JAN 5 1998

1
~~MS 0619~~
~~Review & Approval Desk,~~ 15102
For ~~DOE/O~~STI

SANDIA REPORT
SAND98–2864
Unlimited Release
Printed January 1999

MPSalsa Version 1.5

A Finite Element Computer Program for Reacting Flow Problems Part 1: Theoretical Development

John N. Shadid, Andrew G. Salinger, Rodney C. Schmidt, Thomas M. Smith,
Scott A. Hutchinson, Gary L. Hennigan, Karen D. Devine, and Harry K. Moffat

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

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of
Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

SAND98-2864
Unlimited Release
Printed January 1999

Supersedes SAND95-2752
dated May 1996

MPSalsa Version 1.5

A FINITE ELEMENT COMPUTER PROGRAM FOR REACTING FLOW PROBLEMS

PART 1 - THEORETICAL DEVELOPMENT

John N. Shadid, Andrew G. Salinger, Rodney C. Schmidt, Thomas M. Smith,
Scott A. Hutchinson, and Gary L. Hennigan
Parallel Computational Science Department

Karen D. Devine
Parallel Computing Sciences Department

Harry K. Moffat
Chemical Processing Sciences Department

Sandia National Laboratories
P.O, Box 5800
Albuquerque, New Mexico 87185-1111

Abstract

The theoretical background for the finite element computer program, MPSalsa Version 1.5, is presented in detail. MPSalsa is designed to solve laminar or turbulent, low Mach number, two- or three-dimensional incompressible and variable density reacting fluid flows on massively parallel computers, using a Petrov-Galerkin finite element formulation. The code has the capability to solve coupled fluid flow (with auxiliary turbulence equations), heat transport, multicomponent species transport, and finite-rate chemical reactions, and to solve coupled multiple Poisson or advection-diffusion-reaction equations. The program employs the CHEMKIN library to provide a rigorous treatment of multicomponent ideal gas kinetics and transport. Chemical reactions occurring in the gas phase and on surfaces are treated by calls to CHEMKIN and SURFACE CHEMKIN, respectively. The code employs unstructured meshes, using the EXODUS II finite element database suite of programs for its input and output files. MPSalsa solves both transient and steady flows by using fully implicit time integration, an inexact Newton method and iterative solvers based on preconditioned Krylov methods as implemented in the Aztec solver library.

Acknowledgments

The authors wish to thank D. K. Gartling and M. L. Martinez for their helpful discussions regarding various aspects of the finite element formulation used in MPSalsa and M. A. Christon for helpful comments on the draft of this document. In addition, the work of H. Walker on an initial implementation of the inexact Newton method is gratefully acknowledged. This work was partially supported by the U. S. Department of Energy Office of Scientific Computing under contract DE-AC04-94AL85000, and by a Sandia Laboratory Directed Research and Development proposal.

Table of Contents:

1. Introduction

The theoretical development and numerical procedures for the finite element computer program, MPSalsa, are presented in detail in this document. A companion user's manual provides details on using MPSalsa for specific applications along with a number of example problems [1]. Employing unstructured meshes on massively parallel (MP) computers, MPSalsa is designed to solve two- or three-dimensional problems which exhibit coupled laminar or turbulent fluid flow, heat transport, species transport, and chemical reactions. The modeling equations defined in MPSalsa for fluid flow and mass conservation are the momentum transport and the total mass continuity equation for incompressible or variable density Newtonian fluids (Navier-Stokes equations). The heat transport equation, auxiliary turbulence equations, and an arbitrary number of species transport-reaction equations couple strongly with each other through chemical reaction source terms and with the fluid flow equations through property variation and body force terms.

Several different turbulence models are currently implemented in MPSalsa. For modeling the Reynolds averaged Navier-stokes equations (i.e. the "RANS" turbulence modeling approach) both a standard k-ε type model (with several low Reynolds number modeling options) and the Spalart-Allmaras one-equation turbulent viscosity model [2] are available. For modeling the spatially filtered Navier-stokes equations (i.e. a large eddy simulation or LES modeling approach) a basic Smagorinsky subgrid model and a one-equation subgrid kinetic energy model are available.

The program uses the CHEMKIN suite of library routines to provide a rigorous treatment of ideal-gas multicomponent transport, including the effects of thermal diffusion [3]. The mixture-averaged diffusion approximation is available in addition to the computationally-expensive Dixon-Lewis formulation. Chemical reactions occurring in the gas phase and on surfaces are also treated by calls to CHEMKIN [4] and SURFACE CHEMKIN [5], respectively. Because of this, MPSalsa can handle varying numbers and types of chemical reactions and species in a robust manner. For example, the code can handle the complex temperature and pressure dependence predicted for unimolecular reactions (using the Troe parameterization), important for chemical vapor deposition (CVD) systems, which typically run at sub-atmospheric pressures. Surface site fractions and bulk-phase mole fractions are defined on all reacting surfaces using the SURFACE CHEMKIN package. Through this method, complex Langmuir-Hinshelwood-type and precursor adsorption surface mechanisms, characteristic of many real CVD and catalysis surface systems, can be incorporated into the reacting flow analysis code. The capability of modeling simple dilute species transport and reaction, without the need of linking to CHEMKIN, is also included in MPSalsa. The turbulent transport of species is modeled with a constant Schmidt number approximation when doing RANS type turbulent closure. For LES type modeling, the linear eddy model (LEM) of Kerstein [6] will soon be available (work is still in progress).

The user can extend the models past what has been pre-defined within MPSalsa [1]. Functions can be written to represent additional source terms, special boundary conditions, and variations in physical properties, any of which can be dependent on the current solution, position, or time.

The discretization method is a Petrov-Galerkin finite element method (PGFEM) with pressure stabilization. Both steady and transient flows may be analyzed. The time integration methods include true transient, pseudo-transient, and steady implicit solvers. The overall solution is obtained by fully-coupled, implicit, parallel iterative solvers based on preconditioned nonsymmetric Krylov subspace methods. Presently, MPSalsa can simulate low Mach number (< 0.3) flows, where an algorithm employing an implicit coupling between the pressure and velocity field is required.

MPSalsa employs unstructured grids, using the EXODUS II finite element database suite of programs for its input and output files [7, 8, 9]. Therefore, it can be used in conjunction with the CUBIT mesh generation package [7], as well as other mesh generation packages that support the EXODUS II standard. A number of pre- and post-processing routines for the EXODUS II database can be used. Currently, two- and three-dimensional grids with Cartesian coordinates are supported.

MPSalsa includes both first- and second-order predictor-corrector time integration schemes; these methods use explicit predictors and fully implicit corrector methods based on forward/backward Euler and Adams Bashforth/Trapezoidal rule methods, respectively. At each time step, a prediction of the solution and its time derivative are generated from the appropriate time integration scheme. This prediction is used as the initial guess for the fully coupled non-linear problem generated at each time step. The non-linear problem is solved using an inexact Newton method. At each step of the non-linear problem, a "residual vector" and a "Jacobian matrix" are generated, based on the current solution approximation. The resulting linear problem is solved using iterative methods based on preconditioned Krylov-subspace techniques. The accuracy or convergence criteria for solving the linear subproblem is controlled by the inexact Newton algorithm. This algorithm selects the convergence criteria based on how well the linear subproblems are approximating the underlying nonlinear problem. As is the case with most adaptive ODE integration codes, the accuracy to which the non-linear problem is solved is based on a time-step truncation error estimate. The adaptive time integration method uses a user-specified error tolerance and a time-truncation error estimate from the compatible-order predictor/corrector methods to automatically select time step sizes to control time step truncation error at a user-specified tolerance.

From its inception, MPSalsa has been designed for distributed memory MIMD computers with hundreds to thousands of processors. It also runs on traditional serial workstations and networks of serial workstations. Interprocessor data communication and global synchronization are accomplished by a small number of message passing routines. These routines have been ported to many different message passing protocols, including the MPI standard and the native nCUBE and Intel Paragon protocols. To achieve efficient parallel execution, the unstructured finite element mesh is partitioned or load-balanced in a preprocessing step. Here, each processor is assigned nodes from the mesh such that the computational load is balanced and the total amount of information communicated between neighboring processors is minimized. Each processor is then responsible for calculating updates for all the unknowns at each of its assigned FE nodes. Each processor also stores and performs operations on the rows in the fully-summed, distributed matrix associated with these unknowns. Along processor subdomain boundaries, replicated FE unknowns, called "ghost unknowns," are stored and updated through interprocessor communication. These ghost unknowns, assigned to neighboring processors, are quantities needed for the local residual calculation and matrix-vector multiplication on a processor. Interprocessor communication occurs for each step of the iterative solution of the linear system as well as for each outer step in the non-linear and time-transient algorithms. This communication constitutes the major unstructured interprocessor communication cost in the program, and its algorithm has been extensively optimized within MPSalsa [10].

Solution output from the program is achieved through several means. Output can be written to either a standard serial EXODUS file format [8, 9] or a "parallel extension" of the EXODUS file format [11]. This extension consists of writing an individual standard serial EXODUS file for each processor with an extra array that maps the local node numbering scheme on an individual processor to the global node numbering scheme. The format can be used on both MP computers, such as the Intel Paragon, and distributed computing systems, such as groups of workstations. This parallel I/O capability can be used with today's primitive parallel I/O facilities with nearly linear speedup.

This report serves as an introduction to MPSalsa. A companion user's manual contains a detailed description of the input and solution options, as well as several example problems that have been solved by MPSalsa [1]. The target problem classes of MPSalsa are discussed in Section 2, along with the currently supported material types and equations of state. Section 3 introduces the governing transport-reaction equations. Special sections on the calculation of the multicomponent diffusional fluxes and gas-phase reactions, as well as turbulence models are included as well. The treatment of surface species and surface reaction source terms is also discussed. Subsection 3.9 contains a summary of the bulk transport equations solved within the code. Section 4 contains a general discussion of the implementation of boundary conditions within MPSalsa where boundary conditions specific to each equation are introduced. In Section 5, the finite element implementation of the transport-reaction equations, the supported interpolation functions, quadrature rules, and methodology for calculating surface integrals are introduced. The matrix equations are also presented to display the essential form of the system of coupled equations. Terms included and excluded from the Jacobian matrix are delineated in Appendix C. Section 6 contains the solution methodology at the algorithm level. The parallel implementation of the code is described, and the nonlinear solver and the linear system solvers, along with their respective convergence criteria, are discussed. The algorithmic details of the Aztec library of Krylov solvers and preconditioners are left to companion documents [12].

2. Problem Types and Equations of State

2.1 Problem Types

MPSalsa is designed to solve the governing transport-reaction equations for momentum, total mass, thermal energy, species, and auxiliary turbulence quantities. In addition, MPSalsa allows the user to solve a reasonably general set of coupled transport-reaction equations by specification of general transport coefficients and source terms. The scope of the problem types that a program can handle is determined, in part, by the discretization scheme and solution method. MPSalsa employs a highly coupled approach to the solution of its equation set, by storing all cross terms in the Jacobian. The fully-summed distributed Jacobian is stored so that highly effective general algebraic preconditioners such as ILU with partial fill-in and block ILU factorizations may be used to reduce the total number of iterations in the linear solver. Thus, MPSalsa is most effective on highly coupled problems that require an implicit solution technique. It is less efficient on problems that can be solved with explicit or semi-implicit solution techniques, such as high Mach number flows or weakly coupled systems. Additionally, the filtering of the density by eliminating the hydrodynamic pressure dependence limits the problem classes MPSalsa can currently handle to low Mach number flows. However, within these bounds, the transport-reaction systems and geometric complexity that MPSalsa can handle are quite general.

The determination of which equations are solved, as well as which operators are included, is done by specifying the "problem type." This also determines what types of unknowns are included in the solution vector. Table 2-1 shows the available options for the problem type. As the table points out, diffusion operators are always included, while inclusion of the convection operator depends on the particular problem type. Single, general PDEs that don't fall into any of the categories in Table 2-1 may be handled either with the energy equation/temperature unknown or the species conservation/mass fraction unknown. Systems of general PDEs are handled with the mass species transport equations and can optionally be coupled to the momentum, thermal energy and total mass equations. Each problem type has a default setting for whether the equations are linear or non-linear. MPSalsa contains logic for the efficient handling of both cases. The default linearity setting can be overridden as well.

For heterogeneous or multi-physics problem types, different domains with different material types, such as a solid and an ideal gas, are used. A varying number of transport equations are then solved on each domain. While this type of problem has not been fully implemented in MPSalsa, the underlying data structures are in place. In particular, the matrix storage format, Variable Block Row (VBR) sparse matrix format [13], allows for a different number of equations to be solved for per node.

2.2 Material Properties

The assignment of material properties starts with designating each region, specified by EXODUS II element blocks, with a "material model." Material models are broadly classified within

Table 2-1: Problem Types
x: always on, y: model dependent on

Problem Types	Transport Equation/ Unknowns					Operators Included	
	Energy/ Temperature	Species/ Mass Fraction	Momentum/ Velocity	Mass/ Pressure	Turbulence Unknown(s)	Diffusion	Convection
energy_diff	x					x	
energy_conv_diff	x					x	x
mass_diff		x				x	
energy_mass_diff	x	x				x	
energy_mass_conv_diff	x	x				x	x
stokes_flow			x	x		x	
fluid_flow			x	x		x	x
fluid_flow_energy	x		x	x		x	x
fluid_flow_mass		x	x	x		x	x
whole_enchilada	x	x	x	x		x	x
advection_diff	input	x	input	input	input	x	x
turb_flow			x	x	y	x	x
turb_flow_energy	x		x	x	y	x	x
turb_flow_mass		x	x	x	y	x	x
whole_turb_enchilada	x	x	x	x	y	x	x

MPSalsa as belonging to a "material type" which are listed in Table 2-2. The material type is used extensively within the code for conditional evaluation of equations of state, transport property computations and source terms.

When a CHEMKIN material type is defined in a problem, MPSalsa reads the CHEMKIN binary work arrays produced by CHEMKIN preprocessors. Details of this process can be found in the MPSalsa User's Guide [1]. From these work arrays, MPSalsa obtains the number of gas-phase species, the number of surface phases and surface-phase site fractions, and the number of bulk mole fractions. All gas-phase transport properties are obtained from the TRANLIB library [26], which evaluates gas-phase multicomponent transport properties. The ideal gas equation of state given by Eqn. 1 is used to yield expressions for the density, ρ.

Table 2-2: Material Types

Material Type	Description
CHEMKIN	Ideal Gas - Use the ideal gas mixture equation of state, and calculate transport properties and reaction rates via CHEMKIN.
NEWTONIAN	Newtonian fluid, i.e., has a Newtonian stress tensor formulation. The default is to use constant fluid and transport properties.
BOUSSINESQ	Boussinesq fluid, i.e., a Newtonian fluid with a constant thermal expansion coefficient. Density varies only in the body force term.
SOLID	Bulk solid with isotropic transport properties
NNEWTONIAN	Non-newtonian fluid (not yet implemented)
ANISOTROPIC_SOLID	Material that has an anisotropic thermal conductivity and species diffusivities (not yet fully implemented).

$$P_o = \rho RT \sum_{j=1}^{N_g} \frac{Y_j}{W_j} = \frac{\rho RT}{\sum_{j=1}^{N_g} W_j X_j} \qquad (1)$$

N_g is the number of gas phase species, Y_j is the mass fraction of the j^{th} species, X_j is the mole fraction of the j^{th} species, and W_j is the molecular weight of the j^{th} species. P_o is the thermodynamic pressure.

The CHEMKIN material type assigns a "special species label" to one of the species. The conservation equation for that species is replaced by the condition that the sum of the mass fractions must equal one:

$$\sum_{k=1}^{N_g} Y_k = 1. \qquad (2)$$

The caloric equation of state for an ideal gas mixture is used for CHEMKIN materials. In this model, h, the specific enthalpy of the mixture, does not depend on the total pressure. Eqn. 3 provides the expression for the specific enthalpy in terms of the partial specific enthalpies for each species and the mass fractions. Since an ideal solution is assumed, the partial specific enthalpies are equal to the pure specific enthalpies of each species in its reference state.

$$h = \sum_{j=1}^{N_g} \hat{h}_j(T) Y_j \qquad \hat{h}_j(T) = W_j \Delta H^o_{f,j}(T_o) + \int_{T_o}^{T} \hat{C}_{p,j} dT \qquad (3)$$

$\Delta H^o_{f,j}(T_o)$ is the heat of formation of the j^{th} species in its standard state and at the common reference temperature (which for CHEMKIN is $T_o = 298.15K$). The standard state for gases corresponds to an ideal, pure gas state at 1 atm. Thermodynamic information for the CHEMKIN material type is obtained from the CHEMKIN thermodynamics data base or the CHEMKIN input file. The calculations in Eqn. 3 are carried out within CHEMKIN. $\hat{C}_{p,j}$, the specific heat at constant pressure for species j, is a polynomial function of temperature.

In the NEWTONIAN material type, all transport properties, as well as the density, are assumed constant. This assumption can be overridden by specification of variable properties for a number of the transport properties. In the BOUSSINESQ material type, the default is for the density to be constant in all equations, except for the body force term in the momentum equations. In this term, the density is assumed to be a linear function of the temperature. The density can be expressed in terms of the coefficient of volumetric expansion, $\bar{\beta}$.

$$\rho = \rho(T_0)[1-\bar{\beta}(T-T_0)] \quad , \quad \text{where} \quad \bar{\beta} = -\frac{1}{\rho}\left[\left.\frac{\partial\rho}{\partial T}\right|_{T=T_0}\right]_p \qquad (4)$$

Note that for an ideal gas, $\bar{\beta} = 1/T$, and, thus, it is not a constant. For the BOUSSINESQ material type, $\bar{\beta}$ is supplied by the user.

The SOLID material type is a placeholder set aside for the future anticipated capability to do conjugate heat transfer problems in domains with both solid and fluid regions. These problems have regions where the momentum equations are not solved. Currently, this capability is not available in MPSalsa. In MPSalsa, both constant and variable thermal transport properties can be used. The NNEWTONIAN material type is defined for the specification of non-Newtonian constituitive equations (as well as the required additional Jacobian entries) for viscosity.

The NEWTONIAN, BOUSSINESQ, or SOLID material types can be used if species equations are desired but the CHEMKIN subroutine library for mixtures of ideal gases is not to be used. The default for these non-CHEMKIN materials is to **NOT** enforce Eqn. 2. However, this default can be overridden. The lack of Eqn. 2 represents the situation where all species transport equations represent only dilute components of phases. The majority component of a phase is not represented by a species equation.

For all equation types, there is a capability in MPSalsa for including both volumetric and surface source terms in the residuals and, just as importantly for stiff terms, their Jacobian contributions in the matrix used to relax the equations. Volumetric source terms are specified as part of the materials model using either built-in or user-specified functions. In contrast, surface source terms are specified as surface boundary conditions. They are applied by integrating over surfaces defined in the finite element model. These boundary conditions can also be user-specified functions or built-in functions representing well-known cases, such as those that correspond to convective or radiative heat transfer and sticking coefficient reactions. For boundary conditions at surfaces where deposition or etching of bulk phases occurs, SURFACE CHEMKIN is used to describe the process' kinetics and yield values for surface fluxes of gas-phase species. The capability for solving Stefan flow problems, i.e., problems that have a net normal mass flux at the surface that depends on the surface reaction rate, is built into this "reacting surface" boundary condition.

2.3 Units Within the Program

Non-dimensionalization of the equations is not done within MPSalsa. Except when CHEMKIN is used, no units are *a priori* specified within the program. CHEMKIN produces quantities such as transport properties, densities, pressure, energy, and species rates of production in terms of the CGS units system, i.e., gm, cm, sec, mole, and Kelvin. Therefore, whenever the CHEMKIN material type is used, the user inputs to the program --including boundary condition values -- should also be in CGS units. The specification of the thermodynamic pressure is in atmospheres and the default units for activation energies for gas and surface reaction rates are in cal mole^{-1} for the CHEMKIN material type. When a material type other than CHEMKIN is being used, the user must specify a constant set of units.

Understanding the behavior of a system as a function of non-dimensional numbers, such as the Reynolds number or Grashof number, is a powerful tool. However, this must be carried out by the user indirectly. One way is through use of the continuation routine, where the user can often associate the continuation parameter with a dimensionless group. Another way, which can be seen by comparing the dimensional and non-dimensional formulations of the equations and boundary conditions, is to choose the physical properties such that a single property will represent a dimensionless group; *e.g.*, by setting all other properties to one and using the appropriate domain size and boundary conditions, the gravity unknown will be equivalent to the Rayleigh number. An example of a non-dimensionalization of the equations is provided in the MPSalsa user's manual [1].

2.4 Exact Solutions

MPSalsa is a large code. The use of test problems with known, exact solutions was found to be essential in verifying the code. Much of the code can be checked by comparing numerically derived solutions against exact solutions, and analyzing mesh convergence of numerical solutions. This includes all of the parallelization aspects of the code as well as the implementation of the EXODUS finite element database on multiple processors. For instance, an exact solution to the time-dependent Navier-Stokes equations has been implemented[15]. There are, however, cases where exact solutions are not available to check the validity of the code. Real gases with complicated transport properties are one instance. For these situations, the code was checked against other numerical codes. Two such case studies are included in the user's manual. One case is a comparison of a rotating disk CVD problem to the 1-D numerical code SPIN [14]; the other case is a comparison of a homogeneous, isotropic gas-phase pyrolysis study to the 0-D code SENKIN [15].

3. Governing Transport-Reaction Equations

The equations solved by MPSalsa are based on the governing transport equations for total mass, momentum, energy, individual gas-phase species, and auxiliary turbulence quantities. Constitutive relations for the momentum, heat, and species fluxes are based on one of three models: (a) the non-equilibrium statistical mechanical theory of multicomponent, dilute polyatomic gases [17, 18, 19, 20, 21]; (b) a constant property, Boussinesq fluid model; and (c) constituitive equations supplied and linked in by the user through a set of user subroutines. The Boussinesq fluid approximation is suited to the study of convection in liquids, including liquid metals, while the multicomponent gas model is suitable for a mixture of ideal gases at atmospheric pressures or lower.

The governing transport equations listed below are given in "conservative form" rather than "advective form." In the actual numerical implementation, both the conservative and the nonconservative forms of the equations can be solved. Experience indicates that while greater accuracy is not guaranteed by the conservative formulation, long-time numerical integration stability is enhanced. For this reason, both formulations have been included in the numerical solution procedure, as described in Appendix A.

An acoustically-filtered formulation of the momentum and mass conservation equations is used within MPSalsa [23, 24]. Thus, a distinction between the hydrodynamic and thermodynamic pressure values is employed in the equation set. Variations in the hydrodynamic pressure, which are assumed small compared to the thermodynamic pressure, are not included in the calculation of the density that appears in the conservation of mass, species, and momentum equations. This assumption has been shown to be valid for Mach numbers lower than 0.3 [23] and has the benefit of filtering out shock formation.

3.1 Momentum Transport Equation

The conservation of momentum is expressed by Eqn. 5 and 6. Assuming a Newtonian stress constituitive equation, there are as many scalar components of the momentum equation as there are spatial dimensions in the problem.

$$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla\bullet(\rho\mathbf{u}\mathbf{u}) - \nabla\bullet\mathbf{T} - \sum_{k=1}^{N_g} \rho_k \mathbf{g}_k = 0 \tag{5}$$

$$\text{where} \quad \mathbf{T} = -P\mathbf{I} + \Upsilon = -P\mathbf{I} - \frac{2}{3}\mu_{eff}(\nabla\bullet\mathbf{u})\mathbf{I} + \mu_{eff}[\nabla\mathbf{u} + \nabla\mathbf{u}^{T}] \tag{6}$$

Here, **T** is the stress tensor for a Newtonian fluid, **I** is the unity tensor, Υ is the viscous stress tensor, and P is the isotropic hydrodynamic pressure. In the pressure-filtered formulation, there is a distinction between the hydrodynamic pressure (used in the transport equations) and the pressure level used in the equation of state, P_0. This distinction allows the nearly constant thermodynamic pressure level to be set independently of the relatively small pressure fluctuations due to the hydrodynamic flow. Unlike the treatment in Paolucci [23], there is no global equation for the thermodynamic pressure, P_0, in the equation set. The effective mixture viscosity μ_{eff} is the sum of molecular and turbulent contributions (i.e. $\mu_{eff} = \mu + \mu_t$). Models for the computation of the turbulent contribution are described in Section 3.8. For the molecular contribution (the only contribution for laminar flow) MPSalsa assumes a Newtonian fluid mixture with zero bulk viscosity where μ is a function of the temperature and fluid composition. For a multicomponent ideal gas mixture it is a complex function of the temperature and the species mole fractions with roughly a $T^{0.7}$ dependence on temperature; μ is obtained from a subroutine call to the TRANLIB package [26].

The last term in Eqn. 5 is the body force term where g_k is the sum of all body forces acting on species k, and N_g is the total number of species. In most cases not involving charged particles and/or electromagnetic fields, the body force on each species is the same for all species and reduces to the gravitational force, **g**. In that case, the last term in Eqn. 5 reduces to ρg. Currently, the only body force considered in the code is gravity which is constant for all molecular species. Additional functionality for this term will be application driven.

3.2 Total Mass Conservation Equation

The conservation of total mass within MPSalsa is expressed by Eqn. 7.

$$\frac{\partial \rho}{\partial t} + \nabla \bullet (\rho \mathbf{u}) = 0 \qquad (7)$$

In this equation, ρ is the mass density of the mixture. Two alternate equations of state are allowed for ρ. Either ρ is considered to be a constant (i.e., the incompressible case or the Boussinesq fluid case where ρ is considered to be a constant, except in the body force term), or ρ is calculated from the ideal gas mixture equation of state Eqn. 1. Thus, for an ideal gas, ρ is not a function of the variable hydrodynamic pressure; it is a function of the constant thermodynamic pressure only. Additionally, a user-defined subroutine can be employed to incorporate an alternate equation of state that is dependent on the local temperature and species compositions as well as the thermodynamic pressure.

3.3 Energy Transport Equation

For high speed flows, the conservation equation in the total energy (i.e., the internal energy plus the kinetic energy) form is normally used. This form is particularly useful for inviscid flows. However, the difficulty with this representation is that for flows in which the molecular transport of thermal energy is large, the implicit coupling of the internal energy or enthalpy to the temperature is weak. Given this, for low speed, incompressible flows this equation is generally translated into either the enthalpy or temperature form. These choices work well for the initial class of problems to be addressed by this code - low Mach number CVD problems. In the code, the specific heat/temperature form is implemented. However, future versions of the code may include the enthalpy form as it is natural for control volume formulations in which a local conservation of energy property can be attained.

3.3.1 Temperature Formulation of the Energy Equation

Eqn. 8 is the internal energy equation in terms of the temperature.

$$\hat{C}_p\left[\frac{\partial(\rho T)}{\partial t} + \nabla\bullet(\rho\mathbf{u}T)\right] = -\nabla\bullet\mathbf{q} + \phi + \dot{Q} + \sum_{k=1}^{N_g}\mathbf{j}_k\bullet\mathbf{g}_k + \frac{DP}{Dt}$$
$$\sum_{k=1}^{N_g}\hat{h}_k(\nabla\bullet\mathbf{j}_k) - \sum_{k=1}^{N_g}\hat{h}_k W_k\dot{\omega}_k \qquad (8)$$

In this equation, $\hat{C}_p$ is the specific heat of the mixture at constant pressure. The first term on right hand side is the diffusive heat flux, $\mathbf{q}$, given by Eqn. 9. The second term is the volumetric heat source term from viscous dissipation, ϕ, given by Eqn. 10. The volumetric energy source term $\dot{Q}$ is specified by a user function, and $\mathbf{j}_k$ is the diffusive flux of the k^{th} species relative to the mass-averaged velocity, $\mathbf{u}$. The net change of potential energy from body force terms into heat energy, $\sum\mathbf{j}_k\bullet\mathbf{g}_k$, is zero for the single body force term implemented so far, gravity, because $\mathbf{g}_k$ is equal for all k. The total derivative of pressure, DP/Dt, represents the reversible exchange of mechanical energy into internal energy. The first term on the second line of Eqn. 8 is the production of internal energy due to diffusion, where $\hat{h}_k$ is the partial specific enthalpy of the k^{th} species. The last term in Eqn. 8 is the volumetric production of heat due to chemical reactions using $\dot{\omega}_k$ as the net production rate of the k^{th} species due to homogeneous chemical reaction and W_k as the molecular weight of the k^{th} species.

$$\mathbf{q} = -\lambda_{eff}\nabla T + \sum_{k=1}^{N_g}\hat{h}_k\mathbf{j}_k - \sum_{k=1}^{N_g}\frac{RT}{W_k X_k}D_k^T\mathbf{d}_k + \mathbf{q}_r \qquad (9)$$

$$\phi = -(\tau : \nabla\mathbf{u}) = -\frac{2}{3}\mu(\nabla\bullet\mathbf{u})^2 + \frac{1}{2}\mu\left\|\nabla\mathbf{u} + \nabla\mathbf{u}^T\right\|^2 \qquad (10)$$

The first term in Eqn. 9 is the diffusive flux of energy due to heat conduction. λ_{eff} is the effective heat conductivity of the mixture. The effective heat conductivity is the sum of molecular and turbulent contributions (i.e. $\lambda_{eff} = \lambda + \lambda_t$). Models for the computation of the turbulent contribution are described in Section 3.8. For gases, the molecular contribution (the only contribution for laminar flow) is a complicated isotropic function of the temperature and mass fractions. The second term in Eqn. 9 is the diffusive flux of energy due to species diffusion. The third term is the Dufour effect, the diffusive flux of energy due to thermal diffusion. This term is usually very small and is neglected in the implementation of the code. The last term is the flux of energy due to radiative transport, $\mathbf{q}_r$. It is almost always ignored when solving the gas-phase energy continuity equation; i.e., the gas is assumed to be transparent to radiant energy. However, this term is very important for some applications, such as combustion, and so must be included. MPSalsa has been linked to the SYRINX [25] library to calculate the radiative source terms at every node in the mesh. MPSalsa passes to SYRINX the current temperature field and elemental absorptivities, and SYRINX uses the discrete ordinate approximation to calculate the radiative flux vector. The calculation of this term is often more expensive than all other parts of the reacting flow calculation, and also hinders convergence of Newton's method since all off-diagonal Jacobian entries are not calculated for this non-local term.

For a simple thermodynamic material, the heat flux term from the species diffusive flux and the heat source term originating from the divergence of the species diffusive flux term may be combined to yield a single heat source term due to the diffusive flux, Eqn. 11. This modification is incorporated into Eqn. 8 and 9.

$$-\nabla\bullet\sum_{k=1}^{N_g}\hat{h}_k\mathbf{j}_k+\sum_{k=1}^{N_g}\hat{h}_k\nabla\bullet\mathbf{j}_k = -\sum_{k=1}^{N_g}\mathbf{j}_k\bullet\hat{C}_{p,k}\nabla T \qquad (11)$$

In the initial implementation of the code, some of the terms in Eqn. 8 are not included because of their relatively small contributions. The body-force source term is omitted since the gravity vector, $\mathbf{g}_k$, is equal for all k. The viscous dissipation term and reversible change of mechanical energy into internal energy term (DP/Dt) are dropped since they are small for low Mach number applications. Also, the energy flux terms due to species diffusion, as presented in Eqn. 11, have not yet been included but will be in the near future.

3.3.2 Enthalpy Formulation of the Energy Equation

Eqn. 12 is the conservation of energy equation expressed in terms of the mixture enthalpy, h.

$$\frac{\partial(\rho h)}{\partial t}+\nabla\bullet(\rho\mathbf{u}h) = -\nabla\bullet\mathbf{q}+\phi+\dot{Q}+\frac{DP}{Dt}+\sum_{k=1}^{N_g}\mathbf{j}_k\bullet\mathbf{g}_k \qquad (12)$$

Eqn. 9 and Eqn. 10 are used for $\mathbf{q}$ and ϕ, respectively. The terms in Eqn. 8 due to the volumetric production of heat caused by diffusion and chemical reaction do not appear in Eqn. 12. Therefore, Eqn. 11 is not used to simplify Eqn. 12. The flux of enthalpy due to diffusion in Eqn. 8 must be explicitly evaluated and added to the heat flux caused by conduction in order to determine the total diffusional heat flux. The mixture specific enthalpy can be related to the partial specific enthalpies by Eqn. 13. For ideal gases, the partial specific enthalpy is equal to the pure component enthalpies, which are not functions of the total pressure.

$$h(T,P) = \sum_{k=1}^{N_g}\hat{h}_k(T)Y_k \qquad (13)$$

The dependent variable most easily used with Eqn. 12 is the temperature. If the mixture enthalpy itself were used as the dependent variable, Eqn. 13 would have to be inverted to obtain the temperature. Also, the temperature appears explicitly in Eqn. 9.

Because the total derivative appears on the left hand side of Eqn. 12, the enthalpy can be considered a conserved quantity. Note, this does not occur for Eqn. 8 since $\hat{C}_p$, a complicated function of the temperature and composition, appears outside of the time and convective derivatives. For discretization schemes that employ integral balances over control volumes, such as the control volume finite element methods, local as well as global conservation of ρh can be proven. For the Galerkin finite element method, conservation exists only on a global basis.

3.4 Species Mass Transport Equation

The governing transport-reaction equation for each molecular species mass fraction, Y_k, is expressed by Eqn. 14.

$$\frac{\partial(\rho Y_k)}{\partial t} + \nabla \bullet (\rho \mathbf{u} Y_k) = -\nabla \bullet \mathbf{j}_k + W_k \dot{\omega}_k \qquad k = 1, \ldots, N_g - 1 \tag{14}$$

Here, $\dot{\omega}_k$ is the molar production rate of species k from gas-phase reactions, $\mathbf{j}_k$ is the flux of species k due to diffusion relative to the mass-averaged velocity, $\mathbf{u}$. As described above, for a CHEMKIN material type, there are $N_g - 1$ continuity equations for the molecular species; the continuity equation for the special species is replaced by Eqn. 2, the requirement that the mass fractions Y_k sum to unity. Therefore, that single species in the mechanism employs a different equation to calculate its mass fraction. For the Dixon-Lewis multicomponent diffusion algorithm, this substitution does not cause any loss of accuracy. However, when the mixture-averaged diffusion coefficients are used, the effective continuity equation for the special species may have a different type and generally larger discretization error than other species in the mechanism. An "effective continuity equation" for this species, N_g, can be derived by taking the sum of all species continuity equations (Eqn. 14), $k = 1, \ldots, N_g - 1$, subtracting it from the total continuity equation (Eqn. 7), and then invoking Eqn. 2. To minimize the errors in this "effective continuity equation" for the special species, the special species should be chosen to be the species with the largest mass fraction.

Eqn. 2 doesn't have to be used to ensure that the sum of the mass fraction equals one; it is implied by the continuum equations and by the property that the sum of the diffusion velocities and species mass production rates is zero. This can be seen by summing Eqn. 14 over all species and subtracting the total continuity equation, Eqn. 7. The resulting equation is Eqn. 15.

$$\rho \frac{\partial}{\partial t} \sum_{k=1}^{N_g} Y_k + \rho \mathbf{u} \bullet \nabla \sum_{k=1}^{N_g} Y_k = 0 \tag{15}$$

If Eqn. 2 holds rigorously as an initial condition, Eqn. 15 ensures that the sum remains equal to one everywhere for all time. The presence of reacting surfaces, roundoff error, discretization error, and time-step truncation error, however, changes this result in the numerical problem, necessitating the use of Eqn. 2.

The mass fractions Y_k are the dependent variables solved for in the species conservation equation. However, mole fractions are used for specification of boundary conditions and source terms, $\dot{\omega}_k$. The conversions between mass and mole fractions are shown in Eqn. 16.

$$Y_k = \frac{W_k}{\overline{W}} X_k \qquad \text{where} \qquad \overline{W} = \sum_{k=1}^{N_g} X_k W_k = \frac{1}{\sum_{k=1}^{N_g} Y_k / W_k} \tag{16}$$

Other material types also use Eqn. 14 for the mass transport-reaction equation. However, they default to a different formula for the conversion of mass fraction to mole fraction. For the NEW-

TONIAN, BOUSSINESQ, and SOLID material types, $\overline{W}$ is assumed to be constant. Then, W_k becomes a constant multiplicative factor in Eqn. 14, which can be factored out after some substitutions of definitions. The assumption of constant $\overline{W}$ is appropriate for dilute advection-diffusion of trace species in liquids and solids. When the values of $\overline{W}$ and W_k are defined to be unity, the mole and mass fractions of a species become identical, and the dependent variable in Eqn. 14 can be considered to be the mole fraction.

3.4.1 Diffusion Velocities

In Eqn. 14, $\mathbf{j}_k$ can be written in terms of the diffusion velocity for species k, $\mathbf{V}_k$.

$$\mathbf{j}_k = \rho Y_k \mathbf{V}_k \tag{17}$$

Several different approximations for $\mathbf{V}_k$ are used within MPSalsa depending upon the material type. For the NEWTONIAN, BOUSSINESQ, and SOLID material types, $\mathbf{j}_k$ is expressed by Eqn. 18.

$$\mathbf{j}_k = -\rho D_{k,\,eff} \nabla Y_k, \qquad \text{or} \qquad \mathbf{V}_k = -\frac{D_{k,\,eff}}{Y_k} \nabla Y_k \tag{18}$$

The effective diffusion coefficient $D_{k,\,eff}$ is the sum of molecular and turbulent contributions (i.e. $D_{k,\,eff} = D_k + D_{k,t}$). Models for the computation of the turbulent contribution are described in Section 3.8. For the molecular contribution (the only contribution for laminar flow) the default for these material types is to assume that D_k is constant but the user can override the default and make it a user-specified function of the solution.

For the CHEMKIN material type, two different approximations for the molecular contribution to the diffusion velocity are used in the code: the mixture-averaged diffusion approximation and the Dixon-Lewis formulation [21]. (Note: in the discussion that follows, the flow is assumed to be laminar so the subscript "eff" is not used as was above in Eqn. 18.) In the full Dixon-Lewis formulation, $\mathbf{V}_k$ is expressed in terms of the ordinary multicomponent diffusion coefficients, D_{kj}, and the thermal diffusion coefficient, D_k^T.

$$\mathbf{V}_k = \frac{1}{X_k \overline{W}} \sum_{j \neq k}^{N_g} W_j D_{kj} \mathbf{d}_j - \frac{D_k^T}{\rho Y_k} \frac{\nabla T}{T} \tag{19}$$

In this equation, X_k is the mole fraction for the kth species, and $\mathbf{d}_j$ is the diffusional driving force for the jth species given by Eqn. 20 [27, 27]. Note that $\mathbf{d}_j$ is expressed in terms of the gradient of the mole fractions instead of the mass fractions.

$$\mathbf{d}_j = \nabla X_j + (X_j - Y_j)\frac{\nabla P}{P} + \frac{\rho}{P} \sum_{i=1}^{N_g} Y_j Y_i (\mathbf{g}_i - \mathbf{g}_j) \tag{20}$$

The second term in Eqn. 20 is the pressure diffusion term. Pressure gradients can create driving forces for separation of species with different molecular weights. However, except for applications designed to specifically use this driving force to effect a separation of isotopes, this term is usually negligible compared to other terms. The last term in Eqn. 20 is the driving force for diffusion due to differences in the body forces between species. For neutral gas transport where the only body force is gravity, this term is identically zero. In the initial implementation of the code, only the first term in Eqn. 20 is included. Other terms will be added when warranted by an application.

Values for the ordinary multicomponent diffusion coefficients D_{kj} and the multicomponent thermal diffusivities D_k^T are obtained from library calls to the CHEMKIN transport parameters package [26]. Details concerning their formulation may be obtained from [26]. However, it should be noted here that D_{kj} and D_k^T have the property that the sum of the diffusive fluxes is zero. The full, multicomponent diffusion formulation is extremely expensive and possibly too expensive to be carried out in the two- and three-dimensional applications for which this code is designed. Therefore, the solution strategy concentrates on implementing approximations to the rigorous multicomponent diffusion formulation. A user flag is set to indicate the level of approximation to be used. The full formulation is available, however, to check the accuracy of other approximations with respect to the full multicomponent formulation.

The mixture-averaged diffusion velocity formula, Eqn. 21, does not have the property that the sum of the diffusive fluxes is zero. For two- and three-dimensional applications it is, however, much less expensive. Additionally, it reduces the coupling between species equations, leading to a more efficient iterative solution of the global linear equations.

$$\mathbf{V}_k = -\frac{D_{km}}{X_k}\mathbf{d}_k - \frac{D_k^T}{\rho Y_k}\frac{\nabla T}{T} \qquad (21)$$

3.5 Calculation of Diffusion Velocities (Laminar flow)

As mentioned, the cost of undertaking a full multicomponent diffusion formulation is prohibitive for two- and three-dimensional reacting flow problems. Therefore, several levels of approximation are used by the code which are similar to those used in the 1-D code, SPIN. Each of these approximations calculates the diffusion velocities, $\mathbf{V}_k$, in a different manner by expressing the conservation of species mass density equation for species k in terms of a pseudo-Fickian diffusion coefficient, $\hat{D}_k$, and the thermal diffusion coefficient, D_k^T, as shown in Eqn. 22 and 23.

$$\frac{\partial(\rho Y_k)}{\partial t} + \nabla\bullet(\rho\mathbf{u}Y_k) = -\nabla\bullet(\mathbf{j}_k) + W_k\dot{\omega}_k \qquad (22)$$

where

$$\mathbf{j}_k = \rho Y_k\mathbf{V}_k = -\rho\hat{D}_k\nabla Y_k - D_k^T\frac{\nabla T}{T} \qquad (23)$$

Eqn.'s 22 and 23 assume that the pressure and body-force diffusion terms are negligible. In the limit of a binary mixture or a dilute mixture, $\hat{D}_k$ is equal to the binary diffusion coefficient. The combination of Eqn. 22 and Eqn. 23 has great utility as an approximate form for the Jacobian because it does not require the expensive calculation of all the cross-coupling terms. The Jacobian entries for the row corresponding to an unknown for the mass fraction of species k will be non-zero only for unknowns corresponding to the mass fraction of species k. This assumes that $\hat{D}_k$ is treated as a constant in the calculation of the Jacobian, and that the dependence of ρ on Y_k is also not taken into account. Of course, Jacobian entries corresponding to the reaction term, $\dot{\omega}_k$, will tend to fill in those same entries.Various approximations to the multicomponent diffusion formulation, as well as the rigorous multicomponent diffusion formulation, can now be put in the pseudo-Fickian diffusion form. For example, an expression for $\hat{D}_k$ can be obtained from the full multicomponent diffusion form, Eqn. 19, when forced diffusion and body-force diffusion are negligible.

$$\hat{D}_k = \left[\frac{-1}{\nabla Y_k \bullet \nabla Y_k}\right]\frac{W_k}{\overline{W}^2}\sum_{j \neq k}^{N_g} W_j D_{kj} \nabla X_j \bullet \nabla Y_k \qquad (24)$$

When the full multicomponent diffusion formulation is used, it is expected that Eqn. 19 will be used to calculate the diffusional velocities in the residuals. However, since the multicomponent diffusion velocity has been calculated for evaluation of the residual, $\hat{D}_k$ can be efficiently calculated for use in the Jacobian as follows. If the multicomponent diffusion velocity is represented as $\tilde{\mathbf{V}}_k$, Eqn. 25 defines the pseudo-Fickian diffusion coefficient.

$$\hat{D}_k = \left[\frac{-1}{\nabla Y_k \bullet \nabla Y_k}\right]\left(Y_k \tilde{\mathbf{V}}_k + \frac{D_k^T}{\rho T}\nabla T\right) \bullet \nabla Y_k \qquad (25)$$

In the binary limit, it can be shown from Eqn. 24 that $\hat{D}_1 = \hat{D}_2 = D_{12} = \mathcal{D}_{12}$; the multicomponent diffusion coefficient reduces to the binary diffusion coefficient.

Two simplified approximations to the full multicomponent diffusion formulation that are more computationally economical will now be described. The first approximation is the mixture-averaged diffusion approximation [28, 29]. The second approximation is a more computationally intensive approximate solution of the Stefan-Maxwell equations introduced by Oran and Boris [30]. For steady-state problems, it is expected that the user first obtain a solution to the equations employing the mixture-averaged diffusion approximation. Then, if more accuracy is desired, the full multicomponent diffusion equations may be used. The Stefan-Maxwell equations have not been implemented in the code.

It is expected that the mixture-averaged diffusion coefficient formulation will get the most use in the code. In the mixture-averaged diffusion formulation, Eqn. 19 for $\mathbf{V}_k$ is replaced by Eqn. 26.

$$\mathbf{V}_k = -\frac{D_{km}}{X_k}\mathbf{d}_k - \frac{D_k^T}{\rho Y_k}\frac{\nabla T}{T} \qquad (26)$$

The mixture-averaged diffusion coefficient, D_{km}, can be obtained directly from a call to the CHEMKIN transport library. It is a simple function of the composition and the binary diffusion coefficients, Eqn. 27.

$$D_{km} = \frac{1 - Y_k}{\sum_{j \neq k} X_j / \mathcal{D}_{jk}} \qquad (27)$$

In the above equation, $\mathcal{D}_{jk}$ is the binary diffusion coefficient between species j and k. D_{km} can be formally related to $\hat{D}_k$ by equating expressions for $\mathbf{V}_k$. Assuming that forced diffusion and body-force diffusion are negligible, Eqn. 28 results.

$$\hat{D}_k = D_{km}\frac{W_k}{\overline{W}}\left[\frac{\nabla X_k \bullet \nabla Y_k}{\nabla Y_k \bullet \nabla Y_k}\right] \tag{28}$$

Eqn. 28 is used for $\hat{D}_k$ in formulating the Jacobian needed to relax the residuals when the mixture-averaged diffusion coefficient is used in the residuals. In the binary limit, $\hat{D}_1$ is not equal to D_{1m} ($\hat{D}_1 = (\overline{W}D_{1m})/W_2$), because D_{km} is not equal to the binary diffusion coefficient.

The mixture-averaged diffusion coefficient, D_{km}, has the unfortunate property that it doesn't ensure that the diffusion fluxes sum to zero. Thus, a correction velocity is needed to ensure that this fundamental condition holds [27]. In this approach, the diffusion velocity vector is redefined to be

$$\mathbf{V}_k = \hat{\mathbf{V}}_k + \mathbf{V}^c. \tag{29}$$

$\hat{\mathbf{V}}_k$ is the ordinary diffusion velocity computed by the various methods given above, and $\mathbf{V}^c$ is a constant correction factor (independent of molecular species), defined by Eqn. 30.

$$\mathbf{V}^c = -\sum_{k=1}^{N_g} Y_k\hat{\mathbf{V}}_k \tag{30}$$

The addition of the correction velocity to the diffusive flux expressions either requires additional terms in the Jacobian or the calculation of the entire diffusion term in the Jacobian by numerical differentiation. The current implementation of the code chooses the latter.

3.6 Implementation of Gas Phase Reactions

The gas-phase reaction mechanism enters into MPSalsa through the volumetric production rate for the k^{th} species due to homogeneous chemical reaction, $\dot{\omega}_k$, in the species conservation equations and in the temperature representation of the internal energy conservation equation. $\dot{\omega}_k$ is calculated using the CHEMKIN package [4]. This modular approach to programming complex chemical mechanisms has found a great deal of use in the combustion and CVD community [3, 14, 31, 32] because it allows separation of the specification of a complex reaction mechanism from the programming of the numerical representation of the continuity equations. Additionally, different types of reactions (e.g., reversible and irreversible reactions, unimolecular reactions whose rate constant is parameterized by a Troe form, bimolecular reactions, third body reactions with enhanced third body collision efficiencies, and/or lumped kinetics expressions appropriate for the description of overall combustion processes) may be integrated into the numerical code without having to include complex reaction mechanisms. Moreover, changes to the mechanism do not induce changes in the numerical code, and correspondingly, mechanisms developed for one numerical code may be applied in any other numerical code conforming to the CHEMKIN interface.

The following is a brief review of the formulation of $\dot{\omega}_k$ used by CHEMKIN [4]. Not all of the complexity possible with CHEMKIN will be discussed here. Consider N_R elementary reversible or irreversible reactions involving N_g chemical species that can be represented by Eqn. 31.

$$\sum_{k=1}^{N_g} \nu'_{ki}\chi_k \leftrightarrow \sum_{k=1}^{N_g} \nu''_{ki}\chi_k \qquad i = 1, \ldots, N_R \tag{31}$$

ν'_{ki} is the stoichiometric coefficient of the k^{th} species for the forward direction of the i^{th} gas-phase reaction; it is defined as a non-positive number. ν''_{ki} is the stoichiometric coefficient of the k^{th} species for the reverse direction of the i^{th} gas-phase reaction; it is defined as a non-negative number. The possibility of non-integer stoichiometric coefficients is allowed as long as the reaction satisfies charge and elemental balances. The χ_k represents the chemical symbol for the k^{th} species. The production rate $\dot{\omega}_k$ for the k^{th} species can be written as a summation of the rate-of-progress variables, q_i, for all reactions involving the k^{th} species, Eqn. 32.

$$\dot{\omega}_k = \sum_{i=1}^{N_R} \nu_{ki} q_i \qquad \text{where} \qquad \nu_{ki} = \nu''_{ki} + \nu'_{ki} \tag{32}$$

The default in CHEMKIN is to assume mass action kinetic rate constants. For this case, the rate of progress variable, q_i, for the i^{th} reaction is given by the difference of the forward rates and the reverse rates, expressed by Eqn. 33 where q_i has units of mol cm^{-3} s^{-1}.

$$q_i = k_i^f \prod_{k=1}^{N_g} [X_k]^{|\nu'_{ki}|} - k_i^r \prod_{k=1}^{N_g} [X_k]^{\nu''_{ki}} \tag{33}$$

Here, $[X_k]$ is the molar concentration of the k^{th} species, and k_i^f and k_i^r are the forward and reverse rate constants for the i^{th} reaction, respectively. The forward rate constants for the N_R reactions default to having the following extended Arrhenius temperature dependence:

$$k_i^f = A_i T^{\beta_i} \exp\left(\frac{-E_i}{RT}\right). \tag{34}$$

Other expressions for the reaction rate constants, Eqn. 34, are also allowed, such as fall-off behavior parameterized by a Troe form, Landau-Teller reaction rate forms, and third body reactions. The reverse rate constants k_i^r are generally (but not necessarily) related to the forward rate constants through the concentration-based equilibrium constant for the i^{th} reaction, K_i^c, according to Eqn. 35.

$$k_i^r = \frac{k_i^f}{K_i^c} \tag{35}$$

K_i^c is in turn related to the temperature, the net molar production rate of gas production during the reaction, and the Gibbs free energy of reaction. Thermodynamic information for the equilibrium constant is calculated from CHEMKIN's species thermodynamic information. Thermodynamic information is in a format [33] similar to that used by Gordon and McBride [34] for the thermodynamic database used in the NASA chemical equilibrium program.

Chemical reaction mechanisms usually consist of stiff modes, i.e., reactions which are fast compared to other time scales in the problem. Therefore, it is imperative that the Jacobian terms for $\dot{\omega}_k$ be available. The current procedure is to calculate the $\dot{\omega}_k$ source term only at nodes, in order to reduce the expense of this step. Then, $\dot{\omega}_k$ is interpolated throughout the element using elemental basis functions. The Jacobian contributions for the source terms due to reaction are currently calculated at the nodes in the element via numerical differencing. Details of their implementation are discussed in Appendix A.

3.7 Implementation of Surface Phase Reactions

Surfaces where reactions take place create additional source and sink terms for gas-phase species. The boundary conditions for the gas-phase continuity equations for species must specify the total flux of the species at the domain interface. For the case where the interface is stationary and the growth or etching due to surface reactions can be considered not to move the interface, this boundary condition for species k can be expressed by Eqn. 36.

$$\mathbf{n} \bullet (\rho \mathbf{u} Y_k + \mathbf{j}_k) = -\dot{s}_k W_k \tag{36}$$

The left side of Eqn. 36 represents the total flux of species k, both convective and diffusive. The first term on the left-hand side is the Stefan flux term where $\mathbf{n}$ is the outward facing normal to the domain and $\mathbf{j}_k$ represents the net diffusive flux of k from all diffusive processes, including thermal diffusion. The right-hand side represents the net destruction of gas-phase species k due to chemical reaction. Therefore, $\dot{s}_k$ represents the net molar production rate of gas-phase species k due to chemical reaction. Integration by parts, carried out in the Galerkin formulation (discussed in Appendix A), leads naturally to surface integrals of the normal component of $\mathbf{j}_k$ multiplied by the nodal basis functions. Thus, in applying boundary conditions to the k^{th} gas species continuity equation, the normal component of the diffusive flux for species k is replaced by the right hand side of Eqn. 37.

$$\mathbf{n} \bullet \mathbf{j}_k = -\dot{s}_k W_k - \mathbf{n} \bullet \rho \mathbf{u} Y_k \tag{37}$$

Eqn. 37 can be further simplified by summing Eqn. 36 over all gas-phase species and using the property that diffusive fluxes must sum to zero to yield an expression for the Stefan flow, Eqn. 38. Eqn. 38 can then be used in Eqn. 37 to yield Eqn. 39.

$$\mathbf{n} \bullet \rho \mathbf{u} = -\sum_{k=1}^{N_g} \dot{s}_k W_k \tag{38}$$

$$\mathbf{n} \bullet \mathbf{j}_k = -\dot{s}_k W_k + \left[\sum_{k=1}^{N_g} \dot{s}_k W_k \right] Y_k \tag{39}$$

Eqn. 39 is used within MPSalsa for specification of boundary conditions for gas-phase species equations for the case of a reacting surface. Additionally, Eqn. 38 is used for specification of the normal boundary condition for the momentum equation. The tangential boundary condition for the momentum equation for reacting surfaces is set to the no-slip condition. Thus, the problem is reduced to the calculation of $\dot{s}_k, k = 1, 2, \ldots, N_g$. For CHEMKIN material types, $\dot{s}_k$ is supplied by the SURFACE CHEMKIN package [5]. However, they are functions of additional unknowns corresponding to surface site fractions of surface phases and bulk mole fractions of bulk phases where each surface phase represents a different type of surface site and each bulk phase represents a different type of bulk mixture. (The reader is referred to the manual for the SURFACE CHEMKIN package [5] and to the manual for the Surface PSR program [35] for a more complete description.) Thus, the calculation of $\dot{s}_k$ demands the solution of a subproblem at each node on the reacting surface to calculate the values of the extra unknowns corresponding to the state of the surface. The resulting non-linear system of equations is solved using Newton iteration. Since the subproblem is solved at each node, it is completely local to a processor and, thus, requires no additional communication when run on parallel computers. We now describe the equations that comprise this subproblem.

Let $Z_k(n)$ be the surface site fraction of the k^{th} surface species in the n^{th} surface phase. Let Γ_n be site density for the n^{th} surface phase (e.g. mol cm^{-2}). Let $c_k(n)$ be the concentration of the k^{th} surface species in the n^{th} surface phase (e.g. mol cm^{-2}). Then, Eqn. 40 is the conservation equation expressing the continuity balance for the k^{th} surface species in the n^{th} surface phase.

$$\frac{d(AW_k c_k(n))}{dt} = AW_k \dot{s}_k, k = K_s^f(n), \ldots, K_s^l(n), n = 1, \ldots, N_{phase}^{surf} \tag{40}$$

Here, $\dot{s}_k$ is the production rate from surface reactions for the k^{th} surface species, A is the surface area, and W_k is the molecular weight of the k^{th} surface species. $K_s^f(n)$ and $K_s^l(n)$ are the indices for the first and last surface species in the n^{th} surface phase, respectively. Also, $c_k(n)$ can be related to $Z_k(n)$ by Eqn. 41.

$$c_k(n) = \frac{\Gamma_n Z_k(n)}{\sigma_k} \tag{41}$$

Here, σ_k is the number of surface sites the k^{th} species covers. Substituting Eqn. 41 into Eqn. 40 and assuming A is not a function of time yields the equation for $Z_k(n)$ as a function of time. In general, Γ_n can also be a function of time and this must also be taken into account.

$$\Gamma_n \frac{dZ_k(n)}{dt} = \sigma_k \dot{s}_k - \sigma_k \frac{d\Gamma_n}{dt}, k = K_s^f(n), \ldots, K_s^l(n), n = 1, \ldots, N_{phase}^{surf} \tag{42}$$

For any valid surface mechanism, the following equation also holds true for each surface phase n, regardless of the $Z_k(n)$ used.

$$\sum_{k = K_s^f(n)}^{K_s^l(n)} \sigma_k \dot{s}_k = \frac{d\Gamma_n}{dt} \tag{43}$$

Eqn. 43 is called the surface site conservation equation. For most reaction mechanisms, the right-hand side of Eqn. 43 is identically zero. If this is not the case, MPSalsa expands the solution vector at each surface to include Γ_n and uses Eqn. 43 to solve for the concentration of surface sites for phase n as a function of time.

On each surface, the sum of the surface site fractions must equal one.

$$\sum_{k = K_s^f(n)}^{K_s^l(n)} Z_k(n) = 1, n = 1, ..., N_{phase}^{surf} \tag{44}$$

This implies that the use of Eqn. 40 leads to a singular Jacobian for the steady state case, if used for all surface species site fractions in a surface phase. Thus, one of the surface species balance equations, Eqn. 40, is replaced with Eqn. 44 for each surface phase. This has the disadvantage that all the numerical round-off error is assigned to that one equation. Therefore, the equation corresponding to the species with the largest site fraction in the surface phase is replaced by Eqn. 44.

The amount of material in bulk phases within the domain may not be in steady state; i.e., the bulk phases may be growing or etching (although their growth/etch rate is not assumed to affect either the volume or surface area within the domain). MPSalsa treats the mole fractions of bulk-phase species as well as their growth/etch rates as unknowns to be solved for. The format of these equations depend on whether the bulk phase is growing or being etched.

The following equations apply to a growing phase. In this case, the growth rate of the n^{th} bulk phase, $\mathcal{G}(n)$, can be expressed by the following equation:

$$\frac{d[AL_nC_b(n)]}{dt} = A \sum_{k = K_b^f(n)}^{K_b^l(n)} \mathcal{G}_k(n) = A\,\mathcal{G}(n) \tag{45}$$

$$\text{where} \quad \mathcal{G}_k(n) = \text{Max}(\dot{s}_k, 0)$$

In this equation, L_n is the film thickness for the n^{th} bulk phase. $C_b(n)$ is the average molar concentration of the n^{th} bulk phase; it has units of mol cm^{-3}, $\mathcal{G}_k(n)$ is the growth rate of the k^{th} species in the n^{th} bulk phase, and $\dot{s}_k$ is the production rate of the k^{th} species returned from SURFACE CHEMKIN. It is a function of the gas phase concentrations, pressure, temperature, surface site concentrations, and the bulk phase activities. Having $\dot{s}_k$ less than zero for some species, while it is greater than zero for other species is not appropriate for a growing bulk phase. One positive value of $\dot{s}_k$ for a bulk phase signals that particular phase is growing.

For a growing phase, $X_k^b(n)$, the instantaneous mole fraction of the k^{th} bulk-phase species in the n^{th} bulk phase, is determined from the relative growth rates of all species in that phase, Eqn. 46.

$$0 = \mathcal{G}_k(n) - X_k^b(n)\,\mathcal{G}(n) \tag{46}$$

The condition $\text{Max}(\dot{s}_k, 0)$ may violate the overall elemental balance condition. However, in practice, this does not occur because $X_k^b(n)$ for such a species is set to zero by Eqn. 46. Then, only nonphysical mechanisms involving zeroth-order destruction of a bulk species could possibly create the situation where $\dot{s}_k < 0$ and $X_k^b(n) = 0$.

If all $\dot{s}_k$ for a particular bulk phase are less than zero, that bulk phase is undergoing etching. The user can specify whether a particular phase is expected to be etched and MPSalsa solves a different set of equations for the bulk-phase components for that bulk phase. In this case, the user must also supply the initial composition of the bulk phase to be etched. The time-dependent equations used for the bulk-phase mole fractions and etch rates in the n^{th} bulk phase undergoing etching are then given by Eqns. 47 and 48.

$$0 = X_k^b(n)_{\text{INITIAL}} - X_k^b(n) \tag{47}$$

$$\mathcal{G}_k(n) = \dot{s}_k \tag{48}$$

Here, $X_k^b(n)_{\text{INITIAL}}$ is the user-supplied initial estimate for the mole fraction of species k in bulk phase n, assumed to be normalized so that the sum over all bulk-phase species is one. The idea is that the initial phase is being etched away congruently. Incongruent etching, within the context of a single phase, is not allowed, at least at the level where it affects the concentrations of bulk species.

In order to specify the thermodynamic information needed for bulk phases, the activities a_k^b of the bulk-phase components must be determined. These are the quantities in SURFACE CHEMKIN that appear in the rate expressions for surface reactions. This is done within the code by calling a subroutine that users can modify to specify their own relationships between the bulk activities and the bulk mole fractions, temperature, and pressure. The default subroutine assumes a perfect solution relationship for all bulk phases, Eqn. 49, that almost never occurs in practice.

$$a_k^b(T, P, X_k^b(n)) = X_k^b(n) \tag{49}$$

In summary, the extra unknowns, $Z_k(n)$, $\Gamma_n(n)$, $X_k^b(n)$, and $\mathcal{G}_k(n)$ are not included in the formal solution vector. Instead, a separate subproblem is solved for these unknowns as part of the calculation of the residual and Jacobian entries for the gas-phase problem. The two problems are coupled at the gas-species flux level, Eqn. 39. The surface subproblem depends on the gas-phase concentrations at the surface, while the main gas-phase species problem depends on the fluxes calculated from the surface subproblem. An advantage of this approach is that the surface subproblem calculation can be protected from nonphysical occurrences, such as negative gas-phase mole fractions, and made more robust than it would be if lumped in with the main problem. Also, advanced surface profile simulators may be incorporated into MPSalsa at a later date. These simulators model behavior at the micron feature size, and couple into "reactor simulators" such as MPSalsa, which model behavior at the centimeter or meter feature size, through the gas-phase flux boundary condition described above. Solving a separate subproblem for $Z_k(n)$ and $X_k^b(n)$, however, can create some concerns. For time-dependent reacting flow problems, difficulties typically associated with operator splitting techniques arise if the surface unknowns are allowed to have true time dependence (i.e., if they are not assumed to have a faster transient than the bulk and, thus, are assumed to be in pseudo-steady state at each time step of the gas-phase problem).

3.8 Turbulence Model Transport Equations

As described in Section 2.1, both RANS and LES type turbulence models have been implemented into MPSalsa. The root difference between these two approaches lies in the method used to filter the Navier-Stokes equations. However, in either case, revised forms of Eqn. 5 and 6 must be solved, and additional transport equations may also be required.

3.8.1 Time Averaged Navier-Stokes Equations

Strictly speaking, a RANS model is based on the well known Reynolds averaged Navier-Stokes equations, which are derived by decomposing each physical quantity ϕ (e.g. velocity, density, pressure, etc.) into mean and fluctuating components,

$$\phi = \bar{\phi} + \phi' \qquad (50)$$

where

$$\bar{\phi} = \lim_{\Delta t \to \infty} \frac{1}{\Delta t} \int_{t_0}^{(t_0 + \Delta t)} \phi(t) dt, \qquad (51)$$

and substituting into the Navier-Stokes equations. However, for variable density flows a mass weighted variation of this averaging procedure is strongly preferred. In this approach, sometimes called Favre averaging, a mass weighted mean velocity is defined as follows

$$\tilde{u}_i = \frac{\overline{\rho u_i}}{\bar{\rho}}. \qquad (52)$$

Using these definitions the following revised form of the momentum equation can be derived.

$$\frac{\partial(\bar{\rho}\tilde{\mathbf{u}})}{\partial t} + \nabla \bullet (\bar{\rho}\tilde{\mathbf{u}}\tilde{\mathbf{u}}) - \nabla \bullet \bar{\mathbf{T}} - \sum_{k=1}^{N_g} \bar{\rho}_k \mathbf{g}_k = 0 \qquad (53)$$

$$\text{where} \quad \bar{\mathbf{T}} = -\bar{P}\mathbf{I} + \bar{\Upsilon} + \bar{\Upsilon}^{turb} = -\bar{P}\mathbf{I} - \frac{2}{3}\mu(\nabla \bullet \tilde{\mathbf{u}})\mathbf{I} + \mu[\nabla\tilde{\mathbf{u}} + \nabla\tilde{\mathbf{u}}^{\mathrm{T}}] - \overline{\bar{\rho}\mathbf{u}'\mathbf{u}'} \qquad (54)$$

Note that the form of these equations is identical to Eqns 5 and 6 except for the addition of an extra term in the stress tensor $\bar{\Upsilon}^{turb}$ (usually called the mass weighted turbulent stress tensor). The role of a RANS turbulence model is to provide a means of calculating this term, which is required to close the equations.

3.8.2 The Boussinesq Eddy Viscosity Approximation

A common starting point for many RANS turbulence models is to assume a linear relationship between the turbulent stress and the mean rate of strain. This is often referred to as Boussinesq's eddy-viscosity concept and can be written as

$$-\bar{\rho}\overline{\mathbf{u}'\mathbf{u}'} = \mu_t[\nabla\tilde{\mathbf{u}} + \nabla\tilde{\mathbf{u}}^T] - \frac{2}{3}[\mu_t(\nabla\bullet\tilde{\mathbf{u}}) + \rho k]\mathbf{I} \qquad (55)$$

where μ_t is called the turbulent or eddy viscosity, and k is the turbulence kinetic energy defined as

$$k = \frac{1}{2}(\overline{\mathbf{u}'\mathbf{u}'})\mathbf{I} \qquad (56)$$

When the Boussinesq approximation is used, it is often useful to absorb the $2/3\rho k$ part of the relationship into the pressure so that it is not necessary to explicitly calculate k. Thus the application of this approximation allows the overall stress tensor given by Eqn. 54 to be written as

$$\bar{\mathbf{T}} = -\bar{P}\mathbf{I} - \frac{2}{3}(\mu + \mu_t)(\nabla\bullet\tilde{\mathbf{u}})\mathbf{I} + (\mu + \mu_t)[\nabla\tilde{\mathbf{u}} + \nabla\tilde{\mathbf{u}}^T] \quad , \qquad (57)$$

where the pressure is understood be the sum of the mean static pressure and 2/3k.

3.8.3 The Gradient-Diffusion Approximation for RANS Scalar Transport

Using the mass weighted time averaging methods described above, a turbulent transport equation for a generic scaler quantity ϕ can be written as

$$\frac{\partial(\bar{\rho}\tilde{\varphi})}{\partial t} + \nabla\bullet(\bar{\rho}\tilde{\mathbf{u}}\tilde{\phi}) = \nabla\bullet(\Gamma_\phi\nabla(\tilde{\phi}) - \overline{\rho\mathbf{u}'\phi'}) + \bar{S}_\phi \qquad (58)$$

where Γ_ϕ is a generic diffusion coefficient and S_ϕ is a generic source/sink term. The gradient diffusion approximation states that the turbulent flux of scalar quantities can be written as

$$\overline{\rho\mathbf{u}'\phi'} = -\frac{\mu_t}{\sigma_t}(\nabla\tilde{\phi}). \qquad (59)$$

where σ_t is the turbulent Prandtl/Schmidt number for ϕ. Applying this approximation, the turbulent transport equation for a generic scalar quantity ϕ can be written as

$$\frac{\partial(\bar{\rho}\tilde{\varphi})}{\partial t} + \nabla\bullet(\bar{\rho}\tilde{\mathbf{u}}\tilde{\phi}) = \nabla\bullet\left(\left(\Gamma_\phi + \frac{\mu_t}{\sigma_t}\right)\nabla(\tilde{\phi})\right) + \bar{S}_\phi \qquad (60)$$

3.8.4 The Spalart-Allmaras One-equation RANS Turbulence Model

A fairly recent innovation in the development of one-equation turbulence models is the model of Spalart and Allmaras [2]. In this model, the Boussinesq eddy viscosity approximation is invoked, and a transport equation for the turbulent viscosity is developed. Here we provide a functional synopsis of the equations and relationships of this model, and refer the reader to reference [2] for a definition and justification of these terms and relationships.

The turbulent viscosity is related to the Spalart-Allmaras viscosity variable $\hat{\nu}$ as follows.

$$\frac{\mu_t}{\rho} = \nu_t = \tilde{\nu} f_{v1} \tag{61}$$

where

$$f_{v1} = \left(\frac{\chi^3}{\chi^3 + (C_{v1})^3}\right), \tag{62}$$

$$\chi \equiv \frac{\tilde{\nu}}{\nu}. \tag{63}$$

To find $\hat{\nu}$, the following transport equation is solved.

$$\frac{\partial \tilde{\nu}}{\partial t} + \nabla \bullet (\mathbf{u}\tilde{\nu}) = C_{b1}\tilde{S}\tilde{\nu} + \frac{1}{\sigma}[\nabla \bullet ((\nu + \tilde{\nu})\nabla\tilde{\nu}) + C_{b2}(\nabla\tilde{\nu})^2] - C_{w1} f_w \left[\frac{\tilde{\nu}}{d}\right]^2. \tag{64}$$

The additional empirical functions used in the model are defined as follows

$$f_{v2} = 1 - \left(\frac{\chi}{1 + \chi f_{v1}}\right) \tag{65}$$

$$f_w = g\left[\frac{1 + (C_{w3})^6}{g^6 + (C_{w3})^6}\right]^{\frac{1}{6}} \tag{66}$$

$$g = \tau + C_{w2}(\tau^6 - \tau) \tag{67}$$

$$\tau = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 \langle d_n \rangle^2} \tag{68}$$

where d_n denotes the distance to the nearest wall, and

$$\tilde{S} = \sqrt{\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right)} + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}. \tag{69}$$

The values of the constants in the model are

$\sigma = 2/3$, $\kappa = 0.41$, $C_{b1} = 0.1355$, $C_{b2} = 0.622$, $C_{v1} = 7.1$

$C_{w1} = \frac{C_{b1}}{\kappa^2} + \frac{(1 + C_{b2})}{\sigma} = 3.239$, $C_{w2} = 0.3$, and $C_{w3} = 2.0$.

3.8.5 The Standard High Reynolds Number k-ε RANS Turbulence Model

Since its introduction in 1972 by Jones and Launder, the k-ε two-equation turbulence model has become probably the most well known and heavily used (or abused some might assert) turbulence model available. Although descriptions of this model are available from many sources, a brief description is presented here.

The Boussinesq eddy viscosity approximation is invoked to provide a definition of the turbulent viscosity. The turbulent viscosity is computed from the following expression

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \tag{70}$$

where k is the turbulent kinetic energy, and ε is the turbulence dissipation rate.
The transport equations for k and ε are

$$\frac{\partial(\rho k)}{\partial t} + \nabla \bullet (\rho \mathbf{u} k) = \nabla \bullet \left(\left(\mu + \frac{\mu_t}{\sigma_k} \right) \nabla k \right) + P_k - \rho\varepsilon \tag{71}$$

$$\frac{\partial(\rho \varepsilon)}{\partial t} + \nabla \bullet (\rho \mathbf{u} \varepsilon) = \nabla \bullet \left(\left(\mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \nabla \varepsilon \right) + \frac{\varepsilon}{k}(C_1 P_k - C_2 \rho\varepsilon) \tag{72}$$

where the production of turbulent kinetic energy P_k can be written as

$$P_k = \left[\mu_t [\nabla \tilde{\mathbf{u}} + \nabla \tilde{\mathbf{u}}^{\mathrm{T}}] - \frac{2}{3}[\mu_t (\nabla \bullet \tilde{\mathbf{u}}) + \rho k]\mathbf{I} \right] \bullet \nabla \tilde{\mathbf{u}}. \tag{73}$$

The constants for the "standard" high Reynolds number k-ε model are:

$$C_\mu = 0.09,\ \sigma_k = 1.0,\ \sigma_\varepsilon = 1.3,\ C_1 = 1.44, \text{and } C_2 = 1.92.$$

3.8.6 Low Reynolds Number k-ε RANS Turbulence Models

In regions adjacent to solid walls, the character of turbulent motions is significantly altered. To properly account for this region, additional modifications must be made to the turbulent transport equations. This is usually done by the introduction of so called low Reynolds number (LRN) functions. Although many different proposals have been suggested for introducing LRN functions into the k-ε turbulence model, Patel et al. [36] have shown that it is possible to generalize these variations by writing the basic equations in a manner to be described here. The turbulent viscosity is computed from the following expression

$$\mu_t = \rho C_\mu f_\mu \frac{k^2}{\hat{\varepsilon}} \tag{74}$$

and the k-ε transport equations are written as follows

$$\frac{\partial(\rho k)}{\partial t} + \nabla \bullet (\rho \tilde{\mathbf{u}} k) = \nabla \bullet \left(\left(\mu + \frac{\mu_t}{\sigma_k} \right) \nabla k \right) + P_k - \rho(\hat{\varepsilon} + D) \tag{75}$$

$$\frac{\partial(\rho \hat{\varepsilon})}{\partial t} + \nabla \bullet (\rho \tilde{\mathbf{u}} \hat{\varepsilon}) = \nabla \bullet \left(\left(\mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \nabla \hat{\varepsilon} \right) + \frac{\hat{\varepsilon}}{k}(f_1 C_1 P_k - f_2 C_2 \rho \hat{\varepsilon}) + E. \tag{76}$$

Here, f_1, f_2, and f_μ are LRN functions which modify the flow in the near wall region but are equal to 1 away from the wall. D and E represent other empirical functions that might be used in a model. The top hat symbol has been placed over ε so that differences between the meaning of ε used by various models can be distinguished. By definition,

$$\varepsilon = \hat{\varepsilon} + D. \tag{77}$$

Two LRN k-ε turbulence models that are currently implemented into MPSalsa are the Launder-Sharma model [37] and the Lars Davidson model [38]. These models can be summarized as follows

	Launder-Sharma	Lars Davidson
$f_\mu =$	$\exp\left(\frac{3.4}{[1+0.02Re_t]^2}\right)$	$\exp\left(\frac{3.4}{[1+0.02Re_t]^2}\right)$
$f_1 =$	1.0	$1+\left(\frac{0.14}{f_\mu}\right)^3$
$f_2 =$	$[1-0.3\exp(-Re_t^2)]$	$[1-0.27\exp(-Re_t^2)][1-\exp(-Re_n)]$
D =	$2\frac{\mu}{\rho}\left(\nabla k^{\frac{1}{2}} \bullet \nabla k^{\frac{1}{2}}\right)$	0
E =	$2\frac{\mu\mu_t}{\rho}(\nabla^2\tilde{\mathbf{u}})^2$	0

where

$$Re_t = \frac{\rho k^2}{\mu \varepsilon}, \tag{78}$$

$$Re_n = \frac{\rho \sqrt{k} d_n}{\mu}, \tag{79}$$

and d_n is the normal distance to the wall.

3.8.7 Spatially Filtered Navier-Stokes Equations

In LES turbulence modeling, the flow variables are also decomposed into two parts. But in this approach the meaning of the decomposition is different. In LES the decomposition is into a spatially filtered component and a subgrid-scale component,

$$\phi = \bar{\phi} + \phi' . \tag{80}$$

where ϕ denotes a generic physical quantity, and the filtering operation is defined as

$$\bar{\phi}(x, t) = \int_D \phi(z, t) G(x - z, \Delta) dz . \tag{81}$$

Here, G is the filter kernel, D is the domain of the flow and Δ is the filter width in each spatial direction. The filtering operation is normalized so that

$$\int_D G(x - z, \Delta) dz = 1 \tag{82}$$

and in general it is a desired property that G(-z) = G(z). Various types of filter kernels have been employed in the literature (e.g. the top-hat filter, the gaussian filter, etc.), and the details of this aspect of LES modeling will not be discussed here. However, it is important to note that contrary to traditional Reynolds time averaging $\overline{(\bar{\phi})} \neq \bar{\phi}$ and in general, $\overline{\phi'} \neq 0$.

In LES modeling, "mass weighted" or Favre filtered variables can be defined in much the same way as was done for RANS modeling. In LES we define a Favre filtered variable as

$$\tilde{\phi} = \frac{\overline{\rho\phi}}{\bar{\rho}} . \tag{83}$$

Using these definitions the following mass weighted LES equations of motion can be derived;

$$\frac{\partial(\bar{\rho}\tilde{\mathbf{u}})}{\partial t} + \nabla \bullet (\bar{\rho}\tilde{\mathbf{u}}\tilde{\mathbf{u}}) - \nabla \bullet \bar{\mathbf{T}} - \sum_{k=1}^{N_g} \bar{\rho}_k \mathbf{g}_k = 0 \tag{84}$$

where

$$\bar{\mathbf{T}} = -\bar{P}\mathbf{I} + \bar{\Upsilon} + \bar{\Upsilon}^{sgs} = -\bar{P}\mathbf{I} - \frac{2}{3}\mu(\nabla \bullet \tilde{\mathbf{u}})\mathbf{I} + \mu[\nabla\tilde{\mathbf{u}} + \nabla\tilde{\mathbf{u}}^{\mathrm{T}}] + \bar{\rho}\left(\widetilde{\mathbf{u}\mathbf{u}} - \tilde{\mathbf{u}}\tilde{\mathbf{u}}\right) . \tag{85}$$

The form of these equations is identical to Eqns. 5 and 6 except for the notable addition of an extra term in the stress tensor, often referred to as the subgrid turbulent stress tensor.

$$\bar{\Upsilon}^{sgs} = \bar{\rho}\left(\widetilde{\mathbf{uu}} - \tilde{\mathbf{u}}\tilde{\mathbf{u}}\right). \tag{86}$$

The role of an LES subgrid turbulence model is to provide a means of calculating this term which is necessary for closure.

3.8.8 The Smagorinsky Subgrid Turbulence Model for LES

The first subgrid model for LES was introduced by Smagorinsky [39] in 1963 and it remains, together with its variants, a widely applied model. In this model the subgrid stresses are set proportional to the strain rates of the resolved field

$$\bar{\Upsilon}^{sgs} = \mu_t[\nabla\tilde{\mathbf{u}} + \nabla\tilde{\mathbf{u}}^{\mathrm{T}}] + \frac{1}{3}\bar{\Upsilon}^{sgs}\mathbf{I} \tag{87}$$

and the sub-grid turbulent viscosity μ_t is calculated as

$$\mu_t = \bar{\rho}(C_s\Delta)^2\left|\frac{1}{2}(\nabla\tilde{\mathbf{u}} + \nabla\tilde{\mathbf{u}}^{\mathrm{T}})\right|. \tag{88}$$

Here, C_s is the Smagorinsky constant (normally set to 0.1 in MPSalsa) and Δ is the filter width.

If a pseudo pressure is defined as

$$\bar{P}^{LES} = \bar{P} + \frac{2}{3}\mu(\nabla\bullet\tilde{\mathbf{u}}) + \frac{1}{3}\bar{\Upsilon}^{sgs}\mathbf{I} \tag{89}$$

then the overall shear stress tensor can be rewritten as

$$\bar{\mathbf{T}} = -\bar{P}^{LES}\mathbf{I} + (\mu + \mu_t)[\nabla\tilde{\mathbf{u}} + \nabla\tilde{\mathbf{u}}^{\mathrm{T}}]. \tag{90}$$

3.8.9 An LES Subgrid Turbulent Kinetic Energy Model

Subgrid turbulent kinetic energy models have been proposed by Schumann [40] and Yoshizawa [41] and the model described here is similar to these models. In this model the subgrid stresses are set proportional to the strain rates of the resolved field as per Eqn. 87. Thus just like the Smagorinsky model this model is an eddy viscosity model. However, in this model no assumption about the equivalence of production and dissipation are made and therefore nonequilibrium effects are accounted for in this model.

The subgrid turbulent viscosity μ_t is calculated as

$$\mu_t = \bar{\rho}C_v\Delta\sqrt{k_{sgs}} \tag{91}$$

where C_v is a model constant (set here equal to 0.0854), Δ is the filter width, and k_{sgs} is the subgrid scale kinetic energy defined as

$$k_{sgs} = \frac{1}{2}\left(\widetilde{\mathbf{u}\mathbf{u}} - \tilde{\mathbf{u}}\tilde{\mathbf{u}}\right)\mathbf{I}. \tag{92}$$

k_{sgs} is calculated by solving the following transport equation

$$\frac{\partial(\bar{\rho}k_{sgs})}{\partial t} + \nabla \bullet (\bar{\rho}\tilde{\mathbf{u}}k_{sgs}) = \nabla \bullet \left(\left(\mu + \frac{\mu_t}{\sigma_k}\right)\nabla k_{sgs}\right) + P_k - \bar{\rho}C_\varepsilon \frac{(k_{sgs})^{\frac{3}{2}}}{\Delta} \tag{93}$$

where C_ε is a model constant (set here to 0.916), and the production term P_k is model as

$$P_k = \left\{\mu_t[\nabla\tilde{\mathbf{u}} + \nabla\tilde{\mathbf{u}}^{\mathrm{T}}] - \left(\frac{2}{3}\bar{\rho}k_{sgs}\right)\mathbf{I}\right\} \bullet \nabla\tilde{\mathbf{u}}. \tag{94}$$

3.9 Summary of Transport Equations Implemented

Mixture Momentum:

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla\bullet(\rho\mathbf{u}\mathbf{u}) - \nabla\bullet\mathbf{T} - \rho\mathbf{g} = 0 \tag{95}$$

$$\mathbf{T} = -P\mathbf{I} - \frac{2}{3}\mu_{eff}(\nabla\bullet\mathbf{u})\mathbf{I} + \mu_{eff}[\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}}] \tag{96}$$

Mixture Continuity:

$$\frac{\partial\rho}{\partial t} + \nabla\bullet(\rho\mathbf{u}) = 0 \tag{97}$$

Thermal Energy:

$$\begin{aligned}\hat{C}_p\left[\frac{\partial(\rho T)}{\partial t} + \nabla\bullet(\rho\mathbf{u}T)\right] &= -\nabla\bullet\mathbf{q}_c + \phi + \dot{Q} - \sum_{k=1}^{N_g}\mathbf{j}_k\bullet\hat{C}_{p,k}\nabla T \\ &\quad - \sum_{k=1}^{N_g}h_k W_k\dot{\omega}_k - \nabla\bullet\mathbf{q}_r\end{aligned} \tag{98}$$

$$\phi = -\frac{2}{3}\mu(\nabla\bullet\mathbf{u})^2 + \frac{1}{2}\mu\|\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathrm{T}}\|^2 \tag{99}$$

$$\mathbf{q}_c = -\lambda_{eff}\nabla T \tag{100}$$

Species Continuity:

$$\frac{\partial(\rho Y_k)}{\partial t} + \nabla\bullet(\rho\mathbf{u}Y_k) = -\nabla\bullet\mathbf{j}_k + W_k\dot{\omega}_k \tag{101}$$

$$\mathbf{j}_k = \rho Y_k\mathbf{V}_k \tag{102}$$

$$\mathbf{V}_k = -\frac{\hat{D}_{k,eff}}{Y_k}\nabla Y_k - \frac{D_k^T}{\rho Y_k}\frac{\nabla T}{T} \tag{103}$$

<u>Note</u>: The subscript "*eff*" on diffusional coefficients denotes that both laminar (i.e. molecular) and turbulent contributions to these coefficients are possible. When turbulent flow is modeled, the turbulent contributions to these coefficients are calculated based on the turbulence model chosen (as described in Section 3.8). When laminar flow is modeled, these coefficients are simply equal to their molecular values.

4. Boundary Conditions

4.1 Specification of Boundary Conditions

In general, the second-order transport-reaction equations in MPSalsa need either their dependent variables or their normal derivatives specified at all domain boundaries in order to define a well-posed problem. EXODUS II defines the concept of node sets and side sets on which these boundary conditions are applied. A node set is an arbitrary group of nodes in the domain. A side set is an arbitrary group of element sides in the domain. Only side sets establish the concept of a surface.

Dirichlet boundary conditions specify the value of dependent variables. The usual conservation equation for the dependent variable identified with an element node, where a Dirichlet boundary condition is specified, is discarded and replaced with another equation for that variable. The new equation may be a function of the other independent or dependent variables in the problem. Dirichlet conditions that don't need the concept of a surface may be applied on node sets as well as side sets. MPSalsa also allows for Dirichlet conditions to be applied as surface integrals of functions weighted by the elemental basis functions, *i.e.* Galerkin's method. These surface integral Dirichlet conditions may be applied only on side sets. For example, the concept of a surface is needed to define normal and tangential vectors for normal and tangential velocity boundary conditions.

Neumann and Robin (or mixed) boundary conditions impose conditions on the normal derivative of the dependent variable. This term is specified by replacing the normal derivative in the surface integral that arises from the integration by parts during the Galerkin finite element formulation with the boundary condition. Surface integral conditions may be applied only on side sets and are generally defined as being satisfied in a "weak sense". In other words, they are satisfied only in the limit of no discretization error.

The following is a discussion of the types of boundary conditions permissible in MPSalsa for each of the conservation equations.

4.2 Momentum Equations

For the fluid dynamical part of the problem, either the velocity components or the normal component of the total stress tensor must be specified on the boundary of the domain for each component of the vector momentum equation. On both side and node sets, Dirichlet boundary conditions of the form

$$u_m = f_{u_m}(\mathbf{x}, \mathbf{u}, P, T, \mathbf{Y}, t), \; m = 1, 2, 3 \qquad (104)$$

may be applied to the velocity in the x-, y- and z-directions, respectively. In Eqn. 104, f_{u_m} is a user-specified function of the dependent and independent variables. For these boundary conditions, the corresponding momentum equations are replaced by Eqn. 104 at all nodes of the designated node or side sets.

Surface integrals involving the components of the surface traction vector, ,
on a surface with normal, $\mathbf{n}$, arise naturally in the Galerkin form of the momentum equations and are added to the volumetric contributions of the Jacobian and residuals of all nodes on the surface. The components of the normal stress may be replaced in the surface integrals by user-specified functions f_τ of the dependent and independent variables, as shown in Eqn. 105, where Φ_i is the elemental shape function for node i on the surface.

$$\int_\Gamma \tau_m \Phi_i d\Gamma = \int_\Gamma f_{\tau,m} \Phi_i d\Gamma \qquad , m = 1, 2, 3 \tag{105}$$

Boundary conditions may also be applied to the normal and tangential components of the velocity and normal stress. Each region of the boundary is associated with a unit normal to the boundary, $\mathbf{n}$, and two orthonormal tangential components to the boundary, $\mathbf{t_1}$ and $\mathbf{t_2}$. Specification of the boundary condition for the momentum equations then involves specification of the velocity component or normal tensor component in each of the directions $\mathbf{n}$, $\mathbf{t_1}$, and $\mathbf{t_2}$; that is, the user must specify either $\mathbf{n} \bullet \mathbf{u}$ or $\tau \bullet \mathbf{n}$, and either $\mathbf{t_1} \bullet \mathbf{u}$ and $\mathbf{t_2} \bullet \mathbf{u}$, or $\tau \bullet \mathbf{t_1}$ and $\tau \bullet \mathbf{t_2}$.

Normal and tangential Dirichlet boundary conditions on velocity are enforced using surface integrals along sides of elements. The surface integral form of a Dirichlet boundary condition on the normal velocity is given by Eqn. 106. For each elemental node on a surface, i, the boundary condition is multiplied by the elemental shape function Φ_i. The integral over the surface of the resulting expression is the residual contribution for the corresponding component of the momentum equation for node i. Similar expressions enforce tangential velocity boundary conditions.

$$\int_\Gamma (\mathbf{n} \bullet \mathbf{u} - f_{u_n}(\mathbf{x}, \mathbf{u}, P, T, \mathbf{Y}, t))\Phi_i d\Gamma = 0 \tag{106}$$

Conditions on the normal stress in the normal and tangential directions are enforced by replacing $\tau \bullet \mathbf{n}$, $\tau \bullet \mathbf{t_1}$, and $\tau \bullet \mathbf{t_2}$ in the surface integrals with user-supplied functions, which are then rotated to derive expressions for $\tau \bullet \mathbf{i}$, $\tau \bullet \mathbf{j}$, and $\tau \bullet \mathbf{k}$, which are needed in the surface integral terms in the *x*, *y*, and *z* momentum equations, respectively.

For example, Eqn. 107 specifies traction boundary conditions in a 2-D geometry with $\mathbf{n} = \mathbf{i}$ and $\mathbf{t_1} = \mathbf{j}$. In this examples, $\mathbf{f}_\tau$ is the user-supplied function specifying the traction vector.

$$\begin{aligned} \tau \bullet \mathbf{n} = \tau \bullet \mathbf{i} &= -P - \frac{2}{3}\mu\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + 2\mu\frac{\partial u}{\partial x} = \mathbf{n} \bullet \mathbf{f}_\tau(\mathbf{x}, \mathbf{u}, P, T, \mathbf{Y}, t) \\ \tau \bullet \mathbf{t_1} = \tau \bullet \mathbf{j} &= \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) = \mathbf{t_1} \bullet \mathbf{f}_\tau(\mathbf{x}, \mathbf{u}, P, T, \mathbf{Y}, t) \end{aligned} \tag{107}$$

In Eqn. 107, $\mathbf{f}_\tau$ is shown as a function of all of the independent and dependent variables in the problem $(\mathbf{x}, \mathbf{u}, P, T, \mathbf{Y}, t)$. A common outflow boundary condition is setting the normal stress, $\tau \bullet \mathbf{n}$, to zero. This is the so called natural B.C. on the momentum equation.

For the particular case of a reacting, impermeable wall, the Dirichlet boundary conditions in Eqn. 108 are applied using Eqn. 106.

$$\mathbf{n} \bullet \rho \mathbf{u} = -\sum_{k=1}^{N_g} \dot{s}_k W_k \tag{108}$$

$$\mathbf{t}_1 \bullet \mathbf{u} = \mathbf{t}_2 \bullet \mathbf{u} = 0$$

4.3 Total Continuity Equation

The incompressible Navier-Stokes equations are unchanged when the hydrodynamic pressure is changed by a constant. They are affected only by gradients in the hydrodynamic pressure and MPSalsa's discrete equation set shares this property. Therefore, the pressure scale must be set either implicitly or explicitly somewhere in the domain. This is achieved by specifying $\boldsymbol{\tau} \bullet \mathbf{n}$ somewhere on the boundary since P appears in this expression (see Eqn. 107), or by setting a Dirichlet condition for P on one node in the domain.

4.4 Internal Energy Continuity Equation

Either the temperature or the normal heat flux must be specified on all boundaries of the domain. That is, either Dirichlet boundary conditions in the form of a user-supplied function or value must be specified for the temperature, or surface integral boundary conditions involving the heat conduction must be used. The expression in the surface integral resulting from the Galerkin integration by parts is the normal component of the heat flux vector, $\mathbf{n} \bullet \mathbf{q}_c$, where $\mathbf{q}_c = -\lambda \nabla T$. The user supplies a function that is substituted for $\mathbf{n} \bullet \mathbf{q}_c$ in the surface integral.

Inflow boundary conditions for the energy equation are usually specified by a Dirichlet condition on the temperature. For cases where the energy balance at a surface must be calculated, Eqn. 109 is a useful starting point in the derivation of energy boundary conditions based on heat balances.

$$\mathrm{FLUX}^{-} + \mathrm{PRODUCTION}_{\Gamma} = \mathrm{FLUX}^{+} \tag{109}$$

The heat flux to the boundary from within the solution domain is defined as FLUX^{-}. This, plus the energy stored at the interface, $\mathrm{PRODUCTION}_{\Gamma}$, should be equated to the heat flux exiting the domain, FLUX^{+}.

For the convection of enthalpy inlet boundary condition, $\mathrm{PRODUCTION}_{\Gamma}$ is zero but the flux terms are defined by Eqn. 110. An extra convective heat transfer term, $h_c(T - T_o)$, is added to the inflow heat flux, on the outer side of the domain. In MPSalsa, the user supplies a function returning the value of $\mathbf{n} \bullet \mathbf{q}_c$ as determined by Eqn. 109 and 110.

$$\text{FLUX}^{-} = -\mathbf{n}\bullet\left(\sum_{k=1}^{N_g}\rho Y_k\hat{h}_k\mathbf{u}+\lambda\nabla T+\sum_{k=1}^{N_g}h_k\mathbf{j}_k+\mathbf{q}_r\right)\Bigg|_{-}$$
$$\text{FLUX}^{+} = \left(\sum_{k=1}^{N_g}\rho_o Y_{k,o}\hat{h}_k(T_o)u_o+\mathbf{q}_r\right)\Bigg|_{+}+h_c(T-T_o) \tag{110}$$

For boundary conditions corresponding to outflow areas, where neither the energy flux nor the temperature is known before hand, the specification of a zero normal temperature derivative is used (natural b.c.).

$$\mathbf{n}\bullet\nabla T = 0 \tag{111}$$

For boundary conditions corresponding to solid walls where reactions may be occurring, Eqn. 109 may be used to obtain a heat balance. FLUX^{-} is given by Eqn. 110 and $\text{PRODUCTION}_{\Gamma}$ is nonzero due to the growing or etching film at the interface.

$$\text{PRODUCTION}_{\Gamma} = \sum_{k=K_s^f}^{K_s^l}\dot{s}_kW_kh_k+\sum_{k=K_b^f}^{K_b^l}\dot{s}_kW_kh_k+\dot{Q}_{\Gamma} \tag{112}$$

$\text{PRODUCTION}_{\Gamma}$ includes terms due to the storage of energy due to surface and bulk-phase species and $\dot{Q}_{\Gamma}$ is the heat input to the boundary from external sources (e.g., resistive heating). Typically, FLUX^{+} is specified by a heat transfer coefficient combined with radiative heat input from a black body at a known temperature. However, its exact specification is left undefined at this point. The enthalpy terms in FLUX^{-} and $\text{PRODUCTION}_{\Gamma}$ may be combined with reacting wall boundary conditions on the species conservation equations (Eqn. 37) to yield Eqn. 113.

$$-\mathbf{n}\bullet(\lambda\nabla T+\mathbf{q}_r)\big|_{-}-\sum_{k=1}^{K_b^l}\dot{s}_kW_kh_k = \dot{Q}_{\Gamma}+\text{FLUX}^{+} \tag{113}$$

The sum in Eqn. 113 is over all species defined in the problem: gas, surface, and bulk. For phase change-type reactions, the second term in Eqn. 113 can be identified with the latent heat of the phase change. Radiation contributions, $\mathbf{q}_r$, appear naturally in surface integral expressions for the heat flux. Currently, an MP gray body radiation treatment is under development and will be presented at a later time.

4.5 Gas-Phase Species Continuity Equations

Several types of boundary conditions may be specified on Y_k, $k = 1, \ldots, N_g$. Theoretically, either the value of Y_k or its normal derivative must be specified on a boundary. However, for low pressure systems where diffusive transport dominates, Dirichlet conditions on the species equations are discouraged as a means of specifying the flow rate of species k into the system. The actual flux of species k into the domain, which consists of both convective and diffusive contributions, will be quite different than the intended flux into the domain. Therefore, flux-based conditions should be used on all boundaries of the domain for these systems.

For boundary conditions corresponding to inflow areas where the flow rates of the gas-phase species are known, the flux of species k is specified by what is known as Danckwerts' boundary condition:

$$\mathbf{n} \bullet (\rho Y_k \mathbf{u} + \mathbf{j}_k) = \rho_o u_o Y_{k,o} \qquad (k = 1, \ldots, N_g) \,, \tag{114}$$

where ρ_o, u_o and $Y_{k,o}$ are user-specified values.

For boundary conditions corresponding to solid walls where reactions may be occurring, the flux of species k to the wall should be equated with the negative of the net production rate of species k at the wall.

$$\mathbf{n} \bullet (\rho Y_k \mathbf{u} + \mathbf{j}_k) = -\dot{s}_k W_k \qquad (k = 1, \ldots, N_g) \tag{115}$$

For boundary conditions corresponding to solid walls, where no reactions are occurring, the net flux of species k should be set to zero.

$$\mathbf{n} \bullet (\rho Y_k \mathbf{u} + \mathbf{j}_k) = 0 \qquad (k = 1, \ldots, N_g) \tag{116}$$

For boundary conditions corresponding to outflow areas, where neither the flux nor the concentration of species k is known, the specification of a zero normal diffusion velocity may be employed.

$$\mathbf{n} \bullet \mathbf{j}_k = 0 \qquad (k = 1, \ldots, N_g) \tag{117}$$

The boundary conditions in Eqn. 115-117 are incorporated into the finite element equations representing the continuity equation for species k via the boundary integral involving $(\mathbf{n} \bullet \mathbf{j}_k)$ that appears from the integration by parts of the diffusive flux term. Specifically, $(\mathbf{n} \bullet \mathbf{j}_k)$ is replaced with the appropriate terms from Eqn. 115-117 expressed via a user-supplied function f_k^Y as in Eqn.

$$\mathbf{n} \bullet \mathbf{j}_k = f_k^Y \qquad (k = 1, \ldots, N_g) \tag{118}$$

As with any Neumann or Robin boundary conditions in the finite element method, these boundary conditions are satisfied only in the limit that the discretization error goes to zero. Also, if a determination of the flux of species k is required at a reacting solid wall where Eqn. 115 is used, the flux should be evaluated using the right hand side of Eqn. 115 instead of the left hand side. The accuracy in Y_k is one order of the mesh discretization size greater than the accuracy in the derivatives of Y_k.

For non-CHEMKIN material types, Dirichlet boundary conditions of the form $Y_k = f_k^Y$, $k = 1, \ldots, N_g$, and flux boundary conditions of the form in Eqn. are supported.

5. Finite Element Approximation of the Transport Equations

The governing transport Eqns. 95-103 are approximated by a Petrov-Galerkin finite element method (PGFEM). The summary presented here is intended to provide a sufficiently detailed discussion of the FE development and a practical formulation background to discuss the numerical algorithms that are used to solve the resulting linear algebra problems.

The finite element procedure begins by dividing the physical domain of interest, Ω, into N_e simply shaped regions Ω_e called finite elements. Within each of these elements, the dependent variables $(u_1, u_2, u_3, P, T, Y_k)$, $k = 1, \ldots, N_g$, are interpolated by continuous functions of compatible order, in terms of values to be determined at a set of global node points. To develop the FE equations for these nodal unknowns, we present the finite element expansion in terms of global interpolation functions. This development differs from an elemental basis approach only in the interpretation of the summation scope and the resulting domain of integration of the inner product. Using this approach simplifies the resulting discussion of the node-based matrix-fill algorithms in the parallel implementation of the code.

5.1 The Residuals of the Transport Equations

The residual of the governing transport PDEs are given below in Table 5-1. These residual definitions are used in the subsequent discussion of the Galerkin Least Squares (GLS) formulation.

Momentum	$\mathbf{R}_\mathrm{m} = \rho\frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \bullet \nabla\mathbf{u}) - \nabla \bullet \mathbf{T} - \rho\mathbf{g}$
Total Mass	$\mathrm{R}_\mathrm{P} = \frac{\partial \rho}{\partial t} + \nabla \bullet (\rho\mathbf{u})$
Thermal Energy	$\mathrm{R}_T = \rho\hat{C}_p\left[\frac{\partial T}{\partial t} + \mathbf{u} \bullet \nabla T\right] + \nabla \bullet \mathbf{q}_c - \phi - \dot{Q} + \sum_{k=1}^{N_g} \mathbf{j}_k \bullet \hat{C}_{p,k}\nabla T - \sum_{k=1}^{N_g} h_k W_k \dot{\omega}_k + \nabla \bullet \mathbf{q}_\mathrm{r}$
Species Mass Fraction for Species k	$\mathrm{R}_{Y_k} = \rho\left[\frac{\partial Y_k}{\partial t} + \mathbf{u} \bullet \nabla Y_k\right] + \nabla \bullet \mathbf{j}_k - W_k\dot{\omega}_k \qquad , k = 1, 2, \ldots, N_s - 1$

Table 5-1 Governing Transport PDEs

The continuous problem, defined by the transport equations, is approximated by a Galerkin Least Squares formulation [42,43,44]. This formulation allows for equal order interpolation of pressure and velocity (without spurious pressure solutions), for stabilization of highly convected flows and for a discontinuity capturing operator that smooths oscillation in the vicinity of large gradients. The resulting GLS equations are shown in Table 5-2.

Momentum	$\mathcal{F}_{m,i} = \int_{\Omega} \Phi R_{m,i} d\Omega + \int_{\Omega_e} \rho\tau_m (\mathbf{u}\bullet\nabla\Phi) R_{m,i} d\Omega + \int_{\Omega_e} \rho\tau_c [\nabla\Phi]_i R_P d\Omega + \int_{\Omega_e} \nu_{m,i} \nabla\Phi \bullet \mathbf{G}^c \nabla u_i d\Omega$
Total Mass	$\mathcal{F}_P = \int_{\Omega} \Phi R_P d\Omega + \int_{\Omega_e} (\rho\tau_m \nabla\Phi \bullet \mathbf{R}_m) d\Omega$
Thermal Energy	$\mathcal{F}_T = \int_{\Omega} \Phi R_T d\Omega + \int_{\Omega_e} \rho\hat{C}_p \tau_T (\mathbf{u}\bullet\nabla\Phi) R_T d\Omega + \int_{\Omega_e} \nu_T \nabla\Phi \bullet \mathbf{G}^c \nabla T d\Omega$
Species Mass Fraction for Species k	$\mathcal{F}_{Y_k} = \int_{\Omega} \Phi R_{Y_k} d\Omega + \int_{\Omega_e} \rho\tau_{Y_k} (\mathbf{u}\bullet\nabla\Phi) R_{Y_k} d\Omega + \int_{\Omega_e} \nu_{Y_k} \nabla\Phi \bullet \mathbf{G}^c \nabla Y_k d\Omega \ , k = 1, 2, \ldots, N_s - 1$

Table 5-2 GLS Formulation of Transport PDEs

The stabilization parameters (the τ's and the ν's) are given in [43,44]. For clarity in the following discussion, the Newtonian stress tensor, $\mathbf{T}$, is expanded to include the pressure, P, and the viscous stress tensor term, Υ (see Eqn. 6). The resulting GLS total mass residual equation in expanded form is given in Eqn. 119.

$$\mathcal{F}_P = \int_{\Omega} \Phi\left(\frac{\partial\rho}{\partial t} + \nabla\bullet(\rho\mathbf{u})\right) d\Omega + \int_{\Omega_e} \rho\tau_m \nabla\Phi \bullet \left[\rho\frac{\partial\mathbf{u}}{\partial t} + \rho\mathbf{u}\bullet\nabla\mathbf{u} + \nabla P - \nabla\bullet\Upsilon - \rho\mathbf{g}\right] d\Omega \qquad (119)$$

This expansion exhibits the "Laplacian type" of operator acting on pressure

$$\int_{\Omega_e} \rho\tau_m \nabla\Phi \bullet \nabla P d\Omega \qquad (120)$$

produced by the GLS formulation of the total mass conservation equation. This term plays an important role in the development of effective Krylov based solvers with various preconditioners. Finite element (FE) discretization of the GLS equations gives rise to a system of coupled, nonlinear, nonsymmetric algebraic equations, the numerical solution of which can be very challenging. These equations are linearized using an inexact form of Newton's methods which is discussed in the next section. A block matrix representation of these discrete linearized equations is given in Eqn. 121. In this representation the vector, $\mathbf{v}'$, contains the Newton updates to all the nodal solution variables with the exception of the nodal pressures, P'. The block matrix, A, corresponds to the combined discrete convection, diffusion and reaction operators for all the unknowns; the matrix, $\mathcal{B}$, corresponds to the discrete divergence operator with its transpose the gradient operator; the diagonal matrix, R, results from the group FE expansion of the density and velocity in Eqn. 7; and the matrix, K, corresponds to the discrete "pressure Laplacian" operator discussed above. The matrix, , contains terms associated with the convection and diffusion terms in the momentum residual generated by the GLS formulation of the total mass equation. The vectors $\mathcal{F}_V$, and $\mathcal{F}_P$ contain the right hand side residuals for Newton's method.

The existence of the well behaved nonzero matrix, K, in the FE discretization of the GLS equations allows the solution of the linear systems with a number of algebraic and domain decomposition type preconditioners. This is in contrast to other formulations, such as Galerkin methods using mixed interpolation, that produce a zero block on the total mass continuity diagonal. The difficulty of producing robust and efficient preconditioners for the Galerkin formulation has motivated the use of many different types of solution methodologies. A number of these use two level iteration schemes, penalty methods, pseudo-compressibility techniques or decoupled/segregated solvers [45,46,47,48,49]. A detailed presentation of the characteristics of current solution methods is far beyond the scope of the current manuscript. The intent of fully-coupling the transport PDEs in the nonlinear solver is to preserve the inherently strong coupling of the physics with the goal to produce a more robust solution methodology in the process.

$$\begin{bmatrix} A & -\mathcal{B}^T \\ (\mathcal{B}R + \mathcal{L}) & K \end{bmatrix} \begin{bmatrix} V' \\ P' \end{bmatrix} = \begin{bmatrix} -\mathcal{F}_V \\ -\mathcal{F}_P \end{bmatrix} \tag{121}$$

5.2 Discrete Equations: Interpolation Functions and Quadrature Rules

Within each element the mixture velocity, temperature, species mass fractions, and hydrodynamic pressure are approximated by the expansions in Eqn. 122.

$$\begin{aligned}
u_l(\mathbf{x}, t) &= \sum_{J=1}^{N} (u_l)_J(t)\Phi_J(\mathbf{x}) && l = 1, 2, 3 \\
P(\mathbf{x}, t) &= \sum_{J=1}^{N} P_J(t)\Phi_J(\mathbf{x}) \\
T(\mathbf{x}, t) &= \sum_{J=1}^{N} T_J(t)\Phi_J(\mathbf{x}) \\
Y_k(\mathbf{x}, t) &= \sum_{J=1}^{N} (Y_k)_J(t)\Phi_J(\mathbf{x}) && k = 1, \ldots, N_g
\end{aligned} \tag{122}$$

Here, $\Phi_J(\mathbf{x})$ is the standard polynomial finite element basis function associated with the J^{th} global node, N is the total number of global nodes in the domain, and N_g is the number of gas-phase species. The u_1, u_2, and u_3 components of velocity correspond to velocity in the x-, y-, and z-

directions, respectively. Equal order interpolation of all variables is used. In these and the following expansions, global interpolation functions are denoted with uppercase indices as in the expansions above. The only exception to this convention is the use of a lower case index k to denote the species number.

Thermodynamic and transport properties, as well as volumetric source terms, are interpolated from their nodal values using the finite element shape functions. For example, Eqn. 123 represents the computation of density at a point $\mathbf{x}$. The density is not evaluated from the equation of state with values of the dependent variables at $\mathbf{x}$ but instead is computed at global nodes $J = 1, \ldots, N$ with values of the dependent variables at the global nodes, and the elemental shape functions are used to interpolate the density at $\mathbf{x}$.

$$\rho(\mathbf{x}, t) = \sum_{J=1}^{N} \rho_J(t)\Phi_J(\mathbf{x}) \tag{123}$$

Evaluation of volumetric integrals is performed by standard Gaussian quadrature. For quadrilateral and hexahedral elements, two-point quadrature (in each dimension) is used with linear basis functions, while three-point quadrature is used for quadratic interpolated elements. For example, for tri-linear hexahedral elements, eight Gaussian quadrature points within an element are used to evaluate its volumetric integrals.

5.3 Evaluation of Surface Integrals

Evaluation of surface integrals is performed by standard Gaussian quadrature on the side of the element. As with the volumetric integrals, two-point quadrature (in each direction) is used with linear shape functions, while three-point quadrature is used with quadratic shape functions. For example, for a three-dimensional problem with linear shape functions, four Gaussian quadrature points located on the side of an element are used to evaluate its surface integrals.

6. Solution Procedures

In this section, we present the general procedures used in MPSalsa for the steady state and the time dependent solution of equations that describe the discrete problem. The choice of numerical methods in MPSalsa has been made from the standpoint of robustness, efficiency of implementation on parallel architectures, and the ease of including new solution kernels. The major solution kernels used in MPSalsa are the first- and second-order implicit time integration routines, an inexact Newton procedure and the linear system solvers of the Aztec [12] parallel Krylov solver library, developed in conjunction with MPSalsa. Below we summarize the properties of the discrete matrix problem and consider the details of the major solution kernels in MPSalsa. First, we give a brief overview of the implementation of the unstructured finite element method on multiple processors, since this aspect underlies much of the discussion and implementation of the solution algorithms for the linear system.

6.1 Implementation on Multiple Processors

MPSalsa is designed to solve problems on massively parallel (MP) multiple instruction multiple data (MIMD) computers with distributed memory. For this reason the basic parallelization of the finite element problem is accomplished by a domain partitioning approach. The initial task on an MP computer is to partition the domain among the available processors, where each processor is assigned a subdomain of the original domain. It communicates with its neighboring processors along the boundaries of each subdomain. There are two fundamental ways to partition the FE domain among processors: either element or node assignment. Each method has its own advantages and fundamentally affects the solution strategies and interprocessor communications. Dividing the mesh according to elements quite naturally can lead to an element-by-element (EBE) solution scheme, whereas dividing the mesh according to nodes leads most naturally to a fully-summed distributed matrix solutions. In the EBE case, each element's matrix is stored separately and is not summed with its contributions from neighboring elements. All matrix-vector operations are performed with these dense elemental block matrices and the vector result is obtained only after summing over all elements. This scheme substantially increases the matrix storage requirements and the amount of computation needed relative to fully-summed distributed matrix solution strategies. For example, for 3-D linear hexahedral elements, this method requires approximately 60% more storage and greater than three times as many floating point computations are required for the EBE approach. Although the larger block sizes associated with the EBE approach may yield an increase in the number of operations performed per second, this improved performance is unlikely to compensate for the increased operation count. Because of this, nodal decomposition was chosen in MPSalsa to allow the implementation of computationally efficient, minimum flop algorithms for the matrix-vector multiply kernel. Also, storing the fully summed equations allows the use of robust general preconditioners, such as domain decomposition incomplete factorizations and direct sparse subdomain solvers.

The parallel solution of a particular FE problem proceeds as follows. At the start of the problem, each processor is "assigned" a set of finite element nodes that it "owns." A processor is responsible for forming the residual and the corresponding row in the fully summed distributed matrix for the unknowns at each of its assigned FE nodes. To calculate the residual for unknowns at each assigned node, the processor must perform element integrations over all elements for which it owns at least one element node. To do this the processor requires 1) the local geometry of the element and 2) the value of all unknowns at each of the FE nodes in each element for which it owns at least one node. The required elemental geometry is made available to the processor through the initial partitioning and database distribution part of the algorithm. Here, a broadcast of all information in a serial EXODUS data base to all processors is used in MPSalsa. Then, each processor extracts its geometry information form the FE database. In addition to the broadcast algorithm, MPSalsa has the capability to use a parallel FE database [11] for geometry input as well as all parallel I/O. The unstructured interprocessor communication of FE unknowns is handled by an Aztec routine that exchanges the necessary interprocessor information [12].

Figure 6-1, which depicts a partitioning scheme of an unstructured mesh, graphically represents the above concepts. An unstructured mesh is divided into four regions by assigning ownership of the nodes. Nodes in each processor are classified as "border" and "internal" nodes, at which border and internal unknowns, respectively, are defined. Border unknowns are those unknowns whose values must be communicated to neighboring processors so they may complete their element integrations; the remaining "owned" unknowns on a processor are designated as internal unknowns. Those unknowns required for a processor's element integrations but assigned to a neighboring processor are stored in the local solution vector and designated as "external" unknowns. Interprocessor communication occurs when an owning processor communicates the values of its border unknowns to a neighboring processor to update the value of the neighboring processor's corresponding external unknowns. Figure 6-1 demonstrates how Processor 0 would classify the nodes in the internal, border, and external categories. Processor 0 has three neighboring processors. During the interprocessor communication phase, it sends each neighboring processor a message containing the values of each border unknown that the neighboring processor needs. The value of each border unknown may be needed by more than one processor, as it may appear in the external node lists of more than one of the neighboring processors. Processor 0 also receives a message from each of its surrounding processors containing the values of its external unknowns. Processor 0 doesn't have to know about unknowns defined at elemental nodes which don't have the △, ○, or ◇ symbols attached to them.

On each processor, a solution vector is stored which corresponds to the internal, border, and external unknowns defined on that processor. The solution vector is reordered locally so that local internal unknowns appear first, border unknowns appear second, and external unknowns, grouped by the owning neighboring processor, appear last. A local-to-global mapping vector is maintained, so that the global solution vector may be regenerated using "fan-in" operations. This local reordering scheme minimizes the gather/scatter operations involved in the interprocessor communication step. Only a gather operation at the originating processor to gather all of the border unknowns needed by a single neighboring processor into a contiguous space in memory is required. This message can then be directly sent to the contiguous space in the destination processor's solution vector corresponding to the external unknowns owned by the originating processor. No scatter operations are needed on the destination processor. Moreover, the communications stencil required for this operation may be calculated once and used over and over again for a static mesh discretization. The communications stencil refers to the content of the message that each processor needs to send to each of its neighboring processors and the length of the return message containing the external unknown values from each neighboring processor.

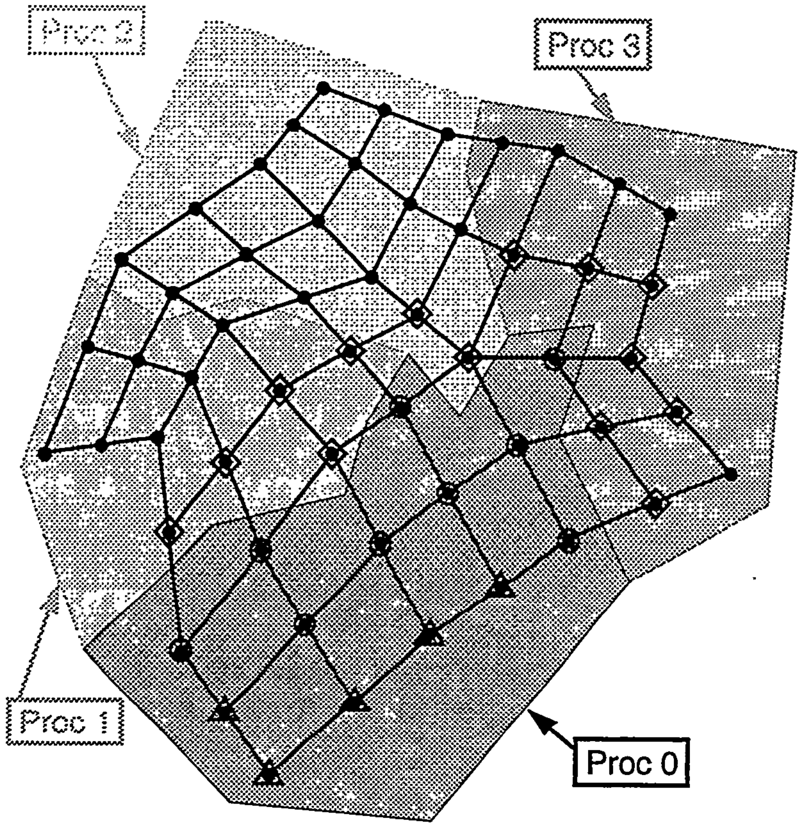

Figure 6-1: Division of the nodes of an element amongst the processors, and the further differentiation of the nodes into interior (△), border (○), and external (◇) categories on Processor 0.

MPSalsa stores the Jacobian matrix in a distributed version of the Variable Block Row (VBR) sparse matrix format [13]. Each processor is responsible for storing rows of the Jacobian corresponding to its unknowns. Once a specific partition and assignment of the unknowns to internal, border, and external sets has been defined and the local solution vector has been reordered a distributed VBR sparse matrix is constructed. Each row of the Jacobian may include column entries corresponding to internal, border, and external unknowns defined on that processor. During the matrix-vector multiply kernel of the Krylov subspace iterative methods, each processor is responsible for carrying this out for its rows. This necessitates an interprocessor communication step wherein all external entries in the vector are updated with values from the neighboring unknowns, before the start of the operation. Calculation of matrix-vector products on rows corresponding to the internal unknowns requires no external node values and can therefore proceed simultaneously with the communication step.

Much of MPSalsa's parallel implementation is designed with the goal of maximizing the speed of this matrix-vector multiplication, which essentially requires minimizing the time needed to perform the communications. This subsection has described several strategies employed by MPSalsa to achieve rapid interprocessor communications: reordering of the solution vector to minimize work involved with the communications step, the pre-setup of the communications stencil, and the ability to do calculations during the communications step. The other basic algorithmic aspect of highly efficient unstructured communication is the partitioning of the FE mesh in a way that reduces the total communication volume and message start-ups while achieving load balance over all of the processors. To do this, MPSalsa currently uses a static partitioning generated by **Chaco** [56], a general graph partitioning code that was developed in conjunction with MPSalsa. **Chaco** supports a variety of new and established graph partitioning heuristics, such as spectral techniques, geometric methods, multilevel algorithms and the Kernighan-Lin method. All of these approaches may be applied in bisection, quadrisection, or octasection mode to recursively partition general graphs for mapping onto hypercube and mesh architectures of arbitrary size. Using these techniques, a problem mapping with low communications volume, good load balancing, minimum message start-ups and small amounts of congestion can be generated.

6.2 Numerical Properties

The system of transport-reaction equations, Eqn.'s 95-103, is a system of nonlinear non-self-adjoint PDEs. The final matrix problem is obtained by applying the Petrov-Galerkin approximation to these equations and then doing a Newton-Kantoravich linearization. These discrete equations form a nonsymmetric system of stiff Differential Algebraic Equations (DAEs). The nonsymmetric global matrix operator is a result of the convection operators in the transport part of the equations, and the stiffness in the equations is the result of the disparate time scales for the fast chemical kinetics terms and the relatively slow transport processes of diffusion and convection.

The stiffness and the strongly coupled nature of the reaction operators, combined with the elliptic behavior of the pressure for incompressible flows, lead to a natural choice of fully implicit time integration techniques to provide stable time integration. The nonsymmetric character of these equations requires the use of nonsymmetric iterative methods.

6.3 Transient Solution Algorithms

The transient time integration methods used in MPSalsa follow closely the development of Gartling [22] in the NACHOS II code and the work of Gresho [50]. When appropriate, we have used the discussion from [22] with the author's permission.

Two types of implicit predictor/corrector integrators are used in MPSalsa: Forward/Backward Euler and Adams-Bashforth/Trapezoidal Rule. As discussed above, implicit solution methods are preferred for transport-reaction equations. Explicit methods suffer from a number of difficulties, including a) the strong elliptic nature of pressure in incompressible flows, b) severe time step limitations needed to maintain stability, c) fully integrated and consistent mass matrices require the inversion - defeating the efficiency of the explicit method, d) the reduction of accuracy due to diagonalizing $\mathcal{M}(\rho)$ to avoid (c). Effective explicit time integration demands 1-pt-quadrature and the associated stable lumped mass matrix. Though computationally expensive, implicit methods are desirable because of their stability and ability to integrate efficiently to steady state solutions for problems where the diffusion operator is important. The implicit time integrators in MPSalsa are based on predictor/corrector methods to improve their accuracy and efficiency. Both integrators may be used with either a constant or dynamic time step selection algorithm. A solution of the resulting nonlinear, algebraic system for each time plane is obtained by the inexact Newton method described in Section 6.4.

6.3.1 Forward /Backward Euler Integration

The first-order integration method in MPSalsa employs a forward Euler scheme as a predictor, with the backward Euler method as a corrector. The scheme uses the forward Euler predictor,

$$V^{p}{}_{n+1} = V_n + \Delta t_n \dot{V}_n,\ \Delta t_n = t_{n+1} - t_n. \tag{124}$$

The implicit backward Euler corrector uses the following approximation for the time derivative of the solution vector

$$\dot{V}_{n+1} = \frac{1}{\Delta t_n}(V_{n+1} - V_n), \tag{125}$$

to solve the residual equations at t_{n+1}.

$$\boldsymbol{R}(\dot{V}_{n+1}, V_{n+1}) = \boldsymbol{0} \tag{126}$$

In Eqn. 124 and 125, the subscript indicates the time plane index, the superscript p denotes the predicted value at time t_{n+1}. The solution of the implicit corrector, Eqn. 126, at t_{n+1} is obtained by the inexact Newton scheme outlined in the Section 6.4. The rate of convergence of Newton's method is greatly increased if the initial solution estimate is "close" to the true solution. The solution predicted from Eqn. 124 provides this initial guess for the inexact Newton scheme. Appendix C provides the details of developing the discrete Newton equations for the governing transport-reaction equations.

6.3.2 Adams-Bashforth/Trapezoidal Rule Integration

An explicit integration method that is the second-order analogue to the forward Euler method is the variable step, Adams-Bashforth predictor given by

$$V^{p}{}_{n+1} = V_n + \frac{\Delta t_n}{2}\left[\left(2 + \frac{\Delta t_n}{\Delta t_{n-1}}\right)\dot{V}_n - \frac{\Delta t_n}{\Delta t_{n-1}}\dot{V}_{n-1}\right]. \tag{127}$$

This formula can be used to predict the solution vector, given the time derivatives at the previous two time steps, $\dot{V}_n$ and $\dot{V}_{n-1}$. A compatible corrector equation is available in the form of the trapezoidal rule. This corrector uses an approximation to the time derivative as

$$\dot{V}_{n+1} = \frac{2}{\Delta t_n}(V_{n+1} - V_n) - \dot{V}_n. \tag{128}$$

Eqn. 128 is then used in Eqn. 126 to find the solution at t_{n+1}. Eqn. 128 is also used to calculate the time derivative at t_{n+1} for later use in the predictor equation, Eqn. 127, in later time steps.

6.3.3 Time Integration Procedures

The integration formulas above form the basis for the solution of time-dependent problems in MPSalsa. The similarity of the first- and second-order methods makes it possible to include both procedures in a single algorithm. The major steps in the time integration procedure are outlined here.

At the beginning of each time step, it is assumed that all of the required solution and time derivative vectors are known and the time increment for the next step has been selected. To advance the solution from time t_n to time t_{n+1} requires the following steps:

1) A tentative solution vector, V^{p}_{n+1}, is computed using the predictor equation (either Eqn. 124 or Eqn. 127).

2) The implicit corrector equation, Eqn. 126, using Eqn. 125 or Eqn. 128 for the time derivative, is solved for the actual solution, V_{n+1}. This involves the iterative solution of the linear matrix equation arising from Newton's method. The predicted values V^{p}_{n+1} are used to initialize the FE residuals and the Jacobian matrix for the Newton iterations.

3) The time derivative vectors are updated using the new solution V_{n+1} and Eqn. 125 or Eqn. 128. These equations can be conveniently described by the following relationship for the time derivative,

$$\dot{V}_{n+1} = CJ(V_{n+1} - V_n) - (\alpha - 1)\dot{V}_n \,, \tag{129}$$

where

$$CJ = \begin{cases} \dfrac{1}{\Delta t_n} \\ \dfrac{2}{\Delta t_n} \end{cases} \quad and \quad \alpha = \begin{cases} 1 & order = 1 \\ \\ 2 & order = 2 \end{cases} . \tag{130}$$

The relation, $d\dot{V}_{n+1}/dV_{n+1} = CJ$, can be used in the formulation of the Jacobian.

4) A new integration time step is computed. The time step selection process is based on the analysis of the time truncation errors in the predictor and corrector formulas as described in the Section 6.3.4. If a constant time step is being used, this step is omitted.

6.3.4 Time Step Control

The time integration procedures above can be used with either a user-defined constant time step or a dynamically controlled time step that is initialized with the user-defined time step size. In general, the *a priori* selection of a time step size can be a very difficult task, especially for stiff reacting flow equations with complex fluid flows. One of the benefits of using the predictor/corrector algorithms is that they provide a rational basis for dynamically selecting the time step size.

The details of time step control algorithm can be found in Gresho et al. [50]. The general formulation of the time step selection process comes from well-established procedures for solving ordinary differential equations. By comparing the time truncation errors for two time integration methods of comparable order, a formula can be developed for predicting the next time step, based on a user-specified error tolerance. In the present case, the time truncation errors for the explicit predictor and the implicit corrector steps are analyzed and provide the required formulas.

The time step estimation formula is given by [50] as

$$\Delta t_{n+1} = \Delta t_n (b r_\varepsilon)^m , \tag{131}$$

where $m = 1/2$, $b = 2$ for the first-order method and $m = 1/3$, $b = 3(1 + \Delta t_{n-1}/\Delta t_n)$ for the second-order scheme. Also, r_ε is a ratio of the desired time integration error to an estimate of the time integration error. Clearly, when r_ε is large, a larger time step can be taken and when r_ε is small, a shorter time step must be used. In practice, we have selected a measure of the time integration error that works well for the combined fluid flow and reaction kinetics problem. In MPSalsa, this ratio is computed as

$$r_\varepsilon = \frac{1}{N_{unk}} \sum_{i=1}^{N_{unk}} \frac{\hat{\varepsilon}_i}{(V_i^p - V_i)} , \tag{132}$$

where the subscript i refers to the component of the solution vector, N_{unk} is the total number of unknowns and $\hat{\varepsilon}_i$ is the desired integration accuracy for this component. For the fluid velocity unknowns, $\hat{\varepsilon}_i = \varepsilon_r \|\mathbf{u}\|_\infty$, where ε_r is the relative accuracy desired; for temperature, $\hat{\varepsilon}_i = \varepsilon_r \|\mathbf{T}\|_\infty$. These measures enforce a minimum relative accuracy of time integration for the computed value locally, compared with a measure of the maximum value of the variable in the domain and are very similar to the values used in NACHOS II [22]. The hydrodynamic pressure, P, does not influence the step size control norm since there is no time derivative of the pressure in the governing transport equations. However, the determination of convergence at each time step does involve the pressure unknown. For the mass fraction unknowns, MPSalsa requires that the local time truncation error be small relative to the magnitude of the local variable and to an absolute measure of accuracy since even trace amounts of a specific chemical species can produce significant changes in the kinetics. To accomplish this, MPSalsa uses $\hat{\varepsilon}_i = \varepsilon_r |Y_{k,i}| + \varepsilon_a$, where ε_a is the desired absolute accuracy.

6.4 Inexact Newton Method with Backtracking

In this section, we briefly discuss an implementation of Newton's method that uses approximate iterative solution techniques to solve the sequence of linear problems produce by the Newton linearization scheme. The particular implementation we use follows the work of Eisenstat and Walker [51,52,53]. This method differs from standard Newton implementations as follows. First the inexact Newton scheme uses iterative solution techniques rather than direct matrix inversion methods. Second, at each stage of the Newton iteration, the algorithm selects an appropriate level of convergence required for the iterative linear solver. This strategy is used to increase robustness of the nonlinear algorithm and to ensure that the linear equations are not over-solved at early stages of the Newton iteration when the Jacobian matrix is not very accurate. Third, this algorithm requires that at each step of the Newton iteration, the nonlinear residual must decrease. If this condition is not satisfied, a backtracking algorithm decreases the Newton step size and re-evaluates the residual at this new proposed solution. The backtracking algorithm is called recursively until the residual reduction criteria is satisfied and a new approximate solution is obtained.

6.4.1 Nonlinear Convergence Criteria

Two separate convergence requirements are enforced for the Newton scheme. The first requires that the ratio of the norm of the current nonlinear residual to the norm of the initial residual be reduced by a preset factor (default: 10^{-2}). The second criterion requires that the Newton correction for any variable be suitably "small" compared to the magnitude of the variable. This criterion is very similar to the ratio used to dynamically control the time step size and is standard in general purpose ODE packages such as LSODE [58]. This convergence criterion is given by

$$\frac{1}{N_{unk}} \sum_{i=1}^{N_{unk}} \frac{|\Delta V_i|}{\varepsilon_r |V_i| + \varepsilon_a} < 1. \tag{133}$$

This criterion requires the ratio of the Newton correction $|\Delta V_i|$ be small relative to the variable $|V_i|$ with constant ε_r, and to be small in absolute terms compared to ε_a. This assures that all variables, even variables with small magnitude (e.g., trace species), are considered in determining when to halt the Newton iteration.

6.5 Linear System Solvers

The linear systems generated by the Newton iteration are iteratively solved using preconditioned Krylov methods. The methods are among the fastest and most robust iterative methods currently available. Our implementation of MPSalsa uses a parallel preconditioned Krylov solver library called Aztec[12]. The Aztec library provides an efficient and well-defined interface to a number of advanced parallel iterative solution methods. These include the well-known conjugate gradient (CG) method for symmetric positive definite systems and a number of closely related algorithms for the solution of nonsymmetric systems (e.g. generalized minimum residual method (GMRES) and transpose free quasi-minimum residual method (TFQMR)) as well as various algebraic and domain decomposition preconditioners.

For robust and efficient solution procedures, MPSalsa and Aztec use a sparse block storage scheme called the variable block row (VBR) format [13]. Storing the matrix in a sparse format allows very efficient iterative computational kernels to be used [50, 54] and allows for the use of robust general preconditioning methods. These robust schemes are critical to the solution of the strongly-coupled physics solved in MPSalsa. In the VBR format, the nonlinear dense coupling of the Jacobian at each FE node is stored intact as a small dense block. Details of the Aztec solver library can be obtained from [12].

References

1 Shadid, J. N., Moffat, H. K., Hutchinson, S. A., Hennigan, G. L., Devine, K. D., Salinger, A. G., "MPSalsa: A Finite Element Computer Program for Reacting Flow Problems. Part 2: User's Guide", *Sandia National Laboratory Report,* SAND96–2331, Albuquerque, NM (1996).

2 Spalart, P.R., Allmaras, S.R., "A One-Equation Turbulence Model for Aerodynamic Flows," *La Recherche Aerospatiale,* **1**, 5-21 (1994).

3 Kee, R. J., Miller, J. A., "A Structured Approach to the Computational Modeling of Chemical Kinetics and Molecular Transport in Flowing Systems," *Sandia National Laboratories Report,* SAND86–8841, Albuquerque, NM (1986).

4 Kee, R. J., Rupley, F. M., and Miller, J. A., "Chemkin-II: A Fortran Chemical Kinetics Package for the Analysis of Gas-Phase Chemical Kinetics," *Sandia National Laboratories Report,* SAND89–8009, Albuquerque, NM (1989).

5 Coltrin, M. E., Kee, R. J., Rupley, F. M., "SURFACE CHEMKIN (v. 4.0) A Fortran Package for Analyzing Heterogeneous Chemical Kinetics at a Solid-Surface —Gas-Phase Interface," *Sandia National Laboratories Report,* SAND90–80033, Albuquerque, NM (1990).

6 Kerstein, A. R., "Linear-Eddy Modeling of Turbulent Transport. Part 6. Microstructure of Diffusive Scaler Mixing Fields," *J. Fluid Mech.*, **231**, 361-394 (1991).

7 T. D. Blacker et. al., "CUBIT Mesh Generation Environment Volume 1: Users Manual", Technical Report, Sandia National Laboratories, SAND94-1100, May 1994

8 Schoof, L. A., Yarberry, V. R., "EXODUS II: A Finite Element Data Model," *Sandia National Laboratories Technical Report,* SAND92–2137, Albuquerque, NM (1994).

9 Rew, R., Davis, G., Emmerson, S., "NetCDF User's Guide: An Interface for Data Access, Version 2.3,", UCAR, April, 1993.

10 Shadid, J. N., Hutchinson, S. A., Moffat, H. K., Hennigan, G. L., Hendrickson, B. A., Leland, R. W., "A 65+ Gflop/s unstructured finite element simulation of chemically reacting flows on the Intel Paragon", *Proceedings of Supercomputing '94,* Washington, DC, pp. 673–679, Nov. 14–18, 1994.

11 Hennigan, G. L., Shadid, J. N., "A Parallel Extension of the EXODUS Unstructured FEM Format," *Sandia National Laboratories Technical Report,* SAND95–XXXX, Albuquerque, NM (to be published).

12 Hutchinson, S. A., Shadid, J. N., Tuminaro, R. S., “Aztec User’s Guide: Version 1.1,” *Sandia National Laboratories Technical Report,* SAND95–1559, Albuquerque, NM. (1995).

13 S. Carney, M. Heroux and G. Li, “A proposal for a sparse BLAS toolkit”, SPARKER Working Note #2, Cray Research, Inc., Eagen, MN, (1993)

14 Coltrin, M. E., Kee, R. J., Evans, G. H., Meeks, E., Rupley, F. M., Grcar, J. F., “SPIN: A Fortran Program for Modeling One-Dimensional Rotating-Disk/Stagnation-Flow Chemical Vapor Deposition Reactors,” *Sandia National Laboratories Report,* SAND91–8003, Albuquerque, NM (1991).

15 Lutz, A. E., Kee, R. J., Miller, J. A., “SENKIN: A Fortran Program for Predicting Homogeneous Gas Phase Chemical Kinetics with Sensitivity Analysis,” *Sandia National Laboratories Technical Report,* SAND87–8248, Albuquerque, NM (1987).

16 Ethier, C.R. and Steinman, D.A., “Exact Fully 3D Navier-Stokes Solutions for Benchmarking,” *Intl. Jrnl. Num. Meth. Fluids,* **19**, 369-375 (1994).

17 Bird, R. B., Stewart, W. E., and Lightfoot, E. N., *Transport Phenomena,* John Wiley (1960).

18 Kuo, K. K., *Principles of Combustion,* John Wiley, NY (1986).

19 Hirschfelder, J. O., Curtis, C. F., Bird, R. B., *Molecular Theory of Gases and Liquids*, John Wiley and Sons, Inc., New York (1954).

20 Chapman, S., Cowling, T. G., *The Mathematical Theory of Non-Uniform Gases,* 3rd Ed., Cambridge University Press, Cambridge (1970).

21 Dixon-Lewis, G., “Flame Structure and Flame Reaction Kinetics II. Transport Phenomena in Multicomponent Systems,” *Proc. Roy. Soc. A.*, **307**, 111–135 (1968).

22 Gartling, D. K., “NACHOS II: A Finite Element Computer Program for Incompressible Flow Problems. Part 1 – Theoretical Background,” *Sandia National Laboratories Report,* SAND86–1816, Albuquerque, NM (1986).

23 Paolucci, S., 1982, “On the Filtering of Sound from the Navier-Stokes Equations,” *Sandia National Laboratories Technical Report,* SAND82–8257, Albuquerque, NM (1982).

24 Martinez, M. J., “Analysis of Anelastic Flow and It’s Numerical Treatment via Finite Elements, *Sandia National Laboratories Technical Report,”* SAND84–0320, Albuquerque, NM (1994).

25 Burns, S. P., “SYRINX Users Manual,” *Sandia National Laboratories Report,* SAND99–XXXX, Albuquerque, NM (1999)

26 Kee, R. J., Dixon-Lewis, G., Warnatz, J., Coltrin, M. E., Miller, J. A., "A Fortran Computer Code Package for the Evaluation of Gas-Phase Multicomponent Transport Properties," *Sandia National Laboratories Report,* SAND86–8246, Albuquerque, NM (1986).

27 Williams, F. A., *Combustion Theory*, Benjamin/Cummings, Menlo Park, CA (1985).

28 Coffee, T. P., Heimerl, J. M., "Transport Algorithms for Premixed Laminar Steady-State Flames," *Combustion and Flame,* **43,** 273–289 (1981).

29 Heimerl, J. M., Coffee, T. P., "Transport Algorithms for Methane Flames," *Combustion Science and Technology*, **34,** 31–43 (1983).

30 Oran, E. S., Boris, J. P., *Numerical Simulation of Reactive Flow,* Elsevier, New York (1987).

31 Mitchell, R. E., Kee, R. J., "A General-Purpose Computer Code for Predicting Chemical Kinetic Behavior Behind Incident and Reflected Shocks," *Sandia National Laboratories Report,* SAND82–8205, Albuquerque, NM (1982).

32 Kee, R. J., Grcar, J. F., Smooke, M. D., Miller, J. A., "A Fortran Program for Modeling Steady Laminar One-Dimensional Premixed Flames," *Sandia National Laboratories Report,* SAND85–8240, Albuquerque, NM (1985).

33 Kee, R. J., Rupley, F. M., Miller, J. A., "The CHEMKIN Thermodynamics Data Base," *Sandia National Laboratories Report,* SAND87–8215, Albuquerque, NM (1987).

34 Gordon, S., McBride, B. J., "Computer Program for Calculation of Complex Chemical Equilibrium Compositions, Rocket Performance, Incident and Reflected Shocks and Chapman-Jouguet Detonations," *NASA Technical Report,* NASA SP–273 (1971).

35 Moffat, H. K., Glarborg, P., Kee, R.J., Grcar, J. F., Miller, J. A., "SURFACE PSR: A Fortran Program for Modeling Well-Stirred Reactors for Gas and Surface Reactions," *Sandia National Laboratories Report,* SAND91–8001, Albuquerque, NM (1991).

36 Patel, V. C., Rodi, W., and Scheuerer, G., "A Review and Evaluation of Turbulence Models for Near Wall and Low Reynolds Number Flows," *AIAA Journal*, **23,** 1308 (1985).

37 Launder, B. E., and Sharma, B. I., "Application of the Energy-Dissipation Model of Turbulence to the Calculation of Flow Near a Spinning Disk," *Letters in Heat and Mass Transfer*, **1,** 131-138 (1974).

38 Davidson, L.,"Second-order Corrections of the k-ε Model to Account for Non-isotropic Effects Due to Buoyancy," *International Journal for Heat and Mass Transfer*, **133,** No. 12, 2599-2608 (1990).

39 Smagorinsky, J., "General Circulation Experiments with the Primitive Equations," *Monthly Weather Rev.*, **3,** 99-164 (1963).

40 Schumann, U., "Subgrid Scale Model for Finite Difference Simulations of Turbulent Flows in Plane Channels and Annuli," *J. Comp. Physics*, **18**, 376-404 (1975).

41 Yoshizawa, A., "Bridging between eddy-viscosity-type and second-order models using a two-scale DIA," presented at the *9th Symposium on Turbulent Shear Flows*, Kyoto, Japan, Aug. 16-18 (1993).

42 Hughes, J. R., Franca, L. P., Balestra, M., "A New Finite Element Formulation for Computational Fluid Dynamics: V. Circumventing the Babuska-Brezzi Condition: A Stable Petrov-Galerkin Formulation of the Stokes Problem Accommodating Equal-order Interpolations", *Comp. Meth. App. Mech. and Eng.*, **59**, 85–99 (1986).

43 Tezduyar, T. E., "Stabilized Finite Element Formulations for Incompressible Flow Computations", *Advances in App. Mech.*, **28**, 1–44, (1992).

44 Shakib, F. "Finite element analysis of the compressible Euler and Navier-Stokes equations," Ph. D. Thesis, Stanford University, (1988).

45 Elman, H.C. and Golub, G.H., "Inexact and Preconditioned Uzawa Algorithms for Saddle Point Problems", Technical Report, Numerical Analysis Project #NA-93-02, June (1993).

46 Haroutunian, V. Engelman. M.S. and Hasbani, I., "Segregated Finite Element Algorithms for the Numerical Solution of Large-Scale Incompressible Flow Problems", *Int. J. Numer. Meth. Fluids*, **17**, 323-348, (1993).

47 Strigbreger, J., Baruzzi, G., and Habashi, W., "Some Special Purpose Preconditioners for Conjugate Gradient-like Methods Applied to CFD", *Int. J. Numer. Meth. Fluids*, **16**, 581-596, (1993).

48 Vincent, C. and Boyer, R., "A Preconditioned Conjugate Gradient Uzawa-type Method for the Solution of the Stokes Problem by Mixed Q1-P0 stabilized Finite Elements", *Int. J. Numer. Meth. Fluids*, **14**, 289-298, (1992).

49 Wille, S.O. "A Preconditioned Alternating Inner-Outer Iterative Solution Method for the Mixed Finite Element Formulation of the Navier-Stokes Equations", *Int. J. Numer. Meth. Fluids*, **18**, 1135-1151, (1994).

50 Gresho, P. M., Lee, R. L., Sani, R. L., "On the time dependent solution of the incompressible Navier-Stokes equations in two and three dimensions," *Recent Advances in Numerical Methods in Fluids*, Vol. 1, Pineridge Press, Swansea, U. K., 27–81 (1980).

51 Eisenstat, S. C. and Walker, H. F. "Choosing the forcing terms in an inexact Newton method." *SIAM J. Sci. Computing*, **17**, 16 (1996).

52 Eisenstat, S. C. and Walker, H. F., Globally convergent inexact Newton methods." *SIAM J. Optimization*, 4:393-422, 1994

53 Walker, H. F., "Summary of Activities and Results at Sandia National Laboratories, June - September, 1995.

54 Schunk, P. R., Shadid, J. N., "Iterative solvers in implicit finite element codes," *Sandia National Laboratories Technical Report,* SAND92–1158, Albuquerque, NM (1992).

55 Lee, L, Gresho, P., Chan, S., Sani, R., and Cullen, M., "Conservation Laws for Primative Variable Formulations for the Incompressible Flow Equations using the Galerkin Finite Element Method," *Lawrence Livermore Laboratory Report,* preprint number UCRL-82868, March, 1981.

56 Hendrickson, B. and Leland, R., "The Chaco User's Guide, Version 2.0," *Sandia National Laboratories Report,* SAND94-2692, Albuquerque, NM (1995).

57 Boyd, J. P., "Chebyshev and Fourier Spectral Methods," Lecture notes in Engineering, Springer-Verlag, Berlin, Heidelberg, 1989

58 A. C. Hindmarsh, "LSODE and LSODEI: Two new Initial Value Ordinary Differential Equation Solvers", *ACM Signum Newsletter*, **15**, No. 4, pp 10-11, 1980

Nomenclature

a_k^b	Activity of the k^{th} bulk-phase species.
A	Surface area.
A_i	Pre-exponential factor in computation of rate constant for reaction i.
$c_k(n)$	Concentration of the k^{th} surface species in the n^{th} surface phase.
$C_b(n)$	Average molar concentration of the n^{th} bulk phase (mol cm^{-3}).
$\hat{C}_P$	Specific heat of the mixture at constant pressure.
$\hat{C}_{p,k}$	Specific heat at constant pressure for species k.
$\mathbf{d}_k$	Diffusional driving force for species k.
D_{kj}	Multicomponent diffusion coefficient.
$\mathcal{D}_{kj}$	Binary diffusion coefficient between species k and j.
D_k^T	Mixture thermal diffusion coefficient for species k.
D_{km}	Mixture-averaged diffusion coefficient.
$\hat{D}_k$	Effective Fickian diffusion coefficient for use in the Jacobian (cm^2 s^{-1}).
E_i	Activation energy for reaction i.
$f_{\mathbf{n} \bullet \mathbf{u}}$	Dirichlet boundary condition on the normal component of the velocity.
$f_{\mathbf{t1} \bullet \mathbf{u}}$	Dirichlet boundary condition on one of the tangential components of the velocity, in the direction **t1**.
$\mathbf{f}_\tau$	Vector value of the surface integral boundary condition applied on the normal component of the stress tensor.
f_k^Y	Value of the surface integral boundary condition for the normal component of the diffusion flux of the k^{th} gas-phase species.
$\mathbf{g}$	External force of gravity.
$\mathcal{G}(n)$	Molar growth rate per unit of surface area for bulk phase n (mol cm^{-2} s^{-1}).
$\Delta H^o_{f,j}(T_o)$	Heat of formation of the j^{th} species at the reference temperature T_o.
h	Mixture enthalpy per unit mass.
h_e^*	Effective element length of element Ω_e.
$\hat{h}_k$	Specific enthalpy of species k (per unit mass).
$\mathbf{i}$	Vector $[1, 0, 0]^T$.
$\mathbf{I}$	Identity matrix or second order tensor.

$\mathbf{j}$	Vector $[0, 1, 0]^T$.
$\mathbf{j}_k$	Diffusive flux for species k (gm cm^{-2} s^{-1}).
$\mathbf{k}$	Vector $[0, 0, 1]^T$.
k_i^f	Forward rate constant for the i^{th} reaction.
k_i^r	Reverse rate constant for the i^{th} reaction.
K_i^c	Equilibrium constant in concentration units for reaction i.
$K_b^f(n)$	First bulk species in the n^{th} bulk phase.
$K_b^l(n)$	Last bulk species in the n^{th} bulk phase.
$K_s^f(n)$	First surface species in the n^{th} surface phase.
$K_s^l(n)$	Last surface species in the n^{th} surface phase.
L_n	Film thickness for the n^{th} bulk phase.
N	Number of global nodes.
N_d	Number of dimensions in the problem.
N_e	Number of elements in Ω.
N_g	Number of gas-phase chemical species; also the number of gas-phase species equations.
N_R	Number of elementary reversible or irreversible reactions.
N_{phase}^{bulk}	Number of bulk phases.
N_{phase}^{surf}	Number of surface phases.
N_{unk}	Total number of solution unknowns.
P	Hydrodynamic pressure.
P_o	Thermodynamic pressure.
$\mathbf{q}$	Total heat flux vector.
$\mathbf{q}_c$	Heat conduction vector.
$\mathbf{q}_r$	Radiative heat flux vector.
q_i	Rate-of-progress variable for the i^{th} gas-phase reaction (mol cm^{-3} s^{-1}).
R	Universal gas constant.
Re^*_e	Modified element Reynolds number for element Ω_e.
$\dot{s}_k$	Surface production rate of gas- or surface-phase species k due to surface reactions (mol cm^{-2} s^{-1}).

$\mathbf{T}$	Shear stress tensor.
τ	Surface traction vector.
T	Temperature (Kelvin).
t	Time.
$\dot{Q}$	Volumetric source term for the energy equation.
$\mathbf{u}$	Mass averaged velocity (cm s^{-1}).
$\mathbf{V}_k$	Diffusion velocity of species k.
W_k	Molecular weight of species k.
$\overline{W}$	Mean molecular weight of mixture.
$\mathbf{x}$	A point in space; $\mathbf{x} = (x, y)$ in 2-D; $\mathbf{x} = (x, y, z)$ in 3-D.
X_k	Mole fraction of species k.
$X_k^b(n)$	Bulk mole fraction for bulk species k in the n^{th} bulk phase.
$[X_k]$	Concentration of species k (moles cm^{-3}).
Y_k	Mass fraction of species k.
$Z_k(n)$	Surface site fraction for surface species k for the n^{th} surface phase.

GREEK

β	Parameter in residual of continuity equation.
$\bar{\beta}$	Coefficient of volumetric expansion.
β_i	Temperature exponent in computation of rate constants for reaction i.
ε	Specific internal energy of the mixture (erg gm^{-1}).
ε_a	User-specified absolute accuracy.
ε_r	User-specified relative accuracy.
Γ	Boundary of computational domain Ω.
Γ_n	Surface site density for surface phase n.
χ_k	Chemical symbol for the k^{th} species.
Υ	Shear stress tensor.
ρ	Mixture density (gm cm^{-3}).
μ	Mixture dynamic viscosity.
λ	Mixture thermal conductivity.
σ_k	Number of surface sites covered by the k^{th} species.
ν_{ki}	$\nu''_{ki} + \nu'_{ki}$.
ν'_{ki}	Stoichiometric coefficient of the k^{th} species for the forward direction of the i^{th} gas-

	phase reaction.
ν''_{ki}	Stoichiometric coefficient of the k^{th} species for the reverse direction of the i^{th} gas-phase reaction.
ϕ	Viscous dissipation term in energy equation.
Φ_J	Global finite element basis function at node J.
$\dot{\omega}_i$	Volumetric molar rate of production of species i (mol cm^{-3} s^{-1}).
Ω	Computational domain.

Distribution

INTERNAL DISTRIBUTION:

1	MS 0321	William Camp, 9200
1	MS 0601	Harry K. Moffat, 1126
1	MS 1111	Sudip Dosanjh, 9221
1	MS 1111	Scott Hutchinson, 9221
20	MS 1111	John N. Shadid, 9221
1	MS 1111	Andrew G. Salinger, 9221
1	MS 1111	Gary L. Hennigan, 9221
5	MS 1111	Rodney C. Schmidt, 9221
1	MS 1111	Thomas M. Smith, 9221
1	MS 1110	Ray S. Tuminaro, 9222
1	MS 1109	Karen Devine, 9224
1	MS 0827	Mario Martinez, 9111
1	MS 0827	Poly Hopkins, 9111
1	MS 0835	Mark A. Christon, 9111

1	MS 9018	Central Technical Files, 8940-2
2	MS 0899	Technical Library, 4916
1	MS 0619	Review & Approval Desk, 15102 For DOE/OSTI