SANDIA REPORT
SAND2019-3965

Printed April 9, 2019

Sandia National Laboratories

SIERRA/Aero User Manual – Version 4.52

Sierra/Aero 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.

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

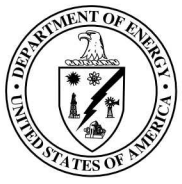
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

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

SIERRA/Aero is a compressible fluid dynamics program intended to solve a wide variety compressible fluid flows including transonic and hypersonic problems. This document describes the commands for assembling a fluid model for analysis with this module, henceforth referred to simply as Aero for brevity. Aero is an application developed using the SIERRA Toolkit (STK). The intent of STK is to provide a set of tools for handling common tasks that programmers encounter when developing a code for numerical simulation. For example, components of STK provide field allocation and management, and parallel input/output of field and mesh data. These services also allow the development of coupled mechanics analysis software for a massively parallel computing environment.

CONTENTS

1. INTRODUCTION

SIERRA/Aero is a compressible fluid dynamics program intended to solve a wide variety compressible fluid flows including transonic and hypersonic problems. This document describes the commands for assembling a fluid model for analysis with this module, henceforth referred to simply as Aero for brevity. Aero is an application developed using the SIERRA Toolkit (STK). The intent of STK is to provide a set of tools for handling common tasks that programmers encounter when developing a code for numerical simulation. For example, components of STK provide field allocation and management, and parallel input/output of field and mesh data. These services also allow the development of coupled mechanics analysis software for a massively parallel computing environment. In the definitions of the commands that follow, the term `Real_Max` denotes the largest floating point value that can be represented on a given computer. `Int_Max` is the largest such integer value.

1.1. OVERVIEW OF THE INPUT FILE STRUCTURE

An Aero analysis model is described by commands contained in an ASCII input file. The structure of the input file follows a specific hierarchy, which exists largely for historical reasons. The terminology associated with this hierarchy is domain, procedure, and region. The domain is the top or highest scope of the analysis model description. It serves as a container for the procedure, which in turn contains a region.

The procedure scope is used to specify the time integration, user functions, and the region. The region scope is used to specify details about the fluid model. These include boundary and initial conditions, solution output, restart and solution options Figure 1.1-1 shows this nesting.

There are two types of commands used. The first type is referred to as a block command. A block command is used to group a set of commands for a specific functionality. As indicated in Figure 1.1-1, block commands are used to define the domain, procedure, and region within the input file. A block command always constitutes a pair of lines in the input file. The first line starts with `Begin`, and the last line starts with `End`:

```
Begin SIERRA name
  ...
End
```

The key words for the block command follow the `Begin` and `End`. In most cases, a user-specified name is added to the end of the beginning and ending command line (optionally it can be left off of the ending command, as shown above). The `SIERRA` command block defines the problem domain. An Aero input file must contain only a single domain.

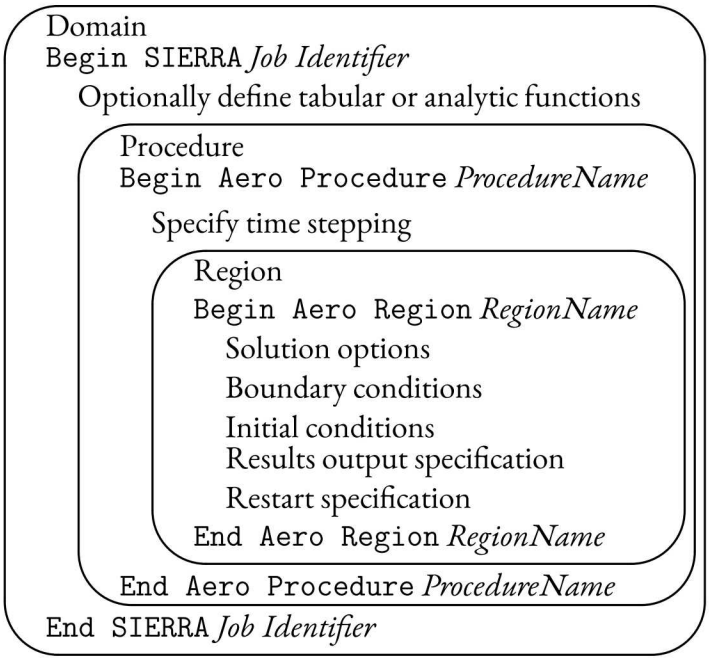


Figure 1.1-1.. Domain, procedure, region hierarchy.

The second type of command is referred to as a line command. The line command is used to specify parameters within a given block command. In the command descriptions, the scope of each block and line command is identified. Note that the ordering of line commands within a block command is arbitrary:

```
Begin Gas Properties
  gamma = 1.4
  specific_r = 287
End
```

is the same as

```
Begin Gas Properties
  specific_r = 287
  gamma = 1.4
End
```

Note that the terms "command block" and "block command" are interchangeable. The ordering of command blocks within the domain or region scope is also arbitrary. This allows freedom for the analyst to arrange and group parts of the fluid model to her or his liking.

The SIERRA command block must contain a block for the Aero procedure:

```
Begin Aero procedure name
  ...
End
```

This block command is used to contain Aero commands that are associated with the time stepping procedure. The name of the procedure is specified by the user. Note that the Aero procedure command block must be present in the input file and must contain exactly one `Aero Region` command block.

The region is defined via a block command of the form:

```
Begin AERO REGION name
  ...
End
```

The region is a container for the boundary conditions, initial conditions, solution options, and so on.

Comments in the input file start with either the $ or # character. All data following these characters on a command line is ignored. A command in the input file can be continued to the next line in the file by using the character pair \$ or \#. All data following these character pairs on that line is ignored. The data on the following line in the command file is joined and parsing continues. An example is the line command used to specify the title of a thermal model:

```
Begin SIERRA Job Identifier
  #
  $ This model for Aero simulates a fin
  #
  Title \$ The title command is used to set the analysis title
  Analysis of a fin.  \$
End
```

In this example, the line commands between the "Begin" and "End" of the block are parsed as "Title Analysis of a fin."

1.2. SYNTAX CONVENTIONS FOR COMMANDS

1.2.1. Case

All of the command key words, delimiters, and parameters are not case sensitive. For example,

```
Use Flow State Freestream
```

and

```
USE fLow sTatE FreESTream
```

are equivalent. The exception to this rule is file names, which are used for input and output. The current operating systems on which SIERRA applications run are UNIX based, and the file names are case sensitive.

1.2.2. Spaces

Command key words, delimiters, and parameters must have spaces around them.

1.2.3. User-Specified Input

There are four types of user-specified input. They are: character strings, integer numbers, real numbers and enumerations. When the parameter to a command is a character string, the parameter will be designated by `(C)`. When the parameter is an integer number, it will be designated by `(I)`. If the parameter is a real number, then it will be designated by `(R)`. Note that real numbers may be entered in decimal form or exponential form. For example 0.0001, .1E-3, 10.0d-5 are all equivalent. Furthermore, an integer is promoted to a Real, but a Real is not demoted to an integer. For example if a Real is expected, it is legal to specify 1, 1., or 1.0. However, if an integer is expected, the user must specify it without a decimal point. Enumerations appear to be strings, but are more akin to named integers. They are described in Section 1.2.6.

1.2.4. Lists of User-Specified Input

For the case when a list of parameters is required for a command, `(C, ...)` is used for a list of character strings, `(I, ...)` for a list of integers, and `(R, ...)` for a list of real numbers. For a list of character strings, the separator between the strings must be one or more spaces or tab characters. Do not surround strings with quotes. For the case of a list of numbers, the comma can also be used as a separator between numbers in the list.

1.2.5. Delimiters

The key words in a command will usually be separated from parameters by a choice between the equal sign(=) or a word. The choices are contained within {}, separated by |. The user may choose one of the delimiters from the available choices. For example, the line command to specify the ratio of specific heats within the Gas Properties block command is given as

```
gamma {=|IS} (R)
```

Valid forms in the input file are

```
Begin Gas Properties
  ...
  gamma = 1.4
  ...
End
```

and

```
Begin Gas Properties
  ...
  gamma is 1.4
  ...
End
```

Numerous commands make use of a delimiter. There is only an aesthetic difference among them: they are intended to be otherwise interchangeable, without regard to the number of arguments that follow. For example, the specification

```
Begin Gas Properties
  ...
  gamma are 1.4
  ...
End
```

is perfectly legal to parse, but very poor grammar.

1.2.6. Enumerated Input Parameters

Certain commands have predefined parameters, called enumerations, which are listed within {}. Each parameter in the list is separated using |. For example, the line command to set the type of the eigenvalue fix is `Eigenvalue Fix Type` . The choices for this parameter are listed as

```
{ maximum|nofix|scaled|unscaled }
```

which indicates the admissible choices, e.g., `Eigenvalue Fix Type = nofix`.

1.2.7. Indentation

All leading spaces and/or tab characters are ignored in the input file. It is recommended that indentation be used to improve readability of the input file to the analysis application.

1.2.8. Including Files

External text files containing input commands can be included at any point in the Aero input file using the `INCLUDEFILE` command. This command can be used in any context in the input file. To use this command, simply use the command `INCLUDEFILE` followed by the name of the file to be included. For example, the command:

```
INCLUDEFILE extrafile.i
```

would include the contents of `extrafile.i` at the locations where it is included in the input file. The included file is contained in the standard echo of the input that is provided at the beginning of the log file.

NOTE: Though this line command works well in many simple circumstances, it is known to cause issues when file names are involved. The most robust method to include files is to use Aprepro's `include` function and pre-process the input file with aprepro before running. An example is below:

```
#{include(extrafile.i)}
```

1.3. UNITS

Aero's focus is on the mathematics. It has no provisions for keeping track of, or performing conversions between, different systems of units. This means that the user can pose a problem in whatever self-consistent system (SI, MKS, British...) he/she chooses, but the onus is on him/her to know what units the output quantities will be in. Be aware that if you mistakenly specify say, thermal conductivity, in BTU/(hr-ft-deg F) while inputting dimensions in meters and temperatures in deg Celsius, Aero will run, but the results will be difficult to interpret.

2. USER FUNCTIONS AND TIME STEPPING

This chapter describes high-level commands such as mathematical functions and the procedure command block. These commands have the highest scope in the input file, and must reside within the outermost sierra command block. The Aero procedure contains the time stepping controls, as well as the region block, which contains the mathematical description of the physics, boundary conditions etc. The Region is described in more detail in Chapter 3.

2.1. SIERRA

Scope:

```
Begin Sierra jobName

    Begin Conchas Procedure ProcedureName

    End

    Begin Definition For Function FunctionName

    End

End
```

Summary This command serves simply as a top level container for the procedure and function definition blocks.

Description This command exists primarily for historical reasons. It must be present to act as a container for the procedure and function definition blocks. In the future it may be deprecated.

2.2. CONCHAS PROCEDURE

Scope: Sierra

```
Begin Conchas Procedure ProcedureName
    Ignore Restart Cfl
    Ignore Restart Dt
    Start Time {=|are|is} start_time
    Termination Time {=|are|is} end_time
    Incremental Steps {=|are|is} numberSteps
    Termination Step {=|are|is} numberSteps
    Begin Conchas Region Regionname
    End

    Begin Run Schedule
    End

End
```

Summary This command block simply groups the commands needed to execute an analysis.

Description At the highest level, this block specifies time-stepping control parameters such as the termination time, how to treat the time step size on restart, and so forth. It also contains the region block, which is the mathematical description of the problem to be solved. The run schedule block, which controls the values of various solution parameters throughout a run, also is defined within this scope.

2.2.1. Ignore Restart Cfl

Scope: Conchas Procedure

Summary This command directs the code to ignore the maximum allowable CFL value that is in the restart file.

Description The default behavior at the beginning of a restart run is for Aero to get the current value of the maximum allowable CFL from the restart file. There are usually parameters specified in the input file that change the CFL during a run. By reading the value from the restart file, such changes will continue without user intervention. This line command allows the user to override the value in the restart file.

2.2.2. Ignore Restart Dt

Scope: Conchas Procedure

Summary This command directs the code to ignore the time step size (Δt) that is in the restart file.

Description The default behavior at the beginning of a restart run is for Aero to get the current value of the time step size from the restart file. There are usually parameters specified in the input file that change the time step size during a run. By reading the value from the restart file, such changes will continue without user intervention. This line command allows the user to override the value in the restart file.

2.2.3. Start Time

Scope: Conchas Procedure

`Start Time {=|are|is}` *`start_time`*

Parameter	Value	Default
start_time	`real`	`0`

Summary This line command specifies the starting value of the simulation time.

Description In some cases, it may be useful to start a simulation at some time other than zero. This command allows such an offset to be added to the simulation time.

2.2.4. Termination Time

Scope: Conchas Procedure

`Termination Time {=|are|is}` *`end_time`*

Parameter	Value	Default
end_time	`real`	`Real_Max`

Summary The termination time is the simulation time at which the code will stop running.

Description This optional command may be useful in time accurate simulations in order to stop the execution of the code at a given time.

2.2.5. Incremental Steps

Scope: Conchas Procedure

`Incremental Steps {=|are|is}` *`numberSteps`*

Parameter	Value	Default
numberSteps	`integer`	`Int_MAX`

Summary This line command specifies the incremental number of time steps that the code will perform.

Description This optional line command allows the user to specify that the code terminate after performing the given incremental number of time steps.

2.2.6. Termination Step

Scope: Conchas Procedure

`Termination Step {=|are|is}` *`numberSteps`*

Parameter	Value	Default
numberSteps	`integer`	`Int_MAX`

Summary This line command specifies the maximum number of time steps that the code will perform.

Description This optional line command allows the user to specify that when the time step counter is equal to the provided termination step, the code will terminate.

2.3. RUN SCHEDULE

Scope: Conchas Procedure

```
Begin Run Schedule
End
```

Summary Specifies a schedule for algorithm options such as time integration scheme, spatial order, limiter, etc. to be changed according to time step index.

Description The run schedule is a table that allows certain code parameters to change according to the time step index. The first row in the table must specify the column names. The column names can appear in any order, with the sole exception that the activation step, or iteration, must appear first. All other parameters may appear in any order. The rows are parsed as strings and then may be converted to real or integer values, as appropriate. The run schedule must contain at least one row, and it must specify an iteration to start (`its`) of 1. For example, consider the following table:

```
Begin Run Schedule
  its       step_type       step   spatial_order    limiter
  1        cfl_global_dt     5        first          foo
  3000     cfl_local_dt      10       second         edge_vl
End
```

At step 1, a CFL of 5 will be used to compute a single global time step size, and a first order advection scheme will be used. During this time, no limiter will be computed and the value specified for the limiter will be ignored. Starting at step 3000, a CFL of 10 will be used to compute a local time step size at each node. A second order advection scheme will be used with the van Leer edge limiter. The total number of time steps to be performed is specified by the `termination step` line command.

its An integer value that specifies the iteration (or time step) at which the parameters on the given row will activate. This parameter is unique in that it must appear first.

step_type If set to cfl_global_dt, then the value given as the step parameter is a CFL and a single time step size will be computed and used everywhere in the mesh. If set to cfl_local_dt, then the value given as the step parameter is a CFL and a time step size will be computed locally at a given node: this is the well-known local time-stepping convergence acceleration technique. If set to dt, then the step is the time step size itself that is fixed for the number of its specified in this row. adaptive will use temporal error control with embedded Runge-Kutta integrators. There is no default value for step_type.

step The value to be used for either the CFL or the time step size. (see step_type). There is no default for the step size.

spatial_order if the value first is given, then a simple first-order accurate scheme is to be used with no reconstruction. If second is given, then a MUSCL scheme is used with reconstructed gradients and the associated limiter to construct left and right states at an edge. highres and LOWDIS are experimental options and should not be used for production calculations. They both activate a hybrid algorithm with behavior dependent on the limiter, inviscid flux type, and hybrid sensor chosen. The default spatial_order is first_order.

limiter If the solution is first order, then this value is ignored. The use of no limiter is indicated by the string none. The allowable limiter function names are barth (Barth-Jesperson), venkat (Venkatakrishnan), va (van Albada), vl (van Leer), and minmod (Osher's min-mod). There are two types of limiters: stencil limiters and edge limiters. Generally speaking, if the mesh is fairly structured, then an edge limiter should be used; otherwise, use a stencil limiter. To specify a stencil limiter, preface the limiter function name with node_. To specify an edge limiter, preface the limiter function name with edge_. The limiter function name by itself is not allowable. For example, the edge version of van Leer's function is specified with the

string edge_vl, whereas the stencil version is specified with the string node_vl, but the unprefixed string vl is not admissible. The default limiter is node_venkat.

limiter_beta_factor This command specifies the β factor used limiter definitions for each limiter. It defaults to 0 for Barth, Van_Leer, and Van_Albada limiters. It defaults to 1 for Venkat and Osher_MinMod limiters.

dco_visc_mult If specified as non-zero, this command activates the discontinuity capturing operator, and uses the non-zero value as a multiplier on the computed artificial viscosity. The default is 0.0. A reasonable range is 0.01..100, but production runs should use 1.0.

time_scheme This command specifies the time integration algorithm. If this string is steady, then an implicit first-order backward difference (backward Euler) method is used, without the time derivative terms in the residual. If this string is bdf1, then the same first-order backward difference is used but with the time terms in the residual. bdf2 specifies the use of an implicit, second-order accurate backward difference formula. There are two explicit options: rk4, which is a fourth-order Runge-Kutta method; and feuler, which is the first-order forward Euler formula.

In addition there are three options for explicit adaptive time-stepping, based on control of local temporal error. adaptive_erk43 uses a fourth-order method with third-order error estimate, while adaptive_erk21 and adaptive_erk12 use second- and first-order methods to update the solution and approximate error, respectively. Implicit adaptive time-stepping with the DIRK method is automatically used if dirk is specified here.

controller_type This command sets the controller type used with adaptive time-stepping methods. elementary uses a common I-controller, pi_m6m1 specifies a PI controller tuned to follow stability boundaries, and PID uses a PID controller provided by Kennedy and Carpenter. If passive is specified here, then the error field will be computed but no control will be performed.

controller_target_error This sets the target error for the adaptive time-stepping methods.

controller_step_acceptance This command determines if the controller will retry steps that produce error larger than the specified target. Setting to loose will not retry steps, while strict requires that each step fall below the target error. loose is the default setting.

ramp_type If this string is increment, then the given value for the step (see above) is changed by adding the ramp to it. If this string is factor, then the value of the step is changed by multiplying it by the value of the ramp. The default ramp_type is increment.

ramp The value by which the given step is to be changed. The default ramp is 0.

max_step_attempts The maximum allowable number of attempts at a single time step.

step_max The maximum allowable value of the step. The default is to allow an arbitrarily large step.

min_nl The minimum number of nonlinear iterations to take. The default is 1.

max_nl The maximum number of nonlinear iterations to take. The default is 1.

max_pi The maximum number of point implicit iterations to take. The default is 5.

e_fix_c The value to use for the entropy fix associated with sonic points. The default is 0.1.

e_fix_u The value to use for the entropy fix associated with stagnation points. The default is 0.1.

turb_conv_order If the value in this column is first, then use first order advection for the turbulent transport equations. If the value is second, then use second order. Note that you cannot specify a higher order of advection for the turbulent transport equations than is used for the Navier-Stokes equations. The default turb_conv_order is first if spatial_order is first, and second otherwise.

hybrid_sensor For hybrid fluxes, the string diss specifies a first order scheme. The string non_diss specifies a second order scheme with no sensor, and the string mod_ducros specifies the use of the modified Ducros sensor. The default hybrid_sensor is diss.

2.4. 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 Titles1 Titles2...
    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.

2.4.1. Abscissa

Scope: Definition For Function

```
Abscissa {=|are|is} Name...
```

Parameter	Value	Default
Name	`string...`	`undefined`

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).

2.4.2. Abscissa Offset

Scope: Definition For Function

`Abscissa Offset {=|are|is}` *`Abscissa_offset`*

Parameter	Value	Default
Abscissa_offset	`real`	`undefined`

Summary Alias for X OFFSET

2.4.3. Abscissa Scale

Scope: Definition For Function

`Abscissa Scale {=|are|is}` *`Abscissa_scale`*

Parameter	Value	Default
Abscissa_scale	`real`	`undefined`

Summary Alias for X SCALE

2.4.4. At Discontinuity Evaluate To

Scope: Definition For Function

`At Discontinuity Evaluate To` *`Option`*

Parameter	Value	Default
Option	`{left|right}`	`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.

2.4.5. Column Titles

Scope: Definition For Function

`Column Titles` *`Titles`*$_1$ *`Titles`*$_2$`...`

Parameter	Value	Default
Titles	`string_1 string_2...`	`undefined`

Summary Name the columns (and also defined the expected number of columns) for Multicolumn Piecewise Linear tabular data.

2.4.6. Data File

Scope: Definition For Function

`Data File = filename [ X From Column xcol Y From Column ycol ]`

Parameter	Value	Default
filename	`string`	`undefined`

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.

2.4.7. Debug

Scope: Definition For Function

`Debug {=|are|is} Option`

Parameter	Value	Default
Option	`{off|on}`	`undefined`

Summary Prints functions to the log file.

2.4.8. Differentiate Expression

Scope: Definition For Function

`Differentiate Expression {=|are|is} Expr`

Parameter	Value	Default
Expr	`(expression)`	`undefined`

Summary Specifies the expression of derivative of evaluation expression.

2.4.9. Evaluate Expression

Scope: Definition For Function

`Evaluate Expression {=|are|is}` *`Expr`*

Parameter	Value	Default
Expr	`(expression)`	`undefined`

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) magnitude 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

2.4.10. Evaluate From

Scope: Definition For Function

`Evaluate From` *`x0`* `To` *`x1`* `By` *`Dx`*

Parameter	Value	Default
x0	`real`	`undefined`
x1	`real`	`undefined`
Dx	`real`	`undefined`

Summary Specifies the range and evaluation interval.

2.4.11. Expression Variable:

Scope: Definition For Function

`Expression Variable:` *`Expr = VarType value_var_name...`*

Parameter	Value	Default
Expr	`string`	`undefined`
VarType	`{element | element_sym_tensor | element_tensor | element_vector | face | global | nodal | nodal_sym_tensor | nodal_tensor | nodal_vector}`	`undefined`
value_var_name	`string...`	`undefined`

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

2.4.12. Expression Variable:

Scope: Definition For Function

`Expression Variable:` *`Expr`*

Parameter	Value	Default
Expr	`string`	`undefined`

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

2.4.13. Ordinate

Scope: Definition For Function

`Ordinate {=|are|is}` *`Name...`*

Parameter	Value	Default
Name	`string...`	`undefined`

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).

2.4.14. Ordinate Offset

Scope: Definition For Function

`Ordinate Offset {=|are|is}` *`Ordinate_offset`*

Parameter	Value	Default
Ordinate_offset	`real`	`undefined`

Summary Alias for Y OFFSET

2.4.15. Ordinate Scale

Scope: Definition For Function

`Ordinate Scale {=|are|is}` *`Ordinate_scale`*

Parameter	Value	Default
Ordinate_scale	`real`	`undefined`

Summary Alias for Y SCALE

2.4.16. Scale By

Scope: Definition For Function

`Scale By` *`x`*

Parameter	Value	Default
x	`real`	`undefined`

Summary Specifies a scale factor to be applied.

2.4.17. Type

Scope: Definition For Function

`Type {=|are|is}` *`Type`*

Parameter	Value	Default
Type	`{analytic|constant|multicolumn piecewise linear|piecewise analytic| piecewise constant|piecewise linear| xtable}`	`undefined`

Summary Specifies the type of function.

2.4.18. X Offset

Scope: Definition For Function

`X Offset {=|are|is}` *`X_offset`*

Parameter	Value	Default
X_offset	`real`	`undefined`

Summary Sets an offset for the x-axis

2.4.19. X Scale

Scope: Definition For Function

`X Scale {=|are|is}` *`X_scale`*

Parameter	Value	Default
X_scale	`real`	`undefined`

Summary Sets a scale factor for the x-axis

2.4.20. Y Offset

Scope: Definition For Function

`Y Offset {=|are|is}` *`Y_offset`*

Parameter	Value	Default
Y_offset	`real`	`undefined`

Summary Sets an offset for the y-axis

2.4.21. Y Scale

Scope: Definition For Function

`Y Scale {=|are|is}` *`Y_scale`*

Parameter	Value	Default
Y_scale	`real`	`undefined`

Summary Sets a scale factor for the y-axis

2.5. EXPRESSIONS

Scope: Definition For Function

```
Begin Expressions empty
    Xvalue Expr
End
```

Summary Lists the expressions for piecewise analytic function. The values should be listed one pair per line, independent variable first, with whitespace or comma as a separator.

2.5.1.

Scope: Expressions

Xvalue Expr

Parameter	Value	Default
Xvalue	`real`	`undefined`
Expr	`(expression)`	`undefined`

Summary For a piecewise analytic function, lists an x-y pair for the nth interpolation point.

2.6. 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.

2.6.1.

Scope: Values

Xyvalues...

Parameter	Value	Default
Xyvalues	`real...`	`undefined`

Summary For a piecewise linear function, lists an x-y pair for the nth interpolation point.

3. REGION

The commands that specify the mathematical models and the algorithms used to approximate them are contained in the region block. For example, the gas model, spatial discretization options, nonlinear solver parameters and boundary conditions will be found here. The flow state, which is used with boundary conditions as well as initial conditions, is defined here as well.

3.1. CONCHAS REGION

Scope: Conchas Procedure

```
Begin Conchas Region Regionname
    Disable Default Restart File
    Mesh Database Name {=|are|is} Path
    Mesh Decomposition Method {=|are|is} type
    Mesh Sequence From coarseMesh
    Surface Geometry Filename {=|are|is} SurfaceGeometryFilename
    Use Solution Steering With Interval {=|are|is} Interval
    Begin Averaging AverageName
    End

    Begin Adaptivity label
    End

    Begin Boundary Layer Data bl_name
    End

    Begin Characteristic Projection On Surface Surfacename
    End

    Begin Donor Mesh transferName
    End

    Begin Data Probe fileName
```

End

Begin Error Transport Equation *Name*

End

Begin External Mesh Output *BlockName*

End

Begin Extrapolation Boundary Condition On Surface *Surfacename*

End

Begin Flow State *stateName*

End

Begin Fsi Description

End

Begin Fixed At State Boundary Condition On Surface *Surfacename*

End

Begin Force And Moment *fileName*

End

Begin Gas Properties

End

Begin Initial Condition Block *BlockName*

End

Begin Linear Solver Options *solver_name*

End

Begin Mesh Motion *motion_name*

End

Begin Periodic *Name*

End

Begin Restart Input *BlockName*

End

Begin Restart Output *BlockName*

```
    End

    Begin Results Output BlockName
    End

    Begin Solution Options OptionsName
    End

    Begin Spatial Average Output BlockName
    End

    Begin Sponge Layer
    End

    Begin Surface Field Output Name
    End

    Begin Tangent Flow Boundary Condition On Surface Surfacename
    End

    Begin Wall Boundary Condition On Surface Surfacename
    End

End
```

Summary This command block contains the mathematical models.

Description A region is a collection of mathematical models that describes the flow problem of interest. There can be only one region in the input file.

3.1.1. Disable Default Restart File

Scope: Conchas Region

Summary Disable the default restart file creation.

Description The default behavior of aero is to create an output restart file at the end of the run, even if no output restart file is specified in the input file. This is to avoid running the code without having a way to restart the calculation. This line command disables the creation of this default restart file.

3.1.2. Mesh Database Name

Scope: Conchas Region

`Mesh Database Name {=|are|is}` *`Path`*

Parameter	Value	Default
Path	`string`	`undefined`

Summary This specifies the name of the mesh database to be used for the simulation.

Description The mesh database must be in Genesis or Exodus format. If it is not in the current working directory, then an absolute or relative path must be given. For a parallel run, the mesh must either be partitioned to the desired number of processors, or the `Mesh Decomposition Method` line command must be given.

3.1.3. Mesh Decomposition Method

Scope: Conchas Region

`Mesh Decomposition Method {=|are|is}` *`type`*

Parameter	Value	Default
type	`{geom_kway|hsfc|kway|linear|rcb|rib}`	`undefined`

Summary This specifies the type of automatic parallel decomposition to be used on the mesh file.

Description The default behavior for a parallel run is that the code expects the mesh file to already be decomposed. If this line command is present, then at startup the code will decompose the input mesh database with the given decomposition type. The decomposition types are

RCB Recursive coordinate bisection

RIB Recursive inertial bisection

HSFC Hilbert space-filling curve

KWAY A graph based approach

GEOM_KWAY A geometric variant of k-way.

Note: Currently, donor meshes do not get automatically decomposed. Also, automatically restarting the code on a different number of processors than the previous run is not supported. Generally, KWAY is a good choice if a graph-based method is desired, and RIB is a good choice if a geometric method is desired.

3.1.4. Mesh Sequence From

Scope: Conchas Region

`Mesh Sequence From` *`coarseMesh`*

Parameter	Value	Default
coarseMesh	`string`	`undefined`

Summary Interpolate solution (conserved variables) from coarse grid.

Description Used for grid sequencing from coarse to finer grids. The solution from the "coarseMesh" exodus database is interpolated onto the grid associated with the current region. The conserved variables, e.g., $(\rho, \rho\mathbf{u}, \rho\mathbf{E})$ are the only fields interpolated and must exist in the source grid, and must be named "conserved_variables". Note that these fields are automatically output to the results file: they need not be specified in a Results Output command block.

3.1.5. Surface Geometry Filename

Scope: Conchas Region

`Surface Geometry Filename {=|are|is}` *`SurfaceGeometryFilename`*

Parameter	Value	Default
SurfaceGeometryFilename	`string`	`undefined`

Summary Name of file containing the 2D or 3D CAD geometry.

Description This is the file resulting from fitting cubic or bi-cubic splines to the mesh for this run. This file is either in OpenNURBS or Exodus format. For the OpenNURBS case, it should have a .3dm extension, and is the result from running Percept's mesh_adapt command to fit the mesh with spline representations of the geometry in 2D, or the result of processing through Cubit to convert a CAD geometry and associated mesh to an OpenNURBS representation of the geometry. Percept can also be used to create a bi-cubic patch-based geometry representation of a given mesh, in which case the output of Percept is an augmented Exodus file, which is then specified here.

3.1.6. Use Solution Steering With Interval

Scope: Conchas Region

`Use Solution Steering With Interval {=|are|is}` *`Interval`*

Parameter	Value	Default
Interval	`integer`	`undefined`

Summary Activates the use of a steering file, which allows the user to change certain parameters during a run by editing the file.

Description This line command instructs the code to create a solution steering file at startup. This file contains parameters that the user can subsequently modify during the course of a solution by editing the file. The file is named `aero_steering_file` and is created in the current working directory. The given interval defines how often the file is read and written, in terms of the number of time steps.

The following example steering file illustrates the available parameters:

```
          500  Interval to check steering file
            0  Abort
            0  Write Restart
            0  Write Output
 1.79769e+308  Step Max
  0.00000e+00  Step Ramp
```

The first line allows the user to change how often the file itself is read. The second line will cause the code to gracefully abort if the value is set to anything other than 0. If the `Write Restart` parameter is set to anything other than 0 then the code will write a restart file at the next step. Similarly, `Write Output` causes the code to write the results files at the next step. `Step Max` is the maximum allowable value of the time step size, or CFL, whichever is active as the step type. `Step Ramp` is the increment or factor by which the step size is being changed.

Note: this feature is currently inactive: If this line command is present, the code will write the steering file, but it will never actually read it.

3.2. SOLUTION OPTIONS

Scope: Conchas Region

`Begin Solution Options` *`OptionsName`*

`Activate Equation` *`Equations`*

`Activate Exact Solution` *`ExactSolutionType`*

`Append Nonlinear Residual File {=|are|is}` *`Switch`*

`Apply Failed Steps`

`Coordinate System {=|are|is}` *`CoordSys`* `[ Rotation Speed {=|are|is}` *`rotationSpeed`* `]`

`Compute Exact Error Pressure`

`Disable Residual Volume Weighting`

`Deactivate Discontinuity Capturing Operator`

`Disable Derived Tangent Flow Viscous Conditions`

Eigenvalue Fix Type {=|are|is} *EigenvalueFixType*

Element Residual Type {=|are|is} *ElemResidualType*

Freeze Limiter At Nonlinear Iteration *nliter*

Freeze Limiter At Step *step*

Interface Rebalance Iterations {=|are|is} *iterations*

Interface Rebalance Target {=|are|is} *target*

Linear Solver Name {=|are|is} *linear_solver_name*

Modify Efix By Edge Factor [{=|are|is} *factor*]

Neglect Cross Term Sensitivity

Nonlinear Residual Norm Tolerance {=|are|is} *Tolerance*

Number Least Squares Gradient Iterations {=|are|is} *gradientIterations*

Parameter *key* {=|are|is} *value*

Post Process *PostProcessorType* On *IoPartList...*

Use Composite Nonlinear Residual [Including Turbulent Variables]

Use Jacobian Free Newton Krylov

Use Relative Nonlinear Residual

Use Spectral Collocation Elements With P {=|are|is} *Order*

Use Continuous Elements

Use Exact Initial Condition

Use Boundary Face Weights For Gradients

Write Exact Errors Linf To File *ExactErrorLinfFileName*

Write Exact Errors To File *ExactErrorFileName*

Artificial Boundary Layer Height {=|are|is} *height*

Dynamic Line Search

Dynamic Line Search Always Search Beginning At Iteration {=|are|is} *enable_at_iteration*

Dynamic Line Search Composite Residual Growth Max {=|are|is} *max_growth*

Dynamic Line Search Linear Measure Ceiling {=|are|is} *ceiling*

Dynamic Line Search Linear Measure Growth Max {=|are|is} *max_growth*

Dynamic Line Search Residual Growth Max {=|are|is} *max_growth*

Equilibrium Constant Calculation {=|are|is} *KeqCalc*

Gradient Method {=|are|is} *GradientMethod*

Inviscid Flux Type {=|are|is} *InviscidFluxType*

Maximum Allowable Residual {=|are|is} *maxAllowableResidual*

Minimum Cfl {=|are|is} *minimumCFL*

Minimum Local Relaxation Factor Size {=|are|is} *limit*

Minimum Timestep {=|are|is} *minimumDt*

Msw Weighting Type {=|are|is} *MswWeightingType*

Nonorthogonal Correction Type {=|are|is} *NOCType*

Omit Species Sources

Pressure Floor {=|are|is} *limit*

Print Local Relaxation Info

Reconstruct Pressure Not Temperature

Reference State {=|are|is} *state*

Residual Norm Growth Limit {=|are|is} *limit*

Residual Print Frequency {=|are|is} *freq*

Schlieren Image Height {=|are|is} *height*

Set Dual Time Cfl {=|are|is} *dual_time_cfl*

Set Dual Time Betainf {=|are|is} *dual_time_beta_inf*

Set Dual Time Max Iter {=|are|is} *dual_time_max_iter*

Set Dual Time Preconditioner Type {=|are|is} *dual_time_preconditioner_type*

Set Dual Time Tolerance {=|are|is} *dual_time_tolerance*

Solution Update Limit Factor {=|are|is} *limit*

Temperature Floor {=|are|is} *limit*

Use Limiter Pressure Smoother

Use Line Search

Use Local Relaxation

```
    Begin Turbulence Model Specification TurbSpecName

    End

End
```

Summary This command block contains algorithmic and discretization options, as well as the turbulence model specification.

Description The `Solution Options` command block allows the user to control certain aspects of the solution algorithm such as the higher-order reconstruction, the limiters, the gradient computation, and the nonlinear solver. It also controls some diagnostic information regarding the nonlinear solution, activates the post-processing of field data such as Mach number, and contains the turbulence model definition.

3.2.1. Activate Equation

Scope: Solution Options

`Activate Equation` *`Equations`*

Parameter	Value	Default
Equations	`{euler|navierstokes}`	`undefined`

Summary This line command controls if the system of equations is for inviscid or viscous flows.

Description If `euler` is specified, then an inviscid flow is modeled. The value `navierstokes` indicates that a laminar or turbulent viscous simulation is to be performed.

3.2.2. Activate Exact Solution

Scope: Solution Options

`Activate Exact Solution` *`ExactSolutionType`*

Parameter	Value	Default
ExactSolutionType	{euler_1d_sodtest | euler_1d_sodtest_contact | euler_1d_sodtest_rarefaction | euler_1d_sodtest_shock | euler_2d_box | euler_2d_box_transient | euler_2d_diamond | euler_2d_periodic_box | euler_2d_steady_isentropic_vortex | euler_3d_annular_cylinder | euler_3d_box | euler_3d_box_transient | euler_3d_cylinder | euler_3d_periodic_box | euler_piston_fsi | euler_piston_fsi_mms | isentropic_vortex_2d | isentropic_vortex_3d | navier_stokes_2d_box | navier_stokes_2d_box_periodic | navier_stokes_2d_bump | navier_stokes_3d_box | navier_stokes_3d_box_periodic | navier_stokes_3d_bump | oblique_shock_2d | prandtl_meyer_2d | shock_expansion | viscous_shock | viscous_shock_steady}	undefined

Summary This line command activate source terms determined by a manufactured solution.

Description This command is used in the verification test suite to apply appropriate source terms for a given manufactured solution. These manufactured solutions are used to verify the order of accuracy of the various algorithms in Aero.

3.2.3. Append Nonlinear Residual File

Scope: Solution Options

`Append Nonlinear Residual File {= | are | is}` *Switch*

Parameter	Value	Default
Switch	{false | off | on | true}	on

Summary This line command controls how the nonlinear residual norms are written to disk.

Description The default behavior is that the nonlinear residual norm file is appended to during a run that has been restarted, but is created or overwritten for a run that has not been restarted. This line command changes this behavior so that the file is never overwritten and always appended to.

3.2.4. Apply Failed Steps

Scope: Solution Options

Summary This option will force the solution to be updated, even when the line search fails. It is ignored if the linear search is not active.

Description The line search "fails" when the residual does not decrease below the growth limit within four reductions of the update vector. The default behavior is to throw away such an update and reduce the time step to compute a new update direction. Using this option will change the behavior to just take the small update that results in a global residual increase.

3.2.5. Coordinate System

Scope: Solution Options

`Coordinate System {=|are|is}` *`CoordSys`* `[ Rotation Speed {=|are|is}` *`rotationSpeed`* `]`

Parameter	Value	Default
CoordSys	`{cartesian|xaxi|yaxi}`	`undefined`

Summary This line command specifies the coordinate system that is used, and optionally activates solid body rotation.

Description If the optional `Rotation Speed` keyword is given, then the mesh is defined to be rotating as a solid body with the given speed about the z-axis. Currently, only `Cartesian` works in Aero, so the only reason to use this line command is to enable solid body mesh rotation. Furthermore, this mesh rotation feature is very limited in that the mesh may only rotate about the z-axis. Also, only a fixed time step may be used with mesh rotation.

3.2.6. Compute Exact Error Pressure

Scope: Solution Options

Summary This line command will enable exact errors in the pressure to be calculated and output.

3.2.7. Disable Residual Volume Weighting

Scope: Solution Options

Summary This line command eliminates the volume weighting of the residual.

Description The default behavior is to compute the nonlinear residuals weighted by the nodal volume. This line command eliminates the residual weighting, which better reflects the actual convergence of the solution impacting QOIs in viscous flow problems.

3.2.8. Deactivate Discontinuity Capturing Operator

Scope: Solution Options

Summary This line command specifies that continuous element method will be used for high-order collocation instead of a discontinuous method.

Description When this experimental option is active, aero will use a continuous high-order algorithm instead of a discontinuous one.

3.2.9. Disable Derived Tangent Flow Viscous Conditions

Scope: Solution Options

Summary This line command disables the addition of viscous conditions for tangent flow boundaries that are derived from the inviscid condition. It has only been demonstrated to be useful when applying error transport equations with two-equation turbulence models.

3.2.10. Eigenvalue Fix Type

Scope: Solution Options

`Eigenvalue Fix Type {=|are|is}` *`EigenvalueFixType`*

Parameter	Value	Default
EigenvalueFixType	`{maximum|nofix|scaled|unscaled}`	`SCALED`

Summary This line command specifies the type of eigenvalue fix to be applied.

Description For acoustic waves, the `scaled` eigenvalue fix has the form $\lambda = \frac{1}{2}\frac{\left(\lambda^2+\delta c^2\right)}{\delta c}$, where $\delta c = \varepsilon(u+c)dA$. The `unscaled` fix has the form $\lambda = \sqrt{\lambda^2 + \delta c^2}$. `nofix` does not modify the eigenvalues.

3.2.11. Element Residual Type

Scope: Solution Options

`Element Residual Type {=|are|is}` *`ElemResidualType`*

Parameter	Value	Default
ElemResidualType	`{entropy_stable_flux_based|flux_based|gradient_based}`	`undefined`

Summary This line command specifies what type of residual assembly approach will be used to assemble the high-order element residual.

Description The default type, `ENTROPY_STABLE_FLUX_BASED` is the most robust when used with the `Inviscid Flux Type = ENTROPY_PRESERVING`. The standard flux based is slightly faster and the gradient based should give approximately the same answer.

3.2.12. Freeze Limiter At Nonlinear Iteration

Scope: Solution Options

`Freeze Limiter At Nonlinear Iteration` *`nliter`*

Parameter	Value	Default
nliter	`integer`	`Int_MAX`

Summary This line command specifies the nonlinear iteration step at which the limiter is frozen within a time step.

Description For some problems, nonlinear convergence can stall because the limiter "chatters". In these situations, using a fixed limiter for the nonlinear iteration within a time step can significantly reduces the nonlinearity. Some cases benefit from freezing the limiter at iteration 0, specifically highly non-linear time-accurate simulations. The type of limiter is specified in the `Run Schedule` command block.

3.2.13. Freeze Limiter At Step

Scope: Solution Options

`Freeze Limiter At Step` *`step`*

Parameter	Value	Default
step	`integer`	`Int_MAX`

Summary This line command specifies the time step at which the limiter is frozen and no longer computed.

Description For some problems, nonlinear convergence can stall because the limiter "chatters". In these situations, using a fixed limiter field significantly reduces the nonlinearity and allows the nonlinear residual to continue to be reduced. The type of limiter is specified in the `Run Schedule` command block.

3.2.14. Interface Rebalance Iterations

Scope: Solution Options

`Interface Rebalance Iterations {=|are|is}` *`iterations`*

Parameter	Value	Default
iterations	`integer`	`5`

Summary This line command specifies the maximum number of iterations the interface rebalance algorithm will perform.

Description The STK tools partition the mesh by elements. This is good for element-based algorithms, but may result in poor partitions of nodes and edges. The interface rebalance algorithm attempts to improve parallel load balancing for node and edge-based algorithms by changing the ownership of nodes on the interprocessor interfaces. This is an iterative algorithm and this line command specifies the maximum number of iterations.

3.2.15. Interface Rebalance Target

Scope: Solution Options

`Interface Rebalance Target {=|are|is}` *`target`*

Parameter	Value	Default
target	`real`	`1.1`

Summary This line command specifies the target load imbalance ratio for the interface rebalance algorithm.

Description The STK tools partition the mesh by elements. This is good for element-based algorithms, but may result in poor partitions of nodes and edges. The interface rebalance algorithm attempts to improve parallel load balancing for node and edge-based algorithms by changing the ownership of nodes on the interprocessor interfaces. This is an iterative algorithm and this line command specifies the target load imbalance ratio, which is defined as the maximum number of locally owned nodes across all processors divided by the minimum number of locally owned nodes across all processors.

3.2.16. Linear Solver Name

Scope: Solution Options

`Linear Solver Name {=|are|is}` *`linear_solver_name`*

Parameter	Value	Default
linear_solver_name	`string`	`undefined`

Summary This line command specifies the linear solver block to use for this physics.

3.2.17. Modify Efix By Edge Factor

Scope: Solution Options

Description EXPERIMENTAL: reduce efix.

3.2.18. Neglect Cross Term Sensitivity

Scope: Solution Options

Summary This line command tells the code to use a reduced left hand side, ignoring cross terms in the Jacobian matrix.

Description When this experimental option is active, aero will ignore the cross term sensitivities in Jacobian matrix, which will speedup linear solves. Depending on the configuration, it may harm overall nonlinear convergence. It is only valid for use with spectral elements.

3.2.19. Nonlinear Residual Norm Tolerance

Scope: Solution Options

`Nonlinear Residual Norm Tolerance {=|are|is}` *`Tolerance`*

Parameter	Value	Default
Tolerance	`real`	0

Summary This line command specifies the nonlinear convergence tolerance within a time step.

Description This tolerance is used to determine when to stop the nonlinear iteration within a time step.

3.2.20. Number Least Squares Gradient Iterations

Scope: Solution Options

`Number Least Squares Gradient Iterations {=|are|is}` *`gradientIterations`*

Parameter	Value	Default
gradientIterations	`integer`	`3`

Summary This line command specifies the number of iterations to use when tt gradient method = iterativeleastsquares.

3.2.21. Parameter

Scope: Solution Options

`Parameter` *`key`* `{=|are|is}` *`value`*

Parameter	Value	Default
key	`string`	`undefined`
value	`string`	

Summary For internal/development use only - set a parameter.

Description set a parameter

3.2.22. Post Process

Scope: Solution Options

`Post Process` *`PostProcessorType`* `On` *`IoPartList...`*

Parameter	Value	Default
PostProcessorType	`{fsi_force|heatflux|mach_number|pressure_coef|q_criterion|reynolds_stress|schlieren|skin_friction|totalmass|wall_shear_stress|yplus}`	`undefined`
IoPartList	`string...`	`undefined`

Summary This line command activates the post processing of fields and that it be written to the output file.

Description If the variable `yplus` is chosen, then `wall_shear_stress` is automatically also calculated. The special part name `all_blocks` indicates that the postprocessing be performed at all locations in the mesh.

3.2.23. Use Composite Nonlinear Residual

Scope: Solution Options

Summary This line command specifies that a single nonlinear residual norm will be computed instead of separate ones for each equation.

Description The default behavior is to calculate and print separate nonlinear residual norms for each equation. If this command is given, then the nonlinear residuals of all of the components are scaled and added together to give a single norm. The turbulence model variables are excluded by default. Optionally the turbulent variables can be included in this composite nonlinear residual norm.

3.2.24. Use Jacobian Free Newton Krylov

Scope: Solution Options

Summary This line command activates the Jacobian Free Newton Krylov nonlinear solver approach for the element scheme.

Description This option activates a preconditioned Jacobian Free Newton Krylov solver approach for use with spectral elements. It is an experimental feature useful for debugging and in general should not be used for analysis. It currently is only supported for bdf1, bdf2, and dirk time integrators.

3.2.25. Use Relative Nonlinear Residual

Scope: Solution Options

Summary This line command changes the way that the nonlinear residual norm is computed.

Description The default behavior is to compute the nonlinear residual norm using absolute residuals. This line command specifies that the residual norms be scaled by their initial values. More specifically, the absolute residual for each degree of freedom is scaled by the initial absolute residual. This norm is used for both output and convergence criterion.

3.2.26. Use Spectral Collocation Elements With P

Scope: Solution Options

`Use Spectral Collocation Elements With P {=|are|is}` *`Order`*

Parameter	Value	Default
Order	`integer`	`undefined`

Summary This line command activates the spectral collocation element algorithm instead of the edge based algorithm

3.2.27. Use Continuous Elements

Scope: Solution Options

Summary This line command specifies that continuous element method will be used for high-order collocation instead of a discontinuous method.

Description When this experimental option is active, aero will use a continuous high-order algorithm instead of a discontinuous one.

3.2.28. Use Exact Initial Condition

Scope: Solution Options

Summary This line command will populate the initial condition from the exact solution.

3.2.29. Use Boundary Face Weights For Gradients

Scope: Solution Options

Summary This line command alters the method used to calculate the nodal gradients at the boundaries.

Description When computing the nodal gradients at the boundaries, this command specifies that the fields be sampled at the center of the subcontrol faces, instead of just grabbing the nearest node value, as is done in the interior. Note: this is currently only supported by the Green-Gauss gradient. This feature is experimental and may help improve gradient calculations at the boundary in some circumstances.

3.2.30. Write Exact Errors Linf To File

Scope: Solution Options

`Write Exact Errors Linf To File` *`ExactErrorLinfFileName`*

Parameter	Value	Default
ExactErrorLinfFileName	`string`	`undefined`

Summary this line command results in a text file that contains the l-infinity norm of the exact solution errors for all primitive variables.

Description This command is used in the verification test suite to apply appropriate source terms for a given manufactured solution. These manufactured solutions are used to verify the order of accuracy of the various algorithms in Aero.

3.2.31. Write Exact Errors To File

Scope: Solution Options

`Write Exact Errors To File` *`ExactErrorFileName`*

Parameter	Value	Default
ExactErrorFileName	`string`	`undefined`

Summary this line command results in a text file that contains the exact solution errors for all primitive variables.

Description This command is used in the verification test suite to apply appropriate source terms for a given manufactured solution. These manufactured solutions are used to verify the order of accuracy of the various algorithms in Aero.

3.2.32. Artificial Boundary Layer Height

Scope: Solution Options

`Artificial Boundary Layer Height {=|are|is}` *`height`*

Parameter	Value	Default
height	`real`	`-1`

Summary This line command specifies height of an artificial boundary layer used in the initial conditions.

Description For more complex problems it may be beneficial to set an artificial boundary layer in the initial condition, which will transition from the conditions set for the mesh block to a no slip wall with adiabatic wall temperature near any viscous wall boundaries. This line command sets the lengthscale of the transition from the set conditions to the wall conditions.

3.2.33. Dynamic Line Search

Scope: Solution Options

Summary This command specifies that the line search solution strategy is to be used, with a heuristic that determines the local necessity of the line search, activating and deactivating it automatically.

Description This line command activates a nonlinear solution strategy that uses multiple residual evaluations to search for a global underrelaxation factor that will limit the solution update at the end of a nonlinear solution step. This factor is computed in such a way as to attempt to prevent the composite nonlinear residual norm from increasing more than a certain factor from one nonlinear step to the next. See the command `residual norm growth limit`. This global relaxation strategy can be used in conjunction with the `use local relaxation` command: both strategies may be used in the same run.

3.2.34. Dynamic Line Search Always Search Beginning At Iteration

Scope: Solution Options

`Dynamic Line Search Always Search Beginning At Iteration {=|are|is}` *`enable_at_iteration`*

Parameter	Value	Default
enable_at_iteration	`integer`	`Int_MAX`

Summary This command specifies an iteration to always turn on the line search.

Description When the dynamic line search is on, it may be desireable to have an iteration at which you always want to enable line search. E.g. You want to further attempt to drive down the residuals. By default, there is no iteration at which the line search will always be active.

3.2.35. Dynamic Line Search Composite Residual Growth Max

Scope: Solution Options

`Dynamic Line Search Composite Residual Growth Max {=|are|is}` *`max_growth`*

Parameter	Value	Default
max_growth	`real`	`1e-3`

Summary This command specifies the largest allowable rate of increase of the composite residual measure as used by the dynamic line search.

Description This command specifies the largest allowable rate of increase of the composite residual measure as used by the dynamic line search. The composite residual option must be specified for this to be used. If unspecified the default value is 1.1.

3.2.36. Dynamic Line Search Linear Measure Ceiling

Scope: Solution Options

`Dynamic Line Search Linear Measure Ceiling {=|are|is}` *`ceiling`*

Parameter	Value	Default
ceiling	`real`	`1e-3`

Summary This command specifies the largest allowable value of the linear solve measure used in activating the dynamic line search.

Description This command specifies the largest allowable value of the linear solve measure used in activating the dynamic line search. For the element algorithm, this refers to the linear iteration count, whereas for the edge algorithm this refers to the linear residual.

If unspecified the default value is 10.

3.2.37. Dynamic Line Search Linear Measure Growth Max

Scope: Solution Options

`Dynamic Line Search Linear Measure Growth Max {=|are|is}` *`max_growth`*

Parameter	Value	Default
max_growth	`real`	`1e-3`

Summary This command specifies the largest allowable rate of increase of the linear solve measure as used by the dynamic line search.

Description This command specifies the largest allowable rate of increase of the linear solve measure as used by the dynamic line search. If unspecified the default value is 1.1.

3.2.38. Dynamic Line Search Residual Growth Max

Scope: Solution Options

`Dynamic Line Search Residual Growth Max {=|are|is}` *`max_growth`*

Parameter	Value	Default
max_growth	`real`	`1e-3`

Summary This command specifies the largest allowable rate of increase of the residual measure as used by the dynamic line search.

Description This command specifies the largest allowable rate of increase of the residual measure as used by the dynamic line search. If unspecified the default value is 1.1.

3.2.39. Equilibrium Constant Calculation

Scope: Solution Options

`Equilibrium Constant Calculation {=|are|is}` *`KeqCalc`*

Parameter	Value	Default
KeqCalc	`{nasa_grc|park_90}`	`PARK_90`

Summary this line command specifies the equilibrium constant type to be used for compute backward reaction rates.

Description There are different equilibrium constant types that can be used to compute backward reaction rates during the evaluations of the species source term. Though the default is `park_90`, `nasa_grc` is the recommended option.

3.2.40. Gradient Method

Scope: Solution Options

`Gradient Method {=|are|is}` *`GradientMethod`*

Parameter	Value	Default
GradientMethod	`{elementaveraged|greengauss|iterativeleastsquares|leastsquares|secondorderleastsquares|secondorderweightedleastsquares|weightedleastsquares}`	`GREENGAUSS`

Summary This line command specifies the nodal gradient calculation method to use.

Description The method `weightedleastsquares` uses inverse distance weighting to compute nodal gradients and is the most reliable for flows over curved geometries. The method `leastsquares` indicates unweighted least squares gradients and is not appropriate on curved geometries. Finally, the choice `elementaveraged` results in a volume-weighted average of nodal gradients in each element based on the shape function of the element.

3.2.41. Inviscid Flux Type

Scope: Solution Options

`Inviscid Flux Type {=|are|is}` *`InviscidFluxType`*

Parameter	Value	Default
InviscidFluxType	`{cpd_msw|entropy_preserving|honein_moin|kinetic_energy_preserving|msw|roe}`	`ROE`

Summary This line command specifies the inviscid flux scheme to use.

Description The choices `cpd_msw` and `msw` should give the same results. They are different implementations of the Modified Steger-Warming flux function. The faster of the two implementations is `msw`. `honein_moin`, `kinetic_energy_preserving`, and `entropy_preserving` are different specifications of the nondissipative flux part used in a Roe-type flux function.
Currently `roe` supports inviscid, laminar, and turbulent ideal gases, `msw` and `cpd_msw` support inviscid and laminar reacting thermochemistry and inviscid, laminar, and turbulent ideal gases. The choices `kinetic_energy_preserving`, `entropy preserving`, and `honein_moin` work with hybrid fluxes for laminar and turbulent ideal gases.

3.2.42. Maximum Allowable Residual

Scope: Solution Options

`Maximum Allowable Residual {=|are|is}` *`maxAllowableResidual`*

Parameter	Value	Default
maxAllowableResidual	`real`	`Real_MAX`

Summary If the final residual evaluation of any nonlinear step exceeds this value, then the code will write to file and exit with an error.

3.2.43. Minimum Cfl

Scope: Solution Options

`Minimum Cfl {=|are|is}` *`minimumCFL`*

Parameter	Value	Default
minimumCFL	`real`	`0.01`

Summary This line command specifies the minimum acceptable CFL in the nonlinear solution update procedure. It applies only when the time integration increment type is the CFL and the line search is active.

Description When the line search fails, the CFL is reduced unless `apply failed steps` has been specified. This option specifies the minimum value to which the CFL can be reduced. If the line search fails when this minimum CFL has been reached, then the code will write data and abort.

3.2.44. Minimum Local Relaxation Factor Size

Scope: Solution Options

`Minimum Local Relaxation Factor Size {=|are|is}` *`limit`*

Parameter	Value	Default
limit	`real`	`1e-3`

Summary When using local relaxation, this parameter is the minimum allowable value for the fraction of the timestep to take.

Description The local relaxation method works by multiplying the solution increment by a locally computed fraction. If this fraction is 1, then the entire step is taken. If zero, then the solution at that location is not changed. This line command specifies the minimum allowable value for this fraction.

3.2.45. Minimum Timestep

Scope: Solution Options

`Minimum Timestep {=|are|is}` *`minimumDt`*

Parameter	Value	Default
minimumDt	`real`	`1e-10`

Summary This line command specifies the minimum acceptable timestep in the nonlinear solution update procedure. It applies only when the time integration increment type is the timestep and the line search is active.

Description When the line search fails, the timestep is reduced unless `apply failed steps` has been specified. This option specifies the minimum value to which the timestep can be reduced. If the line search fails when this minimum timestep has been reached, then the code will write data and abort.

3.2.46. Msw Weighting Type

Scope: Solution Options

`Msw Weighting Type {=|are|is}` *`MswWeightingType`*

Parameter	Value	Default
MswWeightingType	`{classic_sw|modified_sw|pressure_weighted_sw}`	`pressure_weighted_sw`

Summary This line command specifies the weighting type to be used with the modified Steger-Warming scheme.

3.2.47. Nonorthogonal Correction Type

Scope: Solution Options

`Nonorthogonal Correction Type {=|are|is}` *`NOCType`*

Parameter	Value	Default
NOCType	`{minimum|orthogonal|overrelaxed|pure_nodal}`	`MINIMUM`

Summary Specify the type of non-orthogonal correction to use.

Description When computing the gradients at the cell faces for the viscous flux assembly, the edge-directed gradient is augmented by additional terms to correct for non-orthogonality of the cell face and its associated edge. The details of these choices are described in the Aero theory manual.

3.2.48. Omit Species Sources

Scope: Solution Options

Summary this line command sets the reaction source terms in the thermochemical gas model equations to zero.

Description This command is intended primarily for debugging purposes when it is suspected that the source terms associated with the chemistry may be causing numerical difficulties.

3.2.49. Pressure Floor

Scope: Solution Options

`Pressure Floor {=|are|is}` *`limit`*

Parameter	Value	Default
limit	`real`	`10`

Summary When using local relaxation, this parameter is the minimum allowable value for the pressure.

Description The local relaxation method works by multiplying the solution increment by a locally computed fraction. This parameter influences that fraction by setting the minimum allowable value of the pressure, and requiring that the fraction of the step that is taken not reduce the pressure below this value.

3.2.50. Print Local Relaxation Info

Scope: Solution Options

Summary When using local relaxation, print statistics to screen such as the minimum relaxation factor and the number of nodes limited.

Description This information might be helpful to diagnose a failing run and guide the selection of some of the local relaxation parameters. It is not collected and printed by default because it requires global communication to construct and is therefore somewhat expensive.

3.2.51. Reconstruct Pressure Not Temperature

Scope: Solution Options

Summary This line command changes the default behavior so that pressure is extrapolated and limited instead of temperature. this may be beneficial for some flows.

3.2.52. Reference State

Scope: Solution Options

`Reference State {=|are|is}` *`state`*

Parameter	Value	Default
state	`string`	`none`

Summary This line command sets the reference state to use for nondimensionalizing certain postprocessors.

Description Using this line command will set the scaling for degrees of freedom to use a specified flow state. This is only used when computing skin friction and pressure coefficient, or composite residual.

3.2.53. Residual Norm Growth Limit

Scope: Solution Options

`Residual Norm Growth Limit {=|are|is}` *`limit`*

Parameter	Value	Default
limit	`real`	`1`

Summary This line command is used with the line search and specifies that the ratio of the new residual norm to the old residual norm has to be less than the given value.

Description The line search works by computing a single underrelaxation parameter to be applied at the end of the solution step. This underrelaxation parameter is computed in such a way as to attempt to prevent the composite nonlinear residual norm from increasing more than the limit specified by this line command from one step to the next.

3.2.54. Residual Print Frequency

Scope: Solution Options

`Residual Print Frequency {=|are|is}` *`freq`*

Parameter	Value	Default
freq	`integer`	`10`

Summary This line command specifies the frequency that the L^2 norms of the nonlinear residuals are written to the file named 'residuals.txt'.

Description The nonlinear residual norms provide information from which a person can judge how close the solution is to steady-state. This line command allows the user to control how often these norms are written to the text file named 'residuals.txt'.

3.2.55. Schlieren Image Height

Scope: Solution Options

`Schlieren Image Height {=|are|is}` *`height`*

Parameter	Value	Default
height	`real`	`-1`

Summary This line command specifies height of the image plane for the simulated schlieren postprocessor.

Description A simulated Schlieren image of the flowfield is generated and written to the exodus file. This feature currently only works in two dimensions. This line command sets the distance between the image plane and the mesh.

3.2.56. Set Dual Time Cfl

Scope: Solution Options

`Set Dual Time Cfl {=|are|is}` *`dual_time_cfl`*

Parameter	Value	Default
dual_time_cfl	`real`	`1.0`

Summary This parameter sets the dual time CFL number.

Description The dual time CFL number determines the dual time step used in the subiterations. It acts as a relaxation parameter and should be order 1. Increasing very much above 1 risks instability, and setting below 1 will likely stabilize and slow down convergence.

Preconditioned dual timestepping (pdt_bdf1 or pdt_bdf2) must be chosen as the time scheme in order for this option to be set.

3.2.57. Set Dual Time Betainf

Scope: Solution Options

`Set Dual Time Betainf {=|are|is}` *`dual_time_beta_inf`*

Parameter	Value	Default
dual_time_beta_inf	`real`	`1`

Summary This sets the beta_inf parameter in the artificial compressibility preconditioner.

Description This is a pressure scaling parameter in the Turkel, 1987 preconditioner. Beta inf is used to compute the pressure scale. It is computed here with the method of Unrau and Zing, 1995.

Preconditioned dual timestepping must be chosen as the time scheme in order for this option to be set. The preconditioner type must be set to artificial_compressibility.

3.2.58. Set Dual Time Max Iter

Scope: Solution Options

`Set Dual Time Max Iter {=|are|is}` *`dual_time_max_iter`*

Parameter	Value	Default
dual_time_max_iter	`integer`	`1000`

Summary This parameter sets the maximum iteration count in dual time.

Description This is the maximum allowable dual time iterations per physical time step. A runtime warning will be thrown if the error tolerance is not met before reaching this number of iterations.

Preconditioned dual timestepping (pdt_bdf1 or pdt_bdf2) must be chosen as the time scheme in order for this option to be set.

3.2.59. Set Dual Time Preconditioner Type

Scope: Solution Options

`Set Dual Time Preconditioner Type {=|are|is}`
`dual_time_preconditioner_type`

Parameter	Value	Default
dual_time_preconditioner_type	`{artificial_compressibility|identity}`	`IDENTITY`

Summary This parameter sets the type of preconditioner used in dual time.

Description This determines the type of preconditioner used in dual time. Setting to Identity will use no preconditioning matrix. Other options are named by the author and year of their invention. artificial_compressibility scales the time derivative of pressure to precondition the acoustic wave speed.

Preconditioned dual timestepping (pdt_bdf1 or pdt_bdf2) must be chosen as the time scheme in order for this option to be set.

3.2.60. Set Dual Time Tolerance

Scope: Solution Options

`Set Dual Time Tolerance {=|are|is}` *`dual_time_tolerance`*

Parameter	Value	Default
dual_time_tolerance	`real`	`0.000001`

Summary This parameter sets the dual time error tolerance.

Description The dual time tolerance is the global error required for convergence in a physical time step. The global error is computed as the square root of the sum of the squared relative errors.

Preconditioned dual timestepping (pdt_bdf1 or pdt_bdf2) must be chosen as the time scheme in order for this option to be set.

3.2.61. Solution Update Limit Factor

Scope: Solution Options

`Solution Update Limit Factor {=|are|is}` *`limit`*

Parameter	Value	Default
limit	`real`	`1`

Summary This line command is used with local relaxation and limits the solution update so that the magnitude of dP/P and dT/T are less than this factor.

Description This line command is used to locally limit the rate of change in the temperature and the pressure. It may be useful, for example, if there is a rapid expansion of the flow around a corner and the temperature is getting nonphysically low.

3.2.62. Temperature Floor

Scope: Solution Options

`Temperature Floor {=|are|is}` *`limit`*

Parameter	Value	Default
limit	`real`	5

Summary When using local relaxation, this parameter is the minimum allowable value for the temperature.

Description The local relaxation method works by multiplying the solution increment by a locally computed fraction. This parameter influences that fraction by setting the minimum allowable value of the temperature, and requiring that the fraction of the step that is taken not reduce the pressure below this value.

3.2.63. Use Limiter Pressure Smoother

Scope: Solution Options

Summary The limiter pressure smoother adds an additional buffer of reduced spatial order around large-magnitude pressure gradients.

Description This pressure smoother is intended to add addition dissipation around strong shock waves in order to increase the robustness of numerical method and reduce the overall time to convergence. The smoother is based on the pressure gradient across the edge and has the form of a Gaussian function that has a value of 1 when the pressure gradient is zero and has a value of 0 for very large pressure gradients. The smoother modifies the calculated gradient of all of the limited quantities in the inviscid flux calculation.

3.2.64. Use Line Search

Scope: Solution Options

Summary This line command alters the nonlinear solution strategy by using a global underrelaxation factor that is applied during the nonlinear solution update.

Description This line command activates a nonlinear solution strategy that uses multiple residual evaluations to search for a global underrelaxation factor that will limit the solution update at the end of a nonlinear solution step. This factor is computed in such a way as to attempt to prevent the composite nonlinear residual norm from increasing more than a certain factor from one nonlinear step to the next. See the command `residual norm growth limit`. This global relaxation strategy can be used in conjunction with the `use local relaxation` command: both strategies may be used in the same run.

3.2.65. Use Local Relaxation

Scope: Solution Options

Summary This line command alters the nonlinear solution strategy by using a local relaxation algorithm.

Description This command can add robustness to a calculation. it specifies the use of a locally computed relaxation factor that is used during the solution update at the end of a nonlinear step. This relaxation factor is used to constrain the solution update in such a way as to keep the temperature and pressure from changing too fast and from getting too small. See also `solution update limit factor`, `minimum local relaxation factor size`, `pressure floor`, `temperature floor`, `print local relaxation info`. This local relaxation strategy can be used in conjunction with the `line search`:both strategies can be used in the same run.

3.3. TURBULENCE MODEL SPECIFICATION

Scope: Solution Options

```
Begin Turbulence Model Specification TurbSpecName
    Cev CEVParams {=|are|is} Value
    Des Near Wall Region Size {=|are|is} Value
    Des Lengthscale Directions {=|are|is} Values...
    K_Epsilon KEpsilonParams {=|are|is} Value
    Log Law C Constant {=|are|is} Value
    Log Law Kappa Constant {=|are|is} Value
    Log Law Yplus Limit {=|are|is} Value
    Sst SSTParams {=|are|is} Value
    Turbulence Model {=|are|is} TurbModel
    Turbulent Prandtl Number {=|are|is} Value
```

```
    Clip Turbulence Variables

    Des Grid Length Multiplier {=|are|is} Value

    Omit Production Divu

    Omit Turbulent Sources

    Production To Destruction Ratio {=|are|is} Value

    Use Des

    Use Gradients At Startup Hack

    Wall Distance Cutoff Value {=|are|is} Value

    Wall Distance Far Field Value {=|are|is} Value

End
```

Summary This command block activates the turbulence model and specifies its type as well as associated parameters.

Description Aero supports a one equation Spalart-Allmaras turbulence model, as well as $k - \varepsilon$ and $k - \omega$. The details of these models are in the Aero theory manual. All three models include an option for hybrid RANS/DES

3.3.1. Cev

Scope: Turbulence Model Specification

```
Cev CEVParams {=|are|is} Value
```

Parameter	Value	Default
CEVParams	{blend_end | blend_max | blend_min | blend_start | c_1 | c_2 | c_3 | c_4 | c_6 | c_7 | c_mu}	undefined
Value	real	undefined

Summary This line command specifies model parameters that are unique to the Cubic Eddy Viscosity Model of Craft et al.

Description The CEV model is experimental and not suitable for production use.

3.3.2. Des Near Wall Region Size

Scope: Turbulence Model Specification

`Des Near Wall Region Size {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`1e-6`

Summary For the k-epsilon hybrid RANS-LES model, this parameter specifies the size of the near wall region throughout which the scheme is forced to function as a RANS model.

3.3.3. Des Lengthscale Directions

Scope: Turbulence Model Specification

`Des Lengthscale Directions {=|are|is}` *`Values...`*

Parameter	Value	Default
Values	`integer...`	`undefined`

Summary List of flow directions used to evaluate the DES length scale used for RANS/LES switch.

Description The default behavior is to consider all three spatial directions when computing the DES length scale. Some situations may present preferred directions to compute this length scale, e.g., three dimensional calculations of nominally two dimensional flows.

3.3.4. K_Epsilon

Scope: Turbulence Model Specification

`K_Epsilon` *`KEpsilonParams`* `{=|are|is}` *`Value`*

Parameter	Value	Default
KEpsilonParams	`{c_1|c_2|c_mu|sigma_eps|sigma_k|use_cev}`	`undefined`
Value	`real`	`undefined`

Summary This line command specifies model parameters that are unique to the $k-\varepsilon$ model.

Description The default values for these parameters are standard and recommended. Only expert users should consider changing them.

3.3.5. Log Law C Constant

Scope: Turbulence Model Specification

`Log Law C Constant {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`5.1`

Summary Value of log law offset constant, C in equation U+=1/kappa*ln(y+)+C

3.3.6. Log Law Kappa Constant

Scope: Turbulence Model Specification

`Log Law Kappa Constant {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`0.41`

Summary Value of log law slope constant, /kappa in equation U+=1/kappa*ln(y+)+C

3.3.7. Log Law Yplus Limit

Scope: Turbulence Model Specification

`Log Law Yplus Limit {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`11.63`

Summary This line command sets the limit of the log law applicability in yplus.

Description This command is only used for wall function calculations. If the wall spacing is below this value, then the viscous sublayer behavior is assumed.

3.3.8. Sst

Scope: Turbulence Model Specification

`Sst` *`SSTParams`* `{=|are|is}` *`Value`*

Parameter	Value	Default
SSTParams	`{a_1|beta_1|beta_2|beta_star|cdes|pk_to_dk|sigma_k_1|sigma_k_2|sigma_w_1|sigma_w_2}`	`undefined`
Value	`real`	`undefined`

Summary This line command specifies model parameters that are unique to SST.

Description The default values for these parameters are standard and recommended. Only expert users should consider changing them.

3.3.9. Turbulence Model

Scope: Turbulence Model Specification

`Turbulence Model {=|are|is}` *`TurbModel`*

Parameter	Value	Default
TurbModel	`{cev|k_epsilon|k_g|laminar|sa|sst|sst_sas_2005|sst_sas_2007}`	`undefined`

Summary this line command determines the type of turbulence model to be used.

Description The SST k-ω model is specified by the type `sst`, the k-ε model by the type `k_epsilon`, and the Spalart-Allmaras by the type `SA`. The specific forms of these models are given in the Aero theory manual.

3.3.10. Turbulent Prandtl Number

Scope: Turbulence Model Specification

`Turbulent Prandtl Number {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`1`

Summary This line command sets the Turbulent Prandtl number.

3.3.11. Clip Turbulence Variables

Scope: Turbulence Model Specification

Summary This line command may increase robustness by preventing the values for the primitive turbulence model variables from falling below a minimum value.

3.3.12. Des Grid Length Multiplier

Scope: Turbulence Model Specification

`Des Grid Length Multiplier {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`0.65`

Summary Specify factor by which the grid scale for DES is multiplied. This scales the length scale used for switch and dissipation terms for DES.

3.3.13. Omit Production Divu

Scope: Turbulence Model Specification

Summary Specify that the velocity divergence term is omitted from production of turbulent kinetic energy.

Description This line command only applies to the two equation turbulence models. In certain situations, expert users may desire to deactivate this production term.

3.3.14. Omit Turbulent Sources

Scope: Turbulence Model Specification

Summary This line command sets the source terms associated with the turbulence model equations to zero.

Description This command is intended primarily for debugging purposes when it is suspected that the source terms associated with the turbulence model may be causing numerical difficulties.

3.3.15. Production To Destruction Ratio

Scope: Turbulence Model Specification

`Production To Destruction Ratio {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	20

Summary Specify the maximum allowable ration of turbulent production to destruction that is allowed to occur in the source term calculation.

3.3.16. Use Des

Scope: Turbulence Model Specification

Summary Activates Detached Eddy Simulation (DES) or hybrid RANS-LES model.

Description When used with `sa` or `sst` models, this activates DES. When used with the `k_epsilon` model, this activates a hybrid RANS-LES model.

3.3.17. Use Gradients At Startup Hack

Scope: Turbulence Model Specification

Summary This is a backwards compatibility issue.

Description Aero used to do the clipping and gradient calculation in a different order and at the beginning of the timestep. After changing this, if a user has an old result file that they want to use for an initial condition, the code is not robust to making such a switch in the middle of a solution. This switch forces the computation of the gradients before the solution is clipped and uses these gradients in the source term for the SST model, for example. The gradients are again computed after the clipping occurs. This only happens at the first time step. This command is deprecated and will be eliminated in a future release.

3.3.18. Wall Distance Cutoff Value

Scope: Turbulence Model Specification

`Wall Distance Cutoff Value {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`0`

Summary The wall distance will be computed for all nodes that are within this cutoff distance from the walls. The default value of o specifies that the wall distance will be computed everywhere.

Description This line command can significantly reduce the CPU time required to calculate the nearest distance to the wall, which is needed by the turbulence model.

3.3.19. Wall Distance Far Field Value

Scope: Turbulence Model Specification

`Wall Distance Far Field Value {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary All nodes that are further away from the walls than the cutoff value will be assigned this far field value. By default, it is equal to the cutoff value.

Description This line command can significantly reduce the CPU time required to calculate the nearest distance to the wall, which is needed by the turbulence model.

3.4. FLOW STATE

Scope: Conchas Region

```
Begin Flow State stateName
    Direction {=|are|is} Values...
    Use File fileName
    Use Donor Mesh donorMeshName
    Use Exact Solution
    Use Space Function tableName In The Direction Direction
    Use Time Function tableName
    Density {=|are|is} Value
    Direction Of Rotation Axis {=|are|is} vector...
    Length Scale {=|are|is} Value
    Mach Number {=|are|is} Value
    Massfracs {=|are|is} Values...
    Point On Rotation Axis {=|are|is} vector...
    Pressure {=|are|is} Value
    Reynolds Length Scale {=|are|is} Value
    Reynolds Number {=|are|is} Value
    Rotation Speed {=|are|is} omega
    Temperature {=|are|is} Value
    Turbulence Intensity {=|are|is} Value
    Turbulent Dissipation {=|are|is} Value
    Turbulent Kinetic Energy {=|are|is} Value
    Turbulent Viscosity Ratio {=|are|is} Value
    Velocity {=|are|is} Value
End
```

Summary Defines a flow state specification that can be used with initial and boundary conditions.

Description A flow state must be used with a boundary or initial condition. This command block specifies a set of variables to define the state of the gas. Boundary and initial condition blocks use a flow state by name in order to define their needed conditions. Multiple boundary and initial conditions may use the same block. However, the converse is not true: a given boundary or initial condition block can only use a single flow state.

There are many ways to specify the state of a gas. Care must be taken in order to do so consistently, however. Typically, for an ideal gas the Mach number, direction, pressure, and temperature are given in order to specify the flow state. The code attempts to detect that the state has been specified consistently. For example, that only two thermodynamic variables are specified for an ideal gas, and either Mach number or speed is given, but not both.

3.4.1. Direction

Scope: Flow State

`Direction {=|are|is}` *`Values...`*

Parameter	Value	Default
Values	`real...`	`undefined`

Summary This line command specifies the direction of the gas as a list of direction cosines.

Description The number of flow direction cosines must match the spatial dimension. This line command must specify a unit vector in the desired direction of flow.

3.4.2. Use File

Scope: Flow State

`Use File` *`fileName`*

Parameter	Value	Default
fileName	`string`	`undefined`

Summary This line command specifies the name of an ASCII text file from which the state will be read.

Description This line command allows for a crude way to get a spatially varying gas state definition by providing field values at a set of nodes. The file must be present in the current working directory and has the following format:

```
id temperature  pressure turbulent_kinetic_energy turbulent_dissipation
i  t_i          p_i      tke_i                    w_i          v_1_i v_
```

The first line in the file consists of the column titles. The subsequent lines are the state fields, one row per mesh node. If the flow state is to be used with a boundary condition, then the file must specify a value at each node in a sideset. Similarly, if the flow state is going to be used with an initial condition, then the file must specify a value at each node in the element block associated with the initial condition. The `id` column must be the first column and contains the global id of the node. The other column names can appear in any order. For a laminar case, the `turbulent_kinetic_energy` and `turbulent_dissipation` columns must be omitted. The number of velocity components must match the spatial dimension of the flow and must be given in the order x, y and then z, if the flow is three-dimensional. For example, the following file contents specifies the flow state at nodes 7, 333, and 5000 of a laminar flow:

```
id      pressure   velocity      temperature
7       101325      100   0   0     300
333     101325      80   10   0     310
5000    101325      110   0  -50    300
```

3.4.3. Use Donor Mesh

Scope: Flow State

`Use Donor Mesh` *donorMeshName*

Parameter	Value	Default
donorMeshName	`string`	`undefined`

Summary This line command specifies that the the flow state is populated from the mesh described in the Donor Mesh command block named donorMeshName.

Description The use of a donor mesh is currently the only mechanism for defining a flow state that varies in more than one dimension (e.g. x and y or x and t). A `Donor Mesh` command block of the given name must exist at the region scope. See the definition for the `Donor Mesh` for more details.

3.4.4. Use Exact Solution

Scope: Flow State

Summary This line command specifies that the the flow state is populated from the MMS exact solution. It should only be used with verification problems.

Description This command is used in conjunction with `Activate Exact Solution` to specify boundaries that vary in time and space.

3.4.5. Use Space Function

Scope: Flow State

`Use Space Function` *`tableName`* `In The` *`Direction`* `Direction`

Parameter	Value	Default
tableName	`string`	`undefined`
Direction	`{x	y	z}`	`undefined`

Summary This line command specifies that the flow state is varying in one spatial dimension and constant in time. There must be a space function table defined at the domain scope with the given tableName.

Description The function table, which is defined at the domain scope via the Definition for Function command block, must have the proper number of columns. The columns can appear in any order, but must be named direction, "pressure", "temperature", "velocity_0", "velocity_1", (if the problem is three-dimensional, it must also have "velocity_2"). The parameter direction must be "x", "y", or "z". If there is a turbulence model, then there must also be two more columns named "tke" and "sdr".

For example, the following table function definition (which must appear at the domain scope) specifies a space function which varies in the y coordinate direction for a three-dimensional turbulent flow

```
begin definition for function speedup
   type is multicolumn piecewise linear
   column titles y pressure temperature  velocity_0 velocity_1 velocity
   begin values
      -100    71322.717 287.62  204.0043   0  0  6.242666e-2 3.017646e4
      0       71322.717 287.62  304.0043   0  0  6.242666e-2 3.017646e4
      1000    71322.717 287.62  404.0043   0  0  6.242666e-2 3.017646e4
   end
end
```

To use this function in the flow state, include the line command `Use space function speedup`.

Note: table functions are not currently supported for the Spalart-Allmaras turbulence model.

3.4.6. Use Time Function

Scope: Flow State

`Use Time Function` *`tableName`*

Parameter	Value	Default
tableName	`string`	`undefined`

Summary This line command specifies that the flow state is time-varying and spatially constant. There must be a time function table defined at the domain scope with the given tableName.

Description The function table, which is defined at the domain scope via the Definition for Function command block, must have the proper number of columns. The columns can appear in any order, but must be named "time", "pressure", "temperature", "velocity_0", "velocity_1", (if the problem is three-dimensional, it must also have "velocity_2"). If there is a turbulence model, then there must also be two more columns named "tke" and "sdr".

For example, the following table function definition (which must appear at the domain scope) specifies a time function for a two-dimensional laminar or inviscid flow:

```
begin definition for function inflow
  type is multicolumn piecewise linear
  column titles time pressure temperature  velocity_0 velocity_1
  begin values
     0       71429   300       900      0
     0.5     71429   300       1000     0
     2.0     71429   300       1000     0
  end
end
```

To use this function in the flow state, include the line command `Use time function inflow`.

Note: table functions are not currently supported for the Spalart-Allmaras turbulence model.

3.4.7. Density

Scope: Flow State

`Density {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies the value of the density.

Description For an ideal gas without any species transport, this input quantity is simply the gas density. For species transport, this input quantity represents the mixture density, and in

this case the mass fractions must also be specified in the flow state. Then the density of each species is computed according to

$$\rho_s = \rho y_s$$

where ρ is the mixture density and y_s is the mass fraction for species s.

3.4.8. Direction Of Rotation Axis

Scope: Flow State

`Direction Of Rotation Axis {=|are|is}` *`vector...`*

Parameter	Value	Default
vector	`real...`	`undefined`

Summary A no slip wall can rotate. This line command specifies the direction of the axis about which such a wall rotates.

Description An arbitrary axis in 3D is described by a point on the axis and a direction. This line command specifies the direction of the axis, and cannot be specified by itself. The rotation speed and a point on the axis must also be specified. The wall boundary condition will compute the vector-valued velocity from the given rotation speed and axis information, and enforce that value locally. For a 2D simulation, this line command is ignored and the axis is assumed to be in the positive Z direction.

3.4.9. Length Scale

Scope: Flow State

`Length Scale {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary Physical length scale that is used to estimate the eddy length scale and subsequently the dissipation rate.

Description Note: this line command is not currently active.

3.4.10. Mach Number

Scope: Flow State

`Mach Number {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies the Mach number. Either the Mach number or the velocity magnitude must be specified, but not both.

3.4.11. Massfracs

Scope: Flow State

`Massfracs {=|are|is}` *`Values...`*

Parameter	Value	Default
Values	`real...`	`undefined`

Summary For flows involving species transport, this line command specifies the mass fractions.

Description For species transport, this line command specifies the mass fractions of each species. The mass fractions must sum to one, that is

$$\sum_{s=1}^{S} y_s = 1$$

where y_s is the mass fraction for species s. The density of each species is computed according to

$$\rho_s = \rho y_s$$

where ρ is the mixture density.

3.4.12. Point On Rotation Axis

Scope: Flow State

`Point On Rotation Axis {=|are|is}` *`vector...`*

Parameter	Value	Default
vector	`real...`	`undefined`

Summary A no slip wall can rotate. This line command specifies a point on the axis about which such a wall rotates.

Description An arbitrary axis in 3D is described by a point on the axis and a direction. This line command specifies the point on the axis so as to fix its location in space. This command cannot be specified by itself. The rotation speed must be specified. If the problem is three dimensional, then the axis direction must also be specified. The wall boundary condition will compute the vector-valued velocity from the given rotation speed and axis information, and enforce that value locally.

3.4.13. Pressure

Scope: Flow State

`Pressure {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies the static pressure.

3.4.14. Reynolds Length Scale

Scope: Flow State

`Reynolds Length Scale {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies the value of the length scale that is associated with the Reynolds number.

Description The Reynolds number is the nondimensional group,

$$R_e = \frac{\rho V L}{\mu}$$

where ρ is the reference density, V is the reference velocity, L is the length scale specified by this line command, and μ is the reference viscosity.

3.4.15. Reynolds Number

Scope: Flow State

`Reynolds Number {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies the value of the Reynolds number.

Description The Reynolds number is the nondimensional group,

$$R_e = \frac{\rho V L}{\mu}$$

where ρ is the reference density, V is the reference velocity, L is the length scale, and μ is the reference viscosity.

3.4.16. Rotation Speed

Scope: Flow State

`Rotation Speed {=|are|is}` *`omega`*

Parameter	Value	Default
omega	`real`	`undefined`

Summary A no slip wall can rotate. This line command specifies the magnitude of the rotational velocity of such a rotating wall.

Description This line command cannot be specified by itself. The direction of the rotation axis and a point on that axis must also be given. The wall boundary condition will compute the vector-valued velocity from the given rotation speed and axis information, and enforce that value locally. This value must have units of radians divided by time.

3.4.17. Temperature

Scope: Flow State

`Temperature {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies the static temperature.

3.4.18. Turbulence Intensity

Scope: Flow State

`Turbulence Intensity {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies the turbulence intensity, which is subsequently used to compute the turbulent kinetic energy.

Description For two equation turbulence models that solve a turbulent kinetic energy equation such as $k - \epsilon$ and SST, this line command allows the turbulence intensity, T_u, to be specified. You may not specify both the turbulence intensity and the turbulent kinetic energy. The turbulent kinetic energy, k, is computed from the given intensity according to the formula

$$k = \frac{3}{2}\left(T_u V\right)^2$$

where V is the flow speed.

3.4.19. Turbulent Dissipation

Scope: Flow State

`Turbulent Dissipation {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies the turbulent dissipation rate (ϵ or ω depending on the turbulence model).

Description For two equation turbulence models that solve a turbulent dissipation equation such as $k - \epsilon$ and SST, this line command allows the dissipation rate variable(ϵ or ω to be specified. Only one of the quantities turbulent dissipation, turbulent viscosity ratio, or turbulent length scale can be specified.

3.4.20. Turbulent Kinetic Energy

Scope: Flow State

`Turbulent Kinetic Energy {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies the turbulent kinetic energy.

Description For two equation turbulence models that solve a turbulent kinetic energy equation such as $k - \epsilon$ and SST, this line command allows the specific turbulent kinetic energy to be specified. It has units of velocity squared. Either the turbulent kinetic energy can be specified, or the turbulence intensity, but not both.

3.4.21. Turbulent Viscosity Ratio

Scope: Flow State

`Turbulent Viscosity Ratio {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary Ratio of turbulent to laminar viscosity that is used to compute value of dissipation rate

Description For two equation turbulence models that solve a turbulent dissipation rate equation such as $k-\epsilon$ and SST, this line command allows the dissipation rate to be computed from the given ratio of turbulent to laminar viscosity. You may not specify both this ratio and the turbulent dissipation. For SST, the specific turbulent turbulent dissipation rate, ω, is computed from the given viscosity ratio according to the formula

$$\omega = \rho \frac{k}{\mu \alpha}$$

where ρ is the flow density, k is the turbulent kinetic energy, μ is the laminar viscosity, and α is the given ratio of turbulent to laminar viscosity. For the $k-\epsilon$ model, the formula is

$$\epsilon = \rho C_\mu \frac{k^2}{\mu \alpha}$$

where $C_\mu = 0.09$ is the $k-\epsilon$ model parameter.

3.4.22. Velocity

Scope: Flow State

`Velocity {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies the magnitude of the velocity.

Description Either the Mach number or the magnitude of the velocity must be specified, but not both.

3.5. GAS PROPERTIES

Scope: Conchas Region

```
Begin Gas Properties
    Constant_Viscosity {=|are|is} constant_viscosity
    Gamma {=|are|is} gamma
    Gas Model File {=|are|is} gasfile
    Gas Model Type {=|are|is} GasModelType
    Prandtl {=|are|is} prandtl
    Specific_R {=|are|is} specific_r
    Sutherland_C1 {=|are|is} sutherland_c1
    Sutherland_C2 {=|are|is} sutherland_c2
End
```

Summary Specify the gas properties.

3.5.1. Constant_Viscosity

Scope: Gas Properties

```
Constant_Viscosity {=|are|is} constant_viscosity
```

Parameter	Value	Default
constant_viscosity	`real`	`undefined`

Summary Specify Constant Viscosity. This will override Sutherland's Law.

3.5.2. Gamma

Scope: Gas Properties

```
Gamma {=|are|is} gamma
```

Parameter	Value	Default
gamma	`real`	`1.4`

Summary Specific heat ratio.

3.5.3. Gas Model File

Scope: Gas Properties

`Gas Model File {=|are|is}` *`gasfile`*

Parameter	Value	Default
gasfile	`string`	`undefined`

Summary Specify the gas file to read. Required to run a thermochem gas model

3.5.4. Gas Model Type

Scope: Gas Properties

`Gas Model Type {=|are|is}` *`GasModelType`*

Parameter	Value	Default
GasModelType	`{ideal_gas_model|thermo_chem_gas_model}`	`undefined`

Summary Specify the gas model to use.

3.5.5. Prandtl

Scope: Gas Properties

`Prandtl {=|are|is}` *`prandtl`*

Parameter	Value	Default
prandtl	`real`	`0.72`

Summary Prandtl number.

3.5.6. Specific_R

Scope: Gas Properties

`Specific_R {=|are|is}` *`specific_r`*

Parameter	Value	Default
specific_r	`real`	`287.101591850829`

Summary Specific gas constant.

3.5.7. Sutherland_C1

Scope: Gas Properties

`Sutherland_C1 {=|are|is}` *`sutherland_c1`*

Parameter	Value	Default
sutherland_c1	`real`	`1.458e-6`

Summary First Sutherland Law constant.

3.5.8. Sutherland_C2

Scope: Gas Properties

`Sutherland_C2 {=|are|is}` *`sutherland_c2`*

Parameter	Value	Default
sutherland_c2	`real`	`110.4`

Summary Second Sutherland Law constant.

3.6. SPONGE LAYER

Scope: Conchas Region

```
Begin Sponge Layer
    Center {=|are|is} Value1 Value2 [ Value3 ]
    Type {=|are|is} SpongeType
    Use Donor Mesh donorMeshName
    Use Flow State state_name
    R_Max {=|are|is} Value
    R_Min {=|are|is} Value
    Sigma {=|are|is} Value
End
```

Summary This line command specifies that an absorbing "sponge" layer model be used to absorb any waves radiated into the far field.

Description In order to absorb any waves radiated into the farfield. e.g. from a cavity, and prevent any spurious reflections back from the artificial boundaries, a sink, or sponge, term is added to the governing equations. This is especially useful for flows past cavities. Experience has shown this to be helpful in keeping the computational domain to a reasonable size and produce clean, uncorrupted wall pressure histories that agree well with measurements.

The sponge region is implemented by adding a sink term to the governing equations of the form

$$S = f(r)\frac{Q_{\text{target}} - Q}{\Delta t}$$

where

$$f(r) = \sigma \max(\min(1, \frac{r - r_{\text{min}}}{r_{\text{max}} - r_{\text{min}}}), 0)$$

This sink term has the effect of driving the solution towards Q_{target}, the target value. This target value is specified by a `flow state`, similar to the way in which initial and boundary conditions are specified.

3.6.1. Center

Scope: Sponge Layer

`Center {=|are|is}` *Value*$_1$ *Value*$_2$`[` *Value*$_3$`]`

Parameter	Value	Default
Value	`real_1 real_2[ real_3]`	`undefined`

Summary This line command specifies the center of the circle in 2D or the center of the sphere in 3D.

Description If the type is set to `radial`, then this line command is mandatory. If the type is not set to `radial`, then this line command has no effect.

3.6.2. Type

Scope: Sponge Layer

`Type {=|are|is}` *SpongeType*

Parameter	Value	Default
SpongeType	`{radial|x|y|z}`	`undefined`

Summary Describes the spatial manner in which the solution. decays.

Description The type defines the shape of the absorbing region, as well as the interpretation of the geometric parameters r_{min} and r_{max}.

radial The region is circular in two dimensional problems and spherical in three dimensional problems. The parameter r is the distance from any location in the mesh to the given center of the region, which must be set by the `center` command line. The parameters r_{min} and r_{max} are the inner and outer radii of the circle/sphere.

x The region is a semi-infinite box in the y and z coordinate directions. The parameter r is the x coordinate, and r_{min} and r_{max} define the extent of the box in the x direction.

y The region is a semi-infinite box in the x and z coordinate directions. The parameter r is the y coordinate, and r_{min} and r_{max} define the extent of the box in the y direction.

z The region is a semi-infinite box in the x and y coordinate directions. The parameter r is the z coordinate, and r_{min} and r_{max} define the extent of the box in the z direction.

3.6.3. Use Donor Mesh

Scope: Sponge Layer

`Use Donor Mesh` *`donorMeshName`*

Parameter	Value	Default
donorMeshName	`string`	`undefined`

Summary This line command specifies that the the flow state is populated from the mesh described in the Donor Mesh command block named donorMeshName.

Description The use of a donor mesh is currently the only mechanism for defining a flow state that varies in more than one dimension (e.g. x and y or x and t). A `Donor Mesh` command block of the given name must exist at the region scope. See the definition for the `Donor Mesh` for more details.

3.6.4. Use Flow State

Scope: Sponge Layer

`Use Flow State` *`state_name`*

Parameter	Value	Default
state_name	`string`	`undefined`

Summary This line command specifies the name of a flow state to use.

Description The state of the gas (pressure, temperature, velocity, etc) is not defined inside of a command block that uses it. Instead, the flow state is defined inside `aflow state` command block that appears at the region scope. This line command refers to a flow state block named named `state_name`.

3.6.5. R_Max

Scope: Sponge Layer

`R_Max {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This parameter specifies the maximum value of the distance parameter in the sink. See the line command `t` type.

3.6.6. R_Min

Scope: Sponge Layer

`R_Min {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This parameter specifies the minimum value of the distance parameter in the sink. See the line command `t` type..

3.6.7. Sigma

Scope: Sponge Layer

`Sigma {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	0

Summary This parameter specifies the decay factor σ in the sink.

3.7. ADAPTIVITY

Scope: Conchas Region

```
Begin Adaptivity label

    At Step Step

    Element Max Growth Factor {=|are|is} Growth

    Error Indicator {=|are|is} ErrorIndicator

    Max Refinement Level {=|are|is} RefinementLevel

    Refine Fraction {=|are|is} RefineFraction

    Scale Indicator By Volume

    Unrefine Fraction {=|are|is} UnRefineFraction

End
```

Summary This command block specifies options for adapting the mesh during a simulation.

Description The mesh is refined and/or unrefined and reinitialized. Multiple discrete steps can be specified. Only one set of options can be specified for each block but additional adaptivity blocks can be used for to change options. This is currently a PROTOTYPE capability.

3.7.1. At Step

Scope: Adaptivity

```
At Step Step
```

Parameter	Value	Default
Step	`integer`	`undefined`

Summary This line command specifies the incremental step at which to refine/unrefine the mesh.

Description This uses the incremental time step not the global time step.

3.7.2. Element Max Growth Factor

Scope: Adaptivity

`Element Max Growth Factor {=|are|is}` *`Growth`*

Parameter	Value	Default
Growth	`real`	`2.0`

Summary This line command specifies the multiplicative factor of how many total elements can be created compared with original element count.

Description This value is for the maximum growth in element count for an entire simulation. A value of 1.0 means no growth in the total number of elements. A value of 2.0 means that at most the number of elements will double due to adaptation in a simulation. Any value less than 1.0 does not make sense and is not permissible.

3.7.3. Error Indicator

Scope: Adaptivity

`Error Indicator {=|are|is}` *`ErrorIndicator`*

Parameter	Value	Default
ErrorIndicator	`{deslengthscaleratio|ete|highlowflux|limiter}`	`Limiter`

Summary This line command specifies the type of error indicator used to mark elements for adaptation.

3.7.4. Max Refinement Level

Scope: Adaptivity

`Max Refinement Level {=|are|is}` *`RefinementLevel`*

Parameter	Value	Default
RefinementLevel	`integer`	`2`

Summary This line command specifies the maximum number of times that an individual element can be refined.

3.7.5. Refine Fraction

Scope: Adaptivity

`Refine Fraction {=|are|is}` *`RefineFraction`*

Parameter	Value	Default
RefineFraction	`real`	`0.0`

Summary This line command specifies a decimal fraction of the error indicator range to be used to mark elements for refinement.

Description The value should be between 0.0 and 1.0. For example a value of 0.7 will mark elements for refinement that have an error indicator value in the the top 30 percent of the error indicator range (all elements above 70indicator range will be marked for refine).

3.7.6. Scale Indicator By Volume

Scope: Adaptivity

Summary This line command ensures the error indicator is scaled by the nodal volume. It may be useful for targeting global error in adaptivity.

3.7.7. Unrefine Fraction

Scope: Adaptivity

`Unrefine Fraction {=|are|is}` *`UnRefineFraction`*

Parameter	Value	Default
UnRefineFraction	`real`	`0.0`

Summary This line command specifies a decimal fraction of the error indicator range to be used to mark elements for unrefinement.

Description Should be between 0.0 and 1.0. For example a value of 0.1 will mark elements for unrefinement that have an error indicator value in the bottom 10 percent of the error indicator range. Only elements which have already been refined can be unrefined.

4. IO

4.1. DATA PROBE

Scope: Conchas Region

```
Begin Data Probe fileName

    Nodal fieldName Location {=|are|is} location1 location2[
    location3] [ Label label ]

    At Step startingStep Increment {=|are|is} increment

End
```

Summary Creates a text file named "fileName.txt" that contains solution field values at given locations.

Description The mesh is searched for each point location. If a given point lies outside the mesh, then it is projected to the nearest face of the nearest cell and linear interpolation is used to compute the value of the field.

4.1.1. Nodal

Scope: Data Probe

`Nodal` *fieldName* `Location {=|are|is}` *location*$_1$ *location*$_2$`[` *location*$_3$`]` `[ Label` *label* `]`

Parameter	Value	Default
fieldName	`string`	`undefined`
location	`real_1 real_2[ real_3]`	`undefined`

Summary Each line specifies a new probe that will be computed at every probe increment.

Description Sets the nodal field name, spatial location, and optionally the label for each probe. The given field will be interpolated on a cell that contains the point.

4.1.2. At Step

Scope: Data Probe

```
At Step startingStep Increment {=|are|is} increment
```

Parameter	Value	Default
startingStep	integer	undefined
increment	integer	undefined

Summary Write the results of the data probe beginning at step (startingStep) and output every specified number of steps.

4.2. SURFACE FIELD OUTPUT

Scope: Conchas Region

```
Begin Surface Field Output Name

    File Name {=|are|is} fieldName

    Add Surface surface_list...

    Scalar Field {=|are|is} fieldName

    Vector Field {=|are|is} fieldName

End
```

Summary Allows ASCII text output of the named fields on the named surfaces.

4.2.1. File Name

Scope: Surface Field Output

```
File Name {=|are|is} fieldName
```

Parameter	Value	Default
fieldName	string	undefined

Summary Specifies the name of the file to which the fields will be written.

4.2.2. Add Surface

Scope: Surface Field Output

```
Add Surface surface_list...
```

Parameter	Value	Default
surface_list	`string...`	`undefined`

Summary Output the nodes on the named surfaces.

4.2.3. Scalar Field

Scope: Surface Field Output

```
Scalar Field {=|are|is} fieldName
```

Parameter	Value	Default
fieldName	`string`	`undefined`

Summary Specifies the name of the nodal scalar field to be written.

4.2.4. Vector Field

Scope: Surface Field Output

```
Vector Field {=|are|is} fieldName
```

Parameter	Value	Default
fieldName	`string`	`undefined`

Summary Specifies the name of the nodal vector field to be written.

4.3. FORCE AND MOMENT

Scope: Conchas Region

```
Begin Force And Moment fileName
    Add Surface surface_list...
    At Step startingStep Increment {=|are|is} increment
    Moment Center {=|are|is} Values...
    Split Contributions
    Use Solid Walls
End
```

Summary This command sets parameters for calculating the force and moment resulting from the integration of the stress tensor on a surface.

Description There is no default surface. At least one surface description command (`add surface` do `use solid walls`) must be specified. The `add surface` command can be used more than once. If more than one such command is specified, then the integration is performed on all listed surfaces. Multiple specifications of the `use solid walls` command has no effect: the integration will only be performed once on the solid walls no matter how many times the command is included. It is not legal to specify internal surfaces.

The force associated with a momentum flux through the surface is also included, so that thrust is calculated.

The force and moment values are written to a text file called fileName.txt, where fileName is the name of the given command block. If the job is a restart, then the force and moment values are appended to the file. If no `at step` commands are specified, then output is performed once per timestep.

4.3.1. Add Surface

Scope: Force And Moment

`Add Surface` *`surface_list...`*

Parameter	Value	Default
surface_list	`string...`	`undefined`

Summary This line command adds the given boundary surfaces to the force and moment calculation.

Description Multiple surfaces may be added to the force and moment calculation by including a list of them in a single instance of this line command, e.g.

```
add surface surface_1 surface_2 surface_3
```

or by including this line command multiple times. It is not legal to specify internal surfaces-the surfaces must be exposed in the sense that a given face on the surface is connected to exactly one element.

4.3.2. At Step

Scope: Force And Moment

`At Step` *`startingStep`* `Increment {=|are|is}` *`increment`*

Parameter	Value	Default
startingStep	`integer`	`undefined`
increment	`integer`	`undefined`

Summary Write the results file starting with the given step, startingStep, and again every increment steps after that.

Description This line command is optional. If it is omitted, then output will be written every single timestep.

4.3.3. Moment Center

Scope: Force And Moment

`Moment Center {=|are|is}` *`Values...`*

Parameter	Value	Default
Values	`real...`	`undefined`

Summary This line command specifies the location of the point about which to calculate the moment.

Description This line command must be specified: there is no default. You must specify the same number of coordinates as the spatial dimension.

4.3.4. Split Contributions

Scope: Force And Moment

Summary This line command allows the total forces and moments to be split into inviscid and viscous components.

Description This optional line command in the force and moment block allows the total forces and moments to be split into inviscid and viscous components. If it is included, then the forces and moments will be split into inviscid and viscous components when written to the output file. When this option is active for a 3D simulation, a total of 18 columns of force and moment data will be outputted: the first 6 will have the totals in the x-, y- and z- direction, the next 6 will have the inviscid contributions and the last 6 will have the viscous contributions.

4.3.5. Use Solid Walls

Scope: Force And Moment

Summary This line command specifies that the stress is to be integrated on the no-penetration walls.

Description For viscous flows, the code will add all of the surfaces on which `solid wall` boundary conditions have been defined. For inviscid flows, the code will add all surfaces at which `tangent flow` boundary conditions have been defined. Note that the surfaces associated with `tangent flow` boundary conditions are not automatically added to the force and moment calculation for viscous flows. If the `add surface` command is also specified, then the surfaces so defined will be added to the force and moment calculation.

4.4. AVERAGING

Scope: Conchas Region

```
Begin Averaging AverageName

    Favre Average Field RegisteredField As AverageField [ On Output
    Block partName ]

    Reynolds Average Covariance Of Velocity As AverageField [ On
    Output Block partName ]

    Reynolds Average Field RegisteredField As AverageField [ On Output
    Block partName ]

    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.

4.4.1. Favre Average Field

Scope: Averaging

`Favre Average Field` *`RegisteredField`* `As` *`AverageField`* `[ On Output Block` *`partName`* `]`

Parameter	Value	Default
RegisteredField	`string`	`undefined`
AverageField	`string`	`undefined`

Summary Generates a Favre average of the given 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 part. It the part name is not specified, the average will be defined and output on all nodes on which the field to be averaged 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".

4.4.2. Reynolds Average Covariance Of Velocity

Scope: Averaging

`Reynolds Average Covariance Of Velocity As` *`AverageField`* `[ On Output Block` *`partName`* `]`

Parameter	Value	Default
AverageField	`string`	`undefined`

Summary Generates the Reynolds averaged covariance of velocity.

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 Reynolds average based covariance of the velocities is:

$$u\_i' \bar{u}\_j' = u\_i \bar{u}\_j - \bar{u}\_i \bar{u}\_j$$

If the optional mesh part name is not specified, then the average will be defined and output on all nodes on which the field to be averaged exists.

4.4.3. Reynolds Average Field

Scope: Averaging

`Reynolds Average Field` *`RegisteredField`* `As` *`AverageField`* `[ On Output Block` *`partName`* `]`

Parameter	Value	Default
RegisteredField	`string`	`undefined`
AverageField	`string`	`undefined`

Summary Generates a Reynolds average of the given 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. It the optional mesh part name is not specified, the average will be defined and output on all nodes on which the field to be averaged 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".

4.4.4. 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.

4.4.5. 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.

4.5. RESULTS OUTPUT

Scope: Conchas Region

```
Begin Results Output BlockName
    Title title
    Append Iteration
    At Step startingStep Increment {=|are|is} increment
    Catalystfile Name {=|are|is} fileName
    Database Name {=|are|is} fileName
    Nodal Variable {=|are|is} internalName [ As newName ]
End
```

Summary This command block contains the solution field output specification commands.

Description Currently only the ExodusII format is supported. By default, the code will output the solution variables, $\mathbf{U} = (\rho, \rho\mathbf{u}, \rho E)$, called `conserved_variables`. For parallel runs, one output file per processor is created that contains the portion of the mesh local to the processor.

4.5.1. Title

Scope: Results Output

`Title title`

Parameter	Value	Default
title	`string`	`undefined`

Summary This line command specifies the title that will appear in the results file.

4.5.2. Append Iteration

Scope: Results Output

Summary This line command appends the starting iteration of the simulation to the database name.

Description This line command is mainly useful for restarted runs. It will append the iteration count which is persistent across restart to the results output filename.

4.5.3. At Step

Scope: Results Output

`At Step` *`startingStep`* `Increment {=|are|is}` *`increment`*

Parameter	Value	Default
startingStep	`integer`	`undefined`
increment	`integer`	`undefined`

Summary Write the results file starting with the given step, startingStep, and again every increment steps after that.

Description This line command is optional. If it is omitted, then output will be written every single timestep.

4.5.4. Catalystfile Name

Scope: Results Output

`Catalystfile Name {=|are|is}` *`fileName`*

Parameter	Value	Default
fileName	`string`	`undefined`

Summary This line command specifies the name of the paraview catalyst visualization input file.

Description The given file name may specify an absolute or relative path to the file. For example,

```
catalystfile name = foo/bar/file.txt
```

directs the code to expect to find the file named file.txt two directories below the current working directory.

4.5.5. Database Name

Scope: Results Output

`Database Name {=|are|is}` *`fileName`*

Parameter	Value	Default
fileName	`string`	`undefined`

Summary This line command specifies the name of the results file.

Description The given file name may specify an absolute or relative path to the file. For example,

```
database name = foo/bar/file.e
```

directs the code to expect to find the file named file.e two directories below the current working directory.

4.5.6. Nodal Variable

Scope: Results Output

`Nodal Variable {=|are|is}` *`internalName`* `[ As` *`newName`* `]`

Parameter	Value	Default
internalName	`string`	`undefined`

Summary This line command will write the given nodal variable to the associated results file.

Description Optionally, you may change the name of the field to newName as it appears in the results file. For example,

```
nodal variable density as rho
```

will write the internal field named density to the exodus file but with the name changed to rho. The following nodal fields may be available for output:

density For an ideal gas, this is the flow density and is a scalar. For a gas with species transport, this is a vector of the species densities.

mixture_density For a flow with species transport, this is the mixture density of the gas.

velocity the flow velocity

pressure the static pressure

temperature the static temperature

speed_of_sound the speed of sound

grad_density the density gradient

grad_temperature the temperature gradient

grad_pressure the pressure gradient

grad_velocity the velocity gradient. Note that this is a tensor. The elements are ordered $\frac{\partial u_1}{\partial x_1}$, $\frac{\partial u_1}{\partial x_2}$, $\frac{\partial u_1}{\partial x_3}$, $\frac{\partial u_2}{\partial x_1}$ etc.

viscosity The molecular viscosity

turbulent_viscosity For turbulent flows, this is the turbulent viscosity computed as appropriate for the given turbulence model.

turbulent_kinetic_energy the turbulent kinetic energy

grad_turbulent_kinetic_energy the gradient of the turbulent kinetic energy

turbulent_dissipation For SST, this is the specific dissipation rate ω. For $k-\epsilon$, this is the dissipation, ϵ.

grad_turbulent_dissipation For SST, this is the gradient of the specific dissipation rate: $\nabla\omega$. For $k-\epsilon$, this is the gradient of dissipation, $\nabla\epsilon$.

nodal_timestep The time step size (Δt) is stored at the nodes. For global timestepping, each node will have the same value. For local timestepping, each node may have a different value.

nuhat For Spalart-Allmaras this is the working variable, $\hat{\nu}$.

grad_nuhat For Spalart-Allmaras this is the gradient of the working variable, $\nabla\hat{\nu}$.

nodal_limiter If a stencil limiter (as opposed to an edge limiter) is used, then the actual limiter values are stored in this field. They are ordered as follows: density, pressure or temperature, velocity, turbulent kinetic energy, turbulent dissipation.

is_clipped For turbulent flows, this field will have a value of 1 if a turbulent degree of freedom (e.g., turbulent kinetic energy) has been clipped to stay positive, and 0 otherwise.

ndtw For turbulent flows, this is the nearest distance to the wall.

4.6. RESTART INPUT

Scope: Conchas Region

```
Begin Restart Input BlockName

    Restart Instance {=|are|is} instance

    Activate Restart

    Database Name {=|are|is} fileName

    Reset Time {=|are|is} time

End
```

Summary Contains the restart input specification commands.

4.6.1. Restart Instance

Scope: Restart Input

`Restart Instance {=|are|is}` *`instance`*

Parameter	Value	Default
instance	`integer`	`INT_Max`

Summary This optional command specifies which restart will be used. If not specified, the last instance is used.

Description A restart instance is a state of the simulation at a specific time and is created each time a restart is requested. For example, if restart is requested every 10 time steps and the simulation is run for 50 time steps, 5 restart instances will occur. To save space, typically a maximum number of restart instances is specified. Thus, we only save a small number of restart instances. In most circumstances, the last restart written is the desired state to restart from. However, in some cases the last state may be corrupt or otherwise unusable. The user can specify which restart instance–an integer number between 1 and the maximum number of instances–to restart the simulation.

4.6.2. Activate Restart

Scope: Restart Input

Summary This line command must be present to actually restart the simulation.

4.6.3. Database Name

Scope: Restart Input

`Database Name {=|are|is}` *`fileName`*

Parameter	Value	Default
fileName	`string`	`undefined`

Summary This line command specifies the root name of the input restart file.

Description Whatever filename root was used to output the restart should be used here.

4.6.4. Reset Time

Scope: Restart Input

`Reset Time {=|are|is}` *time*

Parameter	Value	Default
time	`real`	`undefined`

Summary This line command resets the time to the user chosen value instead of using the value in the restart file.

Description This only resets the simulation time value to the specified user value. The timestep count is still set from the restart file and cannot be modified. This feature is useful if some information is dependent on the simulation time such as a boundary condition.

4.7. RESTART OUTPUT

Scope: Conchas Region

```
Begin Restart Output BlockName

    Title title

    At Step startingStep Increment {=|are|is} increment

    Database Name {=|are|is} fileName

    Maximum Restart Instances {=|are|is} max_instances

End
```

Summary Contains the restart output specification commands.

Description The restart behavior in Aero is meant to require minimal user intervention. Using the commands described below, a file structure is automatically created.

Simulation restarts are managed by the creation of restart instances–simulation states at a particular simulation time. Each restart instance is written into a folder of restart files, and symbolic links are created to each folder that make user and code interaction easier. The symbolic links of the form, $restart\_i$, are managed by the simulation to ensure that the last simulation state written always has the highest index, i. For this reason, it is important to not run multiple simulations in the same working directory. This will change the symbolic links and lead to non-deterministic restart behavior.

4.7.1. Title

Scope: Restart Output

`Title` *`title`*

Parameter	Value	Default
title	`string`	`undefined`

Summary This optional line command specifies the title that will appear in the restart file.

4.7.2. At Step

Scope: Restart Output

`At Step` *`startingStep`* `Increment {=|are|is}` *`increment`*

Parameter	Value	Default
startingStep	`integer`	`undefined`
increment	`integer`	`1`

Summary Write the restart file starting with the given step, startingStep, and again every increment steps after that.

4.7.3. Database Name

Scope: Restart Output

`Database Name {=|are|is}` *`fileName`*

Parameter	Value	Default
fileName	`string`	`undefined`

Summary This line command specifies the root name of the restart file.

Description Do not prepend a folder name to the $fileName$. It should have the form restartName.rst.

4.7.4. Maximum Restart Instances

Scope: Restart Output

```
Maximum Restart Instances {=|are|is} max_instances
```

Parameter	Value	Default
max_instances	integer	1

Summary This optional line command specifies the maximum number of restart instances to keep.

Description The maximum restart instances should be set to a lower number to save disk space. A typical number is 2 or 3.

4.8. EXTERNAL MESH OUTPUT

Scope: Conchas Region

```
Begin External Mesh Output BlockName

    Source Parts {=|are|is} PartList...  [ Except ExceptionPartList...
    ]

    At Step startingStep Increment {=|are|is} increment

    Convert Nodal Variable internalName To Sideset Variable [ As
    newName ]

    Mesh Database Name {=|are|is} fileName

    Nodal Variable {=|are|is} internalName [ As newName ]

    Output Database Name {=|are|is} fileName

    Target Parts {=|are|is} PartList...  [ Except ExceptionPartList...
    ]

End
```

Summary Contains the external mesh output specification commands. External mesh output interpolates and outputs to an external mesh.

Description External Mesh Output writes solution results to a mesh different than the one that was used to solve the problem. The mesh could be a coarser mesh or a geometrically smaller mesh. This external mesh must wholly fit within the domain of the mesh used to solve the problem. Currently only the ExodusII format is supported.

4.8.1. Source Parts

Scope: External Mesh Output

`Source Parts {=|are|is}` *`PartList...`* `[ Except` *`ExceptionPartList...`* `]`

Parameter	Value	Default
PartList	`string...`	`all_parts`

Summary This command provides the list of source parts to search in the solution mesh.

Description This command describes the part of the original solution mesh that is to be searched for the spatial locations described by the list of target parts. For example,

```
Target Parts = surface_1 surface_2
```

The special part names all_volumes or all_surfaces can be used, instead of the part list, optionally with the keyword `except` and a list of parts to not include in the transfer, e.g.

```
Source Parts = all_volumes except block_4 block_2
```

If this line command is omitted, then the entire mesh will be searched.

4.8.2. At Step

Scope: External Mesh Output

`At Step` *`startingStep`* `Increment {=|are|is}` *`increment`*

Parameter	Value	Default
startingStep	`integer`	`undefined`
increment	`integer`	`undefined`

Summary Interpolate variables and Write the results file starting with the given step, startingStep, and again every increment steps after that.

Description This line command is optional. If it is omitted, then interpolation and output will occur every single timestep.

4.8.3. Convert Nodal Variable

Scope: External Mesh Output

`Convert Nodal Variable` *`internalName`* `To Sideset Variable [ As` *`newName`* `]`

Parameter	Value	Default
internalName	`string`	`undefined`

Summary Convert a nodal field, such as pressure, to a sideset field that is constant on each face.

Description This command is an alternative to the nodal variable command, and will transfer a nodal variable from the solution to a sideset variable that is constant on each face. This sideset variable is only defined on the faces of the output database that are named in the target parts command.

Optionally, you may change the name of the variable to newName as it appears in the results file. For example,

```
convert nodal variable pressure to sideset variable as p_surface
```

will write the internal variable named pressure to a surface in the output database but with the name changed to p_surface.

For a list of legal nodal variables, see the nodal variable command.

4.8.4. Mesh Database Name

Scope: External Mesh Output

`Mesh Database Name {=|are|is}` *`fileName`*

Parameter	Value	Default
fileName	`string`	`undefined`

Summary This line command specifies the name of external mesh file that defines the geometry to interpolate the solution onto.

Description Currently only the ExodusII format is supported. This file is only read from.

The given file name specifies an absolute or relative path to the file. For example,

```
database name = foo/bar/file.e
```

directs the code to expect to find the file named file.e two directories below the current working directory.

4.8.5. Nodal Variable

Scope: External Mesh Output

`Nodal Variable {=|are|is}` *`internalName`* `[ As` *`newName`* `]`

Parameter	Value	Default
internalName	`string`	`undefined`

Summary Output the given nodal variable to the associated results file.

Description Optionally, you may change the name of the variable to newName as it appears in the results file. For example,

```
nodal variable = density as rho
```

will write the internal variable named density to the exodus file but with the name changed to rho. The following nodal variables may be available for output:

density For an ideal gas, this is the flow density and is a scalar. For a gas with species transport, this is a vector of the species densities.

mixture_density For a flow with species transport, this is the mixture density of the gas.

velocity the flow velocity

pressure the static pressure

temperature the static temperature

speed_of_sound the speed of sound

grad_density the density gradient

grad_temperature the temperature gradient

grad_pressure the pressure gradient

grad_velocity the velocity gradient. Note that this is a tensor. The elements are ordered $\frac{\partial u_1}{\partial x_1}, \frac{\partial u_1}{\partial x_2}, \frac{\partial u_1}{\partial x_3}, \frac{\partial u_2}{\partial x_1}$ etc.

viscosity The molecular viscosity

turbulent_viscosity For turbulent flows, this is the turbulent viscosity computed as appropriate for the given turbulence model.

turbulent_kinetic_energy the turbulent kinetic energy

grad_turbulent_kinetic_energy the gradient of the turbulent kinetic energy

turbulent_dissipation For SST, this is the specific dissipation rate ω. For $k - \epsilon$, this is the dissipation, ϵ.

grad_turbulent_dissipation For SST, this is the gradient of the specific dissipation rate: $\nabla\omega$. For $k-\epsilon$, this is the gradient of dissipation, $\nabla\epsilon$.

nodal_timestep The time step size (Δt) is stored at the nodes. For global timestepping, each node will have the same value. For local timestepping, each node may have a different value.

nuhat For Spalart-Allmaras this is the working variable, $\hat{\nu}$.

grad_nuhat For Spalart-Allmaras this is the gradient of the working variable, $\nabla\hat{\nu}$.

nodal_limiter If a stencil limiter (as opposed to an edge limiter) is used, then the actual limiter values are stored in this field. They are ordered as follows: density, pressure or temperature, velocity, turbulent kinetic energy, turbulent dissipation.

is_clipped For turbulent flows, this field will have a value of 1 if a turbulent degree of freedom (e.g., turbulent kinetic energy) has been clipped to stay positive, and 0 otherwise.

ndtw For turbulent flows, this is the nearest distance to the wall.

4.8.6. Output Database Name

Scope: External Mesh Output

`Output Database Name {=|are|is}` *`fileName`*

Parameter	Value	Default
fileName	`string`	`undefined`

Summary This line command specifies the name of the output results file.

Description The mesh topology is read in from `mesh database name`. The solution is then interpolated onto this mesh before output. The mesh file defined in `mesh database name` is not modified.

The given file name specifies an absolute or relative path to the file. For example,

```
database name = foo/bar/file.e
```

directs the code to expect to find the file named file.e two directories below the current working directory.

4.8.7. Target Parts

Scope: External Mesh Output

```
Target Parts {=|are|is} PartList...  [ Except ExceptionPartList...  ]
```

Parameter	Value	Default
PartList	`string...`	`all_parts`

Summary Specify which mesh parts in the output database will receive the variable field data.

Description This command describes the mesh parts in the output database that are to be populated with the field variables. The mesh parts must be of the same kind, e.g., all surfaces or all volumes. They may be a different kind than the Source Parts: it is permissible to transfer from a volume to a surface. For example,

```
Target Parts = surface_1 surface_2
```

The special part names all_volumes or all_surfaces can be used, instead of the part list, optionally with the keyword `except` and a list of parts to not include in the transfer, e.g.

```
Target Parts = all_volumes except block_4 block_2
```

4.9. DONOR MESH

Scope: Conchas Region

```
Begin Donor Mesh transferName

    Donor Parts {=|are|is} IoDonorPartList...  [ Except
    donorExceptionPartList...  ]

    Receiver Parts {=|are|is} IoReceiverPartList...  [ Except
    receiverExceptionPartList...  ]

    Massfracs {=|are|is} Values...

    Mesh Database Name {=|are|is} fileName

    Variable Conserved_Variables [ From varName ]

    Variable Pressure [ From varName ]

    Variable Temperature [ From varName ]

    Variable Turbulent_Dissipation [ From varName ]

    Variable Turbulent_Kinetic_Energy [ From varName ]

    Variable Velocity [ From varName ]
```

```
End
```

Summary This command block defines a donor mesh file and parameters for use with a flow state, a sponge, or other capabilities which may use a separate donor mesh.

4.9.1. Donor Parts

Scope: Donor Mesh

```
Donor Parts {=|are|is} IoDonorPartList...  [ Except
donorExceptionPartList...  ]
```

Parameter	Value	Default
IoDonorPartList	`string...`	`all_parts`

Summary This optional command allows more control in transferring data by providing a list of parts to search in the donor mesh.

Description By default, the full donor mesh is used for transferring data. When this command is included, the transfer is limited to parts provided in the list:

```
Donor Parts = block_1 block_2 block_3 block_5
```

The special part all_parts can be used instead of the part list with `except` and a list of parts to not include in the transfer:

```
Donor Parts = all_parts except block_4
```

4.9.2. Receiver Parts

Scope: Donor Mesh

```
Receiver Parts {=|are|is} IoReceiverPartList...  [ Except
receiverExceptionPartList...  ]
```

Parameter	Value	Default
IoReceiverPartList	`string...`	`all_parts`

Summary This optional command allows more control in transferring data by providing a list of parts on the simulation mesh that data will be transferred to.

Description By default, the full simulation mesh is used to transfer data. However, sometimes it is desired to only transfer data to regions of the mesh where it will be used, such as a volumetric transfer used for a sponge layer. When this command is included, the transfer is limited to parts provided in the list:

```
Receiver Parts = block_1 block_2 block_3 block_5
```

The special part all_parts can be used instead of the part list with `except` and a list of parts to not include in the transfer:

```
Receiver Parts = all_parts except block_4
```

4.9.3. Massfracs

Scope: Donor Mesh

`Massfracs {=|are|is}` *`Values...`*

Parameter	Value	Default
Values	`real...`	`undefined`

Summary For flows involving species transport, this line command specifies the mass fractions.

Description For species transport, this line command specifies the mass fractions of each species. The mass fractions must sum to one, that is

$$\sum_{s=1}^{S} y_s = 1$$

where y_s is the mass fraction for species s. The density of each species is computed according to

$$\rho_s = \rho y_s$$

where ρ is the mixture density.

4.9.4. Mesh Database Name

Scope: Donor Mesh

`Mesh Database Name {=|are|is}` *`fileName`*

Parameter	Value	Default
fileName	`string`	`undefined`

Summary This line command specifies the name of mesh file to use for reading in variables.

4.9.5. Variable Conserved_Variables

Scope: Donor Mesh

Summary reference conserved variables

4.9.6. Variable Pressure

Scope: Donor Mesh

Summary Static pressure.

4.9.7. Variable Temperature

Scope: Donor Mesh

Summary temperature

4.9.8. Variable Turbulent_Dissipation

Scope: Donor Mesh

Summary turbulent dissipation rate (ϵ or ω depending on the turbulence model)

4.9.9. Variable Turbulent_Kinetic_Energy

Scope: Donor Mesh

Summary turbulent kinetic energy

4.9.10. Variable Velocity

Scope: Donor Mesh

Summary velocity

5. INITIAL CONDITIONS

The initial conditions are specified in Aero in a command block, which appears at the region scope. Initial conditions may be specified individually on each element block, or on a group of element blocks, within a command block. The most common way of specifying the initial condition is using a flow state. The flow state block is located in the Aero region and its syntax is found in section ??. Another option is to copy a nodal variable from an exodus database.

5.1. INITIAL CONDITION BLOCK

Scope: Conchas Region

```
Begin Initial Condition Block BlockName

    All Volumes [ Except partList...  ]

    Use Mesh Database

    Use Flow State state_name

    Volume {=|are|is} volumeList...

End
```

Summary This block command allows the specification of piecewise constant initial conditions on any combination of volumes, surfaces and nodes.

Description The initial condition specifies how the `conserved_variables` $(\rho, \rho\mathbf{u}, \rho E)$ are set at the very first time step of a job. The initial conditions have no effect if the job is a restart.

Initial conditions must be specified at every node in the mesh. An initial condition may be specified on a mesh part by mesh part basis by including multiple initial condition command blocks. Typically, the initial condition is set by using a `flow state`, but it may also be read from the input mesh.

5.1.1. All Volumes

Scope: Initial Condition Block

Summary This line command will apply the initial condition to all nodes in the mesh, with the option of excluding certain named parts.

Description This line command is convenient if the mesh consists of more than a few parts. Multiple excluded parts can be specified by separating them with whitespace. For example,

```
all volumes except block_32 block_22
```

5.1.2. Use Mesh Database

Scope: Initial Condition Block

Summary This line command will apply the initial condition from the mesh database that is associated with the region.

Description The field named `conserved_variables` must exist in the mesh database that is specified with the region-scoped line command `mesh database name`. If more than one time step is in the database, then the code will use the data associated with the last available time. Note that it is possible to interpolate an initial condition from any other arbitrary mesh if that mesh is specified in a `flow state` as a `donor mesh`.

5.1.3. Use Flow State

Scope: Initial Condition Block

`Use Flow State` *`state_name`*

Parameter	Value	Default
state_name	`string`	`undefined`

Summary This line command specifies the name of a flow state to use.

Description The state of the gas (pressure, temperature, velocity, etc) is not defined inside of a command block that uses it. Instead, the flow state is defined inside a`flow state` command block that appears at the region scope. This line command refers to a flow state block named named `state_name`.

5.1.4. Volume

Scope: Initial Condition Block

`Volume {=|are|is}` *`volumeList...`*

Parameter	Value	Default
volumeList	`string...`	`undefined`

Summary This line command specifies the mesh parts on which this initial condition applies.

Description You may specify multiple volume names separated by whitespace. This line command may appear multiple times. Surface names may also be given here, and the code will apply the initial condition to all nodes on the given surface. For example, assume that `surface_3` is not a subset of `block_1`. Then

```
volume = block_1 surface_3
```

would result in the initial condition being applied on the set of nodes that is the union of those appearing in the volume `block_1` and the surface `surface_3`. Note that `surface_3` need not be specified if it in fact is a subset of `block_1`.

6. BOUNDARY CONDITIONS

Various boundary condition options exist for Aero. Many depend on a flow state which is specified in a flow state block within the Aero region block. Its syntax can be found in section ??.

Every exposed boundary face must have a boundary condition attached to it. As a preprocess, the code skins the mesh and finds all faces (or edges, if the problem is two-dimensional) that are only attached to one element. It also keeps track of all the faces on which boundary conditions have been applied. If there are any exposed faces on which there are no boundary conditions, the code will report how many there are as well as the global identifiers of the element to which each such face is attached. Finally, the code will abort.

6.1. WALL BOUNDARY CONDITION ON SURFACE

Scope: Conchas Region

```
Begin Wall Boundary Condition On Surface Surfacename
    Function For Heat Flux {=|are|is} FuncName [ In The Direction
    Direction ]
    Function For Temperature {=|are|is} FuncName [ In The Direction
    Direction ]
    Mesh Motion Type {=|are|is} MeshBCType [ Function functionName In
    The Direction Direction ]
    Use Wall Function
    Use Weak Wall
    Velocity Values {=|are|is} Values...
    Wall Heat Flux {=|are|is} Value
    Wall Temperature {=|are|is} Value
    Direction Of Rotation Axis {=|are|is} vector...
    Point On Rotation Axis {=|are|is} vector...
    Rotation Speed {=|are|is} omega
End
```

Summary

This block command defines a boundary condition that models a no-slip wall on the named surface of the mesh.

Description

This boundary condition typically would not be used for inviscid flows: slip walls should be modeled using the `tangent flow boundary condition` block.

A flow state is not used with this boundary condition.

By default, the wall surface is not moving and the specified velocity is zero. Also by default, the energy equation is treated adiabatically, so that the minimum specification of a no-slip wall is

```
Begin Wall Boundary Condition on Surface surface_1
End
```

This empty block specifies that a motionless, adiabatic no-slip wall condition be applied to the surface named surface_1.

If a wall is not adiabatic, then a wall temperature must be specified. If a wall velocity is specified, it may move rectilinearly or rotationally. It is illegal to specify both kinds of motion parameters in the same block. Rectilinear motion is specified via the `Velocity Values` line command. Rotational motion is specified via the three line commands with "rotation" as a keyword.

Note that if the entire mesh is in solid body rotation, then the rotation of the wall boundaries should not be specified in this command block. Instead, if the mesh rotation is specified in the `Solution Options` block, the wall velocity will automatically be set to the local mesh velocity.

Boundary conditions for the turbulent transport quantities are applied automatically and cannot be specified in the input file. The turbulent kinetic energy is set to zero. For SST, the specific dissipation rate, ω, is set to a large but finite value at the wall according to the formula

$$\omega = 60 \frac{\mu}{\beta_1 \rho \delta^2}$$

where $\beta_1 = 0.075$ is the SST model parameter, μ is the laminar viscosity, ρ is the density, and δ is an internally calculated length scale that is representative of the mesh spacing normal to the wall. For the $k - \epsilon$ model,the turbulent dissipation variable, ϵ, is computed from the formula the

$$\epsilon = 2\mu \frac{k_w}{\rho \delta^2}$$

where k_w is an internally calculated scale for the turbulent kinetic energy near the wall. For Spalart-Allmaras, the working variable is set to zero, namely

$$\hat{\nu} = 0$$

6.1.1. Function For Heat Flux

Scope: Wall Boundary Condition On Surface

`Function For Heat Flux {=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
FuncName	`string`	`undefined`

Summary Name of the function to use for the heat flux in the specified direction.

Description This line command specifies a function to be used to specify the wall heat flux that varies in one spatial direction. If the optional parameters are not specified, the x-direction is chosen. A positive heat flux is leaving the fluid domain. If a temperature is specified on the wall, the heat flux cannot be specified.

6.1.2. Function For Temperature

Scope: Wall Boundary Condition On Surface

`Function For Temperature {=|are|is}` *`FuncName`* `[ In The` *`Direction`* `Direction ]`

Parameter	Value	Default
FuncName	`string`	`undefined`

Summary Name of function to use for the temperature on the wall in the specified direction.

Description This line command specifies a function to be used to specify the wall temperature that varies in one spatial direction. If the optional parameters are not specified, the x-direction is chosen. If the temperature is specified, a heat flux cannot be specified.

6.1.3. Mesh Motion Type

Scope: Wall Boundary Condition On Surface

`Mesh Motion Type {=|are|is}` *`MeshBCType`* `[ Function` *`functionName`* `In The` *`Direction`* `Direction ]`

Parameter	Value	Default
MeshBCType	`{fluid_structure_interaction` `|specified_displacement|` `specified_normal_displacement|` `specified_velocity|surface_geometry|` `unconstrained}`	`undefined`

Summary This line command specifies the mesh motion boundary condition and corresponding function name.

Description This line command specifies which type of mesh motion boundary condition to apply. For boundary condition types that require a function, the function is specified as well. The boundary conditions that require a function are: SPECIFIED_DISPLACEMENT and SPECIFIED_NORMAL_DISPLACEMENT. SURFACE_GEOMETRY optionally takes a SPECIFIED_DISPLACEMENT function in time. FSI does not accept a function.

6.1.4. Use Wall Function

Scope: Wall Boundary Condition On Surface

Summary For turbulent flows, this line command specifies that a wall function model be applied, rather than integrate all the way to the wall.

Description By default, the code will integrate the Navier-Stokes equations all the way to the wall. If the mesh spacing is sufficiently fine so that the turbulence is resolved, then this approach is appropriate. However, if the turbulence is is underresolved, then a wall function model is more appropriate. In this case, instead of applying the no-slip condition, a shear stress condition is used, based on the near-wall mesh spacing. For details, see the Aero Theory Manual.

6.1.5. Use Weak Wall

Scope: Wall Boundary Condition On Surface

Summary This line command specifies that a weak, flux-based enforcement of the wall boundary condition be used.

Description By default, the code will apply the no-slip condition as a nodal boundary condition that sets the velocity identically to the desired value at the wall. This line command uses a flux-based approach that allows the velocity value to be determined as part of the solution process. Using such a weak enforcement may increase robustness, but the specified wall values will not be satisfied exactly. For details, see the Aero Theory Manual.

6.1.6. Velocity Values

Scope: Wall Boundary Condition On Surface

`Velocity Values {=|are|is}` *`Values...`*

Parameter	Value	Default
Values	`real...`	`undefined`

Summary This line command specifies the velocity of a wall in constant rectilinear motion.

Description The number of velocity components that are specified must match the spatial dimension of the problem. If this line command is omitted, then there is no motion and a zero velocity is imposed at the wall.

Note: the code assumes that the component of this velocity normal to the wall is zero, although it does not check for that. If the given velocity does not have this property, then the behavior is undefined.

6.1.7. Wall Heat Flux

Scope: Wall Boundary Condition On Surface

`Wall Heat Flux {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies a constant uniform heat flux over the wall boundary.

Description This line command specifies a constant specified heat flux that does not vary in time or space on the wall boundary condition. The default behavior is zero heat flux (adiabatic). If the wall temperature is specified, then the heat flux cannot be specified.

6.1.8. Wall Temperature

Scope: Wall Boundary Condition On Surface

`Wall Temperature {=|are|is}` *`Value`*

Parameter	Value	Default
Value	`real`	`undefined`

Summary This line command specifies the isothermal wall temperature.

Description This line command specifies an isothermal wall boundary condition using the specified temperature that does not vary in time or space. If this line command is specified, then the heat flux cannot be specified.

6.1.9. Direction Of Rotation Axis

Scope: Wall Boundary Condition On Surface

`Direction Of Rotation Axis {=|are|is}` *`vector...`*

Parameter	Value	Default
vector	`real...`	`undefined`

Summary A no slip wall can rotate. This line command specifies the direction of the axis about which such a wall rotates.

Description An arbitrary axis in 3D is described by a point on the axis and a direction. This line command specifies the direction of the axis, and cannot be specified by itself. The rotation speed and a point on the axis must also be specified. The wall boundary condition will compute the vector-valued velocity from the given rotation speed and axis information, and enforce that value locally. For a 2D simulation, this line command is ignored and the axis is assumed to be in the positive Z direction.

6.1.10. Point On Rotation Axis

Scope: Wall Boundary Condition On Surface

`Point On Rotation Axis {=|are|is}` *`vector...`*

Parameter	Value	Default
vector	`real...`	`undefined`

Summary A no slip wall can rotate. This line command specifies a point on the axis about which such a wall rotates.

Description An arbitrary axis in 3D is described by a point on the axis and a direction. This line command specifies the point on the axis so as to fix its location in space. This command cannot be specified by itself. The rotation speed must be specified. If the problem is three dimensional, then the axis direction must also be specified. The wall boundary condition will compute the vector-valued velocity from the given rotation speed and axis information, and enforce that value locally.

6.1.11. Rotation Speed

Scope: Wall Boundary Condition On Surface

`Rotation Speed {=|are|is}` *`omega`*

Parameter	Value	Default
omega	`real`	`undefined`

Summary A no slip wall can rotate. This line command specifies the magnitude of the rotational velocity of such a rotating wall.

Description This line command cannot be specified by itself. The direction of the rotation axis and a point on that axis must also be given. The wall boundary condition will compute the vector-valued velocity from the given rotation speed and axis information, and enforce that value locally. This value must have units of radians divided by time.

6.2. CHARACTERISTIC PROJECTION ON SURFACE

Scope: Conchas Region

```
Begin Characteristic Projection On Surface Surfacename

    Add Perturbations For Boundary Layer {=|are|is} blName [ With Wall
    Normal NormalDirection Direction And Spanwise SpanwiseDirection
    Direction ]

    Mesh Motion Type {=|are|is} MeshBCType [ Function functionName In
    The Direction Direction ]

    Type {=|are|is} SubsonicBCType

    Use Flow State state_name

End
```

Summary Uses a characteristic projection to enforce inflow/outflow boundary conditions on a named surface of the mesh.

Description This command block is used to apply subsonic inflow and outflow boundary conditions. You must use a flow state block in order to define the state associated with this boundary condition. A complete state must be specified, but in general only a subset of the data is used by this condition. For example, if `type = back_pressure`, then only the pressure is used by this boundary condition. You must also specify a type of projection: there is no default.

6.2.1. Add Perturbations For Boundary Layer

Scope: Characteristic Projection On Surface

```
Add Perturbations For Boundary Layer {=|are|is} blName [ With Wall Normal
NormalDirection Direction And Spanwise SpanwiseDirection Direction ]
```

Parameter	Value	Default
blName	`string`	`undefined`

Summary Specify added inflow perturbations using the boundary layer with blName.

Description This line command specifies that the boundary layer blName will be used to define inflow perturbations such that turbulence will be generated downstream of the inflow. It can only be used with characteristic projection and fixed at state boundary conditions.

6.2.2. Mesh Motion Type

Scope: Characteristic Projection On Surface

`Mesh Motion Type {=|are|is}` *`MeshBCType`* `[ Function` *`functionName`* `In The` *`Direction`* `Direction ]`

Parameter	Value	Default
MeshBCType	`{fluid_structure_interaction | specified_displacement | specified_normal_displacement | specified_velocity | surface_geometry | unconstrained}`	`undefined`

Summary This line command specifies the mesh motion boundary condition and corresponding function name.

Description This line command specifies which type of mesh motion boundary condition to apply. For boundary condition types that require a function, the function is specified as well. The boundary conditions that require a function are: SPECIFIED_DISPLACEMENT and SPECIFIED_NORMAL_DISPLACEMENT. SURFACE_GEOMETRY optionally takes a SPECIFIED_DISPLACEMENT function in time. FSI does not accept a function.

6.2.3. Type

Scope: Characteristic Projection On Surface

`Type {=|are|is}` *`SubsonicBCType`*

Parameter	Value	Default
SubsonicBCType	`{back_pressure | farfield | inlet | reservoir}`	`undefined`

Summary This line command specifies the type of the characteristic projection.

Description The `reservoir` type is typically used to model a subsonic inflow boundary for internal flows. In this case, the code will compute the total pressure, total temperature, and flow direction from the given flow state and enforce these three quantities at the inflow boundary.

The `inlet` type is similarly used to model subsonic inflows, but enforces that the flow velocity and and static temperature be equal to the svalues associated with the given flow state.

The `back_pressure` type is typically used to model a subsonic outflow boundary for internal flows. It enforces that the static pressure be set equal to the value associated with the given flow state.

The `farfield` type is typically used to model boundaries for external flows. This type will locally adapt the enforcement among the various combinations of supersonic/subsonic/inflow/outflow according to the local Mach number, flow direction and surface normal.

6.2.4. Use Flow State

Scope: Characteristic Projection On Surface

`Use Flow State` *`state_name`*

Parameter	Value	Default
state_name	`string`	`undefined`

Summary This line command specifies the name of a flow state to use.

Description The state of the gas (pressure, temperature, velocity, etc) is not defined inside of a command block that uses it. Instead, the flow state is defined inside a`flow state` command block that appears at the region scope. This line command refers to a flow state block named named `state_name`.

6.3. FIXED AT STATE BOUNDARY CONDITION ON SURFACE

Scope: Conchas Region

```
Begin Fixed At State Boundary Condition On Surface Surfacename

    Add Perturbations For Boundary Layer {=|are|is} blName [ With Wall
    Normal NormalDirection Direction And Spanwise SpanwiseDirection
    Direction ]

    Mesh Motion Type {=|are|is} MeshBCType [ Function functionName In
    The Direction Direction ]

    Use Flow State state_name

End
```

Summary This command block sets the solution to the given flow state.

Description This boundary condition is typically used for a supersonic inflow, since it completely specifies the solution, as opposed to the characteristic projection boundary condition, which only specifies certain combinations of the solution vector, according to the incoming characteristic modes. Note that this boundary condition does not necessarily imply that the boundary condition is fixed in space and time, since the values specified in the associated flow state may in fact vary. itself can vary.

6.3.1. Add Perturbations For Boundary Layer

Scope: Fixed At State Boundary Condition On Surface

`Add Perturbations For Boundary Layer {=|are|is}` *`blName`* `[ With Wall Normal` *`NormalDirection`* `Direction And Spanwise` *`SpanwiseDirection`* `Direction ]`

Parameter	Value	Default
blName	`string`	`undefined`

Summary Specify added inflow perturbations using the boundary layer with blName.

Description This line command specifies that the boundary layer blName will be used to define inflow perturbations such that turbulence will be generated downstream of the inflow. It can only be used with characteristic projection and fixed at state boundary conditions.

6.3.2. Mesh Motion Type

Scope: Fixed At State Boundary Condition On Surface

`Mesh Motion Type {=|are|is}` *`MeshBCType`* `[ Function` *`functionName`* `In The` *`Direction`* `Direction ]`

Parameter	Value	Default
MeshBCType	`{fluid_structure_interaction | specified_displacement | specified_normal_displacement | specified_velocity | surface_geometry | unconstrained}`	`undefined`

Summary This line command specifies the mesh motion boundary condition and corresponding function name.

Description This line command specifies which type of mesh motion boundary condition to apply. For boundary condition types that require a function, the function is specified as well. The boundary conditions that require a function are: SPECIFIED_DISPLACEMENT and SPECIFIED_NORMAL_DISPLACEMENT. SURFACE_GEOMETRY optionally takes a SPECIFIED_DISPLACEMENT function in time. FSI does not accept a function.

6.3.3. Use Flow State

Scope: Fixed At State Boundary Condition On Surface

`Use Flow State` *`state_name`*

Parameter	Value	Default
state_name	`string`	`undefined`

Summary This line command specifies the name of a flow state to use.

Description The state of the gas (pressure, temperature, velocity, etc) is not defined inside of a command block that uses it. Instead, the flow state is defined inside a `flow state` command block that appears at the region scope. This line command refers to a flow state block named named `state_name`.

6.4. EXTRAPOLATION BOUNDARY CONDITION ON SURFACE

Scope: Conchas Region

```
Begin Extrapolation Boundary Condition On Surface Surfacename
    Mesh Motion Type {=|are|is} MeshBCType [ Function functionName In
    The Direction Direction ]
End
```

Summary Specifies that the flow be left unspecified and values are simply extrapolated from the interior.

Description This is typically used for a supersonic outflow.

6.4.1. Mesh Motion Type

Scope: Extrapolation Boundary Condition On Surface

```
Mesh Motion Type {=|are|is} MeshBCType [ Function functionName In The
Direction Direction ]
```

Parameter	Value	Default
MeshBCType	{fluid_structure_interaction | specified_displacement | specified_normal_displacement | specified_velocity | surface_geometry | unconstrained}	undefined

Summary This line command specifies the mesh motion boundary condition and corresponding function name.

Description This line command specifies which type of mesh motion boundary condition to apply. For boundary condition types that require a function, the function is specified as well. The boundary conditions that require a function are: SPECIFIED_DISPLACEMENT and SPECIFIED_NORMAL_DISPLACEMENT. SURFACE_GEOMETRY optionally takes a SPECIFIED_DISPLACEMENT function in time. FSI does not accept a function.

6.5. TANGENT FLOW BOUNDARY CONDITION ON SURFACE

Scope: Conchas Region

```
Begin Tangent Flow Boundary Condition On Surface Surfacename

    Mesh Motion Type {=|are|is} MeshBCType [ Function functionName In
    The Direction Direction ]

    Use Reflection Enforcement

End
```

Summary Defines a tangent velocity boundary condition on a named surface of the mesh.

Description A `flow state` command block is not used with this boundary condition. This command block is typically used to specify slip conditions for inviscid flows. It is also used to model symmetry boundary conditions in viscous flows. The default method of enforcement is to weakly specify that the component of the velocity normal to the wall vanishes, and the inviscid flux therefore consists only of the pressure.

6.5.1. Mesh Motion Type

Scope: Tangent Flow Boundary Condition On Surface

```
Mesh Motion Type {=|are|is} MeshBCType [ Function functionName In The
Direction Direction ]
```

Parameter	Value	Default
MeshBCType	{fluid_structure_interaction | specified_displacement | specified_normal_displacement | specified_velocity | surface_geometry | unconstrained}	undefined

Summary This line command specifies the mesh motion boundary condition and corresponding function name.

Description This line command specifies which type of mesh motion boundary condition to apply. For boundary condition types that require a function, the function is specified as well. The boundary conditions that require a function are: SPECIFIED_DISPLACEMENT and SPECIFIED_NORMAL_DISPLACEMENT. SURFACE_GEOMETRY optionally takes a SPECIFIED_DISPLACEMENT function in time. FSI does not accept a function.

6.5.2. Use Reflection Enforcement

Scope: Tangent Flow Boundary Condition On Surface

Summary Use stronger, reflective enforcement of tangent flow condition. This option may not be as robust as the default, especially for blunt bodies.

Description This line command specifies that a stronger enforcement of the flow tangency condition be used. In this method, the numerical flux is constructed by taking the left state equal to the current value of the solution. The right state is set equal to the left state, except that the velocity is reflected about the given surface. Then the given flux function (e.g. a Roe flux) is used to evaluate the numerical flux.

6.6. PERIODIC

Scope: Conchas Region

```
Begin Periodic Name

    Master {=|are|is} Master

    Rotation About Point point1 point2[ point3] [ On Axis axis1 axis2[
    axis3] ]

    Search Tolerance {=|are|is} SearchTolerance

    Slave {=|are|is} Slave

    Theta {=|are|is} Theta

End
```

Summary The periodic command block is used to define periodic, or cyclic, boundary conditions.

Description This boundary condition models a domain that is periodic in the sense that fluid that flows out through one surface flows in through another surface. Multiple periodicity is allowed: For example, a cube-shaped domain may have each pair of parallel surfaces be periodic. This would be accomplished with three separate periodic boundary condition command blocks in the same input file, one for each parallel pair of surfaces.

There must be exactly two surfaces defined in this boundary condition: one must be designated the "master" and the other must be designated the "slave". Currently, the two surfaces must match up node-for-node: It must be possible to replicate the coordinates of the nodes in one surface by a simple translation or rotation of the corresponding nodes in the other surface. After this transformation is computed, a geometric check is made to determine if the nodes in fact have this property. The geometric check may be set to a specific tolerance, which is a small positive number (e.g. 10^{-8}), by using the command line `search tolerance`. If after the transformation, each node in the set of

slave nodes lies within a radius defined by the search tolerance from a corresponding node in the set of master nodes, the two sets are considered to match up node-for-node.

If the periodicity is translational, then the transformation is computed automatically from the distance between the master and slave surfaces. Rotational periodicity is specified by the line commands `rotation about point` and `theta`, which provide the parameters from which the transformation is computed.

6.6.1. Master

Scope: Periodic

`Master {=|are|is}` *`Master`*

Parameter	Value	Default
Master	`string`	`undefined`

Summary This line command defines the master surface for this periodic boundary condition.

6.6.2. Rotation About Point

Scope: Periodic

`Rotation About Point` *`point`*$_1$ *`point`*$_2$*`[ point`*$_3$*`]` `[ On Axis` *`axis`*$_1$ *`axis`*$_2$*`[ axis`*$_3$*`]`*
`]`

Parameter	Value	Default
point	`real_1 real_2[ real_3]`	`undefined`

Summary This line command indicates that the periodicity is rotational, not translational, and specifies the axis about which the rotation occurs.

Description The point and the axis define a vector about which the rotation occurs. For two dimensional problems, the axis is assumed to be in the positive 'z' direction. For three dimensional problems, the axis must be specified.

6.6.3. Search Tolerance

Scope: Periodic

`Search Tolerance {=|are|is}` *`SearchTolerance`*

Parameter	Value	Default
SearchTolerance	`real`	`1e-8`

Summary This line command defines the search tolerance used to determine if the master and slave surfaces match up node-for-node.

6.6.4. Slave

Scope: Periodic

`Slave {=|are|is}` *`Slave`*

Parameter	Value	Default
Slave	`string`	`undefined`

Summary This line command defines the slave surface for this periodic boundary condition.

6.6.5. Theta

Scope: Periodic

`Theta {=|are|is}` *`Theta`*

Parameter	Value	Default
Theta	`real`	`undefined`

Summary This line command defines the angle, in degrees, between the master and slave surfaces.

Description For a rotational periodic boundary condition, the master and slave surfaces subtend an angle, theta. A positive angle is defined by sweeping the master surface through the domain to the slave surface in the counterclockwise direction.

7. COUPLING

The Sierra/Aero code can be coupled with other codes to solve multiphysics problems. The coupling strategies are application specific. In the following sections, some of these couplings are outlined.

7.1. CTH TO AERO

Coupling between CTH and Aero is done through file using tracer points. Coupling between 3D, 2DC(axisymmetric), and 1DS(spherical) CTH simulations and 3D Aero simulations are supported. The coupling is done through the cth_tracers_nodes.py script. Details on how to use the script are in a separate document, CTH_Aero_Tracer_Coupling.pdf.

7.2. FLUID-STRUCTURE INTERACTION

Fluid-Structure Interaction(FSI) simulations are done through an MPMD model. Two different executables are run using different MPI communicators. The interface between the two codes is defined such than any fluid code can be coupled with any structure code that implements the interface. The commands needed for Sierra Aero are described below.

7.3. FSI DESCRIPTION

Scope: Conchas Region

```
Begin Fsi Description
    Coupling Type {=|are|is} FsiCouplingType
    Reference Pressure {=|are|is} reference_pressure
    Search Parts {=|are|is} PartList...  [ Except ExceptionPartList...
    ]
End
```

Summary This command block specifies various options for the fluid-structure interaction capability, which couples Sierra/Aero and SIERRA/SD. Currently this coupling is one-way only.

Description The fluid-structure interaction feature is activated by adding the option "–fsi" to the aero command line. Additionally, this command block is required in order to specify the parts in the fluid mesh to search for the points that belong to the structural mesh.

SIERRA/SD sends aero a list of nodes at which the pressure is required. This search occurs only when necessary: that is at start up or whenever the mesh changes. SIERRA/SD runs at a fixed time step size which may be different from aero's time step size. Instead of interpolating in time, aero will alter its time step so that the solution is calculated at the intervals at which data is expected by SIERRA/SD.

For a complete description of the algorithms used for the coupling, see "An Increment Towards an Aeroelastic Fluid Structure Interaction Simulation Capability", by Arunajatesan and Day, and "Design of FSI for Sigma/Salinas".

7.3.1. Coupling Type

Scope: Fsi Description

`Coupling Type {=|are|is}` *`FsiCouplingType`*

Parameter	Value	Default
FsiCouplingType	`{fsi_css|fsi_gss|fsi_one_way}`	`FSI_ONE_WAY`

Summary This line command specifies the coupling type to use. Two way coupling via FSI_GSS is only available with the high-order element algorithm.

7.3.2. Reference Pressure

Scope: Fsi Description

`Reference Pressure {=|are|is}` *`reference_pressure`*

Parameter	Value	Default
reference_pressure	`real`	`0.0`

Summary This line command specifies a reference pressure used to compute the forces for the FSI coupling. It is generally equal to the freestream or initial pressure.

Description It is important to set this value when the entire structure is not contained in the fluid domain or if the structure has voids.

7.3.3. Search Parts

Scope: Fsi Description

`Search Parts {=|are|is}` *`PartList...`* `[ Except` *`ExceptionPartList...`* `]`

Parameter	Value	Default
PartList	`string...`	`all_parts`

Summary This command provides the list of surface parts to search in the fluid mesh.

Description This command describes the fluid surface that is to be searched for the spatial locations from the structural code. For example,

```
Search Parts = surface_1 surface_2
```

The special part name all_parts can be used, instead of the part list. This will search the entire surface of the mesh. Optionally the keyword `except` and a list of parts to not include in the transfer may be used, e.g.

```
Search Parts = all_parts except surface_1
```

FREQUENTLY ASKED QUESTIONS

Below is a compilation of frequently asked questions and problematic scenarios raised by the SIERRA/Aero users.

1. Aero example/tutorial files.
   One can find an Aero training presentation, which includes two PDFs and multiple example files on the following website:
   http://compsim.sandia.gov/compsim/
   To navigate to the presentations, go to "Support & Services -> Documentation" Once there, go to "Version of the Day -> General Release" This will open a new page with an expandable tree. Go to "Training/Tutorials -> Thermal/Fluid" That's where one will find the files.
   If you'd like to view VOTD documentation for Aero and other Sierra codes, and you have access to the CEE LAN (blades) or the HPC platforms, you can look at these locations:

   ```
   /gpfs1/rpshaw/sierra/docs/
   ```

2. Model dies with unexplained error when mesh and BC are changed.
   The root cause of this problem typically lies in the mesh. One of the things that one can do is load the mesh into ParaView and post-process the mesh quality, to check if there are negative Jacobians. Positive skew Jacobians also pose problems.

3. Why are Restart files larger than Results files?
   Restart files typically are larger than Results files because every variable necessary to restart a run (persistent variables) is stored in a Restart file, while typically you will not output as many variables to the Results file, e.g., only velocity, temperature, and turbulence variables.

4. Non-zero mach number at wall BC.
   The mach number is identically zero on surfaces with no slip boundary conditions. If a wall function is used, which is a slip type BC, mach numbers should not be expected to equal zero on such a boundary.

5. Will IC read file read the last time step from an Exodus data set?
   Yes, this command will use the data from the last time step to set the IC.

6. How to use an IC from a file.
   This is possible, but only with `conserved_variables`. You also need to use the exodus file that you'll initialize from as your Finite Element Model database input. Note, the field `conserved_variables` is automatically written to results files.

7. Can I change the number of processors during a restart?
   Aero does not support this type of operation. However, if you have a set of n restart files, you could always concatenate them using EPU and then split them into m restart files using loadbal.

8. `Number of components` error when using wall functions.
   The problem is in defining the flow states. The number of components in the direction (or velocity) vector has to match the dimensionality of the grid.

9. Data planes do not contain the same number of time steps.
   Sometimes after a simulation run concatenation never finishes, which leads to a different number of timesteps in the concatenated files. You can always concatenate the files manually using EPU. The easiest way to use EPU is using the `-auto` option. If you use the auto option, you only need to list one of the decomposed output files. For example:

   ```
   epu -auto output.e.2048.0000
   ```

10. How to list relevant input options.
    A summary of all the available commands will be printed if you pass the executable the following option:

    ```
    --print-syntax
    ```

11. How to submit to sierra and ask only to parse the input for errors.
    The following command is available to check for input syntax:

    ```
    aero -i input.i --check-syntax
    ```

12. What is the syntax to get `mach_number` in the output?
    To output mach number, first the following must be included in the solution options block:

    ```
    post process mach_number on block_1
    ```

DISTRIBUTION

Email—Internal [redacted]

Name	Org.	Sandia Email Address
Technical Library	01177	libref@sandia.gov

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.