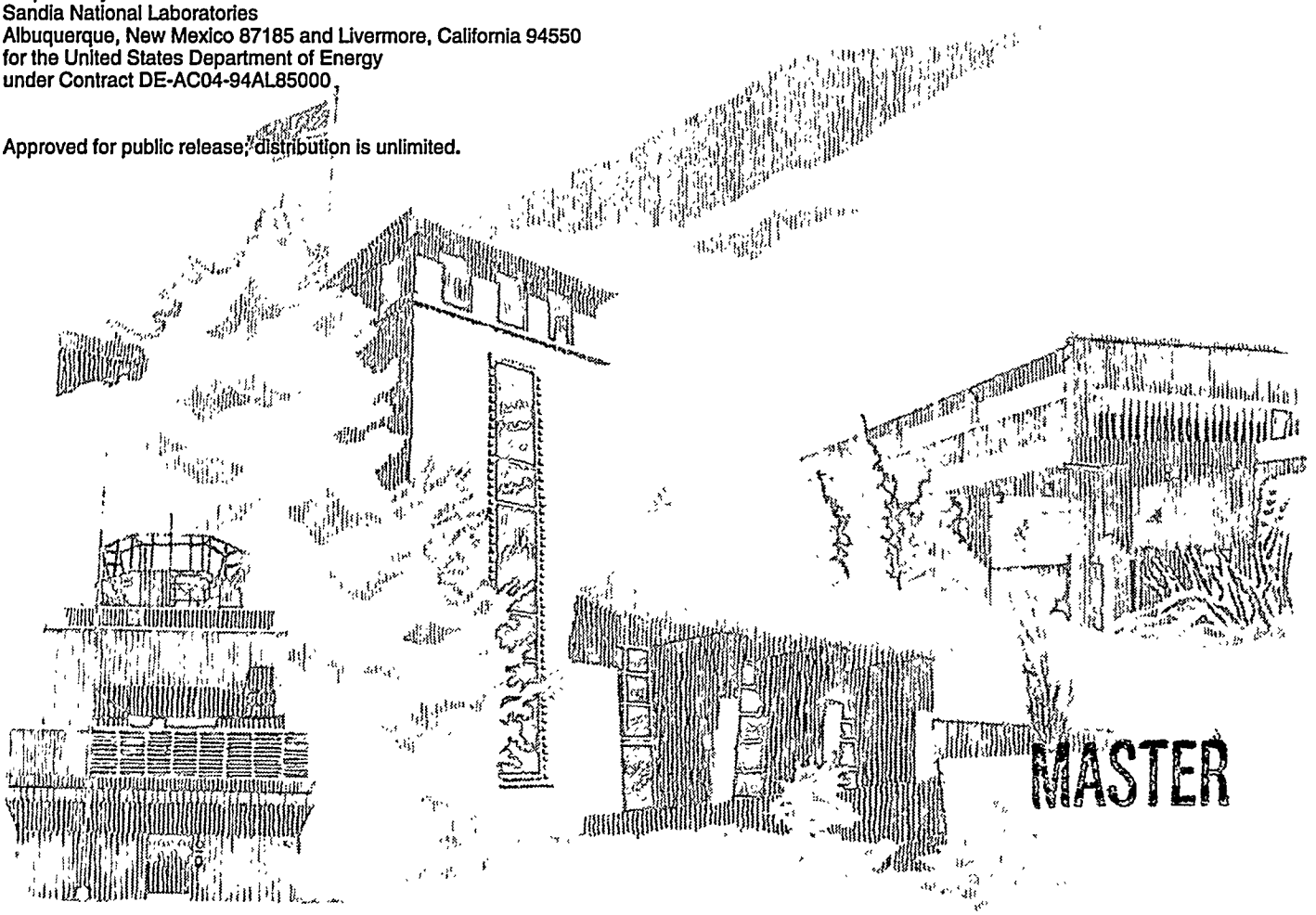
SANDIA REPORT

SAND95–2937 • UC–905
Unlimited Release
Printed January 1996

GOMA – A Full-Newton Finite Element Program for Free and Moving Boundary Problems with Coupled Fluid/Solid Momentum, Energy, Mass, and Chemical Species Transport: User's Guide

P. R. Schunk, P. A. Sackinger, R. R. Rao, K. S. Chen, R. A. Cairncross

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-94AL85000



SF2900Q(8-81)

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A07
Microfiche copy: A01

SAND95-2937
Unlimited Release
Printed January 1996

Distribution
Category UC-905

GOMA - A Full-Newton Finite Element Program for Free and Moving Boundary Problems with Coupled Fluid/Solid Momentum, Energy, Mass, and Chemical Species Transport: User's Guide

P. R. Schunk, P. A. Sackinger, R. R. Rao, K. S. Chen, R. A. Cairncross

Manufacturing and Environmental Fluid Dynamics Department
Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

GOMA is a two- and three-dimensional finite element program which excels in analyses of manufacturing processes, particularly those involving free or moving interfaces. Specifically, the full-Newton-coupled heat, mass, momentum, and pseudo-solid mesh motion algorithm makes GOMA ideally suited for simulating processes in which the bulk fluid transport is closely coupled to the interfacial physics. Examples include, but are not limited to, coating and polymer processing flows, soldering, crystal growth, and solid-network or solution film drying. The code is based on the premise that any boundary can be (1) moving or free, with an apriori unknown position dictated by the distinguishing physics, (2) fixed, according to a global analytical representation, or (3) moving in time and space under user-prescribed kinematics. The goal is to enable the user to predict boundary position or motion simultaneously with the physics of the problem being analyzed and to pursue geometrical design studies and fluid-structure interaction problems.

The moving mesh algorithm treats the entire domain as a computational Lagrangian solid that deforms subject to the physical principles which dictate boundary position. As an added benefit, the same Lagrangian solid mechanics can be exploited to solve multi-field problems for which the solid motion and stresses interact with other transport phenomena, either within the same material phase (e.g. shrinking coating) or in neighboring material phases (e.g. flexible blade coating). Thus, analyses of many fluid-structure interaction problems and deformable porous media problems are accessible.

This document serves as a user's guide and reference for GOMA and provides a brief overview of GOMA's capabilities, theoretical background, and classes of problems for which it is targeted.

Acknowledgment

Development of GOMA was funded in part by the Engineering Science Research Foundation, Laboratory Directed Research and Development, and the Basic Energy Science Program of the DOE.

GOMA - User's Guide

Contents

Figures

Tables

Nomenclature

Cp	heat capacity
C_i	concentration of species i
$\hat{c}$	inertial coefficient
$\boldsymbol{D}$	strain-rate tensor
D	binary diffusion coefficient
d	defined distance between current position and dynamic contact line
$\underline{d}$	displacement vector
E	Young's modulus
$\boldsymbol{e}_\alpha$	unit base vector
G	shear modulus
$\boldsymbol{g}$	momentum source term vector
H	volumetric energy source
$2H$	mean curvature
h	heat transfer or mass transfer coefficient
$\boldsymbol{I}$	identity tensor
$\underset{\sim}{i}, \underset{\sim}{j}, \underset{\sim}{k}$	unit vectors representing right-hand orthogonal basis
$\boldsymbol{J}_i$	mass flux vector, species i
K	bulk modulus
k	permeability
$\underset{\sim}{n}$	normal vector to surface
$\underset{\sim}{n}_w$	normal vector to solid wall surface
$\boldsymbol{q}$	heat flux vector
R_i	component i volumetric source
$\boldsymbol{R}_i$	i-th component of the Galerkin weighted residual vector
T	temperature
$\boldsymbol{T}$	fluid stress tensor
$\boldsymbol{T}_s$	solid stress tensor
$\underset{\sim}{t}$	tangential vector to surface
$\boldsymbol{v}$	velocity vector
$\boldsymbol{v}_m$	mesh velocity vector
$\dot{\underset{\sim}{x}}$	mesh velocity
x, y, z	coordinates for Cartesian coordinate system

y_i	volume fraction of species i
α	scaling for position-dependent slip
II_D	second invariant of the strain-rate tensor
λ	Lame coefficient
μ	Lame coefficient in solid mechanics or viscosity in fluid mechanics
υ	Poisson's ratio
ρ	density
σ	electrical conductivity or surface tension
φ_i	basis (shape) function associated with node i
θ	contact angle
β	slip coefficient
ϕ	porosity or electrical potential

1 Introduction

"GOMA", which means *rubber, gum,* or *elastic* in Spanish, is a two- or three-dimensional finite element program currently being advanced and specialized for the analysis of manufacturing flows and related processes that involve one or more transport fields, i.e., any combination of heat, mass, momentum (solid and fluid) and species transport fields. Specifically, the processes for which GOMA is suited are those which contain free or moving boundaries between dissimilar materials or phases. Whether it is an interface or point whose position or motion is *a priori* unknown governed by boundary physics, or a boundary whose position or motion is prescribed by the user according to specified geometry or kinematics, the multiphysics approach on which GOMA is based allows for rapid convergence to the solution. Unique features which make this possible include: (1) a Lagrangian-Eulerian solid mechanics module for mesh motion, (2) energy and chemical species undergoing convection, diffusion and reaction, and fluid momentum transport modules that are fully and mutually coupled, particularly with the mesh motion module through an analytical Jacobian matrix, (3) a full-Newton based solution algorithm which exploits that Jacobian matrix, and (4) a structure which allows for different physical descriptions of different materials in the same problem, i.e., conjugate problems. The scope of potentially accessible problems is defined by the interaction and close coupling of the individual field equation sets, as shown in Figure 1. Moreover, the analytical Jacobian matrix which provides that coupling facilitates a range of computer-aided nonlinear analyses such as parametric sensitivity (stability), design, and optimization as it provides the building blocks (through chain-rule differentiation) for evaluating sensitivities of process variables to processing conditions.

GOMA originated from an early version of SALSA (Shadid and Moffat 1995), a finite element program designed to simulate chemically reacting flows in massively-parallel computing environments, and was originally extended and adapted to free and moving boundary problems in fluid mechanics, heat transfer, and mass transfer. However, by virtue of a mesh motion algorithm which fills the computational domain with an artificial Lagrangian solid, many multiphysics problems which include the deformation of real solid materials in combination with other transport phenomena are now accessible with GOMA. The detailed algorithm and underlying physical principles together with several advanced examples from capillary hydrodynamics, melting and solidification, and polymer processing may be found elsewhere (Sackinger et al. 1995, Cairncross et al. 1995, Chen et al. 1995). The purpose of this report is to provide a practical introduction and reference to GOMA: to introduce the user to the range of options available in GOMA; to show how easily the code may be adapted to investigate novel situations; and to provide several simple illustrative examples as a tutorial and as a demonstration of the overall utility of the program.

Chapter 2 of this manual provides some background on the physics and numerical methods employed in GOMA. Ancillary issues addressed in Chapter 2 include code portability between computing platforms, necessary software libraries, and other miscellaneous subjects which pertain to obtaining a license to GOMA. Chapter 3 discusses briefly the structure of the programs as well as

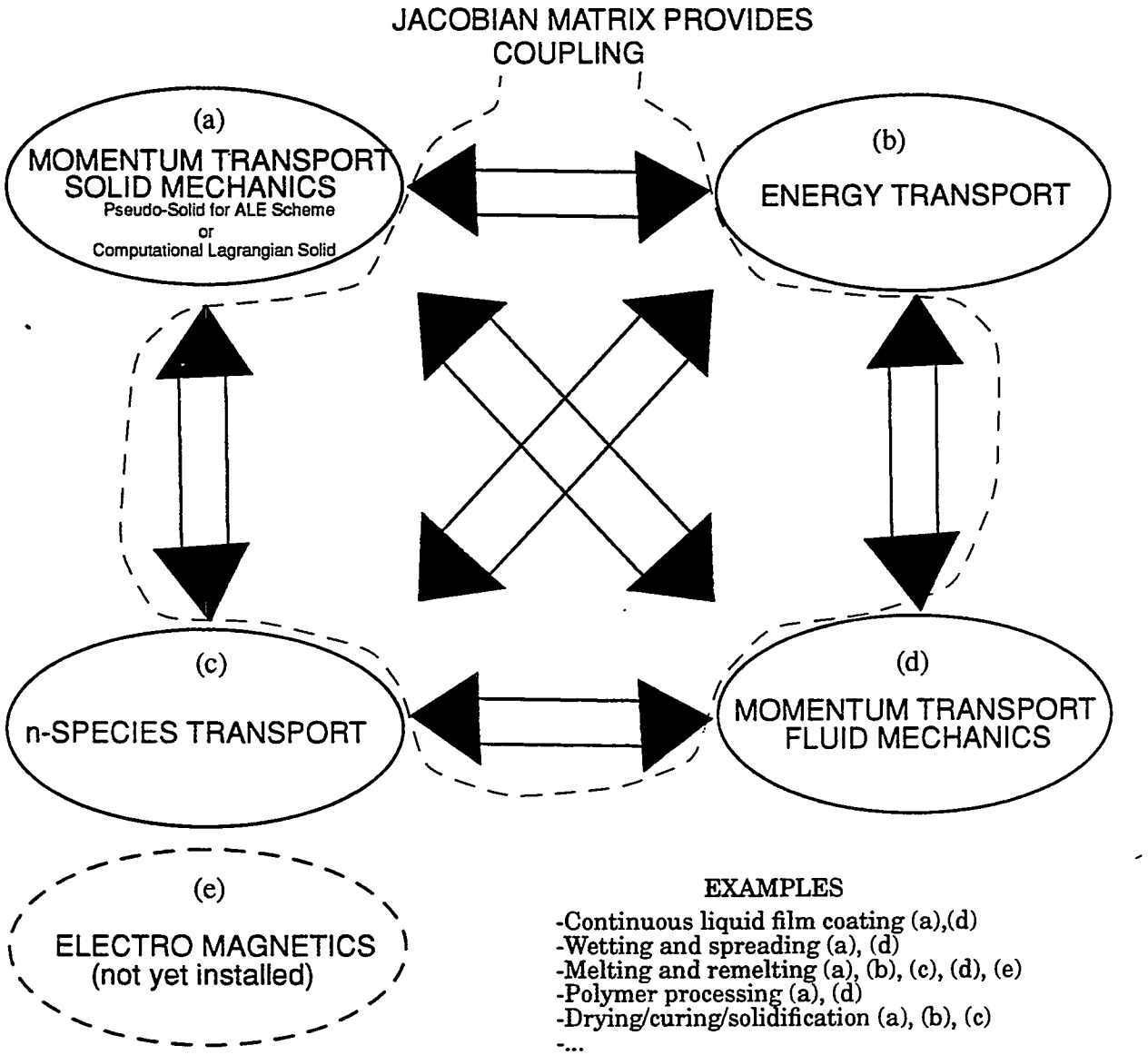
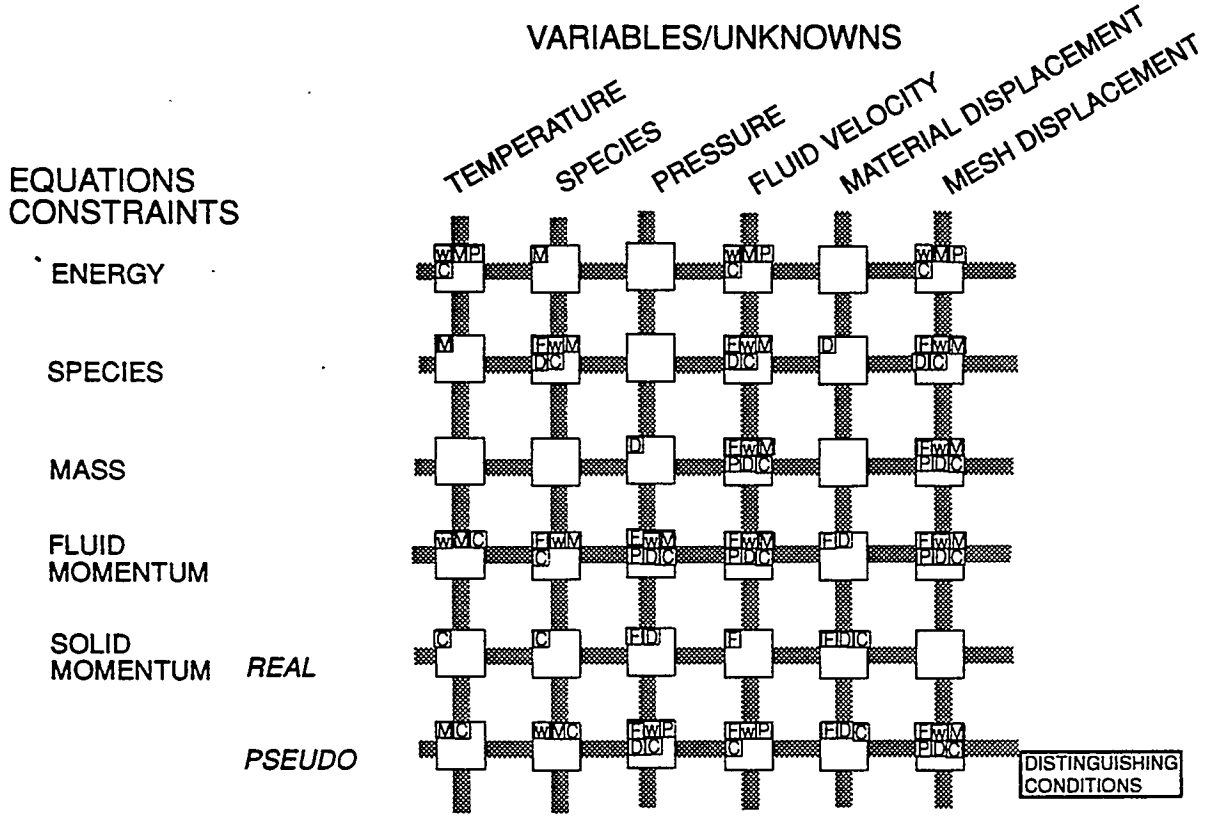


Figure 1 Main physics modules of GOMA together with the scope of potential applica-

the input required and the output generated. Chapters 4 and 5 serve as a guide and reference for the main problem description input and for the auxiliary material property data file formats. Chapter 6 gives a brief overview of the differential equations and some theoretical development all pertaining to the capabilities of GOMA. Finally, Chapter 7 provides a demonstration of the capabilities of GOMA as well as provide a brief tutorial on the use of the software package. This

chapter is designed to demonstrate usage by providing step-by-step instructions on generating a solution to a simple coating flow example. Appendix A describes several start-up strategies which are helpful in attaining a solution to free boundary problems. Developing these strategies requires some experience and intuition, given the nonlinear nature of most processing flows. The usual nonlinearities contributed by the advective terms of the conservation equations are compounded by the geometric nonlinearities associated with solving a free boundary problem.

FULL-NEWTON SOLUTION STRATEGY AND JACOBIAN MATRIX COUPLING



APPLICATION AREAS

- [F] Continuous liquid film coating
- [w] Wetting and spreading (solder)
- [M] Melting, remelting (superalloys)
- [P] Polymer processing
- [D] Drying/curing/solidification (sol-gel)
- [C] Crystal growth (semiconductors, oxides)

Figure 2 Pictoral representation of the global Jacobian matrix which provides full coupling between all physics modules of GOMA, annotated with Jacobian elements activated by application area. Each row and column includes both bulk and boundary terms.

2 Background Information

2.1 Program Features.

GOMA is a general purpose program designed for the solution of both steady and transient, two- and three-dimensional problems involving heat, mass, and momentum (solid and fluid) transport. The program has a unique feature that all boundaries and interfaces are treated as *free* (position unknown) or *moving* (position unknown or prescribed, but variable). If the material domain of interest is a solid, a Lagrangian formulation (i.e., the computational mesh follows the motion of material) of the momentum equations naturally leads to mass conservation and a natural parameterization of the boundaries and interfaces as material surfaces. If the material domain of interest is a fluid, then an Arbitrary-Lagrangian-Eulerian (ALE) formulation allows the boundaries to respond to constraint equations, hereafter referred to as *distinguishing conditions*. These conditions are responsible for the determining the location of all boundaries and interfaces, providing the necessary mathematical closure of the system of equations governing the free boundary problem. Distinguishing conditions available to the user fall into two classes, as described below (see also Appendix A).

The fully-implicit, pseudo-solid, unstructured mesh deformation algorithm sets GOMA apart from other finite element programs. All surfaces, internal and external, together with other geometric features such as corners and junction points, are permitted to move as part of the algorithm. The movement of boundaries, interfaces, and geometric features is dictated by a weighted residual statement of the distinguishing conditions, whether based on specific physical constraints or arbitrary conditions described by the analyst. The internal mesh deforms as if it were embedded in a deforming elastic solid continuum; with the mechanics of the solid governed by either infinitesimal (linear) or finite (nonlinear) deformation theory. Through Newton's method, the deformation of the mesh is determined simultaneously with all of the other physics of the problem.

The key connection between the mesh deformation and the physics of interest is accomplished through a library of distinguishing conditions. Currently, those conditions include (a) kinematic (material surface of a fluid), (b) isotherm (phase transition temperature, such as melting), (c) isoconcentration (d) geometric (either smooth plane curves or fixed point specifications.) As part of the required input for GOMA, the analyst specifies the associations between the particular distinguishing conditions and corresponding sets of material points of the initial pseudo-solid used to embody the mesh. Chapter 4 describes this process in more detail. Essentially, the algorithm causes smooth boundaries of the pseudo-solid to slide tangentially in a "frictionless" fashion. Further details of this algorithm and the corresponding equations can be found in (Sackinger, Schunk, and Rao 1995).

The class of problems treated by GOMA are those described by any one or a combination of the incompressible form of the momentum conservation equation for generalized Newtonian fluids, the energy conservation equation, the equations of quasi-static equilibrium of an elastic solid, and any number of additional or auxiliary species convection-diffusion-reaction equations. Currently,

GOMA has been tested with the following types of flow and heat transfer problems: (a) mixed convection with mesh parameterization of an isotherm, (b) melting, with a parameterization of the liquidus and solidus isotherms, (c) coating and related flows (slide coating, curtain coating, etc.), (d) polymer processing flows (e.g. fountain flow, planar and axisymmetric extrusion, simple mold filling), and (e) drying and shrinking of deformable porous media.

Coordinate systems accessible through this version of GOMA include two-dimensional Cartesian and cylindrical coordinates. Three-dimensional analysis is also available but has only been tested for Cartesian coordinate systems. A limited framework has been built within GOMA to use arbitrary orthogonal curvilinear coordinate systems, but this has not yet been extensively tested. Certain complicated boundary and interface expressions, as well as various solid mechanics strain expressions have not been implemented in this version.

Thermophysical properties in the bulk for all equations may be taken as constant or variable, with dependencies on any of the dependent and independent variables of the problem. General property variation models of this sort can be implemented with a user-defined subroutine capability. Moreover, a growing number of often-used standard models are supported within the core routines. These include a Carreau-Yasuda model for the generalized Newtonian viscosity and a Boussinesq source term for the fluid momentum equation that provides means for simulating flows with thermal and solutal buoyancy. Several other constitutive models are currently being installed. For purposes of maintaining customized, low-level control on the theoretical equations and as an aid to developing and debugging new models, there are also individual "equation term multipliers" which can be used to selectively activate or de-activate whole terms in the governing equations, or to provide a rough mechanism for quickly changing the relative scalings of equations and terms, or, in some cases, the coefficients for physical properties.

To enhance the capability for modeling problems in capillary hydrodynamics, e.g., coating flows, a boundary condition expressing the normal stress balance for two-dimensional Cartesian and axisymmetric problems has been implemented and tested. When capillary forces are activated, a pressure jump term (proportional to the mean curvature) is added to the normal component of the momentum flux balance at specified fluid material interfaces in a natural fashion. At three-phase boundaries (points in two dimensions) a contact angle condition and a surface tangent force condition may be applied. The former is used in place of a specified position on the mesh motion equations, and is best used to set static and dynamic contact angles, and the latter is an additional endpoint force which is added to the momentum balance, necessitated because the curvature term is integrated by parts. The current version of GOMA also includes the ability to model tangential shear forces along capillary surfaces, i.e., those originating from surface tension gradients caused, for example, by variations in temperature or species concentration. To access this capability requires a constitutive equation for the surface tension. A powerful low-level capability has been implemented which allows the user to select which degree of freedom, or variable, is associated with a particular boundary condition. Such a capability is useful at dynamic contact lines, where it is often desirable to replace the liquid-phase momentum equations with auxiliary constraint conditions.

Recently the solid mechanics module of GOMA, which was originally installed as a part of the pseudo-solid ALE mesh motion algorithm, has been exploited to solve problems in transport in deformable porous media and other outstanding problems of elastohydrodynamics. For modeling flow in non-deformable porous media, the Brinkman terms in the fluid momentum equations (cf. Gartling et al. 1995) may be activated. Alternatively, the generalized convection diffusion equations may be transformed into Darcy transport equations, which can be used for simulations of deformable porous media. For incompressible but deformable solids a pressure term was added to the solid momentum balance (e.g. rubber). In continuous shrinking or swelling solids, the dilation is proportional to changes in solvent concentration. In deformable porous media the solid deformation is coupled to the pressure in the fluid-filled interstices of the porous matrix. Several boundary conditions exist to apply normal tractions (i.e. compressive, tensile, or shear boundary forces) to solid surfaces. To effectively simulate coupled fluid/solid interaction problems, boundary conditions which balance the surface tractions exerted by the liquid and solid phases at the common interface have been incorporated.

2.2 Numerical Methods.

GOMA is based primarily upon the Galerkin/finite element method. The element library currently includes 4- and 9-node isoparametric quadrilaterals (i.e., Q_1 and Q_2 interpolations) (in two dimensions) with available interpolations for linear discontinuous (P_1) or piecewise constant (P_0) variables, and 8-node isoparametric hexahedral elements (three dimensions), also with available piecewise constant interpolations. The overall solution algorithm centers around a fully-coupled Newton-Raphson iterative scheme for solving the nonlinear algebraic equations which results from the finite element discretization. That is, all active equations and boundary conditions are solved simultaneously in a single matrix system at the same time plane and during the same Newton iteration. The sparse matrix system is stored in a compressed row format (cf, Hutchinson et al 1995, Schunk and Shadid 1992), implying that the nonzero matrix elements are stored in a single linear array together with column pointers in an associated auxiliary array. If the matrix system is not too poorly conditioned, then iterative solvers of the generalized preconditioned conjugate gradient-type can be used to solve the system (see Hutchinson et al. 1995, Schunk and Shadid 1992). When the Galerkin formulation of the incompressible Navier-Stokes equations is activated, an ill-conditioned matrix system is obtained, and the only robust option available is a general LU decomposition (Gaussian elimination), which "self-optimizes" by setting up a linked list of matrix elements during the first iteration based on equation ordering (cf. Kundert and Sangiovanni-Vincentelli 1988). Note, too, that certain distinguishing conditions applied as part of the solution of free and moving boundary problems can result in poorly-conditioned matrix systems.

The overall differential-algebraic system of equations may be advanced in time with implicit time-integration techniques (simple backward Euler and Adams-Bashforth predictor, trapezoidal corrector algorithms). Time marching offers an alternative, albeit indirect, route to attaining solutions to steady equations (see Appendix A), as well as providing the capability of simulating pro-

cess transients directly. Automatic time step control based on current truncation error is also available.

Perhaps the most complicated part of the algorithm is the construction of the Jacobian sensitivity matrix. Because the mesh point positions are actually unknowns in a free or moving boundary problem, that matrix must include sensitivities of each weighted residual equations with respect to each of the mesh variables unknowns that can affect the value of the residual. Unfortunately, almost every term of the bulk equations and many boundary conditions contribute to this sensitivity, mainly through gradient operators and surface normal and tangent vectors (see Kistler and Scriven 1983), as well as through dependencies on mesh position of the determinant of the elemental Jacobian transformation matrix that maps between a fixed unit element and any element in the computational domain. Great care has been taken to include analytical expressions for all of these mesh sensitivities. However, some of this task inevitably falls to the user when implementing user-defined boundary conditions, material property models, and constitutive equations, particularly when any of these quantities depends directly on spatial position, or depends on spatial gradients of other variables. In order to maintain the strong convergence properties of Newton's method, these sensitivities must be specified in those user-defined routines. Examples that show how this is done in practice are given in Chapter 7. To aid in this task, a debugging option is available which computes a numerical finite-difference approximation of the global Jacobian matrix and compares it with its analytical counterpart. This tool enables users and developers to check the consistency of newly-created equations (whether bulk or boundary constraints) with their corresponding analytic Jacobian contributions.

2.3 Portability, Software Library Infrastructure, and Code Accessibility.

GOMA is written in the C programming language (specifically K&R C with some ANSI extensions; Kernighan and Ritchie 1988), and has been ported to a number of UNIX platforms, including SunOS 4.1.3, Solaris 2.3, HP-UX 9.05, AIX 3.2, IRIX 5.2, and UNICOS 7.0.6, with the Solaris version being the most actively maintained. Many of the machine dependencies in the program have been isolated using C preprocessor directives. Some of the machine dependencies that occur in the I/O routines are insulated from the user by software libraries. Building GOMA requires EXODUS II v2.02 (Schoof and Yarberry 1994), SPARSE 1.3 (cf. Kundert and Sangiovanni-Vincentelli 1988) and NetCDF v2.3.2 (Rew et al. 1993) libraries. The first of these is part of the SEACAS system at Sandia National Laboratories (Sjaardema 1993); the latter two libraries are available publicly. Generally, pre- and post-processing is performed outside of GOMA, although some post-processing of results is available within the program. This separation of the functionality permits the use of alternative solid-modeling and mesh-generation software, and visualization packages of choice, insofar as they may be interfaced with the EXODUS II finite element data model.

Pre-processing options include mesh generation via a FASTQ (Blacker 1988), CUBIT (Blacker et al. 1994), or PATRAN (PDA Engineering 1990). These mesh generators currently support and will output a finite element database in the EXODUS II format, the last through the PATEXO translator.

Post-processing options include BLOT (Gilkey and Glick 1989) and AVS. The latter visualization package, particularly AVS/Express, may be conveniently accessed using MUSTAFA (Glass 1995) or FEAVR (Schoof 1995).

Since GOMA is built around the EXODUS II finite element data model, there are numerous options available for communication with other analysis codes that also exchange data via the same EXODUS II data model. Recent modifications to GOMA permit not only the initialization of unknown values from an EXODUS II file, but also the ability to incorporate field variables into the analysis that are not unknowns. For example, the quasi-static electromagnetic program TORO II has been used to compute electric fields and current fluxes on a specified finite element mesh that are input to GOMA through the EXTERNAL FIELD data card for the input problem description.

3 Code Structure and I/O

3.1 General Code Structure

The files which comprise the source code for GOMA have been organized into six main categories, with a file naming convention based upon different prefixes. While largely historical, the categories are listed here.

rf_ -- Stands for "reacting flow". Basically these routines are slowly being migrated to other categories as major changes are made to them. These routines originated in the original SALSA code of early 1993 (cf. Shadid and Moffat 1995), but have undergone substantial revision since that time.

mm_ -- Stands for "moving mesh". These routines are comprised largely of those which were written to support a general ALE environment, although some are just standard infrastructure routines written after the original branch from SALSA.

el_ -- Stands for "element". These routines are responsible for all element library definitions, element database input, and other related functions.

md_ -- Stands for "machine dependent". Most of the machine dependent functions are pulled into these routines.

bc_ -- Stands for "boundary conditions". These routines are responsible for directing the application of all boundary conditions and actually applying those conditions. There are, however, some boundary condition activities being performed in mm_ routines.

user_ -- Denotes "user-defined subroutines".

sl_ -- Denotes "solver". These routines are all associated with the linear equations solvers used to solve the resulting matrix system.

Although these categories are not strictly adhered to, they are useful for general guidance when studying the source code.

3.2 Files for Data Input

The file I/O structure is diagrammed in Figure 3. Input to the program is divided into five categories: (1) command-line, (2) problem description file, (3) material files, (4) ASCII restart/continuation file, and (5) EXODUS II database file. GOMA is basically set up to run in batch mode, i.e., no input is required on the command line or after the run command is issued. There are, however, several command-line switches which can be used to redirect I/O, control the level of I/O, and activate debugging options.

The *problem-description* file is by default called "input" but can be renamed with the **-i** switch on the command line. This file contains the general description of the problem and directions to GOMA on how to solve it (see Chapter 4). The file is split into seven sections: (1) File Specifications (Section 4.1) which directs I/O, (2) General Specifications (Section 4.2), (3) Time Integration Specifications (Section 4.3), (4) Solver Specifications (Section 4.4), (5) Boundary Conditions

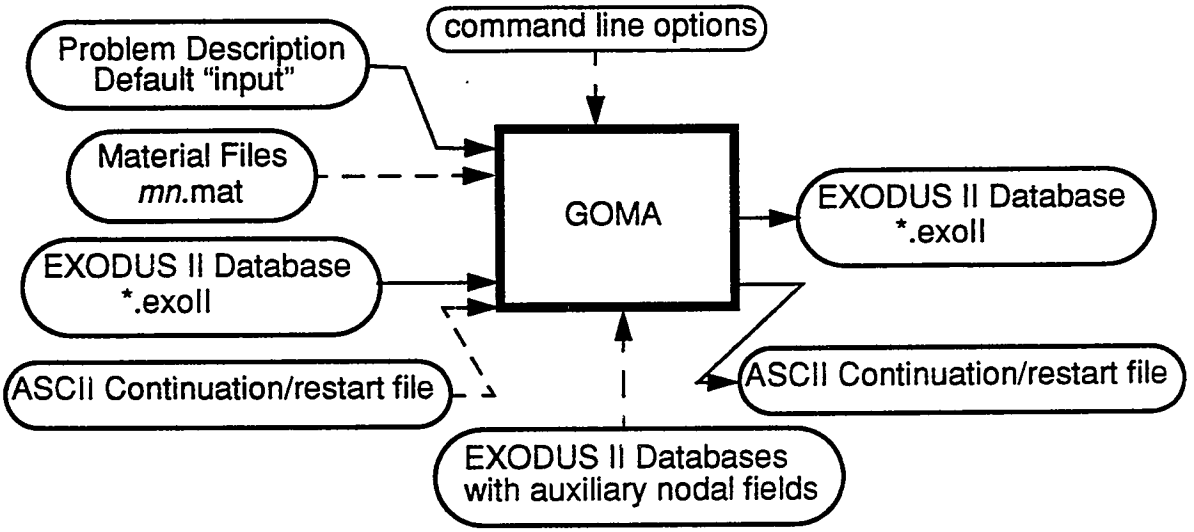


Figure 3 I/O structure for GOMA. Dashed lines indicate that the files are not required.

Specifications (Section 4.5), (6) Problem Description (Section 4.6), and (7) Post Processing Specifications (Section 4.7). The file format is described in detail in Chapter 4. Incidentally, the structure of the data input routines is divided roughly along the same lines as the input data file itself.

The *material description files* (of the format "[material name].mat", see 4.6.2) contain all material property data and material property model and constitutive model specification. The names of these files are specified in the problem description file. The format and options available in these files are described in Chapter 5.

The *ASCII restart/continuation files* (may have any name, see 4.1.3 and 4.1.4) contain an ASCII list of the solution vector (values of field variables at nodes), which can be used as an initial guess for successive runs of GOMA. The names of these files are specified in the problem description file, but may be changed with the **-c** (for input) or **-s** (for output) command-line options. These restart files are "recyclable", in the sense that output from one GOMA simulation may be used as input to another GOMA simulation under certain restrictions.

The *EXODUS II database files* (of the format "[any name].exoII", see 4.1.1 and 4.1.2) contain a description of the finite-element structure for the current problem. These files define the mesh, material blocks, and boundary sets. The output EXODUS II database file also contains the nodal values of all the field variables and post-processing variables, in addition to containing a clone of the input EXODUS II mesh information. The names of these files are specified in the problem description file, but may be changed with the **-ix** (for input) or **-ox** (for output) command-line

options. The only EXODUS II file required when running GOMA is the one containing the current problem mesh. All others are either output for postprocessing or used to supply auxiliary external fields (e.g. magnetic fields).

3.3 Command-line Arguments

GOMA can be run using only the input files (all four listed above) to describe the problem and to direct the input and output; in this case GOMA is run using the command "goma" without any arguments. However, command-line arguments offer additional flexibility for redirecting input or output and for adjusting common run-time parameters. The general command line for running GOMA is:

goma [**-nd**] [**-se** *fn*] [**-so** *fn*] [**-i** *fn*] [**-c** *fn*] [**-s** *fn*] [**-ix** *fn*] [**-ox** *fn*] [**-d** *int*] [**-r** *dbl*] [**-a** *args*]

Here *fn* denotes "file name", *int* denotes "integer", *dbl* denotes "float or double" and *args* denotes multiple sub-options or file names. The input line is parsed into options, which are preceded by a single hyphen (-) and arguments, which normally are *fn, int,* or *dbl* not preceded by a hyphen. The default, if no options are specified, is the -input option (e.g. "`goma input.alt`" is the same as "`goma -i input.alt`). The following is a list of the command-line options and their descriptions (there are two ways to specify each option in abbreviated or verbose form).

-a *args* **-aprepro** *args* — Preprocess input files thru the APREPRO preprocessor [w/ *args* as arguments to APREPRO] before reading into goma. With this option, GOMA performs a UNIX system() call to run APREPRO which will preprocess the input file and the material data files. The input file is preprocessed from "input" or the filename specified by the -input option and written to "tmp.input". Likewise, the material data files are preprocessed from "[material name].mat" to "tmp.[material name].mat". After the "-a" on the command line, options for APREPRO are preceded by two hyphens (--). For example, "goma -i input.pre -a CONSTANT1=0.2 --vd" will preprocess input.pre and the material data files specified in input.pre using APREPRO, and will pass the arguments -vd (prints version number and values of all variables to screen) and CONSTANT=0.2 (sets CONSTANT equal to 0.2 for preprocessing) to APREPRO; the preprocessed files will be "tmp.input" and "tmp.[material name].mat".)

-c *fn* **-contin** *fn* — Change the name of the ASCII input file (specified in problem-description file) to *fn*. (e.g. "goma -c old.soln.dat" uses the file "old.soln.dat" as the ASCII input file) Note that this option has no effect if the initial guess is not read from the ASCII file, i.e. unless "Initial Guess = read" is specified in the input file (cf. Chapter 4).

-d *int* **-debug** *int* — Change the debug flag to *int*. This option is convenient when debugging and the user wants to see more output from GOMA. (e.g. "goma -d -2" will run GOMA with the Debug_Flag set to -2). Higher values

		generally produce more output.
-h	**-help**	Prints a helpful message with brief descriptions of these command line options.
-i *fn*	**-input** *fn*	Redirect GOMA to read the problem description file from *fn*. The normal default option is to read from a file named "input".
-ix *fn*	**-inexoII** *fn*	Redirect GOMA to read the input EXODUSII database file (often called "in.exoII") from *fn*.
-nd	**-nodisplay**	Do not display the run-time information on the screen. With this option, GOMA sends the stdout and stderr output to temporary files that are removed at the end of the run. This command takes no arguments.
-ox *fn*	**-outexoII** *fn*	Redirect GOMA to write the output EXODUSII file (often called "out.exoII") to *fn*.
-r *dbl*	**-relax** *dbl*	Change the value of the Newton relaxation parameter to *dbl*. This is convenient if a few Newton steps with relaxation are desired before using full Newton. (e.g. "goma -r 0.1" will use Newton's method with updates one-tenth of the normal value, as described in Section 4.4).
-s *fn*	**-soln** *fn*	Redirect GOMA to write the output ASCII file (normally called "soln.dat") to *fn*.
-se *fn*	**-stderr** *fn*	Redirect the standard error from GOMA to *fn*. This output is comprised of more urgent diagnostic error and timing messages.
-so *fn*	**-stdout** *fn*	Redirect the standard output from GOMA to *fn*. This output is comprised of less urgent informational messages.

The primary purpose of the command-line options is to allow the user an easy way to redirect the input and output of GOMA or to quickly change problem specifications. Most of the options are overrides of information in the problem description file, so that in some cases it may be easier to edit the problem description file than to use command-line arguments.

4 Data Input-- Problem Description File

The **input** file for GOMA contains the overall description of the problem to be solved together with instructions on solution strategy. The file is split into seven sections: (1) File Specifications (Section 4.1) which directs I/O, (2) General Specifications (Section 4.2), (3) Time Integration Specifications (Section 4.3), (4) Solver Specifications (Section 4.4), (5) Boundary Conditions Specifications (Section 4.5), (6) Problem Description (Section 4.6), and (7) Post Processing Specifications (Section 4.7).

The format of each section below indicates, for each part of the input file specification, the data cards that may be used, followed by the options available for each card and the necessary input data. All input data are specified in a free field format with successive variables separated by blanks or commas. Bold/underlined type denotes a unique string which GOMA looks for in the input deck. The "␣" symbol denotes a single blank space. When such single spaces are specified they are significant; otherwise, the user is free to use white space on a command line. Note, too, that the input parser is case sensitive! The identifying string for a particular specification is followed by an "=" character. In the command descriptions that appear below, the bold/italic type following the "=" denotes text that the analyst must choose. Required input and possible options for each command card are described in this chapter.

Because of the nature of the input parser, the user is free to comment the input deck in any way, so long as the character strings in the comments do not contain the exact strings described in this section, at the beginning on the same line. Simply for the sake of uniformity, it is recommended that a comment card convention of placing some delimiting symbol at the beginning of each comment line be adopted, e.g., $, #, \, etc. Moreover, employing some the basic text processing capabilities provided in the ACCESS system (Sjaardema 1993) makes it possible to connect both the model generation input file, e.g., FASTQ or CUBIT, with the input deck for GOMA; for example, the "include" statement in APREPRO (Sjaardema 1992) makes it convenient to include geometrical information from a FASTQ input file in the GOMA input file for use with commands like PLANE and SPLINE (see 4.5.2 below) that make use of global problem geometry. APREPRO also enables one to generate customized model parameterizations with the algebraic preprocessing capability (see Appendix A). Finally, use of a text preprocessor like APREPRO enables the analyst to give more meaningful names to entities such as side sets, node sets and element blocks than the internal names, which are simple integer identifiers.

The order of the input cards is significant; omitting a required card often will result in an error message from GOMA. A good strategy to avoid such errors is to copy a current version of a working input file and then make changes to it. However, as noted below, some cards are optional.

Some sections such as boundary condition specification and equation specification are not order dependent, but only the number of boundary conditions or equations which are specified by the "Number of BC" and "Number of EQ" cards will be read (regardless of the number of cards in the file). That is, after the specified number of individual equation or boundary condition cards is read, any remaining cards are ignored.

Figure 4 shows an example problem description input deck, indicating which cards (lines) are optional and which are required. The remainder of this chapter describes each card in detail.

4.1 File Specifications

4.1.1 FEM□file

INPUT: = ***exoII_file_name***

exoII_file_name: The permissible values are any EXODUS II file name The recommended form of the file name is "**prefix.exoII**", where "prefix" can be used as a problem type descriptor, because the pre- and post-processors (like AVS) might require the ".exoII" suffix. The length of this filename variable currently limits filenames to 85 total characters.

4.1.2 Output□EXODUS□II□file

INPUT: = ***exoII_file_name***

exoII_file_name: The permissible values are any EXODUS II file name in the form "**prefix.exoII**", where "prefix" can be used as a output file descriptor. This file replicates the input mesh and boundary condition information exactly as it was provided in 4.1.1, but appends the solution field information appropriate to the problem type. If the name of the output EXODUS II file is identical with the name of the input EXODUS II file, then no replication of the input mesh data is performed and any results are simply appended to this single file.

4.1.3 GUESS□file

INPUT: = ***file_name (input)****

This file provides the initial guess for continuation or time integration. Its current format is just a list of unformated floating point numbers in the order of the unknown map. The file is ASCII and has the same format as the file described in 4.1.4. *file_name* is any file name and is read only if the keyword in 4.2.4 is set to **read.** A solution file from a previous simulation may be used.

4.1.4 SOLN□file

INPUT: = ***file_name (output)****

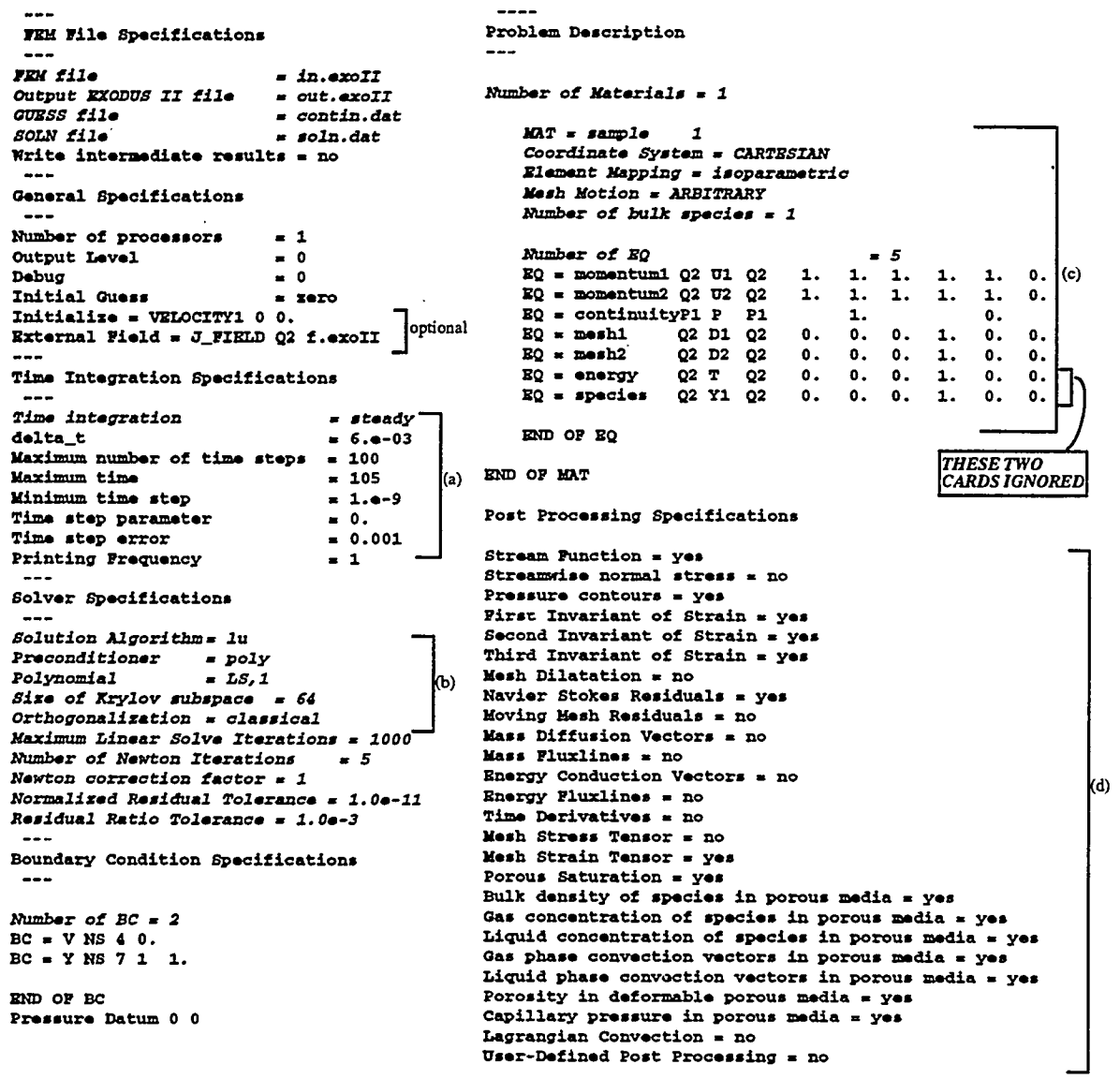


Figure 4 Problem description input deck. Italic type denotes required data cards (lines) and plain type denotes optional cards or cards that in number correspond to the designation above them, e.g., "Number of BC" or "Number of EQ". (a) These cards are optional if the "steady" option is chosen on the Time Integration card. (b) These cards are optional if the "lu" solver is chosen on the Solution Algorithm card. (c) This group of cards is repeated for each different material block in the EXODUS II database file. (d) These cards are all optional and can appear in any order in the input file.

This file provides the initial guess for continuation or time integration. Its format is just a list of unformated floating point numbers in the order of the unknown map that includes every degree of freedom in the problem. The file is ASCII and is written into the file name described in 4.1.3. *file_name* is any file name and is to be copied into the file name in 4.1.3 for continuation. Beyond the first column of numbers in this file may appear other information that is sometimes useful in determining the name and location of the corresponding degree-of-freedom.

4.1.5 Write Intermediate Results

INPUT: = ***keyword***

The permissible values here are **yes** or **no**. If **yes**, the code will output the latest iteration in a file named tmp.*i*.d, where *i* is the Newton iteration number. The format will be similar to the ASCII results data described in the previous two sections. This is useful to guard against machine crashes or accidental job kills, particularly for very large problems. Also, the output EXODUS II database (see Section 4.1.2) accumulates the intermediate iterations as time planes of the solution. This can be a useful debugging tool, giving the user the ability to use highly relaxed Newton iterations to see how a free boundary problem diverges. If **no**, only the last iteration is written to **SOLN file** name in Section **4.1.4** and only the final converged iteration is output to the EXODUS II file.

4.2 General Specifications

4.2.1 Number of processors

INPUT: = ***integer_const***

Specifies the number of processors to *no_proc*: Permissible values are 1 for the present revision of GOMA. No provisions for multiple processors at this time. This card is optional. The default is no_proc = 1.

4.2.2 Output level

INPUT: = ***integer_const***

Specifies the level of information output to standard out. *integer_const*: The permissible values are **1** through **4**, depending on the level of informational (debugging) output desired; higher values of the output level will produce more output on the standard output and standard error output channels. This card is optional and the default output level is **1**.

4.2.3 Debug

INPUT: = ***integer_const***

Specifies the level of information output to standard out. *integer_const*: The permissible values are **-2** through **4** depending on the level of informational (debugging) output desired. A value of **-1** or **-2** triggers a comparison of the analytical Jacobian and the numerical Jacobian, which can be used to check the compatibility of the analytical residual equations and Jacobian. A value of **-1** triggers a comparison of the Jacobians in un-scaled form, and a value of **-2** triggers a comparison of the jacobians scaled by the sum of each row of the analytical jacobian (this helps suppress small errors in large jacobian entries). This card is optional and the default value is **0**.

4.2.4 Initial□Guess

INPUT: = ***character_string***

Directs how the entire unknown vector is initialized. Three mechanisms are provided to set the entire solution field to a constant value based on values of *character_string*. Corresponding permissible specifications are: **zero**, for an initial guess of zeroes in all elements of the unknown vector; **one**, for an initial guess of ones; and **random**, for a random-number generated guess. A second class of mechanisms exists to read the initial guess from a data file. Permissible values are: **read**, to obtain the initial guess from the initial guess file described in 4.1.3 and 4.1.4 above; and **read_exoII**, to obtain the initial guess from the exodus II file output in 4.1.2 above. This card is optional, and the default value is **zero**. Note that if the read mechanisms for initialization are activated, that GOMA will attempt to read in an initial guess value for every active variable in the input deck; extraneous variables in the ".exoII" file are ignored.

4.2.5 Initialize

INPUT: = ***character_string, integer_const, floating_point_const***

Provides a mechanism to set one of the field variables to a constant value across the whole domain. The variable name is specified by *character_string:* The permissible values are **VELOCITY1**, **VELOCITY2**, **VELOCITY3**, **MESH_DISPLACEMENT1**, **MESH_DISPLACEMENT2**, **MESH_DISPLACEMENT3**, **MASS_FRACTION**, **TEMPERATURE**, and **PRESSURE**. The species number to be initialized is specified by *integer_const* if the character_string is MASS_FRACTION; otherwise the *integer_const* should be set to zero. The *floating_point_const* is the value to which the variable should be initialized. This card is optional, and multiple applications of this card are valid; GOMA automatically counts the number of Initialize cards. The variables are initialized by this card after reading the initial guess from a file, so it can be used to override the value in an input file.

4.2.6 External□Field

INPUT: = ***character_string1, character_string2, exoII_file_name***

This card provides a mechanism to read in field variables that are nodal variables from an EXODUS II file. That field variable can then be accessed in any user-defined subroutine, as is described below. *character_string1* is the name of the nodal field to be read, and should correspond to a name of a nodal variable in the EXODUS II file. If *character_string1* is **VX**, **VY**, or **VZ**, then the fields are automatically loaded up in the appropriate velocity component, so they can be used in an advection-diffusion analysis, i.e., **VX**, **VY**, **VZ** are reserved names for *character_string1*. The value of *character_string2* pertains to the interpolation of the nodal variable. It can be either **Q1** or **Q2**, corresponding to linear or quadratic interpolation in two or three dimensions. *exoII_file_name* is the name of the EXODUS II file from which the field is to be read. The following examples shows how this is used:

```
External Field = VX Q2 velocity.exoII
External Field = VY Q2 velocity.exoII
```

These cards would direct GOMA to read two fields, "VX" and "VY", from the file named "`velocity.exoII`". These fields are then automatically accessed when the advection term is left on in the energy or species equations (see Section 4.6). In other words, without solving the momentum equations, one can access an external velocity field for advection diffusion problems. Another example:

```
External Field = JX_FIELD Q2 fields.exoII
External Field = JY_FIELD Q2 fields.exoII
External Field = BTHETA_FIELD Q2 fields.exoII
```

These three cards can be used to read in two components of a current density field (JX_FIELD, and JY_FIELD), and the azimuthal component of a magnetic field (BTHETA_FIELD) from the file "`fields.exoII`", which was generated by some other analysis code. These fields are then accessed in the user-defined subroutines as

`fv->external_field[0]`, `fv->external_field[1]`, and `fv->external_field[2]`,

respectively, as an interpolated value at an integration point.

4.3 Time Integration Specifications

4.3.1 Time□integration

INPUT: = ***character_string***

Used to specify transient or steady state calculation. *character_string:* The permissible values are **STEADY**, for a solution to the steady (time-derivative free) equations, and **TRANSIENT**, the transient simulations. If option **STEADY** is chosen then cards 4.3.2 through 4.3.8 are not

needed.

4.3.2 delta_t =

INPUT: =*floating_point_const*

Sets value of time step for TRANSIENT simulations. *floating_point_const:* Any floating point number to indicate in the appropriate units the time step. If a constant time step size is desired, set delta_t to be a negative number, e.g. -1.0e-6, and the code will use a constant (positive) time step. In this case, the size of the time increment will only be changed if convergence problems occur, in which case delta_t will be reduced by half until convergence is achieved.

4.3.3 Maximum number of time steps

INPUT: = *integer_const*

Sets the maximum number of time steps. *integer_const:* Permissible value is any integer constant to set limit on the number of time steps taken.

4.3.4 Maximum time

INPUT: =*floating_point_const*

Sets value of maximum time achieved, in the same units as delta_t in 4.3.2.

floating_point_const: Any floating point number to indicate the maximum time allowed.

4.3.5 Minimum time step

INPUT: =*floating_point_const*

Sets value of minimum allowable time step, in the same units as delta_t in 4.3.2. *floating_point_const:* Any floating point number to indicate the minimum time step allowed.

4.3.6 Time step parameter

INPUT: =*floating_point_const*

This parameter allows the user to vary the time integration scheme from backward-Euler, to Crank-Nicholson or forward-Euler. Note, the purely explicit forward-Euler integration scheme will not work for problems with algebraic constraints in time, e.g. no time derivatives in the equation, such as the mesh equations or the continuity equation.

floating_point_const: 1. = Forward-Euler, 0.5 = Crank-Nicholson, 0. = Backward-Euler

4.3.7 Time□step□error

INPUT: =*floating_point_const, 5(integer_const)*

The time step error controls the adjustable time step size based on the solution norm. The 5 integer constants allow the user to choose which variable to use in the error norm calculations. The first integer constant is a flag for mesh, the second is for velocities, the third is for temperature, the fourth is for concentration and the fifth is for pressure. 0 means do not include and 1 means include. Note, currently pressure is not included in the norm.

floating_point_const: Any floating point number to indicate the maximum time allowed.
integer_const 1: 1 = include mesh variables in norm calculations, 0 = don't include mesh
integer_const 2: 1 = include velocity variables in norm calculations, 0 = don't include velocity
integer_const 3: 1 = include temperature variables in norm calculations, 0 = don't include T
integer_const 4: 1 = include concentration variables in norm calculations, 0 = don't include C
integer_const 5: 1 = include pressure variables in norm calculations, 0 = don't include pressure

4.3.8 Printing□Frequency

INPUT: = ***integer_const***

This sets the printing frequency. For example, if this is set to 5, GOMA will only print the solution to the exodus or soln.dat file every 5 times steps.

integer_const: Any integer number, other than 0, to indicate the desired printing frequency.

4.4 Solver Specifications

4.4.1 Solution□Algorithm

INPUT: = ***character_string***

Directs the solution of $Ax=b$ matrix system of equations using the KRYSOLVE iterative solver package (Schunk and Shadid 1991). *character_string* : Permissible values are **cg**, conjugant gradient, **gmres**, generalized minimum residual, **cgs**, conjugate gradient squared, **qmrs**, smooth conjugant gradient squared, **cgstab**, stabilized conjugant gradient, and **lu**, LU decomposition (Gaussian elimination). If **lu** is chosen, cards 4.4.2 through 4.4.6 are not required.

4.4.2 Preconditioner

INPUT: = ***character_string***

For all values of the *Solution□Algorithm* keyword except **lu**, this provides the selection of the preconditioner for iterative solutions. For *character_string*, the permissible values are **none**, for no preconditioning, **jacobi**, for Jacobi (diagonal scaling) preconditioning, **sgs**, for sym-

metric Gauss-Seidel preconditioning, **poly**, for polynomial preconditioning, **ilu**, for incomplete LU decomposition preconditioning (nonsymmetric systems only), and **icc**, incomplete Cholesky precondition (symmetric systems).

4.4.3 Polynomial

INPUT: = ***character_string, integer_const***

Allows selection of type and order of the polynomial preconditioning. The information on this card is not used if the preconditioning type is not "`poly`" (see Section 4.4.2). The value of *character_string* can be **LS**, for least-squares polynomials or **NS** for Nuemann series polynomials. The value of *integer_const* can be 1, 3, 5, and 7, indicating the order of the polynomial.

4.4.4 Size□of□Krylov□subspace

INPUT: = ***integer_const***

Specifies the number of orthogonalization directions for the **gmres** option in 4.4.1 above. *integer_const*: Permissible values are any positive integer constant less than or equal to the order of the matrix.

4.4.5 Orthogonalization

INPUT: = ***character_string***

character_string : Permissible values are **classical**, for classical orthogonalization and **modified**, for modified orthogonalization.

4.4.6 Maximum□Linear□Solve□Iterations

INPUT: = ***integer_const***

integer_const: The number of iterations allowed to reach a tolerable solution error to the Ax=b system of equations.

4.4.7 Number□of□Newton□Iterations

INPUT: = ***integer_const1, integer_const2***

integer_const1: number of Newton iterations allowed for the non-linear iteration loop. For unrelaxed Newton iteration with a good initial guess, five or six iterations should be sufficient to achieve convergence for most problems. One iteration will suffice for problems that are linear; two can be specified, the second iteration verifying that the residual norms are small. More iterations may be required for relaxed Newton iteration schemes using the correction factor described in Section 4.4.8. *integer_const2* is the reformation stride for the Jacobian matrix and can be used to invoke a modified Newton iteration. *integer_const2* is not required on the card and, if missing, the stride is set to unity. This capability enables the user to save on assembly

time when near a solution, particularly when doing transient simulations. (N.B. as of 11/28/95 this modified Newton feature is not functional).

4.4.8 Newton□correction□factor

INPUT: =*floating_pt_const*

Indicates the damping (or relaxation) factor for the Newton updates. *floating_pt_const:* Permissible values are any floating point constant between and including 0 (no update) and 1(full Newton iteration update). This parameter can also be controlled from the command line (see -r option in Section 3.3).

4.4.9 Normalized□Residual□Tolerance

INPUT: =*floating_pt_const*

floating_pt_const: Value indicates the maximum L_2 norm of the weighted residual vector allowed for declaration of "converged" in the Newton iteration loop.

4.4.10 Residual□Ratio□Tolerance

INPUT: =*floating_pt_const*

This card is used to set the stopping criterion for the iterative solution options of *4.4.1 floating_pt_const*: Value of epsilon ε for stopping criterion

$$\frac{\|Ax-b\|_2}{\|A\|_\infty} < \varepsilon$$

This condition must be satisfied together with $\frac{\|R\|}{\|R_0\|}$. Here A is the matrix, x is the solution vector and b is the right hand side vector.

4.5 Boundary Condition Specifications

4.5.1 Number□of□BC

INPUT: = *integer_const*

integer_const: Value indicates the number of boundary conditions (number of boundary condition, BC, cards) to follow. If there are more BC cards than specified, GOMA ignores the extras (i.e. only the first Number□of□BC cards are read). If this *integer_const* is set to -1, GOMA will automatically count the number of BC cards between the Number□of□BC card and the END□OF□BC card (see 4.5.7). This latter usage is generally preferable if a large number of boundary conditions are to be specified.

If more than one boundary condition on the same variable is given, only the last one is ap-

plied. If the number of boundary conditions is less than the amount specified, GOMA will stop with an error.

The remaining card types allowable for this section are described in sections 4.5.2 through 4.5.6. Each section contains a table of allowable boundary conditions and the format for input. Some of these boundary conditions are redundant or misnamed, but have been retained for historical reasons. This section may undergo substantial revision in the future as the formalism for boundary condition application undergoes re-evaluation.

4.5.2 BC (boundary conditions requiring 4 floating-point consts)

INPUT: = ***bc_name, bc_type, bc_id, floating_pt_const1, floating_pt_const2...***

These cards are used to provide all boundary condition information. The current allowable values for *bc_name*, *bc_type*, *bc_id*, and *floating_pt_const1*, etc. are given here in tabular form. In Table 4.1 below **NS** indicates a node set *bc_type* (NODEBC or POINBC in EXODUS II) and **SS** indicates side set *bc_type* (ELEMBC in EXODUS II). *bc_id* is an integer constant identifying the boundary condition (set in EXODUS II). All Dirichlet node-set type boundary conditions (i.e. T, U, V, W, DX, DY, DZ, Y) can be applied as a "hard set" condition, or with a residual equation, depending on whether a second floating point number is present. The following table indicates the allowable values of the character string *bc_name* and corresponding meaning of the floating point constants.

Table 4.1 Boundary Conditions with Floating Point Constants

bc_name/ bc_type	brief description	floating_pt_ const1	floating_pt_ const2	floating_pt_ const3	floating_pt_ const4
T/ NS	Set constant temperature - Dirichlet	Value of temperature	If present and not -1.0, condition is applied as residual equation	N/A	N/A
QSIDE/ SS	Constant heat flux specification	Value of heat flux			
QUSER/SS	User-defined heat transfer model	Can have an arbitrary number of floating point numbers, which can be accessed in user-defined routine.			

Table 4.1 Boundary Conditions with Floating Point Constants

bc_name/ bc_type	brief description	floating_pt_ const1	floating_pt_ const2	floating_pt_ const3	floating_pt_ const4
QRAD/ SS* QCONV/ SS	Convective heat transfer flux h(T-T0)	Heat transfer coefficient	Sink Temperature		
DX/ SS, DX/NS[a] DY/SS, DY/NS DZ/ SS, DZ/NS	Constant displacement - Dirichlet	X-displacement Y-displacement Z-displacement	If present and not -1.0, condition is applied as residual equation		
PLANEX/SS* PLANEY/SS* PLANEZ/SS*	Surface (solid) boundary description (penalty on X, Y, or Z mesh equation)	"a" in function f(x,y,z)=ax + by+ cz + d = 0	"b" in function f(x,y,z)=ax + by+ cz + d = 0	"c" in function f(x,y,z)=ax + by+ cz + d = 0	"d" in function f(x,y,z)=ax + by+ cz + d = 0
PLANE/SS	Surface (solid) boundary description - Rotated	"a" in function f(x,y,z)=ax + by+ cz + d = 0	"b" in function f(x,y,z)=ax + by+ cz + d = 0	"c" in function f(x,y,z)=ax + by+ cz + d = 0	"d" in function f(x,y,z)=ax + by+ cz + d = 0
SPLINEX/SS* SPLINEY/SS* SPLINEZ/SS*	General surface (solid) boundary description (penalty on X, Y, or Z mesh equation)	Constant to parameterize any f(x,y,z)=0 Function input in user defined routine fnc.	Constant to parameterize any f(x,y,z)=0 Function input in user defined routine fnc.	Constant to parameterize any f(x,y,z)=0 Function input in user defined routine fnc.	Constant to parameterize any f(x,y,z)=0 Function input in user defined routine fnc.
SPLINE/SS	General surface (solid) boundary description, Rotated.	Constant to parameterize any f(x,y,z)=0 Function input in user defined routine fnc.	Constant to parameterize any f(x,y,z)=0 Function input in user defined routine fnc.	Constant to parameterize any f(x,y,z)=0 Function input in user defined routine fnc.	Constant to parameterize any f(x,y,z)=0 Function input in user defined routine fnc.
DXDISTNG/SS* DYDISTNG/SS* DZDISTNG/SS*	Distinguishing condition (penalty on X, Y, or Z mesh equation) (T - Tmp)=0	Tmp			
DISTNG/SS*	Distinguishing condition - Rotated (T - Tmp)=0	Tmp			

Table 4.1 Boundary Conditions with Floating Point Constants

bc_name/ bc_type	brief description	floating_pt_ const1	floating_pt_ const2	floating_pt_ const3	floating_pt_ const4
U/NS V/NS W/NS	Constant velocity in X, Y, or Z-direction - Dirichlet	Value of velocity component	If present and not -1.0, condition is applied as residual equation	N/A	N/A
UVARY/SS VVARY/SS WVARY/SS	Set variation in velocity in X, Y, or Z-direction through user subroutine in "user_bc.c"	floating point input number to help parameterize the profile	floating point input number to help parameterize the profile	N/A	N/A
VELO_NORMAL / SS[b]	Set constant velocity component in direction normal to surface	Value of normal velocity component (N.B. normal vector always faces out of element)	N/A	N/A	N/A
VELO_SLIP /SS[c]	Allow slip in tangential velocity proportional to shear stress	Slip Coefficient, β	x-component of surface tangent vector	y-component of surface tangent vector	z-component of surface tangent vector
KINEMATIC/SS	Distinguishing condition - kinematic ($n_\bullet v = 0$) on mesh equations.	mass-loss or mass-gain velocity at the free boundary	N/A	N/A	N/A
CAPILLARY/ SS	Adds capillary stress at free surface to the momentum equation	surface tension	applied pressure	coefficient on the surface repulsion term	N/A
CA/NS[d]	Applies contact angle condition to pointbc node set	angle subtended by wall normal and free surface normal	x-component of solid surface normal vector	y-component of solid surface normal vector	z-component of solid surface normal vector
SURFTANG/NS[e]	Adds surface tangent stresses to momentum equation at the endpoint of a free surface	x-component of surface tangent vector at end point	y-component of surface tangent vector at end point	z-component of surface tangent vector at end point	equilibrium surface tension

Table 4.1 Boundary Conditions with Floating Point Constants

bc_name/ bc_type	brief description	floating_pt_ const1	floating_pt_ const2	floating_pt_ const3	floating_pt_ const4
SURFTANG_SCALAR/NS[f]	Same as SURFTANG, but does not require specifying normal direction	equilibrium surface tension	N/A	N/A	N/A
SLOPE/SS	Applies slope at boundary equal to vector: $n \bullet n_{vect} = 0$	x-component of surface tangent vector	y-component of surface tangent vector	z-component of surface tangent vector	N/A
FORCE/SS	Applies force on mesh at boundary equal to vector	x-component of force	y-component of force	z-component of force	N/A
NORM_FORCE/SS	Applies force on mesh at boundary equal to vector	normal-component of force	tangential-component of force	2nd tangential-component of force (in 3-D)	N/A

a.The DX/SS option will be discontinued, eventually. The DX/NS combination will be retained because it offers one way of pinning a node to a specific location. Ditto for DY/NS and DZ/NS.
b.These conditions cannot currently be used on connecting surfaces.
c.These conditions cannot currently be used on connecting surfaces.
d.The contact angle should be input in radians. The corresponding condition which is solved is $\underset{\sim}{n} \cdot \underset{\sim}{n}_w = \cos\theta$.
e.This condition need only be applied at he intersection of outflow or inflow surfaces and the free surface. The sign on the tangent vector depends on whether the computed tangent vector is facing inward or outward. This can be figured by $\underset{\sim}{t} = \underset{\sim}{n} \times \underset{\sim}{k}$.
f.This condition need only be applied at he intersection of outflow or inflow surfaces and the free surface. The sign on the tangent vector depends on whether the computed tangent vector is facing inward or outward. This can be figured by $\underset{\sim}{t} = \underset{\sim}{n} \times \underset{\sim}{k}$.

The cards SPLINE, SPLINEX, SPLINEY, SPLINEZ, UVARY, VVARY, and WVARY require user defined subroutines. Templates for these routines are currently located in the routine "`user_bc.c`". For SPLINE-type cards, both a function routine, `fnc`, for function evaluation and a corresponding routines `dfncd1`, `dfncd2`, and `dfncd3` for the derivative of the function with respect to global coordinates, are required. Likewise for, UVARY, etc. but the routines are named `velo_vary_fnc` and `dvelo_vary_fnc_d1`, etc.

4.5.3 BC (boundary conditions requiring 2 integer and 2 floating pts)

INPUT: = ***bc_name, bc_type bc_id, integer_const1, floating_pt_const1, floating_pt_const2...***

These cards are used to provide boundary condition information. The current allowable values for *bc_name, bc_type, bc_id, integer_const1*, and *floating_pt_const1*, etc. are given here in tabular form. In Table 4.2 below **NS** indicates a node set bc_type and **SS** indicates side set *bc_type. bc_id* is an integer constant identifying the boundary condition (set in EXODUS II). The following table indicates the allowable values of the character string *bc_name* and corresponding meaning of the floating point constants.

Table 4.2 Boundary Conditions with Integer and Floating Point Constants

bc_name/ bc_type	brief description	floating_pt_ const1	floating_pt_ const2	Integer_ const1	Integer_ const2
Y/ NS	Set constant concentration - Dirichlet	Value of concentration	If present and not -1.0, condition is applied as residual equation	species number of concentration	N/A
YFLUX/SS	Set mass flux to transfer coefficient times driving force	Value of mass transfer coefficient	Driving force concentration in external phase	species number of concentration	N/A[a] Used by GOMA as pointer to other YFLUX conditions on this boundary
YFLUX_CONST/ SS	Set mass flux to a constant value.	Value of mass flux	N/A	species number of concentration	N/A
POROUS_FLUX/ SS	Set mass flux to transfer coefficient times driving force for porous media	Value of mass transfer coefficient	Driving force concentration in external phase	species number of tranported species	N/A[b] Used by GOMA as pointer to other POROUS_FLUX conditions on this boundary
KIN_LEAK/SS	Distinguishing condition - kinematic with mass transfer on mesh equations	Mass transfer coefficient for bulk fluid (species n+1)	Driving force concentration in external phase	N/A	N/A[c]Used by GOMA as pointer to YFLUX conditions
VNORM_LEAK/ SS	Normal velocity boundary condition with mass transfer on momentum equations	Mass transfer coefficient for bulk fluid (species n+1)	Driving force concentration in external phase	N/A	N/A[b] Used by GOMA as pointer to YFLUX conditions

a.For KIN_LEAK and V_NORM_LEAK, the information from YFLUX boundary conditions corresponding to each species is needed. Goma automatically searches for these boundary conditions and uses *Integer_const2* to record the boundary condition number of the next YFLUX condition in a linked list (when *Integer_const2*=-1 there are no more YFLUX conditions at this boundary).

b.For KIN_LEAK and V_NORM_LEAK, the information from YFLUX boundary conditions corresponding to each species is needed. Goma automatically searches for these boundary conditions and uses *Integer_const2* to record the boundary condition number of the next YFLUX condition in a linked list (when *Integer_const2*=-1 there are no more YFLUX conditions at this boundary).
c.Goma uses the second integer constant in KIN_LEAK and VNORM_LEAK to point to boundary condition number of a YFLUX condition at this boundary (see footnote a).

Table 4.3 Boundary Conditions with Integer and 3 Floating Point Constants

bc_name/ bc_type	brief description	Integer_ const1	floating_pt _const1	floating_pt _const2	floating_pt _const3
VELO_TANGENT /SS[a]	Set constant velocity component in direction tangent to surface	NS for dynamic contact line location	Value of tangent velocity component (N.B. tangent vector defined as $\mathbf{n}\times\mathbf{k}$)	β, coefficient for slip velocity	α, scaling for the position dependent slip

a.This card can simply set a velocity in the tangent direction, or be used to apply slip at solid boundaries. The slip condition is of the following form: $t \bullet (v - \beta \dot{x} e^{-\alpha d}) = 0$ where d is defined as the distance between the current position and the dynamic contact line.

4.5.4 BC (boundary conditions requiring variable name and const)

INPUT: = ***bc_name,bc_type, bc_id, variable_name, integer_const2***

This card is used to fix the value of a variable at the value read in from the input file. The current allowable values for *bc_name*, *bc_type*, *bc_id*, and *floating_pt_const1*, etc. are given here in tabular form. In Table 4.4 below **NS** indicates a node set *bc_type*. *bc_id* is an integer constant identifying the boundary condition (set in EXODUS II). *variable_name* is a string describing the variable which should be fixed, which can be VELOCITY1, VELOCITY2, VELOCITY3, MESH_DISPLACEMENT1, MESH_DISPLACEMENT2, MESH_DISPLACEMENT3, MASS_FRACTION, TEMPERATURE, or PRESSURE (pressure will have no effect unless Q1 or Q2 basis functions are being used). The following table indicates the allowable values of the character string *bc_name* and corresponding meaning of the integer constants.

Table 4.4 Boundary Conditions with Integer Constants

bc_name/ bc_type	brief description	variable_ name	Integer_ const2
FIX/ NS	Set value of variable at the value in the input file (either contin.dat of exodus II)	Name of variable to be fixed	species number of concentration or zero if variable is not concentration

The following is an example of use of the FIX boundary condition card:

```
BC = FIX  NS  4   MESH_DISPLACEMENT1  0
BC = FIX  NS  4   MESH_DISPLACEMENT2  0
```

In this example, several continuation steps were taken to deform part of an elastic block of material, then the displacements on boundary 4 were maintained while moving another boundary (because the current displacements were not known, FIX was a convenient tool).

4.5.5 BC (generalized boundary conditions)

INPUT: = ***bc_name, bc_type, bc_id, equation_name, integer_const2, variable_name, integer_const4, floating_pt_const1, floating_pt_const2...***

These cards are used to provide all boundary condition information for a generalized dirichlet boundary condition. The generalized dirichlet condition is applied as a pointwise collocation along a given node set; this boundary condition can be applied multiple times to the same side set and equation to build up a general multiparameter condition. The current allowable values for *bc_name*, *bc_type*, *bc_id*, and *floating_pt_const1*, etc. are given here in tabular form. In Table 4.4 below **SS** indicates side set *bc_type*. *bc_id* is an integer constant identifying the boundary condition (set in EXODUS II). *equation_name* is a string describing the equation to which this boundary condition is applied, which can be R_MOMENTUM1, R_MOMENTUM2, R_MOMENTUM3, R_MESH1, R_MESH2, R_MESH3, R_MASS, R_ENERGY, R_CONTINUITY (continuity will have no effect if not using Q1 or Q2 basis functions), R_MESH_NORMAL (rotate mesh equations and apply this condition to normal component), R_MESH_TANG1, R_MESH_TANG2, R_MOM_NORMAL (rotate momentum equations and apply this condition to normal component), R_MOM_TANG1, R_MOM_TANG2. *variable_name* is a string describing the variable which should be fixed, which can be VELOCITY1, VELOCITY2, VELOCITY3, MESH_DISPLACEMENT1, MESH_DISPLACEMENT2, MESH_DISPLACEMENT3, MESH_POSITION1, MESH_POSITION2, MESH_POSITION3, MASS_FRACTION, TEMPERATURE, or PRESSURE (pressure will have no effect if not using Q1 or Q2 basis functions). *integer_const2* and *integer_const4* are the species number of the mass transport equation and concentration variable, respectively, or are zero for other equation and variable types. Currently these conditions assume that the variable is defined at all the nodes at which the equation is defined (no sub-parametric mapping). The following table indicates the allowable values of

the character string *bc_name* and corresponding meaning of the floating point constants.

Table 4.5 Boundary Conditions with String, Integer and Floating Point Constants

bc_name/ bc_type	brief description	floating_pt_ const1	floating_pt_ const2	floating_pt_ const3
GD_CONST/SS	Impose constant value for variable: $x - C1 = 0$	Value of variable, C1	N/A	N/A
GD_LINEAR/SS	Impose linear dependence on variable: $C1 + C2x = 0$	Intercept, C1	Slope, C2	N/A
GD_PARAB/SS	Impose parabolic dependence on variable: $C1 + C2x + C3x^2 = 0$	Intercept, C1	Slope, C2	Acceleration, C3
GD_CIRC/SS	Impose parabolic dependence on variable (convenient for circles): $C1 + C3(x - C2)^2 = 0$	Radius, C1 Should only appear in one GD_CIRC condition on each boundary	Origin, C2	Ellipticity, C3

The following are several examples of uses of the generalized dirichlet conditions:

- For a circular boundary (with radius 1, center at (0,0) : $x^2 + y^2 = 1$):

```
BC = GD_PARAB  SS 1  R_MESH2 0  MESH_POSITION2  0  -1. 0. 1.
BC = GD_PARAB  SS 1  R_MESH2 0  MESH_POSITION1  0  -0. 0. 1.
```

- For a planar boundary ($2x + y = 1$)

```
BC = GD_LINEAR  SS  1    R_MESH1  0  MESH_POSITION1  0 -1. 2.
BC = GD_LINEAR  SS  1    R_MESH1  0  MESH_POSITION2  0  0. 1.
```

- For a parabolic inflow velocity profile ($v = 1 - 2y - 3y^2$):

```
BC = GD_LINEAR SS   4  R_MOMENTUM1 0 VELOCITY1        0 0. -1.
BC = GD_PARAB SS    4  R_MOMENTUM1 0 MESH_POSITION2   0 1. -2. -3.
```

- For a distinguishing condition where the mesh is an iso-concentration surface ($C = 0.2$ with mesh equations rotated):

```
BC = GD_CONST SS 2 R_MESH_NORMAL 0 MASS_FRACTION 0 0.2
```

Note that in the first three examples, two cards are combined to make one boundary condition that is a function of two variables. Thus, with a little creativity, the generalized dirichlet condi-

tions can replace many of the other boundary condition types.

4.5.6 BC (fluid-solid boundaries)

INPUT: = ***bc_name, bc_type, bc_id, integer_const1, integer_const2***

These cards invoke special boundary conditions which apply across the interface between a liquid phase and a solid phase.

The FLUID_SOLID condition equates the normal traction in adjacent fluid and solid materials. This is only to be used if fluid mechanics is being solved for in one phase and solid mechanics in an adjacent phase (e.g. flexible blade coating). With this boundary condition, the local residual and jacobian contributions from the solid-mechanics momentum equations (on the solid side of the boundary) are added into the residual and jacobian entries for the fluid mechanics equations (on the fluid side of the boundary). This works because the normal tractions on both sides of the interface should be equal (and the weak form of the momentum equations is being used). *integer_const1* and *integer_const2* refer to the material block identification numbers from the EXODUS II database. *integer_const1* is the block ID for the solid phase and *integer_const2* is the block ID for the fluid phase. The following line is an example of the syntax of this boundary condition

- `BC = FLUID_SOLID  SS  5  2  1`

In this example, side set 5 is a boundary between a rubber blade and a liquid; material 2 is the rubber blade and material 1 is the fluid.

Table 4.6 Boundary Conditions with two integer constants

bc_name/ bc_type	brief description	Integer_ const1	Integer_ const2
FLUID_SOLID/ SS	Equate solid and fluid tractions across this boundary - add to fluid momentum equations	Material Number (block ID) of Solid phase	Material number (block ID) of Liquid phase
SOLID_FLUID	Equate solid and fluid tractions across this boundary - add to solid momentum equations	Material Number (block ID) of Solid phase	Material number (block ID) of Liquid phase

Table 4.6 Boundary Conditions with two integer constants

bc_name/ bc_type	brief description	Integer_ const1	Integer_ const2
NO_SLIP	Impose no-slip between a Lagrangian Solid and a Viscous Fluid	Material Number (block ID) of Solid phase	Material number (block ID) of Liquid phase

The SOLID_FLUID condition performs the same task as the FLUID_SOLID condition except that the results are applied to the solid-mechanics equations. This requires adding the local residual and jacobian entries from the fluid momentum equations to the local residual and jacobian entries from the solid momentum equations (on the solid side) while decrementing the local residual and jacobian entries from the pseudo-solid momentum equations (on the fluid side).

The NO_SLIP condition requires the velocity of the liquid at the interface to be equal to the velocity in the solid at the interface. The solid phase must be treated as a Lagrangian solid and may be in a convected frame of reference - then the fluid velocity is equal to the velocity of the stress free state mapped into the deformed state (for steady-state problems).

4.5.7 END␣OF␣BC

This card specifies the end of the list of boundary conditions, and is only used when automatic boundary condition counting is used (see 4.5.1). If Number␣of␣BC is set to -1 all BC cards below END␣OF␣BC card are ignored, and GOMA counts the number of BC cards between Number␣of␣BC and END␣OF␣BC.

4.5.8 PRESSURE␣DATUM

INPUT: = ***integer_const, floating_pt***

This card is used to set a pressure datum on fluid mechanics problems which contain no implicit or explicit boundary conditions on stress or pressure. *integer_const* is the element number on which the datum is set and *floating_pt* is the value of the datum. The element number should correspond to that shown when viewing the mesh less one, as the numbering convention in the C language starts at zero. Noteworthy is that this card is optional and if used is placed outside the BC section and just below it.

4.6 Problem Description

4.6.1 Number␣of␣Materials

INPUT: = ***integer_const***

integer_const: Value indicates the number of materials to follow. Each material will require the cards described here, i.e., each material will have its own problem description (Cards 4.6.2 through 4.6.9). If there are more MAT cards (i.e. material sections) than specified by *integer_const*, GOMA ignores the extras (i.e. only the first Number□of□Materials material sections are read). If this *integer_const* is set to -1, GOMA will automatically count the number of MAT cards between the Number□of□Materials card and the END□OF□MAT card (see 4.6.10).

4.6.2 MAT

INPUT: = ***character_string, integer_const1***

character_string corresponds to the prefix of the material file with which all material properties for the current material will be read from. The material file's name is `character_string.mat`, and if the file is not present in the current working directory, all properties and source terms are set to unity and term scaling control is transferred to the equation cards (see Section 4.6.8). The value of *integer_const1* corresponds to the block identification number specified in the EXODUS II database.

4.6.3 Coordinate□System

INPUT: = ***character_string***

Used to specify formulation of the equations to be solved. The permissible values are **CARTESIAN**, for a two or three dimensional cartesian formulation, **CYLINDRICAL**, for an axisymmetric formulation, or **SWIRLING**, for a two-dimensional formulation with a swirling velocity component that is independent of azimuthal coordinate. The **SWIRLING** option is not activated yet.

4.6.4 Element□Mapping

INPUT: = ***keyword***

keyword: The permissible values are **isoparametric**, which means the standard isoparametric mapping is used for the Gaussian quadrature. Future releases will allow **subparametric**, for a mapping which is one order less than that used to represent all of the dependent variables, except the mesh location. In any case, for moving mesh problems, the interpolation of space is determined by the mesh equation interpolation specified in the equation section (see Section 4.6.7).

4.6.5 Mesh□Motion

INPUT: = ***character_string***

Used to specify the method for describing the movement of the nodes within the mesh. **ARBITRARY** triggers the implicit pseudo-solid domain-mapping technique using the constitutive equation designated in the corresponding `file.mat` (see Chapter 5); with this technique, the boundaries of the domain are controlled by distinguishing conditions coupled with the problem physics, and the interior nodes move independently of the problem physics. **LAGRANGIAN** triggers coupling the motion of nodes on the interior of the domain to the deformation of an elastic solid. If the solid is incompressible, this technique uses a pressure (lagrange multiplier) to couple the solid deformation and the local solvent concentration.

4.6.6 Number□of□bulk□species

INPUT: = ***integer_constant***

Used to specify the number of species_bulk equations to be solved. This is equal to one less than the total number of components in the solution (because solving an equation for each component would be a linear combination of the continuity equation). If this number is zero, then no species_bulk equations are solved (and should not be specified as an equations), and if this number is 1 then the fluid is a binary solution.

4.6.7 Number□of□EQ

INPUT: = ***integer_const***

integer_const: Value indications the number of evolution equations (number of equation cards) to follow, including the mesh motion equations. This number of equations corresponds only to the current material. If there are more EQ cards than specified by *integer_const*, GOMA ignores the extras (i.e. only the first Number□of□EQ equations are read). If this *integer_const* is set to -1, GOMA will automatically count the number of EQ cards between the Number□of□EQ card and the END□OF□EQ card (see 4.6.9).

4.6.8 EQ

INPUT: = ***eq_name, Galerkin_wt, variable_name, interpolation_fcn, floating_pt_const1, floating_pt_const2, ..., floating_pt_const7***

These cards are used to provide all information required for each differential equation to be solved. The current allowable values for *eq_name,* etc. are given here in tabular form. *eq_name* is the name of the equation to be solved. *Galerkin_wt* is the weighting function type for this equation. *variable_name* is the name of the variable associated with this equation. *interpolation_fcn* is the interpolation function used to represent the variable of type *variable_name. floating_pt_const1* through *floating_pt_const7* define the constant multipliers in front of each type of term in this equation. If a multiplier is zero, the section of code which evaluates the corresponding term will be skipped. These multipliers are intended to provide a means of activating or deactivating terms of an equation, and hence should be set to zero or

one. However, one can use these multipliers as a way of adjusting the scaling of individual terms. Exercise caution in using these factors as expedients for transport coefficients; the equation term multiplier for the momentum diffusion term affects both the isotropic stress term (pressure) and the deviatoric stress.

Table 4.7 Equations and Equation Term Multipliers.

Eq_Name	Galerkin _wt	Variable_ name	Interpola tion_fnc	floating _pt_ const1	floating _pt_ const2	floating _pt_ const3	floating _pt_ const4	floating _pt_ const5	floating _pt_ const6
mesh1	Q1 or Q2[a]	D1	Q1 or Q2	mass[b]	adv[c]	bndry[d]	diff[e]	source[f]	NA
mesh2	Q1 or Q2	D2	Q1 or Q2	mass	adv	bndry	diff	source	NA
mesh3	Q1 or Q2	D3	Q1 or Q2	mass	adv	bndry	diff	source	NA
momentum1	Q1 or Q2	U1	Q1 or Q2	mass	adv[g]	bndry	diff	source	porous[h]
momentum2	Q1 or Q2	U2	Q1 or Q2	mass	adv	bndry	diff	source	porous
momentum3	Q1 or Q2	U3	Q1 or Q2	mass	adv	bndry	diff	source	porous
continuity	P0 or P1	P	P0 or P1	div	source[i]	NA	NA	NA	NA
energy	Q1 or Q2	T	Q1 or Q2	mass	adv	bndry	diff	source	NA
species_bulk[j]	Q1 or Q2	Y	Q1 or Q2	mass	adv	bndry	diff	source	NA
species_surf[k]	Nothing tested	S	Q1 or Q2	NA	NA	NA	NA	NA	NA

a.Q1 - Linear weight; Q2- Quadratic.
b.Multiplier on "mass-matrix" term (d/dt).
c.Multiplier on advective term for inertia in moving solids (normally neglected).
d.Multiplier on boundary term (n.flux).
e.Multiplier on diffusion term.
f.Multiplier on source term.
g.Multiplier on advective term.
h.Multiplier on porous term (linear source).
i.The source multiplier is equal to the initial volume fraction of solvents for Lagrangian mesh motion with swelling.
j.This equation type should only be listed once regardless of the number of species (the card for "Number of bulk species" specifies the number of species_bulk equations to be solved. Differences in diffusion coefficients between species should be accounted for in materials properties section of GOMA.
k.Equation is not active.

4.6.9 END□OF□EQ

This card specifies the end of the list of equations for this material, and is only used when automatic equation counting is used (see 4.6.7). If Number□of□EQ is set to -1 all EQ cards below this card are ignored, and GOMA counts the number of EQ cards between Number□of□EQ and END□OF□EQ. Note that this card should appear in every material that is using automatic equation counting.

4.6.10 END□OF□MAT

This card specifies the end of the list of materials, and is only used when automatic material counting is used (see 4.6.1). If Number□of□Materials is set to -1 all MAT cards below this card are ignored, and GOMA counts the number of MAT cards between Number□of□Materials and END□OF□MAT.

4.7 Post Processing Specifications

This section lists the post-processing options that are accessible within GOMA. Each card below can trigger calculation of the nodal values of a given function, which is then written to the EXODUS II output file. Normally these values are smoothed before printing them to the output file. For all these cards, a keyword is the only input; if the keyword is **yes**, the post-processing variable is calculated and written to the file, if the keyword is **no**, no output is generated for that variable. All of these cards are optional and can appear in any order.

The sections below list the post-processing options and a brief description of each. For large, time-dependent runs, the output of many post-processing variables may lead to excessively large EXODUS II output files.

4.7.1 Stream□Function

If requested, the stream function is output to the EXODUS II file described in 4.1.2. It is important to construct a mesh whose elements are contiguously ordered in such a way that there are no isolated clusters as the elements are swept, i.e., element n+1 must be in contact with one of the previous n elements. This can be assured by issuing the OPtimize command to the FASTQ meshing module (cf. Blacker 1988). Otherwise, your regions must be constructed in an analogous fashion. This variable is called **STREAM** in the output EXODUS II file.

4.7.2 Streamwise□normal□stress

The stream-wise normal strain rate, $\mathbf{D}_{tt}$, is defined as $\mathbf{tt}:\mathbf{D}$, where $\mathbf{t}$ is the unit tangent vector to the streamlines, and $\mathbf{D}$ is the strain-rate tensor, $\mathbf{D} \equiv \left[\nabla\mathbf{v} + (\nabla\mathbf{v})^{T}\right]$ associated with the Navier-Stokes equations. This variable is called **SNS** in the output EXODUS II file. The streamwise

normal stress is equal to $\mathbf{D}_{tt}$ times the viscosity.

4.7.3 Mean□shear□rate

The mean shear rate is defined as $4\sqrt{II_D}$, where II_D is the second invariant of **D**, the strain-rate tensor, $\mathbf{D} \equiv [\nabla \mathbf{v} + (\nabla \mathbf{v})^T]$ associated with the Navier Stokes equations. This variable is called **SHEAR** in the output EXODUS II file.

4.7.4 Pressure□contours

The hydrodynamic pressure is normally a field variable within GOMA; however, it is usually approximated by discontinuous basis functions. This option enables interpolating and smoothing the hydrodynamic pressure to nodal values. This variable is called **PRESSURE** in the output EXODUS II file

4.7.5 First□Invariant□of□Strain

The strain tensor is associated with the deformation of the mesh. Its first invariant is its trace and represents the volume change in the small strain limit. This variable is called **IE** in the output EXODUS II file

4.7.6 Second□Invariant□of□Strain

The strain tensor is associated with the deformation of the mesh. Its second invariant indicates the level of shear strain of the mesh. This variable is called **IIE** in the output EXODUS II file.

4.7.7 Third□Invariant□of□Strain

The strain tensor is associated with the deformation of the mesh. Its third invariant indicates the volume change from the stress-free-state (IIIE = 1 indicates no volume change). This variable is called **IIIE** in the output EXODUS II file.

4.7.8 Navier□Stokes□Residuals

These nodal variables are constructed from the corresponding velocity weighting function for a Galerkin finite element formulation. They are named **RMX, RMY**, and **RMZ** in the output EXODUS II file, corresponding to each of the independent components of the fluid momentum balance.

4.7.9 Moving□Mesh□Residuals

These nodal variables are constructed from the corresponding displacement weighting functions for a Galerkin finite element formulation. They are called **RDX, RDY**, and **RDZ** in the output EXODUS II file, corresponding to each of the independent components of the solid

(pseudo or real) momentum balance.

4.7.10 Mass□Diffusion□Vectors

This variable is called **Y0dif0** (diffusion of first species in x direction), **Y0dif1** (diffusion of first species in y direction), **Y0dif2** (diffusion of first species in z direction), **Y1dif0** (diffusion of second species in x direction), **Y2dif1** (diffusion of second species in y direction), . . . in the output EXODUS II file

4.7.11 Mass□Fluxlines

This variable is called **YFLUX0, YFLUX1,** . . .(by species number) in the output EXODUS II file; it is analogous to the stream function so that contours of the flux function represent path-lines for each species in solution.

4.7.12 Energy□Conduction□Vectors

This variable is called **Tcond0** (conduction in x direction), **Tcond1** (conduction in y direction) **Tcondf2** (conduction in z direction) in the output EXODUS II file.

4.7.13 Energy□Fluxlines

The energy flux function is analogous to the stream function, such that contours of the flux function represent paths of energy flow through the domain. This variable is called **TFLUX** in the output EXODUS II file.

4.7.14 Time□Derivatives

This option enables writing the time derivative of all the field variables to a file. Currently this routine uses the values in the global vector xdot to report this data. During the first time step, all the xdot values are zero; by the second time step, these data should be realistic. These variables are labeled **XDOT0** (mesh velocity in x direction), **XDOT1** (mesh velocity in y direction), **XDOT2** (mesh velocity in z direction), **VDOT0** (fluid acceleration in x direction), **VDOT1** (fluid acceleration in y direction), **VDOT2** (fluid acceleration in z direction), **TDOT** (rate of temperature change), **Y0DOT** (rate of 1st species concentration change), **Y1D0T** (rate of second species concentration change), . . . in the EXODUS II file.

4.7.15 Mesh□Stress□Tensor

The mesh stress tensor is associated with the equations of elasticity. The stress tensor has six entries (in 3-D, symmetric) called **T11**, **T22**, **T33**, **T12**, **T13**, and **T23** in the output EXODUS II file.

4.7.16 Mesh□Strain□Tensor

The mesh strain tensor is associated with the equations of elasticity. The strain tensor has six entries (in 3-D, symmetric) called **E11, E22, E33, E12, E13,** and **E23** in the output EXODUS II file.

4.7.17 Porous Saturation

In partially saturated porous media, the saturation represents the volume fraction of the pore space that is filled with liquid. This variable is called **SAT** in the output EXODUS II file.

4.7.18 Bulk density of species in porous media

In partially saturated porous media, bulk density is the mass of a species (e.g. water or air) per unit volume of the porous medium. This is a quantity that can be calculated from the porosity and the pressures in the liquid and gas phases. This variable is called **BULK0, BULK1, ...** in the output EXODUS II file.

4.7.19 Gas concentration of species in porous media

In partially saturated porous media, the gas phase concentration of each species is derivable from the porosity and pressures in liquid and gas phases. This variable is called **GAS0, GAS1, ...** in the output EXODUS II file.

4.7.20 Liquid concentration of species in porous media

In partially saturated porous media, the liquid phase concentration of each species is derivable from the porosity and pressures in liquid and gas phases. This variable is called **LIQ0, LIQ1, ...** in the output EXODUS II file.

4.7.21 Gas phase convection vectors in porous media

In partially saturated porous media, the convection is calculated by Darcy's Law. This variable is called **VGAS0, VGAS1, VGAS2** in the output EXODUS II file.

4.7.22 Liquid phase convection vectors in porous media

In partially saturated porous media, the convection is calculated by Darcy's Law. This variable is called **VLIQ0, VLIQ1, VLIQ2** in the output EXODUS II file.

4.7.23 Porosity in deformable porous media

This variable serves as a check on the porosity in porous media by calculating it by two independent means. This variable is called **PORED, POREM** in the output EXODUS II file.

4.7.24 Capillary pressure in porous media

In partially saturated porous media, the capillary pressure is the difference between the gas and liquid pressures. This variable is called **PC** in the output EXODUS II file.

4.7.25 Lagrangian□Convection

In deformable solids with a Lagrangian mesh, convection in the stress-free-state can be mapped to the deformed configuration; this variable stores the velocity vectors of this solid motion. This variable is called **VL1, VL2, VL3** in the output EXODUS II file.

4.7.26 User-Defined□Post□Processing

INPUT: = ***keyword, floating_point_const_list***

This option enables user-defined post-processing in GOMA. An arbitrary number of floating point constants can be loaded in to use in the user-defined subroutine (user_post.c). This variable is called **USER** in the output EXODUS II file.

5 Data Input-- Material Files

The format below indicates the data cards currently required in each material ("mat") file, followed by the options available for each card and the necessary input data. All input data are specified in a free field format with successive variables separated by blanks or commas. It is important to note that there is no limit to the number of floating point data allowed on each property or constitutive equation card. For standard models, all extraneous data on each line is ignored. For user-defined options, the user may parameterize a model with as many data as needed. Like in Chapter 4, bold/underlined type denotes a unique string which GOMA looks for in the input deck. The "□" symbol denotes a blank space. The input parser is case sensitive! The bold/italic type following the underlined, unique string denotes places where input is required.

All of the other features and restrictions associated with the problem-description file (see Chapter 4) apply. In all sections of this chapter, *model_name* is a character string and *floating_point_const_list* is a list of floating point numbers of arbitrary length separated by a comma or one or more white spaces. Figure 5 illustrates a typical material file. The remainder of this chapter covers each card (line) of the material-description file in detail.

All property models will eventually have a **USER** and a **USER_GEN** option. When the former is selected, the user must add the user model to the appropriate routine in the file `user_mp.c`. This file contains a template to simplify the implementation of a model in a full Newton context, but has the restriction that all models cannot contain a dependence on gradients of variables. For more complex models, which contain such dependencies, the user must resort to the more sophisticated mechanism that comprise the routines in `user_mp_gen.c`.

5.1 Physical Properties

5.1.1 Density

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for density. The permissible value for *model_name* is **CONSTANT**. At this time no variable density models are allowed. Boussinesq models, however, can be selected through the Navier-Stokes Source card in Section 5.6.1. The first entry in the *floating_point_const_list* is the value of the density. Entries subsequent to the first are ignored for the **CONSTANT** model of density.

```
---Physical Properties
Density            = CONSTANT 1000.

---Mechanical Properties and Constitutive Equations
Solid Constitutive Equation = LINEAR
Convective Lagrangian Velocity = NONE
Lame MU                        = CONSTANT 1.
Lame LAMBDA                    = CONSTANT 1.
Stress Free Solvent Vol Frac   = CONSTANT 0.
Liquid Constitutive Equation   = NEWTONIAN
Viscosity                      = USER 1. 1. 1. 1. 1.
Low Rate Viscosity             = CONSTANT 0.
Power Law Exponent             = CONSTANT 0.
High Rate Viscosity            = CONSTANT 0.
Time Constant                  = CONSTANT 0.
Aexp                           = CONSTANT 0.

---Thermal Properties
Conductivity              = USER 1. 1.  1. 1. 1. 1.
Heat Capacity             = CONSTANT 1.
Volume Expansion          = CONSTANT 1.
Reference Temperature     = CONSTANT 1.
Liquidus Temperature      = CONSTANT 1.
Solidus Temperature       = CONSTANT 1.

---Microstructure Properties
Media Type                = CONTINUOUS
Porosity                  = DEFORM 0.5
Permeability              = CONSTANT1.
Permeability              = CONSTANT 0.001
Rel Gas Permeability      = SUM_TO_ONE 0.0001
Rel Liq Permeability      = VAN_GENUCHTEN 0.01 0.01 0.667 0.01
Saturation                = VAN_GENUCHTEN 0.01 0.01 3.0 1.

---Species Properties
Diffusion Constitutive Equation= FICKIAN
Diffusivity                    = CONSTANT 0 1.
Latent Heat Vaporization       = CONSTANT 0 0.
Latent Heat Fusion             = CONSTANT 0 0.
Vapor Pressure                 = CONSTANT 0 0.
Species Volume Expansion       = CONSTANT 0 0.
Species Volume Expansion       = CONSTANT 0 0.
Reference Concentration        = CONSTANT 0 0.
*************Species Number*****************

----Source Terms
Navier-Stokes Source      = USER 1.  1. 1. 1. 1.
Solid Body Source         = CONSTANT 0.  0.  0.
Mass Source               = CONSTANT 0.
Heat Source               = CONSTANT 1.
Species Source            = CONSTANT 0 2.
*************Species Number************
```

Lines are repeated for each species

Line is repeated for each species

Figure 5 Sample material-description file format. Bold-face lines are required.

5.2 Mechanical Properties and Constitutive Equations

5.2.1 Solid□Constitutive□Equation

INPUT: = ***model_name***

Specifies the constitutive equation used to control mesh motion. In an arbitrary mesh, permissible values of *model_name* are **LINEAR**, for which the deformations are assumed to be small and thus simplifies the analysis of strain and stress, or **NONLINEAR** for which the deformations can be large (all non-linear models currently default to **NONLINEAR** if the mesh is arbitrary). In a Lagrangian mesh, where the nodes are material points that move with the solid, permissible values are **HOOKEAN_PSTRAIN** (non-linear neo-Hookean model with plane strain), **HOOKEAN_PSTRESS** (non-linear neo-Hookean model with plane stress), **INCOMP_PSTRAIN** (incompressible non-linear neo-Hookean model with plane strain and a Lagrangian pressure constraint on the volume), and **INCOMP_PSTRESS** (incompressible non-linear neo-Hookean model with plane stress and a Lagrangian pressure constraint on the volume) The incompressible options use the theory of Segalman et al. (1992) to control mesh motion and couple the volume dilation to changes in solvent content. Plane strain implies that there is no deformation in the z-direction, but plane stress implies there is no stress in the z-direction.

5.2.2 Convective□Lagrangian□Velocity

INPUT: = ***model_name, floating_point_const_list***

In solid mechanics, when the deformation of the mesh is Lagrangian, i.e., motion of the solid can be described by a mapping between motion of the stress-free-state (undeformed state) and the deformed state, it is often advantageous to prescribe a convective velocity. If the *model_name* on this card is set to **NONE**, then the stress-free-state is assumed to be unmoving, and if this *model_name* is **CONSTANT**, then the stress-free-state is moving uniformly with a velocity specified by three *floating_point_constants* (vx, vy, vz).

5.2.3 Lame□MU

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the Lame coefficient μ for the solid constitutive equation (see Sackinger et al. 1995); this coefficient is equivalent to the shear modulus. The permissible values for *model_name* are **CONSTANT, POWER_LAW, CONTACT_LINE**, or **USER**. The first entry in the *floating_point_const_list* is the standard value of the μ. Note that μ and λ (see 5.2.4 below) are related to Young's Modulus and Poisson's Ratio by the following standard expressions:

$$E = \frac{\mu(3\lambda + 2\mu)}{\lambda + \mu}; \upsilon = \frac{\lambda}{2(\lambda + \mu)}$$

where E is the Young's modulus and ν is Poisson's ratio. A significant limiting case is approached as ν approaches 0.5, in which case the solid becomes incompressible.

The **POWER_LAW** model can be used in deformable porous media where the shear modulus varies as a power law function of the porosity, ϕ (Scherer 1992):

$$G = G_0\left(\frac{1-\phi_0}{1-\phi}\right)^m$$

The three entries in the *floating_point_const_list* are G_0, ϕ_0, and m, respectively.

The **CONTACT_LINE** model is a convenient way to control mesh deformation near a fixed point. This model enables making the shear modulus much larger near the contact line (fixed point) than far away from the contact line, so that elements near the contact line are forced to retain their shape. The shear modulus in this model varies inversely with distance from the contact line:

$$G = G_0 + \frac{0.1}{\left((r/r_0)^3 + 0.1/G_1\right)}$$

r is the distance from the fixed point, r_0 is a decay length G_0 is the modulus far from the contact line, and $G_0 + G_1$ is the modulus at the contact line. The four entries in the *floating_point_const_list* are the node set number of the fixed point (converted to an integer by GOMA), G_0, G_1, and r_0.

5.2.4 Lame□LAMBDA

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the Lame coefficient λ for the solid constitutive equation (see Sackinger et al. 1995). When using a nonlinear constitutive equation for ALE mesh motion, this coefficient is equal to the bulk modulus:

$$K = \lambda + \frac{2}{3}\mu$$

The permissible values for *model_name* are **CONSTANT, POWER_LAW,** or **USER**. The first entry in the *floating_point_const_list* is the standard value of the λ.

The **POWER_LAW** model can be used in deformable porous media where the lame coefficient varies as a power law function of the porosity, ϕ (Scherer 1992):

$$\lambda = \lambda_0\left(\frac{1-\phi_0}{1-\phi}\right)^m$$

The three entries in the *floating_point_const_list* are λ_0 , ϕ_0 , and *m*, respectively.

5.2.5 Stress□Free□Solvent□Vol□Frac

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the stress-free solvent volume fraction, which is volume fraction of solvents in the solid material in its stress-free-state. The permissible value for *model_name* is **CONSTANT**. At this time no variable models are allowed. The first entry in the *floating_point_const_list* is the value of the stress free solvent volume fraction. All other entries are ignored.

5.2.6 Liquid□Constitutive□Equation

INPUT: = ***model_name***

Used to specify the stress, strain-rate/strain constitutive equation associated with the momentum equations (Navier-Stokes equations) described in section below. The permissible values of *model_name* are **NEWTONIAN**, for a simply constant viscosity, Newtonian fluid, **POWER_LAW**, for a power law model, and **CARREAU**, for a Carreau strain-rate thinning or thickening relation. All three models are described in detail in Bird et al. (1987).

The **NEWTONIAN** option requires one floating point constant, μ. μ is the viscosity in the chosen units and is entered with the "Viscosity" card (Section 5.2.7).

With the **POWER_LAW** option, two floating point constants are required. The first, μ_0, is the zero-strain rate limit of the viscosity and is entered with the "Low Rate Viscosity" card (Section 5.2.8). The second, n_{exp}, is the exponent on the strain rate which can take on any value between 1 (Newtonian) and 0 (infinitely shear thinning). n_{exp} is entered with the "Power Law Exponent" card (Section 5.2.9).

The **CARREAU** option requires five floating point constants. The first, μ_0, is the zero-strain rate limit of the viscosity and is entered with the "Low Rate Viscosity" card (Section 5.2.8). The second, n_{exp}, is the exponent on the strain rate which can take on any value between 1 (Newtonian) and 0 (infinitely shear thinning). n_{exp} is entered with the "Power Law Exponent" card (Section 5.2.9). The third, μ_{inf}, is the high-strain-rate limit to the viscosity and is entered with the "High Rate Viscosity" card (Section 5.2.10). The fourth, λ, is the time constant reflecting the strain-rate at which the transition between μ_0 and μ_{inf} takes place. λ is entered with the "Time Constant" card (Section 5.2.11). The fifth, a_{exp}, is a dimensionless parameter that describes the transition between the low-rate and the power-law region.

5.2.7 Viscosity

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the viscosity for the liquid constitutive equation (see Sackinger et al. 1995). The permissible value for *model_name* is **CONSTANT,** for a constant viscosity model, or **USER**, for a user-defined model. For the **USER** option the appropriate modifications to the routine "usr_viscosity" in the user_mp.c file must be undertaken. The first entry in the *floating_point_const_list* is the value viscosity for the **CONSTANT** option. All other entries are ignored.

5.2.8 Low□Rate□Viscosity

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the low-rate viscosity parameter for the power-law fluid or Carreau fluid options of the liquid constitutive equation (Section 5.2.6). The permissible value for *model_name* is **CONSTANT**. At this time no variable models are allowed. The first entry in the *floating_point_const_list* is the value of the low-rate viscosity. All other entries are ignored.

5.2.9 Power□Law□Exponent

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the power-law exponent parameter of the power-law fluid or Carreau fluid options of the liquid constitutive equation (Section 5.2.6). The permissible value for *model_name* is **CONSTANT**. At this time no variable models are allowed. The first entry in the *floating_point_const_list* is the value of the low-rate viscosity. All other entries are ignored.

5.2.10 High□Rate□Viscosity

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the high-rate viscosity parameter of the Carreau fluid options of the liquid constitutive equation (Section 5.2.6). The permissible value for *model_name* is **CONSTANT**. At this time no variable models are allowed. The first entry in the *floating_point_const_list* is the value of the high-rate viscosity. All other entries are ignored.

5.2.11 Time□Constant

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the time constant parameter of the Carreau fluid options of the liquid constitutive equation (Section 5.2.6). The permissible value for *model_name* is **CONSTANT**. At this time no variable models are allowed. The first entry in the

floating_point_const_list is the value of the time constant. All other entries are ignored.

5.2.12 Aexp

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the Aexp parameter of the Carreau fluid options of the liquid constitutive equation (Section 5.2.6). The permissible value for *model_name* is **CONSTANT**. At this time no variable models are allowed. The first entry in the *floating_point_const_list* is the value of Aexp. All other entries are ignored. a_{exp}, is a dimensionless parameter that describes the transition between the low-rate and the power-law region (see Bird et al. 1987).

5.2.13 Polymer Constitutive Equation

INPUT: = ***model_name***

Used to specify the polymer constitutive equation. *model_name* can be **NOPOLYMER**, for generalized Newtonian models, **OLDROYDB**, for the Oldroyd-B constitutive model, and **GIESEKUS**, for the Giesekus model. If options other than **NOPOLYMER** are chosen, additional cards for the polymer properties must be supplied.

5.2.14 Polymer Viscosity

INPUT: = ***model_name, floating_point_const***

Used to specify the stress, strain-rate/strain constitutive equation associated with the polymer constitutive equation described in 5.2.13. The permissible values of *model_name* are **CONSTANT**, for a simply constant viscosity, Newtonian fluid, **POWER_LAW**, for a power law model, and **CARREAU**, for a Carreau strain-rate thinning or thickening relation. All three models are described in detail in Bird et al. (1987). The ***floating_point_const*** is used to specify the zero-rate viscosity.

5.2.15 Polymer Time Constant

INPUT: = ***model_name, floating_point_const***

Used to specify the stress, strain-rate/strain of the polymer time constant associated with the polymer constitutive equation described in 5.2.13. The permissible values of *model_name* are **CONSTANT**, for a simply constant viscosity, Newtonian fluid, **POWER_LAW**, for a power law model, and **CARREAU**, for a Carreau strain-rate thinning or thickening relation. All three models are described in detail in Bird et al. (1987). The ***floating_point_const*** is used to specify the zero-rate time constant.

5.3 Thermal Properties

5.3.1 Conductivity

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the thermal conductivity. The permissible value for *model_name* is **CONSTANT,** for a constant viscosity model, or **USER**, for a user-defined model. For the **USER** option the appropriate modifications to the routine "usr_thermal_conductivity" in the user_mp.c file must be undertaken. The first entry in the *floating_point_const_list* is the value conductivity for the **CONSTANT** option. All other entries are ignored. For the **USER** option, *floating_point_const_list* can be of arbitrary length and used to parameterize the model. These parameters are made available in the subroutine.

5.3.2 Heat□Capacity

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the heat capacity. The permissible value for *model_name* is **CONSTANT,** for a constant heat capacity model, or **USER**, for a user-defined model. For the **USER** option the appropriate modifications to the routine "usr_heat_capacity" in the user_mp.c file must be undertaken. The first entry in the *floating_point_const_list* is the value conductivity for the **CONSTANT** option. All other entries are ignored. For the **USER** option, *floating_point_const_list* can be of arbitrary length and used to parameterize the model. These parameters are made available in the subroutine.

5.3.3 Volume□Expansion

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the coefficient of volume expansion in the energy equation. This property is required for the **BOUSS** option on the "Navier-Stokes Source" card (Section 5.6.1). The permissible value for *model_name* is **CONSTANT,** for a constant volume expansion coefficient model. The first entry in the *floating_point_const_list* is the value of the coefficient for the **CONSTANT** option. All other entries are ignored.

5.3.4 Reference□Temperature

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the reference temperature, which is required by the **BOUSS** option on the "Navier-Stokes Source" card (Section 5.6.1). The permissible value for *model_name* is **CONSTANT**, for a constant reference temperature model. The first entry in the *floating_point_const_list* is the value of the reference temperature for the **CONSTANT** option. All other entries are ignored.

5.3.5 Liquidus□Temperature

INPUT: = ***model_name, floating_point_const_list***

NOT USED YET. Used to specify the model for the liquidus temperature. The permissible value for *model_name* is **CONSTANT**. The first entry in the *floating_point_const_list* is the value of the liquidus for the **CONSTANT** option. All other entries are ignored.

5.3.6 Solidus□Temperature

INPUT: = ***model_name, floating_point_const_list***

NOT USED YET. Used to specify the model for the solidus temperature. The permissible value for *model_name* is **CONSTANT**. The first entry in the *floating_point_const_list* is the value of the solidus for the **CONSTANT** option. All other entries are ignored.

5.4 Microstructure Properties

5.4.1 Media□Type

INPUT: = ***model_name***

This card is used to designate the characteristic type of medium of this material. The current choices are **CONTINUOUS**, **POROUS_SATURATED**, and **POROUS_PART_SAT**. None of these choices require constants. If the type chosen is **CONTINUOUS**, then the material is assumed to amorphous and no further cards are necessary in this section (next required card is 'Diffusion Constitutive Equation', 5.5.1). In a porous medium with one phase in the pores (i.e. a saturated medium), use **POROUS_SATURATED** and only the Permeability (5.4.3) and Porosity (5.4.2) cards are required. In a porous medium with two phases in the pores (i.e. an unsaturated medium, e.g. air-water), use **POROUS_PART_SAT** or **POROUS_UNSATURATED** and all the cards in this section are required.

5.4.2 Porosity

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the porosity, which is required for the Brinkman or Darcy formulations for flow through porous media. The permissible values for *model_name* are **CONSTANT,** for a constant porosity model and **DEFORM** for a porosity which varies with deformation of the porous medium (and in which porosity is a variable associated with the continuity equation, P). The first entry in the *floating_point_const_list* is the value of the porosity; in the **DEFORM** model, this porosity is the porosity of the stress-free-state.

5.4.3 Permeability

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the permeability, which is required for the Brinkman or Darcy formulations for flow through porous media. The permissible values for *model_name* are **CONSTANT**, for a constant permeability model, and **PSD_VOL**, for a deformable medium with a distribution of pore-sizes. The first entry in the *floating_point_const_list* is the value of the permeability for the **CONSTANT** option.

The **PSD_VOL** model treats the medium as a bundle of capillary tubes with a distribution of pores such that over a range of pore-sizes the volume of pores is evenly distributed. For such a model, the maximum pore-size varies with the porosity:

$$r_{max}(\phi) = r_{max}(\phi_0)\left(\frac{\phi}{\phi_0}\right)^{1/2}\left(\frac{1-\phi_0}{1-\phi}\right)^{1/3}$$

Then, the permeability is a function of the maximum pore-size and the pore-size distribution:

$$k = \phi\frac{1}{60\tau^2}\frac{\left(1-\alpha^3\right)}{(1-\alpha)}r_{mx}^2$$

The entries in the *floating_point_const_list* are ϕ_0, $r_{max}(\phi_0)$, α, and $1/\tau^2$.

5.4.4 Capillary□Network□Stress

INPUT: = ***model_name***

This card specifies the mechanism by which capillary stress in the liquid phase of a partially-saturated porous medium is transferred to the solid network. **WETTING** specifies that the porous skeleton has the same hydrostatic pressure as the liquid and **PARTIALLY_WETTING** specifies that the porous skeleton has a hydrostatic pressure that is the average of the liquid and gas phase pressures, weighted by their saturations (See related report on drying of deformable porous media).

5.4.5 Rel□Gas□Permeability

INPUT: = ***model_name, floating_point_const_list***

This card specifies the model for the relative gas phase permeability for flow in a partially saturated porous media, such that the gas flow is the pressure gradient in the gas times the permeability times the relative gas phase permeability divided by the gas viscosity. Currently the permissible values of *model_name* are **CONSTANT** and **SUM_TO_ONE**. The **SUM_TO_ONE** model assumes that the relative liquid permeability and relative gas permeability add to one, and the only parameter is the gas phase viscosity.

5.4.6 Rel□Liq□Permeability

INPUT: = ***model_name, floating_point_const_list***

5.5.6 Reference□Concentration

INPUT: = ***model_name, species_number, floating_point_const_list***

Used to specify the model for the reference concentration, which is required by the **BOUSS** option on the "Navier-Stokes Source" card (Section 5.6.1). *species_number* is an integer designating the species equation. The permissible value for *model_name* is **CONSTANT,** for a constant reference concentration model. The first entry in the *floating_point_const_list* is the value of the reference concentration for the **CONSTANT** option. All other entries are ignored.

5.6 Source Terms

5.6.1 Navier-Stokes□Source

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the source term on the Navier-Stokes equations. The permissible values for *model_name* are **CONSTANT,** for a constant source model, **USER**, for a user-defined model, **BOUSS**, for a generalized Boussinesq source term, or **BOUSS_JXB**, for a generalized Boussinesq source term with Lorentz forces. For the **USER** option one must make the appropriate modifications to the routine "usr_momentum_source" in the user_mp.c file. The three entries in the *floating_point_const_list* are the vector (x,y,z) components of the source, for the **CONSTANT** option. All other entries are ignored. For the **USER** option, *floating_point_const_list* can be of arbitrary length and used to parameterize the model. These parameters are made available in the subroutine. For the **BOUSS** option, the first three entries of the *floating_point_const_list* are the components of the gravity vector. For the **BOUSS_JXB** option, one must supply the current density field vector components and the magnetic field components with the "External Field" cards (see Section 4.2.6). The cards must point to specific names in the EXODUS II files from which the fields are read. The J-field components must be called JX_REAL, JY_REAL, and JZ_REAL. Likewise the B-field components must be called BX_REAL, BY_REAL, and BZ_REAL. These names are the default names coming from TORO II (Gartling 1995). Because of the different coordinate convention when using cylindrical components, the fields have been made compatible with those arising from TORO II. For the **BOUSS_JXB** option, the first three entries of the *floating_point_const_list* are the components of the gravity vector, and the fourth component is a scaling parameter for the JXB term. WARNING: Make sure the equation term multipliers for the source terms are set to unity (see Section 4.6.8).

5.6.2 Solid□Body□Force

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the source term on the equations for solid mechanics. The per-

missible values for model_name are CONSTANT, for a constant force model. The three entries in the floating_point_const_list are the vector (x,y,z) components of the force, for the CONSTANT option. All other entries are ignored. WARNING: Make sure the equation term multipliers for the source terms are set to unity (see Section 4.6.8).

5.6.3 Heat□Source

INPUT: = ***model_name, floating_point_const_list***

Used to specify the model for the source term on the energy equation. The permissible values for *model_name* are **CONSTANT,** for a constant source model, **JOULE**, for a Joule heating source term, **USER**, for a user-defined model, or **USER_GEN**, for a user-defined model with low-level, general capabilities. The **JOULE** model uses the species field which is being employed as a potential field for an electrostatics problem. It computes the heat source as: $\frac{1}{\sigma} J \cdot J = \nabla\phi \cdot \sigma\nabla\phi$ where J is the current flux density which is represented as $-\sigma\nabla\phi$ and ϕ is represented as a concentration species. The only input required for this option is an integer species number. For the **USER** option one must make the appropriate modifications to the routine "usr_heat_source" in the user_mp.c file. For the **USER_GEN** option one must make the appropriate modifications to the routine "usr_heat_source_gen" routine in the `user_mp_gen.c` file. The difference between the **USER** and **USER_GEN** capabilities is described at the beginning of this chapter. The first entry in the *floating_point_const_list* value of the heat source, for the **CONSTANT** option. For the **USER** option, *floating_point_const_list* can be of arbitrary length and used to parameterize the model. These parameters are made available in the subroutine. All other entries are ignored. WARNING: Make sure the equation term multipliers for the source terms are set to unity (see Section 4.6.8).

5.6.4 Species□Source

INPUT: = ***model_name, species_no, floating_point_const_list***

Used to specify the model for the source term on the species convection diffusion equations. The permissible values for model_name are CONSTANT, for a constant force model. The first entry of the floating_point_const_list is the value of the source, for the CONSTANT option. All other entries are ignored. species_number is an integer designating the species equation. WARNING: Make sure the equation term multipliers for the source terms are set to unity (see Section 4.6.8).

6 Theory - a Description of the Equations

GOMA has been constructed to solve a variety of types of single-phase or multi-phase problems and has found many applications. However, all the applications of GOMA are based on a subset of the same set of physical principles chosen from the overall capability of GOMA. The bulk equations solved by GOMA are all conservation equations (also called equations of change, etc.) which govern conservation of total mass, component mass, momentum, energy, and mesh pseudo-momentum. For some applications, the forms of these equations and how they are implemented can vary. This Chapter lists the standard forms of the equations as they are implemented in GOMA and describes how to apply these equations using GOMA. More detailed descriptions of the equations and their origins can be found in other documents (e.g. Chen et al. 1995, Sackinger et al. 1995, Gartling et al. 1995).

6.1 General Form of Equations

The different equations of change all have similar features in how they are derived and how they are applied. Because they are derived from balance equations, the resulting terms in the differential equation all fit into the general form of

accumulation = (flux in - flux out) + generation.

Also, the *(flux in - flux out)* term is normally split into an advective (or convective) part and a diffusive part. The general forms of the five conservation equations are shown in Table 6.1. The accumulation term is used only in transient problems and represents the rate of increase of the conserved quantity; this is called a *MASS* term. The generation term represents the rate at which the conserved quantity is produced per unit volume; this is called a *SOURCE* term. The equations

Table 6.1 General Forms of the Conservation Equations in GOMA

Quantity Conserved	MASS term	ADVECTION term	DIFFUSION term	SOURCE term
Momentum in Fluid Mechanics	$\rho\frac{dv}{dt}$	$= -\rho(v - v_m) \bullet \nabla v$	$+\nabla\bullet T$	$+g$
Energy	$\frac{d(\rho C_p T)}{dt}$	$= -(v - v_m) \bullet \nabla(\rho C_p T)$	$-\nabla\bullet q$	$+H$
Total Mass	0	$= \nabla\bullet v$		
Component Mass	$\frac{dC_i}{dt}$	$= -(v - v_m) \bullet \nabla C_i$	$-\nabla\bullet J_i$	$+R_i$
Momentum in Solid or Pseudo-Solid	0	=	$\nabla\bullet T_s$	$+g_s$

listed in Table 6.1 are the standard forms of the equations; for certain applications listed below in the sections on each equation, the exact form may vary (Malvern 1969, Bird et al. 1960).

The time derivatives in the *MASS* terms of the equations are *total time derivatives* (Bird et al. 1960), which follow the motion of reference points (motion of the mesh in the numerical formulation). The mesh velocity is $\boldsymbol{v}_m$ and is zero in steady-state problems. In transient problems, the mesh velocity is equal to the time derivative of the nodal positions:

$$\boldsymbol{v}_m = \frac{d\boldsymbol{d}}{dt} \tag{6.1}$$

$\boldsymbol{d}$ is the displacement vector of the mesh. The definitions of the variables in the equations in Table 6.1 are listed below in each section on the specific equations.

6.1.1 Application of FEM

GOMA uses the Galerkin method of weighted residuals with finite element basis functions to solve the system of equations presented above. First, the equations are converted to residual equations by moving the *MASS* term to the right-hand-side. The vector equations (the momentum equations for fluid and solid) are converted to scalar equations by a dot product with the basis vectors, $\boldsymbol{e}_\alpha$. Then the residual equations are multiplied by a weighting function, φ_i, and integrated over the whole domain to get the weighted residual equations. For example, for the α component of the fluid momentum equation (a vector equation), we get:

$$R_i^{f,\alpha} = \int_D \varphi_i \boldsymbol{e}_\alpha \bullet \left[-\rho \frac{d\boldsymbol{v}}{dt} - \rho\,(\boldsymbol{v} - \boldsymbol{v}_m) \bullet \nabla \boldsymbol{v} + \nabla \bullet \boldsymbol{T} + \boldsymbol{g} \right] dV = 0 \tag{6.2}$$

In this equation dV accounts for the integral over the volume of the entire domain, and includes the necessary weightings for different coordinate systems. For the energy equation (a scalar equation) we get:

$$R_i^e = \int_D \varphi_i \left[-\frac{d(\rho C_p T)}{dt} - (\boldsymbol{v} - \boldsymbol{v}_m) \bullet \nabla (\rho C_p T) - \nabla \bullet \boldsymbol{q} + H \right] dV = 0 \tag{6.3}$$

The weighted residuals for the other equations can be derived by analogy. Then the *DIFFUSION* term of these weighted residuals are integrated by parts to reduce the order; e.g. for the fluid momentum equations we have:

$$R_i^{f,\alpha} = -\int_D \varphi_i \boldsymbol{e}_\alpha \bullet \rho \frac{d\boldsymbol{v}}{dt} dV \qquad \textit{MASS} \tag{6.4}$$

$$-\int_D \varphi_i \boldsymbol{e}_\alpha \bullet [\rho\,(\boldsymbol{v} - \boldsymbol{v}_m) \bullet \nabla \boldsymbol{v}]\, dV \qquad \textit{ADVECTION}$$

$$+\oint_{\partial D}\varphi_i \boldsymbol{n}\bullet\boldsymbol{e}_\alpha\bullet\boldsymbol{T}dS \qquad \textit{BOUNDARY}$$

$$-\int_D \nabla(\varphi_i\boldsymbol{e}_\alpha):\boldsymbol{T}dV \qquad \textit{DIFFUSION}$$

$$+\int_D \varphi_i\boldsymbol{e}_\alpha\bullet\boldsymbol{g}dV \qquad \textit{SOURCE}$$

This form of the weighted residual equations is often called the *weak* form. The second to last term is the new form of the *DIFFUSION* term and : denotes a double-dot product between two tensors. The third term is a surface integral of the normal traction (flux of momentum normal to the surface) or normal energy flux, and is a new term in the equation that is called a *BOUNDARY* term. Usually, the normal traction in the surface integral is replaced by a relationship specifying the normal traction in terms of known quantities (e.g. the capillary condition states that the normal traction is equal to the external pressure plus the capillary pressure jump due to curvature and surface tension). The weak form of the weighted residuals for the energy equation is:

$$R_i^e = -\int_D \varphi_i\frac{d(\rho C_pT)}{dt}dV \qquad \textit{MASS} \qquad (6.5)$$

$$-\int_D \varphi_i\left[(\boldsymbol{v}-\boldsymbol{v}_m)\bullet\nabla(\rho C_pT)\right]dV \qquad \textit{ADVECTION}$$

$$-\oint_{\partial D}\varphi_i\boldsymbol{n}\bullet\boldsymbol{q}dS \qquad \textit{BOUNDARY}$$

$$+\int_D \nabla\varphi_i\bullet\boldsymbol{q}dV \qquad \textit{DIFFUSION}$$

$$+\int_D \varphi_i H dV \qquad \textit{SOURCE}$$

The *BOUNDARY* term is calculated only around the boundary of the domain (actually it can be calculated as an integral around each finite element, but all the internal integrals from adjacent elements would cancel), and not calculating this integral is equivalent to applying a stress-free or no-flux boundary condition. Such conditions on the normal flux only affect the *BOUNDARY* term on the weighted residual equations.

6.1.2 Equation Term Multipliers

The input deck for GOMA enables the user to turn on or turn off individual terms in the equations by means of the equation term multipliers: *MASS*, *ADVECTION*, *BOUNDARY*, *DIFFUSION*, *SOURCE*, and *POROUS*. The *MASS* term is used to control the transient accumulation part of the residual. The *ADVECTION* term is used to control the convective part of the divergence of the

flux that is *not* integrated by parts. The *BOUNDARY* term is the boundary integral resulting from integrating the diffusion term by parts. The *DIFFUSION* term is used to control the diffusive part of the divergence of the flux that is integrated by parts. The *SOURCE* term is used to control the generation term. The *POROUS* term is used to control special terms in the momentum equation that arise in Brinkman flow in a porous media.

If an equation term multiplier is set to zero, then that term in the equation is not calculated. If an equation term multiplier is set to one, then that term in the equation is calculated based on the physical properties in the material data file. If an equation term multiplier is non-zero and non-unity, then that term is calculated based on the properties in the material data file and scaled by the term multiplier. Thus, it is possible to change the weighting on individual terms in the equations by 'tuning' the equation term multipliers, but this method is **not** recommended because 'tuning' the multipliers changes the meanings of the physical parameters. For example, if a user changes the DIFFUSION equation term multiplier on the momentum equation, this weights both the viscous and pressure parts of the momentum flux (because it weights the whole term) and would not be a good way to adjust the viscosity.

6.1.3 Constitutive Equations and Physical Properties

Several of the terms in Table 6.1 are calculated from physical properties and constitutive equations before being plugged into the equations. These physical properties and constitutive equations are specified in the material data files (see Chapter 5). Thus accurate solution of problems requires consistency between the material data file, the input file, and the geometrical description of the problem (the EXODUS II file). Note that the length scales used in setting up the EXODUSII file should be the same as the length scales in the problem being solved-- they are not rescaled during solution of the problem.

6.2 Conservation of Momentum in Fluids

Conservation of momentum balances the viscous, inertial, gravitational, and pressure forces acting on a fluid (Bird et al. 1960):

$$\rho\frac{d\boldsymbol{v}}{dt} = -\rho\,(\boldsymbol{v}-\boldsymbol{v}_m)\bullet\nabla\boldsymbol{v} + \nabla\bullet\boldsymbol{T} + \boldsymbol{g} \tag{6.6}$$

$\boldsymbol{v}$ is the mass-averaged velocity of the fluid, ρ is the density of the fluid, $\boldsymbol{T}$ is the stress tensor, and $\boldsymbol{g}$ is the body force acting on the fluid.

The stress tensor in Newtonian or generalized-Newtonian fluids is proportional to the rate-of-strain tensor, $\boldsymbol{D}$, and represents the rate at which fluid elements are deformed:

$$\boldsymbol{D} = \frac{1}{2}\left(\nabla\boldsymbol{v} + \nabla\boldsymbol{v}^T\right) \tag{6.7}$$

Currently, GOMA can solve problems in Newtonian ($\boldsymbol{T} = \mu\boldsymbol{D} - p\boldsymbol{I}$), Power Law, and Carreau fluids (see Section 5.1).

The body force vector, $\boldsymbol{g}$, represents forces acting at a distance on the fluid due to gravity or other sources. A Boussinesq model is available to approximate the effect of buoyancy on a flow field (see Section 5.6).

The momentum equations have been augmented for flow in a rigid porous medium using the Brinkman equation. This is turned on using the POROUS term multiplier and setting the Brinkman-equation parameters in the materials file. This changes several terms in the momentum equations (Gartling et al. 1995):

$$\frac{\rho}{\phi}\frac{d\boldsymbol{v}}{dt} = -\frac{\rho}{\phi^2}(\boldsymbol{v}-\boldsymbol{v}_m)\bullet\nabla\boldsymbol{v} + \nabla\bullet\boldsymbol{T}_B + \boldsymbol{g} + \left(\frac{\rho\hat{c}}{\sqrt{k}}\|\boldsymbol{v}\| + \frac{\mu}{k}\right)\boldsymbol{v} \tag{6.8}$$

ϕ is the porosity of the porous medium, $\boldsymbol{T}_B$ is the stress tensor using a Brinkman viscosity, $\hat{c}$ is an inertial coefficient, and k is the permeability.

6.3 Conservation of Momentum in Solids or Pseudo-Solids

In GOMA, conservation of momentum in solids or pseudo-solids (i.e. elasticity) is used to solve for the motion of the computational mesh as material points in the solid or pseudo-solid. The mesh is treated as a computational Lagrangian solid (i.e. mesh is composed of material points) for problems in elasticity. The mesh is treated in an Arbitrary-Eulerian-Lagrangian (ALE) framework for problems in fluid mechanics, heat transfer and mass transfer. In the ALE formulation, the boundary shape is dictated by distinguishing conditions, and the interior of the mesh acts like an elastic solid subject to the boundary conditions. In ALE, the exact motion of the interior of the mesh is not important in solving the physical problem, but maintaining nicely shaped elements is important. The mesh is Lagrangian for problems in solid mechanics - i.e. the mesh moves with the solid material. It is possible to solve problems with an ALE mesh in a fluid phase and a Lagrangian mesh in an adjacent solid phase.

The formulation of the momentum equation for mesh motion in GOMA assumes that the elastic forces are much larger than inertial forces, so the mesh motion is quasi-static (Malvern 1969):

$$\nabla\bullet\boldsymbol{T}_s + \boldsymbol{g}_s = 0 \tag{6.9}$$

$\boldsymbol{T}_s$ is the stress tensor of the solid or pseudo-solid, and $\boldsymbol{g}_s$ is a body force acting on the solid (normally this is zero in the ALE formulation). ALE mesh motion is only solved in Cartesian coordinates regardless of the coordinate system used for the physical problem. Currently only Cartesian coordinates are available for Lagrangian mesh motion.

There are several linear and non-linear constitutive models in GOMA that can be used to calculate the elastic stress tensor, $\boldsymbol{T}_s$. For linear elasticity, Hooke's Law describes the relationship between

stress and strain (Malvern 1969):

$$\boldsymbol{T}_s = 2\mu\boldsymbol{E} + \lambda e\boldsymbol{I} \tag{6.10}$$

μ and λ are the Lame elastic coefficients, $\boldsymbol{E}$ is the small deformation strain tensor, and $e = tr|\boldsymbol{E}|$ is the small deformation volume strain. For small strains, the strain tensor is equal to the symmetric part of the displacement gradient,

$$\boldsymbol{E} = \frac{1}{2}\left(\nabla\boldsymbol{d} + \nabla\boldsymbol{d}^T\right) \tag{6.11}$$

The primary disadvantage of linear elasticity for ALE mesh motion is that artificial stresses are created when one part of the mesh rotates with respect to another part of the mesh; this can cause undesirable mesh distortion.

To create a mesh which can undergo large deformations and rotations, GOMA has a nonlinear ALE constitutive equation:

$$\boldsymbol{T}_s = 2G\boldsymbol{E}^* + Ke^*\boldsymbol{I} \tag{6.12}$$

$G = \mu$ is the shear modulus and K is the bulk modulus of the solid, $\boldsymbol{E}^*$ is a dilation-free measure of the large-deformation strain (Segalman et al. 1992), and e^* is the large deformation volume strain. In this formulation, stress due to shear is proportional to G, and stress due to expansion or contraction is proportional to K, thus G and K can be used to control the deformation of the mesh. If $G>>K$, then the elements are more likely to retain their shape (strong resistance to shearing), and if $G<<K$ then the elements will expand or contract uniformly.

For elasticity in Lagrangian solids, equation (6.10) is used to describe linear elasticity and several constitutive equations are used to describe non-linear elasticity. The non-linear constitutive equations include plane-stress and plane-strain formulations and can be applied incompressible, compressible or swelling materials. These models are described in more detail elsewhere.

6.4 Conservation of Energy

Conservation of energy results in a convection-conduction equation for the flow of energy through the domain:

$$\frac{d(\rho C_p T)}{dt} = -(\boldsymbol{v} - \boldsymbol{v}_m) \bullet \nabla(\rho C_p T) - \nabla\bullet\boldsymbol{q} + H \tag{6.13}$$

T is the temperature, C_p is the heat capacity, $\boldsymbol{q}$ is the heat flux by conduction, and H is the heat generation from a variety of sources. Currently there are several models for internal heat generation in GOMA, including Joule heating from a voltage potential being solved for with a species conservation equation (see Section 5.6). Also, through the user-defined subroutine feature, en-

thalpy methods can be installed to account for the latent heat generated by a phase change in solidifying materials.

6.5 Conservation of Total Mass (Continuity Equation)

The continuity equation describes conservation of total mass. In incompressible fluid mechanics, this equation states that the velocity field is divergence-free (i.e. there are no sources or sinks of mass within the flow-field):

$$\nabla \bullet \boldsymbol{v} = 0 \tag{6.14}$$

GOMA also has the capability to add a source term to this equation for a penalized pressure formulation of the Navier-Stokes equation.

In solid mechanics, the continuity equation relates the dilation of the solid material to the change in density of the overall system (including absorbed or interstitial fluid). The volume change of a solid material is equal to the determinant of the deformation gradient, $\boldsymbol{F}$:

$$\boldsymbol{F} = \boldsymbol{I} + (\nabla \boldsymbol{d})^{-1} \tag{6.15}$$

Here $\boldsymbol{I}$ is the identity tensor and $\boldsymbol{d}$ is the vector field of the displacement of the solid material from its stress-free-state. In porous or continuous materials in which the solid network only changes in volume due to changes in porosity or solvent content, the volume change (determinant of the deformation gradient) is directly related to the porosity or solvent volume fraction:

$$det|\boldsymbol{F}| = \frac{1-\phi_0}{1-\phi} \tag{6.16}$$

Here ϕ is the porosity in porous materials or the solvent volume fraction in continuous materials, and ϕ_0 is the porosity or solvent volume fraction in the stress-free state. Equation (6.16) is the continuity equation used by GOMA for solid mechanics. This equation is used to calculate the porosity in deformable porous media. This equation is used as a constraint equation to calculate the pressure (as a Lagrange multiplier) in deformable continuous solid materials.

6.6 Conservation of Component Mass

These equations are used to solve for the concentrations of an arbitrary number of components in a multicomponent fluid, solid, or porous medium. So far, GOMA has not been tested with more than two species, but it is general enough to handle three or more species. Some users have also used these equations to solve for other fields, such as an electric field, by adjusting the parameters (see for example the Heat Source card in Section 5.6).

Generalizing GOMA to solve for species transport in fluid, solid or porous systems has necessitat-

ed developing several forms of the conservation equations, which all fit into the general form in Table 6.1.

6.6.1 Convection-Diffusion Equation with ALE mesh

For standard convection-diffusion problems in fluids, the mesh moves independent of the fluid (except at free surfaces), and the conservation of component mass equation is:

$$\frac{dy_i}{dt} = -(\boldsymbol{v}-\boldsymbol{v}_m)\bullet\nabla y_i - \nabla\bullet\boldsymbol{J}_i + \boldsymbol{R}_i \tag{6.17}$$

Here y_i is the volume fraction of species i in solution, $\boldsymbol{v}$ is the volume averaged velocity of the fluid (normally a flow field from the Navier-Stokes equations), $\boldsymbol{J}_i$ is the diffusion flux of species i and $\boldsymbol{R}_i$ is the rate of production of species i by reactions. Currently, the formulation in GOMA assumes that the total density of the solution is constant, so that mass-fractions and volume-fractions are interchangeable, and the volume-averaged velocity is equal to the mass-averaged velocity; relaxing this assumption requires reformulating the continuity equation to account for variable mass. The diffusion flux currently is calculated from Fick's law for binary or pseudo-binary diffusion:

$$\boldsymbol{J}_i = -D\nabla y_i \tag{6.18}$$

Here D is the binary diffusion coefficient for species i. Currently the source term in the species equation is inactive, but could be customized through the USER option on the Species Source card (see Section 5.6).

6.6.2 Convection-Diffusion Equation with Lagrangian mesh

For convection-diffusion in solids in which the solid is Lagrangian (i.e. the nodes in the mesh are material points) the time derivative becomes a substantial time derivative with respect to the solid. Then the diffusion equation becomes:

$$\frac{dy_i}{dt} = -(\boldsymbol{v}_{sfs}\bullet\boldsymbol{F}-\boldsymbol{J}_s)\bullet\nabla y_i - \nabla\bullet\boldsymbol{J}_i + \boldsymbol{R}_i \tag{6.19}$$

The only difference between this equation and (6.17) is the advective term. The first part of the advective term, $\boldsymbol{v}_{sfs}\bullet\boldsymbol{F}$, represents steady-state motion of a solid body in "convected coordinates", in which the solid material convects, in a moving frame of reference, through the domain. $\boldsymbol{v}_{sfs}$ is the steady velocity of the undeformed solid, which is zero in most problems. $\boldsymbol{F}$ is the deformation gradient (see (6.15)), which is calculated from the solid deformation field. The second part of the advective term, $\boldsymbol{J}_s$, represents the flux of solid relative to the mass-averaged velocity:

$$\boldsymbol{J}_s = \frac{-\sum_{j=1}^{nc-1} \boldsymbol{J}_j}{1-\sum_{j=1}^{nc-1} y_j} \tag{6.20}$$

Thus, the solid flux is equal to the sum of the fluid diffusion fluxes divided by the volume fraction of solid.

6.6.3 Solvent Transport in Partially-Saturated Porous Media

For problems involving partially-saturated flow in porous media, the conservation equations for water and air are derived for flow and diffusion in both the liquid and gas phases (Martinez 1995):

$$\frac{dC_i}{dt} = (\boldsymbol{v}_m + \boldsymbol{v}_{sfs} \bullet \boldsymbol{F}) \bullet \nabla C_i - \nabla \bullet [\boldsymbol{v}_g \rho_{gi} + \boldsymbol{v}_l \rho_{li} + \boldsymbol{J}_{gi} + \boldsymbol{J}_{li}] \tag{6.21}$$

Here, C_i is the total concentration of species i from both phases (per unit volume of media). Currently GOMA is set up for two-phase transport in porous media in which the liquid phase is pure solvent, and the gas phase contains both air and solvent vapor; species equation one (i=1) corresponds to the mass balance for the solvent, and species equation two corresponds to the mass balance for air. $\boldsymbol{v}_g$ and $\boldsymbol{v}_l$ are the velocities of gas and liquid relative to the solid skeleton and are normally calculated from Darcy's Law. ρ_{gi} and ρ_{li} are the concentrations of species i in the gas and liquid phases, and should be in local equilibrium. $\boldsymbol{J}_{gi}$ and $\boldsymbol{J}_{li}$ are the diffusion fluxes of species i in the gas and liquid phases. In GOMA the whole last term is considered the diffusion term, even though it includes convection in the gas and liquid phases.

6.7 Boundary Conditions

The boundary conditions in GOMA are numerous and some are specific to a problem class; for brevity in this section only a list the boundary conditions currently active for each equation is given. Discussion of how the boundary condition is implemented in a finite element framework is not covered here in any detail.

Boundary conditions are split into five classes: Dirichlet, pointwise collocation, weak integrated conditions, strong integrated conditions, and special conditions. *Dirichlet conditions (DC)* specify the value of a field variable along a node-set boundary; these conditions completely replace the bulk weighted residual equations associated with the nodes. *Pointwise collocation conditions (PCC)* are boundary conditions that are applied at nodal points and are penalized (multiplied by a large number) so they overwhelm the contributions from the bulk weighted residual equations. *Weak integrated conditions (WIC)* replace the BOUNDARY terms in the weighted residuals by a prescribed function (see for example Eqn. (6.5)), which is integrated along the boundary. *Strong*

integrated conditions (SIC) are penalized conditions which are integrated over the boundary. *Special conditions (SC)* are conditions that do not fit into the above three categories; currently all the special conditions are boundary conditions that apply at single points, such as a dynamic contact line. Here the mathematical form of each condition as they pertain to a specific differential equation is presented.

6.7.1 Boundary Conditions on the Fluid Momentum Equations

Table 6.2 tabulates all conditions that can be applied to the fluid momentum equations. It is important to realize that each condition of type DC can be applied as a constraint on any other differential equation through the generalized Dirichlet condition capability described in Table 4.5. The notation in Table 6.2 goes as follows: $\underset{\sim}{v}$ is the vector velocity; $\underset{\sim}{v}_s$ is the velocity of the boundary itself (not independent from the mesh velocity); $\underset{\sim}{n}$ is the normal vector to the surface; $\underset{\sim}{m}$ is the tangent vector to the surface where it intersects a solid boundary; $\underset{\sim}{n}_w$ is the normal vector to the wall or boundary intersected by a capillary surface (at a contact line); $\underset{\sim}{T}$ is the fluid stress tensor; $2H$ is the mean curvature; σ is the surface tension, and $\underset{\sim}{e}_i$ is a base unit vector in the chosen coordinate system. h is the mass-transfer coefficient of species i whose concentrations denoted by y_i.

Table 6.2 Boundary Conditions for Fluid Momentum Equations

Name and Description	Mathematical Form	Implementation
U -- Specified x-component of velocity (or z-component for axisymmetric analysis) V Specified y-component of velocity (or r-component for axisymmetric analysis) W Specified z-component of velocity	$\underset{\sim}{e}_x \cdot \underset{\sim}{v} = v_0$ $\underset{\sim}{e}_y \cdot \underset{\sim}{v} = v_0$ $\underset{\sim}{e}_z \cdot \underset{\sim}{v} = v_0$	DC
UVARY -- Specified function for x-component of velocity VVARY (y-component) WVARY (z-component)	$\underset{\sim}{e}_x \cdot \underset{\sim}{v} = f(y,z)$ $\underset{\sim}{e}_y \cdot \underset{\sim}{v} = f(z,x)$ $\underset{\sim}{e}_z \cdot \underset{\sim}{v} = f(x,y)$	PCC
VELO_NORMAL Specify normal velocity component along boundary	$\underset{\sim}{n} \cdot (\underset{\sim}{v} - \underset{\sim}{v}_s) = v_0$	SIC

Table 6.2 Boundary Conditions for Fluid Momentum Equations

Name and Description	Mathematical Form	Implementation
VNORM_LEAK Specify normal velocity component as a function of mass transfer of species i	$\underset{\sim}{n} \cdot (\underset{\sim}{v} - \underset{\sim}{v}_s) = \sum_i h\left(y_i - y_i^0\right)$	SIC
VELO_SLIP Navier-Slip Condition	$\underset{\sim}{t} \cdot \underset{\sim}{n} \cdot \underset{\sim}{\boldsymbol{T}}\big\|_1 = \frac{1}{\beta}(\underset{\sim}{v} - \underset{\sim}{v}_s) \cdot \underset{\sim}{t}$	WIC
CAPILLARY Balance of viscous stresses with capillary pressure and surface tension gradients at an interface	$\underset{\sim}{n} \cdot \underset{\sim}{\boldsymbol{T}}\big\|_1 = \underset{\sim}{n} \cdot \underset{\sim}{\boldsymbol{T}}\big\|_2 + 2H\sigma\underset{\sim}{n} + \nabla_s\sigma$	WIC
CA Specify contact angle	$\underset{\sim}{n} \cdot \underset{\sim}{n}_w = \cos\theta$	SC
SURFTANG SURFTANG_SCALAR Add endpoint force to capillary free surface	$\underset{\sim}{m}\sigma = \underset{\sim}{m}_0\sigma_0$	SC
FLUID_SOLID Equate solid stresses to fluid stresses at solid/ fluid boundary.	$\underset{\sim}{n} \cdot \underset{\sim}{\boldsymbol{T}}\big\|_s = \underset{\sim}{n} \cdot \underset{\sim}{\boldsymbol{T}}\big\|_f$	WIC

6.7.2 Boundary Conditions on the Solid and Pseudo-Solid Momentum Equations

Table 6.2 presents the mathematical form for all boundary conditions which can be applied to the solid or pseudo-solid momentum equations. Again, it is important to point out that each condition of type DC can be applied as a constraint on any other differential equation through the generalized Dirichlet condition capability described in Table 4.5. Also, there are many interesting and challenging issues when boundary conditions for the solid momentum equations (which are normal treated in a Lagrangian or convected Lagrangian framework) at fluid boundaries (which are always treated in an arbitrary Lagrangian/Eulerian framework). For instance, the SOLID_FLUID boundary condition cannot currently be applied to a boundary which contains a contact line. These and other related issues are not taken up here. The reader is referred to an example problem in Chapter 7 (Section 7.6) for further discussion. The notation in Table 6.2 goes as follows: $\underset{\sim}{d}$ is the displacement vector; $\underset{\sim}{v}$ is the vector velocity; $\underset{\sim}{v}_s$ is the velocity of the boundary itself (not independent from the mesh velocity); $\underset{\sim}{n}$ is the normal vector to the surface; $\underset{\sim}{n}_w$ is the normal vector to the wall or boundary intersected by a capil-

lary surface (at a contact line); $\underset{\sim}{T}$ is the fluid or solid stress tensor; $2H$ is the mean curvature; σ is the surface tension, and $\underset{\sim}{e}_i$ is a base unit vector in the chosen coordinate system; x, y, and z are Cartesian coordinates; and h is the mass-transfer coefficient of species i whose concentrations denoted by y_i.

Table 6.3 Boundary Conditions for Solid and Pseudo-Solid Momentum Equations

Name and Description	Mathematical Form	Implementation
DX Specified x-component of displacement (or z-component for axisymmetric analysis) DY Specified y-component of displacement (or r-component for axisymmetric analysis) DZ Specified z-component of displacement	$\underset{\sim}{e}_x \cdot \underset{\sim}{d} = d_0$ $\underset{\sim}{e}_y \cdot \underset{\sim}{d} = d_0$ $\underset{\sim}{e}_z \cdot \underset{\sim}{d} = d_0$	DC
PLANE Specify planar surface of mesh	$ax + by + cz + d = 0$	PCC
SPLINE Specify general surface geometry	$f(x,y,z) = 0$	PCC
DISTNG Constant isotherm distinguishing condition	$T - T_0 = 0$	PCC
KINEMATIC Kinematic distinguishing condition	$\underset{\sim}{n} \cdot (\underset{\sim}{v} - \underset{\sim}{v}_s) = v_0$	SIC
KIN_LEAK Kinematic distinguishing condition. Mass balance with mass transfer of bulk species.	$\underset{\sim}{n} \cdot (\underset{\sim}{v} - \underset{\sim}{v}_s) = \sum_i h\left(y_i - y_i^0\right)$	
SLOPE Apply slope at boundary (Same as PLANE above without a datum)	$\underset{\sim}{n} \cdot \underset{\sim}{n}_{vector} = 0$	SIC

Table 6.3 Boundary Conditions for Solid and Pseudo-Solid Momentum Equations

Name and Description	Mathematical Form	Implementation
FORCE Apply force on mesh at boundary equal to vector	$\underset{\sim}{n}_0 \cdot \underset{\sim}{F} = F_0$	WIC
SOLID_FLUID Equate solid stresses to fluid stresses at solid/fluid boundary	$\underset{\sim}{n} \cdot \underset{\sim}{T}\big\|_s = \underset{\sim}{n} \cdot \underset{\sim}{T}\big\|_f$	WIC
NO_SL:IP Equate velocity of the liquid at the interface to the velocity in the solid at the interface	$\underset{\sim}{v}\big\|_{liquid} = (\underset{\sim}{v}_m + \underset{\sim}{v}_{sfs} \cdot \underset{\sim}{F})\big\|_{solid}$	PCC

6.7.3 Boundary Conditions on the Energy Equation

Table 6.2 presents the mathematical form for all boundary conditions which can be applied to the energy equation. At this time no enclosure radiation type conditions can be applied; however, one could implement a black body (εT^4) type condition through the user-defined capability, i.e., QUSER. The notation in Table 6.2 goes as follows: $\underset{\sim}{q}$ is the heat flux vector; T is the temperature; $\underset{\sim}{n}$ is the normal vector to the surface; and h is the heat-transfer coefficient.

Table 6.4 Boundary Conditions for the energy equation.

Name and Description	Mathematical Form	Implementation
T Set constant temperature	$T = T_0$	DC
QSIDE Set constant heat flux on side	$\underset{\sim}{n} \cdot \underset{\sim}{q} = q_0$	WIC
QRAD QCONV Convective Heat Transfer Flux	$\underset{\sim}{n} \cdot \underset{\sim}{q} = h\,(T - T_0)$	WIC

6.7.4 Boundary Conditions on the Species Component Equations

Table 6.2 presents the mathematical form for all boundary conditions which can be applied to each component of the species transport equations. Care must be taken here if the species concentration is high enough to be outside the "dilute species" assumption, in which case transport of species through boundaries will effect the volume of the bounding fluids. Several conditions above account for this effect, including the VNORM_LEAK (Table 6.2) and the KIN_LEAK (Table 6.2) boundary conditions. The notation in Table 6.2 goes as follows: $\underset{\sim}{q}$ is the heat flux vector; T is the temperature; $\underset{\sim}{n}$ is the normal vector to the surface; and h is the heat-transfer coefficient.

Table 6.5 Boundary Conditions for Species Component Equations

Name and Description	Mathematical Form	Implementation
Y Set constant species concentration	$y = y_i^0$	
YFLUX/SS	$\underset{\sim}{n} \cdot \underset{\sim}{J}_i + \underset{\sim}{n} \cdot [(\underset{\sim}{v} - \underset{\sim}{v}_m) \nabla y_i] = h\left(y_i - y_i^0\right)$	DC
YFLUX_CONST/SS	$\underset{\sim}{n} \cdot \underset{\sim}{J}_i = q_i^0$	WIC
POROUS_FLUX/SS	$n \cdot \left(v_g \rho_{g_i} + v \rho_{l_i} + J_{g_i} + J_{l_i}\right) = k\phi\left(\rho_{g_i} - \rho_{g_i}^0\right)$	WIC

7 Tutorial and Example Problems

This chapter is intended to serve as a tutorial on the use of GOMA and to illustrate its capabilities. With regard to the tutorial, the details of running several problems with the complication of coupled physics but in simple geometries with coarse meshes are presented. In this way they run rapidly but illustrate how a coupled problem is set up. Also included are several advanced examples which serve to illustrate three advanced features: the free boundary capability, multiple physics and multiple materials (conjugate problem capability), and nonlinear problem analysis. Each example can be located in a companion directory that is supplied with the distribution.

Of course these are not the only test problems available. Others which exercise the various features of GOMA are available on request.

7.1 Graetz-Nusselt Problem

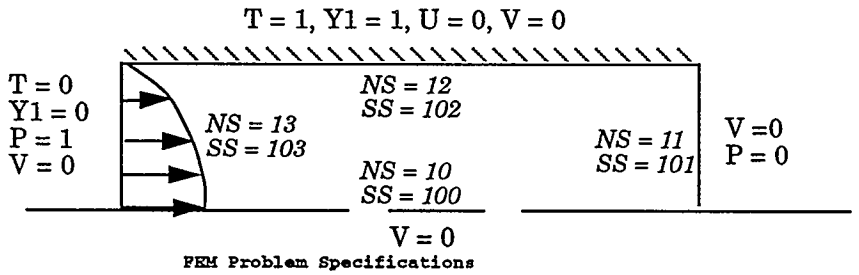
This problem is one of tube flow where the wall temperature suddenly changes at some plane through the tube from one fixed value to another. This is the only fixed-grid example illustrated in this chapter. To illustrate the addition of other field variables, a species field has been added to the problem. Figure 6 depicts the problem geometry and boundary conditions, together with the problem-description file. The "mat" file is not shown because all thermophysical properties are taken as unity.

Noteworthy in the problem description file is the use of the CAPILLARY card to apply pressure boundary conditions. The CAPILLARY BC card (Section 4.5.2) can be used on any side-set boundary (see Schoof and Yarberry 1994) as a means of applying the pressure by simply setting the surface tension and boundary attraction forces to zero.

Upon running GOMA, the following output is received on the standard output device (i.e., the screen if not routed elsewhere on the command line):

```
Copyright (c) 1993-1995 Sandia National Laboratories
                PRS, PAS, RRR, KSC, & RAC
 ToD    itn     L_oo  L_1    L_2     L_oo       L_1    L_2     asm/slv (sec)
-------- --- ------- ------ ------- ------- ------- ------- -----------------
17:05:42 [0] 2.4e-01 4.3e+00 8.6e-01  1.0e+00 7.4e+01 7.7e+00 2.93e+00/1.23e+00
17:05:53 [1] 4.4e-03 2.9e-02 8.2e-03 2.9e-02 5.7e-01 9.7e-02  2.93e+00/4.00e-01
17:06:02 [2] 4.5e-07 4.3e-06 9.7e-07  2.2e-06 5.5e-05 8.2e-06  2.86e+00/3.20e-01
17:06:11 [3] 1.0e-15 8.5e-15 1.9e-15  7.5e-15 6.5e-14 1.5e-14  2.85e+00/3.30e-01
```

Clearly the Newton iteration procedure converged quadratically to the solution. Figure 7 shows the computational mesh, the velocity vectors, and the temperature field.




```
FEM Problem Specifications
--- ------- --------------
FEM file                              = pipe.exoII
Output EXODUS II file                 = pipe_o.exoII
GUESS file                            = pipe.d
SOLN file                             = pipe_o.d
Write intermediate results            = yes

Time Integration Specifications
---- ----------- --------------
Time integration                      = steady

Solver Specifications
------ --------------
Solution Algorithm                    = lu
Number of Newton Iterations           = 7
Newton correction factor              = 1
Normalized Residual Tolerance         = 1e-10
Residual Ratio Tolerance              = 1e-2

Boundary Condition Specifications
-------- --------- --------------
Number of BC                           = 10
#
BC  = U           NS 12                 0
BC  = T           NS 12                 1
BC  = Y           NS 12       0         1
BC  = V           NS 11                 0
BC  = V           NS 13                 0
BC  = T           NS 13                 0
BC  = Y           NS 13       0         0
BC  = V           NS 12                 0
BC  = CAPILLARY    SS 103     0 1 0
BC  = CAPILLARY    SS 101      0 0 0
----
Problem Description
---
Number of Materials                   = 1
MAT                                   = sample    1
Coordinate System = CYLINDRICAL
  Element Mapping   = isoparametric
  Mesh Motion = ARBITRARY
  Number of bulk species = 1

Number of EQ                           = 5
EQ  = momentum1    Q2 U1 Q2    0     1    1    1    1    0
EQ  = momentum2    Q2 U2 Q2    0     1    1    1    1    0
EQ  = energy       Q2 T  Q2          0.   1    1    1    0.
EQ  = species_bulk Q2 Y Q2    0.     1.   1.   1    0.
EQ  = continuity   P1 P  P1   1       0

Post Processing Specifications
---- ---------- --------------
Stream Function                       = yes
Pressure contours                     = yes
Navier Stokes Residuals               = yes
```

Figure 6 Geometry and problem description file for Graetz-Nusselt

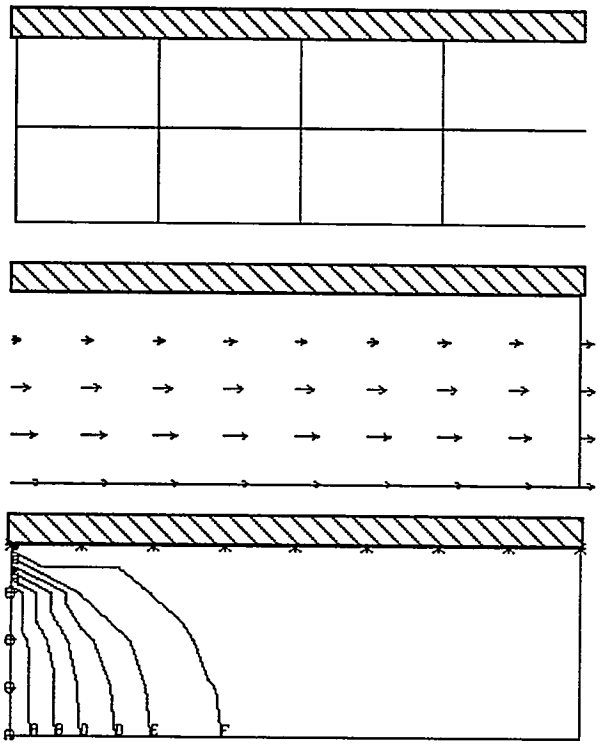


Figure 7 Solution to Graetz-Nusselt example problem. Computational mesh (top), velocity vectors (middle), isotherms (bottom).

7.2 Melting Problem

The idea in this problem is to parameterize the isotherms which characterize the melting front of a multicomponent material (typically a metal alloy). The bounding isotherms of the front are usually called the "liquidus" and the "solidus". By requiring the mesh to conform to these isotherms, the correct physical description can be applied to each phase. For example, in the liquid melt phase, heat transfer and fluid mechanics requires the energy equation and the Navier-Stokes equations. In the mushy zone the transport calls for the equations describing flow through porous media, i.e., either Brinkman or Darcy equations, together with the energy equation. In the solid, the appropriate description involves energy transport and solidification shrinkage. In this example, the porous flow regime and the solidification shrinkage are neglected and the mushy zone and solid are treated as a medium for energy transport. However, the fluid mechanics are added to the melt, in which flow is driven by thermal expansion.

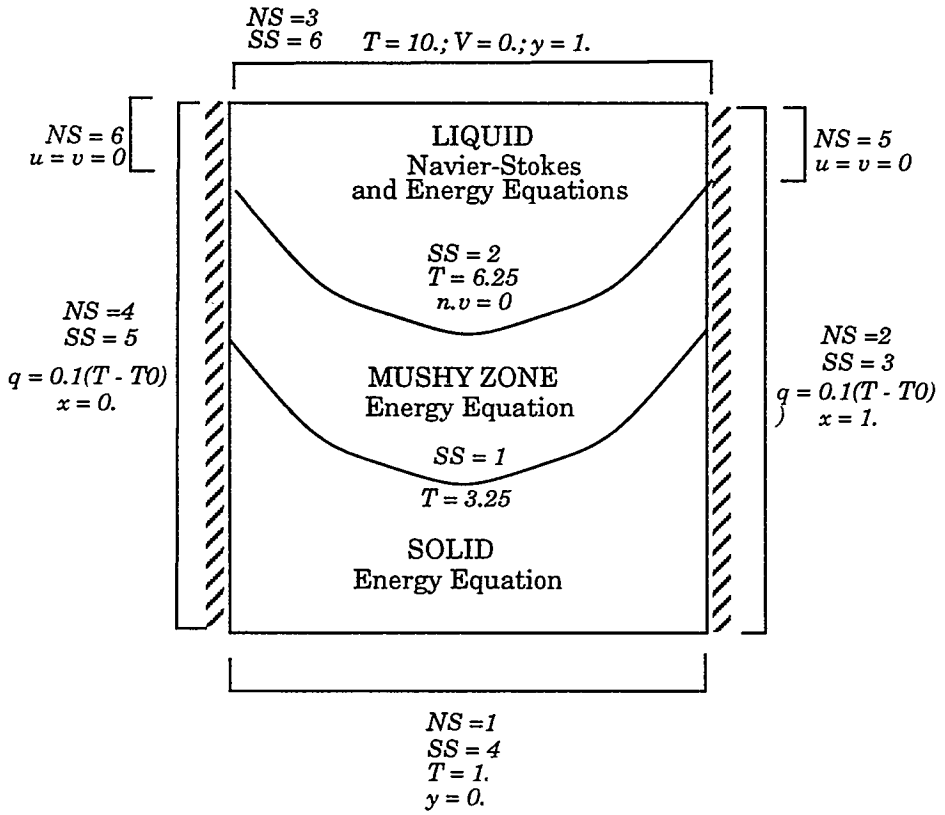


Figure 8 Melting conjugate problem geometry and boundary conditions.

Figure 8 shows the problem geometry (here a two-dimensional slab is considered, not a cylinder), the boundary conditions, and the side and node set information which connect the problem description file with the EXODUS II database. Here several things must be noted. First, there are three materials (three element blocks) which must be designated in the EXODUS II database. This is done in the mesh generation step, which in this case is FASTQ. Second, because this is a free boundary problem, distinguishing conditions must be supplied for all surfaces, including the external "fixed" boundaries. The PLANE command is used to fix the external boundaries and the DISTNG boundary condition is used to distinguish the internal free boundaries. These boundary condition cards are explained in Section 4.5.2. Third, because there are different numbers of physical equations in each zone, care must be taken in the node-set and side-set designation. The fluid mechanics in the "liquid" material requires extra boundary conditions, over and above those for the moving mesh and the energy transport, on the side walls. This makes necessary the node sets 5 and 6, as shown in Figure 8.

The problem-description file is as follows:

```
FEM file                          = m_matl.exoII
Output EXODUS II file             = out.exoII
GUESS file                        = contin.dat
SOLN file                         = soln.dat
Write intermediate results        = no
 ---
General Specifications
 ---
Number of processors              = 1
Output Level                      = 0
Debug                             = 0
Initial Guess                     = read_exoII
 ---
Time Integration Specifications
 ---
Time integration                  = steady
---
Solver Specifications
 ---
Solution Algorithm                = lu
Number of Newton Iterations       = 6
Newton correction factor          = 1
Normalized Residual Tolerance     = 1.0e-11
Residual Ratio Tolerance          = 1.0e-3
 ---
Boundary Condition Specifications
 ---

Number of BC                    = 16
BC = T NS 1 1.
BC = T NS 3 10.
BC                                = QRAD SS 5 0.1 0.
BC                                = QRAD SS 3 0.1 0.
BC                                = PLANE SS 4 -0.0 1.0 0.0 -0.00
BC                                = PLANE SS 3 1.0  0.0 0.0 -1.0
BC                                = PLANE SS 6 0.0  1.0 0.0 -1.0
BC                                = PLANE SS 5 1.0  0.0 0.0  0.0
BC                                = DISTNG SS 2 6.25
BC                                = DISTNG SS 1 3.25

BC                                      = V NS 3  0.
BC                                = VELO_NORMAL SS 2 0
BC                                      = U NS 5  0.
BC                                      = V NS 5  0.
BC                                      = U NS 6  0.
BC                                      = V NS 6  0.

----
Problem Description
---

Number of Materials = 3

MAT = sample    10

  Coordinate System = CARTESIAN
  Element Mapping   = isoparametric
  Mesh Motion = ARBITRARY
  Number of bulk species = 0

Number of EQ                    = 3
EQ = mesh1                      Q2D1Q2    0.      0.        0.   1.       0.  0.
EQ = mesh2                      Q2D2Q2    0.      0.        0.   1.       0.  0.
EQ = energy     Q2       T      Q2              0.      1.00    1.     1.       0.
                                        div  ms  adv   bnd  dif    src   porous

MAT = sample1    20

  Coordinate System = CARTESIAN
  Element Mapping   = isoparametric
  Mesh Motion = ARBITRARY
  Number of bulk species = 0
```

```
Number of EQ                    = 3
EQ = mesh1                      Q2D1Q2    0.      0.        0.   1.      0.  0.
EQ = mesh2                      Q2D2Q2    0.      0.        0.   1.      0.  0.
EQ = energy     Q2       T      Q2              0.      1.00    1.     1.      0.
                                          div  ms  adv   bnd  dif    src   porous

MAT = sample2    30

  Coordinate System = CARTESIAN
  Element Mapping   = isoparametric
  Mesh Motion = ARBITRARY
  Number of bulk species = 0

Number of EQ =6
EQ = mesh1 Q2D1Q2     0.   0.    0.   1.      0.  0.
EQ = mesh2 Q2D2Q2     0.   0.    0.   1.      0.  0.
EQ = energy     Q2       T      Q2     0.   1.    1.   1.     0.
EQ = momentum1  Q2U1Q2     0.   1.    1.   1.0    1.0   0.
EQ = momentum2  Q2U2Q2     0.   1.    1.   1.0    1.0   0.
EQ = continuityP1P P1  1.                          0.
                                       div  ms  adv   bnd  dif    src   porous


Post Processing Specifications

Stream Function = yes
Streamwise normal stress = no
Pressure contours = no
Second Invarient of Strain = no
Mesh Dilatation = no
Navier Stokes Residuals = no
Moving Mesh Residuals = no
Mass Diffusion Vectors = no
Mass Fluxlines = no
Energy Conduction Vectors = no
Energy Fluxlines = no
Time Derivatives = no
Mesh Stress Tensor = no
```

Noteworthy in this file are the three "sub-problem" descriptions representing element blocks 10, 20, and 30. Block 30 is a material called "sample2" and is described with six differential equations, while the others are described with three. In all cases the mesh motion equations are needed to account for the *a priori* unknown positions of the liquidus and solidus. The material file for the molten phase (element block 30 material named "sample2") is

```
Material Data File for SAMPLE
---Physical Properties
Density                                     = CONSTANT 1000.

---Mechanical Properties and Constitutive Equations
Solid Constitutive Equation = NONLINEAR_PLANE_STRAIN
Convective Lagrangian Velocity= NONE
Lame MU                                     = CONSTANT 1.
Lame LAMBDA                                 = CONSTANT 1.
Stress Free Solvent Vol Frac= CONSTANT      0.
Liquid Constitutive Equation = NEWTONIAN
Viscosity                                   = CONSTANT 1.
Low Rate Viscosity                          = CONSTANT 0.
Power Law Exponent                          = CONSTANT 0.
High Rate Viscosity                         = CONSTANT 0.
Time Constant                               = CONSTANT 0.
Aexp                                        = CONSTANT 0.

---Thermal Properties
Conductivity                                = CONSTANT.1
Heat Capacity                               = CONSTANT1.
Volume Expansion= CONSTANT                  3.e-7
Reference Temperature= CONSTANT             0.
Liquidus Temperature= CONSTANT              1.
Solidus Temperature= CONSTANT               0.
```

```
---Microstructure Properties
Media Type                = CONTINUOUS
Porosity                  = CONSTANT      1.0
Permeability              = CONSTANT       9.

---Brinkman-equation parameters
Brinkman Porosity    =  CONSTANT      1.0
Brinkman Permeability    =  CONSTANT      0.9
FlowingLiquid Viscosity    =  CONSTANT      1.5
Inertia Coefficient    =  CONSTANT      0.

---Species Properties
Diffusion Constitutive Equation = FICKIAN
Diffusivity                                    = CONSTANT  0    1.
Latent Heat Vaporization= CONSTANT  0          0.
Latent Heat Fusion                             = CONSTANT  00.
Vapor Pressure                                 = CONSTANT  00.
Species Volume Expansion          = CONSTANT  0    0.
Reference Concentration           = CONSTANT  0    0.
                                               ^
********Species Number**********************|

----Source Terms
Navier-Stokes Source                           = BOUSS 0.  -1.  0.
Solid Body Source                              = CONSTANT 0.  0.  0.
Mass Source                                    = CONSTANT 0.
Heat Source                                    = CONSTANT 0.
Species Source                                 = CONSTANT  0    0.
```

The material files for the other two phases in this case turn out to be identical, which in general is not the case because of varying properties, but is done for brevity here. Noteworthy in this file is the BOUSS option on the Navier-Stokes source card which sets the magnitude of gravity and uses the thermal and solutal expansion coefficients set above.

Complete solution to this problem is accomplished through three or four continuation steps. Although there are several ways to reach the final solution, here is one that works:

1. Begin with "`Number of BC = 8`" and a problem description which includes only the mesh motion and energy equations for all phases, i.e., set "`Number of EQ = 3`" for all phases. This basically is a heat conduction problem with free boundaries, but the boundaries will not move. Start with either a "`zero`" initial guess or a "`read_exoII`". Running GOMA returns:

```
Cannot find TEMPERATURE in exoII database, setting to null
 Cannot find DMX in exoII database, setting to null
 Cannot find DMY in exoII database, setting to null
  ToD     itn   L_oo     L_1     L_2      L_oo     L_1     L_2      asm/slv (sec)
-------- --- ------- ------- -------  ------- ------- -------  -----------------
14:41:44 [0] 2.9e-02 5.3e-01 8.9e-02  9.5e+00 1.2e+03 9.0e+01  1.13e+01/1.25e+01
14:42:38 [1] 1.7e-16 6.9e-15 6.8e-16  1.4e-14 8.3e-13 7.3e-14  1.07e+01/1.91e+00
```

2. Change "`Number of BC = 10`". This basically is a moving boundary problem with two free surfaces distinguished by the T=6.25 and T=3.25 boundary conditions. Copy file "`out.exoII`" to "`m_matl.exoII`" Running GOMA returns:

```
ToD     itn   L_oo     L_1     L_2      L_oo     L_1     L_2      asm/slv (sec)
-------- --- ------- ------- -------  ------- ------- -------  -----------------
14:43:44 [0] 1.0e+00 6.8e+00 2.0e+00  1.0e+00 4.2e+01 4.0e+00  1.00e+01/9.96e+00
14:44:24 [1] 8.9e-04 3.1e-02 3.3e-03  7.0e-02 2.3e+00 2.7e-01  1.02e+01/1.74e+00
14:44:49 [2] 5.5e-04 4.4e-03 1.0e-03  3.2e-02 6.3e-01 9.9e-02  1.10e+01/1.27e+00
14:45:14 [3] 1.1e-04 6.6e-04 1.9e-04  7.0e-03 1.1e-01 1.9e-02  1.07e+01/1.30e+00
14:45:38 [4] 3.1e-06 1.6e-05 5.1e-06  2.2e-04 3.6e-03 6.2e-04  1.10e+01/1.35e+00
14:46:05 [5] 2.6e-09 1.3e-08 4.2e-09  1.9e-07 3.1e-06 5.2e-07  1.05e+01/1.25e+00
```

3. Choose "Number of BC = 16" and "Number of EQ = 6", the latter on the third phase only (element block 30). Set "Volume Expansion" in sample2.mat file to 1.e-8 or smaller. Again copy file "out.exoII" to "m_mat1.exoII" and run GOMA. GOMA returns:

```
Cannot find VX in exoII database, setting to null
Cannot find VY in exoII database, setting to null
  ToD     itn   L_oo     L_1     L_2      L_oo     L_1     L_2      asm/slv (sec)
-------- --- ------- ------- -------  ------- ------- -------  -----------------
14:46:58 [0] 2.6e-08 2.3e-06 1.8e-07  4.5e-05 3.6e-03 2.6e-04  2.28e+01/5.81e+01
14:49:42 [1] 4.4e-12 2.5e-10 1.7e-11  5.4e-06 1.7e-04 2.9e-05  2.29e+01/8.73e+00
14:50:40 [2] 2.2e-15 2.4e-14 3.4e-15  5.4e-10 1.7e-08 2.9e-09  2.29e+01/6.37e+00
```

4. Increase "Volume Expansion" to 1.e-07 and again copy file "out.exoII" to "m_mat1.exoII" and run GOMA. GOMA returns:

```
ToD     itn   L_oo     L_1     L_2      L_oo     L_1     L_2      asm/slv (sec)
-------- --- ------- ------- -------  ------- ------- -------  -----------------
15:27:40 [0] 2.8e-07 2.5e-05 1.8e-06  4.3e-04 3.3e-02 2.4e-03  2.49e+01/6.39e+01
15:30:28 [1] 3.6e-10 2.2e-08 1.4e-09  4.9e-04 1.5e-02 2.6e-03  2.46e+01/9.54e+00
15:31:32 [2] 1.8e-11 1.5e-10 2.6e-11  4.3e-06 1.3e-04 2.3e-05  2.47e+01/7.24e+00
15:32:34 [3] 1.3e-15 2.1e-14 2.5e-15  7.2e-10 2.2e-08 3.8e-09  2.54e+01/7.02e+00
```

The warning messages regarding the missing field variables, e.g., "Cannot find VX in exoII database...", indicate that field variables are being added to the problem. The final results of the last continuation step are shown in Figure 9 for a volume expansion coefficient of 1.e-07.

Note: In conjugate problems, elements must be conforming, especially at phase boundaries.

7.3 Slot Coating

Slot coating flow is commonly used to deliver a uniform sheet of liquid onto a moving substrate over fairly large widths. Although the flow is three dimensional near the edges of a coating, the most desirable flow regime is two dimensional, as is illustrated in this example. The key geometrical features of the problem are shown in Figure 10a. Here operating conditions and material parameters have been chosen similar to those chosen by Chen et al. (1995) and Sartor (1990). Besides the presence of multiple free surfaces (so-called upstream and downstream menisci), this flow is made interesting and challenging by a very narrow range of operating parameters for which a stable, steady solution exists. In this regard, the three most important operating parameters are substrate speed, flow rate (or liquid delivery rate through the slot), and back pressure (i.e., the vacuum applied to the upstream meniscus to stabilize the coating bead at higher coating speeds).

Several interesting features of GOMA are exercised in this analysis. First, the material properties are controlled through the equation term multipliers in the input deck (cf. Section 4.6 and Table 4.7). The appropriate section of the input deck is:

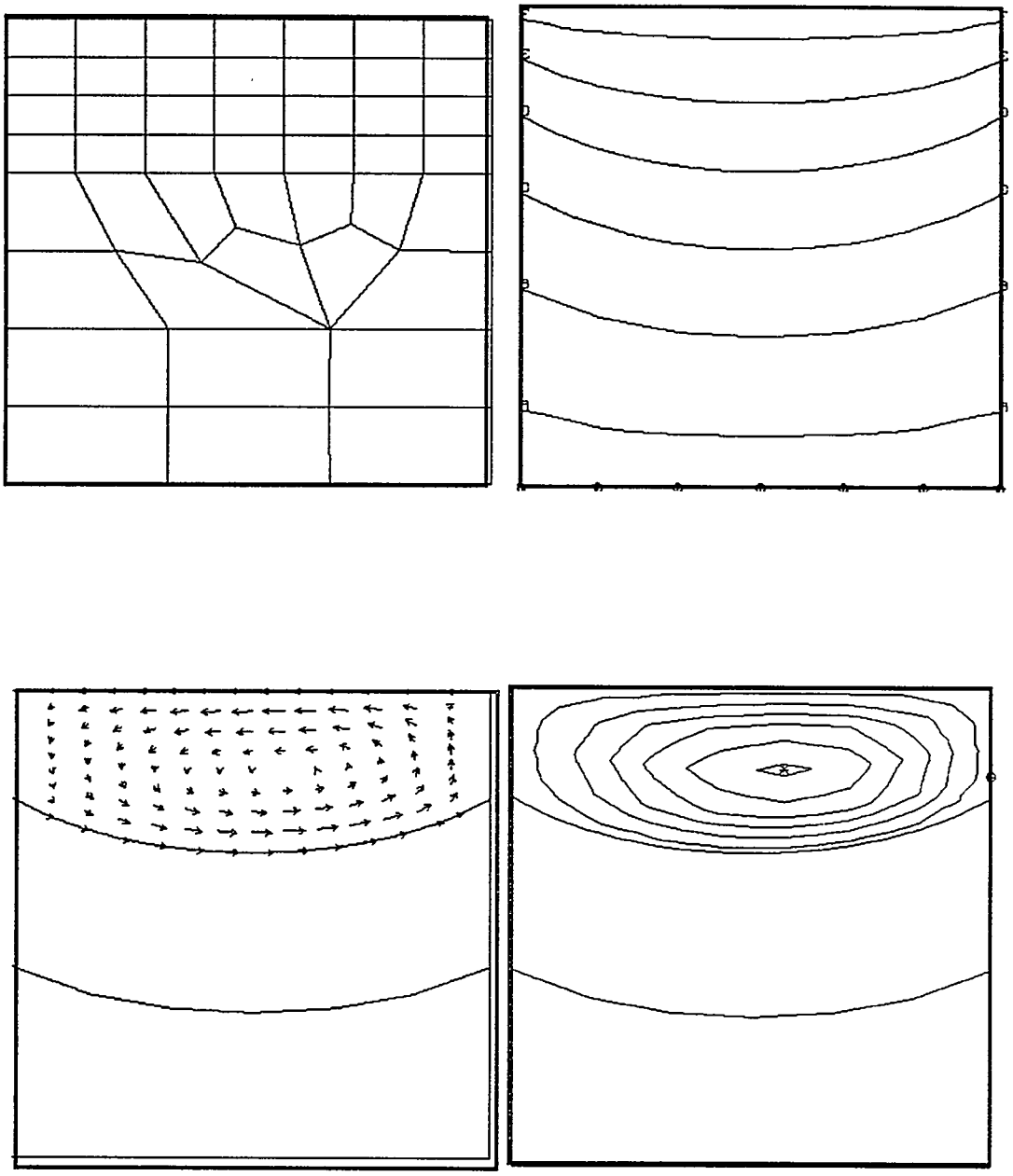


Figure 9 Results from melting problem. Clockwise from upper left: finite element mesh, isotherms, pattern of streamlines in melt phase, and velocity vectors in melt phase.

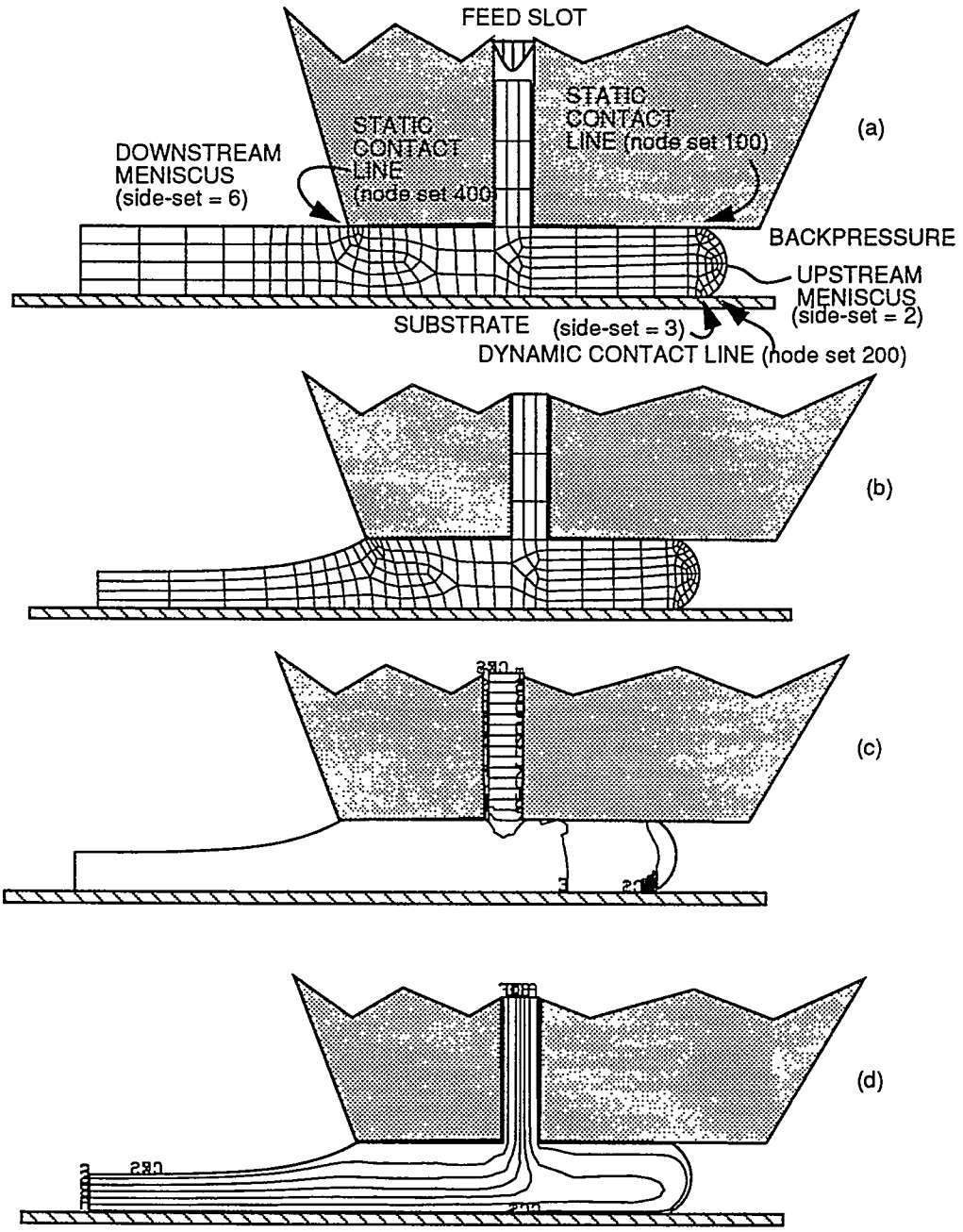


Figure 10 Slot coating -- typical results. a) undeformed mesh (initial guess); b) deformed mesh; c) pressure contours; d) pattern of streamlines. (Web speed = 0.133m/s, slot gap = 0.5 mm, back pressure = -3675 Pa, inflow maximum speed = 0.21 m/s).

```
----
 Problem Description
 ---

 Number of Materials = 1

 MAT = fluid    1

        Coordinate System = CARTESIAN

        Element Mapping                  = isoparametric

        Mesh Motion = ARBITRARY

        Number of bulk species = 0

        Number of EQ            = 5
        EQ = momentum1  Q2      U1      Q2              0. 1000. 1. 1.0 0.0 0.
        EQ = momentum2  Q2      U2      Q2              0. 1000. 1. 1.0 -1000.0 0.
        EQ = continuityP1       P       P1      1.                              0.
        EQ = mesh1      Q2      D1      Q2              0. 0. 0.    1. 0. 0.
        EQ = mesh2      Q2      D2      Q2              0. 0. 0.    1. 0. 0.
                                                        div ms adv bnd dif src porous
```

Particularly noteworthy is the multiplier on the advection terms in the momentum equations are set to the density of the coating liquid (1000 kg/m3); the magnitude of the viscosity is set to 1.0 Pa-s through the diffusion term, and the magnitude of the gravitational body force is set on the *momentum2* source term. The material file, "`fluid.mat`", sets all relevant thermophysical properties and source terms to unity, and so is not shown here. In fact the only non-trivial feature employed in the material file regards the "`Lame MU`" coefficient model (Section 5.2), in which the `CONTACT_LINE` option is employed to help mitigate mesh distortion around a contact line. The card looks like:

```
Lame MU          = CONTACT_LINE 200 0.4 1000. 0.00025
```

In this case the Lame coefficient μ varies from 1000. to 0.4 over a distance of 250 μm.

The boundary conditions on the mesh movement and momentum equations are quite extensive. It is recommended that APREPRO be used to simplify the list. In any case, this problem has many "solid walls" at which velocity and mesh boundary conditions must be applied. The boundary condition section of the input file looks like (the most noteworthy boundary conditions are shown in italic type):

```
Boundary Condition Specifications
 ---
Number of BC            =  -1
BC                              = PLANE SS  1   0.   1.   0    -0.0005
BC                              = PLANE SS  3   0.   1.   0.    0.0
BC                              = PLANE SS  4   0.   1.   0.    0.0
BC                              = PLANE SS  5   1.   0.   0.   -0.003
BC                              = PLANE SS  8   1.   0.   0.   -0.000125
BC                              = PLANE SS  9   1.   0.   0.    0.000125
BC                              = PLANE SS 10   0.   1.   0    -0.00152
```

```
BC =   DX          NS     11        0.0
BC =   DX          NS    300        0.0
BC =   DY          NS    300        0.0
BC =    U          NS      1        0.0
BC =    V          NS      1        0.0
BC =    U          NS     12        0.0
BC =    V          NS     12        0.0
BC =    U          NS      4        0.133
BC =    V          NS      4        0.0
BC =  VVARY        SS     10     0.000125    0.21
BC =    U          NS     10        0.0
BC =    U          NS    100        0.0
BC =    V          NS    100        0.0
BC =    U          NS    200        0.0
BC =    V          NS    200        0.0
BC =    U          NS    300        0.0
BC =    V          NS    300        0.0
BC = VELO_NORMAL SS 3 0.
BC = KINEMATIC     SS      6        0.
BC = CAPILLARY     SS      6        0.065    0.0  0.
BC = SURFTANG      NS    400       -1.0      0.0     0.0     0.065
BC = KINEMATIC     SS      2        0.
BC = CAPILLARY     SS      2        0.065  -3675.0  0.
BC = CA            NS    100        1.000     0.0   -1.0     0.0
BC = CA            NS    200        0.600     0.0    1.0     0.0

END OF BC

BC = VELO_NORMAL SS 6 0.
BC = VELO_NORMAL SS 2 0.
```

First, the DX and DY boundary conditions are used to "freeze" key points. At the downstream static separation line (node set 300) the mesh must be pinned using these cards. If it is not pinned then a contact angle position would have to be specified. In order to help maintain grid integrity, two mesh lines in the middle of the domain emanating from the tip of the feed slot were also pinned in the X-direction so as to isolate the mesh movement effects of the upstream and downstream menisci, i.e. node set 11. The VVARY card is used to specify the velocity field function at the inflow plane. The user-defined subroutine that is appropriate here is in `user_bc.c`. In that file there is a subroutine called `velo_vary_fnc`, in which the functional form

```
f = -a2*(1. - (x1/a1)*(x1/a1))
```

is input under the VVARY_BC option. The first parameter on the VVARY card corresponds to *a1* and the second to *a2*. Clearly, *a2* represents the maximum speed of the parabolic inflow velocity profile and *a1* is half the slot width. The GD_PARAB boundary condition (see Table 4.5 and associated examples) is an alternative way of applying this condition and avoids the use of user-defined subroutines. Finally, the boundary conditions associated with the upstream and downstream menisci are instructive to discuss. First, the VELO_NORMAL boundary conditions on side sets 2 and 6 (corresponding to the upstream and downstream menisci) are not being used; however, they were used during the start-up process, as is discussed below. The boundary conditions being applied to these side sets are the CAPILLARY (Table 4.1, Table 6.2) condition, which balances the viscous stress in the liquid with the capillary pressure (surface tension forces) and the backpres-

sure, if applied. Together with those cards, the KINEMATIC cards (Table 4.1, Table 6.2) are used to distinguish the movement of the free surface and are basically used to put boundary conditions on mesh motion. Perhaps the most critical cards, however, pertain to the contact angle conditions, i.e., CA cards (Table 4.1, Table 6.2), which are used to set the static and dynamic contact angles on the upstream meniscus to 145 degrees. Perfect slip is applied in the region near the dynamic contact line over side set 3 using the VELO_NORMAL card, which allows no shear stress along that side set, but enforces the no-penetration condition. Finally, the SURFTANG card (Table 4.1) is applied to the end point of the downstream free surface where it exits the flow domain. The need for this card arises from applying the surface divergence theorem to the surface term of the weak form of the momentum equation (see Chapter 6).

Four continuation steps were necessary to achieve the steady operating state pictured in Figure 10:

1) First, the boundary condition list was edited to include the VELO_NORMAL cards for all three side sets (2, 3, and 6) and to exclude the KINEMATIC, CAPILLARY, CA, and SURF_TANG cards. This situation mimics a closely associated closed flow with slippery free surfaces. With a zero initial guess (see Section 4.1) it required three Newton iterations to achieve a closed flow solution.

2) The file `soln.dat` was copied into `contin.dat`, the names of which were chosen in the input deck (Section 4.1), and the "`Initial Guess`" card was set to "read". The VELO_NORMAL card for the downstream meniscus (side set 6) was moved out of the active list (i.e., above the END OF BC card) and replaced with the associated CAPILLARY and KINEMATIC cards and the SURFTANG card. Twenty Newton iterations were taken with a relaxation of 0.05 (using the -r option on the command line, as discussed in Section 3.3).

3) Again, the file `soln.dat` was copied into `contin.dat`, and three more iterations with full Newton (i.e., a relaxation factor of 1.0) were needed to converge.

4) Finally, the VELO_NORMAL card for the upstream meniscus was replaced with the associated CAPILLARY and KINEMATIC cards as well as the two CA cards. Ten Newton iterations were taken with a 0.1 relaxation factor, followed by four more at full Newton.

Typical results (deformed mesh, streamlines and pressure contours) are presented in Figure 10 for this specific case study. Process conditions are listed in the figure caption. As expected, pressure in the feed channel drops linearly. It is informative that subambient pressure zones exist in both the upstream and downstream regions, in addition to that near the static and dynamic contact lines. Negative pressure is detrimental in coating flows because it can potentially cause cavitation of the coating liquid and is believed to be responsible for the ribbing instability coating defect. The qualitative flow features as predicted here are consistent with that

documented by Sartor (1990) using the spine technique and structured meshes.

7.4 Dip Coating and Drying of Porous Films

This problem illustrates the capabilities within GOMA to solve problems in both solid mechanics and porous media. During production of sol-gel coatings, a substrate is withdrawn from a bath of solvent; the coating solution solidifies into a porous network (a gel) filled with solvent, and the porous gel dries. In this problem, we predict drying of a porous gel film in a steady-state dip-coating process. The geometry of the film is shown in Figure 11; the coating is withdrawn vertically from the solvent bath, so it enters the domain as a wet film. There is a short entry length in which the surface of the coating is still wet. Then the coating dries by evaporation into a low-humidity gas (as described by a mass transfer coefficient). Eventually the nearly dry coating leaves through the upper boundary of the domain.

This problem contains several unusual novel implementations of GOMA, which we will not explain in detail here. However, we will list them with some brief explanation:

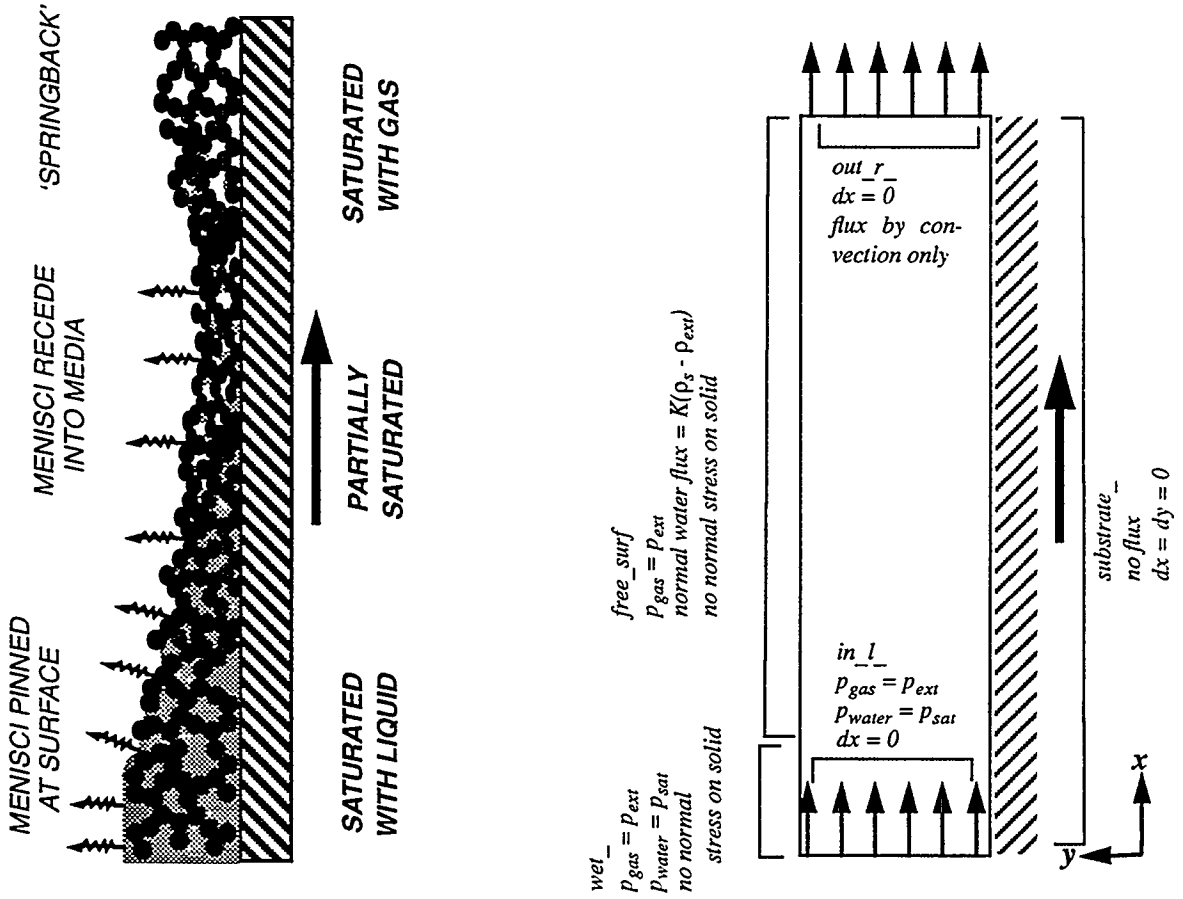


Figure 11 Geometry of domain for predicting drying of dip-coated porous sol-gel films

1) Porous medium deforms as a Neo-Hookean Solid.

2) Lagrangian solid mechanics cast in a convected frame of reference to account for steady-state motion of the porous solid.

3) Solid network stress in porous medium is coupled to the capillary stress in the liquid in the pores by the effective stress principle. The pressure stress is equal to the capillary pressure times the saturation.

4) Two-phase transport of air and water in a partially-saturated porous medium. This uses species transport equations to solve for the transport of air and water in both liquid and gas phases. The concentration variables in GOMA are actually the pressure in the liquid phase and pressure in the gas phase when solving for transport in partially saturated porous media.

5) Properties of the porous medium change due to compression or dilation of the porous medium and due to changes in saturation. The changes in properties are predicted from a simple pore model which treats the pores as a bundle of capillary tubes with a distribution of pore sizes (in which all pore sizes over a range occupy the same volume fraction of the pore-space)

6) Equations rescaled using APREPRO so that the primitive variables are values of order one.

In all of the files used by GOMA, a definitions file, *film.defs*, is included to automate changes in properties, geometry, boundary conditions, etc. This file also sets up the scaling factors to change the units on the variables:

```
$ fill.defs - this file defines geometry, physical parameters, etc. for
$         for inclusion in input files for GOMA and FASTQ
$ 1-D Drying Porous coating
$ Richard Cairncross (1511) 9/18/95
$
$ Definitions for FASTQ and GOMA
$
$ Define scaling of units (all properties are input in MKS, but we need
$   to adapt them so our values are nicer).  Multiply all values in MKS
$   times L, K, S, T, and M corresponding to its units, and you get
$   values in the scaled units  (e.g. if a length is 10.5 meters and L is 1e4 then
$   the scaled length is 10.5 * 1e4 = 105000 in units of 100 microns).
$   Note to convert back, just divide by these factors.
$
$   length:    L = {L = 1e6}  /meter
$   mass:      K = {K = 1e3}  / kg
$   time:      S = {S = 10}     / s
$   temperature: T = {T = 1}  /kelvin
$   moles:     M = {M = K}      /k-mole
$   so pressure unit becomes {K / L / S^2} current units / N/m^2
$
$   To convert output to MKS, multiply output by following values:
$     velocity * {S/L}for m/s
$     concentration * {L^3/K}for kg/m^3
$     pressure* {L * S^2/K}for Pascals  (kg/m/s^2)
$     distance * {1/L}for m
$     time* {1/S} for s
$
$ Geometry
$  initial coating thickness: hcoat  = {hcoat  = 0.000001 * L}
$  length of domain:          length = {length = 0.001 * L}
```

```
$
$ Number of Elements
$  {nex = 120}
$  {ney = 3}
$  {stretchx = 1.015}
$  {stretchy = 0.8}
$
$ Region
$  coating = {coating = 1}
$
$ Node set names...
$ origin - fixed point:       {origin   = 1000}
$
$ right outflow plane:        {out_r_ns = 20}
$ left  inflow plane:         {in_l_ns = 40}
$ free surface:               {free_surf_ns = 30}
$ free surface:               {wet_ns = 60}
$ substrate surface:          {substrate_ns = 10}
$
$ Side set names...
$
$ right symmetry plane:       {out_r_ss = 2}
$ left  symmetry plane:       {in_l_ss = 4}
$ free surface:               {free_surf_ss = 3}
$ free surface:               {wet_ss = 6}
$ substrate surface:          {substrate_ss = 1}
$
$ flag for automatic counting: auto_count = {auto_count = -1}
$
$ Define Species Numbers
$   air or gas phase:      AIR   =  {AIR = 1}
$   water or liquid phase: WATER = {WATER = 0}
$
$ some physical properties
$ liquid viscosity:        viscosity   = {viscosity = 0.001 * K / L / S}
$ gas    viscosity:        gas_visc    = {gas_visc  = 0.000001 * K / L / S}
$ Gas Diffusion:           diffusivity = {diffusivity = 1e-5 * L * L / S}
$ fluid density:       density     = {density = 1000 * K / L^3}
$ Ambient Pressure (1 ATM): Vap_Pres = {Amb_Pres = 1.0133e5 * K / L / S^2}
$ Vapor Pressure (0.1 ATM): Vap_Pres = {Vap_Pres = 1.0133e4 * K / L / S^2}
$ Relative humidity:    Rel_Humid   = {Rel_Humid = 0.03}
$ Initial Relative humidity: Rel_Humid0 = {Rel_Humid0 = 0.999}
$ Operating Temperature:   Temp= {Temp = 300 * T}
$ Elastic Shear Modulus:   Modulus  = {Modulus = 1e6 * K / L / S^2}
$   (0.2 to 0.5 MPa)
$ Poisson Ratio:           Poisson      = {Poisson = 0.4}
$ Initial Porosity:        Porosity0    = {Porosity0 = 0.5}
$ Gas Law Constant:        Rgas  = {Rgas = 8.314e3 * K * L^2 / S^2 / T / M}
$ Mass transfer coefficient: MTC= {MTC = 0.03/2 * L / S}
$ Permiability of gel:     Perm         = {Perm = 1e-18 * L^2}
$
$ Need concentration of water in the gas at saturation
$ Gas Concentration:     gas_conc = {gas_conc = Vap_Pres * 18 * K / M / Rgas / Temp}
$
$ For initial pressure in liquid phase, we need to calculate the
$  needed capillary pressure to be in equilibrium with the relative humidity
$  in the gas (via Kelvin Equation)
$ Liquid Pressure: Pl0 = {Pl0 = Amb_Pres+Rgas*Temp*density/(18*K/M)*ln(Rel_Humid0)}
```

This file is included in all the input files and run through APREPRO to make all the substitutions in those files. The *film.fas* file defines the geometry of the coating, with the definitions from film.defs:

```
$ film.fas  - a file defining the geometry and FEM setup for use
$             with FASTQ and the code GOMA - this file must be
$             preprocessed with APREPRO
{include("film.defs")}
$
 TITLE Dry_Coating

 POINT     1      0.0000000E+00      0.0000000E+00
 POINT     2      {length}            0.0000000E+00
 POINT     3      {length}            {hcoat}
```

```
POINT     4      0.0000000E+00      {hcoat}
POINT     5      {-length / 10}     0.0000000E+00
POINT     6      {-length / 10}     {hcoat}

LINE      1   STR     1     2     0     {nex}  {stretchx}
LINE      2   STR     2     3     0     {ney}  {stretchy}
LINE      3   STR     4     3     0     {nex}  {stretchx}
LINE      4   STR     1     4     0     {ney}  1.0000

LINE      5   STR     4     6     0     {10}  1.3
LINE      6   STR     5     6     0     {ney}  1.0000
LINE      7   STR     1     5     0     {10}  1.3

REGION     {coating}  {coating}    -1    -2    -3   -4
REGION     {coating * 10}  {coating}    -4    -5    -6   -7
SCHEME    {coating}  M
SCHEME    {coating * 10}  M
BODY      {coating} {coating * 10}
POINBC    {origin}    1
POINBC    200     2
POINBC    300     3
POINBC    400     4
NODEBC      {substrate_ns}     1  7
NODEBC      {free_surf_ns}     3
NODEBC      {wet_ns}     5
NODEBC      {out_r_ns}     2
NODEBC      {in_l_ns}     6
SIDEBC      {substrate_ss}     1   7
SIDEBC      {free_surf_ss}     3
SIDEBC      {wet_ss}     5
SIDEBC      {out_r_ss}     2
SIDEBC      {in_l_ss}     6
RENUM
EXIT
```

Before using this file in FASTQ, it must be preprocessed by APREPRO, i.e. "`aprepro film.fas film.fast`" then "`fastq film.fastq`". From fastq a GENESIS file should be created and converted to an EXODUS II file.

The properties of the porous medium, and the water and air within the medium, are listed in the *porous.mat* file (some of the unused parts of the file are left out for brevity):

```
$ porous.mat - file which defines the models and physical properties of
$              materials for use in GOMA
$
 {include("film.defs")}
$
Material Data File for a Drying Porous Gel Coating

---Physical Properties
Density= CONSTANT {density}

---Mechanical Properties and Constitutive Equations
Solid Constitutive Equation = HOOKEAN_PSTRAIN
Convective Lagrangian Velocity  = CONSTANT      {3e-5 * L / S}  0.  0.
Lame MU = POWER_LAW {Modulus}  {Porosity0} {3.}
Lame LAMBDA= POWER_LAW {2 * Modulus * Poisson / (1 - 2 * Poisson)}  {Porosity0} {3.} {0.0001}
{3.}
Stress Free Solvent Vol Frac= CONSTANT{Porosity0}

---Microstructure Properties
Media Type          = POROUS_PART_SAT
Porosity    = DEFORM{Porosity0}
Permeability= PSD_VOL{Porosity0}  {1e-8 * L}  {0.01}  1.
Capillary Network Stress = PARTIALLY_WETTING
Rel Gas Permeability= SUM_TO_ONE    {gas_visc}
Rel Liq Permeability= PSD_VOL   {viscosity}
Saturation          = PSD_VOL   {0.03 * K / S^2}  {0}

---Species Properties
Diffusion Constitutive Equation = DARCY_FICKIAN
Diffusivity  = CONSTANT  {WATER}  {diffusivity}
```

```
Latent Heat Vaporization= CONSTANT  {WATER}  0.
Latent Heat Fusion= CONSTANT  {WATER}  0.
Vapor Pressure  = KELVIN    {WATER}  {Vap_Pres} {density} {18. * K / M} {Rgas} {Temp}
Species Volume Expansion        = CONSTANT  {WATER}  1.
Reference Concentration         = CONSTANT  {WATER}  0.

Diffusivity  = CONSTANT  {AIR}   {diffusivity}
Latent Heat Vaporization= CONSTANT  {AIR}0.
Latent Heat Fusion= CONSTANT  {AIR}0.
Vapor Pressure  = IDEAL_GAS {AIR}   {28. * K / M} {Rgas} {Temp}
Species Volume Expansion        = CONSTANT  {AIR}  1.
Reference Concentration         = CONSTANT  {AIR}   0.


----Source Terms
Navier-Stokes Source= CONSTANT{-9.8 * density * K / L^2 / S^2}  0.   0.
Solid Body Source= CONSTANT0.  0.  0.
Mass Source= CONSTANT0.
Heat Source= CONSTANT0.
Species Source= CONSTANT  {WATER}   0.
Species Source= CONSTANT  {AIR}   0.
```

Solution of this problem requires continuation from an initial solution. It was found easiest to get an initial solution for a wet coating that does not dry (i.e. the relative humidity in the overlying gas is equal to the initial relative humidity, Rel_Humid = Rel_Humid0 in `film.defs`). The input file for getting the initial solution is `input.initial` and contains slightly different boundary conditions and initialization setup than the main input file (only the parts of this file that are different from the standard input file are shown here):

```
$ input.initial - an input file for GOMA which sets up an initial guess
$                 for the drying dip-coated porous gel coating problem in
$                 which the coating does not dry because the overlying gas
$                 is nearly saturated with water
 {include("film.defs")}
$
 FEM File Specifications
 ---
FEM file                        = film.exoII
Output EXODUS II file           = out.exoII
GUESS file                      = contin_in.dat
SOLN file                       = contin.dat
Write intermediate results      = no
 ---
Boundary Condition Specifications
 ---
Number of BC =    {auto_count}
## BC's specifying the liquid and gas pressures in the porous medium
  BC =    Y             NS     {free_surf_ns}   {AIR}      {Amb_Pres}
  BC =    Y             NS     {in_l_ns}        {AIR}      {Amb_Pres}
  BC =    Y             NS     {in_l_ns}        {WATER}    {P10}
  BC =    Y             NS     {wet_ns}        {WATER}    {P10}
## BC specifying the equilibrium between the liquid phase and the external gas (which is satu-
rated with water)
  BC =    POROUS_FLUX   SS     {free_surf_ss}   {WATER}    {MTC}   {Rel_Humid0 * gas_conc}
## BC which allows convection of water out of the domain
  BC =    POROUS_FLUX   SS     {out_r_ss}   {WATER}    {0}   {0}

##BC's on the porous solid - specifying adherence to the moving substrate
  BC =    DX      NS     {substrate_ns}        0.
  BC =    DY      NS     {substrate_ns}        0.
  BC =    DX      NS     {out_r_ns}            0.
  BC =    DX      NS     {in_l_ns}             0.
#########
END OF BC
#########
```

This input file should be run using the APREPRO option in goma - `goma -a -i input.initial`, and should converge in a few iterations:

```
spnode03(11)% goma -a -i input.initial
```

```
READING FROM ALTERNATE INPUT FILE: input.initial
Total of 2 commands found
system: aprepro input.initial tmp.input
reading input from temporary file tmp.input
Copyright (c) 1993-1995  Sandia National Laboratories
                         PRS, PAS, RRR, KSC, & RAC

                "FIPSUMDA, a pig by any other name..."
Memory Usage:
        a[]: 97751 entries * 8 bytes/entry = 782008 bytes
        x[]: 2620 entries * 8 bytes/entry = 20960 bytes
Setting variable 9 to 0.5
Setting variable 4 to -0.37306
Setting variable 4 to 1.0133

  ToD     itn   L_oo    L_1     L_2      L_oo    L_1     L_2      asm/slv (sec)
-------- --- ------- ------- -------  ------- ------- -------  -----------------
09:30:35 [0] 1.0e-03 1.2e-01 1.1e-02  6.2e-03 3.3e+00 1.2e-01  5.11e+00/3.89e+00
09:30:44 [1] 2.3e-05 5.9e-03 2.8e-04  1.4e-04 6.5e-02 2.3e-03  5.08e+00/8.40e-01
09:30:50 [2] 9.4e-09 1.5e-06 9.8e-08  5.6e-08 2.9e-05 1.0e-06  5.08e+00/7.00e-01
09:30:56 [3] 1.8e-15 4.3e-13 2.0e-14  2.1e-14 9.9e-12 3.4e-13  5.09e+00/7.10e-01

-done
```

The solution from this run should be written to a `contin.dat` file for continuation with GOMA. To get solutions with lower relative humidity in the overlying gas, several continuation steps are needed where the Rel_Humid in film.defs is reduced (starting from 0.9, 0.8, 0.5 . . .). However for the first reduction in the relative humidity (and any big steps later), relaxation of Newton's method is needed. The standard input file, input, for these calculations is:

```
$ input - standard input file for drying of a dip-coated porous sol-gel
$         coating with a low humidity external atmosphere
 {include("film.defs")}
$
 FEM File Specifications
 ---
FEM file                          = film.exoII
Output EXODUS II file             = out.exoII
GUESS file                        = contin.dat
SOLN file                         = soln.dat
Write intermediate results        = no
 ---
General Specifications
 ---
Number of processors              = 1
Output Level                      = 0
Debug                             = 0
Initial Guess                     = read
 ---
Time Integration Specifications
 ---
Time integration                  = steady
 ---
Solver Specifications
 ---
Solution Algorithm                = lu
Preconditioner                    = poly
Polynomial                        = LS,1
Size of Krylov subspace           = 64
Orthogonalization                 = classical
Maximum Linear Solve Iterations   = 1000
Number of Newton Iterations       = 16
Newton correction factor          = 1.0
Normalized Residual Tolerance     = 1e-12
Residual Ratio Tolerance          = 1e-2
 ---
Boundary Condition Specifications
 ---
Number of BC =    {auto_count}
## BC's specifying the liquid and gas pressures in the porous medium
BC =    Y             NS     {free_surf_ns}   {AIR}      {Amb_Pres}
BC =    Y             NS     {in_l_ns}        {AIR}      {Amb_Pres}
```

```
BC =    Y             NS     {in_l_ns}        {WATER}    {P10}
BC =    Y             NS     {wet_ns}         {WATER}    {P10}
## BC specifying the flux of water into the overlying gas phase
BC =    POROUS_FLUX   SS     {free_surf_ss}   {WATER}    {MTC}   {Rel_Humid * gas_conc}
## BC which allows convection of water out of the domain
BC =    POROUS_FLUX   SS     {out_r_ss}       {WATER}    {0}     {0}

##BC's on the porous solid - specifying adherence to the moving substrate
BC =    DX      NS     {substrate_ns}        0.
BC =    DY      NS     {substrate_ns}        0.
BC =    DX      NS     {out_r_ns}            0.
BC =    DX      NS     {in_l_ns}             0.
#########
END OF BC
#########

----
Problem Description
---

Number of Materials =  {auto_count}

 MAT =   porous    {coating}

  Coordinate System = CARTESIAN
  Element Mapping   = isoparametric
  Mesh Motion = LAGRANGIAN
   Number of bulk species = 2

 Number of EQ  =  {auto_count}
 EQ = species_bulk  Q1    Y      Q1       1.  1.    1.   1.   0.
 EQ = mesh1      Q1      D1      Q1       0.  0.    0.   1.   0.  0.
 EQ = mesh2      Q1      D2      Q1       0.  0.    0.   1.   0.  0.
 EQ = continuity Q1      P       Q1  1.                      1.
#########
END OF EQ
#########
                                  div  ms  adv   bnd  dif  src   porous
##########
END OF MAT
##########
Post Processing Specifications
Third Invarient of Strain = yes
Time Derivatives = yes
Mesh Stress Tensor = yes
Porous Saturation = yes
Bulk density of species in porous media = yes
Gas concentration of species in porous media = yes
Gas phase convection vectors in porous media = yes
Liquid phase convection vectors in porous media = yes
Lagrangian Convection = yes
```

Then, setting the appropriate value of Rel_Humid in `film.defs`. This input file is run using relaxation of Newton's method and APREPRO, `goma -r 0.1 -a`:

```
spnode03(14)% goma -a -r 0.1
Total of 2 commands found
system: aprepro input tmp.input
reading input from temporary file tmp.input
Copyright (c) 1993-1995  Sandia National Laboratories
                         PRS, PAS, RRR, KSC, & RAC

                "FIPSUMDA, a pig by any other name..."
Memory Usage:
        a[]: 97751 entries * 8 bytes/entry = 782008 bytes
        x[]: 2620 entries * 8 bytes/entry = 20960 bytes

  ToD     itn   L_oo     L_1     L_2      L_oo    L_1     L_2      asm/slv (sec)
-------- --- ------- ------- -------  ------- ------- -------  -----------------
09:48:50 [0] 1.1e-04 1.1e-02 1.0e-03  3.9e+01 1.2e+04 5.8e+02  5.13e+00/4.25e+00
09:49:00 [1] 1.5e-03 2.4e-01 1.3e-02  4.7e+01 1.4e+04 6.8e+02  5.13e+00/8.60e-01
09:49:06 [2] 2.0e-03 3.7e-01 1.9e-02  5.2e+01 1.5e+04 7.5e+02  5.11e+00/7.20e-01
09:49:12 [3] 2.2e-03 4.5e-01 2.3e-02  5.5e+01 1.5e+04 7.9e+02  5.16e+00/7.30e-01
09:49:18 [4] 2.1e-03 4.8e-01 2.4e-02  5.7e+01 1.6e+04 8.1e+02  5.12e+00/7.20e-01
```

```
09:49:24 [5] 2.0e-03 4.9e-01 2.4e-02  5.8e+01 1.6e+04 8.1e+02  5.11e+00/7.20e-01
09:49:30 [6] 1.9e-03 4.9e-01 2.4e-02  5.8e+01 1.5e+04 8.1e+02  5.13e+00/7.20e-01
09:49:36 [7] 1.9e-03 4.8e-01 2.3e-02  5.7e+01 1.5e+04 7.9e+02  5.11e+00/7.30e-01
09:49:42 [8] 1.8e-03 4.6e-01 2.2e-02  5.6e+01 1.5e+04 7.7e+02  5.13e+00/7.30e-01
09:49:48 [9] 1.7e-03 4.3e-01 2.1e-02  5.5e+01 1.4e+04 7.5e+02  5.12e+00/7.20e-01
09:49:54 10] 1.7e-03 4.1e-01 2.0e-02  5.3e+01 1.3e+04 7.2e+02  5.11e+00/7.10e-01
09:50:00 11] 1.6e-03 3.8e-01 1.8e-02  5.1e+01 1.3e+04 6.9e+02  5.10e+00/7.20e-01
09:50:05 12] 1.5e-03 3.5e-01 1.7e-02  4.9e+01 1.2e+04 6.6e+02  5.09e+00/7.20e-01
09:50:11 13] 1.4e-03 3.3e-01 1.6e-02  4.7e+01 1.1e+04 6.2e+02  5.09e+00/7.00e-01
09:50:17 14] 1.3e-03 3.0e-01 1.5e-02  4.5e+01 1.1e+04 5.9e+02  5.10e+00/7.30e-01
09:50:23 15] 1.2e-03 2.8e-01 1.3e-02  4.2e+01 1.0e+04 5.5e+02  5.10e+00/7.20e-01

-done
```

Now, the solution from this run can be continued with full Newton's method until it converges (but first the solution file should be copied to the continuation file, `cp soln.dat contin.dat`), `goma -a`:

```
spnode03(15)% cp soln.dat contin.dat
spnode03(16)% goma -a
Total of 1 commands found
system: aprepro input tmp.input
reading input from temporary file tmp.input
Copyright (c) 1993-1995  Sandia National Laboratories
                         PRS, PAS, RRR, KSC, & RAC

                "FIPSUMDA, a pig by any other name..."
Memory Usage:
        a[]: 97751 entries * 8 bytes/entry = 782008 bytes
        x[]: 2620 entries * 8 bytes/entry = 20960 bytes

  ToD     itn   L_oo    L_1     L_2      L_oo    L_1     L_2      asm/slv (sec)
-------- --- ------- ------- -------  ------- ------- -------  -----------------
09:51:59 [0] 1.1e-03 2.6e-01 1.2e-02  4.0e+01 9.4e+03 5.2e+02  5.13e+00/1.45e+01
09:52:19 [1] 5.3e-03 4.4e-01 2.9e-02  1.9e+01 2.5e+03 1.8e+02  5.11e+00/1.79e+00
09:52:26 [2] 1.3e-03 3.8e-02 4.0e-03  2.6e+00 2.8e+02 2.0e+01  5.12e+00/1.32e+00
09:52:33 [3] 6.7e-05 8.5e-04 1.1e-04  3.4e-01 4.6e+01 3.0e+00  5.15e+00/1.32e+00
09:52:39 [4] 3.2e-05 8.7e-05 3.5e-05  2.1e-01 3.1e+01 2.0e+00  5.14e+00/1.32e+00
09:52:46 [5] 3.9e-08 3.5e-06 2.2e-07  3.7e-04 3.2e-02 2.3e-03  5.13e+00/1.32e+00
09:52:53 [6] 2.7e-11 1.0e-09 1.0e-10  7.6e-09 2.2e-06 1.1e-07  5.13e+00/1.32e+00
09:52:59 [7] 1.0e-14 1.5e-12 7.2e-14  7.0e-12 1.6e-09 8.5e-11  5.13e+00/1.32e+00

-done
```

Further continuation to lower relative humidities is done in the same manner, and relaxation normally is not needed unless big steps in relative humidity are taken or if other parameters are changed.

Standard results from this problem are shown in Figure 12.

7.5 A Simple Mold Filling Problem

Mold filling as a form of polymer processing is a complex phenomenon in which a viscous fluid at high pressure is injected into a mold. The mold geometry is often complex, and the rheology of the fluid is generally nonlinear. In this example problem, the simplest mold filling problem is solved: the transient filling of a straight channel.

When ALE techniques are used on moving boundary problems such as mold filling, care must be taken at the dynamic contact line. The dynamic contact line (DCL) is the position where the front advances or recedes with respect to a stationary wall or a wall moving at a different speed. This

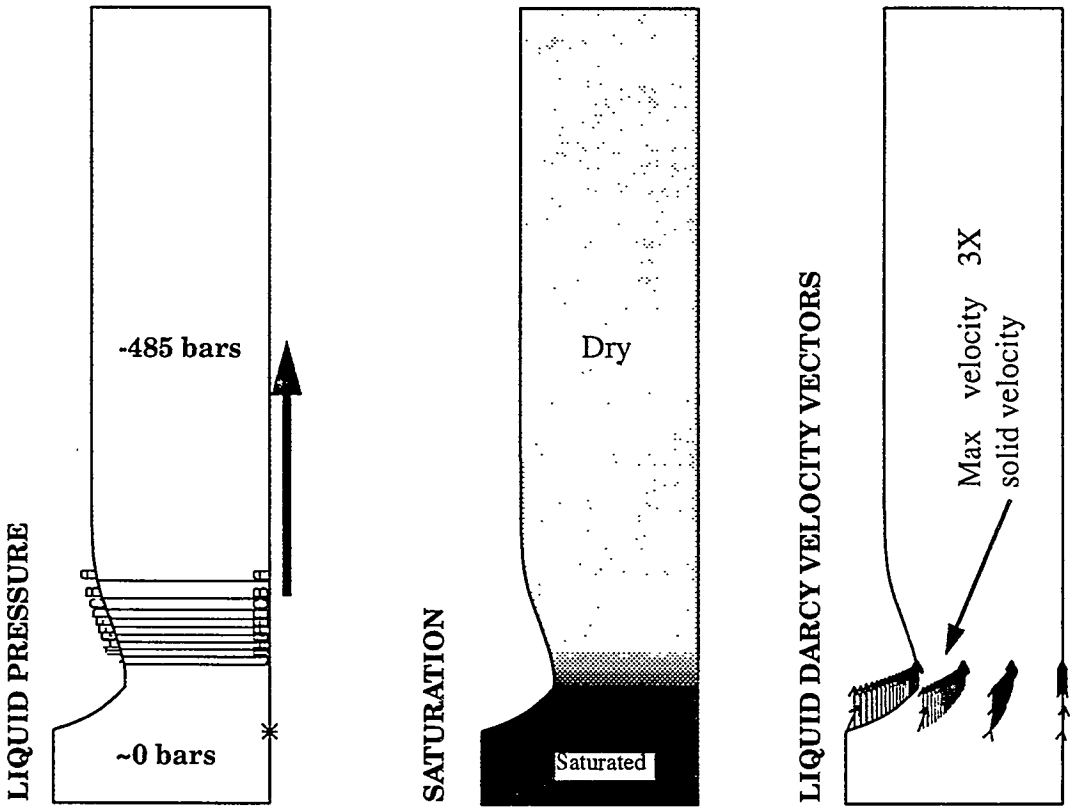


Figure 12 Standard Results for Drying of a dip-coated porous sol-gel coating for a relative humidity of 3%. The horizontal axis here is expanded 300x.

difference in velocity causes a singularity in the governing equations, which must be alleviated in some way so as to obtain a plausible solution. In this example, the singularity at the DCL is alleviated by introducing a position-dependent slip velocity that allows the DCL to advance with the mesh, but decays rapidly to a no-slip boundary condition further along the wall.

Figure 13 is a schematic of the boundary conditions for mold filling. In addition to the boundary conditions shown, a slip law to the solid boundary is also applied. The slip law decays exponentially from the dynamic contact line and has the form:

$$\left(v - \beta e^{-\alpha x_{dcl}} \dot{x}\right) \bullet t = 0$$

$$v \bullet n = 0$$

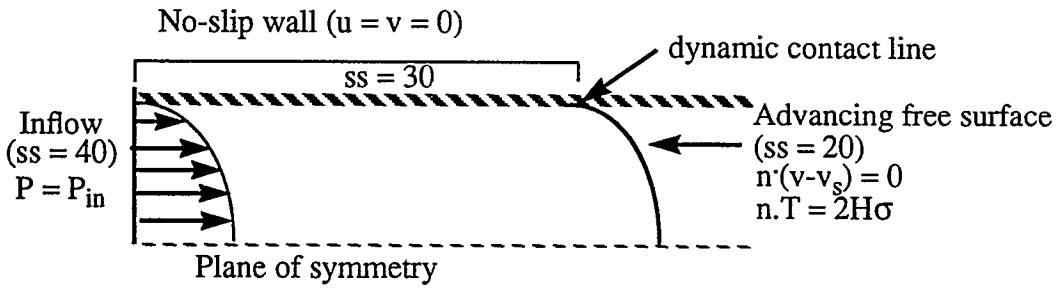


Figure 13 Boundary conditions for a mold filling problem

In this equation v is the fluid velocity, $\dot{x}$ is the mesh velocity, β is a user-specified constant that varies from 0 to 1(it is used to turn the slip on and off), α is another user-specified constant that describes how quickly the tangential velocity will decay from slip to no slip, x_{dcl} indicates the distance from the current position to the dynamic contact line, **t** is the unit tangent to the free surface and **n** is the unit normal to the free surface.

The boundary condition section of the input file is most relevant in this example, as it provides the correct combination of conditions that need to be applied in and around a moving contact line:

```
Number of BC = -1
BC = PLANE SS 10 0. 1. 0.  0.
BC = PLANE SS 30 0. 1. 0. -1.e-4
BC = PLANE SS 40 1. 0. 0.  0.
BC = V     NS     1       0.
BC = V     NS     4       0.
BC = UVARY     SS     40       0.02 .0001
BC = VELO_NORMAL   SS  30   0.
BC = VELO_TANGENT  SS  30 300 0.  1.  200000.
BC = KINEMATIC SS 20 0.
BC = CAPILLARY SS 20 0.02 0. .0
BC = CA    NS    300  2.34   0. -1. 0.
END OF BC
```

Notice that the KINEMATIC condition is applied to the free surface, together with the CAPILLARY condition (side set 20). Where the free surface meets the solid wall (side set 30 at node set 300), the contact angle (CA) card is applied which is used to keep the contact angle at 2.34 radians, or 134 degrees. As it turns out, this condition is used to replace the kinematic condition at the contact line (done automatically). The VELO_NORMAL condition keeps the wall impenetrable and the VELO_TANGENT condition is used to apply the position-dependent slip, as described above (see the card description in Table 4.1).

The start-up of this problem is trivial. With an initial guess of zero, the problem is run transient with a variable time step scheme, viz.

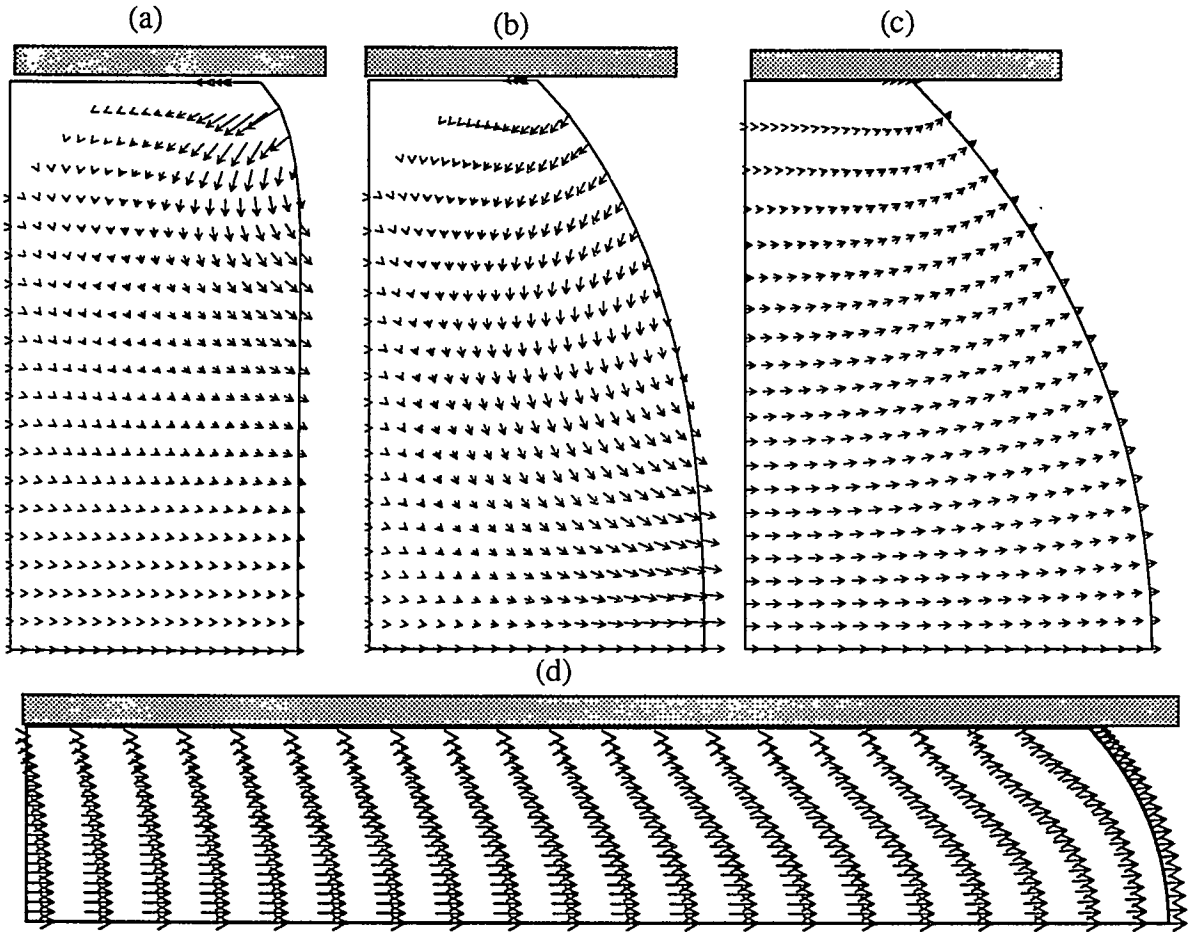


Figure 14 Several time steps of the simple mold filling example. (a) - (c) illustrate the vector velocity field at three early time planes, and (d) at a later time plane. The vector scale is not constant, and was increased for frames (c) and (d) for illustration.

```
------------------------------------------------------------
                Time Integration Specifications
------------------------------------------------------------
Time integration                  = transient
delta_t                           = 5.0e-6
Maximum number of time steps      = 250
Maximum time                      = 10.0e+0
Minimum time step                 = 1.0e-12
Time step parameter               = 0.
#Time step error                  = err=.005  mesh=1  v=1  T=0   C=0  p=0
Time step error                   = .008  1   1   0   0   0
Printing Frequency                = 1
```

Some sample results are shown in Figure 14. The slip region is evident near the dynamic contact line, where a significant slip velocity is visible. The computational mesh is structured the topology remains constant throughout the simulation. The sequence shown are samples from a series of

30 time steps taken at variable intervals.

7.6 Deformable Blade Coating onto a Deformable Substrate

Flexible blade coating flow is the workhorse operation in the paper industry for applying a uniform layer of pigmented or functional coating on the surface of a paper substrate in order to improve its printability or appearance. In such a process coating liquid interacts with two solid materials, namely the flexible blade and the deformable substrate. More specifically, the blade bends and the substrate deforms in response to the hydrodynamic pressure developed during coating in the gap formed between the substrate and the blade tip. The elastohydrodynamics of blade coating, which focuses on the interaction between hydrodynamic traction and the resultant elastic forces along the blade, was analyzed by Pranckh and Scriven (1988) using the theory of thin inextensional shells to describe the blade bending behavior. In their analysis Pranckh and Scriven also took into account of substrate deformation with a one-dimensional spring model. In these analyses liquid-solid interactions were approximated using simplified theories under various assumptions. The two resultant governing equations for locating the blade position, which account for the normal and tangential force balances along the blade and describe the local curvature and tangential stress, are closely coupled second- and first-order ordinary differential equations, respectively, with respect to blade position. Solving such complex governing equations for blade position, which are in turn coupled with the flow problem involving free surfaces, is by itself a very challenging task.

In this example we employ a coherent or unified finite-element analysis framework and analyze the blade coating process in a natural way. We divided into subdomains the blade and substrate solid regions in the same way we do the liquid flow region. We solve the Navier-Stokes equations within each subdomain of the liquid flow region (Eq. (6.6)) and the equilibrium (or divergent-free) stress equations in the solid region (Eq. (6.9)). We employ a neo-Hookean stress-strain constitutive model (Eq. (6.10)) to describe bending of the blade and deformation of the substrate so that approximations regarding the blade and the substrate are not required. Balancing tractions along the liquid-solid boundaries couples the fluid-mechanics and solid-mechanics problems in a natural way. The free boundaries at air-liquid and liquid-solid interfaces are tracked using novel mesh-deformation algorithm, which treats the finite-element mesh as a neo-Hookean solid. In the example here, which is intended for demonstrating GOMA's capability in handling liquid-solid interaction, we take the substrate as impermeable and ignore the liquid penetration phenomena. Results shown here were built upon our previous study on rigid blade coating flows (Chen et al. 1995).

The input deck to GOMA for generating results of a test problem presented below is as follows:

```
FEM File Specifications
FEM file       = blade.exoII
Output EXODUS II file = out.exoII
GUESS file = contin.dat
```

```
SOLN file      = soln.dat
Write intermediate results          = no
General Specifications
Number of processors              = 1
Output Level                      = 0
Debug                             = 0
Initial Guess                     = read
Pad Sparse Matrix                 = yes
Time integration                  = steady
Solver Specifications
Solution Algorithm                = lu
Preconditioner                    = poly
Polynomial                        = LS,1
Size of Krylov subspace           = 64
Orthogonalization                 = classical
Maximum Linear Solve Iterations   = 1000
Number of Newton Iterations       = 10
Newton correction factor          = 1.0
Normalized Residual Tolerance     = 1.0e-12
Residual Ratio Tolerance          = 1.0e-6

Boundary Condition Specifications
Number of BC                      = -1
BC      = PLANE SS  2   1.   0.   0. -0.009
BC      = PLANE SS 13   1.   0.   0. -0.009
BC      = PLANE SS  8   1.   0.   0.  0.025
BC      = PLANE SS  6   0.1763  1.0  0.  0.0202
BC      = PLANE SS 17   0.1763  1.0  0.  0.0202
BC            =    V        NS     20       0.0
BC      =    U        NS     40       0.0
BC      =    V        NS     40       0.0
BC      =    U        NS     50       0.0
BC      =    V        NS     50       0.0
BC      =    U        NS     80       1.0
BC      =    V        NS     80       0.0
BC      =    U        NS    300       0.0
BC      =    V        NS    300       0.0
BC      =    U        NS    400       1.0
BC      =    V        NS    400       0.0
BC      =   DX        NS    400       0.0
BC      =   DY        NS    400       0.0
BC      =   DX        NS    600       0.0
BC      =   DY        NS    600       0.0
BC      =   DX        NS    110       0.0
BC      =   DY        NS    110       0.0
BC      =   DY        NS    120       0.0

$$$BC   =      VELO_NORMAL SS 3 0.0 (Unused BC)
BC      = CAPILLARY   SS      3       0.05   0.0 0.0
BC      = KINEMATIC   SS      3       0.
BC      = SURFTANG    NS    200       1.0    0.0    0.0    0.05

$$$BC   =      VELO_NORMAL SS 7 0.0 (Unused BC)
BC      = KINEMATIC   SS      7       0.
BC      = CAPILLARY   SS      7       0.05    0.0   0.0
BC      = SURFTANG    NS    100      -0.1736  -0.9848   0.0    0.05

BC      = SOLID_FLUID SS 1 2 1
BC      = NO_SLIP     SS 1 2 1
```

```
BC      =  GD_CONST  SS  4  R_MESH1 0 VELOCITY1  0  0.0
BC      =  GD_CONST  SS  4  R_MESH2 0 VELOCITY2  0  0.0
BC      =  GD_CONST  SS  5  R_MESH1 0 VELOCITY1  0  0.0
BC      =  GD_CONST  SS  5  R_MESH2 0 VELOCITY2  0  0.0
BC      = FLUID_SOLID SS 4 3 1
BC      = FLUID_SOLID SS 5 3 1

######
END OF BC
######

Problem Description
Number of Materials  =  3
MAT     =   liquid    1
Coordinate System = CARTESIAN
Element Mapping   = isoparametric
Mesh Motion = ARBITRARY
Number of bulk species = 0
Number of EQ                    = 5
EQ      = momentum1  Q2      U1      Q2          0. 1000.  1.   1.0   0.0    0.
EQ      = momentum2  Q2      U2      Q2          0. 1000.  1.   1.0   1.0    0.
EQ      = continuity P1      P       P1      1.                        0
EQ      = mesh1      Q2      D1      Q2              0.   0.    0.   1.   0.  0.
EQ      = mesh2      Q2      D2      Q2              0.   0.    0.   1.   0.  0.
                                                    div  ms  adv  bnd  dif    src  porous
MAT     = substrate   2
Coordinate System = CARTESIAN
Element Mapping   = isoparametric
Mesh Motion = LAGRANGIAN
Number of bulk species = 0
Number of EQ                    = 2
EQ      = mesh1      Q2      D1      Q2   0.   0.  1.   1.      0.   0.
EQ      = mesh2      Q2      D2      Q2   0.   0.  1.   1.      0.   0.

                                                   div ms adv  bnd dif   src  porous

MAT     = blade   3
Coordinate System = CARTESIAN
Element Mapping   = isoparametric
Mesh Motion = LAGRANGIAN
Number of bulk species = 0
Number of EQ                    = 2
EQ      = mesh1      Q2      D1      Q2   0.   0.  1.   1.      0.   0.
EQ      = mesh2      Q2      D2      Q2   0.   0.  1.   1.      0.   0.
                                                   div ms adv  bnd dif   src  porous
```

There are three materials files required, namely `liquid.mat`, `substrate.mat` and `blade.mat`. Since the coating liquid was taken to be a constant density Newtonian liquid with a constant viscosity and surface tension, the material file for the coating liquid is straightforward. In the substrate material file, the two key lines for specifying the shear and bulk moduli of the substrate are as follows:

```
Lame MU                         = CONSTANT      600.
Lame LAMBDA                     = CONSTANT      600.
```

Similarly in the blade material file, blade shear and bulk moduli are specified as follows:

```
Lame MU                          = CONSTANT      2240.
Lame LAMBDA                      = CONSTANT      2450.
```

Figure 15 through Figure 17 show, respectively, finite element meshes, velocity field, pressure contour, streamlines, and pressure profile along the liquid/solid interfaces. The start-up procedure for this test problem is as follows:

1) A flow field solution was computed with all boundaries being fixed, i.e., both the substrate and the blade were set to be rigid, and the upstream and downstream free surfaces were made shear free with their position fixed. To make the boundaries shear stress free with respect to the liquid, the VELO_NORMAL card is applied and no tangential velocity is specified.

2) A flow field solution together with a downstream free surface position were computed with the downstream free surface (side set 7) being released. The solution from step 1 was used as an initial guess. Side set 7 is released by replacing a VELO_NORMAL card with the KINEMATIC/CAPILLARY/SURF_TANG cards, as shown above. The endpoint SURF_TANG card is required at capillary free boundary endpoints without overriding conditions.

3) A flow field solution and position of the upstream and downstream free surfaces were computed with the upstream free surface (side set 3) also being released. The solution from step 2 was used as an initial guess.

4) A flow field solution, positions of free surfaces, and position of liquid/substrate interface were computed with the substrate being set to be deformable (the substrate moduli was initially set to be very high and final solution was obtained via parameter continuation to the values shown above). The solution from step 3 was used as an initial guess. The distinguishing condition for the substrate is the "SOLID_FLUID" card. The NO_SLIP condition replaces any other cards that might be present to specify the fluid velocity on the substrate.

5) Finally, a solution was obtained with the blade also being made flexible and using continuation from a "high modulus" case as described above. As expected, substrate deformation and blade deflection correspond to hydrodynamic pressure forcing along the liquid/solid interfaces. The distinguishing condition for the blade is the "FLUID_SOLID" card (one for the blade tip, side set 4, and one for the blade/liquid surface, side set 5). The GD_CONST cards are used in this case to specify the fluid velocity at the blade/liquid surface. Notice that the velocity conditions are actually applied to the liquid momentum equations.

The details of running this advanced example are not supplied here. An important fact to remember is that this is a nonlinear operating state, and not all initial guesses will converge for step 1

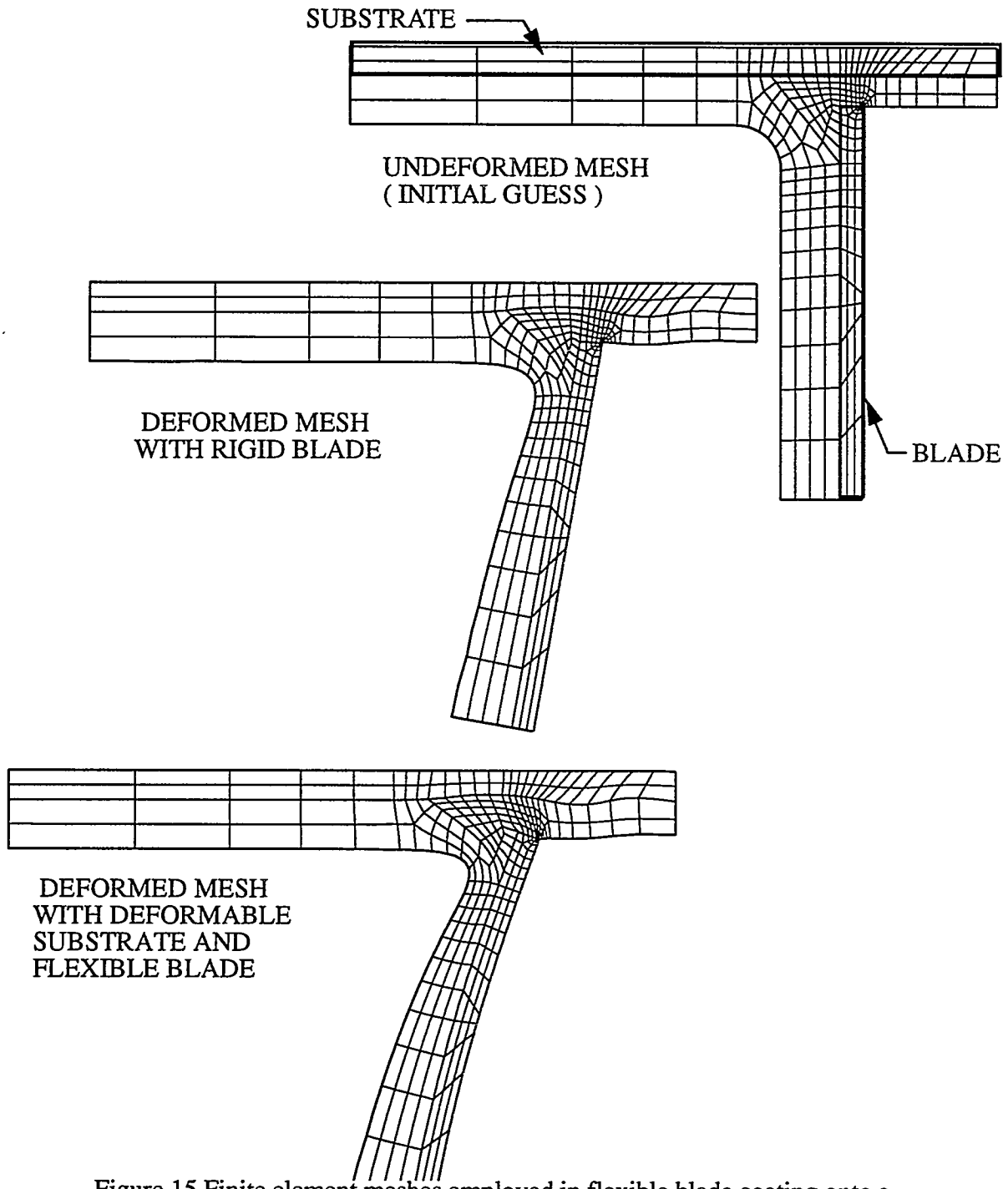


Figure 15 Finite element meshes employed in flexible blade coating onto a deformable substrate.

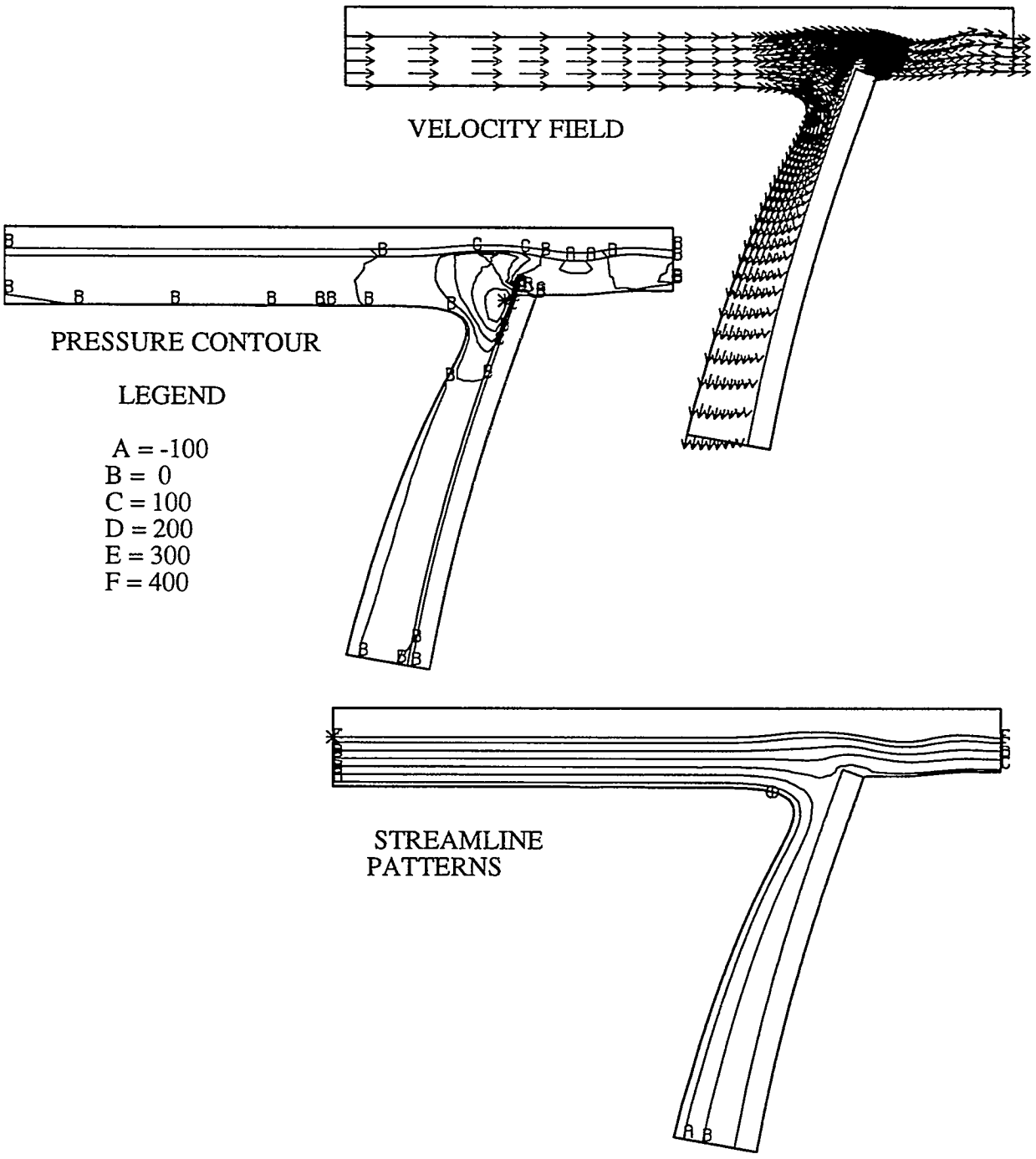


Figure 16 Typical results of flexible blade coating unto a deformable substrate (μ = 50 mPa s, γ = 50 mN/m, U_s = 1 m/s, G_s = 600 Pa, K_s = 600 Pa, G_b = 2240 Pa, K_b = 2250 Pa).

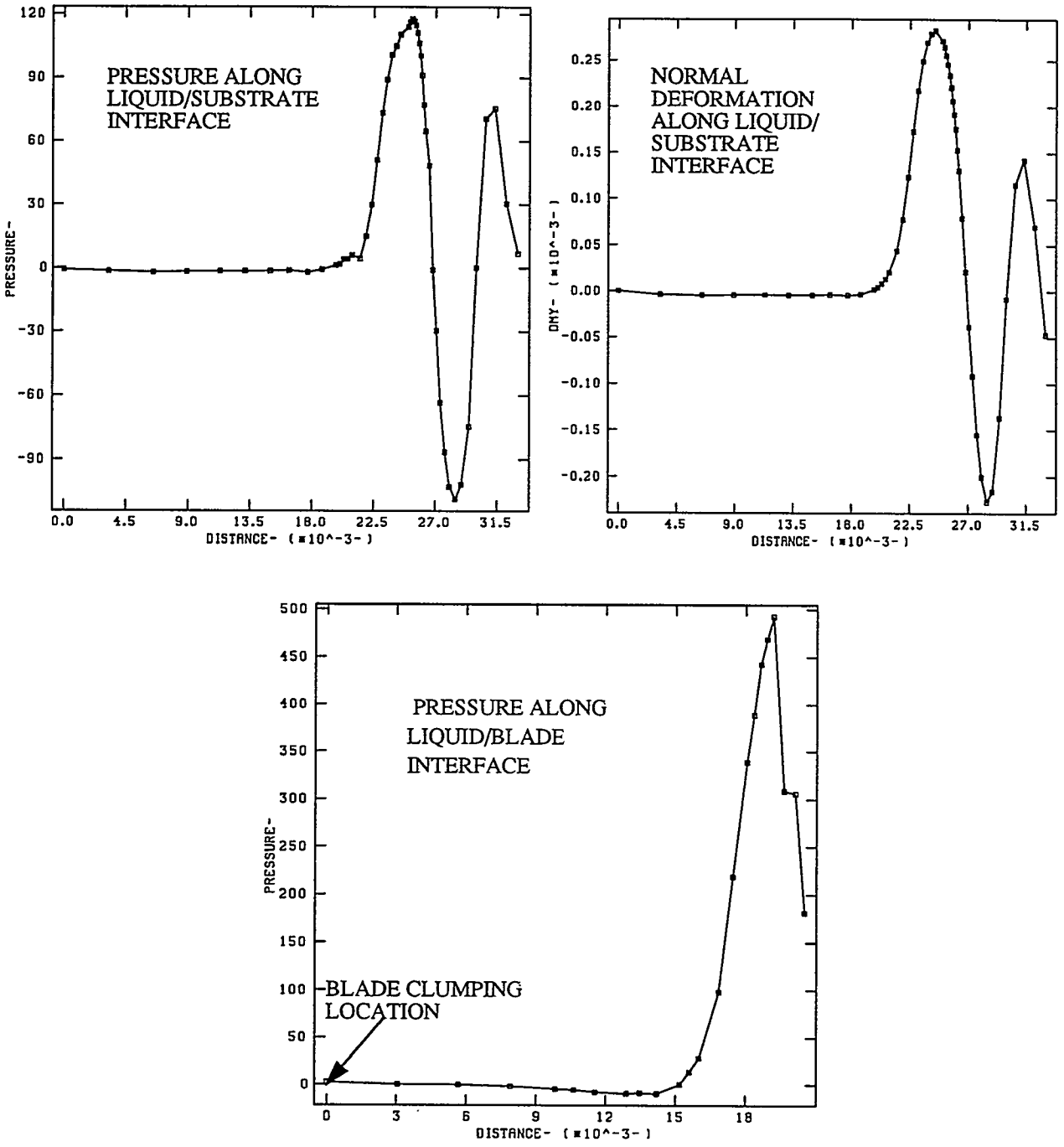


Figure 17 Pressure profiles along substrate/liquid & blade/liquid interfaces.

through 3 above. However, once those steps are accomplished, releasing the blade and substrate follow in the way described.

References

[1] Bird, R. B., Stewart, W. E. , and Lightfoot, E. N., 1960. Transport Phenomena, John Wiley & Sons, New York.

[2] Bird, R. B., Armstrong, R. C., and Hassager, O. 1987. Dynamics of Polymeric Liquids, 2nd ed., Wiley, New York, Vol. 1.

[3] Blacker, T. D. 1988. "FASTQ Users Manual: Version 1.2", Sandia Technical Report SAND88-1326.

[4] Blacker, T. D. and Stephenson, M. B. 1990. "Paving: a new approach to automated quadrilateral mesh generation", Technical Report SAND90-0249, Sandia National Laboratories, Albuquerque, New Mexico.

[5] Cairncross, R. A., Chen, K. S., Schunk, P. R., Brinker, C. J., and Hurd, A. J. 1995. "Recent advances in theoretical modeling of deposition, drying, and shrinkage in sol-gel coating processes," Proceedings on Computational Modeling of Materials and Processing Syposium at the American Ceramic Society National Meeting, Cincinati, OH, 30 April - 3 May.

[6] Chen, K. S., Schunk, P. R., and Sackinger, P. A. 1995 "Finite element analyses of blade and slot coating flows using a Newton-Raphson pseudo-solid domain mapping techniquie and unstructured grids", Proceedings of the 1995 TAPPI conference.

[7] Gartling, D. K. 1995. "TORO II: A finite element program for quasi-static problems in electromagnetics: Part II User's Manual", Sandia Technical Report, in press.

[8] Gartling, D. K., Givler, R. C., Hickox, C. E. 1995. "Simulations of coupled viscous and porous flow problems", *Int. J. Comp. Fluid Dynamics* (in press).

[9] Gilkey, A. P. and Glick, J. H. 1988. "BLOT-A mesh and curve plot program for the output of a finite element analysis" Sandia Technical Report SAND88-1432.

[10]Glass, M. G. 1995. Personal communication.

[11]Hutchinson, S. A., Shadid, J. N., Tuminaro, R. S. 1995. "Aztec User's Guide Version 1.0", Sandia Internal Report, SAND95-1559.

[12]Kernighan, B. W. and Ritchie, D. M. 1988. The C Programming Language, 2nd Ed., PTR Prentice Hall, New Jersey.

[13]Kistler, S. F. and Scriven, L. E. 1983. Coating Flows. In Computational Analysis of Polymer Processing. Eds. J. A. Pearson and S. M. Richardson, Applied Science Publishers, London.

[14]Kundert, K. S. and Sangiovanni-Vincentelli, A. 1988. "Sparse User's Guide: Version 1.3a" Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley.

[15]PDA Engineering, "PATRAN Plus User Manual," Publication No. 2191024, Costa Mesa, California, January 1990.

[16]Rew, R. K., Davis, G. P., and Emmerson, S. 1993. "NetCDF User's Guide: An Interface for Data Access," Version 2.3, University Corporation for Atmospheric Research, Bolulder, Colorado, April 1993.

[17]Sackinger, P. A., Schunk, P. R. , Rao, R. R. 1995. "A Newton-Raphson pseudo-solid domain

mapping technique for free and moving boundary problems: A finite element implementation", accepted for publication, *J. Comp. Phys.*, November 1995.

[18]Sartor, L. S. 1990. "Slot Coating: Fluid Mechanics and Die Design", Ph. D. Thesis, University of Minnesota. Available from University Microfilms International, Ann Arbor Michigan.

[19]Scherer, G.W., 1992, "Recent Progress in Drying of Gels", J. of Non-Crystalline Solids, 147&148, 363-374.

[20]Schoof, L. A. and Yarberry, V. R. 1994. "EXODUS II: A finite element data model", Sandia Technical Report. SAND92-2137.

[21]Schunk, P. R. and Shadid, J. N. 1992. "Iterative solvers in implicit finite element codes" Sandia Technical Report, SAND92-1158.

[22]Segalman, D., Witkowski, W., Adolf, D., and Shahinpoor, M., 1992. "Theory and Application of Electrically Controlled Polymeric Gels", *Smart Mater. Struct.*, 1, 95-100.

[23]Shadid, J. N. and H. K. Moffat, "SALSA: A MP finite element computer program for reacting flows. Part 1 - Theoretical background and equation development." Sandia Technical Report, to be published, May 1995.

[24]Shadid, J. N., Hutchinson, S. A., Moffat, H. K., Hennigan, G. L., Hendrickson, B. A., and 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, 14-18 Nov. 1994.

[25]Sjaardema, G. D. 1992. "APREPRO: An algebraic preprocessor for parameterizing finite element analyses", Sandia Technical Report SAND92-2291.

[26]Sjaardema, G. D. 1993. "Overview of the Sandia National Laboratories engineering analysis code Access System", Sandia Technical Report SAND92-2292.

[27]Malvern, L. E., 1969, Introduction to the Mechanics of a Continuous Medium, Prentice-Hall, Inc., Englwood Cliffs, New Jersey.

[28]Martinez, M. J., 1995, "Formulation and Numerical Analysis of Nonisothermal Multiphase Flow in Porous Media", Sandia Technical Report SAND94-0379.

[29]Pranckh, F. R. and Scriven, L. E. 1988. "The physics of blade coating of deformable substrates" 1988 Coating Conference Proceedings, TAPPI Press, Atlanta, 217-238.

Appendix

A Continuation Strategies for Free Surface Flows

In free surface problems there exists one or more boundaries or internal surfaces whose position(s) are unknown *a priori*. As such, the geometry of the problem becomes part of the problem and must be determined together with the internal physics. Most problems of this sort cannot be solved with a trivial initial guess to the solution vector, mainly because the conditions which determine the surface position are closely coupled to the active physics in the bulk. Thus, these problems required continuation (zero or higher order) to achieve a converged solution to a desired state. The continuation strategy typically involves turning on and off the conditions which distinguish the position of the free surface(s). This appendix describes one such strategy.

Distinguishing conditions in GOMA serve two purposes: (1) they can be used to locate a surface whose position depends on internal and interfacial transport phenomena, and (2) they can be used to prescribe solid boundary position or motion. The first type of condition contains field variables needed to locate the interface or free surface position, and hence ties the mesh motion to the problem physics, i.e., mass, momentum, and energy transport phenomena. Currently, the side-set boundary conditions of type DISTNG, KINEMATIC, and KIN_LEAK (see Section 4.5) fall into this class. The second type of condition requires only geometrical information from the mesh, and, although geometrically couples the mesh motion to the problem physics, it tends not to be so tightly coupled. Currently, boundary conditions PLANE, PLANEX, PLANEY, PLANEZ, SPLINE, SPLINEX, SPLINEY, and SPLINEZ fall into this class.

In two dimensions, there is no need to use PLANEX, PLANEY, PLANEZ, SPLINEX, SPLINEY, and SPLINEZ. Because the code automatically rotates the mesh residual equations and the corresponding Jacobian entries into normal-tangential form on the boundary, SPLINE, PLANE, and DISTNG are the only cards required to specify the position of the boundary. Currently, in three dimensions, the logic for the same rotation concept is not totally functional, and one must use the PLANEX, etc. cards to designate which component of the mesh stress residual equation receives the distinguishing conditions.

If cards DISTNG, KINEMATIC and KIN_LEAK, i.e., distinguishing conditions of type 1, are absent in any simulation, then any initial guess for the transport field equations, i.e., energy and momentum, has a chance of converging, as long as the initial mesh displacement guess is within the radius of convergence of the mesh equations and associated boundary conditions. For example, if the side sets of the EXODUS II database mesh correspond somewhat closely to what is prescribed with PLANE- and SPLINE-type conditions, then an initial guess of the NULL vector has a good chance of converging, so long as the velocities and temperatures are within "converging distance."

When conditions from the first class are present, i.e., either DISTNG, KIN_LEAK or KINE-

MATIC, then the following proceedure should be followed:

a. Set the keyword in the **Initial□Guess** character_string (4.2.4) to **zero, one,** or **random.**

b. Obtain a solution (run GOMA) with the initial guess for the free surfaces distinguished as KINEMATIC (or other) coming from the EXODUS II database, but without the KINEMATIC (or other) card(s). That is, "fix" those surfaces with either a PLANE or SPLINE command, or simply place no distinguishing condition on them (this works only if the grid has not been previously "stressed", i.e., all the displacements are zero). The rest of the "desired" physics should be maintained. If any surface is distinguished as KINEMATIC, then it is highly advantagous to place a VELO_NORMAL condition on that surface for startup, and set the corresponding floating point datum to zero. This effectively allows the fluid to "slip" along that boundary as if it were a shear free condition.

c. Set the keyword in the **Initial□Guess** caracter_string (4.2.4) to **read.**

d. Copy the file named in **SOLN□file** (4.1.4) into the file named in **GUESS□file** (4.1.3).

e. Release the free boundaries by taking off any current distinguishing condition cards and adding the appropriate KINEMATIC (or other) card. Adjust all other boundary conditions appropriately.

f. Run GOMA. There may be need to set the Newton relaxation factor in complex cases to something less than unity but greater than zero, of course (e.g. 0.1) for complex flows.

When dealing with material surface boundaries distinguished by the kinematic boundary condition, the nature of that condition requires a non-zero and substantial component of velocity tangent to the surface upon start-up. In this case, it can be advantagous to use the VELO_TANGENT card to set the velocity along the free surface to some appropriate value in step c above. Of course this card will be removed in subsequent steps. Also, although not necessary, a smooth, "kinkless", initial guess to the free surface shape is helpful because it reduces the amount of relaxation required on the Newton iteration.

Obtaining start-up solutions of most coating flow configurations is still an art. In this Appendix and in Chapter 7 we have tried to compile and record much of our experience towards attaining an initial guess to the desired solution state. In Chapter 7 we give examples of these start-up strategies. Of course the best way to start up a coating flow analysis may be to acquire a "template" developed from a previous analysis of some closely related flows.

Index

O

P

Distribution

Dr. Brett A. Lewis
Aptek Inc.
1257 Lake Plaza Dr.
Colorado Springs, CO 80906

Edward D. Cohen
DuPont Printing and Publishing
Corporate Coating Technology Center
Experimental Station
P. O. Box 80352
Wilmington, DE 19880-0328

Jim Wheeler
Director, Coating Technology
Eastman Kodak Company
1669 Lake Avenue, Building 2
Kodak Park
Rochester, NY 14652-4710

Dr. Thomas K. Winkler
Unit Director
Coatinge Technologies Div. - MREO
Eastman Kodak Company
Building 35, Kodak Park
Rochester, New York 14652-3701

Dr. Brent Bell
Building 2
Kodak Park
Eastman Kodak Company
1669 Lake Avenue
Rochester, NY 14652-4710

Thomas G. DeNoto
Senior Principal Engineer
Industrial Coating Division
Polaroid Corporation
1265 Main Street, Bldg. W5
Waltham, MA 02254

Mark R. Monterastelli
Process Development Engineer
Pre Finish Metals Inc.
2111 E. Pratt Blvd.,
Elk Grove Village, IL 60007

Merle Scharfe
D&M SD&MU
P/R Products Concepts Technology Area
Xerox Corporation
800 Phillips Road, Building 103-05B
Webster, NY 14580

Jim J. Cai
Xerox Corporation
800 Phillip Road 103/05B
Webster, NY 14580

Luigi Sartor, Ph.D.
Avery Dennison
2900 Bradley Street
Pasadena, CA 91107-1599

Dan Logue, Ph.D.
Avery Dennison
2900 Bradley Street
Pasadena, CA 91107-1599

Steve Burdette
Roll Division
Avery Dennison
7670 Auburn Road
Painesville, OH 44077

Dean Benjamin
Research & Development Division
Consolidated Papers, Inc.
300 North Biron Dr. PO BOX 8050
Wisconsin Rapids, WI 54495-8050

Shuzo Fuchigami
3M Engineering Systems and Technology
3M Center, Building 570-1W-03
St. Paul, MN 55144-1000

Robert Secor
3M Engineering Systems and Technology
3M Center, Building 570-1W-03
St. Paul, MN 55144-1000

Rich A. Cairncross
Dept. Mechanical Engineering
University of Delaware
126 Spencer Laboratory
Newark, Delaware 19716-3140

MS 0601	1126	Moffat, H. K.
MS 9043	8743	Callebresi, M.
MS 9043	8743	Kanouff, M. P.
MS 0841	9100	Hommert, P. J.
MS 0828	9102	Skocypec, R. D.
MS 0833	9103	Biffle, J. H.
MS 0828	9104	Gorham, E. D.
MS 0826	9111	Hermina, W.
MS 0826	9111	Chen, Ken S. (5)
MS 0826	9111	Christon, Mark
MS 0826	9111	Gartling, David K.
MS 0826	9111	Givler, Rick C.
MS 0826	9111	Glass, Micheal W.
MS 0826	9111	Kempka, Steven N.
MS 0826	9111	Larson, Don E.

MS 0826	9111	Rao, Rekha R. (5)
MS 0826	9111	Sackinger, Phil A. (5)
MS 0826	9111	Schunk, P. Randall (10)
MS 0826	9111	Schutt, J. A.
MS 0826	9111	Walker, M. A.
MS 0834	9112	Ratzel, A. C.
MS 0834	9112	Baer, T.
MS 0834	9112	Eaton, R. R.
MS 0834	9112	Gelbard, F.
MS 0834	9112	Hopkins, P. L.
MS 0834	9112	Martinez, M. J.
MS 0834	9112	Torczynski, J. R.
MS 0835	9113	Gianoulakis, S. E.
MS 0835	9113	Blackwell, B. F.
MS 0835	9113	Cochran, R. J.
MS 0835	9113	Hogan, R. E.
MS 0835	9113	Lober, R. R.
MS 0835	9113	Romero, V. J.
MS 0827	9114	McGrath, R. T.
MS 0827	9114	Geller, A. S.
MS 0827	9114	Rader, D. J.
MS 0827	9114	Russo, A. J.
MS 0825	9115	Rutledge, W. H.
MS 0836	9116	Peterson, C. W.
MS 0836	9116	Boucheron, E. A.
MS 0443	9117	Morgan, H. S.
MS 0443	9117	Hodapp, A. E.
MS 0437	9118	Thomas, R. K.
MS 1111	9221	Shadid, J. N.
MS 1111	9221	Salinger, H. K.

MS 9018	8523-2	Central Technical Files
MS 0899	4414	Technical Library (5)
MS 0619	12615	Print Media
MS 0100	7613-2	Document Processing for DOE/OSTI

MS ~~Org.~~	Dept. ~~Bldg.~~	Name	Rec'd by	Org.	Bldg.	Name	Rec'd by
0100	7613-2	Doc. Proc. for DOE/OST					

Sandia National Laboratories