computation fluid dynamics example cases, including both input files and meshes may be found in the documentation area of the afore-mentioned website. The URL for VOTD documentation is shown below:

`http://compsim.sandia.gov/compsim/Docs/Sierra/preRelease/GeneralRelease/index.html`

Specifically, simulations exist for a: 1 meter turbulent helium plume, 5 meter pool fire with PMR (both with and without interpolation transfers), laminar natural convection with PMR, turbulent backward facing step, conjugate heat transfer fire mechanics (a 5 meter pool fire with PMR and simple heat conduction cube), driven cavity, heated turbulent duct, laminar developing and specified pressure drop channel flow, turbulent channel flow, turbulent open jet, and the Waterloo object in fire configuration.

21. FUEGO OUTPUT

This chapter describes the output generated from Fuego. The first section, Section 21.1, discusses the data sent to STDOUT that is typically redirected to a LOG file (with the -l flag).

21.1. LOG FILE

The Fuego LOG file contains several sections. To describe this, a snipped example LOG file is given below.

`Preamble:`	This portion lists Fuego build information along with user location statistics.
`Input File:`	This portion is a verbatim copy of a large portion of the user input file. This section begins with the `Reading myInputFile.i` statement where `myInputFile.i` is the user's input file.
`Instantiating System:`	At this stage, Fuego begins parsing the input file and outputs any errors or warnings. This section begins with the `Instantiating system` statement.
`Initializing Procedure:`	Fuego lists the solution procedure it will use to solve this system. This section begins with the `Initializing procedure` statement.
`Executing Procedure:`	This section, which begins with the `Executing procedure` statement, gives simulation-specific output.
`Timing & Memory Summary:`	This section, which occurs after the `Execution complete` statement, gives subroutine-based timings and the memory consumption for a single processor.

Observing the `Executing Procedure` section of the LOG file in real time will give the user an up-to-date status of the simulation; this is useful in troubleshooting complicated simulations or assessing convergence of various outputs, such as massflow or integrated forces. This LOG file, referenced in examples as `myOutput.out` simulation and can be observed in real time with a few *IX commandline utilities, which are given below.

```
tail -f myOutput.out
watch tail myOutput.out
```

To describe the `Executing Procedure` section, which is useful for understanding the simulation's status in real time, a 33-line sample excerpt is given below.

```
01:-------------------------------------------------------------------------------
02:
03:Step 11   , Time = 2.043e+02, dT = 5.107364e+01, Simulation is 102.1% complete
04:
05:  Fuego      Nonlinear Residuals of the Equation Sets     Linear    Linear
06:Nonlinear      Cell Reynolds number(max) = 78.32          Solver    Solver
07:Iteration              CFL(max) =  100                  Iterations Residual
08:--------- ------------------------------------------- ---------- --------
09:1  of 1  : SubMechanicsManager
10:1  of 1  :   MomentumSubMech
11:               X_Momentum                    = 4.48e-08      3      1.36e-07
12:               Y_Momentum                    = 3.77e-08      3      1.35e-07
13:               Z_Momentum                    = 7.76e-08      6      2.98e-07
14:               Continuity                    = 6.54e-08      13     2.73e-08
15:
16:Integrated Kinetic Energy = 0.000000e+00
17:Time Rate of Mass Change  = 0.000000e+00
18:Open boundary condition: surface_1
19:  Mdot = 6.159922e-03 , Total Area = 3.121445e+00
20:  Ax   = 0.000000e+00 , Ay   = 0.000000e+00 , Az   = -3.121445e+00
21:  FPx  = 0.000000e+00 , FPy  = 0.000000e+00 , FPz  = 0.000000e+00
22:  FTx  = 0.000000e+00 , FTy  = 0.000000e+00 , FTz  = -1.631652e-02
23:Open boundary condition: surface_2
24:  Mdot = -6.159918e-03, Total Area = 3.121445e+00
25:  Ax   = 0.000000e+00 , Ay   = 0.000000e+00 , Az   = 3.121445e+00
26:  FPx  = 0.000000e+00 , FPy  = 0.000000e+00 , FPz  = 0.000000e+00
27:  FTx  = -3.883378e-07, FTy  = 9.083555e-07 , FTz  = 1.632432e-02
28:Wall boundary condition: surface_3
29:  Heat = -0.000000e+00, Total Area = 1.254619e+02
30:  Ax   = 1.637579e-14 , Ay   = -2.220446e-16, Az   = 0.000000e+00
31:  FPx  = -3.485293e-07, FPy  = 4.543057e-07 , FPz  = 3.479612e-18
32:  FVx  = 3.305307e-06 , FVy  = 1.477769e-06 , FVz  = 9.279792e-02
33:  YPmn = 1.000000e+09 , YPmx = 0.000000e+00
```

Within this excerpt, Line 9 refers to the nonlinear iterations. Line 10 refers to subiterations. Lines 11-14 contain the residual information for the specified equations within the INPUT file. Then, lines 18-22 list integrated quantities tailored for the boundary condition type of the surfaces. Common listed quantities within this section are listed below.

`Mdot:` The massflow through the surface [mass/time].

`FP(x,y,z):` The static pressure force along the specified axis [force].

`FV(x,y,z)`: The viscous force along the specified axis [force].

`FT(x,y,z)`: The thrust force, i.e. momentum flux, along the specified axis [force].

22. FREQUENTLY ASKED QUESTIONS

Below is a compilation of frequently asked questions and problematic scenarios raised by the SIERRA Low Mach Module: Fuego users.

1. **Latest Fuego documentation and examples.**
   For the release documentation, it is typically recommended to access compsim.sandia.gov, where you can find the Fuego User's and Theory manuals. One can find Fuego documentation here on SIERRA's Compsim website:
   http://compsim.sandia.gov/compsim/index.html
   Specifically, navigate to "Support & Services", then go to "Documentation", then go to "Version of the Day" then "General Release".
   In addition, we also provide VOTD documentation for the T/F codes, including Fuego. You can find this on the CEE LAN and HPC platforms at the following location:

   ```
   /gpfs1/rpshaw/sierra/docs/tf
   ```

2. **The error message indicates that a problem is occurring with the 'Favre averaging'.**
   Favre averaging is only used for output purposes and not all variables can be Favre averaged.

3. **The calculation is for the radiation field from a turbulent hydrogen jet flame simulation made with Fuego and Nalu or Scefire.**
   Note the solver used, BICGSTAB solver is flaky for large numbers of ordinates, GMRES may be a better option.

4. **During input_output transfers, x_velocity has an invalid state.**
   Special care must be taken when performing transfers between an input_output region since it does not have any real notion of state. To save on memory, any field that is registered is a single state field. Therefore, all transfers of variables to it should be "state_none". Solution fields like x_velocity, y_velocity, z_velocity, density, temperature and gas_volume_fraction, on the Fuego region, have a state. Therefore, the state must be specified when being transferred to the input_output region, e.g. either "state_new" or "state_old".

5. **Error from linear solver using Aztec GMRES and DD-ILUT preconditioning.**
   At times when running on a large number of processors, a processor may hit a SEGV or some other error. Using Trilinos GMRES and the multilevel preconditioning method may be a better alternative.

6. **Is Mesh Refinement Implemented in Fuego?**
   Fuego can perform a uniform refinement of a given mesh. The capability is hooked up with the conduction mechanics. Hence, heat conduction mechanics simulation must be run and output a refined mesh to be used for fluids simulation. Encore may also be used to perform an "Offline" uniform refinement of a mesh.

7. **Restart does not work. No matter which restart initialization file is targeted, the restarted run begins executing at time=0.0 with the initial time step.**
   A "restart time = <time_to_restart>" or "restart = automatic" line command at the domain level must be present. This is needed in addition to the restart blocks in the region. When using "restart = automatic" option, the code will pick up the very last output time in restart file. Fuego will run from the restart time (which it gets from the restart file) to the "terminate time is VALUE". With a restart, the code will ignore the start time. Also, to view the number of time steps in an exodus file, use grope.

8. **Restart problem causing time misalignment.**
   For the code to restart correctly, each region must have a common time in the restart files. Since each restart "scheduling block" should have a matching output schedule, it is advised to specify the output schedule in the top section of the input file (before any "begin procedure" line). The scheduling block is defined via a:

   ```
   BEGIN OUTPUT SCHEDULER restart_output
     AT TIME=0.0, INCREMENT=10.0
   END
   ```

   Then, in the "BEGIN RESTART" blocks for each region, instead of specifying the "at time..." lines, instead add the line command "USE OUTPUT SCHEDULER restart_output". This helps keep the restart output scheduling synced.

9. **A 400 processor run aborts due to element with negative volume, but works on less processors using the same input and mesh files.**
   At times when running simulations using a large number of processors, a hardware or network hiccup may be encountered. It is recommended to re-submit a run to see if problem still persists. It is also advised to run jobs for shorter periods and with fewer processors to reduce the probability of encountering platform problems.

10. **What is the best way of identifying and resolving mesh problem?**
    Both Verde and Cubit are useful tools in identify problems with a mesh and assessing mesh quality.

11. **Participating media radiation (PMR) simulation fails.**
    Typically, the root cause of this problem is the missing Stefan Boltzmann Constant in the input deck.

12. **Fuego error with new mesh.**
    Usually the root cause of this problem is with the mesh itself. Make sure material/block IDs for various volumes are defined, e.g. set via the command "block 1 volume all" in Cubit, and that the specification of sideset numbers for the BCs are consistent with the mesh. It is also recommended

to use the automatic sideset creation feature in Cubit via the command: "sideset start 1 enclosure volume all".

13. **When running particle spray simulation, output file size becomes huge and less manageable.**
Reduce the "particle reservoir size" in the input file. The "reservoir" is the number of inactive particles that each processor creates in memory, waiting for them to enter the flow. It is also a known issue that the Sierra framework can only handle a maximum of around 64K particles in a single element.

14. **During a particle simulation, excessive amount of particle output files are encountered.**
Additional files are created when the "particle reservoir size" is exceeded. To avoid aberrant numbers of output exodus files, increase the size of the reservoir.

15. **Using a multicolumn piecewise linear function, an error message shows that the function could not be found when specifying a time-varying mass fraction of species.**
Species titles set in a multicolumn piecewise linear function are used by the code internally to set the order of the species. The code does not use the species order listed in the .xml file

16. **Is there a way to take a mesh file and dump the coordinates of all the nodes and the variable values in a text format that can be read in a text editor?**
There are a couple of SEACAS utilities, namely exotxt, which puts the x, y, and z in columns, and puts all the data values later in the file.

17. **Excessive velocity and diffusive rates when using a fuel boundary condition.**
To achieve a fine control of a given mass flux, it is important to refine the mesh near the boundary.

18. **Fuego calculation crashes during simulation.**
Some of the quick things that can be implemented to increase the stability of calculation include the following:

    a) Reduce the CFL limit

    b) Increase the max iterations for the Aztec solvers

    c) Adjust the upwind factor

19. **Continuity residuals look very good (small), the momentum residuals are reasonable, however, the enthalpy residuals are very high.**
Fuego doesn't do any sort of normalization for its reported residuals, so values can't be really compared directly for different equations. The larger the state variable values, the larger its residuals will be. For enthalpy in CGS units, absolute values up to about 1e10 are typical. The same holds true for turbulent kinetic energy in CGS units.

20. **Strange residual behavior.**
One can introduce a high degree of damping to the nonlinear iterations. Typically, one would use a relaxation factor less than 1 only if the residuals show an oscillatory behavior and that values less than 0.75 are rare. Nonlinear iteration settings and relaxation factor values that worked well in a recent Fuego application include:

```
UNDER RELAX Turbulent_Dissipation by 0.85
UNDER RELAX Turbulent_Kinetic_Energy by 0.85
UNDER RELAX Momentum by 0.75
UNDER RELAX Pressure by 0.75
UNDER RELAX Species by 1.00
UNDER RELAX Enthalpy by 1.00
```

21. **Fuego subroutine running with 2 processors runs OK, but if the number of processors (np) equals to 4, 8, 16, or 32 the results are incorrect.**
Extra care must be taken when writing subroutines in a multiprocessor environment. For example, consider the sinusoidal formula for u and v velocity coded in terms of the calculated min and max coordinates of the Gauss points in the elements composing each sideset:

```
mms = sin( 2*pi*(((y-ymin)/(ymax-ymin))))
```

where ymin and ymax are computed in the j = 1, npts loop. This implementation is a problem for certain np values if the processor boundaries crosses the sideset(s), which means that only part of the sideset is being handled by an individual processor, consequently, the calculation of the ymin and ymax values are not correct.

22. **Problem generating particles in flow field.**
One of the problems might be with the initial temperature of some of the particles being higher than the critical temperature for the heat of vaporization expression that's defined in the fuelEvaporation block. The liquid phase can't exist above the critical temperature, so it's not surprising that things are going haywire inside that particle integration code. So, try setting those particles to a smaller initial temperature. Injecting particles with a temperature above the vaporization temperature results in particle deletion.

23. **At the end of the simulation, the concatenation of the fluid.e* files does not work.**
Regarding the concatenation failure, one can concatenate the exodus files using the SEACAS utility called EPU; the process is first do a "module load sierra" followed by the command "epu -auto exodusfilename.XX.oo" where XX is the number of exodus files.

24. **Are 'BEGIN MATERIAL' and 'BEGIN PROPERTY SPECIFICATION FOR FUEGO MATERIAL' equivalent?**
They are not. The generic BEGIN MATERIAL line command is something that is plugged in by virtue of code coupling with Calore and Adagio. However, the Fuego code base supports its unique material model specification that is within the Afgo_Material.xml file, the BEGIN PROPERTY SPECIFICATION FOR FUEGO MATERIAL. As such, a number of extra line commands are within this material block.

25. **Can mass flux boundary conditions be used for both inflow and outflow?**
Yes.

26. **Is time-varying mass flow boundary condition supported?**
No.

27. **When are the "conversion factors" commands in the input file required?**
Units in a Fuego simulation are handled differently depending on whether a third-party library (TPL), like Cantera, is utilized or not. If a TPL, especially Cantera, is used, then units are assumed to be in terms of the CGS units system, with the reference pressure in material property specified in units of atmospheres. Other pressures such as in boundary and initial conditions are in CGS unit. If it is desired to utilize some other unit system, then the user must specify proper scaling factors for length, mass, time, and temperature. These commands can be found in Chapter 11. In the case where a TPL is not utilized, then all units, except the reference pressure, must be specified in the user's unit system. As in many other application codes, care must always be taken to ensure that all units are consistent in order to attain correct results.

28. **Asymmetric particle behavior in particle spray model.**
Introduction of particles into a domain via a spray mechanism is a stochastic event given the distribution of velocity and particle size. There will always be a non-smoothness associated with the process. As one decreases the simulation time, the smoothness of the particle injection increases as the target mdot is ideally resolved. Therefore, at fine meshes and time steps, the flow might look more smooth whereas coarse meshes at high time steps might see more variability.

29. **Surface normal orientation in Fuego, Nalu, or Scefire.**
The usual practice is to orient surface normal outwards from the domain, but in terms of Fuego and Nalu or Scefire, it doesn't matter which way the surface normals are pointing in the mesh.

30. **Outputting heartbeat-type variable info from a sideset or group of nodes to a text file.**
One can output to a surface mesh that corresponds to a given sideset, then use grope to perform post-processing. One can also use Encore and define multiple Field Functions at the Domain Level for each desired output variables and use an Output Postprocessor. Evaluation of the postprocessor in the Fuego region would then generate a text file.

31. **Can NETWORK BC within Fuego coupled with NETFLOW be used in conjunction with a symmetry plane?**
Yes, Fuego automatically computes a scaling factor, based on the ratio of the area that the NETFLOW input file claims to the area that Fuego itself computes. The scaling factor gets applied when passing mass fluxes around, so that everything works out. There's no need to include anything in the input file.

32. **How to output 1D composite nodal temperature.**
One can print out the "composite_temperature", this will print values of all nodes through the thickness. The "wall_temperature" prints just the front-side temperature and "backside_temperature" prints the back-side temperature field.

33. **How can blocks and sidesets be named for Fuego.**
Block and sideset names can either be generic (e.g., 'block_1') or they can be given specific names (e.g., 'INFLOW_PLANE') using a mesh creation program such as Cubit.

34. **How is the time step change factor used?**
The time step always tries to go to the CFL limit.

35. **What is the difference between "CURRENT_STEP" and "stepCount"?**
The CURRENT_STEP is a solution control construct whereas stepCount is a region construct.

36. **Gas temperature and wall temperature are not the same.**
The wall temperature and gas phase temperature are not expected to be equal at a flux boundary condition, e.g., the use of a wall function. In general, wall function BCs are weakly applied. The user specified temperature in the wall_bc block is used to determine a law of the wall heat flux in conjunction with the gas temperature. As the resolution at the wall increases, the user specified temperature and predicted temperature will match. It is only when using laminar mechanics or resolved turbulence models, e.g., LRKE, etc., that the wall temperature and gas temperature will match.

37. **How to output local skin friction.**
Add the following lines to the Results Output block to get the variables required for u_tau:

```
Nodal Variables = YPlus as yP
Nodal Variables = eff_wall_yp_bnd as ndtw
Nodal Variables = density as rho
Nodal Variables = viscosity as mu
```

and the following line to the Wall Boundary Condition blocks to post-process y+:

```
Post Process YPlus
```

Using these values, one can calculate the friction velocity in EnSight or ParaView.

38. **What's the difference between time control block and solution control?**
Time Control allows one to step through time in a simplified manner, whether it be a fixed time step or adaptive time stepping. However, if one would like to have more control over advanced time stepping options such as multiple time blocks, transfers, etc., then one would use Solution Control.

39. **How to check input deck syntax.**
Use the following command:

```
fuego -i inputName.i --check-syntax
```

40. **Maximum linear iterations not reached though convergence has not been met.**
The key attribute here is the linear residual scaling parameter. In general, the linear residual tolerance is subject to either R0, RHS or none. Usually, R0 is activated. A R0 scaling means that the initial linear residual that the first iteration provides will be reduced by, e.g., three orders of magnitude. Therefore, if the original linear residual was 1.0e6, a "successful" solve would report 1.0e3. It is only when no scaling is requested that the residual target specified will be "absolute".

41. **Does Fuego support wedge elements?**
Yes.

42. **Which models use the wall model for an LES run and which need to be resolved?**
Models the need the boundary layer resolved include: LAM, V2F, LRKE, LRKW, LRSST, SMAG, DSMAG. Models that use a wall model include: KE, KSGS, RNG, KW, SST.

43. **Can one specify a node/location to start fuel ignition?**
At this time, the only way to initiate ignition at a specific point is by creating a small mesh block.

44. **How is particle mass deposition rate outputted?**
Mass_deposition_rate should provides nodal deposition rate over a control face, not a control volume.

45. **Changing boundary conditions over time during a simulation.**
First, add a function under the Sierra block domain. Then, in the boundary condition definition, use the following line:

```
function for x_velocity = x_inflow in the t direction
```

46. **Solid region in CHT model converges much slower than fluid region.**
To reduce the time to steady-state, use the following command in the Solution Options block: Compute Steady Solution Using Pseudo Transient Method

47. **Is eff_wall_yp_bnd and yplus the same thing?**
No, eff_wall_yp_bnd is the area-weighed normal distance to the wall at a boundary node and yplus is non-dimensional wall distance for a wall-bounded flow.

48. **How do I create a Cantera XML file from a Chemkin input file (.inp) or a Cantera input file (.cti)?**
The Sierra tests Git repository includes several Chemkin input files that can be utilized or modified for use with Fuego. In order to obtain a Cantera XML file, the first step is to convert a Chemkin input file into a Cantera input file. The Cantera executable, ck2cti, is utilized for this process. If a user wants to create a Cantera input file from scratch, Cantera has documentation that details how to create an input file. This documentation is accessible at: http://cantera.github.io/docs/sphinx/html/index.html. The final step is to convert a Cantera input file into a Cantera XML file. The script, ctml_writer.py, is used for this process. The following commands are used to obtain a Cantera XML file:

```
> module load sierra
> ck2cti \
    -i  example.inp \
    -t  therm.dat \
    -tr tran.dat
> python ctml_writer.py example.cti
```

The required python script and DAT files can be found along with the documentation on the Compsim website; they are located in: FAQ's and How To's -> Thermal/Fluid -> How To Guides -> Building Cantera XML File. These files should be copied and placed in the working directory before running the above commands.
NOTE: Users can write their own Cantera XML file or ask a developer for assistance with writing or obtaining a Cantera XML file.

49. **List and description of all variables available for output.**
Fuego Region "fluid_region" has the following fields available for output: (Note, the list will change for each model based on the equations specified and features that are included in the model.)
NODE Fields: Ap

backside_temperature
boundary_mask
control_volume
convection_coefficient
convection_temperature
convection_wall_temperature
deltaP
density
density_derivative
dpdx
dpdy
dpdz
dqdx
dqdy
dqdz
e_tmp
eff_wall_area_bnd
eff_wall_yp_bnd
effective_viscosity
enthalpy
flux_balance
gas_volume_fraction
heat_flux
incident_flux
k_tmp
low_reynolds_source_ke
low_reynolds_source_td
minimum_distance_to_wall
model_coordinates
normal_pressure_force_bnd
normal_viscous_force_bnd
p_tmp
pressure
pressure_force_bnd
specific_heat
surface_fluxb_temperature
tangent_pressure_force_bnd
tangent_viscous_force_bnd
temperature
temperature_bc_node
thermal_conductivity
total_force_bnd
turbulent_dissipation
turbulent_dissipation_bc_node
turbulent_kinetic_energy

turbulent_kinetic_energy_bc_node
turbulent_production_ke
turbulent_production_rodi
turbulent_viscosity
u_tmp
v_tmp
viscosity
viscous_force_bnd
w_tmp
wall_area
wall_temperature
wall_temperature_transfer
x_velocity
x_velocity_bc_node
y_velocity
y_velocity_bc_node
yplus
z_velocity
z_velocity_bc_node
FACE Fields:
convection_coefficient_bip
convection_temperature_bip
eff_wall_yp_bnd
heat_flux_bip
mass_flux_bip
pressure_bip
viscous_force_bip
wall_area_bip
wall_yp_bip
ELEMENT Fields:
ap_ip
mass_flux
subcontrol_volume

BIBLIOGRAPHY

[1] Tieszen, S. R., A. R. Lopez, C. D. Moen, T. Y. Chu, V. F. Nicolette, W. Gill, S. P. Burns, and W. C. Moffatt. "SIERRA/Fuego and SIERRA/Syrinx Verification and Validation Plan, Version 2.0". internal report, Sandia National Laboratories, 2001. 17

[2] Bird, R. B., W. E. Stewart and E. N. Lightfoot. Transport Phenomena. John Wiley and Sons, 1960. 22

[3] Burns, S. P. "Turbulence Radiation Interaction Modeling in Combustion Simulations". Technical report, Sandia National Laboratories, Albuquerque, NM, 1999. 28, 38

[4] Modest, M. F. Radiation Heat Transfer. McGraw Hill Book Company, New York, 1993. 28, 29

[5] Siegel R. and J. R. Howell. Thermal Radiation Heat Transfer, 3rd ed. Hemisphere Publishing, Washington, D.C., 1992. 28

[6] Tennekes, H. and J. L. Lumley. A First Course in Turbulence. MIT Press, Cambridge, 1972. 31

[7] Libby, P. A. and F. A. Williams. Turbulent Reacting Flows, Fundamental Aspects, Topics in Applied Physics, V. 44. Springer-Verlag., 1980. 31, 36

[8] Kuo, K. K. Principles of Combustion. John Wiley and Sons, 1986. 31, 34

[9] Wilcox, D. C. Turbulence Modeling for CFD. DCW Industries, 2nd edition, 1998. 33

[10] Moin, P., K. Squires, W. Cabot, and S. Lee. A dynamic subgrid-scale model for compressible turbulence and scalar transport. Phys. Fluids A, 3(11):2746–2757, 1991. 35, 44

[11] Erlebacher, G., M. Y. Hussaini, C. G. Speziale, and T. A. Zang. Toward the large-eddy simulation of compressible turbulent flows. J. Fluid Mech., 238:155–185, 1992. 35

[12] Gran, I. M. C. Melaaen, and B. F. Magnussen. "Numerical Simulation of Local Extinction Effects in Turbulent Combustor Flows of Methane and Air". In 25th Symposium on Combustion, pages 1283–1291. The Combustion Institute, 1994. 36, 37, 38

[13] Tieszen, S. R., S. P. Domino, A. R. Black. Validation of a simple turbulence model suitable for closure of temporally-filtered Navier-Stokes equations using a helium plume. Technical Report SAND Report 2005–3210, Sandia National Laboratories, Albuquerque, NM, June 2005. 39

[14] Launder, G. E. and B. I. Sharma. Application of the energy dissipation model of turbulence to the calculation of flow near a spining disc. Letters in Heat and Mass Transfer, **1**(2):131–138, 1974. 39

[15] Papageorgakis, G. C. and D. N. Assanis. Comparison of linear and nonlinear RNG-based k-ϵ models for incompressible turbulent flows. Numerical Heat Transfer, Part B, 35(1):1–22, 1999. 40

[16] Durbin, P. A. "Near-Wall Turbulence Closure Modeling Without Damping Functions". Theoretical and Computational Fluid Dynamics, **3**:1–13, 1991. 40

[17] Smagorinsky, J. General circulation experiments with the primitive equations. i. the basic experiment. Monthly Weather Review, 91:99–164, 1963. 43

[18] Rogallo, R. S. and P. Moin. Numerical simulation of turbulent flows. Annual Review of Fluid Mechanics, 16:99–137, 1984. 43

[19] Germano, M., U. Piomelli, P. Moin, and W. H. Cabot. A dynamic subgrid-scale eddy viscosity model. Physics of Fluids A, 3(7):1760–1765, July 1991. 44, 46

[20] Germano, M. Turbulence: The filtering approach. Journal of Fluid Mechanics, 238:325–336, 1992. 44

[21] Ghosal, S., T. S. Lund, P. Moin, and K. Akselvoll. A dynamic localization model for large-eddy simulation of turbulent flow. Journal of Fluid Mechanics, 286:229–255, 1995. 45

[22] Lilly, D. K. A proposed modification of the germano subgrid-scale closure model. Physics of Fluids A, 4(3):633–635, March 1992. 45

[23] Nicolette, V. F. and S. R. Tieszen. " Effect of Turbulent Kinetic Energy Source Terms on Pool Fire Simulations with the $k - \epsilon$ Model". Technical report, Internal Memorandum to Distribution, Official Use Only, Sandia National Laboratories, 2000. 47

[24] Rodi W. Turbulence Models and their Applications in Hydrolics - A State of the Art Review. Publication of teh International Association for Hydrolic Research, Delf, Netherlands, 1984. 47

[25] Jones, W. P. and B. E. Launder. "The Prediction of Laminarization with a Two-Equation Model of Turbulence". International Journal of Heat and Mass Transfer, **15**:301–314, 1972. 51, 52

[26] White, F. M. Viscous Fluid Flow. McGraw-Hill, Inc., 2nd ed., 1991. 52

[27] Sondak, D. L. and R. H. Pletcher. " Application of wall functions to generalized nonorthogonal curvilinear coordinate systems ". AIAA Journal, **33**(1):33–41, January 1995. 54

[28] Elkaim, D., M. Reggio, and R. Camarero. "Control Volume Finite-Element Solution of a Confined Turbulent Diffusion Flame". Numerical Heat Transfer, Part A, **23**(3):259–279, 1993. 54

[29] Launder, B. E. and D. B. Spalding. "The Numerical Computation of Turbulent Flows". Computer Methods in Applied Mechanics and Engineering, **3**:269–289, 1974. 55

[30] Versteeg, H. K. and W. Malalasekera. An Introduction to Computational Fluid Dynamincs. Longman Group LTD, 1995. 57

[31] Jayatilleke, C. L. V. "The Influence of Prandtl Number and Surface Roughness on the Resistance of Laminar Sub-Layer to Momentum and Heat Transfer". Progress in Heat and Mass Transfer, **1**, 1969. 58

[32] Magnussen, B. F., G. H. Hjertager, J. G. Olsen, and D. Bhaduri. "Effect of Turbulent Structure and Local Concentrations on Soot Formation and Combustion in C2H2 Diffusion Flames". In Seventeenth Symposium (International) on Combustion, pages 1383–1393. The Combustion Institute, Pittsburgh, 1979. 59, 98

[33] Magnussen, B. F. "On the Structure of Turbulence and a Generalised Eddy Dissipation Concept for Chemical Reactions in Turbulent Flow". 9th AIAA Sc. Meeting, St. Louis, 1981. 59

[34] Byggstyøl, S. and B. F. Magnussen. "A Model for Extinction in Turbulent Flows". In et al. Bradbury, editor, 4^{th} Symposium on Turbulent Shear Flow, pages 381–395. Springer-Verlag, Berlin, 1983. 59

[35] Magnussen, B. F. "Heat Transfer in Gas Turbine Combustors – A Discussion of Combustion, Heat and Mass Transfer in Gas Turbine Combustors". In Conference Proceedings no. 390, Advisory Group for Aerospace Research and Development (AGARD), 1985. 59

[36] Lilleheie, N. I., I. Ertesvåg, T. Bjorge, S. Byggstyøl, and B. F. Magnussen. "Modeling and Chemical Reactions, Review of Turbulence and Combustion Models". Technical report, The Foundation for Scientific and Industrial Research, Norwegian Institute of Technology, SINTEF Report STF15 A89024, July 1989. 59

[37] Gran, I. and B. F. Magnussen. "A Numerical Study of a Bluff-body Stabilized Diffusion Flame. Part 2: Influence of Combustion Modeling and Finite Rate Chemistry". Combustion Science and Technology, **119**:191–217, 1996. 59

[38] N. Peters. Laminar diffusion flamelet models in non-premixed turbulent combustion. Progress in Energy and Combustion Science, 10:319–339, 1984. 60, 70

[39] T. Poinsot and D. Veynante. Theoretical and Numerical Combustion. R. T. Edwards, Inc., Philadelphia, PA, 2005. 70

[40] C. D. Pierce and P. Moin. A dynamic model for subgrid-scale variance and dissipation rate of a conserved scalar. Physics of Fluids, 10(12), 3041–3044. 71, 73

[41] R. W. Bilger. The structure of turbulent nonpremixed flames. In Proceedings of the 22nd Symposium (International) on Combustion, pages 475–488. The Combustion Institute, Pittsburg, PA, 1988. 71

[42] G. Erlebacher, M. Y. Hussaini, C. G. Speziale, and T. A. Zang. Toward the large eddy simulation of compressible turbulent flows. ICASE Report 87-20, NASA Langley Research Center, Hampton, VA, 1987. Also available as NASA CR 178273. 72

[43] S. Ghosal, T.S. Lund, P. Moin, and K. Akselvoll. A dynamic localization model for large-eddy simulation of turbulent flow. Journal of Fluid Mechanics, 286:229–255, 1995. 74

[44] Parente, A. and Malik, M. R. and Contino, F. and Cuoci,A. and Dally, B. B. "Extension of the Eddy Dissipation Concept for turbulence/chemistry interactions to MILD combustion". Fuel, **163**:98–111, 2016. 79

[45] Mencinger, J. and Zun, I. Journal of Computational Physics, **230**:644–663, 2011. 81

[46] Martinez, J. and Chesneau, X. and Zeghmati, B. Computational Mechanics, **37**:182–193, 2006. 81

[47] Sun, M. Advances in Pure Mathematics, **3**:219–225, 2013. 81

[48] Huang, M. and Wu, L. and Chen, B. Numerical Heat Transfer, Part B: Fundamentals, **61**:412–437, 2012. 81

[49] Lee, H. and Rhee, S. Journal of Mechanical Science and Technology, **29**:1849–1857, 2015. 81

[50] Raeini, A. and Blunt, M. and Bijeljic, B. Journal of Computational Physics, **231**:5653–5668, 2012. 81

[51] Jofre, L. and Lehmkuhl, O. and Castro, J. and Olivia, A. In Lisbon Portugal, editor, European Conference on Computational Fluid Dynamics, 2010. 81

[52] Hardt, S. and Wondra, F. Journal of Computational Physics, **227**:5781–5895, 2008. 82

[53] Francois, M. M. and Cummins, S. J. and Dendy, E. D. and Kothe, D. B. and Sicilian, J. M. and Williams, M. W. Journal of Computational Physics, **213**:141–173, 2006. 84, 85

[54] Lin, S. and Yan, J. and Kats, D. and Wagner, G. Journal of Computational Physics, **380**:119–142, 2019. 85

[55] Zukoski, E. E. "Properties of Fire Plumes". In G. Cox, editor, Combustion Fundamentals of Fire. Academic Press, New York, 1995. 98

[56] Magnussen, B. F. and G. H. Hjertager. "On Mathematical Modeling of Turbulent Combustion with Special Emphasis on Soot Formation and Combustion". In Sixteenth Symposium (International) on Combustion, pages 719–729. The Combustion Institute, Pittsburgh, 1977. 98

[57] Tesner, P. A., T. D. Snegiriova, and V. G. Knorre. "Kinetics of dispersed carbon formation". Combustion and Flame, **17**:253–260, 1971. 98

[58] Haynes, B. S. and H. G. Wagner. "Soot Formation". Progress in Energy and Combustion Science, **7**:229–273, 1981. 98

[59] Yoon, S. S. and DesJardin, P. E. "Modeling Spray Impingement using Linear Stability Theories for Shattering Droplets". International Journal for Numerical Methods in Fluids, **50**:469–489, 2006. 103

[60] Brown, A. L., and Jepsen, R. A. "An Improved Drop Impact Model for Lagrangian/Eularian Coupled Codes". In Proceedings of the 2009 ASME IMECE Conference, pages IMECE2009–11675, 2009. 103

[61] Mundo, C., Sommerfeld, M., and Tropea, C. "Droplet-Wall Collisions: Experimental Studies of the Deposition and Breakup Process". International Journal of Multiphase Flows, **21**:151–173, 1995. 103

[62] Zienkiewicz, O. C. and R. L. Taylor. The Finite Element Method, 4th ed., Vol. 1. McGraw-Hill, 1989. 106

[63] Zienkiewicz, O. C. and R. L. Taylor. The Finite Element Method, 4th ed., Vol. 2. McGraw-Hill, 1991. 106

[64] Tezduyar, T. E. "Stabilized Finite Element Formulations for Incompressible Flow Computations". In , editor, Advances in Applied Mechanics, volume **28**, pages 1–44. Academic Press, Inc., 1992. 106

[65] Gresho, P. M, S. T. Chan, R. L. Lee, and C. D. Upson. "A Modified Finite Element Method for Solving the Time-Dependent, Incompressible Navier-Stokes Equations. Part I: Theory". International Journal for Numerical Methods in Fluids, **4**:557–598, 1984. 106

[66] Patankar, S. V. "Recent Developments in Computational Heat Transfer". Journal of Heat Transfer, **110**(4B):1037–1045, 1988. 106

[67] Shyy, W. "Elements of Pressure-Based Computational Algorithms for Comples Fluid Flow and Heat Transfer". In Hartnett, J. P. and T. F. Irvine, editor, Advances in Heat Transfer, volume **24**, pages 191–275. Academic Press, Inc., 1994. 106

[68] Ferziger, J. H. and M. Perić. Computational Methods for Fluid Dynamics. Springer-Verlag, 1996. 106

[69] Rhie, C. M. and W. L. Chow. "Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation". AIAA Journal, **21**(11):1525–1532, November 1983. 107

[70] Schneider, G. E. and M. J. Raw. "Control Volume Finite-Element Method for Heat Transfer and Fluid Flow Using Colocated Variables–1. Computational Procedure". Numerical Heat Transfer, **11**(4):363–390, 1987. 108

[71] Schneider, G. E. and M. J. Raw. "Control Volume Finite-Element Method for Heat Transfer and Fluid Flow Using Colocated Variables–2. Application and Validation". Numerical Heat Transfer, **11**(4):391–400, 1987. 108

[72] Hirsch, C. Numerical Computation of Internal and External Flows, Volume 2. John Wiley & Sons, 1990. 108

[73] Edwards, H. C. "SIERRA Framework Core Services: Theory and Design". version 0.04, unpublished, 2001. 129

[74] S. Gordan and B. J. McBride. Computer program for calculation of complex chemical equilibrium compositions and applications I. analysis. Technical Report Ref. Publication 1311, NASA, October 1994. 382

[75] B. J. McBride and S. Gordan. Computer program for calculation of complex chemical equilibrium compositions and applications II. users manual and program description. Technical Report Ref. Publication 1311, NASA, June 1996. 382

INDEX

Abort If Field Not Defined On Copy Transfer Send Or Receive Object, 233, 234
Abort If Search Object Outside Of Tolerance, 233, 235
Abscissa, 148, 149
Abscissa Offset, 148, 149
Abscissa Scale, 148, 150
Absorption Coefficient Model, 387, 388
Acceleration, 407, 408
Acoustic Transfer Output, 355, 400, 569
Activate Absorption Model, 380, 381
Activate Acoustic Compressibility Algorithm, 353, 356, 567, 571
Activate Backside Convection, 760, 763
Activate Backside Radiation, 760, 763
Activate Bdf2, 390, 391
Activate Bdf2 For Mesh Velocity, 405, 406
Activate Co2 Dissociation Model, 380, 381
Activate Deris Flaming Buoyancy Source Term, 392, 393
Activate Deris Nonflaming Buoyancy Source Term, 392, 393
Activate Equation, 353, 357, 567, 571
Activate Full Surface Cvfem Gradient Operator For Muscl Scheme, 353, 357, 567, 572
Activate Hydrogen Dissociation Model, 380, 382
Activate Lighthill Tensor Postprocessing, 353, 358, 567, 572
Activate Particle Deposition, 760, 763
Activate Rodi Buoyancy Source Term, 392, 394
Activate Rodi Density Buoyancy Source Term, 392, 394
Activate Separate Co Irreversible Oxidation Pathway, 380, 382
Activate Species Enthalpy Calculations, 353, 358, 567, 572
Activate Surface Flux Balance Temperature Algorithm, 723, 726
Activate Viscous Dissipation Source Term, 353, 358, 567, 573
Active For Procedure, 162, 164
Adapt, 259, 260, 264, 265, 267, 268, 272, 273, 276, 280
Adaptivity, 259, 264, 268, 272
Add Particle Interface, 465, 466
Add Particle Material, 465, 466
Additional Steps, 175, 176, 308, 309, 318, 319, 326, 327, 336, 337, 438, 440, 448, 449, 511, 513, 521, 523, 536, 537, 544, 545, 554, 555, 597, 599
Additional Times, 175, 176, 308, 310, 318, 319, 326, 327, 336, 338, 438, 440, 448, 449, 511, 513, 521, 523, 536, 537, 544, 545, 554, 556, 597, 599
Adhesion Model, 508, 509
Advance, 251, 264, 265, 267, 269, 272, 273, 276, 277, 280, 281
Alias, 158
All Fields, 233, 235
Allow Inverted Elements, 284, 288
Aluminum_Oxide Emittance Model, 387, 388
Append, 308, 310, 438, 440
Apply Pressure Penalty, 689, 692
Apub Transfer, 227, 230
Arc_Angle, 495, 497
Arc_Ref, 495, 497
At Discontinuity Evaluate To, 148, 150
At Step, 175, 177, 308, 310, 318, 319, 326, 327, 336, 338, 400, 438, 440, 448, 449, 511, 513, 521, 523, 536, 537, 544, 545, 554, 556, 597, 599

At Time, 175, 177, 308, 311, 318, 320, 326, 328, 336, 338, 400, 401, 438, 441, 448, 450, 512, 513, 521, 524, 536, 538, 544, 546, 554, 556, 597, 599
At Wall Time, 326, 328, 512, 514, 544, 546
Auxiliary Variable, 194, 195
Average Field, 462, 463
Average Region, 227, 535
Averaging, 286, 303
Averaging Type, 462, 463
Aztec Equation Solver, 132, 139

Backside, 723, 726
Bc Enforcement, 139, 140, 200, 201, 211, 212
Beam Radiation Boundary Specification, 355, 569
Bending Hourglass, 162, 164
Bicgstab Solver, 212, 214
Blocks, 407, 409
Breakup Model Type, 463, 464
Bulk Chemistry Description, 179, 181
Buoyancy Model, 378
Buoyancy Model Specification, 355, 378, 569
Buoyancy Reference, 378, 379
Buoyancy Reference Mass_Fraction, 378, 379
Buoyancy Reference Mole_Fraction, 378, 379
Buoyancy Source Term, 378, 380
Buoyant Object Mass, 407, 409
Buoyant Object Pitch Moment, 407, 409
Buoyant Object Roll Moment, 407, 410
Buoyant Object Surfaces, 407, 410
Buoyant Object Yaw Moment, 408, 410
Butterworth A Coefficients, 303, 304
Butterworth B Coefficients, 303, 304
Butterworth Filtered Field, 303, 304
Butterworth Time Step, 303, 305

Calculate Mesh Quality Metrics, 284, 288
Cantera Xml File, 179, 182
Catalyst, 337, 522, 555, 598
Center, 495, 497
Cfl Limit, 246, 247, 256
Cg Solver, 212, 218
Characteristic Length Scale For Absorption Coefficient Determination, 380, 382, 387, 389
Characteristic_Length, 667, 671, 689, 692
Chemistry Description, 180
Chemkin Linking File, 179, 182
Chemkin Transport File, 179, 182
Column Titles, 148, 150
Component Separator Character, 158, 159, 326, 328, 336, 339, 512, 514, 521, 524, 544, 546, 554, 557, 597, 600
Composite, 760, 764
Composite Boundary Condition On Surface, 286, 760
Composite Initial_Mass_Fraction, 760, 764, 779, 782
Composite Interface Boundary Condition On Surface, 286, 779
Composite Material, 760, 764
Compute Continuity Source, 418, 421
Compute Difference, 284, 288
Compute Energy Source, 418, 421
Compute Gas Phase Volume Fraction Using, 386
Compute Indicator On, 259, 260, 264, 265, 268, 269, 272, 273, 276, 277, 280, 281
Compute Mixture_Fraction Source, 418, 421
Compute Momentum Source, 418, 422
Compute Norm, 284, 289
Compute Rte Particle Source, 418, 422
Compute Soot_Mass_Fraction Source, 418, 422
Compute Source Lhs Contribution, 418, 423
Compute Species Source, 418, 423
Compute Steady Solution, 350, 351
Compute Steady Solution Using Pseudo Transient Method, 353, 358, 567, 573
Convective Coefficient Node Variable, 230
Converged When, 252, 253
Convergence Tolerance, 214–216, 218–220
Coordinate System, 158, 159, 353, 359, 567, 573
Copy, 233, 235
Copy Nodal File Variable, 625, 626
Create, 158, 159
Create Edge Field, 284, 289
Create Element Field, 284, 289, 593, 594
Create Face Field, 284, 290
Create Initial Particle Distribution, 419, 428
Create Nodal Field, 284, 290, 593, 594

Create Particles From File, 419, 430
Create Quadrature Field, 284, 291
Creation_Time, 430, 431
Critical Damkohler Number For Suppression, 380, 382
Crown Info, 476, 479
Cycle Count, 326, 329, 512, 514, 544, 547
Cylinder, 349

Data File, 148, 150
Database Name, 158, 160, 318, 320, 326, 329, 336, 339, 448, 450, 512, 515, 521, 524, 536, 538, 544, 547, 554, 557, 597, 600
Database Type, 158, 160, 318, 320, 326, 329, 336, 339, 448, 450, 512, 515, 521, 525, 536, 538, 544, 547, 554, 557, 597, 600
Datum Pressure, 179, 183
Dd-Ilu Preconditioner, 214, 215, 217, 218, 221
Dd-Ilut Preconditioner, 214, 215, 217, 219, 223
Debug, 148, 151
Debug Dump, 326, 330, 512, 515, 544, 548
Debug Level, 227, 228
Debug Output Level, 139, 141, 200, 202
Debug Output Path, 139, 141, 200, 202
Decomposition Method, 158, 160, 326, 330, 512, 516, 544, 548
Define Axis, 132, 134
Define Direction, 132, 134
Define Point, 132, 134
Definition For Function, 133, 148
Density, 199, 200, 350, 351, 435, 436
Density Function, 199, 200
Density Scale Factor, 162, 164
Deposit Specific Internal Energy, 162, 165
Determine Sharing, 139, 141, 200, 202
Determine Utau Via Nonlinear Law Of The Wall Iteration, 392, 394
Developer Command: Run To Run Repeatably, 132, 135
Diagnostic Stream, 132, 138
Diameter, 350, 351, 428–431, 456, 457, 469, 470, 479, 480, 483, 486, 487, 495, 498
Diameter Cutoff High, 497, 506
Diameter Cutoff Low, 497, 506
Diameter Number Represented Vector, 495, 498
Diameter Reduction Factor, 463, 464
Differentiate Expression, 148, 151
Disable Conjugate Heat Transfer Symmetry Checks, 227, 228
Disable Particle Transfer Checks, 418, 423
Distance Function Is Closest Receive Node To Send Centroid, 233, 235
Do Not Clip Mixture Fraction Variables, 179, 183
Drop Tolerance, 223

Edc Fuel Name, 380, 383
Edc Ignition Threshold Temperature, 380, 383
Edc Ignition Time, 381, 383
Edc Minimum Product Fraction, 381, 384
Edc Model Specification, 355, 380, 570
Edc Reaction Time Scale, 381, 384
Edge, 336, 340, 521, 525, 554, 558, 597, 601
Edge Variables, 336, 340, 521, 525, 554, 558, 597, 601
Effective Moduli Model, 162, 165
Element, 308, 311, 318, 321, 336, 340, 438, 441, 448, 451, 521, 525, 536, 539, 554, 558, 597, 601
Element Numerical Formulation, 162, 165
Element Variables, 336, 340, 521, 526, 554, 558, 597, 601
Emissivity, 350, 351, 634, 637, 652, 655, 667, 671, 689, 693, 709, 712, 723, 727, 741, 745, 760, 766, 779, 783
Emissivity Spectral File Name, 634, 636, 651, 654, 667, 671, 689, 693, 709, 711, 723, 726, 741, 744, 760, 765, 779, 782
Enable Large Ids, 336, 341, 521, 526, 554, 559, 597, 602
Enable Porous Media Model, 386, 387
Enable Timer, 132, 138
Energy Iteration Tolerance, 162, 166
Enforce Zero Flux For Turbulence Wall Bc, 723, 726, 796, 797
Enthalpy, 668, 672
Entrain Local Values, 689, 693
Evaluate Expression, 148, 151

Evaluate Flux Of, 284, 291
Evaluate From, 148, 154
Evaluate Function, 284, 292
Evaluate Gradient Of, 284, 292
Evaluate Postprocessor, 284, 292
Evaluate Time Derivative Of, 284, 292
Event, 251, 259, 261, 264, 266, 268, 269, 272, 274, 276, 277, 280, 281
Exclude, 336, 341, 522, 526, 554, 559, 597, 602
Exclude Ghosted, 233, 236
Execute Postprocessor Group, 259, 261, 264, 266, 268, 270, 272, 274, 276, 278, 280, 282, 284, 293
Exists, 308, 311, 318, 321, 326, 330, 336, 341, 439, 441, 448, 451, 512, 516, 522, 526, 536, 539, 544, 548, 554, 559, 597, 602
Expert Belos Solver, 212, 217
Expert Ifpack2 Preconditioner, 214, 215, 217, 219, 225
Expression Variable:, 148, 154
Expressions, 149
External Field For, 625, 627, 634, 636, 647, 648, 652, 654, 667, 671, 689, 693, 709, 712, 723, 727, 741, 744, 760, 765, 779, 783
External Field For Composite, 760, 765

Face, 308, 312, 318, 321, 336, 341, 439, 442, 448, 451, 522, 527, 536, 539, 554, 559, 597, 602
Face Variables, 336, 342, 522, 527, 554, 560, 597, 603
Favre Average Field, 303, 306
Fei Output Level, 139, 142, 200, 203
File Cycle Count, 326, 330, 512, 516, 544, 548
Filename, 430, 431
Filename Prefix, 507
Fill Fraction, 223, 224
Fill Level, 221, 222
Find Maximum Residuals, 284, 293
Finite Element Model, 133, 158
First Order Upwind Factor, 353, 359, 567, 574
Fix Pressure To, 353, 360, 567, 574
Fixed Boundary Condition On Surface, 286, 634
Fixed Time, 593, 595
Flow Must Exit Domain, 689, 694
Fluid Region Name, 418, 424
Fluid Variable Specification, 419, 435
Fluid Velocity Fluctuation Model, 418, 424
Flush Interval, 308, 312, 439, 442
Force Search In Model Coordinates, 400, 401
Format, 308, 312, 439, 442
From, 233, 236
Fuego Procedure, 133, 227
Fuego Region, 227, 284
Function For, 179, 183, 625, 627, 634, 637, 652, 655, 668, 672, 689, 694, 709, 712, 723, 727, 741, 745, 760, 766, 779, 783
Function For Composite, 760, 766
Function For Composite Initial_Mass_Fraction, 760, 767, 779, 784
Function For Emissivity, 634, 638, 652, 656, 668, 673, 689, 694, 709, 713, 723, 728, 741, 746, 760, 767, 780, 784
Function For Heat_Flux, 647, 649
Function For Injected_Temperature, 741, 746
Function For Mass Flux, 741, 746
Function For Mass_Diffusivity, 179, 184
Function For Mass_Fraction, 625, 628, 634, 638, 652, 656, 668, 673, 689, 695, 709, 713, 723, 728, 741, 746, 760, 768, 780, 785
Function For Mole_Fraction, 625, 628, 634, 638, 652, 656, 668, 673, 689, 695, 709, 714, 723, 729, 741, 747, 760, 768, 780, 785
Function For Radiation Boundary Temperature, 634, 639, 652, 657, 668, 674, 689, 695, 709, 714, 723, 729, 741, 747, 760, 768, 780, 785
Function For Radiation Environment Temperature, 634, 639, 652, 657, 668, 674, 689, 695, 709, 714, 723, 729, 742, 747, 761, 768, 780, 785
Function For Species_Density, 179, 184
Function For Species_Emissivity, 179, 185
Function For Species_Enthalpy, 180, 185
Function For Species_Production_Rate, 180, 185
Function For Species_Specific_Heat, 180, 186
Function For Species_Thermal_Conductivity,

180, 186
Function For Total_Pressure, 689, 696
Function For Transmissivity, 634, 639, 652, 657, 668, 674, 689, 696, 709, 714, 724, 729, 742, 747, 761, 769, 780, 786
Function For Transparent Band Emissivity, 634, 639, 652, 657, 668, 674, 689, 696, 710, 715, 724, 730, 742, 748, 761, 769, 780, 786
Function For Wall Temperature, 635, 640, 652, 658, 668, 675, 689, 696, 710, 715, 724, 730, 742, 748, 761, 769, 780, 786

Gas Species, 492–494
Gauss Point Integration Order, 233, 236
General Chemistry, 466
Geometric Tolerance, 233, 237
Global, 308, 312, 318, 322, 336, 342, 439, 442, 448, 452, 522, 527, 536, 540, 554, 560, 597, 603
Global Constants, 133, 173
Global Variables, 336, 342, 522, 528, 554, 560, 597, 603
Gmres Solver, 212, 215
Gravity Vector, 173

Heartbeat, 286, 308, 419, 438, 535, 593
Heat Capacity, 350, 352
Heat Flux Boundary Condition On Surface, 286, 647
Heat_Flux, 648, 649
History Output, 286, 318, 419, 448, 535, 536, 593
Hourglass, 163, 166
Hybrid Upwind Factor, 353, 360, 567, 575
Hybrid Upwind Method, 353, 361, 567, 575
Hybrid Upwind Shift, 353, 361, 567, 576
Hybrid Upwind Width, 353, 362, 567, 576

Ideal Gas Constant, 173
Ignite, 625, 628
Ignition Blending Time, 194, 195
Ignition Time, 194, 195
Ilu Fill, 139, 142, 200, 203
Ilu Omega, 139, 142
Ilu Threshold, 139, 142, 200, 203
Import Field, 285, 293
Inactive For Procedure, 163, 166
Include, 336, 342, 522, 528, 554, 560, 597, 603
Include All Blocks, 163, 166
Include Condensed Phase Convection, 668, 675
Include Continuity Residual Term, 353, 362, 567, 577
Include Edc Laminar Limit Model, 381, 384
Include Molecular Viscosity In K-E Diffusion Coefficient, 392, 395
Incremental Number Of Steps, 252, 253
Indicatemarkadapt, 259, 261, 264, 266, 268, 270, 272, 274, 276, 278, 280, 282
Indicatemarkadapt Using, 285, 293
Inflow Boundary Condition On Surface, 286, 651
Initial Condition Block, 286, 625
Initial Deltat, 252, 253
Initial Temperature, 350, 352
Initial Uniform Refinement For, 285, 294
Initial Velocity, 408, 411
Initialize, 250, 251
Initialize With Continuity Solve, 285, 294
Injected_Temperature, 742, 748
Inline Function For, 625, 628, 635, 640, 652, 658, 668, 675, 689, 697, 710, 715, 724, 730, 742, 748, 761, 769, 780, 786
Inline Function For Injected_Temperature, 742, 749
Inline Function For Mass Flux, 742, 749
Input Database Name, 326, 331, 512, 517, 544, 549
Input Mesh Name, 400, 401
Input Variables For, 180, 186
Input_Output Region, 227, 593
Insert Particle, 419, 456
Insertion Time, 430, 432, 456, 457, 469, 470, 479, 480, 483, 484, 486, 487, 496, 498
Inspect With File, 233, 237
Integer Data For Subroutine, 285, 294, 625, 629, 635, 640, 648, 649, 652, 658, 668, 676, 690, 697, 710, 716, 724, 731, 742, 750, 761, 770, 780, 787
Interaction List Filename, 430, 432
Interface Boundary, 724, 731, 742, 750, 761, 770, 780, 787
Interface Compression Factor, 414, 415

Interface Compression Model, 414, 415
Interface Curvature Algorithm, 414, 416
Interface Location Tolerance, 414, 416
Interface Sharpening Model, 415, 416
Interpolate, 233, 238
Interpolate Fluid Variable, 418, 424
Interpolate Function, 285, 295
Interpolation Function, 234, 238
Inversion Aversion Exponent, 163, 167
Inversion Aversion Stiffness, 163, 167
Inversion Aversion Transition Jacobian, 163, 167
Involve, 251, 252, 268, 270, 272, 274, 276, 278, 280, 282
Irradiation Node Variable, 230, 231
Iteration At Skip Step, 246, 247, 256, 257

Jacobi Preconditioner, 214, 215, 217, 219, 222

K-E Turbulence Model Parameter, 173, 174
K-W Turbulence Model Parameter, 173, 174
Klu2 Solver, 212, 213

Labels, 308, 313, 439, 443
Lagrangian Particles, 418, 424
Law_Of_Wall_Roughness_Parameter, 648, 651, 725, 741, 743, 758, 762, 779, 781, 795, 796, 800
Legend, 309, 313, 439, 443
Length, 350, 352, 496, 499
Length Vector, 496, 498
Length_Scale_Factor, 430, 432
Les Turbulence Model Parameter, 173, 174
Library Hdf5 File, 194, 196
Light Speed, 173, 174
Limit New Particle Splitting, 463, 464
Limit Turbulent Ke Production, 392, 395
Linear Bulk Viscosity, 163, 167
Load User Plugin File, 132, 135
Location, 403, 404
Log, 132, 139

Mark, 259, 262, 264, 267, 268, 271, 272, 275, 276, 279, 280, 283
Markadapt, 259, 262, 264, 267, 268, 271, 272, 275, 276, 279, 280, 283
Markadapt Using, 285, 295
Mass Flux, 668, 676, 742, 750
Mass Flux Boundary Condition On Surface, 286, 667
Mass Matrix, 390, 391
Mass Represented, 430, 432, 456, 457, 469, 471, 479, 480, 483, 484, 486, 488, 496, 499
Mass_Diffusivity, 180, 187
Mass_Flow_Rate, 496, 499
Mass_Fraction, 403, 405, 430, 433, 456, 457, 469, 471, 479, 480, 483, 484, 486, 488, 496, 499, 625, 629, 635, 641, 652, 659, 668, 676, 690, 698, 710, 716, 724, 732, 742, 750, 761, 771, 780, 788
Material, 133, 163, 168
Material =, 163, 168
Matrix Format, 139, 143, 200, 203
Matrix Output, 211, 213
Matrix Output Time Range, 211, 213
Matrix Reduction, 139, 143, 200, 204
Matrix Scaling, 139, 143, 200, 204, 212, 213
Max Energy Iterations, 163, 168
Maximum Iterations, 139, 144, 200, 204, 214–216, 218–221
Maximum Number Of Continuity_Momentum Nonlinear Iterations, 353, 362, 567, 577
Maximum Number Of Energy_Species Nonlinear Iterations, 353, 363, 568, 577
Maximum Number Of Gas_Solid_Momentum Nonlinear Iterations, 353, 363, 568, 578
Maximum Number Of Kepsilon Nonlinear Iterations, 353, 363, 568, 578
Maximum Number Of Komega Nonlinear Iterations, 353, 363, 568, 578
Maximum Number Of Ksgs Nonlinear Iterations, 353, 364, 568, 578
Maximum Number Of Mixture Fraction Nonlinear Iterations, 353, 364, 568, 579
Maximum Number Of Nonlinear Iterations, 227, 229, 354, 364, 568, 579
Maximum Number Of Particles Per Parcel, 463, 464
Maximum Number Of Solid Phase Nonlinear Iterations, 354, 364, 568, 579
Maximum Number Of Soot Nuclei Nonlinear

Iterations, 354, 365, 568, 579
Maximum Number Of Species Nonlinear Iterations, 354, 365, 568, 580
Maximum Number Of Species_Product Nonlinear Iterations, 354, 365, 568, 580
Maximum Number Of V2F Nonlinear Iterations, 354, 365, 568, 580
Maximum Particle Subcycles, 418, 425
Maximum Temperature Allowed From Temperature Extraction, 285, 295
Maximum Time Step, 246, 247, 256, 257
Maximum Wall Time, 354, 366, 568, 580
Maximum Warning Count, 132, 135
Mdot, 403, 404
Membrane Hourglass, 163, 168
Mesh Motion Model On, 405, 406
Mesh Motion Specification, 355, 405, 570
Minimum Effective Dilatational Moduli Ratio, 163, 169
Minimum Effective Shear Moduli Ratio, 163, 169
Minimum Mass Per Parcel, 464, 465
Minimum Number Of Nonlinear Iterations, 354, 366, 568, 581
Minimum Particle Subcycles, 418, 425
Minimum Soot Production Temperature, 381, 384
Minimum Temperature Allowed From Temperature Extraction, 285, 296
Minimum Time Step, 246, 248, 256, 257
Minimum_Diameter, 465, 467
Minimum_Mass, 465, 467
Model, 163, 169
Mole_Fraction, 625, 630, 635, 641, 652, 659, 668, 677, 690, 698, 710, 717, 724, 732, 742, 751, 761, 771, 780, 788
Molecular_Weight, 180, 187, 435, 436
Momentum Entrainment, 690, 698
Monitor, 309, 313, 439, 443
Muelu Preconditioner, 214, 215, 217, 219, 224
Multiphase, 386, 387
Multiphase Model Specification, 355, 386, 570

Nodal, 309, 314, 318, 322, 336, 343, 439, 444, 448, 452, 522, 528, 536, 540, 554, 561, 598, 604
Nodal Variables, 336, 343, 522, 528, 554, 561, 598, 604
Node, 309, 314, 318, 322, 336, 343, 439, 444, 448, 452, 522, 529, 536, 540, 554, 561, 598, 604
Node Variables, 336, 343, 522, 529, 554, 561, 598, 604
Nodes Outside Region, 230, 231, 234, 239
Nodeset, 309, 314, 318, 322, 336, 344, 439, 444, 448, 452, 522, 529, 536, 540, 554, 562, 598, 605
Nodeset Variables, 336, 344, 522, 529, 554, 562, 598, 605
Nonconformal Boundary Condition On Surface, 286, 796
Nonlinear, 268, 272, 276
Nonlinear Residual Norm Tolerance, 354, 366, 568, 581
Nonlinear Residual Plotfile Path, 285, 296
Nonlinear Stabilization Method, 354, 367, 568, 581
Nonlocal Regularization Kmeans Cell Size, 163, 170
Nonlocal Regularization Kmeans Maximum Iterations, 163, 170
Nonlocal Regularization Kmeans Tolerance, 163, 170
Nonlocal Regularization On, 163, 171
Nonlocal Regularization Partitioning Scheme, 163, 171
Normal Treatment, 796, 797
Normal Vector, 496, 500
Nozzle Radius, 496, 500
Num Levels, 140, 144, 200, 205
Num Smoother Iterations, 415, 417
Number, 456, 458
Number Of Adaptivity Steps, 252, 254
Number Of Fluid Species, 418, 425
Number Of Logged Input Clipping Events, 194, 196
Number Of Steps, 252, 254
Number Of Sweeps, 222, 223, 225
Number Of Time Steps, 245
Number Representative Points, 496, 500
Number Represented, 430, 433, 456, 458, 469, 471, 479, 481, 483, 484, 486, 488, 496,

501
Number Represented Vector, 496, 500

Ode Solver Parameters, 181, 466
Offset Time, 593, 595
Omit Block, 158, 161
Omit Density Time Derivative In Continuity Equation, 354, 367, 568, 582
Omit Diffusion Terms, 652, 659, 690, 699
Omit Low Reynolds Turbulence Dissipation Source Term, 392, 395
Omit Near Wall Combustion, 381, 385
Omit Near Wall Turbulent Ke Transport Equation, 392, 396
Omit Pressure Stabilization Term, 796, 798
Omit Turbulence Source Terms On Block(S), 392, 396
Omit Velocity Divergence In Turbulent Production Term, 392, 396
Omit Volume, 158, 161
Open Boundary Condition On Surface, 286, 689
Opposing Sides, 796, 798
Optional, 326, 331, 512, 517, 544, 549
Ordinate, 148, 155
Ordinate Offset, 149, 155
Ordinate Scale, 149, 155
Orientation, 350, 352
Origin, 408, 411
Orthog Method, 140, 144
Output, 259, 262, 268, 271, 272, 275, 276, 279, 280, 283
Output Database Name, 326, 331, 512, 517, 544, 549
Output Indicator Global Value Of, 285, 296
Output Library Variable, 194, 196
Output Mesh, 336, 344, 522, 530, 554, 562, 598, 605
Output Mesh Name, 400, 401
Output Nonlinear Residual Field For Equation, 354, 367, 568, 582
Output Number Of, 285, 297
Output Number Of Timesteps, 285, 297
Output On Signal, 176, 177, 309, 314, 318, 323, 326, 332, 336, 345, 439, 444, 448, 453, 512, 517, 522, 530, 536, 541, 544, 550, 554, 563, 598, 606
Output Postprocessor History Of, 285, 297
Output Preferred Integration Order Of, 285, 297
Output Scheduler, 133, 175
Output Variable, 507, 508
Overlap Drop Tolerance, 400, 402
Overlay Count, 326, 332, 512, 518, 544, 550
Overwrite, 318, 323, 326, 332, 337, 345, 448, 453, 512, 518, 522, 530, 536, 541, 544, 550, 555, 563, 598, 606
Oxidizer, 493, 494

Param-Bool, 200, 205, 209, 210
Param-Int, 140, 144, 201, 205, 209, 210
Param-Real, 140, 145, 201, 206, 209, 210
Param-String, 140, 145, 201, 206, 209, 211
Param-Sublist, 209, 211
Parameters For, 250, 252
Parameters For Block, 158, 162
Parameters For Fuego Region, 245, 246, 252, 256
Parametric Tolerance, 234, 239
Particle Average, 419, 462
Particle Average_Diameter, 496, 501
Particle Average_Velocity, 496, 501
Particle Breakup Model, 419, 463, 465, 467
Particle Breakup Parameter, 464, 465
Particle Definition, 419, 428, 430, 435, 456, 462, 465, 470, 476, 479, 482, 483, 486, 487, 490, 497, 507
Particle Diameter Distribution Parameter, 469, 471, 496, 501
Particle Diameter Distribution Type, 469, 472, 496, 502
Particle Energy Equation, 465, 467
Particle Evaporation, 490, 492
Particle Filled Shape, 419, 469
Particle Filled Tree, 419, 476
Particle Inflow Boundary Condition On Surface, 419, 486
Particle Interaction Model, 420, 465, 468
Particle Interface, 420, 490
Particle Is Stationary, 465, 468
Particle Log File, 496, 502

Particle Material, 420, 494
Particle Maximum Search Grid Dimension, 418, 425
Particle Open Boundary Condition On Surface, 420, 495
Particle Random Number Generator Seed, 418, 426
Particle Reaction, 490, 493
Particle Rebalance Imbalance Threshold, 419, 426
Particle Rebalance Step Frequency, 419, 427
Particle Region, 227, 418
Particle Reservoir Size, 419, 427
Particle Shape Parameter, 470, 473
Particle Shape Type, 470, 474
Particle Species, 490, 491, 761, 771
Particle Spray, 420, 495
Particle Stats Output, 420
Particle Surface Interaction Type, 508, 509
Particle Tabular Output, 420, 507
Particle Tree Parameter, 476, 477
Particle Tree Type, 476, 479
Particle Type, 465, 468
Particle Velocity Distribution Parameter, 469, 472, 496, 503
Particle Velocity Distribution Type, 470, 473, 496, 503
Particle Velocity Equation, 466, 468
Particle Wall Boundary Condition On Surface, 420, 508
Particle_Contact_Angle, 508, 510
Particle_K_Pf_Crit, 508, 510
Particle_Maximum_Number_Fingers, 508, 510
Particle_Random_Graze_Angle_Maximum, 508, 510
Particle_Random_Graze_Angle_Minimum, 508, 511
Particle_Splash_Parameter_A, 508, 511
Particle_Splash_Parameter_B, 508, 511
Perform Initial Particle Rebalance, 419, 428
Periodic, 287
Periodic Boundary Condition On Surface, 287
Periodic Constant Momentum Body Source Term, 354, 368, 568, 583
Periodicity Time, 593, 596
Phase, 163, 171
Phase Change Model, 415, 417
Pitch Acceleration, 408, 411
Pitch Initial Velocity, 408, 412
Planck Constant, 173, 175
Point Source, 355, 403, 570
Polynomial Order, 140, 146, 201, 206
Pool, 742, 751
Porosity, 430, 433, 456, 458, 470, 475, 479, 481, 483, 485, 487, 488, 496, 504
Porosity Field, 668, 677
Porous Condensed Temperature Field, 668, 677
Porous Flux Field For, 668, 678
Porous Penalty Field For, 668, 678
Porous Target Value Field For, 668, 679
Porous Volumetric Heat Transfer Coefficient, 669, 680
Post Process Delta Area, 652, 660, 669, 680, 690, 699, 724, 732
Post Process Delta Heat Flux, 724, 733
Post Process Delta Mass Flow Rate, 652, 660, 669, 680, 690, 699
Post Process Delta Pressure, 652, 660, 669, 681, 690, 700, 724, 733
Post Process Delta Thrust, 652, 661, 669, 681, 690, 700
Post Process Delta Viscous Stress, 724, 733
Post Process Interface Normal, 415, 417
Post Process On Surface, 287, 759
Post Process Yplus, 724, 734, 742, 751, 761, 772, 780, 788, 796, 798
Post Processing File Name, 759
Postprocess, 635, 641, 648, 650, 652, 661, 669, 682, 690, 700, 710, 717, 724, 734, 742, 752, 761, 772, 780, 789
Postprocessor Group, 287
Prandtl_Number, 180, 187
Precision, 309, 315, 439, 445
Preconditioner Subdomain Overlap, 140, 146, 201, 207
Preconditioning Method, 140, 146, 201, 207
Preconditioning Steps, 140, 146, 201, 207
Predictor Algorithm, 390, 391
Preset Solver, 212, 220
Pressure Penalty Multiplier, 796, 799

Print Timer Information Every, 132, 139
Print Values Of, 285, 298
Process Initial Condition, 285, 298
Progress_Variable, 403, 405, 625, 630, 635, 643, 652, 662, 669, 683, 690, 702, 710, 718, 724, 735, 742, 753, 761, 773, 780, 790
Project Nodes, 652, 662, 724, 735, 742, 753, 761, 773, 780, 790
Projection Method, 354, 368, 568, 583
Property, 318, 323, 326, 333, 337, 345, 448, 453, 512, 518, 522, 531, 536, 541, 544, 551, 555, 563, 598, 606
Property Specification For Fuego Material, 133, 179
Property Specification For Material, 133, 199

Quadratic Bulk Viscosity, 163, 171

Rad Transport Spectral Model Specification, 356, 570
Radiation Boundary Temperature, 635, 643, 653, 663, 669, 683, 690, 702, 710, 719, 724, 736, 742, 753, 761, 774, 780, 790
Radiation Boundary Temperature Field, 635, 643, 653, 663, 669, 683, 690, 702, 710, 719, 724, 736, 742, 754, 761, 774, 780, 791
Radiation Environment Temperature, 635, 644, 653, 663, 669, 684, 690, 703, 710, 719, 724, 736, 742, 754, 761, 774, 780, 791
Radiation Environment Temperature Field, 635, 644, 653, 664, 669, 684, 690, 703, 710, 719, 724, 737, 742, 754, 761, 775, 780, 791
Radiation Transport Equation Model Specification, 356, 387, 570
Randomize Pressure, 354, 369, 568, 584
Reaction Rate Parameter, 490, 491
Real Data For, 180, 188
Real Data For Subroutine, 285, 298, 625, 630, 635, 643, 648, 651, 653, 663, 669, 683, 690, 702, 710, 718, 724, 736, 742, 753, 761, 774, 780, 790
Receive Blocks, 234
Rectangle, 349, 350
Reference, 180, 188
Reference Mass_Fraction, 180, 188
Reference Mole_Fraction, 180, 189
Reference Progress_Variable, 180, 189
Reference Temperature, 759
Reference Temperature Node Variable, 230, 231
Reflection Model, 508, 509
Region Participates In Time Step Selection When Inactive, 285, 298
Reinitialize Transient, 252, 254
Remove Block, 163, 172
Reordering, 221–224
Reset Filter On Restart, 303, 306
Residual Norm Scaling, 140, 147, 201, 207
Residual Norm Tolerance, 140, 147, 201, 208
Residual Scaling, 214–216, 218, 220, 221
Restart, 132, 136, 326, 333, 512, 519, 544, 551
Restart Data, 287, 326, 420, 511, 535, 544, 593
Restart Iterations, 140, 147, 201, 208, 215, 217
Restart Time, 132, 136, 326, 334, 512, 519, 544, 552
Results Output, 287, 336, 420, 521, 535, 554, 593, 597
Reynolds Average Field, 303, 307
Rigid Body Motion Model, 405, 407
Roll Acceleration, 408, 412
Roll Axis, 408, 412
Roll Initial Velocity, 408, 413
Row Ordering, 201, 208

Scale By, 149, 156
Schmidt_Number, 180, 190
Search Coordinate Field, 234, 240
Search Expansion Factor, 796, 799
Search Geometric Tolerance, 234, 240
Search Method, 796, 799
Search Surface Gap Tolerance, 234, 240
Search Tolerance, 796, 800
Search Type, 234, 241
Section, 163, 172
Select One Receiver For Each Send Object, 234, 241
Select One Unique Receiver For Each Send Object, 234, 241
Send, 234, 242
Send Block, 234, 242, 400, 402

Send Blocks, 234
Send Field, 234, 242
Sequential, 260, 264, 267
Serialized Io Group Size, 132, 137
Set Function Parameter, 285, 299
Set Length Unit Conversion Factor, 285, 299
Set Mass Unit Conversion Factor, 285, 299
Set Schmidt Number For Progress Variable, 180, 190
Set Temperature Units, 285, 300
Set Time Unit Conversion Factor, 285, 300
Set Unit System To, 285, 301
Sgs Preconditioner, 214, 216, 217, 219, 225
Shared Ownership Rule, 140, 147, 201, 208
Sideset, 337, 346, 522, 531, 555, 564, 598, 607
Sideset Variables, 337, 346, 522, 531, 555, 564, 598, 607
Sierra, 132
Simulation Max Global Iterations, 259, 262
Simulation Start Time, 259, 263
Simulation Termination Time, 259, 263
Skip Pressure Update If Continuity Solve Fails, 354, 369, 569, 584
Skip Step, 246, 248, 256, 258
Skip Steps For Pmr, 227, 229, 246, 248, 256, 258
Smoother Fourier Number, 415, 417
Solid Mechanics Use Model, 163, 172
Solid Object, 287, 349, 625, 631
Solution Control Description, 227, 250
Solution Method, 140, 148, 201, 209
Solution Options, 287, 353, 535, 567
Solve Transpose, 201, 209
Solver Type, 220, 221
Source Term For, 194, 197
Source Term Function, 354, 370, 569, 584
Source Term Subroutine, 354, 370, 569, 585
Source Vector Name, 400, 402
Species Name For, 387, 390
Species_Density, 180, 191
Species_Emissivity, 180, 191
Species_Enthalpy, 180, 191
Species_Production_Rate, 180, 192
Species_Specific_Heat, 180, 192
Species_Thermal_Conductivity, 180, 193
Specific Surface Area Field, 669, 684
Specific_Heat, 435, 436
Sphere, 349, 350
Sponge Layer Blending Function, 690, 703
Sponge Layer Thickness, 690, 703
Sponge Layer Viscosity, 690, 704
Sponge Layer Viscosity Type, 690, 704
Spray_Angle, 496, 504
Spray_Angle_End, 496, 504
Spray_Angle_Start, 496, 504
Start Time, 176, 178, 245, 252, 254, 309, 315, 318, 324, 326, 334, 337, 346, 439, 445, 448, 454, 512, 520, 522, 532, 536, 542, 544, 552, 555, 564, 593, 596, 598, 607
Starting Time, 303, 307
Stefan Boltzmann Constant, 173, 175
Stop Simulation If Peak Velocity Exceeds, 354, 371, 569, 585
Stream Name, 309, 315, 439, 445
Subcycle, 272, 276, 280
Subdomain Overlap Level, 221–224
Subgrid Mixing Model, 387, 390
Subroutine For, 180, 190, 625, 631, 635, 644, 653, 664, 669, 684, 690, 704, 710, 720, 724, 737, 742, 754, 761, 775, 780, 791
Subroutine For Composite, 761, 775
Subroutine For Composite Initial_Mass_Fraction, 780, 792
Subroutine For Emissivity, 635, 645, 653, 664, 669, 685, 690, 705, 710, 720, 724, 737, 742, 755, 761, 776, 781, 792
Subroutine For Heat_Flux, 648, 651
Subroutine For Mass_Fraction, 625, 631, 635, 645, 653, 665, 669, 685, 690, 705, 710, 720, 724, 738, 742, 755, 761, 776, 781, 793
Subroutine For Mole_Fraction, 625, 632, 635, 645, 653, 665, 669, 686, 690, 706, 710, 721, 724, 738, 742, 756, 761, 776, 781, 793
Subroutine For Radiation Boundary Temperature, 635, 645, 653, 665, 669, 686, 690, 706, 710, 721, 724, 738, 743, 756, 761, 777, 781, 793
Subroutine For Radiation Environment Temperature, 635, 646, 653, 665, 669,

686, 690, 706, 710, 721, 724, 738, 743, 756, 762, 777, 781, 793
Subroutine For Total_Pressure, 691, 707
Subroutine For Transmissivity, 635, 646, 653, 666, 669, 686, 690, 706, 710, 721, 725, 739, 743, 756, 762, 777, 781, 794
Subroutine For Transparent Band Emissivity, 635, 646, 653, 666, 669, 687, 690, 707, 710, 722, 725, 739, 743, 757, 762, 777, 781, 794
Subroutine For Wall Temperature, 635, 646, 653, 666, 669, 687, 691, 707, 710, 722, 725, 739, 743, 757, 762, 778, 781, 794
Superlu Solver, 212, 218
Surface, 337, 346, 522, 532, 555, 564, 598, 607
Surface Gap Tolerance, 230, 231
Surface Variables, 337, 347, 522, 532, 555, 565, 598, 608
Symmetry Boundary Condition On Surface, 287, 709
Synchronize Output, 176, 178, 309, 316, 318, 324, 326, 334, 337, 347, 439, 446, 448, 454, 512, 520, 522, 532, 536, 542, 544, 552, 555, 565, 598, 608
System, 250, 259

Table Interpolation Method, 194, 197
Table Units Are In, 194, 197
Tabular Property Library, 181, 194
Temperature, 430, 434–436, 456, 459, 470, 475, 479, 481, 483, 485, 487, 489, 496, 505
Termination Time, 176, 178, 244–246, 252, 254, 309, 316, 318, 324, 326, 335, 337, 348, 439, 446, 448, 454, 512, 520, 522, 533, 536, 542, 544, 553, 555, 566, 598, 609
Test Error Messages To File, 132, 137
Teuchos Parameter Block, 201, 209
Thermal_Conductivity, 435, 437
Time Average Using Period, 462, 463
Time Control, 227, 244
Time Filter, 392, 397
Time Integration Specification, 356, 390, 570
Time Interval Length, 303, 308
Time Scale Factor, 158, 161
Time Start, 227, 229
Time Step, 245, 246
Time Step Change Factor, 246, 249, 256, 258
Time Step Quantum, 252, 255
Time Step Style, 252, 255
Time Stepping Block, 244, 245
Time_Scale_Factor, 430, 433
Timeseries Name, 337, 348, 522, 533, 555, 566, 598, 609
Timestamp Format, 309, 316, 439, 446
Timestep Adjustment Interval, 176, 179, 309, 317, 318, 325, 326, 335, 337, 348, 400, 403, 439, 447, 448, 455, 512, 521, 522, 534, 537, 543, 544, 553, 555, 566, 598, 609
Title, 132, 137, 319, 325, 337, 348, 449, 455, 522, 534, 537, 543, 555, 566, 598, 609
Total Change In Time, 252, 255
Total_Pressure, 691, 708
Tpetra Equation Solver, 133, 211
Track Particles For Tabular Output, 430, 434, 456, 458, 470, 475, 479, 481, 483, 485, 487, 489, 496, 505
Transfer, 228, 233, 251, 252, 259, 263, 264, 267, 268, 271, 272, 275, 276, 279, 280, 283
Transfer Between Fuego Fluid Region, 230, 232
Transfer Between Input Region, 230, 232
Transfer Between Region, 230, 232
Transfer Problem Definition, 230, 233
Transient, 260, 264, 272
Transient Step Type, 246, 249, 256, 259
Transmissivity, 635, 647, 653, 667, 669, 687, 691, 708, 710, 722, 725, 740, 743, 757, 762, 778, 781, 795
Transparent Band Emissivity, 635, 647, 653, 666, 669, 687, 691, 707, 710, 722, 725, 739, 743, 757, 762, 778, 781, 794
Transverse Shear Hourglass, 164, 172
Trilinos Equation Solver, 133, 200
Trunk Info, 476, 483
Turbulence Model, 173, 175, 392, 397
Turbulence Model Parameter, 393, 397
Turbulence Model Specification, 356, 392, 570
Turbulence Postprocessor, 393, 398
Turbulence_Intensity, 669, 688, 691, 708
Turbulent Mixing Model, 180, 193
Turbulent_Dissipation, 435, 437

Turbulent_Kinetic_Energy, 435, 437
Type, 149, 156

Umfpack Solver, 212, 217
Under Relax, 354, 371, 569, 586
Under Relax Momentum By, 354, 372, 569, 586
Under Relax Pressure By, 354, 373, 569, 587
Under Relax Solid_Momentum By, 354, 373, 569, 587
Under Relax Temperature_Extraction By, 355, 373, 569, 587
Under Relax Turbulent_Viscosity By, 393, 398
Under Relaxation, 408, 413
Update Enthalpy And Density, 625, 632
Upwind Limiter, 355, 374, 569, 588
Upwind Method, 355, 374, 569, 588
Use Approximate Flamelet Enthalpy Reconstruction, 194, 198
Use Buoyant Vorticity Generation, 393, 398
Use Consistently Volume Averaged Properties, 194, 198
Use Dynamic Ksgs, 393, 399
Use Edc Product Transport, 381, 385
Use Equation Solver, 355, 375, 569, 589
Use Equilibrium Production Model, 725, 740, 743, 758, 762, 778, 781, 795, 796, 800
Use Explicit Treatment Of Edc Source Terms, 381, 385
Use External Continuity Source, 355, 375, 569, 590
Use External Energy Source, 355, 376, 569, 590
Use External Mixture_Fraction Source, 355, 376, 569, 590
Use External Momentum Source, 355, 376, 569, 590
Use External Soot_Mass_Fraction Source, 355, 377, 569, 591
Use External Species Source, 355, 377, 569, 591
Use Field, 194, 198
Use Finite Element Model, 286, 303, 419, 428, 535, 536, 593, 597
Use Fluxes, 653, 667
Use Generic Names, 158, 162
Use Initialize, 259, 263
Use Library Source, 194, 198
Use Library Variable, 194, 199
Use Lumped Velocity Density Interpolation, 355, 377, 569, 591
Use Material, 158, 162
Use Mpmd Radiation, 285, 301
Use Output Scheduler, 176, 179, 309, 317, 319, 325, 326, 335, 337, 349, 439, 447, 449, 455, 512, 521, 522, 534, 537, 543, 544, 553, 555, 567, 598, 610
Use Particle Species, 466, 469
Use Porous Coupling For, 669, 688
Use Pure Oxygen For Oxidizer Mixture, 381, 386
Use Radiation Source From External Region, 355, 377, 569, 592
Use Reduced Diffusion Stencil, 285, 302
Use Second Order Implicit Time Integration, 390, 392
Use Simple Radiation Source With Tref, 286, 302
Use Sintef Soot Model, 381, 386
Use Skew Symmetric Central Operator, 355, 378, 569, 592
Use Solution Steering With Interval, 286, 302
Use Species Material, 466, 469
Use Sponge Layer, 691, 708
Use System, 250
User Subroutine File, 132, 137

Values, 149, 157
Variable, 309, 317, 319, 325, 439, 447, 449, 455, 537, 543
Velocity Is Relative To Mesh, 725, 740
Virtual Thermocouple Model On Block, 287, 350
Viscosity, 435, 437
Vof Gas Material, 180, 193
Vof Liquid Material, 180, 194
Vof Model Specification, 356, 414, 570
Vof Shape For, 626, 632
Volume, 626, 633

Wall Boundary Condition On Surface, 287, 723
Wall Distance Band Size, 393, 399
Wall Mass Inject Boundary Condition On Surface, 287, 741

Wall Model, 393, 399
Wall Temperature, 635, 647, 653, 667, 669, 688, 691, 709, 710, 723, 725, 740, 743, 758, 762, 779, 781, 795
Water, 493, 494
Width, 496, 505

X, 456, 459
X Offset, 149, 156
X Scale, 149, 156
X1, 456, 459
X2, 456, 459
X_Velocity, 428–430, 434, 435, 438, 456, 460, 470, 475, 479, 482, 483, 485, 487, 489, 496, 505
Xml File, 217, 218, 224–226

Y, 456, 460
Y Offset, 149, 157
Y Scale, 149, 157
Y1, 456, 460
Y2, 456, 460
Y_Velocity, 428–430, 434, 435, 438, 456, 461, 470, 476, 479, 482, 483, 486, 487, 489, 496, 506
Yaw Acceleration, 408, 413
Yaw Axis, 408, 414
Yaw Initial Velocity, 408, 414

Z, 456, 461
Z1, 456, 461
Z2, 456, 461
Z_Velocity, 428–430, 435, 438, 456, 462, 470, 476, 479, 482, 483, 486, 487, 490, 497, 506

DISTRIBUTION

Email—Internal [redacted]

Name	Org.	Sandia Email Address
Technical Library	01177	libref@sandia.gov

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.