SANDIA REPORT

SAND2002-2775
Unlimited Release
Printed October 2002

ALEGRA: User Input and Physics Descriptions Version 4.2

Edward A. Boucheron, Kevin H. Brown, Kent G. Budge, Shawn P. Burns, Daniel E. Carroll, Susan K. Carroll, Mark A. Christon, Richard R. Drake, Christopher G. Garasi, Thomas A. Haill, James S. Peery, Sharon V. Petney, Joshua Robbins, Allen C. Robinson, Randy Mr. Summers, Thomas E. Voth, and Michael K. Wong

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

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

Sandia National Laboratories

SAND2002-2775
Unlimited Release
Printed October 2002

ALEGRA: User Input and Physics Descriptions Version 4.2

Edward A. Boucheron, Kevin H. Brown, Kent G. Budge, Shawn P. Burns, Daniel E. Carroll, Susan K. Carroll, Mark A. Christon, Richard R. Drake, Christopher G. Garasi, Thomas A. Haill, James S. Peery, Sharon V. Petney, Joshua Robbins, Allen C. Robinson, Randy M. Summers, Thomas E. Voth, and Michael K.Wong
Computational Physics R&D Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0819

Abstract

ALEGRA is an arbitrary Lagrangian-Eulerian finite element code that emphasizes large distortion and shock propagation. This document describes the user input language for the code.

Acknowledgment

ALEGRA development began nearly a decade ago, and many individuals have contributed to its development over the years. We would like to particularly acknowledge efforts on the part of George Allshouse (deceased), who made important contributions to the early planning of the ALEGRA project and facilitated its beginning. Mike McGlaun launched the code development effort in our department, and Keith Matzen and Tim Trucano also contributed to early planning activities for the project. Below we list major contributors from the core ALEGRA development team for specific areas of the code. In addition, several developers from other projects at Sandia and LANL have provided guidance and assistance in importing various models and features to ALEGRA, including Steve Attaway, Pang Chen, Ben Cole, Dave Crawford, Doug Drumheller, Jonathan Hu, Frank Mello, Steve Montgomery, Judy Sturtevant, Bryan Oliver and Ray Tuminaro. Others have utilized ALEGRA as a framework for developing special purpose capabilities, including Mike Glass, Steve Kempka, Josh Robbins, Rob Schmitt, Dave Turner, Mike Pasik, Dave Siedel. A number of users of the code have provided very valuable feedback to the development team that helped improve the code enormously, including John Aidun, Mark Boslough, Lalit Chhabildas, Paul Demmie, Jeff Lawrence, Rich Jensen, Steve Schraml, Dan Mosher, Scott Wunsch, Ray Lemke, Ray Campbell. A legion of students have helped with various aspects of the code and with the development of test suites including Mary Chen, Rachel Bixler, Elena Agustin. Finally, several project teams are developing tools that will help make effective use of ALEGRA in specialized analyses, including the Dakota, DOOMSDACE, SimTracker, and Simulation Intranet teams.

Ray Bell	SMYRA Material Interface Tracker
Ed Boucheron	Management, Mesh adaptivity, Desktop tools
Rebecca Brannon	Electromechanics material models
Kevin Brown	Contact algorithms, Architecture, Solid Mechanics
Tom Brunner	Advanced Physics
Kent Budge	Architecture, Lagrangian physics, Advanced physics, Penetration
Dan Carroll	Code Team Leader, ALE algorithms, Applications
Sue Carroll	Configuration management, Regression testing, Support tools
Mary Chen	Verification & Validation
Mark Christon	Code coupling, I/O, Visualization
Kyle Cochrane	Advanced Physics
Rich Drake	Code coupling, Error estimation
Archie Farnsworth	Electromechanics material models, Applications
Grant Farnsworth	Visualization
Chris Garasi	Verification & Validation
Thomas Haill	Advanced physics
Will McLendon	Advanced physics
James Peery	Architecture, ALE algorithms, Parallelization
Sharon Petney	Mesh material insertions, Material models
Josh Robbins	Electromechanical models, Applications
Allen Robinson	Electromechanics models, Advanced physics, Parallelization

Peter Stoltz	Advanced Physics
Randy Summers	Contact algorithms, Platform Adaptation, Advanced Physics
Tim Trucano	Verification & Validation, Applications
Randy Weatherby	Mesh adaptivity, Desktop tools, Visualization
Mike Wong	Mesh adaptivity, Material models

List of Tables

List of Figures

Summary

In 1990, an effort was launched at Sandia National Laboratories to develop a state-of-the-art code that combined the modeling features of modern Eulerian shock codes, such as CTH, with the improved numerical accuracy of modern Lagrangian finite element codes. The resulting code, called ALEGRA, uses an arbitrary Lagrangian-Eulerian (ALE) formulation on an unstructured finite element mesh. This formulation allows the user to designate whether material should flow through a stationary mesh (pure Eulerian), whether the mesh should move with the material (pure Lagrangian), or whether the mesh should move independently from the material motion (arbitrary). The latter capability permits a calculation to proceed in Lagrangian fashion until the mesh becomes too highly distorted. At that time, mesh points in the most deformed portion of the mesh are moved to reduce the distortion to acceptable levels. The advantage is that numerical dissipation is avoided until large deformations occur and then is limited to only those regions where there are severe mesh distortions and the mesh must be moved.

ALEGRA is written predominantly in the C++ programming language, although we have limited the use of some features of C++ to avoid efficiency problems. This allows us to take advantage of object-oriented programming techniques in managing the inherent complexity of the physics being modeled. However, we have also recognized the utility of incorporating various Fortran-based models and libraries if they best serve our modeling needs and if they are sufficiently mature and robust. In many cases there is little advantage to rewriting such software.

ALEGRA has been designed to run on distributed-memory parallel computers. This was done because the enormous memories and processor speed of massively parallel processor (MPP) computers are needed to analyze large, three-dimensional problems. The memory requirements for ALEGRA scales inversely with the cube of the zone size. For example, if we halve the mesh size in each direction, then the memory requirement increases by a factor of eight. ALEGRA uses explicit time integration schemes so the time step scales inversely with the mesh size. For example, if we halve the mesh size, the code cuts the time step in half. Therefore, the Floating Point OPerations (FLOPs) scale as the fourth power of the mesh size. Since the FLOP requirements increase faster than the memory requirements, simply increasing the memory on existing super computers is not a good solution to running larger problems because the run time quickly becomes excessive.

The database of a large, three-dimensional problem is too large to fit on any single compute node. Therefore, ALEGRA was designed with the Single Program Multiple Data (SPMD) paradigm, in which the mesh is decomposed into sub-meshes so that each processor gets a single sub-mesh with approximately the same number of elements. Good mesh decomposition is important to minimize the data passed between compute nodes. Whereas rectangular meshes are relatively

Table of Contents
Summary

easy to decompose, subdividing the arbitrary connectivity meshes used by ALEGRA is much more difficult. We use Sandia's Chaco package to decompose these meshes.

ALEGRA uses one layer of ghost elements around the sub-mesh perimeter for subdomain boundary conditions. The database for the ghost elements must be updated once each computational cycle by interprocessor communication. These additional ghost elements represent a parallel processing cost that can be quite large for compute nodes with a small number of elements. For example, a cube meshed in a 10 by 10 by 10 regular pattern, approximately half the elements are boundary elements. For large, roughly cubic meshes, the fraction of boundary elements goes as approximately $6/N^{1/3}$, where N is the number of elements, so a million-element mesh will include only about 6% boundary elements. Therefore, we use sub-meshes that fill each compute nodes's memory to minimize the number of ghost elements. Minimizing the number of boundary elements also minimizes the amount of data passed between compute nodes.

ALEGRA also utilizes adaptive mesh refinement techniques to provide efficient, high precision simulations without the computational cost of using a highly resolved mesh everywhere. This capability is critical for simulations over large spatial domains that require high precision in the presence of certain features such as shocks, burn fronts, pressure stagnation areas, and regions of large deformation.

This document describes how to use ALEGRA and contains a thorough description of the input keywords, their parameters, and how they may be used to develop an input file.

Nomenclature

Table 1: Terms and Acronyms

Term	Definition
ALE	Arbitrary Lagrangian Eulerian
ALEGRA	Arbitrary Lagrangian Eulerian General Research Application
BC	Boundary Condition
EOS	Equation of State
IC	Initial Condition
MMALE	Multi-Material Arbitrary Lagrangian Eulerian
SMALE	Single-Material Arbitrary Lagrangian Eulerian
MPP	Massively Parallel Processor
PDS	Parallel Data Set
SPMD	Single Program Multiple Data

1.0 Introduction to ALEGRA

ALEGRA [11],[39] is an ALE (*A*rbitrary *L*agrangian-*E*ulerian) multi-material finite element code that emphasizes large deformations and strong shock physics. As an effort to combine the modeling features of modern Eulerian shock codes with the improved numerical accuracy of modern Lagrangian finite element codes, ALEGRA is a descendant of the PRONTO transient dynamics code [40],[41] and contains elements of the CTH family of shock wave codes [29],[21]. This capability permits a calculation to proceed in Lagrangian fashion until portions of the finite element mesh become highly distorted, at which time the nodal points in the most deformed portion of the mesh are moved to reduce the distortion to acceptable levels. The advantage is that numerical dissipation is avoided until large deformations occur and this is limited to only those regions where severe distortions require mesh movement. In addition to mesh smoothing, the ALEGRA remesh algorithm can also move nodes to better resolve mesh regions with specific values of selected variables or their gradients.

ALEGRA is written predominantly in the C++ programming language [15], though we have limited our use of some features of C++ to avoid efficiency problems. This allows us to take advantage of object-oriented programming techniques in managing the inherent complexity of the physics models being implemented. However, we have also recognized the utility of incorporating various Fortran-based models and libraries if they best serve our modeling needs and if they are sufficiently mature and robust. Recent efforts in ALEGRA have focused on transforming the code into a “framework” for new simulation applications. To this end, most of the physics-specific treatments and references have been removed for the base classes of the code. The goal which has been achieved with Version 4.0 is to enable a stand-alone library of ALEGRA capability to be built separately and then to implement a simple physics model in a standalone manner. The framework has come to be known as NEVADA, to emphasize its complimentary nature to the other main framework at SNL, SIERRA.The NEVADA framework is described by the Application Programmer’s Guide to the NEVADA Framework, by Kent Budge. this document has not yet been formally distributed but a draft is available upon request.

ALEGRA has been designed to run on distributed-memory parallel computers using the Single Program Multiple Data (SPMD) paradigm, in which the mesh is decomposed into submeshes(i.e., domain decomposition). This was done because we needed the enormous memories and processor speed of massively parallel processor (MPP) computers to analyze large, three-dimensional problems.

The mesh used by ALEGRA is taken from a GENESIS database[1] [40] prepared by a preprocessing package. At present FASTQ [6], GEN3D [18], GJOIN [36], and CUBIT [38] are the preprocessing packages used. In addition, there is an “exodus preference” available for Patran and the CUBIT tool accepts several common solid modeler output formats (e.g. ACIS and IGES)

1. GENESIS is the non-transient part of the EXODUS database

as input and then produces an finite element mesh in exodus format. The NEMESIS utilities based on Sandia's Chaco package [19] determine the decomposition of the mesh. Good mesh decomposition is important to minimize the memory requirements, balance the work on the compute nodes, and minimize the data passed between compute nodes. Whereas rectangular meshes are relatively easy to decompose, subdividing the arbitrary connectivity meshes used by ALEGRA is much more difficult. Recombination of the parallel result output files produced by ALEGRA also is done with the NEMESIS utility NEM_JOIN.

The plot output file from ALEGRA is in EXODUS database format (Mills-Curran, 1988). The plot file can contain as many or few variables as the user desires and can include all the standard nodal and elemental variables as well as material internal state variables. The EXODUS format output files may be postprocessed using the BLOT [17], MUSTAFA (an internal Sandia visualization package), Ensight (a commercial visualization package from Computational Engineering International, Inc.) graphics packages. In addition, ALEGRA provides both Eulerian and Lagrangian tracer particles that record time history data for selected variables (e.g., pressure vs. time at the tracer location) in the HISPLT format, allowing post-processing with the HISPLT code [44]. The restart output file from ALEGRA is in a separate format and cannot be used with current postprocessing tools.

ALEGRA accepts mnemonic, free-format input. Descriptions of the keywords recognized by ALEGRA and their syntax requirements are provided in **Section 3.0 on page 45**.

At present, ALEGRA 2-D and 3-D versions are built separately. The 2-D versions handle both planar and axisymmetric geometries, permitting application to a reasonably broad class of problems. The 3-D versions are restricted to Cartesian geometry.

ALEGRA is supported on the following platforms. In addition, ALEGRA can be run using MPI message passing on workstation and PC clusters.

Table 2: Supported Platforms

Platform	OS	Compiler
`SUN Workstation`	`Solaris 8 (OS 5.8)`	`Workshop 6.0, gcc 2.953`
`Linux PC`	`Red Hat 7`	`gcc 2.96`
`SGI Origen 2100`	`IRIX 6.5 64 bit`	`Native`
`IBM RS 6000`	`AIX 4.3`	`Native`
`SNL TeraFlop`	`Couger 4.0`	`SNL developed`
`SNL Cplant`	`SNL developed`	`SNL developed`

The first released version of this version of ALEGRA will be numbered 4.2.1. This is the version number that will be on the CDs that are sent out to external users. Internal users of the ALEGRA executable saved in the “Stable” directory will be using this 4.2.1 version of the code. Subsequent versions of this base code, released for bug correction, will be numbered 4.2.2, 4.2.3 and so forth.

2.0 Overview

This chapter describes the use of the ALEGRA program, including mesh generation, problem specification, and post-processing. This chapter is arranged to take a user through the steps required to successfully run a simulation with ALEGRA. This section is not intended to describe in detail the physics or algorithms in ALEGRA. Several reference documents are available that cover these areas. Nor will this document provide all the detail necessary to perform the pre- and post-processing phases of an ALEGRA simulation. If you plan to use ALEGRA, you will want to acquire up-to-date manuals for the pre- and post-processing tools.

To successfully run a simulation with ALEGRA, one must 1) create a finite element mesh with tools like **FASTQ**, **GEN3D**, and/or **CUBIT**, 2) create an ALEGRA problem specification deck, 3) run ALEGRA, and 4) examine the results using visualization tools such as **BLOT**, **MUSTAFA**, **ENSIGHT** and **HISPLT**. In addition, for parallel runs, one must learn how to use the **loadbal** script and in some cases the **Concat or Combinemp** scripts. This chapter is divided into the following sections:

- Basic ALEGRA Environment
- Running ALEGRA
- ALEGRA Example Problems

2.1 Basic ALEGRA Environment

The ALEGRA run time environment is based on:

- Environment variables
- Shell scripts
- Sandia's ACCESS system of finite element support tools.

Anyone wanting to use ALEGRA will need to be acquainted with UNIX-based operating systems.

2.1.1 Environment Variables

The ALEGRA code relies heavily on environment variables set within the user's .cshrc file. These environment variables are used within every context of the ALEGRA environment and allow code developers and users tremendous freedom in maintaining several versions of the ALEGRA code. At a minimum, the following environment variables must be set in order to run ALEGRA:

```
ALEGRA_ARCH
ALEGRA_EXE
ALEGRA_MP
ALEGRA_MIGDATA
ACCESS
```

ALEGRA_EXE should define the path to the ALEGRA executable. Expert users will set this based on the version of ALEGRA they want to run.

ALEGRA_MP should be set to mp_none for serial compilations and mp_mpi for parallel compilations of ALEGRA.

ALEGRA_ARCH refers to an operating system or computer vendor along with a compiler description, if necessary. Acceptable values for ALEGRA_ARCH are: solaris-sw-6.0, solaris-gnu, irix64-6.4, linux-gcc, cplant, and xtflop

ALEGRA_MIGDATA sets the path to the directory containing input parameters for the KEOS models for many predefined materials.

The ACCESS environment variable refers to the "root" location of the ACCESS system. Under this "root" location you would find the "etc", "inc", and "lib" directories. The ACCESS system provides many of the pre- and post-processing tools for finite element codes along with libraries for the EXODUS database. ALEGRA can be run without the ACCESS system, but it is not recommended. Usually the system administrator is responsible for determining the correct setting for the ACCESS environment variable.

Sandia users (valinor lan, tflop lan, tflop-s lan, tesla) should establish the proper environment by sourcing the alegra.users script available in $ALEGRA_ROOT/etc on each machine. This script will define ALEGRA_ROOT on each platform according to the following table:

Table 3: ALEGRA_ROOT Paths

Platform	Path
valinor lan	/pr/alegra
janus, tesla	/.../dce.sandia.gov/fs/proj/alegra
cplant	/projects/alegra
edison	/projects/alegra
wizard lan	/home/projects/alegra

The alegra.users script will also modify the users PATH Unix variable to point to appropriate directories.

2.2 Running ALEGRA

ALEGRA uses a common prefix for all of the files either read or written. This prefix is called the *runid*. The *runid* prefix can be any valid UNIX character string. The suffixes appended to *runid* determine the characteristics of the file. **Figure 1** displays the possible files that are used in an ALEGRA simulation.

To use ALEGRA, a user must become proficient in the following procedures:

- Preprocessing or mesh creation
- User problem specification creation
- ALEGRA execution
- Postprocessing or visualization

Throughout this process, users should consider where the natural boundaries among materials exist and how the materials should behave. In ALEGRA, these boundaries define the boundaries of blocks of elements. In the preprocessing phase, the element blocks are given unique identifiers that can be tied to the problem specification input file. This coupling of identifiers allows the user to control physical and algorithmic parameters associated with the block. In addition, boundary conditions in the problem specification are tied back to unique identifiers in mesh creation that represent collections of nodes or surfaces. As users create a mesh, they will want to uniquely identify nodes, lines, and surfaces upon which to apply boundary conditions. Finally, some attention should be given to how blocks and materials will be used in the visualization process. For example, if a user wants to differentiate between two areas of the same material, separate blocks can be created and the material duplicated in the problem specification.

Table of Contents
Overview

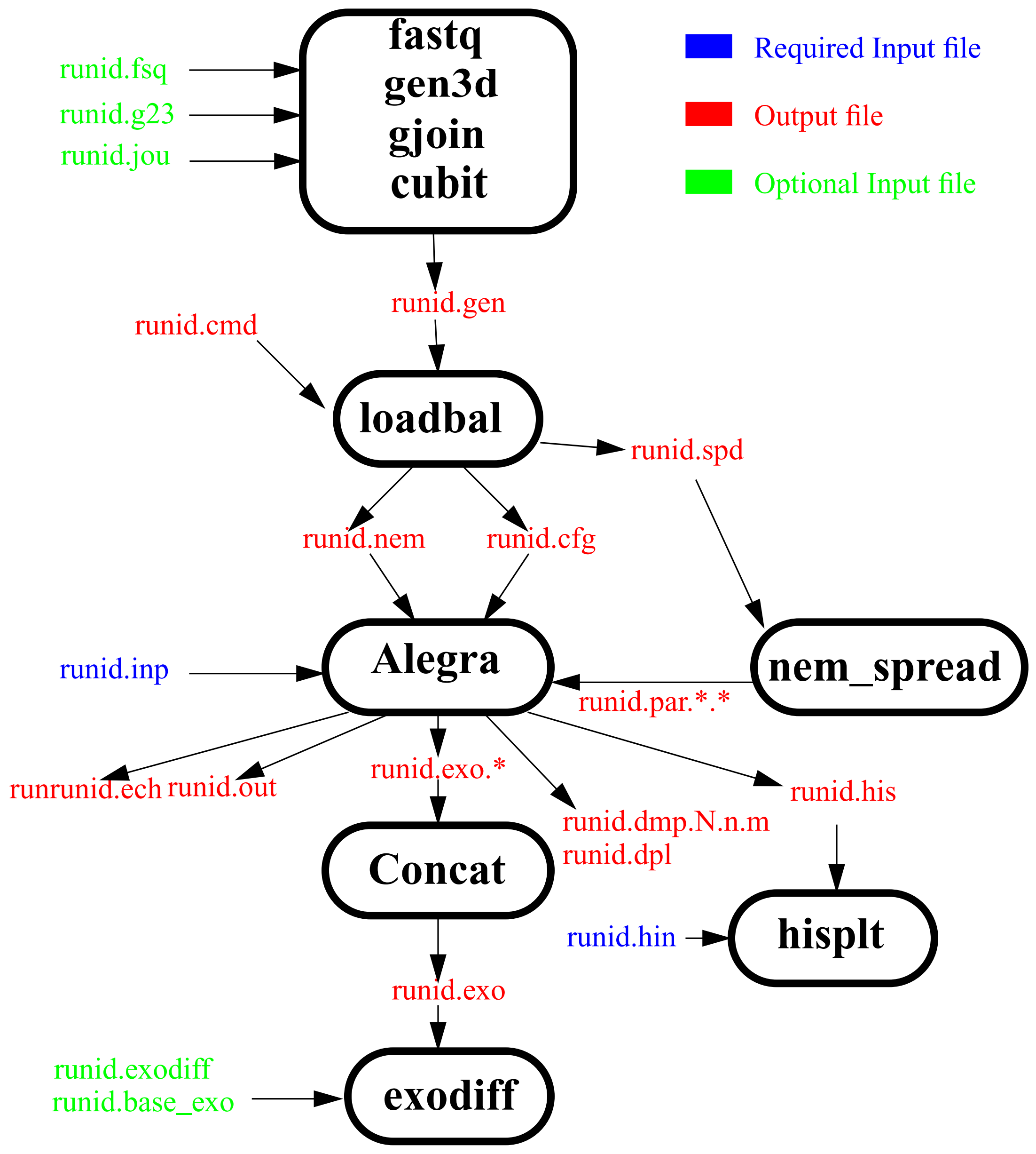


Figure 1: The ALEGRA File Soup

2.2.1 Preprocessing

ALEGRA uses an unstructured finite element discretization for solving the governing partial differential equations. ALEGRA can begin with a body fitted mesh, a background mesh in which “shapes” are inserted, or a combination of both. The shape insertion option is useful for very complex geometries which can be accurately modeled in an Eulerian framework. This is the model that all Eulerian hydrodynamic codes use. ALEGRA accepts the same “shape” input (the “DIATOM” specification) as the CTH code.

The preprocessing steps for ALEGRA include:

1. Mesh Creation
2. Mesh Partitioning (Parallel Runs)

The creation of three-dimensional body-fitted meshes can be very time-consuming. Many years of research have gone into automating the meshing of solid model geometry. Most three-dimensional meshes require the user to decompose the geometry into simple parts that are essentially two-dimensional. These parts are then translated, warped, or rotated to make three-dimensional meshes. Great care must be taken to ensure that the simple three-dimensional pieces can be "glued" together to form a coherent three-dimensional mesh.

There are two tools that are commonly used for mesh creation in ALEGRA: **FASTQ/GEN3D**, and **CUBIT**. In addition, the products of both tools can be combined with **GJOIN** to create an aggregate mesh.

FASTQ is a two-dimensional meshing tool. Using this tool, a user creates point data, connects points to make lines and then connects lines to make blocks (regions) that can be meshed. The two-dimensional meshes can be rotated, translated and warped by **GEN3D** to produce a three dimensional mesh. Three-dimensional meshes can be combined with **GJOIN**. The majority of the time required to produce a complete three-dimensional mesh goes into assuring that the nodes of the three-dimensional sub-meshes match up at the connecting surfaces. Discontinuous mesh can be handled with the contact library used in ALEGRA. The user must ensure that where contact is to be used, there are separate, unique nodes belonging to each side of the interface and that these nodes have not been merged in some fashion.

CUBIT is a current mesh generation project at Sandia. The ultimate goal for **CUBIT** is to be able to take solid model geometry as input and provide a quality three-dimensional mesh based on user-defined tolerances. At present, **CUBIT** provides the functionality of **FASTQ**, **GEN3D** and **GJOIN** in a single package along with many enhancements and the ability to automatically mesh simple solid model geometries.

For execution in parallel, the mesh created by the preprocessing tools must be decomposed for the number of processors the run is to be executed on. This decomposition is performed using the **loadbal** script, which in turn uses the **NEMESIS** utilities.The minimum information required by the **loadbal** script is the number of processors on which the run will take place and the *runid* of the problem. For example,

```
loadbal -p 7 runid
```

would break the mesh up for 7 processors for the default parallel platform. Additional options can be reviewed by running loadbal with no parameter (or the -h parameter).

2.2.2 Problem Specification File

ALEGRA uses a free-formatted ASCII input file to control the execution of the code. The units are assumed to be CGSK (cm-gm-sec-^{o}K). However, this can be changed with the user keyword UNITS. Note that output from tracers to the HISPLT database is always in SI units.

There is a one-to-one correspondence between the problem specification file and the mesh file. Integer identifiers used for blocks, nodesets and sidesets can be used in the problem specification to attach attributes. Chapter 3 is devoted to this subject.

2.2.3 Executing ALEGRA

ALEGRA is invoked using the **Alegra** script:

`Alegra` *runid*

where *runid* is a character string chosen by the user to identify the problem. This string is used to construct the names of the files that will be used for the calculation. For example, the command

```
Alegra run_id
```

will cause ALEGRA to look for a binary file named `run_id.gen` containing the problem mesh and a text file named `run_id.inp` containing the problem specification. **Figure 1** depicts the files ALEGRA will input and output. **Table 4** describes the contents of each file: The files

Table 4: ALEGRA Input and Output Files

File Name	Description
`run_id.inp`	Problem specification. This file contains specifications on how long to run the problem, how many output dumps to make, what materials belong to the mesh blocks, how the mesh blocks behave and what physics package to run. Most of this document deals with the contents of this file. **(INPUT)**
`run_id.gen`	GENESIS database. This file contains the description of the finite element mesh. A user can generate this mesh using various tools (**FASTQ**, **GEN3D**, **CUBIT**, etc.). This file contains the number of mesh blocks, the topology of the mesh, and the node and element sets that boundary conditions can be applied to. The mesh block, element set, and nodeset id's have a one-to-one correspondence with those used in the problem specification.**(INPUT)**
`run_id.nem`	NEMESIS file. Describes to the NEMESIS NEM_SPREAD utility how the finite element mesh is to be decomposed onto N processors. This file is produced as a result of running the **loadbal** script. **(INPUT, Parallel)**
`run_id.spd`	Spread script. A script produced by the **loadbal** script and run by the **Alegra** script to send portions of the .gen database to each of the N parallel processors, using the NEM_SPREAD utility.**(INPUT, Parallel)**
`run_id.cfg`	Configuration file. A text file produced by the **loadbal** script that is used by the NEMESIS utilities to both spread the files before the run and to combine the output files after the run. Its presence is a sign to the **Alegra** script that this is a parallel run.**(INPUT, Parallel)**
`run_id.cmd`	Termination signal file. Using the Unix "touch" command to create a file of this name in the running directory will cause the code to terminate gracefully. **(INPUT)**
`run_id.nqs`	Batch job file. As produced by **loadbal**, this file is targeted to the NQS system on the Tflop machine. The script contains a commented batch system submission command that can be stripped out and entered on a command line. The command submits the script to the batch system, which then runs the **Alegra** script for this problem file. The nqs file can be modified to be used on any batch execution environment**. (INPUT, Parallel)**

Table 4: ALEGRA Input and Output Files (Continued)

File Name	Description
`run_id.ech`	Problem specification echo. This file contains an echo of the problem specification. If errors occur in processing the problem specification, this file will point out the trouble spots. **(OUTPUT)**
`run_id.out`	ASCII output file. This file contains a detailed list of options set by the user and options set by the code. In addition, this file provides initial mass, momentum and energy associated with each mesh block. Finally, this file records when output is written to other files.**(OUTPUT)**
`run_id.exo`	EXODUS database. This file contains both the GENESIS database and all element and node transient data requested by the user. By default, this file will contain nodal displacements and element volume fractions. Additional data must be requested in the problem specification file.**(OUTPUT)**
`run_id.his`	HISPLT database. This file contains global, material and tracer location transient data. The HISPLT code can be used to extract this data and produce plots.**(OUTPUT)**
`run_id.dpl`	List of restart files. This ASCII file is generated by ALEGRA and contains a list of all restart dumps that have been written for the problem. Upon restarting the code, ALEGRA searches this list for the restart dump specified in the input file, by dump number or time.**(OUTPUT)**
`run_id.dmp.*`	RESTART database. A restart dump file, together with the original GENESIS database and problem specification, contains all the information required to restart ALEGRA from a given time. However, this file does not work with any plotting packages. If no EMIT RESTART keyword is present, ALEGRA will write a restart dump file only at the successful conclusion of a run. If an EMIT RESTART keyword is present, ALEGRA will by default retain only the last two restart dump files written. For serial runs, the dump files are named run_id.dmp.n and run_id.dmp.m, m>n, where run_id.dmp.m will always be at a later time. The user can specify how many restart dump files to retain via the RESTART DUMPS keyword. These files can be very large. For parallel runs, the files are named run_id.dmp.N.n.m, where N is the total number of processors used in the run, n is the number from 0 to (N-1) of a particular processor, and m is the dump number described above.**(OUTPUT) (INPUT, Restart)**

Table 4: ALEGRA Input and Output Files (Continued)

File Name	Description
`run_id.dbg`	DEBUG file. With the correct user commands, extensive debugging information can be obtained and is written to this file. This information can be very useful to code developers but is of little use to most users.**(OUTPUT)**

denoted with “Parallel” are only used when the code is run in multiprocessor mode.

The **Alegra** script has the following options:

Usage: Alegra [-12] [-a] [-hBQ] [-C buffer_size] [-spF] [-b <batch queue>] [-I pfs_mode] [-H hostfilename] [-x executable] runid

Options:

1 - Mode 1 (default). Diagnostics written to screen (UNIX “standard out”)

*2 - Mode 2. Diagnostics written to *.con.*

a - Use Aprepro to preprocess the input file before running ALEGRA (See note below concerning Aprepro)

h - Print this message

B - Beep to signal when calculation is complete

Q - Quick look at user input correctness and mesh quality

C buffer_size - For some architectures this changes the default buffer size

F - Force rerun of nem_spread before running ALEGRA

s - run in serial

p - run in parallel (appropriate files must exist)

P - set proc mode for tflop; 0 = default; 1 proc/node; 3 = 2 proc/node

I pfs_mode - mode for using the pfs (TFLOPs specific)

x executable

- Use executable to perform the ALEGRA calculation. The default executable is defined by $ALEGRA_EXE

b batch queue

- batch queue to submit run to

H hostfilename

- list of machine names for use in parallel runs

(mpirun -machinefile option) A default file should have been created at the time of MPI installation. A personal default may be defined via the MPI_HOST_FILE environment variable.

Generally speaking, the p and s options are not needed. The **Alegra** script along with the *runid.cfg* file created by the loadbal script control how the ALEGRA executable is run.

Thus, the command line:

Alegra -2 -x /home/username/alegra.exe runid

would use alegra.exe in /home/username and write the output to a file named runid.con.

The APREPRO utility [37] is an algebraic preprocessor that reads a file containing both general text and algebraic, string, or conditional expressions. It interprets the expressions and outputs them to the output file along with the general text. The utility is part of the SEACAS suite of pre and post processors. This capability replaces the "INCLUDE" function that ALEGRA formerly supported.

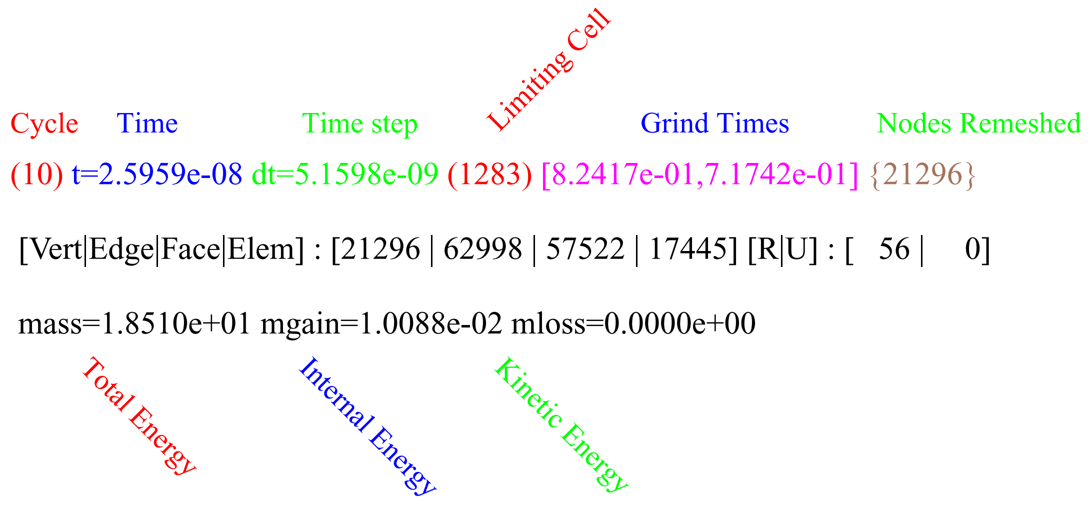
In the process of running ALEGRA, many messages are printed to the console (UNIX "standard out"). There are five types of messages:

- Information
- Warning
- Fatal
- Global Record
- I/O Generation

Generally speaking, the information and warning messages that appear during the initialization of ALEGRA should be ignored. However, in the process of matching the problem specification deck to the mesh file a warning message can be generated that later causes a fatal message. For example, the code initially warns the user that a Sesame file could not be found if one does not exist. If the user requests a Sesame equation of state and a Sesame file could not be found, the code will reach a point of creating the sesame equation of state object and the warning becomes a fatal message. When a fatal message is printed by ALEGRA, you should look through the warning messages to help determine the cause.

Fatal messages cause the code to stop executing. There can be many causes for this. If the user determines that the error is not due to the problem specification or mesh file, contact the ALEGRA team (alegra-help@sandia.gov or alegra-bugs@sandia.gov) and report the problem.

The global record message contains the total mass, total energy, internal energy, kinetic energy, time, time step, cycle, grind time (with I/O and without I/O) and the number of nodes remeshed since the last global record print. The record also shows a count of the number of nodes, edges, faces, and elements in the problem, as well as the refine and unrefine count. The latter two values are related to “h-adaptivity”, a research area of the code. The mass in the problem, mass loss and mass gain are also given. An annotated example of this is shown in **Figure 2**.



```
(10) t=2.5959e-08 dt=5.1598e-09 (1283) [8.2417e-01,7.1742e-01] {21296}

 [Vert|Edge|Face|Elem] : [21296 | 62998 | 57522 | 17445] [R|U] : [   56 |     0]

 mass=1.8510e+01 mgain=1.0088e-02 mloss=0.0000e+00
```

Figure 2: ALEGRA Global Record Output

I/O generation messages inform the user of when the code sends information to the disk drives.

2.2.4 Postprocessing

ALEGRA writes both EXODUS and HISPLT databases. The user can control the frequency at which the transient results are written to these files. The user can also control which variables are written to the EXODUS database. By default, mesh displacements and material volume fractions are written to EXODUS. HISPLT contains global, material and tracer particle information. The user can specify the number, type and location of tracer particles. Tracer points must be specified in the user input in order to have a valid HISPLT database. Variable included in the EXODUS and HISPLT databases can now be separately specified.

The EXODUS data base can be visualized by **BLOT**, **MUSTAFA**, **Eigen-VR** and **Ensight**. Most external users will use **BLOT** due to its portability. **MUSTAFA** is an SNL developed tool based on AVS Express and provides the most robust visualization environment for ALEGRA. **Ensight** is a commercially available tool that supports input of EXODUS formatted files. It is currently used for ALEGRA’s largest parallel simulations.

The HISPLT database is post-processed with the **HISPLT** code. Using the *runid*.hin file (user created input file) **HISPLT** creates a series of x-y plots that can be viewed by a number of device drivers. **HISPLT** and the device drivers are part of the CTH distribution.

Table of Contents
Overview

2.3 Problem Example

This section presents an example of mesh construction, problem specification and graphical output of an ALEGRA run, the 3D Lagrangian Taylor Anvil Impact problem. Numerous other examples are contained in the ALEGRA benchmark directories. These problems will be documented in the ALEGRA Verification document which is currently under construction.

3D Lagrangian Taylor Anvil Impact

The Taylor Anvil impact problem is a validation benchmark for both solid dynamics algorithms and constitutive models. In this problem, a solid cylindrical bar strikes a rigid target. Although this problem is two-dimensional axisymmetric, a quarter cylinder, three-dimensional mesh can be used to demonstrate the steps in using ALEGRA for three-dimensional Lagrangian calculations. The problem parameters are given in **Figure 3**.

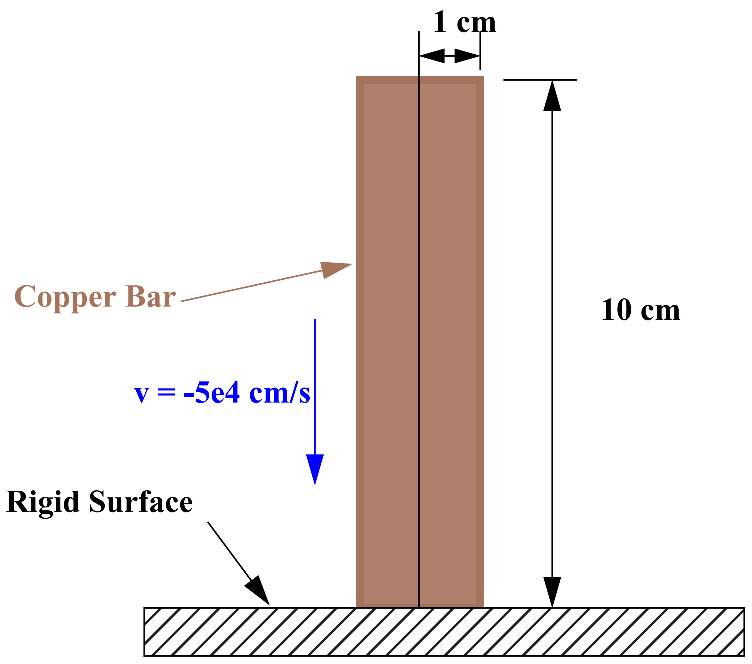


Figure 3: Taylor Anvil Simulation

The first step in running this problem requires building a mesh. For a three-dimensional mesh, this is a two-step process: 1) create a two-dimensional mesh and 2) rotate the two-dimensional mesh to create a three-dimensional mesh. Using **FASTQ** for the two-dimensional mesh, one can interactively input point data, create lines and regions and then mesh the region. **FASTQ** can also read this data from a text file. For this problem, only four points are needed to create the two-dimensional mesh: (0, 0), (0,1), (1, 10), and (0, 10). These four points are then connected to form four lines. The four lines are connected to form a block. When creating the lines, the user will input the number of intervals desired on a line. In general, parallel lines will have the same number of intervals. In addition, the user will group lines to form boundary sets. For this problem the **FASTQ** input file, `taylor.fsq`, would look like:

```
POINT     1      0.0000000E+00      0.0000000E+00
POINT     2      1.0000000E+00      0.0000000E+00
POINT     3      1.0000000E+00      1.0000000E+01
POINT     4      0.0000000E+00      1.0000000E+01
LINE      1   STR     1     2     0      2 1.0000
LINE      2   STR     2     3     0     20 1.0000
LINE      3   STR     4     3     0      2 1.0000
LINE      4   STR     1     4     0     20 1.0000
REGION     1     1     -1    -2    -3    -4
SCHEME    0 M
BODY       1
ELEMBC     1     1
ELEMBC     2     4
EXIT
```

Graphically, this file is depicted in **Figure 4**.

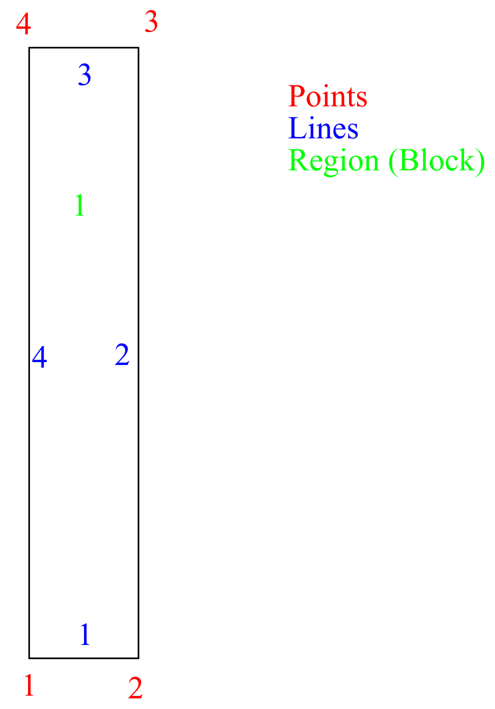


Figure 4: Point, Line and Block ids for Taylor Anvil Problem

Table of Contents
Overview

The meshed block is shown in **Figure 5**.

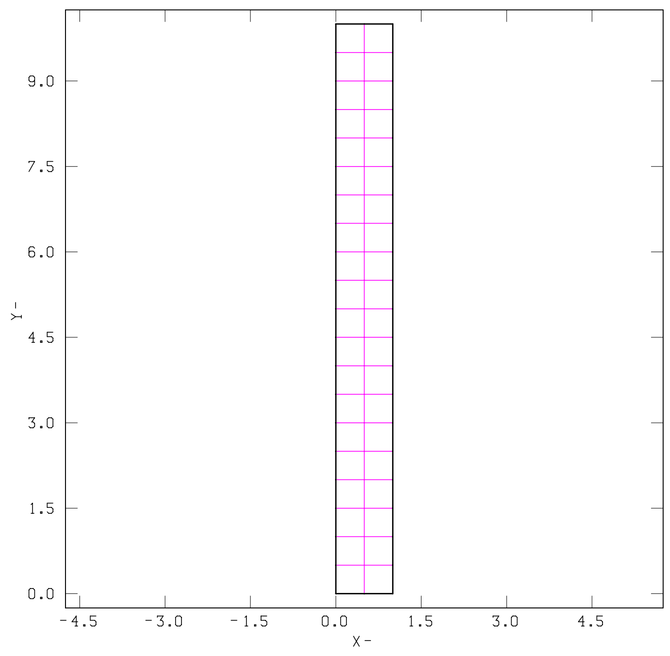


Figure 5: A Coarsely Meshed Taylor Anvil Problem

The following command generates a two-dimensional mesh file from the above taylor.fsq file:

```
fastq -m taylor.g2 taylor.fsq
```

The two-dimensional mesh is now stored in the file taylor.g2. The next step requires using this mesh as input to **GEN3D**. The following command begins the process

```
gen3d taylor.g2 taylor.gen
```

Using the commands below, **GEN3D** generates a three-dimensional quarter cylinder with boundary sets on xy and yz planes and saves the resulting mesh in the file, taylor.gen.

```
rotate 4 90
ssets front 5
ssets back 6
exit
```

These commands may also be entered into a data file, taylor.g3d, and used as input to **GEN3D.** The above command would then become:

```
gen3d taylor.g2 taylor.gen < taylor.g3d
```

The resulting mesh is shown in **Figure 6**.

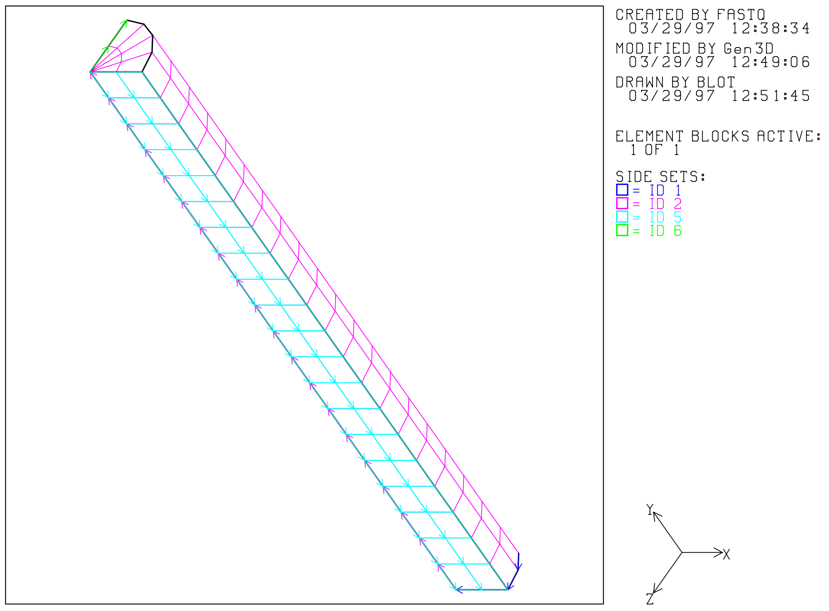


Figure 6: Three-dimensional Taylor Anvil Mesh

With the mesh completed, the next step is to create the problem specification deck. Chapter 3 is devoted to describing the commands that can be used in running ALEGRA. In this problem, the user needs to describe the boundary conditions and how the material should behave. At this point one knows that:

1. The impacting material is copper. Therefore, it has strength and thus solid dynamics is the correct physics. For this problem, an elastic-plastic constitutive model and a Mie-Grüneisen u_s-u_p equation of state will suffice.
2. XZ, XY, and YZ planes cannot displace (ids 1, 5, and 6)
3. Block 1 has a velocity of (0, -5e4, 0)
4. Deformation probably will not be large and thus a Lagrangian mesh motion treatment will work. Selection of Lagrangian, ALE, or Eulerian mesh motion becomes more apparent with experience.

ALEGRA uses the concept of domain, blocks, materials, and material models. The domain describes attributes that apply to all of the blocks. A block can have several materials and has attributes that describe how the nodes that form the elements in the block will behave. Each material can have several models. Boundary conditions and I/O control form other areas of the problem specification deck. **Figure 7** depicts an annotated problem specification deck for this problem.

```
title
 Taylor Impact
  termination cycle 200
```

Job Control

```
  emit SCREEN: cycle interval 10
  emit plot:   cycle interval 20
```

I/O Control

```
solid dynamics
```

Physics to be run

```
    no displacement, sideset 1, y
    no displacement, sideset 5, z
    no displacement, sideset 6, x
```

Kinematic Boundary Conditions

```
   initial block velocity: block 1 y -5.0e4
```

Initial Conditions

```
   domain
   end
```

“Default Domain Attributes”

```
   block 1
    lagrangian mesh
    material 1
   end
```

Block 1 has material 1 and its nodes will behave Lagrangian

```
 end
```

End of Physics Specification

```
material 1
 model = 100
 model = 2
 density = 8.932   $g/cm^3
 temperature = 298. $K
end
```

Material 1 has initial density and temperature. In addition the stress tensor will be evaluated with constitutive model 100. Pressure, temperature and sound speed will be evaluated with equation of state 2

```
model, 100, elastic plastic $Copper
 youngs modulus      = 1.076e+12   $dyne/cm^2
 poissons ratio      = 0.355
 yield stress        = 6.0e+09     $dyne/cm^2
 hardening modulus   = 2.0e+09     $dyne/cm^2
 beta                = 0.5
end
```

This constitutive model is elastic plastic and is populated with Copper’s material parameters.

```
 model 2, keos miegruneisen
 matlabel = ‘COPPER’
 end
```

This equation of state model is that specified in the EOS_data file for the material labelled ‘COPPER’.

```
exit
```

Figure 7: Taylor Anvil Annotated Problem Specification Deck

It is usually easiest to create a new problem specification by modifying an existing problem.This problem is run for 200 cycles using default parameters for time step control, artificial viscosity, and hourglass control. In the process of the simulation, plot dumps are added to `taylor.exo` every 20 time steps (cycles). In addition, global information records are written to the screen every 10 cycles. Since this input did not contain any additional plotting variable requests, only the mesh displacements and material volume fractions were written to `taylor.exo`. This data can be viewed with **blot** or **MUSTAFA**. An image of the deformed mesh from the last plot dump is given in **Figure 8**.

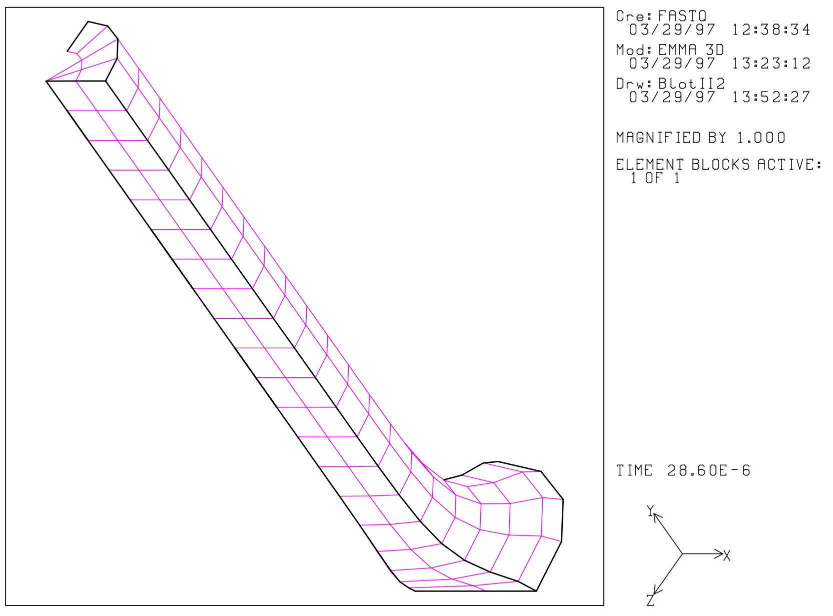


Figure 8: Taylor Anvil Deformed Mesh at 200 Cycles

2.4 Problem Reporting

The ALEGRA team uses a tool called DDTS to track problems reported by users and enhancement requests. The DDTS tool is available to users working within the SNL firewall and thus these users can enter problem reports directly. Users in other locations should send E-mail describing their problem to alegra-help@sandia.gov or alegra-bugs@sandia.gov. Please send a description of the problem, an input file, files to generate the mesh (FASTQ, GEN3D, or CUBIT), and any output that might help diagnose the problem, such as a capture of standard out during the run.

2.5 New for Version 4.2

This section provides short descriptions of new features that have been made available since the last public release of the ALEGRA code in October 1999. These features include those made available in Version 4.1, which was a SNL-internal release.

1. New remesh triggers and weights have been added. Also the capability to combine remesh triggers is available. A solid angle trigger capability has also been added. The default behavior of the remesh package has been changed so that the angle and volume remesh triggers are no longer turned on by default.
2. A newer version of the SMYRA interface tracker is available for experiment by users, as an option.
3. Diagnostic information has been added to the output that gives insight into the time step control used by the code during a simulation.
4. Cell Doctor and the Discard option have been made operational again.
5. Version 4.2 includes the first release of the capability to load balance multiprocessor simulations. This allows the h-adaptive features of ALEGRA to be used in a production environment. The ability to run problems with advection and h-adaptivity is now also ready for use.
6. An intermaterial fracture capability has been added to the material models.
7. The ACME contact capability is available in the code is the default service. The old legacy contact package is still available and is the only contact option for 2D problems.
8. The timed termination and I/O actions can now be exactly specified.
9. A degenerate surface option is provided that will collapse specified nodes to a center of convergence and enforce identical movement for all these nodes.
10. Options have been added to control the artificial viscosity terms individually.
11. There is now a separate specification for variable values to be placed on the HISPLT file, thus allowing the user to select what information is put on the file, as opposed to having all variables for each tracer point put there.

3.0 General Input

3.1 Overview

ALEGRA input is divided into four general categories: physics specification, material modeling, adaptivity specification and execution control. This section provides an overview of the input organizational structure and demonstrates the ways ALEGRA input can be organized with several examples. The ALEGRA input files should be organized into four sections for the four different categories of input. Any category can come first in the input file, but they can not be intermixed. A generic input file will look like the example shown below. Note that the adaptivity specification is an optional capability to control the H-adaptive feature of ALEGRA.

```
$Execution control Section
Title
  A very general input file
Output Control Keywords
Other Control Keywords

$Physics Specification Section
Physics Specification Keyword
$Begin Physics specification subsections
  Region
    $Region Keywords
  End

  Unstructured Region
    $Unstructured Region Keywords
  End

  Mechanics
    $Mechanics Keywords
  End

  Energetics
    $Energetics Keywords
  End
```

```
$ Continuation of the example input file
   Dynamics
       $Dynamics Keywords
    End

    $Other Physics subsections
$ End of Physics sub-section specification
End $ of the Physics specification

$ Adaptivity Parameter Input
Adaptivity Specification
    Enable Adaptivity
$ Adaptivity control parameters
End $ of the Adaptivity specification

$ Material Modeling Section
Material 1
   Model 1
   Model 2
    $Additional model specifications and variable initializations for Material 1
End

Model 1
    $Model 1 Keywords
End

Model 2
    $Model 2 Keywords
End

$Additional Material and Model specifications

exit
```

In this very general input file, the ALEGRA keywords for each subsection (i.e., level) of the physics specification are segregated into their own section. As we shall see later, this division is not necessary. It is done here to help explain the organization of this manual, which reflects the organization of the ALEGRA simulation object hierarchy. As described in **Section 3.2 on page**

53, the “$” is the comment character in ALEGRA input files. **Section 3.4.1 on page 74** describes the legal physics specification keywords. Keywords available for the control of the physics specification are given in the sections following **Section 3.4 on page 74**. The order of entry of the various physics levels is of no importance. A specific example of an input file is shown below.

```
TITLE
  SOD PROBLEM

SOLID DYNAMICS

   BLOCK 1
    MATERIAL 1
  END

  BLOCK 2
    MATERIAL 2
  END

  TRACER POINTS
    LAGRANGIAN TRACER 1 X= 0.795 Y= 0.1
    LAGRANGIAN TRACER 2 X= 0.895 Y= 0.1
    LAGRANGIAN TRACER 3 X= 0.995 Y= 0.1
    LAGRANGIAN TRACER 4 X= 1.005 Y= 0.1
    LAGRANGIAN TRACER 5 X= 1.105 Y= 0.1
    LAGRANGIAN TRACER 6 X= 1.205 Y= 0.1
  END

  MECHANICS
     NO DISPLACEMENT, NODESET 1, Y
     NO DISPLACEMENT, NODESET 2, X
  END

  DYNAMICS
      PRONTO ARTIFICIAL VISCOSITY
         LINEAR = 0.15
         QUADRATIC = 1.2
      END
  END

END

TERMINATION TIME 0.085
$TERMINATION TIME=0.5
EMIT PLOT, TIME=0.005
EMIT HISPLT, TIME=0.005
```

```
PLOT VARIABLES
  no underscores
  VELOCITY, as "VEL"
  PRESSURE: AVG
  DENSITY: AVG

$ the following added to test min/max variable plots
  DENSITY:  MAX, AS "DENSI_MX"     $ material scalar
  DENSITY:  MIN, AS "DENSI_MN"
  VOLUME          $ element scalar
  VOLUME:  MAX, AS "VOLUM_MX"
  VOLUME:  MIN, AS "VOLUM_MN"
  MASS         $ vertex scalar
  MASS: MAX
  MASS: MIN
  VELOCITY:  MAX, AS "VEL_MX"     $ vertex vector
  VELOCITY:  MIN, AS "VEL_MN"
END

MATERIAL 1
  MODEL 1
END

MODEL 1 IDEAL GAS
  GAMMA 1.4
  RHO REF 1.0
  CV 2.066E7
  TREF 1.21E-7
END

MATERIAL 2
  MODEL 2
END

MODEL 2 IDEAL GAS
  GAMMA 1.4
  RHO REF 0.125
  CV 2.066E7
  TREF 9.68E-8
END

crt: off
EXIT
```

As a convenience to users, the segregation of physics specification control keywords into their subsections (Energetics, Dynamics, etc.) is **NOT** required in the ALEGRA input file. The only physics keyword that must be present is that which describes the most specific characterization of the physics models, the options for which are found in **Section 3.4.1 on page 74**. An example of this feature is shown below, where the previous input file has been cast into this simpler format. Note that the *NO DISPLACEMENT* keywords are not placed within a *MECHANICS* keyword group, and that the *PRONTO ARTIFICIAL VISCOSITY* keyword group is not placed within a *DYNAMICS* keyword group, since in both cases, they already appear within the *SOLID DYNAMICS* keyword group.

```
TITLE
  SOD PROBLEM

SOLID DYNAMICS

   BLOCK 1
    MATERIAL 1
  END

  BLOCK 2
    MATERIAL 2
  END

  TRACER POINTS
    LAGRANGIAN TRACER 1 X= 0.795 Y= 0.1
    LAGRANGIAN TRACER 2 X= 0.895 Y= 0.1
    LAGRANGIAN TRACER 3 X= 0.995 Y= 0.1
    LAGRANGIAN TRACER 4 X= 1.005 Y= 0.1
    LAGRANGIAN TRACER 5 X= 1.105 Y= 0.1
    LAGRANGIAN TRACER 6 X= 1.205 Y= 0.1
  END

   NO DISPLACEMENT, NODESET 1, Y
   NO DISPLACEMENT, NODESET 2, X

    PRONTO ARTIFICIAL VISCOSITY
      LINEAR = 0.15
      QUADRATIC = 1.2
    END

END

TERMINATION TIME 0.085
$TERMINATION TIME=0.5
EMIT PLOT, TIME=0.005
EMIT HISPLT, TIME=0.005
```

```
PLOT VARIABLES
  no underscores
  VELOCITY, as "VEL"
  PRESSURE: AVG
  DENSITY: AVG
$ the following added to test min/max variable plots
  DENSITY:  MAX, AS "DENSI_MX"     $ material scalar
  DENSITY:  MIN, AS "DENSI_MN"
  VOLUME          $ element scalar
  VOLUME:  MAX, AS "VOLUM_MX"
  VOLUME:  MIN, AS "VOLUM_MN"
  MASS          $ vertex scalar
  MASS: MAX
  MASS: MIN
  VELOCITY:  MAX, AS "VEL_MX"     $ vertex vector
  VELOCITY:  MIN, AS "VEL_MN"
END

MATERIAL 1
  MODEL 1
END

MODEL 1 IDEAL GAS
  GAMMA 1.4
  RHO REF 1.0
  CV 2.066E7
  TREF 1.21E-7
END

MATERIAL 2
  MODEL 2
END

MODEL 2 IDEAL GAS
  GAMMA 1.4
  RHO REF 0.125
  CV 2.066E7
  TREF 9.68E-8
END

crt: off
EXIT
```

3.2 Format and syntax

ALEGRA takes as input a text file containing free format lines built around keywords or keyword groups. With few exceptions, the keywords or keyword groups may be in any order the user finds convenient. A keyword is a short sequence of English words denoting some action or quantity. For example,

```
TITLE
PISCES HOURGLASS CONTROL
VELOCITY VECTOR
```

are all examples of keywords. Keywords may or may not require a numeric field or other grammatical construct to follow it (see the next subsection). **Adjacent keywords must be separated by a comma, colon, semicolon, equals sign, (",", ":", ";", "=") or newline; a blank is sufficient ONLY to separate a keyword from a numeric field, not one keyword from another. (These delimiters may not always appear explicitly in the command descriptions that follow.)** The user may optionally separate keywords and numeric fields using blanks, commas, semicolons, colons, newlines, or equal signs as seems appropriate. **The number of characters on an input line is limited to 160**. Placing more characters on a line can lead to platform-dependent results.

A keyword group is a sequence of keywords and numeric values bounded by a main keyword and the keyword *END*. All keywords within a keyword group may be in any order. For example:

```
PLOT VARIABLES
  PRESSURE, DENSITY, VONMISES
END

MODEL 4 LINEAR ELASTIC
  YOUNGS MODULUS = 2.71E8
  POISSONS RATIO = 0.25
END
```

The input routines are case insensitive and only enough characters of each word of a keyword need be entered to uniquely identify it. The number of words per keyword is significant and varies according to the specific keyword or keyword group. In the input syntax descriptions that follow, all keywords will be presented in italicized upper case, while common grammatical constructs and numerical parameters whose values are supplied by the user will be shown in italicized lower case. Optional keywords, constructs, or parameters will be enclosed in square brackets. Alternative choices for a keyword may be enclosed in curly braces and separated by an OR symbol, "|", as in {*ABC* | *DEF*}, meaning *ABC* or *DEF*.

Users may enter comments at any point by using a dollar sign or asterisk ("$" or "*"). All text that follows a dollar sign on a line is ignored. Any line may be continued and lines may be combined. For example:

```
$ MATERIAL MODEL SPECS
MODEL 1 ELASTIC PLASTIC $ALUMINUM
```

New users may wish to start their input files with the keyword

```
DEBUG MODE, PARSER, END
```

which causes ALEGRA to print more extensive diagnostics of any errors it detects in the input.

3.2.1 Common Parameter Constructs

There are a number of common grammatical constructs that are used by more than one keyword. These constructs are described here, and they will be rendered in italicized lower case in the descriptions of the keywords that use them. Users should replace the construct with the keywords described here. Remember that if a construct directly follows a keyword, an appropriate delimiter (e.g., a comma or equals sign) must separate them.

Other parameters that must be supplied by the user will also be presented in italicized lower case in the input descriptions and should be replaced by the appropriate data type. Integer fields are specified by *int*, floating point fields are specified by *real*, and character strings are specified by *string* or some other descriptive name. Default values for these parameters may be shown in parentheses following the descriptive name (e.g. *real (2.5)* indicates that the default value of the real valued parameter is 2.5). All other options or sub-keywords will be discussed in the keyword descriptions and summarized in lists or tables.

3.2.1.1 block-id

BLOCK int

The GENESIS mesh file format used by ALEGRA assigns material numbers to subsets of the mesh. ALEGRA uses these material numbers to identify element blocks. The input specification file uses the *block-id* construct to identify these block numbers in keywords that specify material properties or initial conditions. For example:

```
INITIAL BLOCK VELOCITY, BLOCK 3, X = 20.0
```

The analyst must make sure that the proper block numbers are assigned during mesh generation. A reference to a nonexistent block number will cause ALEGRA to reject the input

3.2.1.2 block-ids

BLOCK int {int {int ...}}

This is similar to the *block-ids* keyword, except that any number of blocks may be specified.

3.2.1.3 nodeset

NODESET int

The *nodeset* construct specifies a nodeset from the GENESIS mesh file. It is used in keywords that apply to a set of nodes, such as a nodal boundary condition. For example:

```
NO DISPLACEMENT, NODESET 101, Y
```

The analyst must make sure that the proper nodesets are defined during mesh generation. A reference to a nonexistent nodeset will cause ALEGRA to reject the input.

3.2.1.4 sideset

SIDESET int

The *sideset* construct specifies a sideset from the GENESIS mesh file. This sideset can be used in keywords that must specify a set of edges (2D) or faces (3D), such as a traction boundary condition. For example:

```
PRESSURE BC, SIDESET 23, 0.25003
```

Sidesets are best thought of in terms of an element and associated face (3D) or edge (2D) so that an orientation for the sideset surface is implied. Sidesets interior to the mesh can be either single or double sided and correct usage depends on the particular application.

3.2.1.5 function-set

FUNCTION int [SCALE real] [SHIFT real]

or

real [SCALE real]

The *function-set* construct specifies a user-defined function table with an optional scaling factor and shift. If a keyword requires a function-set but only a constant value is needed, the function-set construct may take the form of a single real value. The scale option may be applied to this value.

ALEGRA makes extensive use of user-defined functions that are specified in the input specification file. (See the *FUNCTION* keyword on **page 58**.) These can be referenced one or more times by other keywords that require a functional specification of time or spatial dependencies. For example:

```
PRESCRIBED FORCE, NODESET 4, X, FUNCTION 1 SCALE 0.0005
```

Some keywords that use function sets support the *SHIFT* option. This option allows the user to specify the start time for the function data if it is different from the start time of the simulation. That is, the function is evaluated at time $t - t_{SHIFT}$. For example:

```
FUNCTION 4 SHIFT 0.001
```

The function referenced by a keyword need not appear before the keyword, but must appear somewhere in the input specification file.

3.2.1.6 vector

X real [Y real [Z real]]

R real [Z real]

The *vector* construct specifies a 1-, 2- or 3-dimensional vector. For example:

```
INITIAL VELOCITY, NODESET 10, X=0.300009 Y=0.033540 Z=0.699963
```

The second form of the *vector* construct applies only to 2D cylindrical simulations.

3.2.1.7 vector-function-set

X, function-set [Y, function-set [Z, function-set]]

The *vector-function-set* construct specifies a 1-, 2- or 3-dimensional vector function. Each component of the vector function is a separate ALEGRA *function-set* specification. For example:

```
  X, FUNCTION 1, Y, FUNCTION 2, Z, FUNCTION 3
```

3.2.1.8 symtensor

XX real XY real YY real

or

XX real XY real XZ real YY real YZ real ZZ real

The *symtensor* construct specifies a 2- or 3-dimensional symmetric tensor. The first form should be used for 2D and the second form for 3D. For example, in 3D:

```
    TRACTION BC, SIDESET 3,
   XX 0.0 XY 1.0 XZ 0.0
          YY 0.0 YZ 0.0
                 ZZ 0.0
   FUNCTION 3 SCALE 1.0
```

3.2.1.9 direction-function

{X | Y | Z | R}, function-set

or

{RADIAL | NORMAL | TANGENT}, function-set, vector, [CENTER, vector]

The *direction-function* construct combines a direction with a function that is to be applied. In the first form a direction along one of the primary axes is specified, namely *X*, *Y* or *Z* in Cartesian geometry or *R* or *Z* in 2D cylindrical geometry. This is followed by a *function-set*. For example:

```
  PRESCRIBED FORCE, NODESET 4, X, FUNCTION 1 SCALE 0.0005 [DYNE]
```

In the second form a direction relative to a reference point is specified, namely *RADIAL*, *NORMAL* or *TANGENT*. This is followed by a *function-set* and a *vector*. Lastly comes the reference point as specified by the optional *CENTER* keyword. It is assumed to be the origin if omitted. For example:

```
  PRESCRIBED VELOCITY, RADIAL, FUNCTION 2, X 1.0 Y 1.0, CENTER, X
0.0 Y 0.0
```

3.2.1.10time-or-cycle-interval

[EXACT] TIME [INTERVAL] real

or

CYCLE [INTERVAL] int

The *time-or-cycle-interval* construct specifies an interval of time or interval of cycles used by the keyword. Note that time intervals are specified as floating point values and cycle intervals are specified as integer values.

The extra word *EXACT* is optional for the time interval. If specified, ALEGRA will modify the timestep so that output is emitted at the exact interval. Otherwise the default behavior of ALEGRA is to emit output when a time interval is matched or exceeded. The intent of the *EXACT* modifier is to synchronize long running verification simulations on various computer platforms. *EXACT* is available for the *EMIT PLOT*, *EMIT HISPLT*, *EMIT PDS* and *EMIT RESTART* commands.

The extra word *INTERVAL* is optional and may be included for readability. For example:

```
  EMIT PLOT, TIME INTERVAL 0.1
  EMIT OUTPUT, CYCLE INTERVAL 25
```

3.2.1.11 time-range

FROM [TIME] real TO real

The *time-range* construct specifies a range of time values for which a keyword will apply. The extra word *TIME* is optional and may be included for readability. For example:

```
  EMIT PLOT: CYCLE INTERVAL 1, FROM TIME 0.0 TO 0.3e-6
```

3.2.2 EXIT

EXIT

The *EXIT* keyword signals the end of processing for an input specification file. ALEGRA will ignore the remaining contents of the file. This allows an extended explanation of the problem to be placed at the end of the input (a highly recommended practice).

3.2.3 FUNCTION

FUNCTION int

real real

real real

... ...

END

The *FUNCTION* keyword group defines a function in terms of a table of *x*-*y* *real* pairs. This function, identified by the integer field after the *FUNCTION* keyword, can then be referenced by other keywords used to define the problem, and so may be used to specify such things as boundary conditions as functions of time or space. Linear interpolation is used between table points. To approximate steps in a function, the user should use *x*-values that are close together. For example:

```
FUNCTION 11
  0.0    0.0
  1.0    0.0
  1.0001 1.0
  2.0    1.0
END
```

ALEGRA provides a predefined function with id 0, equivalent to

```
FUNCTION 0
  -BIG_DBL/2.  1.
   BIG_DBL     1.
END
```

that allows a constant function to be referenced conveniently in the code input. Here "BIG_DBL" is the largest floating point number available to the code. Since FUNCTION 0 has been predefined, the user cannot use 0 as a function id when defining additional functions.

If automatic unit conversion is desired (**Section 3.2.5 on page 59**), ***the unit conversion string must be entered after each applicable real value in the function***.

3.2.4 TITLE

TITLE

string

The line of input text following a **TITLE** keyword is used as a title string for the problem. Currently, only eighty characters are recorded. For example:

```
TITLE
  CASE 4 WITH 40,000 ELEMENTS
```

3.2.5 UNITS

UNITS, {CGS | SI}

Default units for ALEGRA are cgs units (cm, g, sec, °K). However, the UNITS keyword may be used to change the default units to SI (System International, m, kg, sec, °K). Default temperatures are always Kelvin. An example:

```
UNITS, SI
```

Units for individual numerical (float) quantities can be entered in the input immediately following the value. This functionality is applicable for all keywords that accept a floating point argument ***except those in the DIATOMS input section***. The units specified will be used to convert the float value to the run units according to a set of basic fundamental unit exponents (mass, length, time, and temperature). The units labels and definitions that can currently be parsed by Alegra are shown in **Table 5**. The unit label designation of eV (temperature associated with electron volts) is included for compatibility with CTH input variables.

To be in a readable format, the units string must be entered with square brackets []. The following operators are recognized: parentheses () for multiplication, the forward slash / for division, and the carat ^ for exponentiation. Units are not required for any keyword in the input set, but are merely intended as a user convenience. An example of acceptable unit designations are shown below.

```
R0 2730.    [kg/m^3]
CV 1.4e11   [erg/((gm)(eV))]
cv 1.1e11   [erg/gm/ev]
```

Table 5: Allowable Unit Designators

Full Name	Unit System Association	Allowable Designator	Fundamental Unit (L=length, M=mass, t=time, T=temperature)
meter	SI	m	L
kilogram	SI	kg	M

Table of Contents
General Input

Table 5: Allowable Unit Designators (Continued)

Full Name	Unit System Association	Allowable Designator	Fundamental Unit (L=length, M=mass, t=time, T=temperature)
second	SI, CGS	s	t
Kelvin	SI, CGS	K	T
centimeter	CGS	cm	L
Electron volt[a]		eV	T
ergs	CGS	erg	ML^2/t^2
dynes	CGS	dyn	ML/t^2
Newtons	SI	N	ML/t^2
Joules	SI	J	ML^2/t^2
Pascals	SI	Pa	$M/(t^2 L)$
Watts	SI	W	ML^2/t^3

a.Electron volts require conversion from either SI to CGS systems into Kelvin.

3.3 Execution Control

3.3.1 Job Initiation and Termination

3.3.1.1 READ RESTART

READ RESTART TIME real

or

READ RESTART DUMP int

In its first form, this keyword specifies the time at which ALEGRA is to read a restart dump to obtain initial values for all variables. The *real* value specified for the restart time may be positive or negative. The second form specifies the number of the actual restart dump to be read.

ALEGRA generates restart files with names of the form *"run_id".dmp.n* at time intervals specified by the *EMIT RESTART* keyword, where *n* is an integer, starting with 0. The number of dumps that are retained can be controlled through the *RESTART DUMPS* keyword. By default only the last two files generated will be retained. Earlier restart dumps are deleted as the calculation progresses to conserve disk space. For parallel runs, dump files are written for each processor and are named *"run_id".dmp.N.m.n*. Here N is the total number of processors, m is the number of an individual processor and n is the dump number, as above.

A dump list file named *"run_id".dpl* is generated that contains a list of all restart dumps that have been written for the problem. ALEGRA searches this list for the desired restart dump. The restart dump file to be read will be the one whose time is closest to the specified time, if the restart dump number is not specified directly. Thus one can specify a restart time that is close to the time at which the restart dump was written and that dump will be selected.

If a restart time greater than that of the last restart dump is specified, the last dump will be read. If a restart time less that the problem *START TIME* is specified, the last dump will be read provided the dump list file exists and contains a list. Otherwise, the calculation will begin as a new one starting at the initial time.

No restart dump is used if this keyword does not appear. Instead, a new calculation is started with all quantities set to initial values calculated from the input specification file.

Examples:

```
  READ RESTART TIME 26.5E-6
  READ RESTART DUMP 12
```

Some output files (e.g., *"run_id".out* and *"run_id".his)* generated from a restart will have an *"_n"* appended to their name, where *n* is selected to be unique in the directory in which the files are written, unless the *OVERWRITE FILES* option is chosen.

The *READ RESTART DUMP* keyword allows a negative dump number, e.g.,

```
        READ RESTART DUMP = -1
```

Specification of a negative dump number causes Alegra to check for a *.dpl restart file. If this file does not exist or is empty, then the simulation will begin as a new simulation. Otherwise, the simulation will restart at the latest available restart dump. This behavior is similar to specifying a restart time prior to the simulation start time on the *READ RESTART TIME* command.

At this time, ALEGRA does not permit the user to change the type of plot file upon restart. For example, a computation in which *EMIT PLOT* was used to generate EXODUS-II plot files may not be restarted with *EMIT PDS*. Similarly, a computation in which PDS files were generated may not be restarted with EXODUS-II plot files.

3.3.1.2 START TIME

START TIME real

This keyword specifies the starting time for the problem, if it is different from zero. The *real* value specified for the start time may be positive or negative. This option is useful if the simulation is driven by experimental data that may have a non-zero start time.

3.3.1.3 TERMINATION CPU

TERMINATION CPU real

This keyword specifies the maximum cpu time, measured in seconds, at which the problem calculation is to terminate. If any processor running a parallel ALEGRA calculation exceeds this limit then the whole calculation will shut down gracefully. Generally this option is used when running in batch mode. The termination cpu value should be set to a value smaller than the batch request time limit in order to allow the problem to terminate before it is killed by the system batch manager. The user must determine this value for a given run depending on the number of files that will be written at the end of the calculation and the expected cycle time since there is no cycle time estimation capability in the code.

The *TERMINATION CPU* keyword must be used in conjunction with either the *TERMINATION TIME* and/or a *TERMINATION CYCLE* keyword. *TERMINATION CPU* by itself is not sufficient and the problem will not run. If more than one of *TERMINATION CPU*, *TERMINATION TIME*, and *TERMINATION CYCLE* are specified, the problem will terminate when any of the criteria are satisfied. A restart record will be written when this option ends the run. A history record and an EXODUS/PDS dump will also be written if these have been specified with *EMIT* keywords. The termination of the run due to cpu limit will be noted in the user’s terminal window (i.e the “standard output” device/file).

3.3.1.4 TERMINATION CYCLE

TERMINATION CYCLE int

This keyword specifies the cycle on which the problem calculation is to terminate.

If more than one of *TERMINATION CPU*, *TERMINATION TIME*, and *TERMINATION CYCLE* are specified, the problem will terminate when any of the criteria are satisfied. This form of termination control can be used by itself. A restart record will be written when this option terminates the run. A history record and an EXODUS/PDS dump will also be written if these have been specified with *EMIT* keywords. Termination due to cycle limit will not be reflected in the user’s terminal window - the code will simply stop.

3.3.1.5 TERMINATION TIME

[EXACT] TERMINATION TIME real

This keyword specifies the problem time at which the problem calculation is to terminate. By default, Alegra may overshoot the termination time by some fraction of a timestep. If the optional prefix *EXACT* is added to this command, then the time steps for the last ten cycles will be adjusted as necessary to ensure that the calculation terminates at the exact time specified.

If more than one of *TERMINATION CPU*, *TERMINATION TIME*, and *TERMINATION CYCLE* are specified, the problem will terminate when any of the criteria are satisfied. This form of termination control can be used by itself. A restart record will be written when this option terminates the run. A history record and an EXODUS/PDS dump will also be written if these have been specified with *EMIT* keywords. Termination due to time limit will not be reflected in the user's terminal window (i.e the "standard output" device/file) - the code will simply stop.

3.3.2 I/O Control

3.3.2.1 CRT

CRT, {ON | OFF}

This keyword enables or disables the function which checks the keyboard buffer for any keystrokes. This method allows for termination of the calculation at the end of the current cycle when the user enters the word *STOP* into the keyboard buffer. Executive and query menus are also accessed by this method when the user enters the word *HELLO* into the keyboard buffer. The crt is on by default. However, this precludes straightforward background running of a calculation. So, to run a calculation in the background, the user should turn the crt function off, i.e. *CRT, OFF.*

3.3.2.2 DEBUG MODE

DEBUG MODE: {FILE | LOCATION | TRACKER | REMAP | PERIODIC BC | PROCSET | DUMP PROCSET | EVATTR | TIMESTEP | EXODUS | CONNECTIVITY | STORAGE | MFLOPS | RESTART | ADAPT(OR ADAPTIVITY) | PARSER} END

This keyword enables debugging information output. These options are generally used by developers for diagnostic purposes and are provided with only limited user support. Any or all of the flags may be turned on. The keyword *DEBUG MODE* must be followed by a terminator as described in **Section 3.2 on page 53**. An *END* **must** be placed after all debug modes have been specified.

FILE sends information to the *"run_id".dbg* output file on all processors and is used in conjunction with other debugging flags. Also turns on debug outputs associated with the opening of plot output files with the adaptivity option active.

LOCATION gives information on the current program step location.

TRACKER gives information on the interface tracker.

REMAP gives information on the remap step.

PERIODIC BC prints diagnostic information about nodesets used in the periodic boundary conditions

DUMP PROCSET forces extensive output dumps of processor set related information.

PROCSET gives information on the processor sets.

EVATTR outputs information on element and vertex attributes.

TIMESTEP gives time step control information.

EXODUS turns on EXODUS II warnings. (This option will cause the EXODUS library flag EX_VERBOSE to be turned on, leading to messages being printed from these EXODUS library routines.)

CONNECTIVITY turns on comprehensive connectivity checking of the topology database (vertex, edge, face, and element connectivity). If an invalid database is found the calculation is terminated.

STORAGE turns on memory usage information.

MFLOPS activates diagnostics for performance measurement. At this time, hardware performance measurement tools are only functional on the Sandia TFLOPs machine.

RESTART activates diagnostics for debugging restarts.

ADAPT activates h-adaptivity diagnostics.

PARSER turns on verbose mode in the parser. This will result in more information being delivered when errors occur in the parsing of the input. For example, the entire sequence of valid keywords will be written to the output following the error message, or suggestions about possible causes of the error will be printed.

COMM STATS will display information about the number and types of interprocessor communication. This display will take place at the end of the run.

For example,

```
  DEBUG MODE: LOCATION, PARSER, EXODUS, END
```

3.3.2.3 DOUBLE PRECISION EXODUS

DOUBLE PRECISION EXODUS

This keyword indicates that the GENESIS input file uses double precision for floating point numbers. The default is for the GENESIS file to use single precision for floating point numbers.

3.3.2.4 EMIT HISPLT

EMIT HISPLT, time-or-cycle-interval [time-range]

This keyword causes ALEGRA to emit a HISPLT record to the "*run_id.his*" file at specified intervals over an optionally specified time range. If no time range is specified, the records are emitted throughout the run at the specified time or cycle interval. The user can have multiple entries of this card, thereby creating a time table. The input specification can take one of three forms, depending on the form of the *time-or-cycle-interval* construct and whether the *time-range* construct is specified. For example:

```
  EMIT HISPLT, TIME INTERVAL = 0.0001
  EMIT HISPLT, TIME INTERVAL = 0.0003, FROM 0.001 TO 0.002
  EMIT HISPLT, CYCLE INTERVAL = 20
```

For the second example above, information will be written at time=0.001 and at times separated from 0.002 by 0.0003, but information will not be written for time=0.002. Thus, this example would produce HISPLT dumps at 0.001, 0.0011, 0.0014, and 0.0017. Several specifications of the second type can be used to setup a series of outputs at varying time intervals throughout the problem run. For the first and third specifications, however, only the first occurrence is used to set the interval of outputs for the entire run.

Note that at least one tracer point is required for creation of a valid HISPLT database. Other limitations are no more than 20 materials in a calculation and no more than 1002 tracer points [44].

3.3.2.5 EMIT OUTPUT

EMIT OUTPUT, time-or-cycle-interval [time-range] [PROFILE]

This keyword causes ALEGRA to emit various summary information to the "*run_id".out* file. If the *PROFILE* keyword is also included, then any profiling requests which have been compiled into the code will be also be output (currently this is at the *EMIT SCREEN, time-or-cycle-interval*).

3.3.2.6 EMIT PLOT

EMIT PLOT, time-or-cycle-interval [time-range]

This keyword causes ALEGRA to emit an EXODUS plot record at specified intervals over an optionally specified time range. If no time range is specified, the records are emitted throughout the run at the specified time or cycle interval. As explained in the *EMIT HISPLT* keyword **(Section 3.3.2.4 on page 65)**, the user can have multiple entries of this card, thereby creating a time table. EXODUS plot records are written in one of two possible forms. For an unchanging initial topology, such as topologies produced with no adaptivity, initial mesh refinement, or SHISM-pair generation, a traditional EXODUS file is produced: a GENESIS mesh database with subsequent time slice information appended for each plot event. For adaptive problems and conditions where the topology is not guaranteed to be constant, then individual EXODUS plot records are written to separate files. The current GENESIS mesh database corresponding to the EXODUS output is contained in these files. The single plot file has the form *"run_id".exo* while the multiple plot files have the form *"run_id".hat.n* where *n* corresponds to the *n*th plot dump.

3.3.2.7 EMIT RESTART

EMIT RESTART, time-or-cycle-interval [time-range]

This keyword causes ALEGRA to emit a restart dump at specified intervals over an optionally specified time range. If no time range is specified, the dumps are emitted throughout the run at the specified time or cycle interval. The user can have multiple entries of this card, thereby creating a time table, as explained in the *EMIT HISPLT* (**Section 3.3.2.4 on page 65**) description.

Restart dumps permit a user to break a single calculation into several runs. They also help to avoid losing everything if the computer crashes in the middle of a run.

If no *EMIT RESTART* option is specified, then the only restart dump that will be written is one at the end of the problem, if the problem terminates as planned.

The dump files will be named *"run_id".dmp.n*, where "run_id" is the prefix of the ALEGRA input specification file, i.e. the file name used on the Alegra execute line, and *n* is a dump number. The dump number begins with 0 for the first restart dump and is incremented by 1 each time a restart dump is written. By default, only the two most recent restart dump files are retained; earlier dump files are deleted as the calculation progresses to conserve disk space. However, the number of files retained can be controlled with the *RESTART DUMPS* keyword. A list of the times of all restart dumps is written to the file *"run_id".dpl*, which is then read when restarting to determine the proper dump file from the restart time specified.

For parallel runs, dump files are written for each processor and are named *"run_id".dmp.N.m.n*. Here *N* is the total number of processors, *m* is the number of an individual processor and *n* is the dump number, as above.

Restart dumps are operational with the *INITIAL REFINEMENT* option specified in the *DOMAIN* **Section 3.4.6.2 on page 96**.

3.3.2.8 EMIT SCREEN

EMIT SCREEN, time-or-cycle-interval [time-range] [PROFILE]

This keyword causes ALEGRA to emit a short summary of the state of the calculation at specified intervals over an optionally specified time range to the users terminal (i.e the "standard output" device/file) If no time range is specified, output is emitted throughout the run at the specified time or cycle interval. The user can have multiple entries of this card, thereby creating a time table, as explained in the *EMIT HISPLT* description. If the *PROFILE* keyword is also included then any profiling requests which have been compiled into the code will be also be output to the screen. If this keyword is omitted, the default screen output is to emit a summary every 10 cycles throughout the calculation.

3.3.2.9 EXODUS VERSION

EXODUS VERSION TWO

EXODUS VERSION TWO indicates that the GENESIS input file and the EXODUS output file are in EXODUS Version II format. The Alegra script that can be used to run ALEGRA will convert EXODUS version one files to version two format. The *EXODUS VERSION TWO* keyword is ignored in the input stream.

EXODUS VERSION ONE

This is an obsolete, but still legal, input keyword.

ALEGRA can still read EXODUS version one files by not using the Alegra script to run the code, running the code directly from the command line and using the *EXODUS VERSION ONE* keyword. Note that an immediate abort is caused if this keyword is in the input file and the script is used to run the code, since trying to read an EXODUS version two file with the *EXODUS VERSION ONE* keyword is immediately fatal. Note also that the Alegra script will convert EXODUS version one files to version two format and **overwrite the file in place**.

3.3.2.10OVERWRITE FILES

OVERWRITE FILES

This keyword causes output files from a preceding run to be overwritten on a restart. The default action is to not overwrite the plot and history files, instead appending an "_n" to an output file name on a restart, where n is an integer 0,1,2,... selected to be unique in the directory in which files are being written.

3.3.2.11PLOT VARIABLES

PLOT VARIABLES

  name [conversion]

  name [conversion]

  ...

END

This keyword permits the user to specify the variables to be included in the EXODUS or PDS plot file. By default, *VOLUME FRACTION* and the node displacements are the only variables written to the database. Note that if a variable name is repeated in the *PLOT VARIABLES* keyword group, then it will be written to the database twice.

With the exception of global variables, the name of any registered variable can be used as a value for *name.* Registration of variables is dependent upon which physics and material models are requested by the user in the input specification. For convenience, some more common names are given in the following tables. The variable types are also given to aid in the proper choice of conversions, if any. The list of registered variables for any given problem can be found in the "*run_id*".out file produced by a run of ALEGRA. Global variables, which can not be used a plot variable names, are listed as "REGION_VAR" in the listing in the "*run_id*".out file. The material variables available for plotting are very dependent upon the particular material models, so only the most generic are listed. The names of the variables available for plotting in each material model are given in the tables in **Section 3.9 on page 143**.

In the output EXODUS files, the variable names will appear and be used to specify what data is available for display by various postprocessors. The EXODUS variable names will be the same as the names included in the *PLOT VARIABLES* keyword group, except that blanks within a name will be replaced by underscores. An exception to this rule can be forced by the user. (See *NO UNDERSCORES* below). For material variables, the EXODUS name of the variable describing the value of the quantity for a given material is indicated by appending a "_N", where N is the material id number assigned by the user in the input file with the *MATERIAL* keyword. For example, the temperature of the material that is labeled 101 by the user is written to the EXODUS file with the name TEMPERATURE_101. For vector variables, such as *VELOCITY*, each vector component will be written to the EXODUS file with the name of the component direction appended. For example, the x component of velocity will be named VELOCITY_X. Tensor quantities are labeled similarly, as for example ARTIFICIAL_VISCOSITY_XY, for the xy component of the artificial viscosity tensor in an element. Tensor quantities that are specific to a material will be named with a "_N" appended to the variable name before the direction designation. For example, STRESS_202_XX will be the name of the xx component of stress in material 202 in an element. (The list of names of variables that are on any EXODUS files can be found by using the ACCESS GROPE utility and the LIST NAMES command.)

Those variables marked with an asterisk (*) are not normally stored for the entire grid; inclusion of these in the data base causes ALEGRA to allocate additional dynamic memory for their storage.

Valid values for the optional *conversion* keyword are:

{*VOLUME AVERAGE (AVG)* | *MASS AVERAGE* | *MAGNITUDE* | *MAXIMUM* | *MINIMUM*}

For variables that vary by material, such as *ENERGY*, the default is to dump the values for each material separately. However, the *VOLUME AVERAGE* conversion keyword causes the volume-weighted average to be dumped instead. Thus

```
  PLOT VARIABLES
```

```
    STRESS: VOLUME AVERAGE
  END
```

causes each stress component to be plotted as a single volume-averaged value, rather than as separate values for each material. The *AVG* keyword is an alias for *VOLUME AVERAGE.* Similarly, the *MASS AVERAGE* keyword produces a value for a multimaterial element that is the mass weighted average of the quantity. Both *VOLUME AVERAGE* and *MASS AVERAGE* are applicable only to variables associated with a material, e.g., PRESSURE or DENSITY. For both options the EXODUS file variable name will not have the "_N" pattern appended at the end of the name.

The *MAGNITUDE* keyword causes the magnitude of a vector variable to be dumped instead. Thus

```
  PLOT VARIABLES
    VELOCITY: MAGNITUDE
  END
```

causes the magnitude of the velocity to be dumped, rather than each of the vector components. The EXODUS variable will have a "_MAG" appended at the end of the variable name, e.g. VELOCITY_MAG.

The *MAXIMUM* and *MINIMUM* keywords cause the spatially-dependent, time-independent maximum or minimum of a variable to be dumped since the start of the simulation. For scalar variables such as *MASS* or *VOLUME*, the maximum or minimum is dumped. For vector variables such as *VELOCITY*, the maximum or minimum of the magnitude is dumped. For material variables such as *DENSITY*, the maximum or minimum of the volume-weighted average is dumped. Thus

```
  PLOT VARIABLES
    VOLUME
    VOLUME: MINIMUM
    DENSITY: AVG
    DENSITY: MINIMUM
    DENSITY: MAXIMUM
    VELOCITY
    VELOCITY: MAXIMUM
  END
```

causes the volume and its minimum to be dumped, as well as the material-averaged density and its minimum and maximum, and the velocity and its maximum magnitude.

Users can override the default plot name using the syntax:

```
   PLOT VARIABLES
    VELOCITY, AS, VEL
  END
```

Table of Contents
General Input

or, if you find the second comma irritating,

```
PLOT VARIABLES
  VELOCITY, AS “VEL”
END
```

Case is not significant.

If the user is comparing old and new EXODUS files and wants precisely identical spellings of plot variables, the underscores that now separate the material and component tags can be removed in the variable names in the EXODUS plot file. This can be done by using

```
PLOT VARIABLES
  NO UNDERSCORES
  VELOCITY, AS “VEL”
END
```

which causes VELOCITY to be plotted as VELX, VELY, and VELZ, just as in some old EXODUS plot files created by ALEGRA.

Table 6: Plot Variables for General Region Quantities

VARIABLE NAME	TYPE	Explanation
ASPECT RATIO	scalar	element
MASS	scalar	nodal mass
EL MASS	scalar	element mass
COORDINATES	vector	nodal coordinates
VOLUME	scalar	element volume
PROC ID	scalar	for parallel runs, the processor id where the element’s computations are done
VOID FRC	scalar	fraction of an element’s volume occupied by void

Table 7: Plot Variables for Dynamic Hydrodynamic Quantities

VARIABLE NAME	TYPE	Explanation
FLUX	material	
*REACTION**	vector	if REACTION enabled

Table 7: Plot Variables for Dynamic Hydrodynamic Quantities (Continued)

VARIABLE NAME	TYPE	Explanation
*REMESH**	vector	if REMESH enabled
VARIATIONAL FLUX	material	
VELOCITY	vector	

Table 8: Plot Variables for Hydrodynamic Quantities

VARIABLE NAME	TYPE	Explanation
ACCELERATION	vector	
ARTIFICIAL VISCOSITY	tensor	
DEFORMATION RATE	symtensor	
DETONATION TIME	scalar	if PROGRAMMED BURN enabled
EXTERNAL ENERGY	scalar	
HE ENERGY ADDED	scalar	if PROGRAMMED BURN enabled
HE TOTAL ENERGY	scalar	if PROGRAMMED BURN enabled
HOURGLASS_RESISTANCE	vector	
HOURGLASS_STIFFNESS	vector	
MIDCOOR	vector	

Table 9: Plot Variables for Solid Dynamic Quantities

VARIABLE NAMES	TYPE	Explanation
STRESS	symtensor	
ROTATION	tensor	
STRETCH	symtensor	

Table 10: Plot Variables for Physics Quantities and Material Properties

VARIABLE NAMES	TYPE	Explanation
ENERGY	material	
ENERGY CHANGE	scalar	
EQPS	scalar	Equivalent plastic strain
PRESSURE	scalar	
SOUND SPEED	scalar	
SPECIFIC HEAT VOL	scalar	
STRESS	tensor	
STRETCH	tensor	
TEMPERATURE	scalar	
YIELD STRESS	material	

3.3.2.12 HISTORY PLOT VARIABLES

HISTORY PLOT VARIABLES

name [conversion]

name [conversion]

...

END

This keyword permits the user to specify the node and element variables that will be associated with the tracers listed in the *TRACER POINTS* input; data for these variables are ultimately written to the HISPLT plot database file. By default (i.e., no *HISTORY PLOT VARIABLES* input) all registered element and node variables are written to the HISPLT database, along with all global variables. If no *TRACER POINTS* are included in the input file, then only the global data (including material global data) will be written to the HISPLT plot file [44].

The guidelines for using the *HISTORY PLOT VARIABLES* input are the same as the *PLOT VARIABLES*, **Section 3.3.2.11 on page 67,** ***with the following exceptions***.

In the output HISPLT files, the variable names will appear as requested, but will have minor differences because of the limitations of HISPLT. The "outhis" file produced by HISPLT, available after the first HISPLT test run, will list the exact variable names within the database that are available for display. These HISPLT variable names will be the same as the names included in the *HISTORY PLOT VARIABLES* keyword group, except that underscores within a name will be replaced by hyphens (or will be omitted if "no underscores" is designated in the keyword list), and the root name is truncated to limit the string to 16 characters.

For variables describing the value of the quantity for a given material within an element at the tracer location, the HISPLT name is indicated by appending the material id number. This is the number assigned by the user in the input file with the *MATERIAL* keyword. For example, the temperature of the material that is labeled 101 by the user is written to the HISPLT file with the name TEMPERATURE-101. If the material averaged quantity is requested, the material id will be omitted. For material global variables, the HISPLT name of the variable describing the value of the quantity for a given material has nothing appended until the variable is requested in the HISPLT input, at which time the user appends the material number [44].

For vector variables, such as *VELOCITY*, each vector component will be written to the HISPLT file with the name of the component direction appended. For example, the x component of velocity will be named VELOCITY-X. Tensor quantities are labeled similarly, as for example STRESS-XY, for the xy component of the stress tensor in an element. Tensor quantities that are specific to a material will have the material id appended to the variable name before the component designation. For example, STRESS-202-XX will be the name of the xx component of stress in material 202 in an element. If the magnitude of the vector or tensor quantity is requested, the component designators will be omitted. For names longer than 16 characters, including the material number and/or the component designation, it is recommended that the user override the default name to be consistent with the HISPLT database requirements.

For all element variables, the value of the variable at the element center will be written to the database (i.e., interpolation to the tracer location within the element is not performed for element variables). For nodal variables, interpolation to the tracer location is performed by default. If no interpolation is desired, this should be designated in the *TRACER POINTS* input section for each individual tracer (see **Section 3.4.7 on page 106**).

3.3.2.13 RESTART DUMPS

RESTART DUMPS int

This keyword specifies the maximum number of restart dump files that an ALEGRA calculation can retain, *nmax*. These files will be named *"run_id".dmp.n*, where *"run_id"* is the prefix of the ALEGRA input specification file, i.e., the name used on the Alegra script execute line, and *n* is a file number. The value of *n* will start at 0 and increment by 1 each time a restart dump is written. When *nmax* files have been written, successive restart dumps will be matched by deletion of the earliest restart dump to keep the total number of restart dump files at *nmax*. By default, if no *RESTART DUMPS* keyword is used, *nmax* is 2 and only two restart dump files will be retained.

Table of Contents
General Input

3.4 General Physics Keywords

3.4.1 PHYSICS CHOICES

Keywords specify the physics type(s) to use for a calculation. The options are listed in the following subsections.

3.4.1.1 HYDRODYNAMICS

The choice of *HYDRODYNAMCIS* as the physics specification will produce a simulation that results in deformation of the elements with zero stress deviators. Only the equations of state in the material model section can be used in these calculations.

3.4.1.2 SOLID DYNAMICS

The choice of *SOLID DYNAMICS* as the physics specification will produce a simulation that results in deformation of the elements with nonzero stress deviators. All of the available material models can be used in such a simulation.

3.4.1.3 MRDYNAMICS

The choice of *MRDYNAMICS* as the physics specification allows multiple *HYDRODYNAMCIS* and *SOLID DYNAMICS* physics to be analyzed in the same calculation. The primary purpose of *MRDYNAMICS* is to provide a prototype for multiple region simulations. There is no coupling between regions or physics. However, multiple, separate problems may be run in separate regions. Each individual region uses the same keyword sequences as used for a standard single region problem.

3.4.2 Geometry

3.4.2.1 CARTESIAN

CARTESIAN [int D]

This keyword specifies that Cartesian geometry is to be used. It optionally specifies the dimensionality, which will be checked against the dimensionality of the executable used.

3.4.2.2 CYLINDRICAL

CYLINDRICAL [int D]

This keyword specifies that axisymmetric cylindrical geometry should be used. It optionally specifies the dimensionality, which will be checked against the dimensionality of the executable used. Cylindrical geometry assumes an r-z coordinate system with the axis of symmetry along z. In x-y mesh geometry, the axis of symmetry corresponds to the y axis. Note that cylindrical geometry is not supported by the 3D executable.

3.4.2.3 VOLUMETRIC SCALE FACTOR

[VOLUMETRIC SCALE FACTOR real (1.)]

This keyword specifies a multiplier for global tallies such as mass and energy. This multiplier is a simple volumetric-based scaling factor. It may be useful for problems involving periodic boundary conditions to scale quantities by the degree of symmetry. It also may be useful for 2D Cartesian simulations to scale global quantities by the actual length of an object as opposed to assumed unit lengths of 1 m in SI units or 1 cm in CGS units (see the *UNITS* keyword **Section 3.2.5 on page 59**). The user is cautioned that this single factor may not correctly scale all surface tallies if there are multiple symmetries present (e.g., translational, rotational, and mirror). This keyword previously was SYMMETRY FACTOR and this syntax is still supported for now. Users should move to the more precise keyword.

3.4.3 General Initial Conditions

3.4.3.1 DIATOMS

DIATOMS subkeywords ENDDIATOM

The *DIATOMS* capability is a method of inserting material into a mesh. The capability originates with the CTH Eulerian Solid Dynamics simulation code. ***The DIATOM capability is intended for use only with structured, orthogonal meshes aligned with coordinate axes.*** The user is not prevented from using DIATOMS with other meshes, but it must be recognized that sometimes very large inaccuracies will result from the approximation made for the cell volume. The DIATOMS package assumes that the cell volume is a rectangle (or rectangular solid) defined by the minimum and maximum cell coordinates and aligned with the coordinate axes. For the CTH code, this would uniquely span the cell volume.

The input format for *DIATOMS* in Alegra follows the DIATOMS usage in CTH for the virtual objects package, which itself is modeled after the CTHGEN input format for material insertion. The parsing of diatom keywords is done by the CTH diatom library, and therefore follows parsing rules for CTH. Therefore, normal ALEGRA unit conversions (**Section 3.2.5 on page 59**) are NOT allowed in the *DIATOMS* input section. Some functionality that exists in CTH is not currently available in Alegra. For example, the CTH virtual objects package allow objects to be inserted into the spatial mesh at times other than initialization. Functionality available in Alegra is listed in the tables below. Acceptable abbreviation limits are denoted by an asterisk (*) in the keyword.

There are a multitude of keywords that can be used to define the insertion objects. These determine geometry or placement of the objects (e.g., *XROTATE, TRANSLATE*), the initial conditions of the material in the objects (e.g., *DENSITY, XVELOCITY,* and the graded options such as *AGRADED*), or the initial resolution of the shape in the given mesh (e.g., *ITERATIONS, NUMSUB*). The properties defined by the keywords within a package will apply to all following inserts until the *ENDP* keyword is reached. The properties will apply to all packages in the set if they are specified before the package keyword.

Most of these keywords are optional. The minimum keywords required to define the insertion of an object are *PACKAGE*, *MATERIAL*, and *INSERT*. A complete set of the package keywords are provided in alphabetical order in **Table 11**. A complete set of the package subkeywords are provided in alphabetical order in **Table 12**. Examples are provided following the tables. Shape options are provided in the **Table 13**. The shape options are a subset of those available in the CTH code.

Table 11: Package Keywords for DIATOMS

Subkeyword	Argument	Meaning
*ENDDIA*TOM*		*ENDDIA*TOM* required to end material insertion input.
*PA*CKAGE* ... *ENDP*	*string*	Package name.This keyword begins a set of data that applies to all shapes specified within the *PACKAGE/ENDP* keyword group. The package name must be enclosed in single quotes if the following delimiters are within the package name: blank, comma, parentheses(), equal sign, or asterisk. A keyword identified in the table as “package subkeyword” must be within the *PACKAGE* input set to be recognized by the input parser. (Required keyword).
*SCAL*E* ... *ENDS*	*real*	Scale all shapes until the ENDS keyword by this real value.
*STA*RT_ANYWHERE*	*none*	This keyword is required if the problem start time is nonzero and an initial velocity is specified. Note that the underscore is required between the two words, or the keyword can be abbreviated to as few as three letters.
*TRA*NSLATE* ... *ENDT*	*real x, real y, real z*	Translate all shapes until the ENDT keyword by x,y,z.

Table 11: Package Keywords for DIATOMS (Continued)

Subkeyword	Argument	Meaning
*XROT*ATE* *...* *ENDX*	*real*	Rotate all shapes about the x axis by this real value in degrees. The center of rotation is the mesh origin. The rotation affects all shapes listed between keywords XROTATE and ENDX.
*YROT*ATE* *...* *ENDY*	*real*	Rotate all shapes about the y axis by this real value in degrees. The center of rotation is the mesh origin. The rotation affects all shapes listed between keywords XROTATE and ENDX.
*ZROT*ATE* *...* *ENDZ*	*real*	Rotate all shapes about the z axis by this real value in degrees. The center of rotation is the mesh origin. The rotation affects all shapes listed between keywords ZROTATE and ENDZ.

Table 12: Package SubKeywords for DIATOMS

Subkeyword	Argument	Meaning
*AGR*ADED P1= P2=*	*real x, real y, real z, real x real y, real z*	Material properties in the package are graded axially away from P1 in the direction P2. The relevant material property must specify a table number; properties in the table should be defined as functions of the projection of $(x-x_1,y-y_1,z-z_1)$ from the vector $(x_2-x_1,y_2-y_1,z_2-z_1)$. *This feature can not be used in combination with CGRADED.*
*CGR*ADED P1= P2=*	*real x, real y, real z, real x, real y, real z*	Material properties in the package graded radially away from the axis defined by P1 and P2. The relevant material property must specify a table number; properties in the table should be defined as functions of the distance of $(x-x_1,y-y_1,z-z_1)$ from the vector $(x_2-x_1,y_2-y_1,z_2-z_1)$. *This feature can not be used in combination with AGRADED.*

Table 12: Package SubKeywords for DIATOMS (Continued)

Subkeyword	Argument	Meaning
*DEL*ETE* *shape* *ENDD*	*string*	Shape to be removed (see choices with *INSERT* keyword.) End each delete set with *ENDD*. Multiple deletes are allowed per package. The deletes are only valid for insertions in the same package.
*DEN*SITY*	*real or T(int)*	Initial density of the material in the insertion set and all subsequent insertions sets. Must reset density of subsequent sets to desired value or zero to obtain the initial ALEGRA values. A table number can also be input to specify density as a time varying function or for use with a graded option.
*INS*ERT* *shape* *ENDI*	*string*	Shape to be inserted (see allowable shapes and associated shape subkeywords in **Table 13**). After each shape, enter the required shape subkeywords. If the entire mesh is to be filled, a shape should extend beyond the mesh. End each insert set with the *ENDI*NSERT* keyword. Note that only the blocks that specify *ADD DIATOM INPUT* will have the inserted object (see ***BLOCK*** main keyword. The delete operations within a package are only effective on the material shapes inserted within the same package (required).
*IT*ERATION*	*int*	Number of iterations to recursively subdivide cell in each direction for insertion. Relates to CTH *NUMSUB* as follows: iter=NINT(ln(numsub)/ln(2.)), where NINT is the nearest integer. Recommend *ITER*=3, 4, or 5.

Table 12: Package SubKeywords for DIATOMS (Continued)

Subkeyword	Argument	Meaning
*M*ATERIAL*	*int*	Id of the material that the next *INSERT/ DELETE/REPLACE* keyword will insert within this package. An individual package can have more than one material keyword if the user decides to operate on another material within the same package; however, most packages will logically contain only one material, which must be specified before the *INSERT/DELETE/REPLACE* operation is specified (required).
*N*UMSUB*	*int*	In CTH is number of subdivisions per cell in each direction. In ALEGRA, the value given for *NUMSUB* will be used to calculate the iteration level as above.
*RGR*ADED P1=*	*real x, real y, real z*	Material properties in the package are graded radially about point P1. The relevant material property (density or temperature) must reference a table. Tabular properties should be defined as functions of r=sqrt($(x-x_1)^2+(y-y_1)^2+(z-z_1)^2$).
*T*EMPERATURE*	*real or T (int)*	Initial temperature of the material in the insertion set and all subsequent insertions sets. Must reset temperature of subsequent sets to desired value or zero to obtain the initial ALEGRA values. A table number can also be input to specify temperature as a time varying function or for use with a graded option.
*MV*ELOCITY*	*real x, real y, real z*	Material velocity vector for the object inserted. If a material velocity is input for a package in a problem with a nonzero start time, the problem will start with the material velocity given. (Compare with XVELOCITY). This velocity will apply to all subsequent insertions sets within the package.

Table 12: Package SubKeywords for DIATOMS (Continued)

Subkeyword	Argument	Meaning
*XV*ELOCITY [YV*ELOCITY] [ZV*ELOCITY]*	*real*	Initial Lagrangian velocity of the material in the insertion set and all subsequent insertions sets. Must set velocity of subsequent sets to zero if no initial velocity desired.

.

Table 13: Shape Subkeywords for DIATOM Insertions

Shape Keyword	Keyword/Numeric Input	Meaning

BOX	2D: P1=<x,y>
	P2=<x,y>
	3D: P1=<x,y,z>
	P2=<x,y,z>
		2D & 3D
		Points p1 and p2 are the minimum and maximum coordinates of the box, aligned with the coordinate system.

*CI*RCLE*	CE*NTER=<x,y> R*ADIUS=<r> RI*NNER=<r=0>	2D Only The circle is defined by the center and the radius. An inner radius is optional.

UDS	P1=<x,y>
	P2=<x,y>
	P3=<x,y>
	...
	Pn=<x,y>
		2D Only
		This is a user defined shape defined by points given. The first and last points will be connected by the diatom package to close the shape. (Limited to 3000 coordinate points.)

*CY*LINDER*	CE1 = <x,y,z>
	CE2 = <x,y,z>
	R*ADIUS = <r>
	RI*NNER = <r=0>
		3D Only
		The points CE1 and CE2 define the end points of the axis of the cylinder. The outer radius is given by R and the inner radius (optional) defined by RI.

R2DP	CE1 =<x,y,z>
	CE2 = <x,y,z>
	P1=<u,v>
	P2=<u,v>
	P3=<u,v>
	...
	Pn=<u,v>
		3D Only
		The points CE1 and CE2 define the end points of the axis of rotation. The points Pn define the shape along the axis, with v=distance along the axis, starting at ce1, and u=distance from the axis.

*S*PHERE*	CE*NTER = <x,y,z> R*ADIUS = <r> RI*NNER = <r=0>	3D Only The sphere is defined by the center and the radius. An inner radius is optional.

Table 13: Shape Subkeywords for DIATOM Insertions

Shape Keyword	Keyword/Numeric Input	Meaning

*TET*RAHEDRON*	P1= <x,y,z>
	P2= <x,y,z>
	P3= <x,y,z>
	P4= <x,y,z>
		3D Only
		Points define the vertices of the tetrahe-dron.

*TO*ROUS*	CE*NTER= <x,y,z>
	R*ADIUS = <r>
	P1= <x,y,z>
	P2= <x,y,z>
		3D Only
		P1 and P2 define the axis of rotation for the circle of radius R at center given.

FNF	FILE='*filename*' MAT=n	3D Only The insertions are to be read from an external file in the FNF format output from ProMesh. The filename must be given in quotes, along with the material number to be extracted from the FNF file.

The following example illustrates the use of the *DIATOMS* option.

```
block 1
  eulerian mesh
  add diatom input
end

diatoms
  package cylinder_one
    material=34
     insert cylinder
      ce1 0. 10. 10.
      ce2 0. 10.  0.
      radius 5.0
      rinner 2.0
    endinsert
  endpackage
  package ring
    material 34
    iteration 4
    yvelocity 1.e4
    insert torus
      center 4. 10. 10.
      radius 2.0
      p1  7.  0. 10.
```

```
        p2  7. 10. 10.
      endinsert
    endpackage
    package plate
      mat 2
      iteration 4
      insert r2dp
        ce1 0.  0.  10.
        ce2 13. 0.  10.
        p1 0.  0.
        p2 4.  3.
        p3 4.  10.
        p4 0.  13.
      endinsert
    endpackage
    package plate
      material 34
      insert fnf
        file='track.fnf' mat=2
      endi
    endp
$GRADED TEMPERATURE
    package 'part 1'
      temperature = t1
      rgraded p1 = 20., 10., 10.
      insert sphere
        ce = 20., 10., 10.
        r=11.
      endi
    endp
  enddiatom

function 1
  0.0  596.
  11.  1500.
end
```

3.4.4 General Boundary Conditions

3.4.4.1 Periodic Boundary Conditions

The ALEGRA user must supply an initial mesh which supports any periodic boundary conditions requested. This means that matching nodesets are required for each set of periodic boundaries. The nodes in the matching nodesets on the mesh boundary must be consistent with a periodic mesh. A necessary condition for the mesh to be allowable is that the number of nodes in the matching nodesets must be the same. Each node in one nodeset corresponds one-to-one to a node in the matching nodeset. The user may pick either a translation or a rotation for each periodic boundary condition. Details of the current implementation of periodic boundary conditions in ALEGRA can be found in (Robinson 1999).

3.4.4.2 Translational periodicity

PERIODIC BC, {nodeset1}, TRANSLATE {u}, {nodeset2} [TOLERANCE real (1.e-5)]

where nodeset1 and nodeset2 are the first and second nodesets, u is the translation vector which maps the nodes in nodeset1 to nodeset2. That is,

$$x^2 = x^1 + u$$

Example:

```
periodic bc, nodeset 3, translate, x 1. y 0., nodeset 1
```

A 2D Cartesian mesh may be made doubly periodic by associating the two pairs of opposite nodesets and the two pairs of diagonally opposite corners. For example, suppose the opposite sides are labelled by nodesets 11 to 14, and the four corner vertices are labelled by nodesets 101 to 104 (each of these latter nodesets containing only one node), and the mesh is 10 units by 1 unit, as shown in the follow figure.

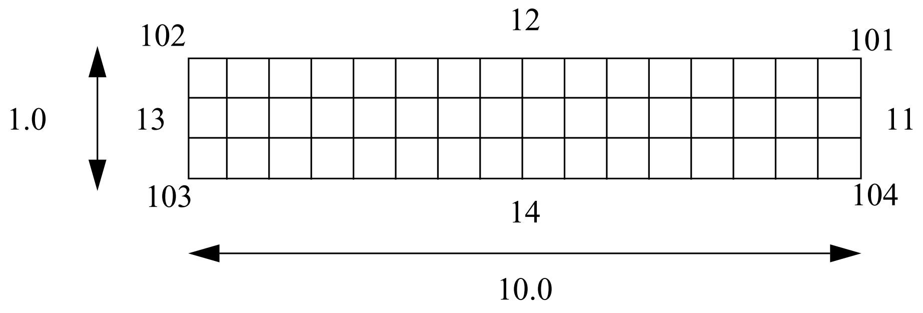


Figure 9: Periodic mesh example.

The input to make this mesh doubly periodic in X and Y are:

```
periodic bc, nodeset 13,  translate, x 10. y 0.,  nodeset 11
periodic bc, nodeset 14,  translate, x 0.  y 1.,  nodeset 12
periodic bc, nodeset 103, translate, x 10. y 1.,  nodeset 101
periodic bc, nodeset 102, translate, x 10. y -1., nodeset 104
```

A similar method may be used to make a 3D Cartesian mesh triply periodic. In this case one must associate the three pairs of opposite faces, the six pairs of diagonally opposite (and parallel) edges, and the four pairs of diagonally opposite corners. All associations pass through the center of the rectangular parallelepiped.

3.4.4.3 Rotational Periodicity

User input for rotational periodicity in two dimensions is

PERIODIC BC, {nodeset1}, ROTATE {θ}, POINT {p}, {nodeset2} [TOLERANCE real (1.e-5)]

Example:

```
periodic bc, nodeset 3, rotate 45., point, x 0. y 0., nodeset 4
```

and in three dimensions

PERIODIC BC,{nodeset1}, ROTATE {θ} POINT {p} AXIS {a}, {nodeset2}
[TOLERANCE real (1.e-5)]

a is the rotation axis passing through the point p which rotates the nodes in nodeset1 to overlie the nodes in nodeset2 through an angle θ given in degrees.

Example:

```
periodic bc, nodeset 3, rotate 45., point, x 0. y 0. z 0.,
     axis, x 0. y 0. z 1., nodeset 4
```

3.4.5 Time Step Control

3.4.5.1 GRADUAL STARTUP FACTOR

GRADUAL STARTUP FACTOR real (0.01)

This keyword specifies the factor by which the initial physics-based time step should be multiplied. This has the effect of gradually marching into an abrupt transient. This value should always be greater than zero and less than or equal to 1.0. See also the ***MAXIMUM INITIAL TIME STEP*** keyword.

3.4.5.2 MAXIMUM INITIAL TIME STEP

MAXIMUM INITIAL TIME STEP real

This keyword permits the user to specify a maximum initial time step, otherwise ALEGRA computes an initial timestep based upon the initial conditions. It is useful where unusual mechanical transients would otherwise result in an instability in the starting time step. For example, a small starting time step should be specified to permit the material a few cycles to begin responding to energy deposition. Likewise, a large initial velocity on part of a mesh may require a small initial time step. See also the ***GRADUAL STARTUP FACTOR*** keyword.

3.4.5.3 MAXIMUM TIME STEP LIMIT

MAXIMUM TIME STEP LIMIT real

This keyword permits the user to specify a maximum time step value that the simulation will never exceed. Otherwise, the maximum time step is virtually unlimited.

3.4.5.4 MAXIMUM TIME STEP RATIO

MAXIMUM TIME STEP RATIO real (1.2)

This keyword sets the maximum ratio by which the time step may grow from one cycle to the next.

3.4.5.5 MINIMUM TIME STEP

MINIMUM TIME STEP real (1.0e-20)

This keyword specifies the minimum time step permitted. If the stable time step is computed to be less than this value, the calculation will cease and write a the final output records for a normal completion.

3.4.5.6 TIME STEP SCALE

TIME STEP SCALE real (0.67 in 2D, 0.9 otherwise)

The internal calculation of the maximum stable time step occasionally overestimates this quantity. The factors specified by this keyword allows a smaller time step to be used. This option is particularly useful when instabilities appear; by setting these factors to small values, one may be able to distinguish physical from numerical instability.

This keyword causes the multiplication of the calculated time step by the specified *real* factor. The calculated time step is the minimum time step that has been selected from all other physics-based time-step constraints. Values less than 1 will shorten the time step, while values greater than 1 will increase the time step. Increasing the time step is not recommended since a loss of numerical accuracy may occur even though the numerical algorithms may be implicitly stable.

Example:

The following input will cause the time step used in a calculation to be one half of the value ALEGRA would normally use.

```
  TIME STEP SCALE 0.5
```

Table of Contents
General Input

3.4.5.7 CONSTANT TIME STEP

CONSTANT TIME STEP real

This keyword specifies that the simulation will be run with a constant time step value equal to the value specified here, otherwise ALEGRA determines a new time step each cycle.

```
  CONSTANT TIME STEP 1.0e-8
```

3.4.6 General Algorithm Control

A number of controls are available for setting up a problem and specifying how the mesh is to behave. It is important to understand some basic concepts about an ALEGRA mesh. An ALEGRA mesh is composed of a set of mesh blocks which are all composed of the same type of elements. Currently only quadrilateral and hexahedral meshes have extensive support and testing in ALEGRA. This means that generally a mesh block simply refers to a separately numbered portion of the mesh. The mesh block is used to specify a focus region for initial conditions and Arbitrary Lagrangian-Eulerian (ALE) algorithm controls. SMALE (Single Material ALE) refers to ALE remeshing within a single block in the sense that no material is allowed to flow across the Lagrangian block boundaries. In ALEGRA SMALE does not imply that only a single material is present within the block since ALEGRA can be always be initialized with multiple materials in each element. The single material part of the SMALE name has its roots in the early technique or concept of block material initialization in which a block was initialized to a single material. A better name today would be Single Block ALE but the SMALE name is till in use. An MMALE (Multi-Material ALE) designation allows two adjoining mesh blocks to be treated as a single large ALE block. Thus MMALE blocks do not necessarily have Lagrangian boundaries. The name Multi-Block ALE is more precise but the MMALE name is currently used. Two adjoining MMALE blocks are treated as a single ALE region and material is allowed to flow between these two blocks. Additional ALE designators can be given on sidesets and nodesets to precisely define the ALE algorithms intended on block boundaries.

3.4.6.1 BLOCK

BLOCK int

  [subkeyword-list]

END

The *BLOCK* keyword group allows the user to specify the materials that are contained in a block and the type of mesh movement desired in the block. The default is for a block to be a voided lagrangian block (i.e., if all subkeywords are omitted). The *BLOCK* subkeywords are described in the tables below.

3.4.6.1.1 Material Specification

A principal use of the *BLOCK* is to insert materials into the mesh. This is performed on a block by block basis. ALEGRA will process *DIATOMS* first, the *VOLUME FRACTION FILE* second, and the *MATERIAL* input last.

If multiple *DIATOM* packages affect the same block, they will be processed in the order they appear in the input deck. If two or more *DIATOM* shapes associated with the packages overlap in the same mesh cell, ALEGRA will compute a volume fraction for the material associated with the first package and the remainder of the cell volume will be temporarily assigned to void. The volume fraction is determined from the intersection of the shape with the volume of the cell. Next ALEGRA will compute a volume fraction for the material associated with the second package and use the minimum of this volume fraction and the void fraction as the available volume fraction for the second material. The volume fraction for the second package is computed as if there is no knowledge of the first package. It is only the available void fraction that provides one package with a knowledge of a previous package. The process continues until all *DIATOMS* are exhausted. Overlaps can be used to the user's benefit to prevent miniscule amounts of void or stray material from creeping into cells due to round off error in computing volume fractions on irregular meshes. The user is cautioned, however, that the results may be order dependent and the results of multiple *DIATOM* inserts should be carefully checked.

If both the *ADD DIATOM INPUT* and the *MATERIAL* keywords are present in the same block, all of the *DIATOMS* will be processed first. Any remaining void fraction will be replaced with the material specified on the *MATERIAL* keyword.

If multiple MATERIAL keywords are present in given block, then the available volume fraction is equally divided among the materials specified.

Table 14: Block Material Initialization Keywords

Sub-Keyword	Numeric Input	Meaning
MATERIAL	*int*	Value corresponds to an id specified by a valid *MATERIAL* keyword. There can be multiple material entries.
ADD DIATOM INPUT		Indicates that volume fractions will be calculated in this block from the information given in the *DIATOMS* keyword group. The materials are identified in the *DIATOMS* keyword group input.

Table 14: Block Material Initialization Keywords (Continued)

Sub-Keyword	Numeric Input	Meaning
VOLUME FRACTION FILE		Indicates that volume fraction input is to be read from an external ASCII file of the name runid.VF. The volume fraction file is a formatted file that requires a structured mesh with logical *ijk* ordering and uniform spacing in each direction for the block(s) specified in the *run_id.VF* file.

Example:

```
block 1
   add diatom input   $ diatoms are processed first
   material 1         $ mat 1 fills remainder of block
   lagrangian mesh
end
```

3.4.6.1.2 Mesh Type

A second principal use of the *BLOCK* input is specification of the type of mesh. The specification of *LAGRANGIAN MESH, SMALE MESH*, *MMALE MESH*, or *EULERIAN MESH* initially sets the mesh movement for the block to these types. A question arises regarding how to treat the mesh nodes on the boundary between blocks of two different mesh types. By default, ALEGRA assumes a certain hierarchy of mesh types.

EULERIAN < *MMALE* < *SMALE* < *LAGRANGIAN*

ALEGRA assigns the greater type from this hierarchy to the nodes on a common boundary. However, particular nodes within the mesh may be changed to a different type depending on nodeset/sideset definitions and on the blocks that neighbor this particular block. See the ***DOMAIN*** keyword for more information on how to use the nodeset/sideset keywords for specific remesh control.

Table 15: Block Mesh Specification Keywords

Sub-Keyword	Meaning
LAGRANGIAN MESH	In the Lagrangian formulation the nodes move at the material velocity. This is the default mesh type.

Table 15: Block Mesh Specification Keywords (Continued)

Sub-Keyword	Meaning
SMALE MESH	In a single-material (single block) ALE formulation, the mesh moves with the material until the distortion becomes large enough to trigger the remeshing of the nodes and a corresponding remapping of material quantities. Material cannot flow out of the mesh block.
MMALE MESH	In a multi-material (multi-block) ALE formulation, the mesh moves with the material until the distortion becomes large enough to trigger the remeshing of the nodes and a corresponding remapping of material quantities. Material may flow out of the mesh block.
EULERIAN MESH	In an Eulerian formulation, the nodes remain fixed.
SMOOTHED EULERIAN MESH	In this formulation, the nodes are repositioned to their original coordinates (as for *EULERIAN MESH*), but the mesh is then smoothed.

Example:

```
block 1
   material 10
   smale mesh
end
block 2
   material 20
   lagrangian mesh
end
```

3.4.6.1.3 Remap Control

Another important function of the *BLOCK* input is to control the remap or advection. *BLOCK* keywords are used in conjunction with various *DOMAIN* keywords to produce the desired level of control over mesh behavior. An ALEGRA simulation cycle consists of a Lagrangian step where the material and the mesh move together. If the mesh type is not Lagrangian, then the mesh nodes are moved back to their former positions if the mesh is Eulerian, or perhaps to some other position if the mesh is ALE. This movement of nodes is termed the remesh phase of a remap. When the mesh nodes are moved, the material position must remain fixed. Thus, a certain fraction of the material in a remeshed cell must be transferred to adjacent cells depending on the amount of volume that fluxes or passes through each element face. This fluxing of material is known as advection. In ALEGRA, “remapping” means mesh movement or “remeshing” followed by advection.

Several remesh methods are available. These remesh methods move nodes interior to the mesh. The boundary nodes remain fixed. Therefore, there must be interior mesh nodes for remeshing to occur. A consequence of this is that in simple 1D-like problems, where there is only one row of mesh cells, nodes cannot be remapped. (1D-like problems may be used for simple scoping studies.) The frequency of remaps is also controlled by the keywords in **Table 16**.

The *AVERAGE REMESH METHOD* moves a node to the average of the element centers to which the node is connected, although it produces instabilities in some test cases. The *BUDGE REMESH METHOD* attempts to produce an orthogonal mesh

The *TIPTON REMESH METHOD* solves the inverted Laplace equation through a variational approach. The resulting set of equations are solved using Jacobi iteration. It is unnecessary to find the exact solution to these equations. Excessive node movement may over empty a cell. (See *REMESH MOVEMENT* controls.) This algorithm without weights attempts to equalize the volumes of the elements about a node. With weights, one can move nodes towards regions of interest. The *WINSLOW REMESH METHOD* is the *TIPTON REMESH METHOD* without weights

Table 16: Block Remesh Methods Sub-Keywords

Sub-Keyword	Numeric Input	Meaning
AVERAGE REMESH METHOD *BUDGE REMESH METHOD* *TIPTON REMESH METHOD* *WINSLOW REMESH METHOD*		*(TIPTON)* Turns on the indicated interior remesh method.
REMESH FREQUENCY	*int (1)*	Number of time steps between a remap step
EULERIAN MOVEMENT {X | Y | Z}		Specifying *X*, *Y*, or *Z* cause the mesh to remain fixed relative to material motion in that direction.
RADIAL CONSTRAINT		Constrains remesh movements to the radial direction. (Assumed center at (0,0,0) for now.

Remeshing is a multi-step process. The first step is to determine which nodes in the mesh are eligible to be remeshed. Lagrangian nodes are not remeshed. Eulerian nodes are remeshed to their former positions, prior to the Lagrangian step. ALE nodes are conditionally remeshed depending on whether some threshold condition is met. If the condition is not met then the node is not remeshed and the mesh behaves in a Lagrangian manner. If the condition is met, then the node is remeshed and the mesh behaves in an ALE manner. SMOOTHED EULERIAN nodes are treated as Eulerian unless a remesh condition is met. When the remesh conditions are satisfied they are remeshed just as an ALE node would be. While ALE nodes will behave in a Lagrangian manner unless the remesh conditions are triggered, the SMOOTHED EULERIAN nodes will behave in an EULERIAN manner unless the conditions for remesh are met. The threshold conditions are known as triggers.

Triggers may be based on geometric or physical considerations as described in the following table.

Table 17: Block Remesh Trigger Sub-Keywords

Sub-Keyword	Numeric Input	Meaning
ANGLE TRIGGER	*real*	Minimum node angle on which to trigger a nodal remesh. This value is entered as the absolute value of the cosine of the minimum angle. An ideal angle is 90^o or a value of zero for the cosine of the angle. A value of zero will guarantee that every node is remeshed. Note that this trigger is NO LONGER ON BY DEFAULT.
SOLID ANGLE TRIGGER	*real*	Trigger a nodal remesh based on the minimum solid angle at a node. This value is entered as a ratio, which should not be exceeded, of an ideal solid angle to the minimum solid angle. The ideal solid angle is the total interior solid angle (4π for an interior node) divided by the number of elements connected to the node. A value of 1.0 will guarantee that every node is remeshed.
VOLUME TRIGGER	*real*	Minimum adjacent element volume on which to trigger nodal remesh. A value of 1.0 will guarantee that every node is remeshed. Note that this trigger is NO LONGER ON BY DEFAULT.
VARVOL TRIGGER	*real*	Analogous to the volume trigger, but uses the “variational volume” which in 2D cylindrical geometry is the area of the element.

Table of Contents
General Input

Table 17: Block Remesh Trigger Sub-Keywords (Continued)

Sub-Keyword	Numeric Input	Meaning
DENSITY TRIGGER	*real*	This trigger will flag nodes for remeshing when the average density in its associated elements is *greater than or equal* to the input threshold value. The average is a simple unweighted numerical average of the density values.
TEMPERATURE TRIGGER	*real*	This trigger will flag nodes for remeshing when the average temperature in its associated elements is *greater than or equal to* the input threshold value. The average is a simple unweighted numerical average of the temperature values.
PRESSURE TRIGGER	*real*	This trigger will flag nodes for remeshing when the average pressure in its associated elements is *greater than or equal to* the input threshold value. The average is a simple unweighted numerical average of the pressure values.
INVERSE DENSITY TRIGGER	*real*	This trigger will flag nodes for remeshing when the average density in its associated elements is *less than or equal to* the input threshold value. The average is a simple unweighted numerical average of the density values.
INVERSE TEMPERATURE TRIGGER	*real*	This trigger will flag nodes for remeshing when the average temperature in its associated elements is *less than or equal to* the input threshold value. The average is a simple unweighted numerical average of the temperature values.
INVERSE PRESSURE TRIGGER	*real*	This trigger will flag nodes for remeshing when the average pressure in its associated elements *is less than or equal to* the input threshold value. The average is a simple unweighted numerical average of the pressure values.

Table 17: Block Remesh Trigger Sub-Keywords (Continued)

Sub-Keyword	Numeric Input	Meaning
COMBINE TRIGGERS		This option causes triggers to be applied in conjunction, i.e. logically “and-ed” together. The normal default is for the triggers to operate separately, i.e. logically “or-ed” together.

As described above, the *TIPTON* method solves the inverted Laplace equation through a variational approach. Using this method with weights, one can move nodes toward regions of interest. Weight can be based on geometric or physical considerations. Various weights are described in the following table. The weight used in the *TIPTON* method is the maximum of the computed value or the threshold value. Node movement is biased towards regions of the mesh where the computed weight exceeds the threshold. Regions of the mesh where the weight value is below the threshold are equally weighted. **The *TIPTON* scheme must be used in all blocks that share a remeshed node**. If this is not done, the remeshed node on the block boundary will migrate rapidly towards the interior of the block.

Once the weights associated with a given method are computed, these weights are then scaled to the range 1.0 to *NORMALIZATION FACTOR*. This prevents excessive node movement. Each weighting method is individually normalized. The normalization may be linear or logarithmic and is specified after the weight keyword. For example, in some explosive or ablative simulations the density may vary over several orders of magnitude. In this case it may be wise to use a logarithmic normalization to capture the wide range of densities.

Normalizing each weight also puts each weight on an equal basis should multiple weights be specified in a single *BLOCK*. Multiple weights are linearly summed after each is individually scaled. The resultant sum is then linearly rescaled to the range 1.0 to *NORMALIZATION*

FACTOR. The relative importance of multiple weights can be controlled through judicious choice

Table 18: Block Remesh Weight Sub-Keywords

Sub-Keyword	Numeric Input	Meaning
DENSITY GRADIENT WEIGHT *DENSITY WEIGHT* *TEMPERATURE GRADIENT WEIGHT* *TEMPERATURE WEIGHT* *PRESSURE GRADIENT WEIGHT* *PRESSURE WEIGHT* *INVERSE VOLUME WEIGHT* *INVERSE RADIUS WEIGHT* *INVERSE XYRADIUS WEIGHT* *INVERSE YZRADIUS WEIGHT* *INVERSE XZRADIUS WEIGHT*	*real*	Any combination of these keywords can be entered. Their entry turns the specific type of weighting on for elements specified with Tipton smoothing. The numeric input ($x \geq 0$) serves as a threshold to control application of the weight. The weight is applied when the element quantity *exceeds* the threshold.
INVERSE DENSITY WEIGHT *INVERSE TEMPERATURE WEIGHT* *INVERSE PRESSURE WEIGHT*	*real*	Any combination of these keywords can be entered. Their entry turns the specific type of weighting on for elements specified with Tipton smoothing. The numeric input ($x \geq 0$) serves as a threshold to control application of the weight. The weight is applied when the *inverse* of the element quantity *exceeds* the *inverse* of the threshold.

{*CHOICE OF WEIGHT real*}
{*LINEAR NORMALIZATION* |
LOG NORMALIZATION}
		(LINEAR)
		These two keywords modify the above weights and determine the method used to normalize weights for Tipton smoothing. Weights are normalized between 1.0 and the value given by the *NORMALIZATION FACTOR* keyword.

NORMALIZATION FACTOR	*real* *(4.0)*	Maximum weight value. ($x \geq 1$) Larger values may cause extreme mesh movement and over-fluxing of an element. If x = 1.0, weighting is ineffective.

of the various thresholds.

Example:

```
block 1
  material 7
  SMALE mesh
  TIPTON remesh method

  density trigger = 30.0
  density weight  = 50.0  $ defaults to linear scaling

  temperature trigger = 3000.
  temperature weight  = 5000. linear norm

  inverse density trigger = 0.01
  inverse density weight  = 0.005 log norm

  norm factor 7.0  $ allow greater node movement
end
```

This example would trigger remeshing in regions of the mesh where there are high densities and temperatures or low densities, leaving regions of the mesh with intermediate values alone. The inverse density weight uses a logarithmic normalization.

The user can also control the method used to advect material variables between elements and momenta between nodes. The keyword is given in **Table 19**.

Table 19: Advection Control Sub-Keywords

Sub-Keyword	Numeric Input	Meaning
DONOR ADVECTION *SUPERB ADVECTION* *VANLEAR ADVECTION*		*(VANLEAR)* Turns on the indicated advection method.

3.4.6.1.4 Other BLOCK Controls

Other methods that can be controlled with *BLOCK* keywords include with types of artificial viscosity and hourglass control are used in the mesh block. A velocity can be imposed upon the material in an Eulerian or Smoothed Eulerian block. And finally it is possible to entirely remove either the material in a block (*DELETE DATA*) or the entire mesh block (*DELETE TOPOLOGY*) at a specific cycle or time.

Table 20: Other Block Sub-Keywords

Sub-Keyword	Numeric Input	Meaning
ARTIFICIAL VISCOSITY	*int*	Specifies the artificial viscosity model to be used for this block (see *ARTIFICIAL VISCOSITY* keyword input).
PRESCRIBED {X | Y | Z} VELOCITY	*real*	Apply a fixed non-zero velocity to the nodes in this block. Valid for *EULERIAN* and *SMOOTHED EULERIAN* blocks.
HOURGLASS CONTROL	*int*	Specifies the hourglass control model to be used for this block (see *HOURGLASS CONTROL* keyword input).
DELETION CYCLE	*int*	Specifies the cycle at which the element block is deleted from the problem.
DELETION TIME	*real*	Specifies the time at which the element block is deleted from the problem. Time must be greater than or equal to *DELETION TIME* to trigger deletion.
DELETE DATA		Delete element block by deleting all vertex, edge, face, and element data associated with the block. Note that coordinates are reset to original value and entire block is filled with void.
DELETE TOPOLOGY		Delete element block by deactivating all vertices, edges, faces and elements associated with the block.

3.4.6.2 DOMAIN

DOMAIN

[subkeyword-list]

END

The *DOMAIN* keyword group allows the user to specify how an entire domain is to behave, i.e., global behavior that cannot be broken down to the block level. The following tables describe the allowed subkeywords that can be specified for a domain.

3.4.6.2.1 Boundary Remesh Control

In order to properly use the sideset and nodeset specifications available in the *DOMAIN* input, a few key points must be understood about the functional characteristics of the ALEGRA rezone method. By using the word "rezone", this discussion is limited to the methods ALEGRA uses to reposition nodes for the ALE method.

The first basic understanding needed is that, unless told differently by the user, ALEGRA will only rezone ***interior*** nodes in a mesh block. The user ***must*** tell the code that rezoning is desired on the sides of these blocks, using the sideset and nodeset keyword commands available in *DOMAIN*. The sideset rezone methods will compute new locations for nodes in the sideset that are interior to the side, i.e., not located on the edges outlining the side. To control rezone of the nodes on the edges, the user ***must*** employ the nodeset keyword commands available in *DOMAIN*.

The next point that must be understood is the method ALEGRA uses to resolve overlapping specifications for nodes on sides and edges. A node is considered to belong to up to four entities: a block, a sideset, a nodeset and a pointset (a nodeset with only a single node included). The node can also have one of four rezone characteristics: LAGRANGIAN, ALE, SMOOTHED EULERIAN or EULERIAN. These characteristics are set by the keyword commands available in the *BLOCK* and *DOMAIN* inputs.

The *BLOCK* mesh specification (e.g., *EULERIAN MESH*) will label the nodes in the mesh block in a manner that depends upon the type of block and the whether the node is in the interior of the block or not. The **Table 21** explains this labeling.

Table 21: Node Rezone Control by Mesh Specification

Block type	Interior Nodes	Non-Interior Nodes
Lagrangian	LAGRANGIAN	LAGRANGIAN
Smale	ALE	LAGRANGIAN
Mmale	ALE	ALE
Smoothed Eulerian	SMOOTHED EULERIAN	SMOOTHED EULERIAN
Eulerian	EULERIAN	EULERIAN

When conflicts between block labels occur, as when nodes are common to two or more blocks, a precedence of node labels is enforced. The node attributes are enforced in the following manner:

LAGRANGIAN > ALE > SMOOTHED EULERIAN > EULERIAN.

The “>” symbol means “overrides”. Thus a node will have the most restrictive setting applied to it by multiple overlapping specifications.

The next specification allowed, in 3D problems only, is a sideset specification. For a sideset, the distinction between types of nodes are those that are on a material interface and those that are not on such an interface. In this context the nodes on each sideset are labeled according to the rules explained in **Table 22**.

Table 22: Node Rezone Control by SideSet Specification

Sideset type	Non-Interface Nodes	Interface Nodes
Lagrangian	LAGRANGIAN	LAGRANGIAN
Smale	ALE	LAGRANGIAN
Mmale	ALE	ALE
Smoothed Eulerian	SMOOTHED EULERIAN	SMOOTHED EULERIAN
Eulerian	EULERIAN	EULERIAN

Once again the node label precedence is enforced for nodes that share one or more sidesets.

The nodesets are processed and labeled in a manner that is exactly the same as the sidesets, with the distinction of node type being the presence of the node on a line where a material interface intersects a mesh boundary. Finally, the point sets are processed, these being the one-entry nodesets.

Once all of the above has been done, the code has a block label, a sideset label (if 3D), a nodeset label and a point label for each node in each block. Note that the latter three labels will only be present if the user specified the node by means of node or sidesets. Each node will however have a label from the block specification. The final step in labeling the nodes to allow remesh motion is to allow the following node movement enforcement.

Point type > Nodeset Type > Sideset Type (if 3D) > Block type.

Here, the “>” symbol indicates “overrides”. With these controls, the user can determine how the Alegra rezone package will operate within the mesh.

Table 23: DOMAIN Keywords for Remesh Control

Sub-Keyword	Numeric Input	Meaning
LAGRANGIAN NODESET *EULERIAN NODESET* *SMALE NODESET* *SMALE XLINE NODESET* *SMALE YLINE NODESET* *SMALE ZLINE NODESET* *[SMALE STRAIGHT LINE NODESET]* *[SMALE CURVED LINE NODESET]* *MMALE NODESET* *MMALE XLINE NODESET* *MMALE YLINE NODESET* *MMALE ZLINE NODESET* *[MMALE STRAIGHT LINE NODESET]* *[MMALE CURVED LINE NODESET]* *SMOOTHED EULERIAN NODESET* *SMOOTHED EULERIAN XLINE NODESET* *SMOOTHED EULERIAN YLINE NODESET* *SMOOTHED EULERIAN ZLINE NODESET*	*int*	Value corresponds to a nodeset id and indicates how the nodes of the set should behave relative to the material motion. The *DOMAIN* keyword can have multiple entries of this type. Note: The *SMALE*, *MMALE* and *SMOOTHED EULERIAN* features produce slightly different results in parallel runs than serial runs. The *LAGRANGIAN, EULERIAN, SMALE, MMALE, and SEUL NODESET* commands control the setting of nodal flags on the nodes of the nodeset. [...] indicates a future capability not yet functional

Table 23: DOMAIN Keywords for Remesh Control (Continued)

Sub-Keyword	Numeric Input	Meaning
LAGRANGIAN SIDESET *EULERIAN SIDESET* *SMALE SIDESET* *SMALE XYFACE SIDESET* *SMALE YZFACE SIDESET* *SMALE XZFACE SIDESET* *[SMALE PLANAR FACE SIDESET]* *[SMALE CURVED FACE SIDESET]* *MMALE SIDESET* *MMALE XYFACE SIDESET* *MMALE YZFACE SIDESET* *MMALE XZFACE SIDESET* *[MMALE PLANAR FACE SIDESET]* *[MMALE CURVED FACE SIDESET]* *SEUL SIDESET* *SEUL XYFACE SIDESET* *SEUL YZFACE SIDESET* *SEUL XZFACE SIDESET*	*int*	Value corresponds to a sideset id and indicates how the nodes of the set should behave relative to the material motion. The *DOMAIN* keyword can have multiple entries of this type. The *LAGRANGIAN, EULERIAN, SMALE, MMALE, and SEUL SIDESET* commands control the setting of nodal flags on the nodes of the sideset. [...] indicates a future capability not yet functional
REMAP ITERATIONS	*int (1)*	Number of remaps to perform at the end of a Lagrangian step.
REMESH ITERATIONS	*int (10)*	Number of remesh smoothing iterations per remap.
INITIAL REMESH MOVEMENT LIMITER	*real (1.0)*	Fraction x of calculated smoothing movement to allow at problem startup. $0 \le x \le 1$. Use this to avoid large mesh movements at problem startup which can cause overfluxing in the advection phase of the remap.

Table 23: DOMAIN Keywords for Remesh Control (Continued)

Sub-Keyword	Numeric Input	Meaning
REMESH MOVEMENT RATIO	*real (1.0)*	Change remesh movement limiter by this factor each time through a remesh smoothing iteration. Generally ($x \geq 1$).
REMESH MOVEMENT LIMITER	*real (1.0)*	Maximum fraction of calculated remesh movement to allow ($0 \leq x \leq 1$).

3.4.6.2.2 DOMAIN Advection Controls

The *DOMAIN* keyword group also holds controls for the advection methods used by ALEGRA. These are given in **Table 24**.

Table 24: DOMAIN Keywords for Advection Control

Sub-Keyword	Numeric Input	Meaning
VOIDED SIDESET	*int*	Value corresponds to a valid sideset id and indicates that void exists on the other side of this Eulerian mesh boundary. By default, all Eulerian, Ale and Smoothed Eulerian mesh boundaries are assumed to reflect their conditions on the exterior of a problem. Lagrangian boundaries are by default free surfaces.
SALE ADVECTION *MSALE ADVECTION* *SHALE ADVECTION* *MSHALE ADVECTION* *HIS ADVECTION* *MHIS ADVECTION*		*(MHIS ADVECTION)* Turns on the indicated nodal advection method. These are documented in detail elsewhere [30].
SLIC INTERFACE TRACKER *SMYRA INTERFACE TRACKER* *NEW SMYRA INTERFACE TRACKER*		*(SMYRA INTERFACE TRACKER)* Turns on the indicated material interface tracker. SLIC is present only for testing purposes. NEW SMYRA is a recent modification of the algorithm by R. Bell.

The *VOIDED SIDESET* command controls the inflow/outflow of material from the mesh. By default, the material state outside the mesh will be the same as the material state just inside the mesh boundary. By using *VOIDED SIDESET* the user can tell ALEGRA that there is void outside the mesh and material can leave the mesh or void can enter the mesh. In using this capability, the Lagrangian boundary conditions control the motion of the material. So, if there is a no displacement boundary condition on the mesh boundary that inhibits motion normal to the boundary, then the *VOIDED SIDESET* command will have no effect on advection.

The nodal advection method concerns the advection of momentum between nodes of the mesh. All other advection concerns element centered quantities and advection is a matter of finding a value for the quantity on an element face and then moving a volume of the quantity between elements in a conservative manner. For momentum however, the ALEGRA velocities are centered on nodes and there is no convenient “face” to advect momentum through. Thus some method must be used to find the momentum associated with a node, advect this value and then redistribute to the nodes.

The interface tracker determines the location of the material interfaces within an element so advection can move the appropriate material into or out of a face. There is actually only one choice here. The SLIC interface tracker leads to spurious “streaming” effects in regular mesh. It is present in the code only for testing purposes.

3.4.6.2.3 INITIAL REFINEMENT

The *DOMAIN* keyword group also contains controls for the initial refinement capability of ALEGRA. Using this feature, a user can specify that a quadrilateral or hexahedral initial mesh can be refined prior to the problem starting. Each level of refinement multiplies the number of elements by a factor of 4 in 2D and 8 in 3D. ALEGRA has the capability of running with “1-irregular” mesh, meaning that an element may neighbor another element that has been refined to one higher level. Since this is restricted to a difference of only one level of refinement, neighboring blocks with different levels of initial refinement will have their borders refined such that the 1 level difference is enforced.

This “h-adaptivity” is a developing capability in ALEGRA and has not reached the level of full support for all users. However the capability is included here for the initial refinement, since often the user will want to refine the entire mesh. Initial refinement of the entire mesh is the recommended use of this feature. *DOMAIN* keywords controlling initial refinement are given in **Table 25**.

Table 25: DOMAIN Keywords for Initial Refinement

Sub-Keyword	Numeric Input	Meaning
MAXIMUM LEVELS	*int*	Specifies the maximum number of initial refinement levels.
INITIAL REFINEMENT		see **Section 3.4.6.2.3 on page 102**

INITIAL REFINEMENT

[subkeyword-list]

END

This input is contained as a subset within the *DOMAIN* input. Valid input identifies any combination of sidesets, element blocks, or detonation objects for initial refinement. During initialization all elements on a sideset, in an element block, or adjacent to a detonation object are refined to the maximum level of refinement. This capability is particularly useful for increasing the resolution of a given mesh using the original GENESIS file. Including all element blocks in the list of initial refinement objects, increases the mesh resolution by a factor of four in 2D and eight in 3D.

Table 26: INITIAL REFINEMENT Sub-Keywords

Sub-Keyword	Meaning
ALL BLOCK	Specifies that all element blocks will be refined to the highest refinement level.
BLOCK BOUNDARY	Specifies that elements on the boundaries of all element blocks will be refined to the highest refinement level.
BLOCK BOUNDARY, PATRIARCH	Same as block boundary, but refines elements whose patriarch element has a face on the block boundary.
SIDESET int	Refine the sideset to the highest refinement level.

Table 26: INITIAL REFINEMENT Sub-Keywords (Continued)

Sub-Keyword	Meaning
SIDESET int, PATRIARCH	Same as above but refines elements whose patriarch element has a face on the sideset.
SIDESET int SPHERE real vector	Refine the sideset to the highest refinement level while mapping surface nodes to a sphere of radius r centered at position p. In 2D CIRCLE is an alternate keyword here.
SIDESET int CYLINDER real vector vector	Refine the sideset to the highest refinement level while mapping surface nodes to a cylinder of radius r with vectors p1 and p2 lying on the axis of the cylinder. Primarily applicable in 3D.
SIDESET int RECONSTRUCT	Reconstructs a local smooth surface through the sideset nodes using approximate surface normals at these nodes.
BLOCK int	Refine the element block to the highest refinement level.
DETONATION POINT {int}	Refine the detonation point to the highest refinement level.

```
DOMAIN
  MAXIMUM LEVELS = 2
  INITIAL REFINEMENT
    SIDESET 1
    BLOCK 5
    DETONATION POINT 7
  END
END
```

Limitations:

1. Currently, the only detonation object supported for initial refinement is the detonation point (see Section 3.7.5.1 on page 135). Additional detonation objects will eventually be supported.
2. ***Nodesets cannot be used with initial refinement.*** Sidesets must be used to specify both Neumann (e.g., pressure) and Dirichlet (e.g., kinematic) boundary conditions.

Thus, for a 2D cylindrical geometry, the axial no displacement condition must be specified with a sideset, not a nodeset. The data layout of a nodeset inhibits correct refinement.

3.4.6.3 CELL DOCTOR

```
CELL DOCTOR
  [DISCARD int
    [FREQUENCY int]
    [MINIMUM POSITION, vector]
    [MAXIMUM POSITION, vector]
    [TIME RANGE MIN real MAX real]
    [TRIGGER, variable_name, MINIMUM real MAXIMUM real]
  END]
  ...
END
```

The *CELL DOCTOR* keyword provides facilities for modifying the material content of a cell in an *ad hoc* manner during a simulation. This section describes its most general form, usable from any physics package. A more highly specialized version of *CELL DOCTOR*, available under *HYDRODYNAMICS* and its children, is described in Section 3.7.4.2 on page 132.

3.4.6.3.1 DISCARD

The *CELL DOCTOR* keyword provides one subkeyword, *DISCARD.* This keyword has the syntax

```
DISCARD int
    [FREQUENCY int]
    [MINIMUM POSITION, vector]
    [MAXIMUM POSITION, vector]
    [TIME RANGE MIN real MAX real]
    [TRIGGER, variable name, MINIMUM real MAXIMUM real]
END
```

The DISCARD subkeyword provides the user with the capability to remove a selected material within a specified geometrical box when its trigger variable falls within the specified range.

The integer field after the DISCARD keyword can be any material id.

The trigger variable may be any scalar variable of the specified material. A listing of all variables for each material can be found in the .out file produced by an ALEGRA run. Examples of permissible trigger variables that are defined for many materials are DENSITY, PRESSURE, or TEMPERATURE. For example, one could simulate the effects of melting in a solid dynamics simulation with the input

```
CELL DOCTOR
 DISCARD 1 $ material 1 is a fusible material
  TRIGGER, TEMPERATURE, MINIMUM 1600K MAXIMUM 1.0e100K
 END
END
```

This specifies that material 1 is discarded when it reaches its melting temperature of 1600K. The material will be checked every cycle for all elements throughout the simulation, since the FREQUENCY, MINIMUM and MAXIMUM POSITION, and TIME RANGE keywords have not been specified. DISCARD thus can serve as a poor man’s substitute for more sophisticated damage or phase transition models, or even in conjunction with such models.

The user can enter as many DISCARD keyword blocks as desired. Overlapping discard specifications are “OR”ed together to determine if a material should be discarded.

Void is used to replace any material that is deleted. The mass and volume of material that is deleted is tracked and is available for output.

3.4.7 TRACER POINTS

TRACER POINTS

LAGRANGIAN TRACER int vector [NO INTERP]

EULERIAN TRACER int vector [NO INTERP]

ALE TRACER int vector [NO INTERP]

END

The *TRACER POINTS* keyword group begins a list of tracer points, which are used to track material or mesh motion in a calculation. As many tracers of any type may be included.

An Eulerian tracer never changes its spatial position. A Lagrangian tracer is moved with the interpolated velocity at its local position. It thus moves with the local material velocity. The natural mesh coordinates should not change for a Lagrangian tracer in a Lagrangian mesh. An ALE tracer stays with the original natural coordinates of the mesh and is thus a “mesh tracer.” An ALE tracer will be an Eulerian tracer in an Eulerian mesh and a Lagrangian tracer in a Lagrangian mesh and a mesh tracer in an ALE mesh. The ALE tracer is the most efficient tracer to use. Lagrangian tracers in Eulerian meshes and Eulerian tracers in Lagrangian meshes are the most expensive.

An example tracer specification is:

```
TRACER POINTS
  lagrangian tracer   1  x  1.0    y 2.0      z 3.0
  eulerian tracer     2  x  3.45   y 2.65     z 2.0
  ale tracer          3  x -0.009  y 198.345  z 3.3 no interp
  lagrangain tracer  30  x -0.009  y 198.345  z 3.3
  eulerian tracer   300  x -0.009  y 198.345  z 3.3
END
```

The location of the tracer can be modified using the *ROTATE* keyword which rotates the tracer about the axes in degrees and in the order given.

Tracer locations are written to the HISPLT post-processing database. History variables providing tracer locations in the local coordinate system (*PSX-X,PSY-Y,* and *PSY-Z*) are described in **Section 4.3 on page 245**. Currently HISPLT will not recognize an ALE tracer, listing the type of the tracer as UNKNOWN in the catalog but LAGRANGIAN on the plot header. This problem will be corrected in future versions of HISPLT.

By default, interpolation of the data to the tracer location in the element is performed for nodal variables. If no interpolation is desired, the keyword string “no interpolation” can be appended after the tracer coordinates for each tracer point. In that case, the value at the nearest node is written to the database. Interpolation is not yet available for element variables, so the value of an element variable written to the database is simply the value at the element center.

3.4.7.1 EULERIAN TRACER

[EULERIAN TRACER int vector [ROTATE, vector]]

In two dimensions, use *[ROTATE, Z real].* In three dimensions, specify the angle of rotation in degrees about each axis, such as ROTATE, x 30. y -10. z 45.

3.4.7.2 LAGRANGIAN TRACER

[LAGRANGIAN TRACER int vector [ROTATE, vector]]

In two dimensions, use *[ROTATE, Z real].* In three dimensions, specify the angle of rotation in degrees about each axis.

3.4.7.3 ALE TRACER

[ALE TRACER int vector [ROTATE, vector]]

In two dimensions, use *[ROTATE, Z real].* In three dimensions, specify the angle of rotation in degrees about each axis.

Table of Contents
General Input

3.5 Energetics Keywords

3.5.1 File I/O Additions

3.5.1.1 DETAILED ENERGY TALLIES

In many problems of interest it is often desirable to know the energy budget. How much energy is related to a given physical process? What is the value of the kinetic and internal energies? How much energy is supplied by a given source or is lost to a given sink? How fast does energy change from one form to another?

Alegra provides the user with a detailed set of energy and power tallies to answer such questions. The tallies are written to both the hisplt and EXODUS output files.Typically the hisplt file will contain tallies at more frequent intervals compared to the EXODUS file (depending on the user specification -- see the *EMIT HISPLT* and *EMIT PLOT* commands) because it is smaller and does not contain mesh information or plot variables.

The **Table 93** and **Table 94** in Chapter 4 summarize the various energy and power tallies that are available. The tallies are grouped by choice of the physics specification keyword, described in **Section 3.4.1 on page 74**. Extra global power tallies marked with an asterisk (*) are omitted from the output files unless the *DETAILED ENERGY TALLIES* keyword is specified.The *DETAILED ENERGY TALLIES* keyword causes extra global power tallies to be included in the history and plot dump files.

3.5.2 Energy Sources

3.5.2.1 ENERGY DEPOSITION

{EULERIAN | LAGRANGIAN} ENERGY DEPOSITION, block-id

RADIAL function-set

TIME function-set

[CYLINDRICAL]

[DENSITY real]

END

This keyword group causes energy to be deposited in the specified block over time. Use *EULERIAN* for Eulerian-mesh blocks and *LAGRANGIAN* for Lagrangian-mesh blocks. The power distribution is given as a separable function of radius from the origin and time, and it should be in dimensions of energy per mass per second. If the *CYLINDRICAL* keyword is included, the radius is interpreted to be the cylindrical radius, otherwise it is interpreted as the spherical radius. If the *DENSITY* keyword is included, the energy deposited is scaled by the current density divided by the value specified. This feature is an attempt to correct the energy deposited in problems where the density will change significantly over the time of the deposition.

The function sets used here support the *SHIFT* parameter described in Section 3.2.1.5 on page 55, which allows the user to choose the zero time.

ALEGRA deposits the energy into the material contained in a block of mesh. If the problem includes advection and the boundaries of the block are allowed to pass material, the mass contained within the block may change as a function of time. Since the energy deposition is in terms of energy per mass per second, the user may not get the total energy deposition that is desired. In cases where the energy deposition occurs very rapidly relative to material motion, one solution for this problem is to run the problem with the block specified as Lagrangian during the period of energy deposition and stop just after the deposition is finished. The problem can then be restarted with the block specified other than Lagrangian, so advection is allowed.

3.6 Mechanics Keywords

Mechanics algorithms deal with motion of material in space. The only released model for this type of algorithm is a dynamically evolving distribution of matter, modeled with explicit advancement of the variables at every step in time. Another class of Mechanics algorithms being developed for ALEGRA are statics models, where an implicit scheme is used to determine the final state of some material distribution subject to a set of internal and external forces.

3.6.1 Boundary Conditions and Body Forces

3.6.1.1 GRAVITY

GRAVITY vector

This keyword specifies that the force of gravity will be applied in a calculation in the direction specified. The gravitational potential energy is output as an energy diagnostic. Normally no gravity is included.

3.6.1.2 NO DISPLACEMENT

NO DISPLACEMENT, {nodeset | sideset} {X | Y | Z | R | RADIAL | NORMAL | TANGENT} [vector]

This keyword allows the user to hold one or more coordinates of a set of nodes fixed. If *X*, *Y, Z*, or *R* are specified, then *vector* should not be entered. Multiple *NO DISPLACEMENT {X | Y | Z | R}* boundary conditions may be specified for the same set of nodes to constrain motion in more than one direction. Note that in 3D or 2D Cartesian geometry, *X*, *Y* and *Z* refer to their standard components. In 2D cylindrical geometry, *R* and *Z* refer to the radial and axial components, respectively.

If *RADIAL* is specified, then *vector* must be entered. It is assumed that *vector* specifies the center through which the constraining force acts. Nodes are constrained to not move along the line connecting the node with *vector.*

If *NORMAL* or *TANGENT* is specified, then *vector* must be entered. It is assumed that the nodes are coplanar (colinear in 2D) and that *vector* specifies the outward boundary normal. Nodes are then constrained to not move in either the normal or tangential direction.

Note that if a node is constrained by both a {*NORMAL* | *TANGENT*} boundary condition and an *X*|*Y*|*Z*|*R* boundary condition and the boundary normals are not orthogonal, then application of the *NORMAL*|*TANGENT* constraint will add in an acceleration component along {*X* | *Y* | *Z* | *R*}. In this case, the common node should be placed in its own nodeset and multiple {*X* | *Y* | *Z* | *R*} constraints should be specified.

All nodes from the mesh database which are part of the specified nodeset or sideset will be subject to the boundary condition.

3.6.1.3 NO CYLINDRICAL DISPLACEMENT

NO CYLINDRICAL DISPLACEMENT, {nodeset | sideset}

{RADIAL | CIRCUMFERENTIAL | AXIAL}

This keyword allows the user to hold one or more coordinates of a set of nodes fixed in the radial, circumferential, or axial directions. Multiple *NO CYLINDRICAL DISPLACEMENT* boundary conditions may be specified for the same set of nodes to constrain motion in more than one direction.

3.6.1.4 PRESCRIBED FORCE

PRESCRIBED FORCE, nodeset, direction-function

This keyword allows the user to prescribe a force acting on a nodeset as a function of time in the direction specified by the *direction-function* construct as described in Section 3.2.1.9 on page 56.

3.6.1.5 RIGID SEGMENT

RIGID SEGMENT, nodeset

ENDPOINT1, vector

ENDPOINT2, vector

END

This keyword group allows the user to specify a rigid segment in two dimensions or, in three dimensions, a rigid plane that extends infinitely in the z direction parallel to the segment that cannot be penetrated by the nodes of the nodeset. In two dimensions, a series of rigid segments allows for the creation of a rigid body.

3.6.1.6 RIGID SURFACE

RIGID SURFACE, {nodeset | sideset}

CENTER, vector

NORMAL, vector

STATIC COEF real

VELOCITY COEF real

DECAY COEF real

END

This keyword group allows the user to specify a rigid, plane surface (not part of the database mesh) that cannot be penetrated by the (slave) surface specified by the *nodeset* or *sideset*.

3.6.1.7 TRACTION BC

TRACTION BC, sideset symtensor function-set

This keyword allows the user to specify a stress given by *symtensor* that acts on a surface. The stress is "dotted" with the unit normal of the element's surface to give a force that is proportioned and applied to the nodes that determine the surface

3.6.2 Mechanics Algorithm Control

3.6.2.1 HOURGLASS CONTROL

{PISCES | PRONTO} HOURGLASS CONTROL [int]

[parameter real]

END

Hourglass stiffening is added to a problem to couple the element hourglass modes to the element internal energy. This prevents runaway distortion from uncoupled, uncontrolled hourglass modes.

Two models for hourglass control are available: *PISCES*, and *PRONTO*, as described in the following subsections. The optional integer identifier after the *HOURGLASS CONTROL* keyword identifies an instance of the model that can be applied selectively to different blocks in the problem (see the ***BLOCK*** keyword).

Each model uses various *parameter* keywords to specify viscosity and/or stiffness values appropriate to the model.

3.6.2.1.1 PISCES

PISCES HOURGLASS CONTROL [int]

[VISCOSITY real (0.05)]

END

The bulk viscosity defined by the *VISCOSITY* keyword must be less than 0.25 to assure stability.

Table of Contents
General Input

3.6.2.1.2 PRONTO

PRONTO HOURGLASS CONTROL [int]

[VISCOSITY real (plane strain: 0.05) (cylindrical: 0.01)]

[STIFFNESS real (plane strain:0.0) (cylindrical: 0.03)]]

END

The stiffness coefficient is the most effective since it is adds a permanent restoring force rather than a viscous correction. What this does to the accuracy of the calculation is unclear. The PRONTO name is a misnomer since the coding attempts to follow [16] rather than what might be implemented in Pronto.

3.6.2.2 MOVING MESH

MOVING MESH vector

This keyword causes all Eulerian or smoothed Eulerian regions of the computational mesh to move through space at a prescribed velocity. The velocity of the material contained in the mesh is not prescribed. This keyword can be used to keep the mesh centered on the region of interest when the velocity of the region of interest is known *a priori*. The *BLOCK* input subkeyword *SMOOTHED EULERIAN MESH* may be helpful to smooth mesh irregularities with this option.

3.6.2.3 TRACK

TRACK, block-ids, [X | Y | Z | XY | XZ | YZ | TRANSLATION AND ROTATION]

When this keyword is in effect, the computational mesh in Eulerian or smoothed Eulerian regions moves at the mean velocity of the tracked blocks, so as to follow the blocks. In the case of smoothed Eulerian regions, the computational mesh is translated prior to smoothing. This is mainly useful in conjunction with the *SHISM* option when performing soft/hard interaction calculations.

By default, the computational mesh tracks the specified blocks in all coordinate directions. The user may specify that tracking takes place in only one or two directions.

The user may specify that both the translation and the rotation of the specified blocks should be tracked. This is useful for following the motion of a tumbling penetrator when using the *SHISM* option.

3.7 (Hydro)Dynamics Keywords

3.7.1 Dynamics Initial Conditions

3.7.1.1 INITIAL VELOCITY

INITIAL VELOCITY, nodeset, vector

This keyword allows the user to specify an initial velocity for each node in a nodeset. This input will take precedence over any values set by *INITIAL BLOCK VELOCITY* or *INITIAL ANGULAR VELOCITY* input. If the same node is in multiple nodesets then the last specified value will be used.

3.7.1.2 INITIAL ANGULAR VELOCITY

INITIAL ANGULAR VELOCITY, {block-id | nodeset}, CENTER, vector,VALUE, {real | vector}

This keyword allows the user to specify an initial angular velocity (radians/second) for each node of a *block* or a *nodeset*. The *CENTER* keyword specifies a point on the axis of rotation. The *VALUE* keyword specifies the angular velocity. In 2D Cartesian geometry, the *real* magnitude of the angular velocity is specified, as the axis of rotation is assumed to be orthogonal to the mesh, *i.e.*, along the positive z axis. In 3D the full angular velocity *vector* is specified, *i.e.*, the direction of the vector specifies the axis of rotation and the magnitude of the vector specifies the rate of rotation. The initial velocity of a node is the cross product of the angular velocity with the distance of the node from the center point. For example,

```
 INITIAL ANGULAR VELOCITY, NODESET 1
     CENTER, X 0.0 Y 0.0 Z 0.0
     VALUE,  X 0.0 Y 0.0 Z 6.283185E2
```

specifies 100 revolutions per second about the z axis.

The initial angular velocity keyword is not supported for 2D axisymmetric geometry.

3.7.1.3 INITIAL BLOCK VELOCITY

INITIAL BLOCK VELOCITY, block-id, {X | Y | Z} real

INITIAL BLOCK VELOCITY, block-id, {R | Z} real

or

INITIAL BLOCK VELOCITY, block-id, VECTOR, vector

or

INITIAL BLOCK VELOCITY, block-id, {RADIAL | NORMAL | TANGENT}, vector real

This keyword allows the user to specify an initial velocity for each node of a block. Three different forms of this keyword are available, as shown above.

In the first form, a principal direction (*X*, *Y*, or *Z* in Cartesian geometry or *R* or *Z* in 2D cylindrical geometry) is specified, followed by the magnitude of the velocity in that direction. For example,

```
  INITIAL BLOCK VELOCITY, BLOCK 3, X 1.0E4
```

will set the *x*-component of the velocity for each node in block 3 to 1.0e4.

In the second form, a velocity vector is specified after the *VECTOR* keyword. For example,

```
INITIAL BLOCK VELOCITY, BLOCK 3, VECTOR, X 1.0E4 Y 2.0E4 Z 1.0E4
```

will set the velocity of each node in block 3 to (1.0e4, 2.0e4, 1.0e4).

In the third form, the *RADIAL* keyword specifies that the velocity vector lies along the line from the point given by *vector* to each node in the block. The *TANGENT* keyword (2D only), on the other hand, specifies that the velocity vector tangent to the vector from each node to the point given by *vector*. In either case, this vector is normalized and multiplied by a velocity magnitude given by *real*. For example,

```
INITIAL BLOCK VELOCITY, BLOCK 3, RADIAL, X 0. Y 0. Z 0. -1.0E8
```

will initialize each node in block 3 to a velocity of magnitude 1.0e8 moving toward the origin.

Finally, the *NORMAL* keyword specifies a vector from the origin to the point given by *vector*. This vector is NOT normalized, and it is multiplied by the factor given by *real* and applied to each node in the block. For example,

```
INITIAL BLOCK VELOCITY BLOCK 3 NORMAL X 1.0 Y 2.0 Z 1.0 1.0E4
```

is equivalent to the previous example using the *VECTOR* keyword. If the same block is specified more than once or if a node is in multiple *block-id*s then the last specified value will be used.

3.7.1.4 RANDOM BLOCK VELOCITY

RANDOM BLOCK VELOCITY, block-id, VECTOR, vector

A random velocity will be generated for each non-zero vector component with a magnitude ranging between +/- of the specified vector component.

```
RANDOM BLOCK VELOCITY, BLOCK 1, VECTOR, X 2.0 Y 0.0 Z 0.0
```

will generate x-velocities ranging from -2 to 2.

3.7.1.5 SINUSOID VELOCITY

SINUSOID VELOCITY, nodeset, SCALE vector, LAMBDA real, {X | Y | Z | R} real

This keyword allows the user to specify an initial sinusoidal velocity perturbation for a given nodeset. *SCALE* ($\vec{v}_0$) specifies the absolute amplitude and direction of the perturbation, *LAMBDA* (λ) specifies the wavelength, and *X*, *Y*, *Z*, or *R* (x_0) specifies both the direction and the phase. *R* is restricted to the radial coordinate in 2D cylindrical simulations. Perturbations are calculated according to the formula:

$$\vec{v} = \vec{v}_0 \cdot cos\left(\frac{2\pi}{\lambda}(x - x_0)\right)$$

Example:

```
sinusoid velocity, nodeset 4, scale, x 0.001 y 0. lambda 1. y 0.
```

3.7.2 Dynamics Initial Density and Surface Perturbations

3.7.2.1 CYLINDRICAL MODE DENSITY

CYLINDRICAL MODE DENSITY, block-id, RANGE real,
[ANGULAR WAVENUMBER int,] [THETA real,]
[WAVELENGTH real,] [RADIAL WAVELENGTH real,] [RADIAL PHASE real,]
{X | Y | Z} real

This keyword allows the user to specify an initial cylindrical mode density perturbation for a given block. This keyword does not apply to 2D cylindrical geometry.

All parameters default to the value 0.0 if omitted. Except for the *block-id*, they may be listed in any order. Axial coordinates are specified in the native problem units (e.g., m or cm), while angles are specified in degrees.

The block should be cylindrical in shape and centered on one of the principal coordinate axes. *RANGE* specifies the relative amplitude ε of the perturbation.

The *ANGULAR WAVENUMBER* specifies the number of azimuthal periods N the perturbation has about the axis and *THETA* specifies the angular phase θ_0. If the wavenumber is zero, the perturbation will be rotationally uniform.

The *WAVELENGTH* specifies the axial wavelength λ. If the wavelength is zero, the perturbation will be axially uniform. If the wavelength and the wavenumber are both nonzero, the perturbation will be helical. The direction of rotation of the helix can be controlled by the sign of the wavelength.

The *RADIAL WAVELENGTH* specifies the radial wavelength λ_r. The *RADIAL PHASE* specifies the radial phase r_0. If the radial wavelength is zero, the perturbation will be radially uniform. The radius is computed relative to the cylindrical axis.

One of *X*, *Y*, or *Z* specifies both the coordinate axis that coincides with the cylinder axis and the axial phase z_0. (*Z* is permissible in 2D Cartesian geometry.) Perturbations are calculated according to the formula:

$$\rho = \rho_0 \cdot \left(1 + \varepsilon \cdot cos\left(\frac{2\pi}{\lambda}(z - z_0) + \frac{2\pi}{\lambda_r}(r - r_0) + N(\theta - \theta_0)\right)\right)$$

3D Cartesian example:

```
  cylindrical mode density, block 1,
     range       = 0.5
     ang wavenum = 3
     theta       = 0.  $ any multiple of 120. produces the same
     wavelength  = -1.
     z           = 0.
```

2D Cartesian example:

```
  cylindrical mode density, block 1,
     range       = 1.
     rad wave    = 1.
     rad phase   = 0.5
     z           = 0.   $ direction of axis
```

3.7.2.2 CYLINDRICAL MODE SURFACE

CYLINDRICAL MODE SURFACE, nodeset, RANGE real,

ANGULAR WAVENUMBER int, [THETA real,]

[WAVELENGTH real,] [RADIAL WAVELENGTH real,] [RADIAL PHASE real,]

{X | Y | Z} real

This keyword allows the user to specify an initial cylindrical mode surface perturbation for a given *nodeset*. This keyword does not apply to 2D cylindrical geometry.

All parameters default to the value 0.0 if omitted. Except for the *nodeset*, they may be listed in any order. Radial and axial coordinates are specified in the native problem units (e.g., m or cm), while angles are specified in degrees.

The original *nodeset* should be cylindrical in shape and centered on one of the principal coordinate axes. *RANGE* specifies the absolute amplitude ε of the perturbation. The direction of the perturbation is orthogonal to the cylinder's axis. The amplitude should be less than the width of a mesh element that abuts the surface, otherwise inverted elements may be generated.

The *ANGULAR WAVENUMBER* specifies the number of azimuthal periods N the perturbation has about the axis and *THETA* specifies the angular phase θ_0. If the wavenumber is zero, the perturbation will be rotationally uniform.

The *WAVELENGTH* specifies the axial wavelength λ. If the wavelength is zero, the perturbation will be axially uniform. If the wavelength and the wavenumber are both nonzero, the perturbation will be helical. The direction of rotation of the helix can be controlled by the sign of the wavelength.

The *RADIAL WAVELENGTH* specifies the radial wavelength λ_r. The *RADIAL PHASE* specifies the radial phase r_0. If the radial wavelength is zero, the perturbation will be radially uniform. The radius is computed relative to the cylindrical axis.

One of *X*, *Y*, or *Z* specifies both the coordinate axis that coincides with the cylinder axis and the axial phase z_0. (*Z* is permissible in 2D Cartesian geometry.) Perturbations are calculated according to the formula of the form:

$$\vec{x} = \vec{x}_0\left(1 + \frac{\varepsilon}{|\vec{x}_0|} \cdot cos\left(\frac{2\pi}{\lambda}(z - z_0) + \frac{2\pi}{\lambda_r}(r - r_0) + N(\theta - \theta_0)\right)\right)$$

3D Cartesian example shown in **Figure 10**:

```
cylindrical mode surface, nodeset 1,
   range       = 0.1
   ang wavenum = 4
   theta       = 0.  $ any multiple of 90. produces the same
   length      = 1.
   z           = 0.
```

2D Cartesian example shown in **Figure 11**:

```
cylindrical mode surface, nodeset 1,
   ang wavenum = 12
   theta       = 0.  $ any multiple of 30. produces the same
   range       = 0.02
```

Table of Contents
General Input

```
     z           = 0.
```

3.7.2.3 DEGENERATE SURFACE

DEGENERATE SURFACE, nodeset, POINT, vector *(2D or 3D)*

DEGENERATE SURFACE, nodeset, AXIS, vector1 vector2, TOLERANCE real (1e-12) *(3D only)*

This keyword allows the user to specify a degenerate surface for a given *nodeset*. A degenerate surface collapses all nodes down to a single point or a set of points along a line. The primary intent for this mesh modification is to remove “holes” in cylindrically or spherically symmetric meshes, therefore the user should construct a mesh that is a close approximation to the desired geometry. Even though many nodes appear to be one point, all points in the original *nodeset* are retained.

The *POINT* subkeyword collapses all nodes in the *nodeset* to the point specified by *vector*. This option is available in both 2D and 3D.

The *AXIS* subkeyword collapses all nodes in the *nodeset* to the line defined by *vector1* and *vector2*. *vector1* specifies a point on the line and *vector2* specifies the direction of the line and need not be normalized. The algorithm computes the perpendicular projection of a node to the axis. The *TOLERANCE* keyword is a matching criteria for grouping points along the axis into sets of degenerate points. Points that are separated by a distance less than the tolerance are treated as a single degenerate point. Points that are separated by a distance greater than the tolerance belong to different degenerate points. This option is available only in 3D since it makes no sense for 2D.

DEGENERATE SURFACE may be used in conjunction with other boundary conditions such as *DEGENERATE BC*, *NO DISPLACEMENT*, or *PRESCRIBED VELOCITY* to produce a variety of desired results.

3D example projecting a spherical cutout around the origin to the origin:

```
  degenerate surface, nodeset 1, point, x 0. y 0. z 0.
```

3D example projecting a cylindrical cutout along the z-axis to that axis:

```
  degenerate surface, nodeset 1, axis, x 0. y 0. z 0.,
                                       x 0. y 0. z 1.,
                                 tol   1.e-9
```

2D cylindrical example projecting a circular cutout around the origin to the origin:

```
  degenerate surface, nodeset 1, point, r 0. z 0.
```

3.7.2.4 RANDOM DENSITY

RANDOM DENSITY, block-id, RANGE real

This keyword allows the user to specify an initial random density perturbation for a given block. *RANGE* specifies the relative amplitude ε of the perturbation. Perturbations are calculated according to the formula:

$$\rho = \rho_0 \cdot (1 + \varepsilon(2 \cdot random(\ ) - 1))$$

where the random number is between zero and 1.

Example:

```
  random density, block 8, range 0.01
```

specifies a 1% random density perturbation.

3.7.2.5 RANDOM SURFACE

RANDOM SURFACE, nodeset, RANGE vector

This keyword allows the user to specify an initial random surface perturbation for a given nodeset.

RANGE specifies the absolute amplitude $\vec{x}_{range}$ of the perturbation components. Each coordinate for each node of the nodeset is randomly perturbed according to its respective amplitude component according to the formula:

$$\vec{x} = \vec{x}_0 + \vec{x}_{range} \cdot (2 \cdot random(\ ) - 1)$$

and the random number is between zero and one.

3D Cartesian example:

```
  random surface, nodeset 29, range, x 0.1 y 0.02 z 0.005
```

3.7.2.6 SINUSOID DENSITY

SINUSOID DENSITY, block-id, RANGE real, WAVELENGTH real, {X | Y | Z | R} real

This keyword allows the user to specify an initial sinusoidal density perturbation for a given block. *RANGE* specifies the relative amplitude ε of the perturbation, *WAVELENGTH* specifies the wavelength λ, and *X*, *Y*, *Z*, or *R* specifies both the coordinate direction and the phase x_0. *R* is restricted to the radial coordinate in 2D cylindrical simulations. Perturbations are calculated according to the formula:

$$\rho = \rho_0 \cdot \left(1 + \varepsilon cos\left(\frac{2\pi}{\lambda}(x - x_0)\right)\right)$$

Perturbations in multiple directions are defined by repeated use of this keyword.

Example:

```
  sinusoid density, block 8, range .01, wavelength 1., z 0.
```

3.7.2.7 SINUSOID SURFACE

SINUSOID SURFACE, nodeset, RANGE vector, WAVELENGTH real, {X | Y | Z | R} real

This keyword allows the user to specify an initial sinusoidal surface perturbation for a given nodeset. *RANGE* specifies the absolute amplitude $\vec{x}_{range}$ of the perturbation, *WAVELENGTH* specifies the wavelength λ, and *X*, *Y*, *Z*, or *R* specifies both the coordinate direction and the phase x_0. *R* is restricted to the radial coordinate in 2D cylindrical simulations. Perturbations are calculated according to the formula:

$$\vec{x} = \vec{x}_0 + \vec{x}_{range} \cdot cos\left(\frac{2\pi}{\lambda}(x - x_0)\right)$$

Perturbations in multiple directions are defined by repeated use of this keyword.

3D Cartesian example shown in **Figure 10**:

```
  sinusoid surface, nodeset 1,
     range,      x 0.0 y 0.0 z 0.02,
     wavelength, 1.,
     x 0.
```

2D Cartesian example:

```
  sinusoid surface, nodeset 5,
     range  = x 0.1,
     length = 1.0,
     y      = 0.
```

2D cylindrical example:

```
  sinusoid surface, nodeset 5,
     range  = x 0.05 y 0.,
     length = 1.0,
     z      = -0.25
```

3.7.2.8 TWISTED MESH

TWISTED MESH, [THETA0 real], [THETA1 real], [Z0 real], [Z1 real],
[R0 real], [R1 real], [R2 real], [R3 real]

This keyword allows the user to twist the mesh around the z axis in 2D or 3D. All parameters default to the value 0.0 if omitted. They may be listed in any order. Radial and axial coordinates are specified in the native problem units (m or cm), while angles are specified in degrees.

The mesh is rotated by the angle θ_0 when $z \le Z_0$ and by the angle θ_1 when $Z_1 \le z$. When $Z_0 < z < Z_1$, the point is projected to the $z = Z_0$ and $z = Z_1$ planes, the projected points are rotated, and the new coordinates are found by linear interpolation. Thus lines in the unmodified mesh transform to lines in the twisted mesh (perhaps with kinks at Z_0 and Z_1).

If only a radial annulus of the mesh between $R_1 \le R \le R_2$ is to be twisted, then the angles of rotation are adjusted to rise linearly from zero between R_0 and R_1, and then return linearly to zero between R_2 and R_3, providing smooth transition regions.

It is possible to divide the total angular twist θ equally between a counter rotation at Z_0 and a rotation at Z_1, i.e., let $\theta_0 = -\theta/2$ and $\theta_1 = \theta/2$. The following two examples actually produce similar results:

3D Cartesian examples shown in **Figure 10**:

```
  twisted mesh,
     theta 15.0,
     r0 2., r1 2.5, r2 3.5, r3 4.0
     z0 -2., z1 2.


  twisted mesh,
     theta0 -7.5,
     theta1  7.5,
     r0 2., r1 2.5, r2 3.5, r3 4.0
     z0 -2., z1 2.
```

For 2D Cartesian meshes, θ_0, Z_0 and Z_1 are ignored, and all rotations are based upon the angle θ_1 (in this case the parameter name *THETA* will match *THETA1*). Radial parameters still apply.

2D Cartesian example:

```
twisted mesh,
   theta 10.0
   r0 0.3, r1 0.5, r2 0.7, r3 0.9
```

For 2D cylindrical meshes, this command will transform straight lines parallel to the z axis into hyperbola of one sheet (and other curves accordingly), because the rotations are computed in 3D and then converted back to the RZ plane. The following two examples actually produce the same results:

2D RZ examples:

```
twisted mesh, theta0 -5.0, theta1 5.0
              r0 0.2, r1 0.4, r2 0.6, r3 0.8
              z0 0.0, z1 1.0


twisted mesh, theta 10.0,
              r0 0.2, r1 0.4, r2 0.6, r3 0.8
              z0 0.0, z1 1.0
```

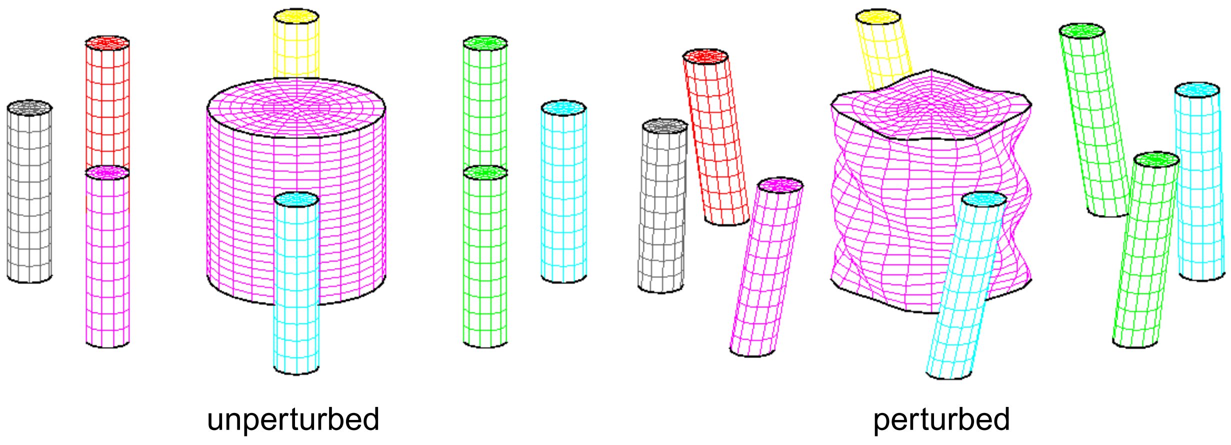


Figure 10: Initial perturbations in 3D Cartesian geometry.

A CYLINDRICAL MODE SURFACE perturbation is applied to the lateral surface of the central cylinder, while a SINUSOID SURFACE perturbation is applied to its top surface. The outer ring of cylinders shows the effect of a TWISTED MESH perturbation.

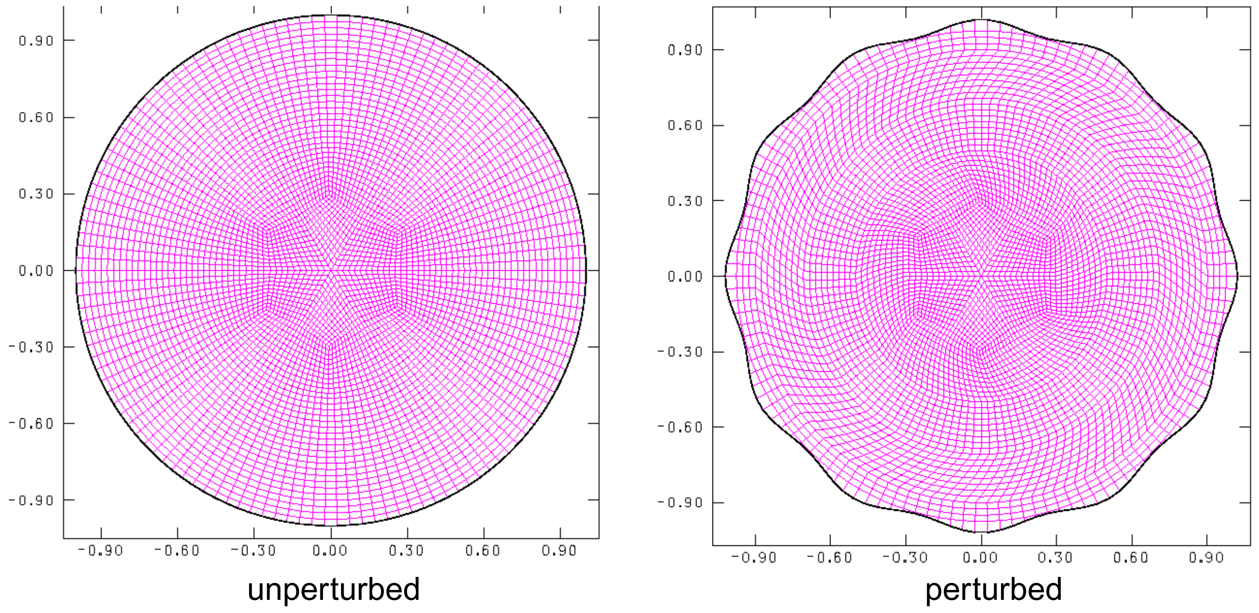


Figure 11: Initial perturbations in 2D Cartesian geometry.

A CYLINDRICAL MODE SURFACE perturbation is applied to the outer boundary of the circular mesh, while the central portion shows the effect of a TWISTED MESH perturbation.

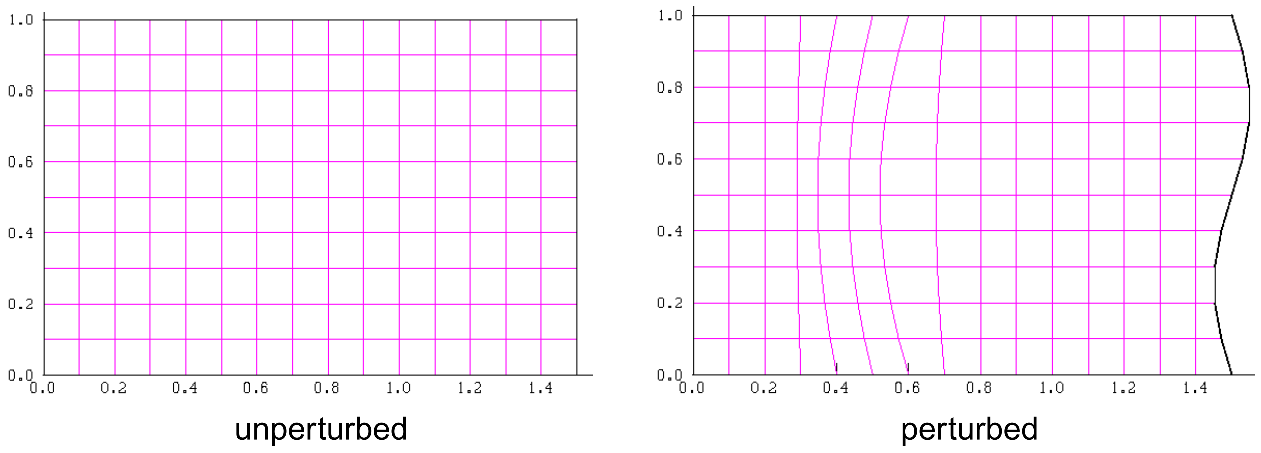


Figure 12: Initial perturbations in 2D cylindrical geometry.

A SINUSOID SURFACE perturbation is applied to the outer radius of the mesh, while the interior hyperbolic curves show the effect of a TWISTED MESH perturbation.

Table of Contents
General Input

3.7.3 Dynamic Boundary Conditions

3.7.3.1 DEGENERATE BC

DEGENERATE BC, nodeset, POINT, vector *(2D or 3D)*

DEGENERATE BC, nodeset, AXIS, vector1 vector2, TOLERANCE real (1e-12) (3D only)

This keyword allows the user to specify a degenerate boundary condition for a given *nodeset*. A degenerate boundary condition constrains all nodes belonging to a degenerate point to move at the same center-of-mass velocity. *DEGENERATE BC* is intended to be used in conjunction with the *DEGENERATE SURFACE* surface perturbation to keep degenerate nodes together.

The *POINT* and *AXIS* subkeywords are for compatibility with the *DEGENERATE SURFACE* keyword and are not used. The *TOLERANCE* keyword is a matching criteria for grouping points along the axis into sets of degenerate points. Points that are separated by a distance less than the tolerance are treated as a single degenerate point. Points that are separated by a distance greater than the tolerance belong to different degenerate points.

DEGENERATE BC may be used in conjunction with other boundary conditions such as *NO DISPLACEMENT* or *PRESCRIBED VELOCITY* to produce a variety of desired results.

3D example projecting a spherical cutout around the origin to the origin:

```
degenerate bc, nodeset 1, point, x 0. y 0. z 0.
```

3D example projecting a cylindrical cutout along the z-axis to that axis:

```
degenerate bc, nodeset 1, axis, x 0. y 0. z 0.,
                                x 0. y 0. z 1.,
                          tol   1.e-9
```

2D cylindrical example projecting a circular cutout around the origin to the origin:

```
degenerate bc, nodeset 1, point, r 0. z 0.
```

3.7.3.2 PRESCRIBED ACCELERATION

PRESCRIBED ACCELERATION, nodeset, {X | Y | Z}, function-set

This keyword allows the user to prescribe an acceleration for a nodeset in the direction indicated as a function of time.

3.7.3.3 PRESCRIBED VELOCITY

PRESCRIBED VELOCITY, {nodeset | sideset}, direction-function

This keyword allows the user to prescribe the velocity of a nodeset or sideset as a function of time as specified by the *direction-function* keyword.

Example:

```
prescribed velocity, nodeset 1, y, function 1
function 1
  0.0  1000.0E5
  1.0  1000.0E5
end
```

3.7.3.4 PRESSURE BC

PRESSURE BC, sideset function-set

or

PRESSURE BC, sideset

[MOMENT int function-set]

[MOMENT int function-set]

...

END

This keyword allows the user to specify a spatially constant pressure that acts on a surface as a function of time. Alternatively, a keyword group may be used to specify the pressure over the surface as a set of spherical harmonic moments.

3.7.3.5 PRESSURE WAVE

PRESSURE WAVE, sideset, vector, PEAK, function-set, RISE, function-set

PROPAGATION SPEED real, ARRIVAL TIME real

This keyword allows the user to simulate a shock wave resulting from a disturbance at a single point given by the *vector* keyword.

3.7.3.6 CONTACT SURFACE (2D algorithm)

CONTACT SURFACE, sideset sideset

[STATIC FRICTION real]

[PARTITION FACTOR real]

[VELOCITY FRICTION real]

[VELOCITY DECAY real]

[RETRACK INTERVAL real]

END

This keyword group identifies a pair of surfaces that may come into contact during the calculation. By using this keyword, the user ensures that the two surfaces will not penetrate each other, but instead will exert a force on each other to prevent penetration. **This input format is valid only for 2D calculations.**

The partition factor determines the master/slave relationship. A value of 0 makes the first surface the complete master and the second the complete slave. The opposite is true for a value of 1. The default (0.5) gives a symmetric relationship. If one surface is much more massive, it should be given a stronger master role. Otherwise, the symmetric value should be used, since it is the only one which conserves momentum.

The velocity decay factor (gamma) determines how the coefficient of friction changes from static to kinetic as a function of relative velocity. A large value implies a rapid change to kinematic friction when the surfaces begin moving with respect to each other.

3.7.3.7 GLOBAL CONTACT (3D global algorithm)

Alegra uses the ACME library [9] for the 3D contact algorithms. Contact is ***not*** currently supported with adaptivity or dynamic load balance. The "contact surface" for ACME is made up of a collection of sidesets, element block exteriors and analytic surfaces. The contact behavior between these entities can be specified pair by pair or at once through default data. The frictional behavior of the contact is specified using friction models (analogous to material models for elements). Data must be specified for every pair of sidesets, element blocks and analytic surfaces in the problem. Suitable defaults do not exist that will work for all problems so the user is required to input at least default data.

```
GLOBAL CONTACT
    [SIDESET | BLOCK] int
     :
    [additional SIDESET or BLOCK keywords]
     :
    [ANALYTIC SURFACE]
      TYPE =  [PLANE | CYLINDER INSIDE | CYLINDER OUTSIDE | SPHERE]
     [Type Specific Subkeywords (see Table 27)]
    [END]


   [additional analytic surface keywords]
```

```
    [ENFORCEMENT ITERATIONS int]
    [TRIVIAL MASS real]


   [DEFAULT DATA]
        [SEARCH NORMAL TOLERANCE real]
       [SEARCH TANGENTIAL TOLERANCE real]
       [KINEMATIC PARTITION [AUTO | real] ]
       [FRICTION MODEL ID  int]
    [END]
     :
   [DATA [SIDESET|BLOCK|ANALYTIC SURFACE] int,
             [SIDESET|BLOCK|ANALYTIC SURFACE] int   ]
        [SEARCH NORMAL TOLERANCE real]
       [SEARCH TANGENTIAL TOLERANCE real]
       [KINEMATIC PARTITION [AUTO | real] ]
       [FRICTION MODEL ID  int]
    [END]


    [additional data blocks]


     [FRICTION MODEL ID [FRICTIONLESS | CONSTANT | TIED | SPOT WELD |
            PRESSURE DEPENDENT |VELOCITY DEPENDENT ]    ]
         [Type Specific Keywords (see Table 28)]
    [END]
       :
     [additional friction model keyword blocks]
       :
```

END

This keyword group identifies the surfaces that may come into contact with other similarly specified surfaces during the calculation. Contact of a surface with itself is allowed and the surface must be specified if this is anticipated. By using this keyword block, the user ensures that the specified surfaces will not interpenetrate, but instead will exert a force to prevent penetration. For convenience, a block-id may be specified instead of a sideset so that all exterior surfaces of that block will be treated as a contact surface. However, if parts of the block will never come into contact with other surfaces, this can result in significant inefficiencies and slow down a calculation unnecessarily.

Table 27: Subkeywords for analytic surfaces.

Type	Subkeyword	Argument	Description
Plane	*Point*	*X real, Y real, Z real*	The x, y, z locations of a point in the plane
	Normal	*X real, Y real, Z real*	The x, y, z components of a vector normal to the plane.
Cylinder Inside & Cylinder Outside	*Center*	*X real, Y real, Z real*	The x, y, z locations of the center of the cylinder
	Axis	*X real, Y real, Z real*	The x, y, z components of a vector defining the axis of the cylinder
	Radius	*real*	The radius of the cylinder
	Length	*real*	The length of the cylinder.
Sphere	*Center*	*X real, Y real, Z real*	The center of the sphere.
	Radius	*real*	The radius of the sphere.

Table 28: Subkeywords for friction models.

Type	Subkeyword	Argument	Description
Frictionless			There are no subkeywords.
Constant	*Friction Coefficient*	*real*	The coefficient of friction.
Tied			There are no subkeywords.
Spot Weld	*Normal Capacity*	*real*	The normal force capacity of the spot weld.
	Tangential Capacity	*real*	The tangential force capacity of the spot weld.
	Failure Steps	*int*	The number of time steps over which the failure will occur.
	Failed Model ID	*int*	The ID of the friction model to be used once failure has occurred.
Pressure Dependent	*Friction Coefficient*	*real*	The coefficient of friction.
	Reference Pressure	*real*	The reference pressure for the pressure dependence.
	Pressure Exponent	*real*	The exponent to be used in the pressure dependence.
Velocity Dependent	*Static Coefficient*	*real*	The coefficient of friction if the two surfaces are “sticking”.
	Dynamic Coefficient	*real*	The coefficient of friction if the two surfaces are “sliding”.
	Velocity Decay	*real*	The decay exponent to transition between the static and dynamic coefficients of friction.

An arbitrary number of “surfaces” may be specified by an arbitrary number and mixed set of sidesets, blocks, and analytic surfaces. For example, to specify the surfaces given by sidesets 11 and 12 plus the exterior surfaces of block 3 with frictionless contact, the minimum input would appear as the following:

```
  GLOBAL CONTACT
```

```
    SIDESET 11
    SIDESET 12
    BLOCK 3
    DEFAULT DATA
      FRICTION MODEL ID = 1
    END
    FRICTION MODEL ID 1 FRICTIONLESS
      $no subkeywords
    END
  END
```

To specify parameters (e.g., friction coefficients) that govern how a pair of surfaces will interact, the *DATA* (or *DEFAULT DATA*) keyword subgroup should be used. The kinematic partition factor determines the master/slave relationship. The default behavior is to use the automatic, dynamic determination built into ACME. This can be overridden with a specific value but should be avoided unless the user is very familiar with the ACME algorithms. A poor choice of kinematic partition value by the user can cause energy generation, and in extreme cases element inversion. A value of 0 makes the first surface the complete master and the second the complete slave. The opposite is true for a value of 1. The explicit transient dynamic enforcement provided by ACME includes an iterative capability to increase the accuracy of the contact forces. By default, the number of iterations is set to 5. This has been found to be a good trade-off between speed and accuracy. The ACME enforcement has an implicit assumption of Lagrangian meshes. The implication is that every node MUST have a non-zero mass. For SHISM and other ALE problems, it is possible to have zero mass nodes (i.e., void filled elements). To account for this, ALEGRA assigns a "trivial mass" to all zero mass nodes to prevent division by zero. The default value is 1.0E-12 but can be reset with the *TRIVIAL MASS* subkeyword.

3.7.3.8 SHISM

The SHISM feature is used for calculations involving penetration of soft material by a hard structure. The input specification differs depending on the dimensionality of the problem. For a two dimensional problem, the input is described by the following.

SHISM, soft_sideset hard_sideset

  [STATIC FRICTION real]

  [PARTITION FACTOR real]

  [VELOCITY FRICTION real]

  [VELOCITY DECAY real]

  [RETRACK INTERVAL real]

END

The first sideset, here labeled soft_sideset, should be a sideset of a *SMOOTHED EULERIAN MESH* (see the ***BLOCK*** keyword group) corresponding to the soft material, while the second sideset, labeled hard_sideset, should be a sideset of a *LAGRANGIAN MESH* corresponding to the hard structure. The two sidesets must correspond exactly, in the sense that each node of the Smoothed Eulerian sideset must be at the same location as exactly one node of the Lagrangian sideset. The two sets of nodes must be logically distinct. Typically the two sidesets will be closed, with the Lagrangian region contained within the Smoothed Eulerian region, and the frame of reference chosen is that in which the hard structure is at rest.

ALEGRA treats the specified sidesets as contact surfaces (currently using the older contact algorithms), but each node of the Smoothed Eulerian sideset is repositioned to the location of the corresponding node of the Lagrangian sideset whenever a remesh takes place. Normally, the remesh frequency should be set to 1 and the Smoothed Eulerian sideset should be flagged as a *VOIDED SIDESET* in the *DOMAIN* keyword group.

Best results seem to be obtained by choosing the *BUDGE REMESH METHOD* in the ***BLOCK*** keyword group and by specifying the *TRACK* keyword option for the Lagrangian mesh.

For three dimensional problems, the 3D SHISM algorithm now specifies contact in 3D only from the sidesets that are generated during the mesh “shattering” process (these are denoted in the example below as sideset 501 and 502). The choice of ‘blocks’ or ‘sidesets’ has been removed for the global contact option. The following illustrates the 3D interface:

```
SHISM,
  soft: block 10, sideset 501
  hard: block 1 2 3, sideset 502
  [face2face]
  [CONTACT DATA
    kinematic partition 2.0
    friction static 0.1
    :
    :
  END]
TRACK, block 2
```

3.7.3.9 TRANSMITTING BC

TRANSMITTING BC, sideset, BULK IMPEDANCE real, SHEAR IMPEDANCE real

This keyword specifies a no reflection boundary condition, applicable to semi-infinite media, along the surface specified by *sideset*. The acoustic impedances for p- and s-waves are specified after the *BULK IMPEDANCE* and *SHEAR IMPEDANCE* keywords, respectively. The acoustic impedance for p-waves can be calculated from the bulk and shear moduli K and G as:

$$sqrt\left(\rho\left(K+\frac{4}{3}G\right)\right)$$

3.7.4 Dynamics Algorithm Control

3.7.4.1 BULK ARTIFICIAL VISCOSITY

PRONTO ARTIFICIAL VISCOSITY [int]

[LINEAR real(0.15)]

[QUADRATIC real(1.2)]

[CUTOFF real(1.0e-10)]

[TENSION LINOFF]

[TENSION QUADON]

END

Artificial viscosity is added to shock problems to “smear” the shock front across several elements and prevent numerical oscillation behind the shock. The optional integer field identifies an instance of the model that can be accessed selectively by different blocks in the problem (see the ***BLOCK*** keyword).

The *CUTOFF* keyword specifies a minimum absolute value for the artificial viscous pressure, below which it is set to zero. The *TENSION LINOFF* keyword causes the linear component of the artificial viscous pressure to be set to zero when the material is in tension. The *TENSION QUADON* keyword causes the quadratic component of the artificial viscous pressure to *not* be set to zero when the material is in tension (i.e. to have a non-zero value in elements in tension). It is rarely appropriate to leave the quadratic artificial viscous pressure on in tension, but it is recommended [4] that the linear artificial viscosity pressure be left on for elastic materials and turned off for ideal gases.

3.7.4.2 HYDRO CELL DOCTOR

HYDRO CELL DOCTOR

[TIME STEP RATIO real]

[SOUND SPEED RATIO real]

[VOLUME FRACTION CUTOFF real]

[all CELL DOCTOR options]

END

HYDRO CELL DOCTOR is a specialization of the *CELL DOCTOR* keyword (Section 3.4.6.3 on page 105) applicable to *HYDRODYNAMICS* and its children. In addition to the facilities provided by *CELL DOCTOR*, the *HYDRO CELL DOCTOR* keyword provides facilities with which the user can attempt to alleviate the problem of a small fragment with a completely unrealistic thermodynamic state controlling the time step of the entire problem. The idea is that a very small fragment with a completely unrealistic sound speed should be removed from the problem. Logic has been added to the code to allow the user to eliminate such small fragments from the calculation.

One does not have to enter all the data above; however, it is unlikely that the default values will allow any material to be removed. All three conditions must be met to result in the removal of material. Once the fragment is removed, its volume fraction is proportioned out to the other materials in the cell, including void.

The *TIME STEP RATIO* subkeyword specifies the ratio of the two smallest cell time steps. For example, if cell A has a limiting time step of 1.0e-6 and cell B has the next smallest time step of 1.0e-5 then the ratio would be 0.1. Therefore the value specified should be between 0 and 1, where the smaller the value, the more unlikely that a fragment will be removed.

The *SOUND SPEED RATIO* subkeyword specifies the ratio of the two largest material sound speeds in the limiting cell. The value specified should be greater than 1, where the larger the value, the more unlikely that a fragment will be removed.

The *VOLUME FRACTION CUTOFF* subkeyword specifies the maximum fragment size that can be removed. The value specified should be between 0 and 1, where the smaller the value, the more unlikely that a fragment will be removed.

From the user input above, it is obvious that a user could easily empty the time step limiting cell every time step and eventually totally destroy the simulation. Therefore, extreme care must be used when this option is employed in order to maintain a credible calculation.

3.7.4.3 MATERIAL FRACTION FORCE LIMIT

MATERIAL FRACTION FORCE LIMIT real [POWER real(0.0)]

The MATERIAL FRACTION FORCE LIMIT keyword allows the user to limit the forces acting on mixed material/void elements. For some coupled physics options the force on the material in a mixed material/void element might not scale proportionately with the amount of material present in the element. In the limit that the material fraction (i.e., 1 - void_frac) tends to zero, the forces on the nodes surrounding the element may be disproportionately large compared to the mass of the node. In this case the acceleration of the nodes may become large causing the mesh to tangle by inverting elements. This keyword is intended to rectify this situation by limiting the forces on the nodes. *This feature is only functional for serial problems; its use in parallel problems will lead to an abort.*

After the forces are determined for each node in the mesh, the material fractions of attached elements are averaged. If the average material fraction is less than the specified limit, the nodal force is limited, otherwise the force is not changed. The multiplier of the nodal force used when average material fraction (matfrac) is less than the specified limit is given by the following expression.

$$\begin{array}{ll} 0 & , \; p = 0 \\ \left(\dfrac{\text{matfrac}}{\text{limit}}\right)^{p} & , \; p > 0 \end{array}$$

For p = 0 the limiter is a step function, for p = 1 the limiter is linear, for p > 1 the limiter drops rapidly near the threshold, and for p < 1 the limiter drop rapidly near 0. Graphically the multiplier can be displayed as:

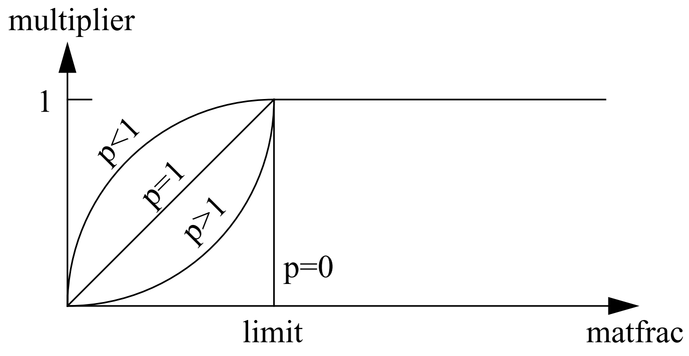


Figure 13: Material Fraction Force Limiter

3.7.4.4 MAXIMUM VOLUME CHANGE TIME STEP CONTROL

MAXIMUM VOLUME CHANGE real(0.2)

This keyword specifies the maximum volume change permitted for an element in a single cycle, as a fraction of the original element volume.The time step will be limited to ensure that this volume change limit will be observed.

3.7.4.5 MINIMUM ELEMENT SIDE TIME STEP CONTROL

MINIMUM ELEMENT SIDE TIMESTEP CONTROL

The internal calculation of the time step also may underestimate the maximum stable time step because of a conservative calculation of the characteristic element length. A typical use of the *MINIMUM ELEMENT SIDE TIMESTEP CONTROL* keyword would be in Eulerian calculations, where the analyst may designate that the minimum element side be used in the calculation of the maximum stable time step, which will typically result in time steps similar to what the CTH algorithm would produce. For Lagrangian or ALE calculations, however, use of this option may not give stable behavior if the mesh becomes too distorted by shear.

Example:

The following input will cause the time step used in a calculation to be one half of the value ALEGRA would normally use and will use the minimum side length for the characteristic dimension of an element.

```
TIME STEP SCALE 0.5
MINIMUM ELEMENT SIDE TIMESTEP CONTROL
```

3.7.5 Dynamics Supplementary Models

3.7.5.1 PROGRAMMED BURN

```
PROGRAMMED BURN
  [NONBURNING ELEMENTS ALLOWED]
  MATERIAL int
    [DETONATION POINT, vector, AT TIME real, BURN RADIUS real]
    [DETONATION LINE,  vector1, vector2, AT TIME real, BURN RADIUS real]
    [DETONATION PLANE,  vector1, vector2, vector3 AT TIME real,
       BURN RADIUS real]
    [BURN TRANSITION RATIO real]
    [BURN FRONT THICKNESS real]
  [MATERIAL int]
    [DETONATION POINT, vector, AT TIME real, BURN RADIUS real]
    [DETONATION LINE,  vector1, vector2, AT TIME real, BURN RADIUS real]
    [DETONATION PLANE,  vector1, vector2, vector3 AT TIME real,
       BURN RADIUS real]
    [BURN TRANSITION RATIO real]
    [BURN FRONT THICKNESS real]
END
```

The programmed burn option allows simulation of a detonation by considering detonation points, lines, or planes to be energy sources. Detonation times are calculated from information provided with the detonation object to determine the detonation velocities throughout the burn material. If a burn due to pressure loading or shock is desired, then one of the KEOS SESAME two-state burn models is recommended (See ***Section 3.9.3.4 on page 157***)

If some of the burn material is discontinuous from the rest, burn times will not be calculated and detonation would not be simulated in the discontinuous section. The default behavior is to stop the calculation from proceeding. The *NONBURNING ELEMENTS ALLOWED* keyword is an optional flag that allows the calculation to proceed if detonation times cannot be calculated for some of the programmed burn material. If this keyword is used, it must be the first keyword in the *PROGRAMMED BURN* input section; it will apply to all programmed burn materials.

The *MATERIAL* number must be a valid material id that incorporates the *PROGRAMMED BURN JWL* model or the *KEOS JWL* burn model with *BRN*=1. Multiple burn materials are allowed; the input for each material is considered complete when the subsequent *MATERIAL* keyword is read, or when the *END* keyword is read

At least one detonation object (point, line, or plane) must follow the *MATERIAL* input. The *DETONATION POINT/LINE/PLANE* command specifies the location of the primer for a high explosive, the time of detonation, and the burn radius within which no obstacles or corners will be circumvented. Within the *BURN RADIUS*, the radial distance between each cell and the detonation point is used to calculate the detonation time. (Thus, any intervening non-explosive materials are completely ignored within the burn radius). The burn radius must encompass at least one vertex of an element in the material for which the detonation object is defined. If no cells are found to be within the region defined by the detonation object plus burn radius, the default behavior is to force a fatal code error. Detonation points may be specified for use in one, two or three dimensional geometries. Detonation lines are specified by the two end points of the line. Lines may be specified for use in two or three dimensional geometries. Detonation planes are specified by any three points in the plane; the plane is assumed to be of infinite extent. Detonation planes may only be specified for use in three dimensional geometries.

The *BURN TRANSITION RATIO* (default=2.) is the ratio applied to the amount of energy added in a cycle (less than one decreases the time for burn, greater than 1 increases the time). The *BURN FRONT THICKNESS* (default=2.) is the factor used to multiply the largest element side. The result is applied to the amount of energy added in a cycle (less than 1 decreases the time for burn, greater than 1 increases the time).

Note that Alegra provides for the detonation front to “turn a corner” so that the minimum route along cell centers is calculated within the burn material region, outside the burn radius. This is not the minimum radial distance but provides satisfactory burn contours for many problems of interest. If the burn material is not continuous, an additional detonation point with the appropriately modified detonation time will be required in each discontinuous region if detonation is to be simulated in that region. The detonation velocity and other burn material parameters are input with the ***PROGRAMMED BURN JWL*** model or the ***KEOS JWL*** input set.

Note that, if the explosive can be detonated by the shock wave alone (reactive burn model) and the shock propagates through the material surrounding the explosive at a speed comparable to the detonation front, the reactive burn configuration may be a better representation of the problem.

3.7.5.2 INTERMATERIAL FRACTURE

FRACTURE

VOID, real

MIXED, real

END

The intermaterial fracture option allows material in multimaterial elements to separate. Normally, in a multimaterial element, materials behave as in a welded fashion. Thus, if two materials are moving apart from each other in an ALE or Eulerian mesh, tension will form in the elements containing the material interfaces and work to restrain the motion. By allowing INTERMATERIAL FRACTURE, tension at the material interfaces is limited by allowing the materials to fracture in that element, effectively allowing the materials to separate. The VOID and MIXED inputs specify the pressure response in mixed elements containing void and material or multiple materials, respectively.

This algorithm also looks at adjacent elements connected through a face. If the element is adjacent to an element of a different material, the tensile pressure limits are also applied. This allows for intermaterial fracture between an ALE or Eulerian element and a single material Lagrangian element where the multimaterial condition would never be met, but separation of different materials is allowed.

The values of the fracture pressures are negative, indicating tensile pressure. Also, fracture material models must be defined for all materials.

3.8 Adaptivity Control Parameters

ALEGRA provides the capability of dynamically refining the mesh during a simulation. This refinement, which divides elements into uniformly smaller subunits, is called H-adaptivity. Currently, this feature is provided only for 2D quadrilateral meshes and 3D hexahedral meshes. In two dimensional calculations, a quadrilateral element will be divided into four smaller quadrilaterals and, in three dimensions, a hexahedron will be divided into 8 uniformly smaller hexahedral elements. This capability can be specified to take place at simulation initiation, via the INITIAL REFINEMENT keyword in the DOMAIN specification (see **Section 3.4.6.2.3 on page 102**). Refinement can also take place dynamically, as the simulation is run. In the later case, the adaptivity option is controlled by user selections in the ADAPTIVITY SPECIFICTION input section, which is placed outside of the overall physics input specification. The dynamic refinement is driven by error metrics that are computed in each element in specified regions. Those elements exceeding some limit for the error metric value will be refined, subject to limitations on the number of levels of refinement desired by the user. Similarly, refined elements where the error metric falls below a threshold value will be removed and combined into the element one level higher in the refinement hierarchy.

In this section of input, the controls on the dynamic refinement are specified. ALEGRA currently supports only one error metric to control mesh refinement. This is the traction jump error metric. The traction is defined at an element face and is the dot product of the element stress tensor and the outward normal to the face. The traction jump at a face is then the difference in traction computed using the stress tensors in the two elements sharing a face. The error metric for an element is the average face traction jump. (Note that specialized versions of ALEGRA have included other error metric controls specific to the additional physics being modeled by those special versions.)

3.8.1 Adaptivity Controls

```
ADAPTIVITY SPECIFICATION
  ENABLE ADAPTIVITY
  JUMP METRIC
    [subkeywords]
  END
 ELEMENT BUDGET
    [subkeywords]
  END
  LAYERING
   [subkeywords]
  END
  UNREFINEMENT CONTROL
   [subkeywords]
  END
  BLOCK ADAPT LEVELS
END
```

3.8.1.1 ENABLE ADAPTIVITY

The ENABLE ADAPTIVITY keyword indicates to ALEGRA that the adaptivity features will be active for the simulation. The presence of this keyword in the ADAPTIVITY SPECIFICATION input section is a requirement for adaptivity to operate in a simulation.

3.8.1.2 JUMP METRIC

JUMP METRIC

JUMP REFINE THRESHOLD real

JUMP UNREFINE THRESHOLD real

JUMP NORMALIZER real

JUMP TRACTION SCALING

JUMP TRACTION FLOOR real

END

The jump metric measures the traction difference across a surface of an element. The magnitude of this difference is maximized across all surfaces of the element to obtain a value on the element. The resulting value can be used to indicate error on the element. To use this value to drive the adaptivity, refinement and unrefinement thresholds must be defined. There are two ways to do this: set the JUMP REFINE THRESHOLD and JUMP UNREFINE THRESHOLD to traction values that the problem is known to exhibit (if given, the JUMP NORMALIZER will divide all values before applying the thresholds), or indicate JUMP TRACTION SCALING which will pick a normalizer based on the average value of the tractions across the whole mesh.

Table 29: ADAPTIVITY Keywords for JUMP METRIC

Sub-Keyword	Numeric Input	Meaning
JUMP REFINE THRESHOLD	*real*	Value of traction jump error value above which to refine an element
JUMP UNREFINE THRESH-OLD	*real*	Value of traction jump error value below which to unrefine a previously refined element
JUMP NORMALIZER	*real*	Value by which to divide the traction jump error values to obtain a value to compare to refine/unrefine thresholds
JUMP TRACTION SCALING		Flag keyword to indicate that the jump error values are to be scaled by the average value found among all faces
JUMP TRACTION FLOOR	*real*	Minimum value of the jump error metric above which comparisons to the refine/unrefine thresholds will occur.

An example input is:

```
  ADAPTIVITY SPECIFICATION
```

```
    JUMP REFINE THRESHOLD 0.75
    JUMP UNREFINE THRESHOLD 0.50
    JUMP TRACTION SCALING
    JUMP TRACTION FLOOR 10.0
  END
```

which would automatically scale the traction values to the average over the whole mesh and ignore values that are less than 10.0 (which can be used to prevent the adaptivity from refining on "noise"). If the magnitude of the pressures or the tractions is approximately known in the problem, one can use input like this:

```
  ADAPTIVITY SPECIFICATION
    JUMP REFINE THRESHOLD 0.75
    JUMP UNREFINE THRESHOLD 0.50
    JUMP NORMALIZER 1.E8
  END
```

which will divide all element traction values by 1.E8 before applying the thresholds.

3.8.1.3 ELEMENT BUDGET

The element budget method is not an error indicator but instead limits the number of elements that can exist during the simulation. If the element budget is reached, the elements with the higher error values are refined before the elements with a lower error value. At this point, the element budget control can be used in conjunction with the traction jump error indicator..

Table 30: ADAPTIVITY Keywords for ELEMENT BUDGET

Sub-Keyword	Numeric Input	Meaning
MAXIMUM ELEMENTS	*int*	The maximum number of elements to ever be used in the simulation
MAXIMUM LOCAL ELEMENTS	*int*	The maximum number of elements to be used on any one processor in a parallel simulation
NUMBER OF ERROR BINS	*int*	Level of granularity of the error value sorting algorithm (higher values modestly increase the amount of parallel communication)
APPLY TO JUMP ERROR		Flag keyword to indicate that the refinement prioritization will be based on the traction jump error
JUMP WEIGHT FACTOR	*real*	(Not applicable with only one error indicator)
JUMP ERROR VALUE FLOOR	*real*	A cutoff below which elements are never refined

An example input is:

```
ADAPTIVITY SPECIFICATION
  MAXIMUM ELEMENTS 10000
  APPLY TO JUMP ERROR
  JUMP ERROR VALUE FLOOR 1.E2
END
```

which indicates that there should not be more than 10,000 elements, apply it to the jump error indicator for prioritization, and even if there are less than 10,000 elements, do not refine elements if the jump error value is less than 100.

3.8.1.4 Adaptivity Layering

LAYERING

HYDRO JUMP WIDTH int

END

Layering spreads the adaptation of elements out over a user selected width of elements, leading to a more gradual onset of higher resolution. The number input after the keyword HYDRO JUMP WIDTH is the number of elements over which to spread out the refinement.

3.8.1.5 Block Specific Control of Adaptivity

BLOCK ADAPT LEVELS

int int int int ..... int int

END

The BLOCK ADAPT LEVELS keyword is followed by one or more ***pairs*** of integers. In each pair of integers, the first number refers to a mesh block while the second refers to the maximum adaptivity level that will be allowed in the block. This is a method of overriding the maximum levels of adaptivity that is set for the whole mesh in the DOMAIN keyword section.

3.8.1.6 Control of Unrefinement

UNREFINEMENT CONTROL

CYCLE LAG int

TIME LAG real

LOCK INITIAL REFINEMENT

UNTIL CYCLE int

UNTIL TIME real

NEVER UNREFINE

END

The adaptivity options in the UNREFINEMENT CONTROL input allow the user to control when unrefinement takes place. The first two controls are CYCLE LAG and TIME LAG. These two values control a delay between the indication of unrefinement by the error metric and the actual unrefining of the element. CYCLE LAG introduces a specific number of computational cycles between the unrefine indication and the actual unrefinement of an element, while TIME LAG introduces a delay of a set amount of problem time between the indication and the action. Both features are present to reduce noise in the unrefinement of elements.

The LOCK INITIAL REFINEMENT flag is tied to the next three options: UNTIL CYCLE, UNTIL TIME and NEVER UNREFINE. These options allow the user to control the unrefinement of the refined mesh introduced during an initial refinement step. (Initial refinement is controlled by the DOMAIN section of the ALEGRA input (see **Section 3.4.6.2.3 on page 102**). The LOCK INITIAL RFINEMENT input forces the initially refined mesh to remain at its level of refinement until a certain computational cycle is reached (UNTIL CYCLE) or until a given problem time is reached (UNTIL TIME). The NEVER UNREFINE keyword forces the initial refinement to persist indefinitely.

Table 31: ADAPTIVITY Keywords for UNREFINEMENT CONTROL

Sub-Keyword	Numeric Input	Meaning
CYCLE LAG	*int*	number of cycles between an unrefinement indication from the error metric and the actual element unrefinement
TIME LAG	*real*	amount of problem time between an unrefinement indication from the error metric and the actual element unrefinement
LOCK INITIAL REFINEMENT		Flag keyword
UNTIL CYCLE	*int*	computational cycle at which an initially refined mesh can unrefine
UNTIL TIME	*real*	computational time at which an initially refined mesh can unrefine
NEVER UNREFINE		Flag keyword

3.8.1.7 Dynamic Load Balancing

To keep the load balanced over many processors during adaptivity, the ZOLTAN load balancer can be invoked. This will produce a mesh restructuring. In the current release, load balancing is either on or off and ZOLTAN is called for each time step although it may decide that no mesh movement is required. Load balancing is turned on with the following syntax:

```
HYDRODYNAMICS
```

```
  ...

    LOAD BALANCE
    END
  END
```

That is, it resides in the physics specification block.

3.9 Materials and Material Models

Specification of the materials and material models used for an ALEGRA calculation is expressed in rather general terms. A material, for purposes of an ALEGRA calculation, is defined by a set of material models and initial values of critical independent variables that may differ from the default reference values. Thus, ALEGRA input for material specification consists of a *MATERIAL* keyword group and corresponding sets of material ***MODEL subkeyword*** groups. The remainder of this section provides an overview of the material and material model input via examples. More detailed descriptions and syntax follow in the subsequent subsections. The final subsection contains complete examples of material and model input for all the material models.

As a simple example, the material and material model input that might be used to describe air for hydrodynamics calculations is shown below. For this simple material, only one material model is needed to describe the material response of interest. The initial density in this example was desired to be lower than the default, which would be equal to rhoref.

```
  MATERIAL 200 air
    model 201
    density     = 0.001 $ g/cm^3, lower init density from rhoref
  end

  MODEL 201 ideal gas
    rho ref     = 0.0011756624 $ g/cm^3
    temperature = 298.         $ K
    gamma       = 1.4
    cv          = 0.7165e+07   $ erg/g/K
  end
```

Generally, the material and material model identification numbers are arbitrary and for the convenience of the user. However, the model number specified in the *MATERIAL* keyword group must correspond to the model id of the material *MODEL* keyword group.

The string following the *MATERIAL* keyword and id is optional and is for the convenience of the user. All text up to a comment or end-of-line will be read as the optional string.

The optional string must not match any main or reserved keyword and must not be an integer or real value. It will be parsed as such and an error will result. To avoid this, the string may be placed within double quotes (see the string following `material 200` in the example below - `6061` would be parsed as an integer if not quoted).

The comments denoting units which follow a number of the parameter lines are for the convenience of the user. However, a units parser may become available in the future. At that time, the unit strings would no longer need to be commented.

The initial values of independent variables (e.g., temperature and density) do not need to be specified unless the initial state of the material and the reference state of the model are different.

A more complex example of material and material model input for a solid dynamics calculation is shown below. The units in square brackets placed after the density are simply a demonstration of the code's capability of handling unit conversions of the input (see ***Section 3.2.5 on page 59***).

```
material 100 tungsten penetrator
  model 101         $ mie-gruniesen us-up
  model 102         $ elastic-plastic
  model 103         $ pressure-dependent fracture
  density 1850. [kg/m^3] $demonstration of unit, not req'd
end

model 101 mg us up
  c0                = 3.998e5  $ cm/s
  sl                = 1.237
  gamma0            = 1.54
  rho ref           = 18.5     $ g/cm^3
  cv                = 2.5e10   $ erg/g/K
  pref              = 0.0
  tref              = 298.0
  e shift           = 1.0e+10
end

model 102 elastic plastic
  youngs modulus    3.42e12    $ dyne/cm^2
  poissons ratio    0.29
  yield stress      1.5e+10    $ dyne/cm^2
  hardening modulus 0.0        $ dyne/cm^2
  beta              1.0
end

model 103 frac presdep
  init frac pres = -5.e10
  density tolerance = 1.e-6
  pressure tolerance = 1.e+2
end
```

```
material 200 “6061-t6 aluminum target plate”
  model 201 $ elastic-plastic
  model 202 $ keos mie-gruniesen
  model 203 $ pressure-dependent fracture
  density = 2.66     $ not needed if same as r0, g/cm^3
  temperature = 298. $ not needed if same as t0, K
end

model 201 elastic plastic
  youngs modulus       = 73.00e+10   $ dyne/cm^2
  poissons ratio       = 0.3225
  yield stress         = 2.76e+09     $ dyne/cm^2
  hardening modulus    = 1.24e+09     $ dyne/cm^2
  beta                 = 1.
end

model 202 keos miegrun  $ Newer version of mg us up
  cs         = 5.328e5  $ cm/s
  sl         = 1.338
  g0         = 2.18
  r0         = 2.66     $ g/cm^3
  cv         = 1.034e07 $ erg/g/K
  t0         = 298.0    $ K
end

model 203 frac presdep
  init frac pres     = -1.e9   $dyne/cm^2
  density tolerance  = 1.e-6
  pressure tolerance = 1.e+2
end
```

Note that each material is described by a set of material models. Generally, any set of material models may be specified for a material as appropriate for the physics performed in the ALEGRA calculation (see ***Section 3.4.1 on page 74***). This generality provides substantial flexibility for the user to describe very complex material behavior. However, it also is relatively easy to specify inconsistent material models which produce invalid results. Limited consistency checking exists, but it should become more robust with future versions of the code. Refer to the subsequent sections on the *MATERIAL* and *MODEL* keywords for more information on constructing consistent input.

All materials described by a *MATERIAL* keyword block may occupy any element in the mesh. Thus, all elements provide for the possibility of all materials being present. This capability is, in general terms, achieved by providing an array of *MATERIAL DATA* objects, one for each material defined in the user input file. These *MATERIAL DATA* objects contain the variables for each material in that particular element. The material models defined for the material operate on this data. Thus, knowledge of the variables operated on by the models comprising a particular material can provide useful insight into the nature and type of response of the models. These variables are listed in the tables of registered plot variables for each material model discussed in the following sections (see **Section 3.9.2 on page 148**).

In general, the element maintains an array of material volume fractions of all materials present within that element. Thus, the sum of all of the material volume fractions for an element must equal one. Furthermore, individual values of material volume fraction range from zero to one. In some cases an element has material volume fractions which sum to a value less than one. In such a case, a portion of the element is considered *empty.* A fictitious material called *Void* occupies the remaining empty volume of the element. Void is a material with no material models defined. Thus, it generally has no properties and variables. Void is essentially vacuum.

The users should be aware that there are a number of ways in which a material is deposited in an element (see **Section 3.4.6.1.1 on page 87**). At initialization, material may be assigned to all elements of an element block through the *MATERIAL* keyword in the ***BLOCK*** input. Note that multiple material keywords in the ***BLOCK*** input produce equal material fractions for each material. Alternatively, the ***DIATOMS*** input also allows material to be inserted into elements meeting a spatial criteria. During a calculation, material may enter or exit an element due to advection. In all of these cases, multi-material elements may occur, and may also have void present.

3.9.1 MATERIAL

MATERIAL int [string]

MODEL int

[MODEL int]

...

[variable_name real]

[variable_name real]

...

END

The *MATERIAL* keyword begins a block of user of input specifying the models and initial state of a material to be used in the calculation. The *MATERIAL* keyword is followed by an identifying integer which refers to the particular material throughout the input specification file. For example, the material which fills an element block must refer to a valid material id.

```
BLOCK 1
  material 201 $ see BLOCK input for this item
```

```
END

MATERIAL 201 “6061-T6 Aluminum”
  ...
END
```

In this example, the *BLOCK* input specification that material 201 fills the element block corresponds to the *MATERIAL* keyword with id 201.

The identifying material number is followed by an optional string which may be used to describe the material. This string is parsed to the end of line or next valid keyword. Supplying the optional descriptive string improves readability of the input file. The string is printed to the output file of the calculation, also improving its readability, as well as used in various error messages. In the above example, the descriptive string is *6061-T6 Aluminum.*

The string may or may not be enclosed in quotes. Recognition of the string as a material description assumes the string does not match any valid material keyword. For example, the string *Be* (short for beryllium) matches the material keyword *BETA* and an input error occurs. The ambiguity is removed by expanding *Be* to *Beryllium* or by surrounding *“Be”* with quotes.

3.9.1.1 MODEL subkeyword

MODEL int

The *MODEL* subkeyword is used to construct a list of material models which will be used to describe the response of the material. Any number of models, as appropriate for the material and physics of the problem, may be specified. It is important to note that **THE ORDER OF THE MODELS SPECIFIED FOR A MATERIAL IS THE ORDER IN WHICH THE MODELS WILL BE APPLIED** in the calculation. For example, the input fragment:

```
MATERIAL 10 Administratium
  MODEL 11
  MODEL 25
  MODEL 8
END
```

applies models 11, 25, and 8, in order, when computing the state of material 10. This ordering has implications for properly computing the state of the material. The variables of some models may depend on variables computed in other models. This implies that models with such dependent variables be specified AFTER the models which compute those variables.

For example, if MODEL 25 requires temperature as a known input value, MODEL 11 must calculate temperature. Otherwise, the temperature used in MODEL 25 will be the temperature from the preceding cycle (to make matters more complicated, in some cases this may be exactly what the model requires!).

To address the difficulty in specifying the correct order of material models in a material, all material model implementations specify all variables, identifying each as either input, input/output, or output. Thus, to properly use each model, all input variables of the model are provided to the material (via element data) or computed by a preceding material model. The tables in the material model subsections list all the variables and the corresponding input/output type.

A dependency checking algorithm is planned for implementation to validate the material models specified in the input. Currently, this is the responsibility of the user.

3.9.1.2 variable_name subkeyword

variable_name real

The variable_name input specification is used to define a non-zero initial value for an independent variable of the material. The most common (and currently the only use of this facility) is to specify initial density and temperature of a material for equation of state material models. Currently, it is possible to initialize any scalar variable which can be written to the plotting database.

This facility is provided as a general capability for models which require non-zero initial values for certain scalar variables. The generality should make this capability usable for the most complicated material models. Using this facility is highly dependent on the particular material model and generally requires detailed knowledge of the implementation of the model. Thus, it is recommended that this capability not be used indiscriminately.

3.9.2 MODEL

MODEL int model_name

  parameter [{int | real}]

  parameter [{int | real}]

  ...

END

The *MODEL* keyword begins a block of input which specifies a particular material model and provides for input of the appropriate material model parameters. The *MODEL* keyword is followed by an integer identifier selected by the user, followed by the name of the material model to be used. The integer id is used to cross-reference the material model with the material model list contained in a *MATERIAL* keyword group.

The body of the material *MODEL* keyword group consists of *parameter* value input. Parameters are set by the name of the parameter followed by the integer or real value, as appropriate. The parameter names and input value types are listed in the material model catalog in the subsequent subsections. The value following a parameter keyword is parsed as either integer or real. Thus, an integer cannot be coerced to a real value (e.g., 298 is an integer and 298. is a real). Instead, a parsing error with an appropriate diagnostic will occur.

Examples:

```
MATERIAL 1 elastic copper
  MODEL 1
  MODEL 2
  temper  = 500. $ K, can change initial temp from ref value
END

MATERIAL 2 elastic plastic copper
  MODEL 1
  MODEL 3
END

MODEL 1 mg us up
  ...
END
MODEL 2 linear elastic
  ...
END
MODEL 3 elastic plastic
  ...
END
```

The model name must be a valid name for an available model. Note that a particular model may be used in more than one material specification. Furthermore, models may also be included in the user input that are not used by any material. Such models are not used (although the input is parsed). This may be convenient when running problems from the same input deck but desiring to vary material models used by certain materials.

In ALEGRA, a material model is a module which computes the values of a set of dependent material variables given a set of independent material variables and other appropriate data. No distinction is made among models as being of a specific type such as an equation of state model or a plasticity model. Instead all models are considered *equal* so artificial constraints regarding what types of variables a model can compute are avoided.

For complex materials, a number of models may need to be sequenced in for a material so that all variables of interest are computed. In other cases, a very complex material model may exist which computes all variables of interest in a single model.

A list of the currently available models is provided in **Table 32**. The models are approximately categorized into general model types although in some cases, a model may contain more or less capability than is typically assumed for such models.

Table 32: Material Model Types and Model Names

General Material Model Type	Model Name
Equation of State	***GENERIC EOS*** ***IDEAL GAS*** ***JWL*** ***KEOS Arb*** ***KEOS Ffrb*** ***KEOS Hvrb*** ***KEOS Igrb*** ***KEOS Ptran*** ***KEOS Ideal Gas*** ***KEOS JWL*** ***KEOS MieGruneisen*** ***KEOS Sesame*** ***MG POWER*** ***MG US UP***
Constitutive	***ELASTIC PLASTIC*** ***LINEAR ELASTIC*** ***SOIL CRUSHABLE FOAM***
Yield	***STEINBERG GUINAN LUND*** ***JOHNSON COOK EP*** ***ZERILLI ARMSTRONG*** ***BAMMANN CHIESA JOHNSON*** ***VON MISES YIELD***
Plasticity	***SIMPLE RADIAL RETURN*** ***EP RADIAL RETURN***
Combined Models	***CTH ELASTIC PLASTIC*** ***BFK CONCRETE***
Fracture	***FRAC PRESDEP***
Burn	***PROGRAMMED BURN JWL***

The following subsections provide individual descriptions of the material models available in ALEGRA. The models are described in the order in which they appear in **Table 32**. Each model will have two tables, one listing model input parameters and another listing the registered model variables.

In the model input parameter tables, the full parameter name for parsing is listed, followed by the parameter type, either *real* for floating point number or *int* for integer. A description of the parameter is also provided. **Note that the initial value of any unspecified parameter defaults to zero, unless indicated otherwise in the parameter description**. In the registered model variable tables, the variable name is followed by the variable type (*int, real, vector, symtensor, tensor, or antitensor*), variable mode (*INPUT, IOPUT, or OUTPUT)*, and a description. In general, all variables in the model variable tables may be plotted by listing the variable name (without underscores) within the ***PLOT VARIABLES*** keyword construct. Extra variables for each model that may be plotted by the HISPLT program are listed in an additional table where applicable. HISPLT variables that are not model-dependent for the various ALEGRA physics options are listed in **Section 4.3 on page 245.**

The variable mode *INPUT,* indicates that a particular variable is used as input to the model, implying that the value is current when the model is applied. The *IOPUT* mode indicates that a variable contains a current value upon entry to the model and is updated in the course of the model. The *OUTPUT* mode indicates that the variable is computed by the model. Generally, a consistent set and sequence of material models for a material will have all *INPUT* variables provided by the element or computed as *IOPUT* or *OUTPUT* variables in preceding models.

The directory and files containing source code for the model are also listed, relative to the $ALEGRA_DEVEL path. Many of the models are specialized and are only applicable to particular versions of the code or particular physics selections. Where possible, these requirements are noted, although this information may be incomplete.

3.9.3 Equation of State Models

Equation of state models relate the internal energy to other state variables (such as density and pressure). The equation of state models in ALEGRA range from simple models that exist only to provide required thermodynamic quantities (such as the *GENERIC EOS*) to complex models that extend the basic concepts of equilibrium thermodynamics to include the effects of micro-structure and time-dependent behavior (such as the reactive burn models).

To simplify input for several of the equation of state models, a specially formatted ASCII file named *EOS_data* is distributed with ALEGRA that provides input parameters for many predefined materials. Incorporation of any particular data set into the *EOS_data* file does not imply correctness or validation of the data set, but it is intended as a starting point for materials of interest in the absence of experimental data for the particular application. Parameters in the *EOS_data* file are given in cgseV units, which are standard CTH units (the *EOS_data* file is also distributed with CTH). The data are read and automatically converted to problem units during ALEGRA's set up phase.

Models that are shared with CTH use the MIG (Material Interface Guidelines) [7]. These models have an associated data file that lists the required Fortran routines and other information specified in the guidelines. If this file is available with the ALEGRA source code, it is listed with the description of the material model. The actual source code that is shared with CTH is in an external library that is distributed with the ALEGRA code. The equation of state models that are shared by CTH and ALEGRA are named with the prefix "KEOS." Several of the KEOS equation of state models include an optional parameter called *ESHIFT*. This parameter is used to temporarily shift the zero energy condition or reference energy, *Eref*, up to a range where a solution algorithm is not likely to return negative energies and temperatures, which are problematic for some codes. The KEOS models calculate a reasonable value of *ESHIFT* internally as an aid to the user, although it can be overridden by user input.

3.9.3.1 GENERIC EOS

The generic equation of state is used for materials which have a mechanical stress tensor model but would otherwise have no computation of temperature or sound speed. Note that sound speed is required of all materials in the *DYNAMICS* hierarchy (***HYDRODYNAMICS***, ***SOLID DYNAMICS***). Thus, in the absence of any other model that computes sound speed for the material, this model should be specified. The specific heat, C_v , is taken as a constant. A reference sound speed, C_{So}, must be specified. This may be derived from elastic moduli specified in other models describing the material. Pertinent equations are shown below.

$$T = T_{ref} + \frac{(E - E_{ref})}{C_v}$$

$$C_{So} = \sqrt{\frac{K_o + \frac{4}{3}G_o}{\rho_o}}$$

$$C_S = \sqrt{\frac{\rho_o C_{So}^2}{\rho}}$$

- Modules: material_libs/standard_models
  - matmod_generic_eos.h
  - matmod_generic_eos.C
- Physics: solid dynamics

- References: none

Table 33: Input Parameters for GENERIC EOS

Parameter Name	Type	Description
RHO REF	*real*	Density at the reference state, ρ_0
TREF	*real*	Absolute temperature at the reference state, T_{ref}
CV	*real*	Specific heat at constant volume, C_v
EREF	*real*	Specific internal energy at the reference state, E_{ref}
REF SOUND SPEED	*real*	Sound speed at the reference state, C_{So}
E SHIFT	*real*	Arbitrary shift of reference energy (optional)

Table 34: Registered Plot Variables for GENERIC EOS

Variable Name	Type	Mode	Description
DENSITY	*real*	input	Material density, ρ
ENERGY	*real*	input	Specific internal energy per unit mass, E
TEMPERATURE	*real*	output	Absolute temperature, T
SOUND_SPEED	*real*	output	Sound speed, C_s

```
$Sample input the generic eos model
$A material with this eos would probably also specify some
$strength model, such as elastic-plastic.

  model 32 generic eos in cgsK units
    rho ref    = 8.932
    tref       = 298.
    cv         = 3.924e+06
    ref sound speed = 4.4468e+05
  end
```

3.9.3.2 IDEAL GAS

The ideal gas model [22] computes the pressure, P, the temperature, T, and the sound speed, C_s, for the material using the standard ideal gas equations.

$$P = \rho(\gamma - 1)(E - E_{ref})$$

$$T = \frac{(E - E_{ref})}{C_v(1 + \bar{Z})}$$

$$C_s = \sqrt{\frac{\gamma P}{\rho}}$$

For plasmas, there can be electron contributions to the gas energy and pressure. To account for any electron effects, the *FIXED ZBAR* parameter is provided. Set *FIXED ZBAR* = 1. for single ionization, *FIXED ZBAR* = 2. for double ionization, etc. This parameter defaults to zero.

- Modules: material_libs/standard_models
  - matmod_idlgas.h, matmod_idlgas.C
- Physics: hydrodynamics

Table 35: Input Parameters for IDEAL GAS

Parameter Name	Type	Description
RHO REF	*real*	Density at the reference state, ρ_0
TREF	*real*	Absolute temperature at the reference state, T_{ref}
CV	*real*	Specific heat at constant volume, C_v
GAMMA	*real*	Ratio of specific heats, $\gamma = C_p/C_v$
FIXED ZBAR	*real*	Fixed ionization state (to account for electrons)
E Shift	*Real*	Arbitrary Shift Of Reference Energy (optional)

Table 36: Registered Plot Variables for IDEAL GAS

Variable Name	Type	Mode	Description
SPECIFIC_HEAT_VOL	*real*	initialize	Initialized from C_v parameter
DENSITY	*real*	input	Material density
ENERGY	*real*	input	Specific internal energy per unit mass
PRESSURE	*real*	output	Pressure
TEMPERATURE	*real*	output	Absolute temperature

Table 36: Registered Plot Variables for IDEAL GAS (Continued)

Variable Name	Type	Mode	Description
SOUND_SPEED	*real*	output	Sound speed

```
$Sample input for ideal gas model in cgsK units.

  model 11 ideal gas
    gamma = 1.667
    rho ref = 1.624e-4  $g/cm^3
    tref = 300.         $K
    cv = 3.116324e+07   $erg/g/K
  end
```

3.9.3.3 JWL

This is the Jones-Wilkins-Lee equation of state for high explosives. The equations are described fully by Kerley [22]. This model describes the equation of state of high explosive materials in a fully detonated state. The parameters are unique for a given undetonated density and temperature. This particular implementation of JWL expects units that are consistent within the model, which is not usually the way JWL parameters are published[13]. Input parameters must also be consistent with the problem units.

- Modules: material_libs/standard_models
  - matmod_jwl.h
  - matmod_jwl.C
- Physics: hydrodynamics.

Table 37: Input Parameters for JWL

Parameter Name	Type	Description
RHO REF	*real*	Density in unreacted reference state
TREF	*real*	Temperature in unreacted reference state
E SHIFT	*real*	Arbitrary shift of reference energy (optional)
A	*real*	JWL parameter in units of problem
B	*real*	JWL parameter in units of problem
C	*real*	JWL parameter in units of problem
OMEGA	*real*	JWL parameter in units of problem
R1	*real*	JWL parameter in units of problem

Table 37: Input Parameters for JWL (Continued)

Parameter Name	Type	Description
R2	*real*	JWL parameter in units of problem
E0	*real*	JWL parameter in units of problem
PCJ	*real*	Chapman-Jouget pressure
DCJ	*real*	Chapman-Jouget detonation front velocity
TCJ	*real*	Chapman-Jouget temperature

Table 38: Registered Plot Variables for JWL

Variable Name	Type	Model	Description
SPECIFIC_HEAT_VOL	*real*	initialize	Initialized from C_v parameter
DENSITY	*real*	input	Material density
ENERGY	*real*	input	Specific internal energy per unit mass
PRESSURE	*real*	output	Pressure
TEMPERATURE	*real*	output	Absolute temperature
SOUND_SPEED	*real*	output	Bulk sound speed
DPDRHO	*real*	output	Derivative of pressure with respect to density

```
$Sample input for JWL model, cgsK units

  model 21 jwl
    a       = 4.63e+12  $ a; dyne/cm^2
    b       = 8.873e+10 $ b; dyne/cm^2
    c       = 1.349e+10 $ c; dyne/cm^2
    omega   = 0.35
    r1      = 4.55
    r2      = 1.35
    rho ref = 1.65      $ rho0; g/cm^3
    e shift = 1.e+10
    pcj     = 2.15E+11  $ pcj; dyne/cm^2
    dcj     = 7.03e+05  $ dcj; cm/s
    tcj     = 4062.0    $ tcj; K
    tref    = 298.0     $ K
  end
```

3.9.3.4 KEOS Reactive Burn Models

Five Kerley reactive burn models (***KEOS Arb***, ***KEOS Ffrb***, ***KEOS Hvrb***, ***KEOS Igrb***, and ***KEOS Ptran***) [23],[24] are available in ALEGRA. For each of these KEOS models, the material decomposition is included in the equation of state, and the time evolution of the reaction is described by a rate equation. Detailed descriptions of these models and tips for using the reactive burn models are provided in these references. The basic equations will be presented here as a summary, followed by the specific rate equation and input parameters for each burn model.

The reactive burn models require the equation of state for the unreacted material, the reaction products, and the partially reacted explosive. They are called "composite" models because they are constructed from the basic Kerley EOS models: ***KEOS Sesame***, ***KEOS MieGruneisen***, ***KEOS JWL***, and ***KEOS Ideal Gas*** for the unreacted materials and reaction products. The partially reacted explosive is described by the following equations.

$$P(\rho, T, \lambda) = (1 - \lambda)P_{UR}(\rho, T) + \lambda P_{RP}(\rho, T)$$

$$E(\rho, T, \lambda) = (1 - \lambda)E_{UR}(\rho, T) + \lambda E_{RP}(\rho, T)$$

where λ is the extent of reaction (λ=0 for no reaction, λ=1 for complete reaction). The subscripts UR and RP denote the equation of state for the unreacted explosive and the reaction products, respectively. The equation for advancing λ in time is given for each of the five models in the following subsections.

- Modules: material_libs/kerley_eos
  - rbn_mig.h
  - rbn_mig.C
- Physics: hydrodynamics

The extra HISPLT variables for the KEOS reactive burn models are listed in **Table 39**. Other HISPLT variables are listed in **Section 4.3 on page 245.**

Table 39: Extra HISPLT Variables for KEOS Reactive Burn Models

Variable Name	Description
EXT_REACTION	The extent of reaction of the burn. The value ranges from 0 to 1.
ALPI	The porosity parameter if the initial state submodel is a porosity model. This parameter represents the ratio solid density/porous density.
ALPF	The porosity parameter if the final state submodel is a porosity model. This parameter represents the ratio solid density/porous density.

Table 40: Registered Plot Variables for KEOS Reactive Burn Models

Variable Name	Type	Model	Description
DENSITY	*real*	input	Material density
ENERGY	*real*	input	Specific internal energy per unit mass
PRESSURE	*real*	output	Pressure
TEMPERATURE	*real*	output	Absolute temperature
SOUND-SPEED	*real*	output	Bulk sound speed
DPDRHO	*real*	output	Derivative of pressure with respect to density
EXT_REACTION	*real*	output	Extent of reaction parameter, non dimensional.
ALPI	*real*	output	Porosity parameter, alpha, from submodel for initial, unreacted state.
ALPF	*real*	output	Porosity parameter, alpha, from submodel for final state for reaction products.

3.9.3.4.1 KEOS Arb

The *KEOS Arb* (Arrhenius Reactive Burn) model can be used for the initiation and propagation of detonations in homogenous explosives. The rate law is a function of the temperature.

$$\frac{d\lambda}{dt} = (1-\lambda)F\,exp\left(\frac{\Theta}{T}\right)$$

where F is the frequency factor, and Θ is the activation temperature.

$$\Theta = \Theta_0(1 + A_P P)$$

The parameters F, Θ_0, and A_P are obtained by fitting data from wedge tests and other experiments.

- Modules: material_libs/kerley_eos
  - arb_mig.h
  - arb_mig.C
  - arb.doc is the ASCII data file that lists associated Fortran routines.

Table 41: Input Parameters for KEOS Arb

Parameter Name	Type	Description
MATLABEL	*char**	Label listed in the EOS_data file for a pre-defined material for this ARB model. If this keyword is used, no other keyword is required. Enclose the label in single quotes.
DATAFILE	*char**	Name and location of the specially formatted EOS_data file if different than the default $ALEGRA_MIGDATA/EOS_data. Enclose the label in single quotes.
AT	*real*	Rate equation constant Θ_0, [0.] Required unless MATLABEL given.
FF	*real*	Rate equation constant F [1.E10s^{-1}] Required unless MATLABEL given.
ATP	*real*	Rate equation constant A_P Required unless MATLABEL given.
TI	*real*	Threshold temperature [initial temp]
RMIN[a]	*real*	Minimum density, unreacted explosive. Reaction is complete for densities < RMIN. [0.]
RMAX[a]	*real*	Maximum density, unreacted explosive. Reaction is complete for densities > RMAX. [1.e30]

Table 41: Input Parameters for KEOS Arb (Continued)

Parameter Name	Type	Description
TMAX[a]	*real*	Maximum temperature, unreacted explosive. Reaction is complete for temperatures > TMAX. [1.e30]
VF[a]	*real*	Volume ratio, initial state/final state [1.]
TF[a]	*real*	Temperature ratio, initial state/final state [1.]
ESFT[a]	*real*	Shift in energy zero (optional).
EOSUR MODNUMBER	*int*	The model number in the input deck that will be used for the unreacted material. [Required unless defined in EOS_data file or by EOSUR MATLABEL.]
EOSRP MODNUMBER	*int*	The model number in the input deck that will be used for the reacted products. [Required unless defined in EOS_data file or by EOSRP MATLABEL.]
EOSUR MODLABEL	*char**	Model label listed in the EOS_data file for a predefined material for the unreacted material. Enclose the label in single quotes. If this parameter in used, EOSUR MATLABEL is also required.
EOSRP MODLABEL	*char**	Model able listed in the EOS_data file for a predefined material for the reacted products. Enclose the label in single quotes. If this parameter in used, EOSRP MATLABEL is also required.
EOSUR MATLABEL	*char**	Label listed in the EOS_data file for a pre-defined material for the unreacted material. Enclose the label in single quotes. If this parameter in used, EOSUR MODLABEL is also required.
EOSRP MATLABEL	*char**	Label listed in the EOS_data file for a pre-defined material for the reacted products. Enclose the label in single quotes. If this parameter in used, EOSRP MODLABEL is also required.

Table 41: Input Parameters for KEOS Arb (Continued)

Parameter Name	Type	Description
EOSUR	*char**	This keyword is only read from the EOS_data file. It points to the model for the unreacted material in the EOS_data file.
EOSRP	*char**	This keyword is only read from the EOS_data file. It points to the model for the reacted products in the EOS_data file.

a.This parameter is normally not input by the user, being left to default value

```
$Two sample inputs for KEOS Arb model, cgsK units

$Sample 1: initial (UR) and final (RP) states are specified in
$EOS_data for PETN.

  model 201 keos arb
    matlabel = 'PETN'
  end

$If the user specifies initial (UR) and final (RP) states.

  model 22 keos arb
    eosur modnumber = 221    $UR is model 221 below.
   eosrp modnumber = 222    $RP is model 222 below.
  end

  model 221 keos miegrun
    matlabel = 'petn'
    rp = 1.75
  end

  model 222 keos jwl
    matlabel = 'petn'
    brn = 0.  $EOS_data file assumes jwl is programmed burn.
 end
```

3.9.3.4.2 KEOS Ffrb

The *KEOS Ffrb* (Forest Fire Reactive Burn) model can be used for the initiation and propagation of detonations in heterogeneous explosives. The rate law is pressure dependent.

$$\frac{d\lambda}{dt} = (1-\lambda)^f R(P)$$

where f is the order of the reaction $(0 \le f \le 1)$. The function *R(P)* is derived from the Pop plot, using the "single-curve buildup principle" and fit to an analytic expression. The required input parameters are the Pop plot variables and P_{max}.

- Modules: material_libs/kerley_eos
  - ffrb_mig.h
  - ffrb_mig.C
  - ffrb.doc is the ASCII data file that lists the associated Fortran routines.

Table 42: Input Parameters for KEOS Ffrb

Parameter Name	Type	Description
MATLABEL	*char**	Label listed in the EOS_data file for a predefined material for this FFRB model. If this keyword is used, no other keyword is required. Enclose the label in single quotes.
DATAFILE	*char**	Name and location of the specially formatted EOS_data file if different than the default $ALEGRA_MIGDATA/EOS_data. Enclose the label in quotes.
P1,X1	*real*	Pressure and run distance at 1st point on Pop plot. Required unless MATLABEL given.
P2,X2	*real*	Pressure and run distance at 2nd point on Pop plot. Required unless MATLABEL given.
PMAX	*real*	Pressure for instantaneous reaction. Required unless MATLABEL given.
PMIN	*real*	Threshold pressure for reaction [0.01*PMAX].
FR	*real*	Order of reaction f [0.]
NRH	*real*	Hugoniot option. NRH=0 for reactive Hugoniot, NRH=1 for nonreactive Hugoniot [0.]
RMIN[a]	*real*	Minimum density, unreacted explosive. Reaction is complete for densities < RMIN. [0.]
RMAX[a]	*real*	Maximum density, unreacted explosive. Reaction is complete for densities > RMAX. [1.e30]
TMAX[a]	*real*	Maximum temperature, unreacted explosive. Reaction is complete for temperatures > TMAX. [1.e30]

Table 42: Input Parameters for KEOS Ffrb (Continued)

Parameter Name	Type	Description
VF[a]	*real*	Volume ratio, initial state/final state [1.]
TF[a]	*real*	Temperature ratio, initial state/final state [1.]
ESFT[a]	*real*	Shift in energy zero (optional)
EOSUR MODNUMBER	*int*	The model number in the input deck that will be used for the unreacted material. [Required unless defined in EOS_data file or by EOSUR MATLABEL.]
EOSRP MODNUMBER	*int*	The model number in the input deck that will be used for the reacted products. [Required unless defined in EOS_data file or by EOSRP MATLABEL.]
EOSUR MODLABEL	*char**	Model label listed in the EOS_data file for a pre-defined material for the unreacted material. Enclose the label in single quotes. If this parameter in used, EOSUR MATLABEL is also required.
EOSRP MODLABEL	*char**	Model label listed in the EOS_data file for a pre-defined material for the unreacted material. Enclose the label in single quotes. If this parameter in used, EOSRP MATLABEL is also required.
EOSUR MATLABEL	*char**	Label listed in the EOS_data file for a predefined material for the unreacted material. Enclose the label in single quotes. If this parameter in used, EOSUR MODLABEL is also required.
EOSRP MATLABEL	*char**	Label listed in the EOS_data file for a predefined material for the reacted products. Enclose the label in single quotes. If this parameter in used, EOSRP MODLABEL is also required.
EOSUR	*char**	This keyword is only read from the EOS_data file. It points to the model for the unreacted material in the EOS_data file.
EOSRP	*char**	This keyword is only read from the EOS_data file. It points to the model for the reacted products in the EOS_data file.

a.This parameter is normally not input by the user, being left to default values.

```
$sample input for ffrb model, cgsK units.
$note that initial temperature and density are from the
$initial state given eosur model.
```

```
model 221 keos ffrb
  eosur modnumber =  222
  eosrp modnumber =  223
  X1 = 0.05           $cm
  P1 = 18.1567e10     $dyn/cm^2
  X2 = 0.50           $cm
  P2 = 4.4315e10      $dyn/cm^2
  FR = 1.0            $
  PMIN = 1.e9         $dyn/cm^2
  PMAX = 3.53e11     $dyn/cm^2
  NRH = 1.           $
 end

model 222, keos miegrun
  matlabel = 'pbx9404'
end

model 223, keos sesame
  matlabel = 'pbx9404_dp'
end
```

3.9.3.4.3 KEOS Hvrb

The *KEOS Hvrb* (History Variable Reactive Burn) model can be used for the initiation and propagation of detonations in heterogeneous explosives. The rate law is pressure dependent with a time delay before initiation of rapid reaction. The equation is written in integral form by introducing a history variable.

$$\lambda = 1 - \left(1 - \frac{\phi^M}{X}\right)^X$$

$$\phi(t) = \tau_0^{-1} \int_0^t \left[\frac{(P - P_I)}{P_R}\right]^z dt$$

The parameters P_R and z determine the pressure dependence of the time and distance to detonation (these are usually fit to wedge test data). The exponent M controls the time delay to pressure buildup behind the shock front, and X determines the rate at which the reaction goes to completion. P_I is the threshold pressure for initiation, and the coefficient τ_0 is used to make ϕ dimensionless and is not an independent constant.

- Modules: material_libs/kerley_eos
  - hvrb_mig.h

- hvrb_mig.C
- hvrb.doc is the ASCII data file that lists associated Fortran routines.

Table 43: Input Parameters for KEOS Hvrb

Parameter Name	Type	Description
MATLABEL	*char**	Label listed in the EOS_data file for a predefined material for this HVRB model. If this keyword is used, no other keyword is required. Enclose the label in single quotes.
DATAFILE	*char**	Name and location of the specially formatted EOS_data file if different than the default $ALEGRA_MIGDATA/EOS_data.
PR	*real*	Rate equation constant P_R. This parameter, along with ZR, determines the overall rate of the reaction. It is closely related to constants in Pop plot expressions. Required unless MATLABEL given.
ZR	*real*	Rate equation constant *z*. This parameter, along with PR, determines the overall rate of the reaction. It is closely related to constants in Pop plot expressions. Required unless MATLABEL given.
MR	*real*	Rate equation constant M. This parameter primarily affects the shape of the pressure wave behind the shock wave during buildup to detonation (M typically ranges from about 1 to 2.) Required unless MATLABEL given.
PI	*real*	Threshold pressure for shock initiation, P_I. Required unless MATLABEL given.
XR	*real*	Rate equation constant. Required unless MATLABEL given.
RMIN[a]	*real*	Minimum density, unreacted explosive. Reaction is complete for densities < RMIN. [0.]
RMAX[a]	*real*	Maximum density, unreacted explosive. Reaction is complete for densities > RMAX. [1.e30]
TMAX[a]	*real*	Maximum temperature, unreacted explosive. Reaction is complete for temperatures > TMAX. [1.e30]
VF[a]	*real*	Volume ratio, initial state/final state [1.]

Table 43: Input Parameters for KEOS Hvrb (Continued)

Parameter Name	Type	Description
TF[a]	*real*	Temperature ratio, initial state/final state [1.]
ESFT[a]	*real*	Shift in energy zero (optional).
EOSUR MODNUMBER	*int*	The model number in the input deck that will be used for the unreacted material. [Required unless defined in EOS_data file or by EOSUR MATLABEL.]
EOSRP MODNUMBER	*int*	The model number in the input deck that will be used for the reacted products. [Required unless defined in EOS_data file or by EOSRP MATLABEL.]
EOSUR MODLABEL	*char**	Model label listed in the EOS_data file for a predefined material for the unreacted material. Enclose the label in single quotes. If this parameter in used, EOSUR MAT-LABEL is also required.
EOSRP MODLABEL	*char**	Model label listed in the EOS_data file for a predefined material for the unreacted material. Enclose the label in single quotes. If this parameter in used, EOSRP MAT-LABEL is also required.
EOSUR MATLABEL	*char**	Label listed in the EOS_data file for a predefined material for the unreacted material.
EOSRP MATLABEL	*char**	Label listed in the EOS_data file for a predefined material for the reacted products.
EOSUR	*char**	This keyword is only read from the EOS_data file. It points to the model for the unreacted material in the EOS_data file.
EOSRP	*char**	This keyword is only read from the EOS_data file. It points to the model for the reacted products in the EOS_data file.

a.This parameter is normally not input by the user, being left to default values.

```
$Simple sample input for keos hvrb.  Alternative, the user can
$specify models for the unreacted initial state and the final
$reacted products as shown in keos arb or keos ffrb.
$The user can also modify parameters given in the EOS_data
$file as shown below.

  model 161 keos hvrb
    matlabel = 'HMX'
    rmin = 0.2
  end
```

3.9.3.4.4 KEOS Igrb

The *KEOS Igrb* (Ignition and Growth Reactive Burn) model describes the initiation and propagation of detonations in heterogeneous explosives. It is a three-step model that describes the initiation of reaction in hot spots, followed by propagation of reaction to the rest of the explosive, and a rapid completion phase that occurs during coalescence of the hot spot regions. The rate law is a function of pressure and density in the following equation.

$$\lambda = (1-\lambda)^{s_0} G_0 \frac{\rho}{(\rho_0 - 1 - a_0)^{y_0}} + (1-\lambda)^{s_1}\lambda^{q_1} G_1 P + (1-\lambda)^{s_2}\lambda^{q_2} G_2 P^{y_2}$$

The 12 constants in the above equation are chosen to fit experimental data for a given explosive. Three other constants are also used in the model. W0 is the value of λ at which the first term (ignition) is turned *off*, W1 is the value of λ at which the second term (growth) is turned *off,* and W2 is the value of λ at which the third term (completion) is turned *on.*

- Modules: material_libs/kerley_eos
  - igrb_mig.h
  - igrb_mig.C
  - igrb.doc is the ASCII data file that lists associated Fortran routines.

Table 44: Input Parameters for KEOS Igrb

Parameter Name	Type	Description
MATLABEL	*char**	Label listed in the EOS_data file for a predefined material for this HVRB model. If this keyword is used, no other keyword is required. Enclose the label in single quotes.
DATAFILE	*char**	Name and location of the specially formatted EOS_data file if different than the default $ALEGRA_MIG_DATA/EOS_data. Enclose the label in single quotes.
G0,GO	*real*	Ignition parameter G_0. Required unless MATLABEL given.
S0,SO	*real (0.667)*	Ignition parameter s_0

Table 44: Input Parameters for KEOS Igrb (Continued)

Parameter Name	Type	Description
A0,AO	*real (0.)*	Ignition parameter A_0
Y0,YO	*real (20.0)*	Ignition parameter y_0
W0	*real (0.3)*	Ignition turnoff parameter.
G1	*real*	Growth parameter G_1. Required unless MATLABEL given.
S1	*real (0.667)*	Growth parameter s_1.
Q1	*real (0.111)*	Growth parameter Q_1.
Y1	*real (1.0)*	Growth parameter y_1.
W1	*real (0.5)*	Growth turnoff parameter.
G2	*real*	Completion parameter G_2. Required unless MATLABEL given.
S2	*real (0.333)*	Completion parameter s_2.
Q2	*real (1.0)*	Completion parameter Q_2.
Y2	*real (2.0)*	Completion parameter y_2.
W2	*real (0.0)*	Completion turn-on parameter.
NSUB	*real (1)*	Number of subcycles in one time step
RMIN[a]	*real (0.)*	Minimum density, unreacted explosive. Reaction is complete for densities < RMIN.
RMAX[a]	*real (1.e30)*	Maximum density, unreacted explosive. Reaction is complete for densities > RMAX.
TMAX[a]	*real (1.e30)*	Maximum temperature, unreacted explosive. Reaction is complete for temperatures > TMAX.

Table 44: Input Parameters for KEOS Igrb (Continued)

Parameter Name	Type	Description
VF[a]	*real (1.0)*	Volume ratio, initial state/final state
TF[a]	*real (1.0)*	Temperature ratio, initial state/final state
ESFT[a]	*real*	Shift in energy zero (optional)
EOSUR MODNUMBER	*int*	The model number in the input deck that will be used for the unreacted material. [Required unless defined in EOS_data file or by EOSUR MATLABEL.]
EOSRP MODNUMBER	*int*	The model number in the input deck that will be used for the reacted products. [Required unless defined in EOS_data file or by EOSRP MATLABEL.]
EOSUR MODLABEL	*char**	Model label listed in the EOS_data file for a pre-defined material for the unreacted material. Enclose the label in single quotes. If this parameter in used, EOSUR MATLABEL is also required.
EOSRP MODLABEL	*char**	Model label listed in the EOS_data file for a pre-defined material for the unreacted material. Enclose the label in single quotes. If this parameter in used, EOSRP MATLABEL is also required.
EOSUR MATLABEL	*char**	Label listed in the EOS_data file for a predefined material for the unreacted material.
EOSRP MATLABEL	*char**	Label listed in the EOS_data file for a predefined material for the reacted products.
EOSUR	*char**	This keyword is only read from the EOS_data file. It points to the model for the unreacted material in the EOS_data file.
EOSRP	*char**	This keyword is only read from the EOS_data file. It points to the model for the reacted products in the EOS_data file.

a.This parameter is normally not input by the user, being left to default values.

```
$Sample input for keos igrb model. As with other reactive burn
$models, the user can specify individual models for the initial
$and final states, and model parameters specified in the EOS_data
$file can be changed.  The EOS_data file location and name can
$also be modified from the default value.
```

```
model 211 keos igrb
  datafile = '/home/svpetne/mynewdata/EOS_data.new'
  matlabel = 'PETN'
end
```

3.9.3.4.5 KEOS Ptran

The *KEOS Ptran* (Phase Transition Reactive Burn) model [24] simulates a material that has a transition between two phases. This model is similar to the two-state reactive burn models, but the transition is described by a phase boundary rather than a rate equation. The phase boundary is input by the user.

The user specifies the pressure in the transition region by the formula

$$P(\rho, T, \lambda) = P_T + \beta_T\left(1 - \frac{\rho_T}{\rho}\right) + A_T(T - T_0) + A_\lambda\lambda$$

where P_T and ρ_T are the transition pressure and density of phase 1 at room temperature T_0, β_T is the bulk modulus in the transition region, and A_T and A_λ are derivatives of the transition pressure with respect to T and λ, respectively. P_T, β_T , A_T and A_λ are input parameters. λ is the mass fraction of phase 2.

- Modules: material_libs/kerley_eos
  - ptran_mig.h
  - ptran_mig.C
  - ptran.doc is the ASCII data file that lists associated Fortran routines.
- Physics: hydrodynamics

Table 45: Input Parameters for KEOS Ptran

Parameter Name	Type	Description
MATLABEL	*char**	Label listed in the EOS_data file for a predefined material for this HVRB model. If this keyword is used, no other keyword is required. Enclose the label in single quotes.
DATAFILE	*char**	Name and location of the specially formatted EOS_data file if different than the default $ALEGRA_MIG_DATA/EOS_data. Enclose the label in single quotes.

Table 45: Input Parameters for KEOS Ptran (Continued)

Parameter Name	Type	Description
PT	*real*	Pressure at start of transition.
BT	*real*	Bulk modulus in transition region.
AT	*real*	Temperature derivative of transition pressure.
AX	*real*	Spatial derivative of transition pressure.
HF	*real (0.)*	Hysteresis flag. HF=0 for reversible case and HF=1 for irreversible case
RMIN[a]	*real (0.)*	Minimum density for initial state
RMAX[a]	*real (1.e30)*	Maximum density for initial state
TMAX[a]	*real (1.e30)*	Maximum temperature for initial state
VF[a]	*real (1.)*	Volume ratio, initial state/final state
TF[a]	*real (1.)*	Temperature ratio, initial state/final state
ESFT[a]	*real*	Shift in energy zero (optional)
EOSUR MODNUMBER	*int*	The model number in the input deck that will be used for the initial phase of the material. [Required unless defined in EOS_data file or by EOSUR MATLABEL and EOSUR_MODLABEL.]
EOSRP MODNUMBER	*int*	The model number in the input deck that will be used for the final phase of the material. [Required unless defined in EOS_data file or by *EOSRP MATLABEL* and *EOSRP MODLABEL*.]
EOSUR MODLABEL	*int*	The model type that will be used for the initial phase of the material. [If this parameter is input, *EOSUR MATLABEL* is also required.]
EOSRP MODLABEL	*int*	The model type that will be used for the final phase of the material. [If this parameter is input, *EOSUR MATLABEL* is also required.]
EOSUR MATLABEL	*int*	The model type that will be used for the initial phase of the material. [If this parameter is input, *EOSUR MODLABEL* is also required.]

Table of Contents
General Input

Table 45: Input Parameters for KEOS Ptran (Continued)

Parameter Name	Type	Description
EOSRP MATLABEL	*int*	The model type that will be used for the final phase of the material. [If this parameter is input, *EOSUR MODLABEL* is also required.]

a.This parameter is normally not input by the user, being left to the default value.

```
* Iron, PTRAN model, phase 1--alpha, phase 2--epsilon.
* Note esft for epsilon used to include transition energy.
* Set BT=1.85E12 to match transition pressure to Rayleigh line.
* SET HF=1 to make transition irreversible.

  model 231 keos ptran
    eosur modnumber = 232
    eosrp modnumber = 233
    pt=13.0E10
    bt=1.0E9
    at=-1.034082e7  $ -1.2E11/11604.5
    ax=-4.0E10
  end

  model 232 keos miegrun
    r0=7.87
    t0=298.
    cs=4.6E5
    s1=1.46
    g0=1.7
    cv=4.601663e6   $5.34E10/11604.5
  end

  model 233 keos miegrun
    r0=8.29
    t0=298.
    cs=4.6E5
    s1=1.51
    g0=2.4
    cv=4.515490e+6   $5.24E10/11604.5
  end
```

3.9.3.5 KEOS Ideal Gas

The Kerley EOS model for ideal gas [23] computes the equation of state for the material using the standard ideal gas equations.

$$P(\rho, E) = (\gamma - 1)\rho E / (1 - B_V \rho)$$

$$E(\rho, T) = C_v T\,.$$

where γ is the ratio of constant pressure and constant volume specific heats, C_v is the specific heat, and B_v is the co-volume, representing the volume excluded by molecules of finite size. B_v can be used to make a crude correction for nonideality.

The reference density and temperature (*R0, T0*) in this equation of state are not used for initialization in the current implementation of ALEGRA. Rather, the initial density and temperature in the material input section are used for the initial state. This will change in a future release so that if the material input does not specify the independent variables, the reference conditions in the model input will be used.

- Modules: material_libs/kerley_eos/idgas
  - ideal_gas_mig.h
  - ideal_gas_mig.C
  - idgas.doc is the ASCII data file that lists associated Fortran routines.
- Physics: hydrodynamics

Table 46: Input Parameters for KEOS Ideal Gas

Parameter Name	Type	Description
GM1	*real (2/3)*	γ–1. Default is monatomic gas value
CV	*real*	Specific heat
BV	*real (0.)*	Co-volume
R0 (or RO)	*real*	Initial density
T0 (or TO)	*real*	Initial temperature
E0 (or EO)	*real*	Energy at zero pressure and temperature
TYP	*real*	Model type (default value set by EOS package). Intended primarily for internal use in EOS package.

Table 47: Registered Plot Variables for KEOS Ideal Gas

Variable Name	Type	Model	Description
DENSITY	*real*	input	Material density
ENERGY	*real*	input	Specific internal energy per unit mass
PRESSURE	*real*	output	Pressure
TEMPERATURE	*real*	output	Absolute temperature
SOUND_SPEED	*real*	output	Bulk sound speed
DPDRHO	*real*	output	Derivative of pressure with respect to density

```
$Sample input for keos ideal gas model.

  model 141 keos ideal gas
    gm1     = 0.667
    r0      = 1.6245e-04    $ g/cm^3
    t0      = 298.          $ K
    cv      = 3.116324e+07  $ erg/g/K
  end
```

3.9.3.6 KEOS JWL

This is the Kerley EOS version of the Jones-Wilkins-Lee equation of state for high explosives [23]. This model is useful for computing the equation of state of high explosive materials in a fully detonated state. The parameters are unique for a given undetonated density and temperature. This particular implementation of JWL expects consistent units. Thus, input parameters must be in consistent units, which is not usually the way JWL parameters are published [13]. The setup for this model will fail unless R0, WG, and either CV or TCJ are given.

This model can also be used with the *PROGRAMMED BURN* option (Section 3.7.5.1 on page 135) in ALEGRA. Many of the JWL predefined materials in the *EOS_data* file are assumed to be operating with the *PROGRAMMED BURN* option. To turn off that option, set *BRN*=0.

- Modules: material_libs/kerley_eos/jwl
  - jwl_mig.h
  - jwl_mig.C
  - jwl.doc is the ASCII data file that lists associated Fortran routines.
- Physics: hydrodynamics

Table 48: Input Parameters for KEOS JWL

Parameter Name	Type	Description
MATLABEL	*char*	Material label in the EOS_data file If this parameter is provided, no other parameters are required. Enclose the string in single quotes.
DATAFILE	*char*	Name and location of the specially formatted EOS_data file if different from the default $ALEGRA_MIGDATA/EOS_data. Enclose the string in single quotes.
R0 (or RO)	*real*	Density in unreacted reference state (required)
T0 (or TO)	*real*	Temperature in unreacted reference state
AG	*real*	Constant A in JWL formula in units of problem
BG	*real*	Constant B in JWL formula in units of problem
R1	*real*	Constant R1 in JWL formula in units of problem
R2	*real*	Constant R2 in JWL formula in units of problem
WG	*real*	Gruneisen parameter in JWL formula (required)
E0 (or EO)	*real*	Detonation energy parameter E0 (required for reactive burn). May also be needed to compute specific heat. (If E0 >0, PCJ and DCJ will be recomputed. Otherwise, E0 is computed from DCJ and PCJ if CV=0).
CV	*real*	Specific heat
BRN	*real*	Set to 0 for no heburn with this model. If BRN=1, then the ***PROGRAMMED BURN*** input is required.
PCJ	*real*	Chapman-Jouget pressure. Used only if CV=0 and E0=0.
DCJ	*real*	Chapman-Jouget detonation front velocity. Used only if CV=0 and E0=0.
TCJ	*real*	Chapman-Jouget temperature. If TCJ=0, TCJ is reset to 0.35 eV or 4061.6 K (required if CV=0).
TDQ	*real (0.)*	Delayed heat release [0 for no time dependent reaction]

Table 48: Input Parameters for KEOS JWL (Continued)

Parameter Name	Type	Description
TDA	*real (0.)*	Rate prefactor for time dependent rxn [default 0 for no time dependent reaction].
TDM	*real (0.)*	Exponent for (1-LAMB) [0 for no time dependent reaction]
TDN	*real (0.)*	Exponent for P [0 for no time dependent reaction]

Table 49: Registered Plot Variables for KEOS JWL

Variable Name	Type	Model	Description
DENSITY	*real*	input	Material density
ENERGY	*real*	input	Specific internal energy per unit mass
PRESSURE	*real*	output	Pressure
TEMPERATURE	*real*	output	Absolute temperature
SOUND_SPEED	*real*	output	Bulk sound speed
DPDRHO	*real*	output	derivative of pressure with respect to density

```
$Two sample inputs for keos jwl model
$First sample uses user specified parameters, no programmed burn.

  model 151 keos jwl
    r0      = 1.65      $ rho0; g/cm^3
    t0      = 298.0     $ K
    ag      = 4.6310e+12  $ a; dyne/cm^2
    bg      = 8.8730e+10 $ b; dyne/cm^2
    r1      = 4.550
    r2      = 1.350
    wg      = 0.35
    cv      = 0.            $calculated internally 9.651e6
    tcj     = 4061.575    $ tcj; K
    e0      = 0.  $.0745e12
    esft    = 0.   $1.e10
    dcj     = 7.03e+05  $ dcj; cm/s
    pcj     = 2.15E+11  $ pcj; dyne/cm^2
    brn     = 0.        $no heburn
  end

$Second sample uses a simple EOS_data file, which assigns
$BRN=1 so that programmed burn is expected. A separate programmed
$burn input section is required to define detonation objects,
$time, and other programmed burn properties.

  model 150
    matlabel='HMX'
  end
```

3.9.3.7 KEOS MieGruneisen

The Kerley EOS version of the Mie-Grüneisen equation of state is the most recent version of the ***MG US UP*** model in ALEGRA [23]. It is based on the Mie-Grüneisen approximation, with the Hugoniot as the reference curve, together with the expression $\Gamma=\Gamma_0\rho_0/\rho$ for the Grüneisen function. The pressure P and energy E are given by

$$P(\rho, E) = P_H(\rho) + \Gamma_0\rho_0[E - E_H(\rho)],$$

$$E(\rho, T) = E_H(\rho) + C_v[T - T_H(\rho)],$$

where P_H, E_H, and T_H are the Hugoniot pressure, energy, and temperature. The Grüneisen parameter Γ_0 and specific heat C_v are taken to be constants. There are two options for representing the Hugoniot. The first is a quadratic relation between shock velocity U_s and particle velocity u_P,

$$U_S = C_S + S_1 u_P + \left(\frac{S_2}{C_S}\right)u_P^2 \quad ,$$

where C_s, S_1, and S_2 are constants. The second option uses a modified form for nonlinear behavior at low pressures.

$$U_S = 2C_S[1 - S_1\mu + \sqrt{(1-S_1\mu)^2 - 4S_2\mu^2}]^{-1} - B\,exp[-(\mu/\mu^*)^N] \quad .$$

This model reduces to the first option if B=0. Both of these options are described in detail by [23].

The *KEOS MieGruneisen* model allows incorporation of the p-alpha porosity model if the appropriate parameters are provided. The p-alpha model includes a distention parameter, α, which relates the macroscopic material density to the density ρ_M of the solid, void-free material.

$$\alpha = \rho_M/\rho\,.$$

The response includes a reversible elastic region and an irreversible compaction region. The maximum distention at a given value of pressure is given by

$$\alpha_{max}(P) = 1 + (\alpha - 1)\left(\frac{P_S - P}{P_S - P_E}\right)^2.$$

The parameter P_S is the maximum pressure for complete compaction. All voids are crushed out and α=1 for P greater than or equal to P_S. P_E is the upper pressure limit of the elastic crush region.

The distention parameter α is advanced in time using an integration scheme, dividing the global computational time step in to several subintervals. The number of subintervals can be adjusted by the user with the NSUB parameter.

The p-alpha model is intended for modeling materials with low porosity (less than 20 percent). It can be used as a submodel with the composite (two-state KEOS models), and is also an option in the ***KEOS Sesame*** equation of state. For materials with relatively higher porosities, the *KEOS Sesame* model is the recommended choice for the p-alpha option.

- Modules: material_libs/kerley_eos/mgrun
  - mgrun_mig.h
  - mgrun_mig.C
  - mgrun.doc is the ASCII data file that lists associated Fortran routines.
- Physics: hydrodynamics

Table 50: Input Parameters for KEOS MieGruneisen

Parameter Name	Type	Description
MATLABEL	*char*	Material label in the EOS_data file If this parameter is provided, no other parameters are required. Enclose the string in single quotes.
DATAFILE	*char*	Name and location of the specially formatted ASCII file which contains the properties for this model. The default is $ALEGRA_MIGDATA/EOS_data. Enclose the string in single quotes.
R0 (or RO)	*real*	Initial density in of Hugoniot (required).
CS	*real*	Sound speed in Hugoniot (required).
CV	*real*	Specific heat (required).
T0 (or TO)	*real (298.)*	Initial temperature.
S1	*real (0.)*	Linear coefficient in Hugoniot fit.
G0 (or GO)	*real (0.)*	Grüneisen parameter.
S2	*real (0.)*	Quadratic coefficient in Hugoniot fit.
B	*real (0.)*	Low-pressure Hugoniot parameter B (default=0 for no low pressure model)
XB	*real*	Low pressure Hugoniot parameter, required for the low pressure option.

Table 50: Input Parameters for KEOS MieGruneisen (Continued)

Parameter Name	Type	Description
NB	*real*	Low-pressure Hugoniot parameter N, required for the low pressure option.
RP	*real (=R0)*	For the p-alpha porosity model[a], initial density of porous material.
PS	*real (1.e9)*	For the p-alpha porosity model[a], pressure for complete compaction.
PE	*real (0.)*	For the p-alpha porosity model[a], maximum elastic pressure.
CE	*real*	For the p-alpha porosity model[a], sound speed in elastic compaction region (default for CE is sound speed of solid matrix).
NSUB	*int (10)*	For the p-alpha porosity model[a], number of subcycles within time step.
ESFT	*real*	Shift in energy zero (optional).
TYP	*real*	Model type (default value set by EOS package). Intended primarily for internal use in EOS package (optional).

a.Providing the parameter RP not equal to R0 designates the use of the p-alpha porosity model. Optional input for the porosity model are the parameters PS, PE, CE, and NSUB.

The extra HISPLT variable for the *KEOS MieGruneisen* model is listed in **Table 52**. Other HISPLT variables are listed in **Section 4.3 on page 245**.

Table 51: Registered Plot Variables for KEOS MieGruneisen

Variable Name	Type	Model	Description
DENSITY	*real*	input	Material density
ENERGY	*real*	input	Specific internal energy per unit mass
PRESSURE	*real*	output	Pressure
TEMPERATURE	*real*	output	Absolute temperature
SOUND_SPEED	*real*	output	Bulk sound speed
DPDRHO	*real*	output	Derivative of pressure with respect to density

Table 52: Extra HISPLT Variables for KEOS MieGruneisen Model

Variable Name	Description
ALPHA	The porosity parameter if the model parameters describe a porosity model. This parameter represents the ratio solid density/porous density.

```
$Simplest sample input for KEOS MieGrun model.  This model uses
$the p-alpha option since RP and PS are defined in EOS_data,
$but the value of RP has been adjusted to the solid density (R0 in
$the EOS_data file) by the user so that the porosity option is not
$used.

  model 131 keos miegruneisen
    matlabel = 'BTF'
    RP = 1.901      $turn off p-alpha model.
  end
```

3.9.3.8 KEOS Sesame

The *KEOS Sesame* model is the most up-to-date implementation of the Sesame equation of state in ALEGRA [23]. It uses the most complete and most recent tables, which are identical to the tables used in CTH. For ALEGRA these tables - *aneos* and *sesame* - are located in the directory set by the $ALEGRA_MIGDATA environment variable. Like most KEOS models, predefined materials are catalogued in the EOS_data file. This model allows simplified input in the form of either the MATLABEL keyword (followed by a predefined material label listed in the EOS_data file) or the NEOS keyword (followed by the table number). Other parameters specified in the model input will overwrite those in the standard model.

This model also allows incorporation of the p-alpha porosity model, which was described in the *KEOS MieGruneisen* model (**Section 3.9.3.7 on page 178**). The Sesame tabular equation of state provides a more robust equation of state during the compaction of a very porous material and is thus more suitable than the *KEOS MieGruneisen* model for many materials. The user should note that a somewhat smaller time step may be required for very porous materials to facilitate the solution during sudden compaction. The p-alpha model is discussed in greater detail in the reference listed below.

If the EOS_data file is used for a predefined material, the user should examine parameters in the EOS_data file to see if the porosity model is included (i.e., if RP>0). If the porosity model is not desired, set RP = R0 in the model input.

Modules: material_libs/kerley_eos/sesame

- sesame_mig.h
- sesame_mig.C
- sesame.doc is the ASCII data file that lists associated Fortran routines.

• Physics: hydrodynamics

Table 53: Input Parameters for KEOS Sesame

Parameter Name	Type	Description
MATLABEL	*char*	Material label in the EOS_data file If this parameter is provided, no other parameters are required. Enclose the string in single quotes.
DATAFILE	*char*	Name and location of the specially formatted ASCII file which contains the properties for this model. The default is $ALEGRA_MIGDATA/EOS_data. Enclose the string in single quotes.
R0 (or RO)	*real*	Initial density in of Hugoniot.
T0 (or TO)	*real*	Initial temperature.
SR	*real (1.0)*	Factor for scaling density and energy. Ratio of molecular weight for table to molecular weight of actual material, SR=MW(table)/MW(material).
FEOS	*real*	Name of file containing table. If the name is given with imbedded “/” characters, the name is taken literally. Otherwise, the path $ALEGRA_MIGDATA is prepended to the name.
NEOS	*real*	Number of table for material. Refer to on-line documentation to find material numbers for specific materials.
RP	*real (=R0)*	For the p-alpha porosity model[a], initial density of porous material.
PS	*real (1.e9)*	For the p-alpha porosity model[a], pressure for complete compaction.
PE	*real (0.)*	For the p-alpha porosity model[a], maximum elastic pressure.

Table 53: Input Parameters for KEOS Sesame (Continued)

Parameter Name	Type	Description
CE	*real*	For the p-alpha porosity model[a], sound speed in elastic compaction region (default for CE is sound speed of solid matrix).
NSUB	*int (10)*	For the p-alpha porosity model[a], number of sub-cycles within time step.
CLIP	*int*	If nonzero, the temperature is constrained to lie within the bounds of the underlying table, and the pressure is calculated from this constrained ("clipped") temperature. If zero, no clipping of the temperature is done, and the pressure is extrapolated off the table.

a.Providing the parameter RP not equal to R0 designates the use of the p-alpha porosity model. Optional input for the porosity model are the parameters PS, PE, CE, and NSUB.

Table 54: Registered Plot Variables for KEOS Sesame

Variable Name	Type	Model	Description
DENSITY	*real*	input	Material density.
ENERGY	*real*	input	Specific internal energy per unit mass.
PRESSURE	*real*	output	Pressure.
TEMPERATURE	*real*	output	Absolute temperature.
SOUND_SPEED	*real*	output	Bulk sound speed.
DPDRHO	*real*	output	Derivative of pressure with respect to density.
ALPHA	*real*	output	Porosity parameter if p-alpha model used.

The extra HISPLT variable for the *KEOS Sesame* model is listed in the following table. Other HISPLT variables are listed in Section 4.3 on page 245.

Table 55: Extra HISPLT Variables for KEOS Sesame

Variable Name	Description
ALPHA	The porosity parameter if the model parameters describe a porosity model. This parameter represents the ratio solid density/porous density.

```
$Sample input for keos sesame with p-alpha model
$included.  Table 4020 is in the table names 'aneos', which is
$located in the $ALEGRA_MIGDATA directory.
model 191 keos sesame
  neos = 4020
  feos = 'aneos'
  r0     = 2.785
  rp     = 2.15
  pe     = 4.5e8
end
```

3.9.3.9 MG POWER

This a general Mie-Grüneisen power law equation of state form generally applicable to solid equations of state but with a fair amount of flexibility on the actual shape of the reference curves. It is assumed that the pressure is related to the density and specific energy through

$$P(\rho, E) = P_R(\rho) + \Gamma\rho[E - E_R(\rho)]$$

and

$$E(\rho, T) = E_R(\rho) + C_V[T - T_R(\rho)]$$

where

$$\Gamma\rho = \Gamma_0\rho_0$$

and C_V are constants. The subscript R refers to a reference state curve which can be an isentrope or Hugoniot for example. This particular model utilizes three regions: a compressive region, a tension region and a "fracture" region as described below. We define the volumetric strain η as

$$\eta = 1 - \frac{\rho_0}{\rho} = 1 - \frac{v}{v_0}.$$

For $\eta > 0$ we define

$$P_R = P_H = K_0\eta(1 + K_1\eta + K_2\eta^2 + K_3\eta^3 + K_4\eta^4 + K_5\eta^5)$$

where

$$K_0 = \rho_0 C_0^2$$

defines the bulk modulus.

$$E_R = E_H = \frac{P_H\eta}{2\rho_0} + E_0$$

Using the method of Walsh and Christian [45] it can be shown that the temperature on the Hugoniot curve is given by

$$T_H = T_0 e^{\Gamma_0\eta} + \frac{e^{\Gamma_0\eta}}{2C_V\rho_0}\int_0^{\eta} e^{-\Gamma_0 z} z^2 \frac{\mathrm{d}}{\mathrm{d}z}\left(\frac{P_H}{z}\right)dz$$

which leads to

$$T_H = T_0 e^{\Gamma_0\eta} + \frac{e^{\Gamma_0\eta}}{2C_V\rho_0}K_0(K_1 I_2 + 2K_2 I_3 + 3K_3 I_4 + 4K_4 I_5 + 5K_5 I_6)$$

where

$$I_n = \int_0^{\eta} e^{-\Gamma_0 z} z^n dz$$

The associated incomplete gamma function is evaluated as a recursion for $\Gamma_0\eta \geq 1$ and as a series otherwise for accuracy purposes.

For $\eta < 0$ we define the reference curve by an isentrope:

$$P_R = P_{ISEN} = K_0\eta$$

$$E_R = E_{ISEN} = \frac{K_0\eta^2}{2\rho_0} + E_0$$

$$T_R = T_{ISEN} = T_0 e^{\Gamma_0\eta}$$

For $\eta < \eta_{min} = \dfrac{P_{min}}{K_0}$ we have

$$P_R = P_{ISEN} = P_{min}$$

$$E_R = E_{ISEN} = \frac{K_0\eta_{min}^2}{2\rho_0} + E_0 + \frac{P_{min}}{\rho_0}(\eta - \eta_{min})$$

$$T_R = T_{ISEN} = T_0 e^{\Gamma_0\eta}$$

In all cases the sound speed is given by

$$c^2 = v^2(\rho_0 P'_R(\eta) - \rho_0\Gamma_0(-p + \rho_0 E'_R(\eta)))$$

- Modules: material_libs/standard_models
  - mgpower.h
  - mgpower.C
- Physics: hydrodynamics

Table 56: Input Parameters for MG POWER

Parameter Name	Type	Description
RHO REF	*real*	Material density at the reference state (required)
TREF	*real*	Absolute temperature at the reference state (T_0).
CV	*real*	Specific heat at constant volume (C_V).
E SHIFT	*real*	Arbitrary energy shift (optional) (E_0).
GAMMA0	*real*	Gruniesen parameter (Γ_0).
C0	*real*	Reference bulk wave speed (C_0). If C0=0, the bulk modulus will be used to calculate the wave speed: C0=SQRT(K0/RHOREF). Either C0 or K0 must be greater than 0; the other must be zero.

Table 56: Input Parameters for MG POWER (Continued)

Parameter Name	Type	Description
K0	*real*	Bulk modulus (K_0). If K0=0 and C0>0, the wave speed will be used to calculate the bulk modulus: K0=RHOREF*C0*C0. Either C0 or K0 must be greater than 0; the other must be zero.
K1	*real*	Hugoniot coefficient (K_1).
K2	*real*	Hugoniot coefficient (K_2).
K3	*real*	Hugoniot coefficient (K_3).
K4	*real*	Hugoniot coefficient (K_4).
K5	*real*	Hugoniot coefficient (K_5).
PRESSURE CUTOFF	*real*	P_{min}

Example:

```
model 61 mg power
  rho ref = 2.37 $ g/cm^3
  tref    = 273. $ K
  gamma0  = 1.
  cv      = 1.5e7 $ erg/g/K
  k0      = 1.96e11 $ dyne/cm^2
  k1      = -4.9
  k2      = 31.0
end
```

Table 57: Registered Plot Variables for MG POWER

Variable Name	Type	Mode	Description
SPECIFIC_HEAT_VOL	*real*	initialize	Initialized to C_v parameter.
DENSITY	*real*	input	Material density.
ENERGY	*real*	input	Specific internal energy per unit mass.
PRESSURE	*real*	output	Pressure.
TEMPERATURE	*real*	output	Absolute temperature.

Table 57: Registered Plot Variables for MG POWER (Continued)

Variable Name	Type	Mode	Description
SOUND_SPEED	*real*	output	Bulk sound speed.

3.9.3.10MG US UP

Mie-Grüneisen Us-Up equation of state is applicable to high compression of metal solids [33]. It provides a convenient way to define an equation of state from shock Hugoniot data. This equation of state model assumes a linear relation between shock velocity, U_s, and particle velocity, u_p, according to

$$U_s = c_0 + sU_p$$

The Grüneisen parameter Γ is a function of density according to

$$\Gamma(\rho) = \Gamma_0 \frac{\rho_o}{\rho}$$

and the material pressure, internal energy and temperature are related to the reference state by

$$P(\rho, E) = P_R(\rho) + \Gamma_0 \rho_0 [E - E_R(\rho)]$$

and

$$E(\rho, T) = E_R(\rho) + C_V[T - T_R(\rho)]$$

where P_R, E_R, and T_R are analytic functions of the density. The above formulation can give misleading results the $\rho < \rho_0$, thus the model has been modified to use an “expansion equation of state” when the material goes into tension. This model was implemented into an early version of CTH [22] where the particular forms of the reference curves used in this model are documented.

- Modules: material_libs/standard_models
  - matmod_mgusup.C
  - matmod_mgusup.h
- Physics: hydrodynamics

Table 58: Input Parameters for MG US UP

Parameter Name	Type	Description
SL	*real*	Linear constant s in U_s-u_p equation

Table 58: Input Parameters for MG US UP (Continued)

Parameter Name	Type	Description
C0	*real*	Constant c_0 in U_s-u_p equation
GAMMA0	*real*	Grüneisen parameter, Γ_0
RHO REF	*real*	Material density at reference state, ρ_0
EREF	*real*	Internal energy at reference state, E_R
PREF	*real*	Pressure at reference state, P_R
TREF	*real*	Absolute temperature at reference state, T_R
CV	*real*	Specific heat at constant volume
E SHIFT	*real*	Arbitrary energy shift (optional)
PRESSURE CUTOFF	*real*	Minimum pressure allowed

Table 59: Registered Plot Variables for MG US UP

Variable Name	Type	Mode	Description
SPECIFIC_HEAT_VOL	*real*	initialize	Initialized from C_v parameter
DENSITY	*real*	input	Material density
ENERGY	*real*	input	Specific internal energy per unit mass
PRESSURE	*real*	output	Pressure
TEMPERATURE	*real*	output	Absolute temperature
SOUND_SPEED	*real*	output	Bulk sound speed
DPDRHO	*real*	output	Derivative of pressure with respect to density

```
$Sample input for mg us up.
$KEOS MieGruneisen will eventually supersede this model.

model 112 mg us up
  c0       = 3.94e5   $cm/s
  sl       = 1.489
  gamma0   = 1.99
  rho ref  = 8.93    $ g/cm^3
  cv       = 3.929e06 $ erg/g/K
```

```
    pref     = 0.0
    tref     = 298.  $ K
  end
```

3.9.4 Constitutive Models

The constitutive models define the equations for the stress and deformation relationships in a material. This section focuses on the basic constitutive models in ALEGRA. A later section provides descriptions of models that may include the constitutive relations with more complex yield behavior.

Let us begin with one comment regarding the use of Young's modulus in the following models. Young's modulus is generally an unfamiliar elastic constant for people used to working with shock waves. Rather, they are used to working with the bulk modulus. Below, we indicate the relationship between these two moduli, given the Poisson ratio ν.

The bulk modulus is defined by

$$B_0 = \rho_0 c_0^2$$

where the subscript "0" refers to reference conditions, B is the bulk modulus, ρ is the density, and c is the sound speed. The bulk modulus is easily computed for most materials from this formula since sound speed and density are almost always known at reference conditions for the shock wave applications we are interested in. If E is the Young's modulus, then it can be found from the bulk modulus by the following formula:

$$E_0 = 3(1-2\nu)\ B_0 = 3(1-2\nu)\rho_0 c_0^2$$

3.9.4.1 ELASTIC PLASTIC

This is a classical elastic plastic constitutive model using a generalized Hooke's Law for elastic stress-strain response, von Mises yield criteria, combined isotropic and kinematic hardening, and simple radial return [12],[40]. The ALEGRA implementation has been adapted from the elastic-plastic model implemented in *PRONTO3D* [41]. The hardening mode is weighted by the input parameter, *BETA*, where 0 is fully kinematic, and 1 is fully isotropic hardening.

Due to input parsing issues, this model is often inadvertently used instead of the Linear Elastic model. For example, a user, intending to apply the Linear Elastic material model to a material might provide the following input:

```
  MODEL 121 ELASTIC
    YOUNGS MODULUS = 1.E12
    POISSONS RATIO = 0.3
```

```
END
```

In this case, the model assumes that a linear elastic material has been specified and sets the initial yield stress to a large number to ensure no yield occurs.

- Modules: material_libs/standard_models
  - matmod_ep.h, matmod_ep.C
- Physics: solid dynamics

Table 60: Input Parameters for ELASTIC PLASTIC

Parameter Name	Type	Description
Youngs Modulus	*real*	Young's Modulus of the material
Poissons Ratio	*real*	Poisson's ratio of the material
Yield Stress	*real*	Initial yield stress in uniaxial tension
Hardening Modulus	*real*	Hardening modulus for yield stress
Beta	*real*	Weight for kinematic/isotropic hardening (0=fully kinematic, 1=fully isotropic).

Table 61: Registered Plot Variables for ELASTIC PLASTIC

Variable Name	Type	Mode	Description
BULK_MODULUS	*real*	initialize	Bulk modulus
SHEAR_MODULUS	*real*	initialize	Shear modulus
DEFORMATION_RATE	*symtensor*	input	Rate of deformation tensor
STRESS	*symtensor*	ioput	Cauchy stress tensor
BACK_STRESS	*symtensor*	ioput	Cauchy back stress tensor for kinematic hardening
EQPS	*real*	ioput	Equivalent plastic strain
YIELD_STRESS	*real*	ioput	Yield stress in uniaxial tension

```
$Sample input for elastic plastic model
model 31 elastic plastic
  youngs modulus    = 1.076e+12   $ dyne/cm^2
  poissons ratio    = 0.355
```

```
    yield stress       = 6.0e+09     $ dyne/cm^2
    hardening modulus = 2.0e+09     $ dyne/cm^2
    beta               = 1.0
  end
```

3.9.4.2 LINEAR ELASTIC

This model [40] computes the stress tensor using an incremental, generalized Hooke's Law.

$$d\sigma_{ij} = \lambda d\varepsilon_{kk}\delta_{ij} + 2\mu d\varepsilon_{ij}$$

where σ and ε are the stress and strain, respectively, and λ and μ are Lame constants. The incremental strain is obtained by integrating the deformation rate over the time interval of the step.

- Modules: material_libs/standard_models
- Physics: solid dynamics

Table 62: Input Parameters for LINEAR ELASTIC

Parameter Name	Type	Description
Youngs Modulus	*real*	Young's Modulus of the material
Poissons Ratio	*real*	Poisson's ratio of the material

Table 63: Registered Plot Variables for LINEAR ELASTIC

Variable Name	Type	Mode	Description
BULK_MODULUS	*real*	initialize	Bulk modulus
SHEAR_MODULUS	*real*	initialize	Shear modulus
DEFORMATION_RATE	*symtensor*	input	Rate of deformation tensor
STRESS	*symtensor*	ioput	Cauchy stress tensor

```
  $Sample input for linear elastic model.
  model 41 linear elastic
    youngs modulus    = 1.076e+12   $ dyne/cm^2
    poissons ratio    = 0.355
 end
```

3.9.4.3 SOIL CRUSHABLE FOAM

This is an implementation of the PRONTO soil and crushable foam model [40]. This model has a pressure-volumetric strain response superposed on a basic elastic plastic response. The pressure-volumetric strain controls pressure lockup, unload, fracture, and reloading response of the material. The yield stress is specified as a polynomial in pressure, p (positive in compression)

$$\sigma_D = a_0 + a_1 \cdot P + a_2 \cdot P^2$$

The volumetric stress-strain curve input as a function has the constraint that the slope of the function must be, everywhere, less that or equal to the unloading bulk modulus, K_0. The function is specified as pairs of pressure and volumetric strain values, where volumetric strain is ε_v,

$$\varepsilon_v = -\ln(\rho_0 / \rho)$$

If no function is specified, then the model produces a purely elastic behavior.

- Modules: material_libs/standard_models
- Physics: solid dynamics

Table 64: Input Parameters for SOIL CRUSHABLE FOAM

Parameter Name	Type	Description
SHMOD	*real*	Shear modulus, μ
BULK	*real*	Bulk modulus, K_0
A0	*real*	Constant parameter for yield stress
A1	*real*	Linear parameter for yield stress
A2	*real*	Quadratic parameter for yield stress
PFRAC	*real*	Fracture pressure or tensile limit
PMAX	*real*	Not input; normally set to -A1/(2.*A2), or a large number of A2=0. If the pressure > PMAX, the yield stress is set using PMAX.
FUNCTION TABLE	*real*	Function table id for an input P-v curve

Table 65: Registered Plot Variables for SOIL CRUSHABLE FOAM

Variable Name	Type	Mode	Description
DEFORMATION_RATE	*symtensor*	INPUT	Deformation rate tensor
STRESS	*symtensor*	IOPUT	Cauchy stress tensor
EV	*state*	IOPUT	Volumetric strain
EVFRAC	*state*	IOPUT	Volumetric strain at fracture
EVMAX	*state*	IOPUT	Maximum previous volumetric strain
NUM	*state*	IOPUT	Last increment in pressure function where interpolate was found.

```
$Sample input for soil crushable foam model.
model 51 soil crushable foam
  shmod = 1.4e10
  bulk  = 2.76e11
  a0    = 398083.
  a1    = 0.019245
  a2    = -1.163e-10
  pfrac = -18595369.
  pmax  = 82738607.
  func table = 51
end

function 51  $ P-mu curve for soil material
$   mu           P
  0.00000    0.00000E+00
  0.00500    9.38215E+07
  0.01000    1.88356E+08
  0.01500    2.95625E+08
$   (data omitted in sample for the sake of brevity)
  0.51500    1.15825E+11
  0.52000    1.17203E+11
end
```

3.9.5 Yield Models

3.9.5.1 STEINBERG GUINAN LUND

This model is the MIG implementation of the Steinberg-Guinan-Lund model [42], which is also used in the CTH code. The model and its implementation is described fully by Taylor, but the basic equations will be outlined here as a brief introduction. The Steinberg-Guinan-Lund model predicts the viscoplastic response of various materials (principally metals) based on a consideration of thermally-activated dislocation mechanics. The strain rate dependent form of the SGL model defines the yield stress as

$$Y = [Y_T(\dot{\varepsilon}, T) + Y_A f(\varepsilon^p)]\frac{G(P, T)}{G_o},$$

where the athermal and thermally activated components $Y_A f(\varepsilon^{p)}$ and Y_T are defined by

$$Y_A f(\varepsilon^p) = Y_A\{1 + \beta(\varepsilon^p + \varepsilon_i)\}^n$$

and

$$\dot{\varepsilon}^p = \left(\frac{1}{C_1} exp\left[\frac{2U_K}{T}\left(1 - \frac{Y_T}{Y_P}\right)^2\right] + \frac{C_2}{Y_T}\right)^{-1}.$$

In these equations, P and T are the pressure and temperature, ε^p and $\dot{\varepsilon}^I$ are the equivalent plastic strain and its time derivative, Y_A is the yield strength at the Hugoniot elastic limit, $f(\varepsilon^p)$ is the work-hardening function with β, ε, and n used as fitting parameters. G_o is the initial shear modulus, Y_P is the Peierls stress, and $2U_K$ is the energy necessary to form a pair of kinks in a dislocation segment. The quantities C_1 and C_2 are defined in terms of various dislocation mechanics parameters and are specific to the material being modeled. Two limits are imposed in this model.

$$Y_A f(\varepsilon^p) \le Y^{\circ}_{max}$$

and

$$Y_T \le Y_P,$$

where Y°_{max} is the work-hardening maximum in the rate-dependent version of the model.

The rate -independent model is a special case of the rate depend form with Y_T set to zero and the following limit applied:

Table of Contents
General Input

$$Y_0 f(\varepsilon^p) \le Y_{max}.$$

- Modules: material_libs/steinberg-guinan-lund
  - sgl_mig.h, sgl_mig.C
  - st.dat is the ASCII data file that lists the associated Fortran routines.
- Physics: solid dynamics

Table 66: Input Parameters for STEINBERG GUINAN LUND

Parameter Name	Type	Description
R0ST	*real*	Initial density ρ_0
TM0ST	*real*	Melting temperature T_m
ATMST	*real*	Material constant a in Lindemann melting law
GM0ST	*real*	Initial Grüneisen coefficient γ_0
AST	*real*	Material constant A in definition of shear modulus
BST	*real*	Material constant B in definition of shear modulus
NST	*real*	Parameter n in work hardening function f
C1ST	*real*	Material constant C_1 for thermal yield stress
C2ST	*real*	Material constant C_2 for thermal yield stress
G0ST	*real*	Initial shear modulus G_0
BTST	*real*	Parameter β in work-hardening function f
EIST	*real*	Initial equivalent plastic strain ε_i
YPST	*real*	Peierls stress and max. value of Y_T
UKST	*real*	Activation energy
YSMST	*real*	Max yield stress of athermal yield component $\overset{\circ}{Y}_{max}$
YAST	*real*	Athermal yield stress prefactor Y_A
Y0ST	*real*	Initial yield stress for rate-independent version Y_0
YMST	*real*	Max yield stress for rate-independent version Y_{max}

Table 67: Registered Plot Variables for STEINBERG GUINAN LUND

Variable Name	Type	Mode	Description
DEFRATE	*symtensor*	input	Rate of deformation tensor
MATFRAC	*real*	input	Volume fraction of material
TEMPERATURE	*real*	input	Absolute temperature
DENSITY	*real*	input	Material density
PRESSURE	*real*	input	Pressure
EQPS	*real*	input	Equivalent plastic strain (previous cycle)
STRESS	*real*	input	Cauchy stress (deviator from previous cycle, mean stress is current pressure)
YIELD_STRESS	*real*	output	Yield stress in uniaxial tension
SHEAR_MODULUS	*real*	output	Shear modulus

```
$Sample input for steinberg guinan lund yield model.

model 103 steinberg guinan lund
  R0ST   = 16.69
  TM0ST  = 4340.
  ATMST  = 1.30
  GM0ST  = 1.67
  AST    = 1.45E-12
  BST    = 1.508650
  NST    = 0.10
  C1ST   = 0.71E+06
  C2ST   = 0.12E+06
  G0ST   = 0.690E+12
  BTST   = 10.0
  EIST   = 0.00
  YPST   = 8.2E+09
  UKST   = 0.31
  YSMST  = 4.5E+09
  YAST   = 3.75E+09
  Y0ST   = 7.7E+09
  YMST   = 1.1E+10
end
```

3.9.5.2 JOHNSON COOK EP

This model is the MIG implementation of the Johnson-Cook viscoplastic model [34]. This MIG module is also used in the CTH code. In this model the yield stress is dependent on temperature, rate of deformation, and history of deformation. The governing equation has the following form:

$$Y = [A + B(\varepsilon^p)^N][1 + C\,ln(max(0.002, \dot{\varepsilon}^p))][1 - \theta_h^m],$$

where A, B, C, N, and m are constants that depend on the material, and $\dot{\varepsilon}^p$ is the time derivative of the equivalent plastic strain. θ_h is the homologous temperature, defined by

$$\theta_h^m = \frac{T - T_r}{T_M - T_r}$$

where T_r and T_M are room temperature and the material melting temperatures. Silling [34] describes the implementation of the equations in detail and also provides the parameters for 12 materials.

- Modules: material_libs/johnson_cook_ep
  - jcep_mig.h, jcep_mig.C
  - jcep.dat is the ASCII data file that lists the Fortran files.
- Physics: solid dynamics

Table 68: Input Parameters for JOHNSON COOK EP

Parameter Name	Type	Description
AJO	*real*	Material constant A
BJO	*real*	Material constant B
CJO	*real*	Material constant C
MJO	*real*	Material constant m
NJO	*real*	Material constant N
TJO	*real*	Material melting temperature T_m

Table 69: Registered Plot Variables for JOHNSON COOK EP

Variable Name	Type	Mode	Description
DEFRATE	*symtensor*	input	Rate of deformation tensor
MATFRAC	*real*	input	Volume fraction of material
TEMPERATURE	*real*	input	Absolute temperature
EQPS	*real*	input	Equivalent plastic strain (previous cycle)
STRESS	*real*	input	Cauchy stress (deviator from previous cycle, mean stress is current pressure)
SHEAR_MODULUS	*real*	input	Shear modulus
YIELD_STRESS	*real*	output	Yield stress in uniaxial tension

```
$Sample input for Johnson-Cook model for copper.

model 113 johnson cook ep
  ajo = 8.970000E+08
  bjo = 2.918700E+09
  cjo = 2.500000E-02
  mjo = 1.090000E+00
  njo = 3.100000E-01
  tjo = 1.189813E-01
end
```

3.9.5.3 ZERILLI ARMSTRONG

This model is the MIG implementation of the Zerilli-Armstrong viscoplastic model [34]. This MIG module is also used in the CTH code. Like the Johnson-Cook model, in this model the yield stress is dependent on temperature, rate of deformation, and history of deformation. The Zerilli-Armstrong model is based on a physical model of the crystal structure of the material. The simplified governing equation has the following form:

$$Y = \nabla\sigma'_G + k\sqrt{l} + (c_1 + c_2\sqrt{\varepsilon^p})\,exp(-c_3 T + c_4 T\,ln\dot{\varepsilon}^p) + c_5(\varepsilon^p)^N,$$

where the parameters are defined in the table below, T is temperature, and ε^p is the equivalent plastic strain. Silling [34] describes the implementation of the equation in detail and also provides the parameters for copper and iron.

- Modules: material_libs/zerilli_armstrong
  - za_mig.h, za_mig.C
  - zadrv.f
  - za.dat is the ASCII data file for further information.
- Physics: solid dynamics

Table 70: Input Parameters for ZERILLI ARMSTRONG

Parameter Name	Type	Description
C1ZE	*real*	Material constant c_1 (units of pressure)
C2ZE	*real*	Material constant c_2 (units of pressure)
C3ZE	*real*	Material constant c_3 (units of temperature^{-1})
C4ZE	*real*	Material constant c_4 (units of temperature^{-1})
C5ZE	*real*	Material constant c_5 (units of pressure)
AZE	*real*	Material constant $\nabla\sigma'_G + k\sqrt{l}$ (units of pressure)
NZE	*real*	Material constant N (dimensionless)

Table 71: Registered Plot Variables for ZERILLI ARMSTRONG

Variable Name	Type	Mode	Description
DEFRATE	*symtensor*	input	Rate of deformation tensor
MATFRAC	*real*	input	Volume fraction of material
TEMPERATURE	*real*	input	Absolute temperature
EQPS	*real*	input	Equivalent plastic strain (previous cycle)
STRESS	*real*	input	Cauchy stress (deviator from previous cycle, mean stress is current pressure)

Table 71: Registered Plot Variables for ZERILLI ARMSTRONG (Continued)

Variable Name	Type	Mode	Description
SHEAR_MODULUS	*real*	input	Shear modulus
YIELD_STRESS	*real*	output	Yield stress in uniaxial tension

```
$Sample input for zerilli armstrong

model 123 zerilli armstrong
 C1ZE = 0.000000E+00
 C2ZE = 8.900000E+09
  C3ZE = 2.80e-3
  C4ZE = 1.15e-4
  C5ZE = 0.000000E+00
  AZE  = 6.500000E+08
  NZE  = 1.000000E+00
end
```

3.9.5.4 BAMMANN CHIESA JOHNSON

This model is the MIG implementation of the Bammann-Chiesa-Johnson viscoplastic damage model [43], which is also used in the CTH code. Taylor describes the model and implementation fully and also provides the parameters for five metals.

The viscoplastic component of this model incorporates isotropic and kinematic hardening as well as strain rate and thermal effects. Damage modeling is based on an analytic expression for spherical void growth. The damage growth rate is dependent on the effective stress, tensile pressure, plastic strain rate, and the current damage level. The basic equations are outlined below.

A linear elastic response is represented by

$$\hat{\underset{\sim}{\sigma}} = \lambda(1-\varphi)tr(\underset{\sim}{D}^e)\underset{\sim}{1} + 2\mu(1-\varphi)\underset{\sim}{D}^e - \frac{\dot{\varphi}}{(1-\varphi)}\underset{\sim}{\sigma},$$

where the Cauchy stress $\underset{\sim}{\sigma}$ is convected with the spin $\underset{\sim}{W}$ according to the Jaumann rate

$$\hat{\underset{\sim}{\sigma}} = \dot{\underset{\sim}{\sigma}} - \underset{\sim}{W}\underset{\sim}{\sigma} + \underset{\sim}{\sigma}\underset{\sim}{W}.$$

$\underset{\sim}{D}^e$ is the elastic part of the rate of deformation tensor $\underset{\sim}{D}$, λ and μ are the Lame' elastic constants, and φ $(0 \le \varphi \le 0.99)$ is the damage.

- Modules: material_libs/bammann_chiesa_johnson
    - bcjvpd_mig.h, bcjvpd_mig.C
    - bcjvpd.dat is the ASCII data file that lists the associated Fortran files.

Table 72: Input Parameters for BAMMANN CHIESA JOHNSON

Parameter Name	Type	Description
DENSITY0	*real*	Initial density
YOUNGS MODULUS	*real*	Young’s modulus, used to determine the shear modulus.
POISSONS RATIO	*real*	Poisson’s ratio, used to determine the shear modulus.
TEMPERATURE0	*real*	Initial temperature
HC	*real*	Heat coefficient ($1/\rho * C_v$)
C1	*real*	Coefficient and exponent for function $V(\theta)$
C2	*real*	Coefficient and exponent for function $V(\theta)$
C3	*real*	Parameter for function $Y(\theta)$
C4	*real*	Parameter for function $Y(\theta)$
C5	*real*	Coefficient and exponent for function $f(\theta)$
C6	*real*	Coefficient and exponent for function $f(\theta)$
C7	*real*	Coefficient and exponent for function $r_d(\theta)$
C8	*real*	Coefficient and exponent for function $r_d(\theta)$
C9	*real*	Parameter for function $h(\theta)$
C10	*real*	Parameter for function $h(\theta)$
C11	*real*	Parameter for function $r_s(\theta)$
C12	*real*	Parameter for function $r_s(\theta)$
C13	*real*	Coefficient and exponent for function $R_d(\theta)$
C14	*real*	Coefficient and exponent for function $R_d(\theta)$
C15	*real*	Parameter for function $H(\theta)$
C16	*real*	Parameter for function $H(\theta)$
C17	*real*	Coefficient and exponent for function $R_s(\theta)$

Table 72: Input Parameters for BAMMANN CHIESA JOHNSON (Continued)

Parameter Name	Type	Description
C18	*real*	Coefficient and exponent for function $R_s(\theta)$
C19	*real*	Parameter for function $Y(\theta)$
C20	*real*	Parameter for function $Y(\theta)$
A1	*real*	Initial value of backstress component α_{11}
A2	*real*	Initial value of backstress component α_{22}
A3	*real*	Initial value of backstress component α_{12}
A4	*real*	Initial value of backstress component α_{23}
A5	*real*	Initial value of backstress component α_{23}
A6	*real*	Initial value of scalar hardening variable κ
DEX	*real*	Value of exponent m in definition of damage variable φ
D0	*real*	Initial value of the damage variable φ
FS0	*real*	Initial value of the material spall strength p_0^f

Table 73: Registered Plot Variables for BAMMANN CHIESA JOHNSON

Variable Name	Type	Mode	Description
DEFRATE	*symtensor*	input	Rate of deformation tensor
TEMPERATURE	*real*	input	Absolute temperature
PRESSURE	*real*	input	Pressure
STRESS	*real*	input	Cauchy stress
BACK_STRESS	*real*	input	Back stress
EQPS	*real*	input	Equivalent plastic strain
PLSNRT	*real*	output	Plastic strain rate
DAMAGE	*real*	ioput	Damage fraction
KAPP	*real*	ioput	Scalar hardening variable κ
BETA	*real*	ioput	Viscoplastic rate β

Table 73: Registered Plot Variables for BAMMANN CHIESA JOHNSON (Continued)

Variable Name	Type	Mode	Description
DAMR	*real*	ioput	Damage rate $\dot{\phi}$
BCJP	*real*	ioput	BCJ tensile pressure

Table 74: Extra HISPLT Variables for BAMMANN CHIESA JOHNSON

Hisplt Variable Name	Description
PLSNRT	Plastic strain rate
DAMAGE	Damage fraction
KAPP	Scalar hardening variable κ
BETA	Viscoplastic rate β
DAMR	Damage rate $\dot{\phi}$
BCJP	BCJ tensile pressure

```
$Sample input for Bammann-Chiesa-Johnson
$HY-80_STEEL (taken from cth VP_data file)

model 123 bammann chiesa johnson
  DENSITY0        =  7.831000E+00
  YOUNGS MOD      =  2.069000E+12
  POISSONS RATIO =  3.000000E-01
  TEMPERATURE     =  2.536947E-02
  HC              =  0.000000E+00
  C1              =  0.000000E+00
  C2              =  0.000000E+00
  C3              =  5.449000E+09
  C4              =  0.000000E+00
  C5              =  1.000000E+00
  C6              =  0.000000E+00
  C7              =  5.728000E-09
  C8              =  0.000000E+00
  C9              =  4.262000E+10
  C10             =  0.000000E+00
  C11             =  0.000000E+00
  C12             =  0.000000E+00
  C13             =  1.069000E-10
```

```
    C14              =  0.000000E+00
    C15              =  2.262000E+09
    C16              =  0.000000E+00
    C17              =  0.000000E+00
    C18              =  0.000000E+00
    A1               =  0.000000E+00
    A2               =  0.000000E+00
    A3               =  0.000000E+00
    A4               =  0.000000E+00
    A5               =  0.000000E+00
    A6               =  0.000000E+00
    DEX              =  3.700000E+00
    D0               =  1.000000E-04
    FS0              =  3.670000E+10
    C19              =  0.000000E+00
    C20              =  0.000000E+00
  end
```

3.9.5.5 VON MISES YIELD

This model [12],[40] provides a calculation of yield stress based upon an isotropic hardening, von Mises criterion. When this model is sequenced with ***LINEAR ELASTIC***, and ***SIMPLE RADIAL RETURN***, it is exactly equivalent to ***ELASTIC PLASTIC*** for BETA=1.0.

```
  MATERIAL 10 COPPER
    MODEL 11 $ linear elastic
    MODEL 12 $ von mises yield
    MODEL 13 $ simple radial return
    MODEL 14 $ generic eos
    DENSITY = 8.932
    TEMPERATURE = 298.
  END
  MODEL 11 LINEAR ELASTIC
    ...
  END
  MODEL 12 VON MISES YIELD
    ...
  END
  MODEL 13 SIMPLE RADIAL RETURN
  END
  MODEL 14 GENERIC EOS
    ...
  END
```

- Modules: material_libs/standard_models
  - von_mises_yield.h, von_mises_yield.C

- Physics: solid dynamics:

Table 75: Input Parameters for VON MISES YIELD

Parameter Name	Type	Description
Youngs Modulus	*real*	Young's Modulus of the material
Poissons Ratio	*real*	Poisson's ratio of the material
Yield Stress0	*real*	Initial yield stress in uniaxial tension
Hardening Modulus	*real*	Hardening modulus for yield stress

Table 76: Registered Plot Variables for VON MISES YIELD

Variable Name	Type	Mode	Description
STRESS	*symtensor*	input	Trial cauchy stress
YIELD_STRESS	*real*	ioput	Yield stress in uniaxial tension

3.9.6 Plasticity Models

3.9.6.1 SIMPLE RADIAL RETURN

This model [12],[40] performs stress relaxation to return the deviatoric stress onto the yield surface and computes the equivalent plastic strain incurred by this relaxation. This model is an extraction of the plasticity algorithm used in the ***ELASTIC PLASTIC*** model. This model has no input parameters.

- Modules: material_libs/standard_models
  - simple_radial_return.h, simple_radial_return.C
- Physics: solid dynamics

Table 77: Input Parameters for SIMPLE RADIAL RETURN

Parameter Name	Type	Description
no parameters		

$

Table 78: Registered Plot Variables for SIMPLE RADIAL RETURN

Variable Name	Type	Mode	Description
STRESS	*symtensor*	ioput	Cauchy stress tensor (input: trial stress, output: stress on yield surface)
EQPS	*real*	ioput	Equivalent plastic strain

```
$Example of request for simple radial return model.

model 43 simple radial return
end
```

3.9.6.2 EP RADIAL RETURN

This is the MIG implementation of the radial return plasticity algorithm used in CTH. This model performs stress relaxation to return the deviatoric stress onto the yield surface and computes the equivalent plastic strain incurred by this relaxation. This model is equivalent to the SIMPLE RADIAL RETURN model. However, it is implemented for use as a submodel in the CTH ELASTIC PLASTIC model. This model has no input parameters.

- Modules: material_libs/ep_radial_return
  - eprr_mig.h, eprr_mig.C
  - eprmig.F
  - ep_radial_return.dat is the associated data file.
- Physics: solid dynamics

Table 79: Input Parameters for EP RADIAL RETURN

Parameter Name	Type	Description
no parameters		

Table 80: Registered Plot Variables for EP RADIAL RETURN

Variable Name	Type	Mode	Description
MATFRAC	*real*	input	Material fraction of element volume
DEFRATE	*symtensor*	input	Rate of deformation tensor

Table 80: Registered Plot Variables for EP RADIAL RETURN (Continued)

Variable Name	Type	Mode	Description
SPIN	*antitensor*	input	Spin tensor, antisymmetric part of velocity gradient
SHEAR_MODULUS	*real*	input	Shear modulus
YIELD_STRESS	*real*	input	Yield stress in uniaxial tension
STRESS	*symtensor*	ioput	Cauchy stress tensor (input: trial stress, output: stress on yield surface)
EQPS	*real*	ioput	Equivalent plastic strain

```
$Example of request for ep radial return model.

model 43 ep radial return
end
```

3.9.7 Combined Models

3.9.7.1 CTH ELASTIC PLASTIC

This model replicates the elastic-plastic algorithm in CTH [34]. It consists of three submodels, two of which may be user specified: an equation of state submodel and a yield stress submodel. The third model is the ***EP RADIAL RETURN*** model. The calculational sequence of this model begins with the equation of state model to compute pressure, temperature, and sound speed. Next, the shear modulus is calculated as a function of bulk sound speed and constant Poisson's ratio.

$$G = \frac{3(1-2\nu)}{2(1+\nu)}\rho C^2$$

The shear modulus is used to compute an elastic deviatoric stress increment. Next, the yield model is applied to determine if the deviatoric stress increment exceeds the current yield criterion. Then, the ***EP RADIAL RETURN*** model is used to perform the plasticity calculation to enforce the yield criterion.

This model is an example of a combined model which has a specific sequence and set of submodels to be used in computing the material state. Currently, the models in #### are compatible with ***CTH ELASTIC PLASTIC*** and may be used as submodels. A sample input fragment is shown below.

Table 81: Compatible Models for CTH ELASTIC PLASTIC

Submodel Type	Models
Equation of State Models	***KEOS MieGruneisen*** ***KEOS Sesame*** ***MG US UP*** ***KEOS Sesame***
Yield Models	***STEINBERG GUINAN LUND*** ***JOHNSON COOK EP*** ***ZERILLI ARMSTRONG*** ***BAMMANN CHIESA JOHNSON***

```
MATERIAL 10 OFHC Copper
  MODEL 11 $ CTH EP
  DENSITY = 8.932
  TEMPERATURE = 298.
END
MODEL 11 CTH EP
  EOS MODEL = 12   $ mg us up
  YIELD MODEL = 13 $ johnson cook
  POISSON RATIO = 0.3
END
MODEL 12 MG US UP
  ...
END
MODEL 13 JOHNSON COOK EP
  ...
END
```

- Modules: material_libs/combined_models
  - cth_ep.h, cth_ep.C
- Physics: solid dynamics

Table 82: Input Parameters for CTH ELASTIC PLASTIC

Parameter Name	Type	Description
EOS MODEL	*int*	Model id of equation of state model
YIELD MODEL	*int*	Model id of yield stress model
POISSONS RATIO	*real*	Poisson’s ratio

Table 83: Registered Plot Variables for CTH ELASTIC PLASTIC

Variable Name	Type	Mode	Description
DENSITY	*real*	input	Material density
SOUND_SPEED	*real*	ioput	Sound speed
SHEAR_MODULUS	*real*	output	Shear modulus

3.9.7.2 BFK CONCRETE

This model replicates the brittle fracture kinetics (BFK) concrete model in CTH [35]. This combined model consists of submodels which may NOT be user specified, including models for the equation of state, yield, radial return, and fracture. The calculational sequence of this model begins with a call to the equation of state to compute pressure, temperature, and sound speed. This is followed by the calculation of the flow stress. The radial return algorithm is called to enforce the yield criterion and provide the plastic strain rate. Then follows the calculation of the extra variables, including the equivalent plastic strain. Finally, the fracture algorithm is called to compute the void insertion required based on the fracture pressure calculated previously. In Eulerian problems, after the remap step the equation of state and the fracture algorithms are called again.

Currently only one model is predefined for concrete. This model, SAC5, represents small aggregate concrete with an unconfined compressive strength around 6,000 psi. The input parameter COSFAC can be used to scale all strength parameters. For example, if the desired material has an unconfined compressive strength of 5,000 psi, the default parameters for SAC5 can be used with COSFAC=0.8333.

```
MATERIAL 10 Concrete
  MODEL 11 $ BFK CONCRETE
END
MODEL 11 BFK CONCRETE
  MATLABEL = 'SAC5'   $ predefined concrete model
  cotau  = 1000.e-6   $ modify parameter
END
```

- Modules: material_libs/bfk_concrete
  - bfk_concrete_mig.h, bfk_concrete_mig.C
  - MIG driver routines: elvpco.F, elivco.F, eoscos.F, eoscov.F, eoscox.F
  - MIG input, data check, and extra variable routines: eoscoi.F, concck.F, concxv.F

- MIG utility routines: random.F, splint.F, gauss.F,splco.F
- MIG routines, miscellaneous: condis.F, conmfs.F, conhro.F

- Physics: solid dynamics

Table 84: Input Parameters for BFK CONCRETE

Parameter Name	Type	Description
DATAFILE	*char*	Name and location of the specially formatted EOS_data file if different from the default $ALEGRA_MIGDATA/ EOS_data. Enclose the string in single quotes.
MATLABEL	*char*	Material label in the EOS_data file If this parameter is provided, no other parameters are required. Enclose the string in single quotes. The only material currently predefined in the EOS_data file is called SAC5.
COCRX	*real*	Switch for crack visualization (not used).
COSC0	*real*	Unconfined compressive strength, s_{c0}.
COST0	*real*	Unconfined tensile strength, s_{t0}.
COSD0	*real*	Brittle-ductile transition stress, s_{d0}.
COSHRI	*real*	Instantaneous shear modulus, μ_i.
COCC	*real*	Compression at crush, μ_c
COFRA1	*real*	Fragment size coefficient, A_1
COSHRF	*real*	Failed shear modulus, μ_f.
COSTI	*real*	Instantaneous unconfined tensile strength, s_{ti}.
COCH	*real*	Hardening coefficient, C_h
COK0	*real*	Initial bulk modulus, k_0
COKN	*real*	Limiting bulk modulus, k_0
COCV	*real*	Specific heat, C_v
CORHO	*real*	Reference density, ρ_0.
COY0	*real*	Initial yield stress, Y_0.
COREC	*real*	Recovery compression, μ_{rec}.

Table 84: Input Parameters for BFK CONCRETE (Continued)

Parameter Name	Type	Description
COTAU	*real*	Fundamental time to failure, τ_0
COSCI	*real*	Instantaneous unconfined compressive strength, s_{ci}.
COGRU	*real*	Gruneisen coefficient, Γ_0
COYMIN	*real*	Residual flow stress, Y^0_{res}
COYMAX	*real*	Limiting flow stress, $Y_{\max}$.
COKF	*real*	Failed bulk modulus, k_0
COCRXS	*real*	Critical overload for crack growth (not used)
COCRXE	*real*	Critical strain for crack growth, $\varepsilon_{\mathrm{grow}}$ (not used)
COCRXV	*real*	Crack growth velocity (not used)
COCRXM	*real*	Maximum number of crack tracer particles per cell (not used).
COQDF	*real*	Maximum fragment size for CDF, $\lambda_{\max}$. If 0, no CDF made.
COFRA2	*real*	Fragment size coefficient, A_2
COTSOF	*real*	Thermal softening temperature,T_{soft}
COED1	*real*	Equivalent plastic strain at onset of ductile damage, ε_{d1}
COED2	*real*	Equivalent plastic strain at full ductile damage, ε_{d1}
COPD1	*real*	Pressure at onset of compressive damage, p_{d1}
COPD2	*real*	Pressure at full compressive damage, p_{d2}
COTSPL	*real*	Principal stress at spall, T_{spall}
COBFIC	*real*	Critical brittle fracture impulse, $\overline{\Omega}_1$
CODIL	*real*	Dilantancy parameter (not currently used).
COTTRJ	*real*	Trajectory time parameter.
COFTRJ	*real*	Trajectory parameter.
COFMOB	*real*	Surface mobility coefficient, C_{surf}
COFDEP	*real*	Surface effect depth, D_{surf}

Table 84: Input Parameters for BFK CONCRETE (Continued)

Parameter Name	Type	Description
COSFAC	*real*	Multiplier for strength parameters.
COTREL	*real*	Time at which discard flag is set.
COVREL	*real*	Minimum velocity for discard.
COBREL	*real*	Minimum value of COBFIC ($\overline{\Omega}_1$) for discard.
COTDIS	*real*	Discard time (greater than or equal the to COTREL)
NXSRF *[NYSRF]* *[NZSRF]*	*int*	Number of potential spall surfaces in planes aligned with X, Y, and Z coordinated planes.
CXSRF0 *CXSRF1* *CXSRF2* *...* *CXSRF9*	*real*	Potential spall surfaces defined by a constant value of x-coordinate. (This basic capability was brought from CTH, later to be extended so that the code can find the surface.)
CYSRF0 *CYSRF1* *CYSRF2* *...* *CYSRF9*	*real*	Potential spall surfaces defined by a constant value of y-coordinate. (This basic capability was brought from CTH, later to be extended so that the code can find the surface.)
CZSRF0 *CZSRF1* *CZSRF2* *...* *CZSRF9*	*real*	Potential spall surfaces defined by a constant value of z-coordinate. (This basic capability was brought from CTH, later to be extended so that the code can find the surface.)
NTBLCO	*int*	Number of data pairs for experimental Hugoniot data (data given by CTBLXn and CTBLYn.) These data are used for a cubit splint fit for the loading part of the curve. At compressions greater than those provided in the table, the reference curve follows a straight line in p-μ space with slope COKN.
CTBLX1 *CTBLX2* *CTBLX3* *...* *CTBLXF*	*real*	Data for compression, $\mu=(\rho/\rho_0) - 1$, used for the experimental Hugoniot data.

Table 84: Input Parameters for BFK CONCRETE (Continued)

Parameter Name	Type	Description
CTBLY1 *CTBLY2* *CTBLY3* ... *CTBLYF*	*real*	Pressure data for the experimental Hugoniot data.

Table 85: Registered Plot Variables for BFK CONCRETE

Variable Name	Type	Mode	Description
DENSITY	*real*	ioput	Material density
ENERGY	*real*	input	Specific energy.
PRESSURE	*real*	output	Pressure.
TEMPERATURE	*real*	output	Temperature
SOUND_SPEED	*real*	output	Sound speed
DPDRHO	*real*	output	Derivative $dP/d\rho$.
STRESS	*symtensor*	ioput	Cauchy stress tensor.
YIELD_STRESS	*real*	ioput	Yield in tension.
SHEAR_MODULUS	*real*	output	Shear modulus
DEFRATE	*symtensor*	input	Rate of deformation tensor.
PLAS_STRN_RATE	*real*	output	Scalar plastic strain rate.
SPIN	*antitensor*	input	Spin tensor, antisymmetric part of velocity gradient
EQPS	*real*	output	Equivalent plastic strain.
MD	*real*	ioput	Maximum density, $1/\bar{v}$.
DMG	*real*	ioput	Damage variable, ϕ.
FS	*real*	ioput	Fracture stress, $-T_b$.
OL	*real*	ioput	Overload, Ω.
BFI	*real*	ioput	Brittle fracture impulse,$\bar{\Omega}$.

Table 85: Registered Plot Variables for BFK CONCRETE (Continued)

Variable Name	Type	Mode	Description
STN	*real*	ioput	Strength, Y_{inf}
MFS	*real*	ioput	Mean fracture size, λ_f.
REL	*real*	ioput	Release (discard) flag (not yet implemented.)

3.9.8 Fracture Models

3.9.8.1 FRAC PRESDEP

This pressure dependent fracture model [28] uses a void insertion algorithm to allow the volume occupied by a material in an element to decrease, thus allowing the density to increase and the pressure to relax to zero. This algorithm is triggered when the pressure in the material is less than the fracture pressure. A Newton iteration scheme is used in which the density is increased and the equation of state model computes the new pressure. When the pressure converges to the fracture pressure, the volume of void inserted is determined by the density change required to produce the fracture pressure. In subsequent cycles, the fracture pressure is gradually decreased until it is zero.

Note that the material input must have an equation of state model preceding this fracture model to compute pressure and energy. A typical application of this model would use the following input.

```
MATERIAL 10 some material
  MODEL 11 $ mg us up
  MODEL 12 $ frac pres dep
  ...
END
MODEL 11 MG US UP
  ...
END
MODEL 12 FRAC PRESDEP
  INIT FRAC PRES = -5.E10
  DENSITY TOLERANCE = 1.E-6
  PRESSURE TOLERANCE = 1.E+2
END
```

- Modules: material_libs/standard_models
  - frac_presdep.h, frac_presdep.C
- Physics: hydrodynamics

Table 86: Input Parameters for FRAC PRESDEP

Parameter Name	Type	Description
INIT FRAC PRES	*real*	Initial fracture pressure
DENSITY TOLERANCE	*real*	Density tolerance for convergence of iteration
PRESSURE TOLERANCE	*real*	Pressure tolerance for convergence of iteration
MAX NUM OF ITERATIONS	*real*	Maximum number of iteractions per cycle
CYCLES TO FAIL	*real*	Number of cycles to relax from the initial fracture pressure to zero
FAILURE INCREMENT	*real*	

Table 87: Registered Plot Variables for FRAC PRESDEP

Variable Name	Type	Mode	Description
FAILURE FRACTION	*real*	ioput	
FRACTURE PRESSURE	*real*	ioput	Current fracture pressure
DENSITY	*real*	ioput	Material density
ENERGY	*real*	ioput	Specific internal energy
PRESSURE	*real*	ioput	Pressure

3.9.9 Burn Models

3.9.9.1 PROGRAMMED BURN JWL

This model is nearly identical to the **JWL** equation of state model [22]. However, it has been modified to work only with the *PROGRAMMED BURN* option in ALEGRA. When using this programmed burn JWL equation of state model, it is also necessary to include the *PROGRAMMED BURN* input. This model is being superseded by the KEOS JWL model, which can be used with or without the *PROGRAMMED BURN* option.

- Modules: material_libs/standard_models
  - progburn_jwl.h, progburn_jwl.C
- Physics: hydrodynamics

Table 88: Input Parameters for PROGRAMMED BURN JWL

Parameter Name	Type	Description
RHO REF	*real*	Density in unreacted reference state
TREF	*real*	Temperature in unreacted reference state
E SHIFT	*real*	Arbitrary shift of reference energy (optional)
A	*real*	JWL parameter in units of problem
B	*real*	JWL parameter in units of problem
C	*real*	JWL parameter in units of problem
OMEGA	*real*	JWL parameter in units of problem
R1	*real*	JWL parameter in units of problem
R2	*real*	JWL parameter in units of problem
E0	*real*	JWL parameter in units of problem
PCJ	*real*	Chapman-Jouget pressure
DCJ	*real*	Chapman-Jouget detonation front velocity
TCJ	*real*	Chapman-Jouget temperature
PB MIN RHO	*real*	Defaults to 0.9*RHO_REF
PB TMAX	*real*	Defaults to maximum of 2.*TREF or 870 K

Table 89: Registered Plot Variables for PROGRAMMED BURN JWL

Variable Name	Type	Mode	Description
SPECIFIC HEAT VOL	*real*	Initialize	initialized from Cv parameter
DENSITY	*real*	ioput	Material density
ENERGY	*real*	ioput	Specific internal energy per unit mass
PRESSURE	*real*	ioput	Pressure
TEMPERATURE	*real*	output	Absolute temperature
SOUND SPEED	*real*	output	Bulk sound speed

Table 89: Registered Plot Variables for PROGRAMMED BURN JWL (Continued)

Variable Name	Type	Mode	Description
DPDRHO	*real*	output	derivative of pressure with respect to density

3.9.10Complete Material and Model Example

The following are three benchmark regression tests (soldyn_mat, burn_mat and detonation_point) that checkout all the material models. The first two Lagrangian problems consists of several (15 in solydyn_mat, 5 in burn_mat) independent blocks of material. Given an initial velocity, each block impacts a rigid boundary. The materials include those with a single equation of state model, those with combined models (which combine the equation of state and a specified yield model), and reactive burn models (which include both predefined materials in the EOS_data file and user-specified submodel examples). The third problem is an example of the older programmed burn model input.

```
$ ALEGRA INPUT SET FOR 15 BLOCKS, TEST OF MATERIAL MODELS
$ Block 1:  Ideal Gas
$ Block 2:  Jones-Wilkins-Lee (JWL)
$ Block 3:  Elastic-plastic with Generic EOS
$ Block 4:  Linear Elastic, Von Mises Yield, and Simple Radial Return
$ Block 5:  Soil and Crushable Foam
$ Block 6:  Power-law Mie-Gruniesen
$ Block 7:  Mie-Gruniesn Us-Up, Steinberg-Guinan-Lund Yield MIG Model
$  Block 8:  Mie-Gruniesn Us-Up, Johnson-Cook Viscoplastic Yield MIG Model
$  Block 9:  KEOS Sesame, Zerilli-Armstrong Viscoplastic Yield MIG Model
$  Block 10:  KEOS MieGruniesen, Bamman-Chiesa-Johnson Viscoplastic Damage MIG Model
$ Block 11:  KEOS Ideal Gas
$ Block 12:  KEOS Jwl (Jones-Wilkins-Lee)
$ Block 13:  Elastic Plastic Power Law Hardening PRONTO Model
$ Block 14:  Power Law Hardening with Strength PRONTO Model
$ Block 15:  KEOS Sesame with P-alpha

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Job Control and I/O
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

title
  Mechanical Material Model Tests

term cyc 40
term time 8.0e-6

emit PLOT:   cycle interval 1
emit SCREEN: cycle interval 1$0

PLOT variable
  no underscores
  density: avg
  temperature: avg, as "TEMPERAT"
  pressure: avg
  sound speed: avg, as "SOUND_SP"
  stress: avg
  yield stress: avg, as "YIELD_ST"
  eqps: avg
end

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Physics Specification
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

solid dynamics
  max initial time step 1.e-9
  TIME STEP SCALE 0.8

$$ Aprepro code for generating block input
$$ Number of element blocks = { num_blocks=15 }
$$ Iterator: {i=0}, Velocity: {vel="-5.e4"}
$
${loop(num_blocks)}
$
$  initial block velocity: block  {++i}  x = {vel} $ cm/s
$
$  no displacement, nodeset {i}1 y
$  no displacement, nodeset {i}1 z
$  no displacement, nodeset {i}3 y
$  no displacement, nodeset {i}3 z
$  no displacement, nodeset {i}4 x
$
$  block {i}
$    lagrangian mesh
$    material {i}
$  end
$
${endloop}

$ Number of element blocks = 15
$ Iterator: 0, Velocity: -5.e4


  initial block velocity: block  1  x = -5.e4 $ cm/s

  no displacement, nodeset 11 y
  no displacement, nodeset 11 z
  no displacement, nodeset 13 y
  no displacement, nodeset 13 z
  no displacement, nodeset 14 x

  block 1
    lagrangian mesh
    material 1
  end


  initial block velocity: block  2  x = -5.e4 $ cm/s

  no displacement, nodeset 21 y
  no displacement, nodeset 21 z
  no displacement, nodeset 23 y
  no displacement, nodeset 23 z
  no displacement, nodeset 24 x

  block 2
    lagrangian mesh
    material 2
  end


  initial block velocity: block  3  x = -5.e4 $ cm/s

  no displacement, nodeset 31 y
  no displacement, nodeset 31 z
  no displacement, nodeset 33 y
  no displacement, nodeset 33 z
  no displacement, nodeset 34 x

  block 3
    lagrangian mesh
    material 3
  end


  initial block velocity: block  4  x = -5.e4 $ cm/s

  no displacement, nodeset 41 y
  no displacement, nodeset 41 z
  no displacement, nodeset 43 y
  no displacement, nodeset 43 z
  no displacement, nodeset 44 x

  block 4
    lagrangian mesh
    material 4
  end


  initial block velocity: block  5  x = -5.e4 $ cm/s

  no displacement, nodeset 51 y
  no displacement, nodeset 51 z
  no displacement, nodeset 53 y
  no displacement, nodeset 53 z
  no displacement, nodeset 54 x

  block 5
    lagrangian mesh
    material 5
  end


  initial block velocity: block  6  x = -5.e4 $ cm/s

  no displacement, nodeset 61 y
  no displacement, nodeset 61 z
  no displacement, nodeset 63 y
  no displacement, nodeset 63 z
  no displacement, nodeset 64 x

  block 6
```

```
    lagrangian mesh
    material 6
  end


  initial block velocity: block  7  x = -5.e4 $ cm/s

  no displacement, nodeset 71 y
  no displacement, nodeset 71 z
  no displacement, nodeset 73 y
  no displacement, nodeset 73 z
  no displacement, nodeset 74 x

  block 7
    lagrangian mesh
    material 7
  end


  initial block velocity: block  8  x = -5.e4 $ cm/s

  no displacement, nodeset 81 y
  no displacement, nodeset 81 z
  no displacement, nodeset 83 y
  no displacement, nodeset 83 z
  no displacement, nodeset 84 x

  block 8
    lagrangian mesh
    material 8
  end


  initial block velocity: block  9  x = -5.e4 $ cm/s

  no displacement, nodeset 91 y
  no displacement, nodeset 91 z
  no displacement, nodeset 93 y
  no displacement, nodeset 93 z
  no displacement, nodeset 94 x

  block 9
    lagrangian mesh
    material 9
  end


  initial block velocity: block  10  x = -5.e4 $ cm/s

  no displacement, nodeset 101 y
  no displacement, nodeset 101 z
  no displacement, nodeset 103 y
  no displacement, nodeset 103 z
  no displacement, nodeset 104 x

  block 10
    lagrangian mesh
    material 10
  end
```

```
initial block velocity: block  11  x = -5.e4 $ cm/s

no displacement, nodeset 111 y
no displacement, nodeset 111 z
no displacement, nodeset 113 y
no displacement, nodeset 113 z
no displacement, nodeset 114 x

block 11
  lagrangian mesh
  material 11
end


initial block velocity: block  12  x = -5.e4 $ cm/s

no displacement, nodeset 121 y
no displacement, nodeset 121 z
no displacement, nodeset 123 y
no displacement, nodeset 123 z
no displacement, nodeset 124 x

block 12
  lagrangian mesh
  material 12
end


initial block velocity: block  13  x = -5.e4 $ cm/s

no displacement, nodeset 131 y
no displacement, nodeset 131 z
no displacement, nodeset 133 y
no displacement, nodeset 133 z
no displacement, nodeset 134 x

block 13
  lagrangian mesh
  material 13
end


initial block velocity: block  14  x = -5.e4 $ cm/s

no displacement, nodeset 141 y
no displacement, nodeset 141 z
no displacement, nodeset 143 y
no displacement, nodeset 143 z
no displacement, nodeset 144 x

block 14
  lagrangian mesh
  material 14
end


initial block velocity: block  15  x = -5.e4 $ cm/s
```

```
no displacement, nodeset 151 y
no displacement, nodeset 151 z
no displacement, nodeset 153 y
no displacement, nodeset 153 z
no displacement, nodeset 154 x

block 15
  lagrangian mesh
  material 15
end


function 51  $ P-mu curve for soil material
$   mu           P
  0.00000    0.00000E+00
  0.00500    9.38215E+07
  0.01000    1.88356E+08
  0.01500    2.95625E+08
  0.02000    4.17546E+08
  0.02500    5.56035E+08
  0.03000    7.13009E+08
  0.03500    8.90386E+08
  0.04000    1.09008E+09
  0.04500    1.31402E+09
  0.05000    1.56410E+09
  0.05500    1.84226E+09
  0.06000    2.15041E+09
  0.06500    2.49046E+09
  0.07000    2.86433E+09
  0.07500    3.27395E+09
  0.08000    3.72122E+09
  0.08500    4.20806E+09
  0.09000    4.73640E+09
  0.09500    5.30814E+09
  0.10000    5.92520E+09
  0.10500    6.58951E+09
  0.11000    7.30298E+09
  0.11500    8.06753E+09
  0.12000    8.88507E+09
  0.12500    9.75751E+09
  0.13000    1.06868E+10
  0.13500    1.16748E+10
  0.14000    1.27235E+10
  0.14500    1.38348E+10
  0.15000    1.50105E+10
  0.15500    1.62527E+10
  0.16000    1.75632E+10
  0.16500    1.89439E+10
  0.17000    2.03968E+10
  0.17500    2.19238E+10
  0.18000    2.35268E+10
  0.18500    2.49044E+10
  0.19000    2.62820E+10
  0.19500    2.76596E+10
  0.20000    2.90372E+10
  0.20500    3.04148E+10
  0.21000    3.17924E+10
  0.21500    3.31699E+10
```

```
    0.22000    3.45475E+10
    0.22500    3.59251E+10
    0.23000    3.73027E+10
    0.23500    3.86803E+10
    0.24000    4.00579E+10
    0.24500    4.14355E+10
    0.25000    4.28130E+10
    0.25500    4.41906E+10
    0.26000    4.55682E+10
    0.26500    4.69458E+10
    0.27000    4.83234E+10
    0.27500    4.97010E+10
    0.28000    5.10786E+10
    0.28500    5.24562E+10
    0.29000    5.38337E+10
    0.29500    5.52113E+10
    0.30000    5.65889E+10
    0.30500    5.79665E+10
    0.31000    5.93441E+10
    0.31500    6.07217E+10
    0.32000    6.20993E+10
    0.32500    6.34768E+10
    0.33000    6.48544E+10
    0.33500    6.62320E+10
    0.34000    6.76096E+10
    0.34500    6.89872E+10
    0.35000    7.03648E+10
    0.35500    7.17424E+10
    0.36000    7.31200E+10
    0.36500    7.44975E+10
    0.37000    7.58751E+10
    0.37500    7.72527E+10
    0.38000    7.86303E+10
    0.38500    8.00079E+10
    0.39000    8.13855E+10
    0.39500    8.27631E+10
    0.40000    8.41406E+10
    0.40500    8.55182E+10
    0.41000    8.68958E+10
    0.41500    8.82734E+10
    0.42000    8.96510E+10
    0.42500    9.10286E+10
    0.43000    9.24062E+10
    0.43500    9.37837E+10
    0.44000    9.51613E+10
    0.44500    9.65389E+10
    0.45000    9.79165E+10
    0.45500    9.92941E+10
    0.46000    1.00672E+11
    0.46500    1.02049E+11
    0.47000    1.03427E+11
    0.47500    1.04804E+11
    0.48000    1.06182E+11
    0.48500    1.07560E+11
    0.49000    1.08937E+11
    0.49500    1.10315E+11
    0.50000    1.11692E+11
    0.50500    1.13070E+11
    0.51000    1.14448E+11
    0.51500    1.15825E+11
    0.52000    1.17203E+11
  end

end

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Material Specification
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$
$ Block 1:  Ideal Gas
$ Material: Helium
$

material  1, Helium
  model  11
end

model 11 ideal gas
  gamma   = 1.667
  rho ref = 1.6245e-04    $ g/cm^3
  tref    = 300.          $ K
  cv      = 3.116324e+07  $ erg/g/K
end

$
$ Block 2:  Jones-Wilkins-Lee (JWL)
$ Material: HNS Explosive
$

material  2, HNS Explosive
  model  21
end

model 21 jwl
  a       = 4.63e+12  $ a; dyne/cm^2
  b       = 8.873e+10 $ b; dyne/cm^2
  c       = 1.349e+10 $ c; dyne/cm^2
  omega   = 0.35
  r1      = 4.55
  r2      = 1.35
  rho ref = 1.65      $ rho0; g/cm^3
  e shift = 1.e+10
  pcj     = 2.15E+11  $ pcj; dyne/cm^2
  dcj     = 7.03e+05  $ dcj; cm/s
  tcj     = 4062.0    $ tcj; K
  tref    = 298.0     $ K
end

$
$ Block 3:  Elastic-plastic with Generic EOS
$ Material: OFHC Copper
$

material  3, OFHC Copper
  model  31
  model  32
end
```

```
model 31 elastic plastic
  youngs modulus    = 1.076e+12   $ dyne/cm^2
  poissons ratio    = 0.355
  yield stress      = 1.0e+08     $ fictitous
  hardening modulus = 5.0e+10     $ fictitous
  $yield stress      = 6.0e+09     $ dyne/cm^2
  $hardening modulus = 2.0e+09     $ dyne/cm^2
  beta              = 1.0 $ 0.5
end

model 32 generic eos
  rho ref         = 8.932       $ g/cm^3
  cv              = 3.924e+06   $ erg/g/K
  ref sound speed = 4.4468e+05  $ cm/s
  tref            = 298.0       $ K
end

$
$ Block 4:  Linear Elastic, Von Mises Yield, and Simple Radial Return
$ Material: OFHC Copper
$

material  4, OFHC Copper
  model  41
  model  42
  model  43
  model  44
end

model 41 linear elastic
  youngs modulus    = 1.076e+12   $ dyne/cm^2
  poissons ratio    = 0.355
end

model 42 von mises yield
  youngs modulus    = 1.076e+12   $ dyne/cm^2
  poissons ratio    = 0.355
  yield stress      = 1.0e+08     $ fictitous
  hardening modulus = 5.0e+10     $ fictitous
  $yield stress      = 6.0e+09     $ dyne/cm^2
  $hardening modulus = 2.0e+09     $ dyne/cm^2
end

model 43 simple radial return
end

model 44 generic eos
  rho ref         = 8.932       $ g/cm^3
  cv              = 3.924e+06   $ erg/g/K
  ref sound speed = 4.4468e+05 $ cm/s
  tref            = 298.0       $ K
end

$
$ Block 5:  Soil and Crushable Foam
$ Material: Backfill soil
$
```

```
material  5, Backfill soil
  model  51
  model  52
end

model 51 soil crushable foam
  shmod = 1.4e10
  bulk  = 2.76e11
  a0    = 398083.
  a1    = 0.019245
  a2    = -1.163e-10
  pfrac = -18595369.
  pmax  = 82738607.
  func table = 51
end

model 52 generic eos
  rho ref = 1.97
  cv = 1.e7
  ref sound speed = 3.74314e+05
  tref = 298.0       $ K
end

$
$ Block 6:  Power-law Mie-Gruniesen
$ Material: Funky Material
$

material  6
  model  61
end

model 61 mg power
  rho ref = 2.37 $ g/cm^3
  tref    = 273. $ K
  gamma0  = 1.
  cv      = 1.5e7 $ erg/g/K
  k0      = 1.96e11 $ dyne/cm^2
  k1      = -4.9
  k2      = 31.0
end

$
$ Block  7:  Mie-Gruniesn Us-Up, Steinberg-Guinan-Lund Yield MIG Model
$ Material: Tungsten
$

material  7, Tungsten
  model  71
end

model  71 cth elastic plastic
  eos model      =  72
  yield model    =  73
end

model  72 mg us up
  c0      = 4.029E5  $ 3.414e5  $ cm/s
  sl      = 1.237    $ 1.2
  gamma    = 1.54      $ 1.67
  rho ref = 19.30     $ 16.69
  cv       = 1.3530e+06 $ 1.400e06 $ erg/g/K
  tref     = 298.0
end

model  73 steinberg guinan lund
  R0ST  = 19.30
  TM0ST = 4519.61
  ATMST = 1.30
  GM0ST = 1.67
  AST   = 0.938E-12
  BST   = 1.3801e-04
  NST   = 0.13
  C1ST  = 0.71E+06
  C2ST  = 0.12E+06
  G0ST  = 1.60E+12
  BTST  = 7.70
  EIST  = 0.00
  YPST  = 1.6E+10
  UKST  = 0.31
  YSMST = 1.5E+10
  YAST  = 1.1E+10
  Y0ST  = 2.2E+10
  YMST  = 4.0E+10
end

$
$ Block  8:  Mie-Gruniesn Us-Up, Johnson-Cook Viscoplastic Yield MIG Model
$ Material: OFHC Copper
$

material  8
  model  81
end

model  81 cth elastic plastic
  eos model       =  82
  yield model     =  83
  poissons ratio = 0.3
end

model  82 mg us up
  c0        = 3.94e5   $cm/s
  sl        = 1.489
  gamma0    = 1.99
  rho ref   = 8.93    $ g/cm^3
  cv        = 3.929e06 $ erg/g/K
  pref      = 0.0
  tref      = 298.  $ K
end

model  83 johnson cook ep
  ajo = 8.970000E+08
  bjo = 2.918700E+09
  cjo = 2.500000E-02
  mjo = 1.090000E+00
  njo = 3.100000E-01
  tjo = 1380.718   $1.189813E-01 ev
```

```
end

$
$ Block  9:  KEOS Sesame, Zerilli-Armstrong Viscoplastic Yield MIG Model
$ Material: OFHC Copper
$

material  9, OFHC Copper
  model  91
end

model  91 cth elastic plastic
  eos model      =  92
  yield model    =  93
  poissons ratio = 0.3
end

model  92 keos sesame
  datafile = 'EOS_data'
  matlabel = 'COPPER'
end

model  93 zerilli armstrong
  C1ZE = 0.000000E+00
  C2ZE = 8.900000E+09
  C3ZE = 2.80e-3
  C4ZE = 1.15e-4
  C5ZE = 0.000000E+00
  AZE  = 6.500000E+08
  NZE  = 1.000000E+00
end

$
$ Block 10:  KEOS MieGruniesen, Bamman-Chiesa-Johnson Viscoplastic Damage MIG Model
$ Material: Aluminum
$

material 10, "Aluminum"
  model 101
  model 102
end

model 101 keos miegruneisen
  datafile = 'EOS_data'
  matlabel = '6061-T6_AL'
end

model 102 bammann chiesa johnson
  density        = 2.700e+00
  youngs modulus = 1.000e+12
  poisson ratio  = 3.300e-01
  temp           = 298.0
  hc             = 0.000e+00
  c1             = 1.200e+07
  c2             = 0.000e+00
  c3             = 2.500e+09
  c4             = 10.003000
  c5             = 1.000e-05
  c6             = 0.000e+00
  c7              = 0.000e+00
  c8              = 0.000e+00
  c9              = 0.000e+00
  c10             = 0.000e+00
  c11             = 0.000e+00
  c12             = 0.000e+00
  c13             = 1.800e-08
  c14             = 127.60000
  c15             = 2.800e+10
  c16             = 2.801e+07
  c17             = 0.000e+00
  c18             = 0.000e+00
  a1              = 0.000e+00
  a2              = 0.000e+00
  a3              = 0.000e+00
  a4              = 0.000e+00
  a5              = 0.000e+00
  a6              = 0.000e+00
  dex             = 1.000e+00
  d0              = 0.000e-00
  fs0             = 4.000e+09
  c19             = 4.000e-03
  c20             = 603.40000
end

$
$ Block 11:  KEOS Ideal Gas
$ Material: Helium
$

material 11, Helium
  model 111
end

model 111 keos ideal gas
  gm1      = 0.667
  r0       = 1.6245e-04    $ g/cm^3
  t0       = 298.          $ K
  cv       = 3.116324e+07  $ erg/g/K
end

$
$ Block 12:  KEOS Jwl (Jones-Wilkins-Lee)
$ Material: HNS Explosive
$

material 12, hns
  model 121
end

model 121 keos jwl
  r0       = 1.65      $ rho0; g/cm^3
  t0       = 298.0     $ K
  ag       = 4.6310e+12  $ a; dyne/cm^2
  bg       = 8.8730e+10 $ b; dyne/cm^2
  r1       = 4.550
  r2       = 1.350
  wg       = 0.35
  cv       = 0.          $calculated internally 9.651e6
  tcj     = 4061.575    $ tcj; K
  e0      = 0.  $.0745e12
  esft    = 0.   $1.e10
  dcj     = 7.03e+05  $ dcj; cm/s
  pcj     = 2.15E+11  $ pcj; dyne/cm^2
  brn     = 0.        $no heburn
end

$
$ Block 13:  Elastic Plastic Power Law Hardening PRONTO Model
$ Material:  approximately 6061-T6 aluminum
$

material 13, "approximately 6061-T6 Al"
  model 131
  model 132
end

model 131 ep power hard
  youngs modulus     = 6.89e11      $ dyne/cm^2
  poissons ratio     = 0.334
  yield stress       = 2.76e+09     $ dyne/cm^2
  hardening constant = 1.00e+09     $ est $ dyne/cm^2
  hardening exponent = 0.2          $ est
  luders strain      = 0.002        $ est
end

model 132 generic eos
  cv = 1.e7
  rho ref = 2.703
  ref sound speed = 6.19123e+05
  tref = 298.0       $ K
end

$
$ Block 14:  Power Law Hardening with Strength PRONTO Model
$ Material:  approximately 6061-T6 aluminum
$

material 14, "approximately 6061-T6 Al"
  model 141
  model 142
end

model 141 plh strength
  youngs modulus     = 6.89e11      $ dyne/cm^2
  poissons ratio     = 0.334
  yield stress       = 2.76e+09     $ dyne/cm^2
  hardening constant = 1.00e+09  $ est   $ dyne/cm^2
  hardening exponent = 0.2       $ est
  luders strain      = 0.002     $ est
  failure value      = 0.5       $ est
  decay constant     = 0.5       $ est
end

model 142 generic eos
  cv = 1.0e7
  rho ref = 2.703
  ref sound speed = 6.19123e+05
  tref = 298.0       $ K
end

$
$ Block 15:  KEOS Sesame (with palpha model)
$ Material:  Porous Aluminum
$

material  15, Porous Aluminum
  model  151
end

model 151 keos sesame
  neos = 4020
  feos = 'aneos'
  r0     = 2.785
  rp     = 2.15
  pe     = 4.5e8
end

exit

$ ALEGRA INPUT SET FOR 5 BLOCKS, TEST OF REACTIVE MATERIAL MODELS
$ Block 1:  KEOS Hvrb (History Variable Reactive Burn)
$ Block 2:  KEOS Arb  (Arrhenius Reactive Burn)
$ Block 3:  KEOS Igrb (Ignition and Growth Reactive Burn)
$ Block 4:  KEOS Ffrb (Forest Fire Reactive Burn)
$ Block 5:  KEOS Ptran (Phase Transition Reactive Burn)

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Job Control and I/O
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

title
  Reactive Burn Material Model Tests

term cyc 100
term time 8.0e-6

emit PLOT:   cycle interval  2
emit SCREEN: cycle interval 10

PLOT variable
  no underscores
  density: avg
  temperature: avg, as "TEMPERAT"
  pressure: avg
  sound speed: avg, as "SOUND_SP"
  EXT_REACTION: avg
end

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Physics Specification
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

solid dynamics
  max initial time step 1.e-15
  TIME STEP SCALE 0.5
```

```
$$ Aprepro code to generate block input
$$ Number of element blocks = { num_blocks=5 }
$$ Iterator: {i=0}, Velocity: {vel="-1.e5"}
$
${loop(num_blocks)}
$
$  initial block velocity: block  {++i}  x = {vel} $ cm/s
$
$  no displacement, nodeset {i}1 y
$  no displacement, nodeset {i}1 z
$  no displacement, nodeset {i}3 y
$  no displacement, nodeset {i}3 z
$  no displacement, nodeset {i}4 x
$
$  block {i}
$    lagrangian mesh
$    material {i}
$  end
$
${endloop}

$ Number of element blocks = 5
$ Iterator: 0, Velocity: -1.e5


  initial block velocity: block  1  x = -1.e5 $ cm/s

  no displacement, nodeset 11 y
  no displacement, nodeset 11 z
  no displacement, nodeset 13 y
  no displacement, nodeset 13 z
  no displacement, nodeset 14 x

  block 1
    lagrangian mesh
    material 1
  end


  initial block velocity: block  2  x = -1.e5 $ cm/s

  no displacement, nodeset 21 y
  no displacement, nodeset 21 z
  no displacement, nodeset 23 y
  no displacement, nodeset 23 z
  no displacement, nodeset 24 x

  block 2
    lagrangian mesh
    material 2
  end


  initial block velocity: block  3  x = -1.e5 $ cm/s

  no displacement, nodeset 31 y
  no displacement, nodeset 31 z
  no displacement, nodeset 33 y
  no displacement, nodeset 33 z
  no displacement, nodeset 34 x

  block 3
    lagrangian mesh
    material 3
  end


  initial block velocity: block  4  x = -1.e5 $ cm/s

  no displacement, nodeset 41 y
  no displacement, nodeset 41 z
  no displacement, nodeset 43 y
  no displacement, nodeset 43 z
  no displacement, nodeset 44 x

  block 4
    lagrangian mesh
    material 4
  end


  initial block velocity: block  5  x = -1.e5 $ cm/s

  no displacement, nodeset 51 y
  no displacement, nodeset 51 z
  no displacement, nodeset 53 y
  no displacement, nodeset 53 z
  no displacement, nodeset 54 x

  block 5
    lagrangian mesh
    material 5
  end

end

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Material Specification
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

$
$ Block 1:  KEOS Hvrb (History Variable Reactive Burn)
$ Material:  HMX
$

material 1, HMX
  model 11
end

model 11 keos hvrb
  matlabel = 'HMX'
end

$
$ Block 2:  KEOS Arb (Arrhenius Reactive Burn)
$ Material:  PETN
$
```

```
material 2, petn
  model 21
end

model 21 keos arb
  matlabel = 'PETN'
  $compare with PETN for the IGRB model to see model features
end

$
$ Block 3:  KEOS Igrb (Ignition and Growth Reactive Burn)
$ Material:  PETN
$

material 3, petn
  model 31
end

model 31 keos igrb
  matlabel = 'PETN'
  $compare with petn parameters from keos arb model
end

$
$ Block 4:  KEOS Ffrb (Forest Fire Reactive Burn)
$ Material:  pbx9404
$

material 4, pbx9404
  model 41
end
*
model 41 keos ffrb
 eosur modnumber =  42
 eosrp modnumber =  43
 X1 = 0.05
 P1 = 18.1567e10
 X2 = 0.50
 P2 = 4.4315e10
 FR = 1.0
* RMIN = 0.
* RMAX = 5.0
* TMAX = 1.e30
* VF   = 1.
* TF   = 1.
 ESFT = 0.
 PMIN = 1.e9
 PMAX = 3.53e11
 NRH = 1.
end

model 42, keos miegrun
  matlabel = 'pbx9404'
end

model 43, keos sesame
  matlabel = 'pbx9404_dp'
end
```

```
$
$ Block 5:  KEOS Ptran (Phase Transition Reactive Burn)
$ Material: Iron
$

* Iron, PTRAN model, phase 1--alpha, phase 2--epsilon.
* Note esft for epsilon used to include transition energy.
* Set BT=1.85E12 to match transition pressure to Rayleigh line.
* SET HF=1 to make transition irreversible.

material  5, iron
  model  51
end

model 51 keos ptran
  eosur modnumber = 52
  eosrp modnumber = 53
  pt=13.0E10
  bt=1.0E9
  at=-1.034082e7  $ -1.2E11/11604.5
  *ax=-4.0E10
end

model 52 keos miegrun
  r0=7.87
  t0=298.
  cs=4.6E5
  s1=1.46
  g0=1.7
  cv=4.601663e6   $5.34E10/11604.5
end

model 53 keos miegrun
  r0=8.29
  t0=298.
  cs=4.6E5
  s1=1.51
  g0=2.4
  cv=4.515490e+6   $5.24E10/11604.5
  esft=0.0727E10
end

exit

$------------------------------------BEGIN_QA------------------------------------
$ ID:         detonation_point
$ Title:      Detonation Test in 2D, Programmed Burn JWL Model
$ Category:   Regression
$ Physics:    comprehensive
$ Dimension:  2D
$ Owner:      Sharon Petney
$
$ Description:
$   This problem uses the older model for programmed burned
$   detonations, the model named PROGRAMMED BURN JWL.  It is kept
$   in the Benchmark for now for backward compatibility since the
$   input parameters for KEOS JWL (the preferred model, det-jwl.inp)
$   are different. Eventually this model will be discontinued.
$   This calculation has one block and is a simple example of
$   detonation propagation outside a purposely small burn radius.
$   A barrier is placed to test for circumvention.
$
$ References: SAND91-0344, CTH Reference Manual: The Equation of State Package"
$ Directory:  Benchmarks/Regression/2D/comprehensive
$ Tags:       ?
$ CVS: ?
$------------------------------------END_QA------------------------------------
TITLE
  Detonation point test problem for detonation propagation

termination cycle = 20

termination time = 1.5e-6

emit plot, cycle interval = 1

solid dynamics
  no displacement, nodeset 1, x
  no displacement, nodeset 2, y

  programmed burn
    material 1
      burn transition ratio 1.0
      burn front thickness 1.0
      detonation point
        x 0.0 y 0.0 at time 0.0, burn radius 0.2
  end

  diatoms
    package barrier
     material 2
     insert cyl
       ce1 0.,.6
       ce2 .6,.6
       radius .1
     endi
    endp
  enddia

  block 1
    material 1
    add diatom input
  end

end

material 1      $ JWL explosive
  model 1
end

model, 1, PROGRAMMED BURN JWL
  pb t max        = 870.0
  pb min rho      = 1.47
  a               = 3.712e12
  b               = 3.231e10
  c               = 1.045e10
  omega           = 0.30
  r1              = 4.15
  r2               = 0.95
  rho ref          = 1.630     $rho0; g/cm^3
  e shift          = 1.e+10
  pcj              = 2.10E+11 $pcj; dyne/cm^2
  dcj              = 6.93e+05 $dcj; cm/s
  tcj              = 3660.0   $tcj; K
  tref             = 298.0
end

material 2, "6061-T6 Aluminum"
  model 2
end

model 2 keos miegruneisen
  matlabel = '6061-T6_AL'
end

plot variable
  no underscores
  velocity
  temperature, as "TEMPERA"
  pressure, as "PRESSUR"
  artificial viscosity, as "ARTVIS"
  detonation time, as "DET TIME"
end

crt: off
exit
```

4.0 Diagnostics

To evaluate the progress of the calculation or analyze the results of a simulation, there are several diagnostic methods that are available within ALEGRA. The first method discussed in this section briefly describes an interactive menu through which calculation status information can be queried. The second method discussed in this section describes global diagnostic variables that are tallied by ALEGRA and written to the Exodus, PDS, and HISPLT databases (in addition to the registered plot variables listed with each material model). The third method describes the additional variables written to the HISPLT database, many of which are directly related to the tracer points specified in the *TRACER POINTS* section of the ALEGRA input file.

4.1 Interactive Menu

Through the interactive menu in ALEGRA, the user can query calculation status information, examine cell and node data for specified elements or nodes, change the stop time or cycle for the calculation, or force the calculation to write a restart file, an Exodus plot dump, or a HISPLT dump. The full menu can be obtained by typing *HELLO* into the standard input stream. A limited set of information (calculation status information) can also be obtained by typing '!' or 'status'. In subsequent releases of ALEGRA, the interactive menu will be enhanced to allow interactive changes to the remesh and remap parameters, options for meshview, and cell doctor capabilities.

For the interactive menu to be functional, the ***CRT*** keyword must be set to *ON* in the user input file. (The default is *CRT: ON*.)

4.2 Global Diagnostic Variables

Mass, momentum, and energy each obey a conservation law. ALEGRA tallies the mass, momentum and energy as a diagnostic. These tallies can be used as indicators of potential inaccuracies or errors in a simulation to the extent that the conserved quantities are not maintained. These global and material global variables are written to EXODUS, PDS, and HISPLT databases if the relevant physics is exercised by the ALEGRA calculation. The variable names are the same for all database types with the following exception. Since the HISPLT database limits the variable names to 16 characters, the root name is truncated to accommodate the component designators and the material ids while remaining with the 16-character string length limit. Material identifiers are added to the material global variable names (prefixed with "MAT") before they are written to the EXODUS or PDS database; the material identifier is appended to the variable name as ".n" in the plot request input file for the HISPLT database.The material identifier is the integer material id specified in the material section of the ALEGRA input file.

4.2.1 Time Step Tallies

The ALEGRA simulation timestep is a composite of the maximum stable timestep from numerous numerical considerations. Sometimes the timestep of an ALEGRA simulation seemingly may become unreasonably small. Usually there is a good explanation for such behavior. To assist in the diagnosis of small timesteps a set of tallies has been established that report the timesteps from each factor that can affect the overall timestep.

Table 90: Timestep Tallies for Hydrodynamics

GLOBAL VARIABLE NAME	Explanation
TM_STEP	The actual timestep use in the ALEGRA simulation. This timestep may be smaller than the minimum of the following timesteps if the TIME STEP SCALE is non-unity.
DT_HYDRO	This is the maximum stable timestep for all of HYDRODYNAMICS. It is a composite of the following timesteps.
DT_SOUND	Courant-limit timestep based upon the material sound speed computed as if this timestep acted alone.
DT_ELASTIC	Courant-like timestep based upon the elastic wave speed computed as if this timestep acted alone. The elastic wave speed is computed from the bulk and shear moduli, B and S, as $v = \sqrt{\left(B + \frac{4}{3}S\right)/\rho}$
DT_MATVEL	Timestep based upon the material velocity relative to the mesh computed as if this timestep acted alone. This timestep affects the simulation only if ADVECTION is enabled. The intent is to prevent material from advecting to cells other than a neighboring cell.
DT_ARTVIS	Timestep based upon ARTIFICIAL VISCOSITY limitations.
DT_VOLUME	Timestep based upon the allowed MAXIMUM VOLUME CHANGE for a mesh element in a single cycle.

4.2.2 Mass Tallies

Mass is a conserved quantity. The following table summarizes the mass tallies appearing in the ALEGRA output files.

Table 91: Mass Tallies for Region (All Physics Options)

GLOBAL VARIABLE NAME	Explanation
MASSTOT	Sum of all element masses.
MASSGAIN	Mass gain due to advection through the boundary and into the mesh. Since this tally represents a gain, it should be added to the initial mass in mass balance equations.
MASSLOSS	Mass loss due to advection through the boundary and out of the mesh and due to material discarded by the *Cell Doctor.* Since this tally represents a loss, it should be subtracted from the initial mass in mass balance equations.
NODEMASS	Sum of all node masses, should equal the element mass.
MASSERR	Mass conservation check: present mass - (initial mass + mass gain - mass loss) This check should be negligible compared to other mass tallies. Large errors may be due to remapping errors on under-resolved meshes. The user should try increasing the mesh resolution.
MAT_MASS	The global value of the mass of material “n,” where n is the integer material id.

4.2.3 Momentum Tallies

Momentum is a conserved quantity. At the present time ALEGRA does not separately tally the momentum sources or sinks due to various boundary conditions or other driving forces. The following table summarizes the momentum tallies appearing in the ALEGRA output files.

Table 92: Momentum Tallies for Dynamics and All Derived Physics Options

GLOBAL VARIABLE NAME	Explanation
XMOM *YMOM* *ZMOM*	The x, y, and z components of the total momentum for Cartesian simulations.

Table 92: Momentum Tallies for Dynamics and All Derived Physics Options

GLOBAL VARIABLE NAME	Explanation
RMOM *ZMOM* *THETAMOM*	The r, z, and θ components of the total momentum for cylindrically symmetric simulations.
MOM_UP_X *MOM_UP_Y* *MOM_UP_Z* *MOM_DOWN_X* *MOM_DOWN_Y* *MOM_DOWN_Z*	The up (positive axial direction) and down (negative axial direction) x, y, and z components of the total momentum for Cartesian simulations.
MAT_MOM_X *MAT_MOM_Y* *MAT_MOM_Z*	The x, y, and z components of the momentum of material “n,” where “n” is the integer material id.
MAT_MOM_U X *MAT_MOM_U_Y* *MAT_MOM_U_Z* *MAT_MOM_D X* *MAT_MOM_D_Y* *MAT_MOM_D_Z*	The up (positive axial direction) and down (negative axial direction) x, y, and z components of the momentum of material “n,” where “n” is the integer material id.

4.2.4 Energy Tallies

In many problems of interest it is often desirable to know the energy budget. How much energy is related to a given physical process? What is the value of the kinetic and internal energies? How much energy is supplied by a given source or is lost to a given sink? How fast does energy change from one form to another?

ALEGRA provides the user with a detailed set of energy and power tallies to answer such questions. Typically the HISPLT file will contain tallies at more frequent intervals compared to the EXODUS file (depending on the user specification -- see the *EMIT HISPLT* and *EMIT PLOT* commands) because it is smaller and does not contain mesh information or plot variables.

The following tables summarize the various energy and power tallies. The tallies are grouped by choice of the *PHYSICS* option. Extra global power tallies marked with an asterisk (*) are omitted from the output files unless the *DETAILED ENERGY TALLIES* keyword is specified.

Table 93: Energy Tallies for Region (All Physics Options)

GLOBAL VARIABLE NAME	Explanation
ETOT *PTOT**	Total of the kinetic and internal energies and rate of change.
EINT *PINT**	Sum of all element internal energies and rate of change.
EINTLOSS *EINTGAIN*	Sum of all element internal energy changes.
*EERROR** *PERROR**	Energy conservation check and rate of change. This check should be small compared to other energy tallies. Large errors may be due to too large a timestep, remapping errors on under-resolved meshes, or incompletely tallied sources, sinks, or boundary conditions. The user should try decreasing the timestep or increasing the mesh resolution.
MAT_EINT	The global internal energy of a material, where the material id designated by appending the material id to the variable name.
MAT_ETOT	The global total energy of a material, where the material id designated by appending the material id to the variable name.

Table 94: Energy Tallies for Dynamics (Hydrodynamics)

GLOBAL VARIABLE NAME	Explanation
EKIN *PKIN**	Kinetic energy and rate of change.
EVELBC *PVELBC**	Work done on the system by various kinematic boundary conditions and rate of change.
EGRAV *PGRAV**	Gravitational potential energy and rate of change. The tally is included only if a non-zero *GRAVITY* option is specified. Zero potential energy is defined to be at the origin.

Table 94: Energy Tallies for Dynamics (Hydrodynamics) (Continued)

GLOBAL VARIABLE NAME	Explanation
EINTGAIN *EINTLOSS*	Internal energy gain or loss due to advection through the boundary of the mesh and internal energy discarded by the *Cell Doctor.* Energy gain is plotted as a positive value and energy loss is plotted as a negative value.
EKINGAIN *EKINLOSS*	Kinetic energy gain or loss due to advection through the boundary of the mesh and kinetic energy discarded by the *Cell Doctor.* Energy gain is plotted as a positive value and energy loss is plotted as a negative value.
EK_UP_X *EK_UP_Y* *EK_UP_Z* *EK_DOWN_X* *EK_DOWN_Y* *EK_DOWN_Z*	Partial kinetic energies associated with each coordinate axis and direction (D = down, U = up).
MAT_EK	The global kinetic energy of a material, where the material id designated by appending the material id to the variable name. In EXODUS or PDS, the id is appended to the end of the variable name in the plotting database. For HISPLT, the material id is appended by the user in the plot request within the HISPLT input file.
MAT_EK_UP_X *MAT_EK_UP_Y* *MAT_EK_UP_Z* *MAT_EK_DOWN_X* *MAT_EK_DOWN_Y* *MAT_EK_DOWN_Z*	Partial material kinetic energies associated with each coordinate axis and direction (D = down, U = up). The material id is designated by appending the material id to the variable name. In EXODUS or PDS, the id is appended to the end of the variable name in the plotting database. For HISPLT, the material id is appended by the user in the plot request within the HISPLT input file.

4.2.5 Additional Diagnostic Variables

Additional variables are written to the EXODUS, PDS, and HISPLT databases. The grind time is useful for comparing the relative effort of a calculation per element, per cycle. This is calculated with and without the time required for reading and writing all input and output files. The CPU time is also provided, excluding the I/O time.

Table 95: Global Variables In Addition to Energy/Mass/Momentum Tallies

GLOBAL VARIABLE NAME	Explanation
GRIND	Grind time during the calculation, defined as the computation time divided by the product of the number of processors X number of elements X number of cycles.
CPUNOIO	CPU time ignoring the contribution of IO.
GRINDNOIO	Grind time ignoring the contribution of IO.

4.3 Additional HISPLT Database Variables

In addition to all global and material global variables listed in Section 4.2, the HISPLT database contains variables relevant to the tracer points. History variables calculated at tracer particle locations include those specific to the material models assigned to each material (see the HISPLT variable tables in the material ***MODEL*** input section) and general output variables listed below(**Table 96**).

Hisplt input files can be constructed by referring to these variable lists or by using the CATALOG input keyword in an initial run of HISPLT to obtain the list of variables specific to the database being read.

Table 96: Point History Variables

Variable Name	Explanation
REGION-ID	Only one region exists for ALEGRA at this time.
BLOCK-ID	The block id for the tracer particle.
ELEMENT-ID	The global element id within which the tracer particle resides.
ON-OFF	In a parallel calculation, number of processors that contain the coordinates of this tracer location. (This variable is currently inactivated but will be enabled in a future release.)
PSY-X, PSY-Y, PSY-Z	Position of the tracer within the local coordinate system of the element on which the tracer resides.
ELEMENT-VOLUME	Volume of the element within which the tracer particle resides.
ELEMENT-MASS	Mass of the element within which the tracer particle resides.

Table 96: Point History Variables (Continued)

Variable Name	Explanation
XPOSITION	X position of the tracer particle resides.
YPOSITION	Y position of the tracer particle resides.
ZPOSITION	Z position of the tracer particle resides.

Table 97: Global History Variables Specific to HISPLT

Variable Name	Relevant Model	Explanation
TIME	Hydrodynamics	The solution time in the calculation.
DT	Hydrodynamics	Time step size in the calculation.
CPU	Hydrodynamics	Accumulated computation time during the calculation.
CYCLE	Hydrodynamics	Accumulated number of cycles during the calculation.

5.0 Frequently Asked Questions

If I am familiar with running CTH, how do I run a similar problem with ALEGRA?

One difference in getting started with ALEGRA is that the mesh is generated outside of ALEGRA. This allows ALEGRA to use sophisticated unstructured meshes for Lagrangian and ALE calculations, as well as simple unstructured meshes similar to those used in CTH. The CTH mesh must be a orthogonal structured mesh, so it is a simple thing to generate the mesh in CTHGEN. In ALEGRA, the mesh is generated separately by another program. Examples of such programs include FASTQ (2D), a combination of FASTQ and GEN3D (3D) or from the CUBIT mesh generation tool (2D and 3D).

If the mesh is a structured mesh similar to a CTH mesh, you can insert materials into the mesh just like in CTH using the DIATOMS capability (see the above ALEGRA documentation for slight differences, most importantly in the keywords). If the mesh is an unstructured mesh, each “region” created by FASTQ corresponds to a “block” in ALEGRA. Individual blocks are assigned material numbers in the ALEGRA block input. (Note that one can still use DIATOM to insert material into an empty unstructured element block).

FASTQ and GEN3D are part of ACCESS. (Make sure your ALEGRA environment is specified in your path before the ACCESS paths, since at least one of the files has a duplicate name). Complete FASTQ and GEN3D documentation should be provided with your ACCESS distribution.

Mesh generation for ALEGRA consists of only the geometric discretization and the assignment of sideset and nodeset flags that may be referenced by boundary conditions or certain physics packages in the ALEGRA run. Whereas CTHGEN assigns materials and material properties prior to the CTH run, these functions are performed in a single ALEGRA run. Sample input files for both mesh generation and ALEGRA can be found in the Benchmark directories.

What does “Segmentation violation” mean?

This is the message used by the UNIX operating system to tell you that your program has tried to access memory that has not been assigned to it. It is *always* the result of a bug if you see it while running ALEGRA. Such bugs should be reported to the development team promptly with an associated sample input file and mesh.

How seriously should I take warnings?

A warning indicates a condition when initializing or running the code that one of the ALEGRA code developers thinks is suspicious but not necessarily wrong or perhaps not worth stopping the calculation since it may still be possible to get useful results. Since ALEGRA is very good at solving a discrete representation of your physics equations, but not very good at evaluating the results, it only notices the most obvious problems, such as negative temperatures. Warnings should therefore be taken seriously and investigated. However, after issuing a warning, ALEGRA will continue the calculation on the assumption that the problem is transitory and will not spoil the calculation as a whole.

My calculation reports an "element inversion." What does this mean?

ALEGRA has detected an element with a negative volume, indicating that the element has turned inside out. If this occurs right away, it may simply mean that one of the initial conditions you specified is causing an element to collapse before it has a chance to respond. This can often be corrected by specifying a smaller initial time step (see **Section 3.4.5.2 on page 84**.)

Element inversions later in a calculation usually arise from one of two causes. If the element inversion is preceded by a sharp drop in the time step, it is usually the result of mesh tangling. Susceptibility to such tangling is a weakness of the quadrilateral or hexahedral elements used in most ALEGRA calculations. The usual way to fix this is to identify the portion of the mesh that is tangling and make it part of an Eulerian or ALE element block. (Using a triangular or tetrahedral mesh is usually NOT a good way to fix the problem, since these elements are numerically very stiff.)

Abrupt element inversion (with no warning signs preceding it) usually indicates either that a large amount of energy has suddenly been deposited in an element, or a material in the element has run off the bounds of an equation of state table. The CLIP option for the Kerley SESAME equation of state can be useful (see **Section 3.9.3.8 on page 181**.) If no explanation for this behavior can be found, it may indicate a program bug that should be reported to the development team.

The time step has become so small that the calculation is going nowhere. What do I do?

Most of the problems that lead to element inversion can also lead to a sharp drop in the time step. In addition, a very hot material in an element can give rise to a large sound speed and a very small stable time step. The Cell Doctor package (see **Section 3.7.4.2 on page 132**) can detect and remove tiny volumes of very hot material that sometimes are left in an otherwise cold or empty cell by the advection package.

ALEGRA is also known to use a very conservative time step calculation. There are several user-controllable features that can affect the time step calculation. One is the *MINIMUM ELEMENT SIDE TIMESTEP CONTROL* (see Section 3.7.4.5 on page 134) which controls the calculation of the characteristic length for the element used in the time step calcuylation. Another parameter is the *MAXIMUM VOLUME CHANGE* (see Section 3.7.4.4 on page 134) value which can often cause ALEGRA to use very small time steps in Eulerian calculations. Finally, the *TIME STEP SCALE* (see Section 3.4.5.6 on page 85) parameter controls the overall multiplier that ALEGRA applies to the computed time step to determine what value to actually use in the calculation.

I am using ALE in my calculation and I get an almost immeadiate abort with a message *in the output that says*

> Error: Dynamics::Determine_Volume_Fluxes()

> Element = 864 Volume Flux = 1.39057e-11 is greater than the volume

> = 5.1859e-12

What's happening here is that ALEGRA's idea of a "good" mesh and the mesh generator's idea of good do not agree, so ALEGRA rezone is doing it's own thing. In the course of doing that, it is moving the nodes too far in one step and resulting in an "overflux" where the amount of "space" moved out of an element exceeds the amount there to start. What needs to happen is to let ALEGRA do it's thing but do it more gradually. There are inputs in Domain called

```
initial remesh movement limiter real
remesh movement ratio real
remesh movement limiter real
```

The first 2 set an initial limit multiplicative factor on the movement of any node and then a growth factor that is applied every remesh pass (usually remesh is done 10 times every cycle). The last value is an ultimate limit to which the remesh limiter will grow (has to be <= 1.0). So what you could do is put in the domain section:

```
initial remesh movement limiter  0.10
remesh movement ratio 1.05
remesh movement limiter 1.0
```

so the limit factor will start at 10%, grow by 5% per application and stop at 1.0.

My run is stopping before it ever gets to the first cycle print. I have no idea what is going wrong. What information should I gather to make it easier to find out what is going on?

Use the Debug print capability to help locate whare the code is dying. As described in **Section 3.3.2.2 on page 63**, the “location” debug flag will produce a trail on standard out that will help us track down where the code is dying. If you are going to send us a problem for us to take a look at, we usually need the .inp and any file used to generate the .gen file: FASTQ, GEN3D input files or CUBIT journal files. Any special input files, like volume fraction files for DIATOM, are also needed.

My run aborts before anything seems to happen. The screen message says to look at the .out file but all I see in that file is the following.

```
Error: Volume_Plot_Expression::Set_Up
       Cannot find a material variable named VELOCITY. See the
*.out file for valid variable names.
Error: Region::Set_Up
        Unable to set up plot database
```

You may see this occur for variables other than VELOCITY. In you input file, the .inp file, you have a plot variable specified with the “:avg” modifier after the name. This should tell ALEGRA to use a volume-weighted average of the value of this quantitiy over all materials in an element and report one value for the element, as opposed to reporting one value for each material in the element. This is described in **Section 3.3.2.11 on page 67**. The problem is that the variable, VELOCITY in this particular case, is not a “material” variable. So remove the “:avg” modifier and your run will proceed.

Index

Table of Contents
Index

1.0 References

[1] S. W. Attaway, *et al*., 1996, *"Parallel Contact Detection Algorithm for Transient Solid Dynamics Simulations Using PRONTO3D,"* Symposium on Development, Validation, and Application of Inelastic Methods for Structural Analysis and Design, 1996 ASME International Mechanical Engineering Congress and Exposition (MECE96), Atlanta, GA.

[2] T. Belytschko, J. Ong, W. Liu, and J. Kennedy, 1984, "*Computer Methods in Applied Mechanics and Engineering"*, Vol 43, pp. 251-276.

[3] D. J. Benson, 1989, "*Vectorizing the Right-Hand Side Assembly in an Explicit Finite Element Program,*" Computer Methods in Applied Mechanics and Engineering, Vol 73, pp. 147-152.

[4] D. J. Benson, 1991, "*A new two-dimensional flux-limited shock viscosity for impact calculations,"* Computer Methods in Applied Mechanics and Engineering, Vol 93, pp. 39-95.

[5] V. L. Bergman, 1991, "*Transient Dynamic Analysis of Plates and Shells with PRONTO 3D,"* SAND91-1182, Sandia National Laboratories, Albuquerque, NM.

[6] T. D. Blacker, 1988, "*FASTQ Users Manual Version 1.2*," SAND88-1326, Sandia National Laboratories, Albuquerque, NM.

[7] R. M. Brannon and M. K. Wong, 1996, "*MIG Version 0.0 Model Interface Guidelines: Rules to Accelerate Installation of Numberical Models Into Any Compliant Parent Code*," SAND96-2000, Sandia National Laboratories, Albuquerque, NM.

[8] A. Breckenridge, 1998, "*Teraflop Visualization*", Virtual Environments '98, Stuttgart, Germany.

[9] K. H. Brown, R. M. Summers, M. W. Glass, A. S. Gullerud, M. W. Heinstein, R. E. Jones, "*ACME Algorithms for Contact in a Multiphysics Environment API Version 1.0*," SAND2001-3318, Sandia National Laboratories, Albuquerque, NM.

[10] K. G. Budge, 1991, "*PHYSLIB: A C++ Tensor Class Library*," SAND91-1752. Sandia National Laboratories, Albuquerque, NM.

[11] K. G. Budge and J. S. Peery, 1993, "*RHALE: A MMALE Shock Physics Code Written in C++,*" International Journal of Impact Engineering, Vol. 14, pp. 107-120.

[12] J. Chakrabarty, 1987, "*Theory of Plasticity,"* McGraw-Hill Book Company, New York.

[13] B. M. Dobratz and P. C. Crawford, January 1985, "*LLNL Explosives Handbook*," UCRL-52997, Lawrence Livermore National Laboratory, Livermore, California.

[14] D. S. Drumheller, 1982, "*On the Dynamical Response of Particulate-loaded Materials. II. A Theory with Application to Alumina Particles in an Epoxy Matrix*," Journal of Applied Physics, Vol. 53(2), pp. 957-969.

[15] M. A. Ellis and B. Stroustrup, 1990, "*The Annotated C++ Reference Manual.*" Reading, MA: Addison-Wesley Publishing Company.

[16] D. P. Flanagan and T. Belytschko, 1981, *"A Uniform Strain Hexahedron and Quadrilateral with Orthogonal Hourglass Control,"* International Journal for Numerical Methods in Engineering, Vol. 17, pp. 679-706.

[17] A. P. Gilkey and J. H. Glick, 1989, *BLOT - "A Mesh and Curve Plot Program for the Output of a Finite Element Analysis*," SAND88-1432, Sandia National Laboratories, Albuquerque, NM.

[18] A. P. Gilkey and G. D. Sjaardema, 1989, "*GEN3D: A GENESIS Database 2D to 3D Transformation Program*," SAND89-0485, Sandia National Laboratories, Albuquerque, NM.

[19] B. Hendrickson and R. Leland, 1993, "*The Chaco User's Guide Version 1.0*," SAND93-2339, Sandia National Laboratories, Albuquerque, NM.

[20] G. L. Hennigan, M. St. John, and J. N. Shadid, May 1998, *"NEMESIS I: A Set of Functions for Describing Unstructured Finite-Element Data on Parallel Computers,"* Sandia National Laboratories, Albuquerque, NM.

[21] E. S. Hertel, Jr., R. L. Bell, M. G. Elrick, A. V. Farnsworth, G. I. Kerley, J. M. McGlaun, S. V. Petney, S. A. Silling, P. A. Taylor, and L. Yarrington, 26-30 July 1993, *"CTH: A Software Family for Multi-Dimensional Shock Physics Analysis,"* Proceedings of the 19th International Symposium on Shock Waves Vol. I, edited by R. Brun and L. D. Dumitrescu, Marseille, France

[22] G. I. Kerley, 1991, "*CTH Reference Manual: The Equation of State Package,"* SAND91-0344, Sandia National Laboratories, Albuquerque, NM.

[23] G. I. Kerley, 1998. "*CTH Reference Manual: The Equation of State Package",* SAND98-0947, Sandia National Laboratories, Albuquerque, NM.

[24] G. I. Kerley, March 1999, "*Recent Improvements to the CTH EOS Package,"* KPS99-1, Kerley Publishing Services.

[25] G. I. Kerley, 1992, "*CTH Equation of State Package: Porosity and Reactive Burn Models,"* SAND92-0553, Sandia National Laboratories, Albuquerque, NM.

[26] L. G. Margolin, 1988, "*A Centered Artificial Viscosity for Cells with Large Aspect Ratios*," UCRL-53882, Lawrence Livermore National Laboratory, Livermore, CA.

[27] L. G. Margolin and J. J. Pyun, 1987, "*A Method for Treating Hourglass Patterns*," Proceedings of the Fifth International Conference on Numerical Methods in Laminar and Turbulent Flow, Montreal, Canada.

[28] J. M. McGlaun, 1991, "*CTH Reference Manual: Cell Thermodynamics,"* SAND91-0002, Sandia National Laboratories, Albuquerque, NM.

[29] J. M. McGlaun, S. L. Thompson, and M. G. Elrick, 1989, "*A brief description of the three-dimensional shock wave physics code CTH*," SAND89-0607, Sandia National Laboratories, Albuquerque, NM.

[30] J. S. Peery and D. E. Carroll, 1999, "ALE *Remap Algorithms in ALEGRA*," Unpublished technical report, Sandia National Laboratories, Albuquerque, NM.

[31] E. Ranke, C. Katz, and H. Werner, 1983, "*International Journal for Numerical Methods in Engineering*, Vol. 19, pp. 1771-1782.

[32] J. N. Reddy, 1984, "*An Introduction to the Finite Element Method*," McGraw-Hill, Inc., New York.

[33] M. H. Rice, R. G. McQueen, and J. M. Walsh, 1958. *"Compression of Solids by Strong Shock Waves",* in Solid State Physics, Vol. 6*,* Advances in Research and Applications, edited by F. Seitz and D. Turnbull, Academic Press, New York.

[34] S. A. Silling, 1991, "*CTH Reference Manual: Viscoplastic Models*," SAND91-0292, Sandia National Laboratories, Albuquerque, NM.

[35] S. A. Silling, 1997, "Brittle Failure Kinetics Model for Concrete," Structures Under Extreme Loadng Conditions, PVP Vol. 351, ASME, pp. 263-268.

[36] G. D. Sjaardema, 1992, "*GJOIN: A Program for Merging Two or More GENESIS Databases*," SAND92-2290, Sandia National Laboratories, Albuquerque, NM.

[37] G. D. Sjaardema, 1992, "APREPRO: An Algebraic Proprocessor for Parameterizing Finite element Analyses," SAND92-2291,Sandia National Laboratories, Albuquerque, NM.

[38] G. D. Sjaardema, *et al*., 1994, "*CUBIT Mesh Generation Environment, Vol. 2: Developers Manual*," SAND94-1101, Sandia National Laboratories, Albuquerque, NM.

[39] R. M. Summers, *et al.*, 1997, "*Recent progress in ALEGRA development and application to ballistic impacts,"* International Journal of Impact Engineering, Vol. 20, pp. 779-788.

[40] L. M. Taylor and D. P. Flanagan, 1987a, *PRONTO-2D: "A Two-Dimensional Transient Solid Dynamics Program*," SAND86-0594, Sandia National Laboratories, Albuquerque, NM.

[41] L.M. Taylor and D. P. Flanagan, 1987b, "*PRONTO3D: A Three-Dimensional Transient Solid Dynamics Program,"* SAND87-1912*,* Sandia National Laboratories, Albuquerque, NM.

[42] P. A. Taylor, 1992, "*CTH Reference Manual: The Steinberg-Guinan-Lund Viscoplastic Model,"* SAND92-0716.

[43] P. A. Taylor, 1996, "*CTH Reference Manual: The Bammann-Chiesa-Johnson Viscoplastic/ Damage Model,"* SAND96-1626, Sandia National Laboratories, Albuquerque, NM.

[44] S. L. Thompson and L. N. Kmetyk, 1991, "*HISPLT, A Time-History Graphics Postprocessor, Users' Guide*," SAND91-1767, Sandia National Laboratories, Albuquerque, NM.

[45] J. M. Walsh and R. H. Christian, 1955, Phys. Rev., Vol. 97, pp. 1544.

[46] M. K. Wong, J. R. Weatherby, C. D. Turner, A. C. Robinson, T. A. Haill, D. E. Carroll, "Physics Applications in the ALEGRA Framework," First MIT Conference on Computational Fluid and Solid Mechanics, June, 2001.

Distribution

EXTERNAL DISTRIBUTION

Almquist, Brian
Commander Indian Head Division
NSWC
Code 420 (Brian Almquist), Bldg. 301
Indian Head, MD 20640-5035

Anderson, Charles E.
Southwest Research Institute
P. O. Drawer 28510
San Antonio, TX 78228-0510

Ari, Nasit Dr.
Kaman Sciences Corporation
P. O. Box 7463
Colorado Springs, CO 80933

Baker, Ernest L.
U. S. Army ARDEC
SMCAR-AEE-WW
Pictinny Arsenal, NJ 07806-5000

Becker, J. D.
Engineering & Research Dvl.Center
CEERD-IH-N
3903 Halls Ferry Road
Vicksburg, MS 39180-6199

Bilyk, Stephan
USARL
AMSRL-WT-TD
APG, MD 21005-5066

Calcaterra, Jeffrey R.
Building 63
1901 Tenth Street
Wright-Patterson AFB, OH 45433-7605

Chin, Chuck
U. S. Army ARDEC
SMCAR-AEE-WW -
Bldg. 3022
Picatinny Arsenal, NJ 07806-5000

Ding, Jow L.
Institute for Shock Physics
Washington State University
P.O. Box 642814
Pullman, WA 99164-2814

Bobby Doney
USARL
AMSRL-WT-TA
APG, MD, 21005-5066

Fermen-Coker, M
U.S. Army Research Laboratory
AMSRL-WT-TC
APG, MD 21005-5066

Filbey, Gordon L., Jr.
AMSRL-WT-TA
APG, MD 21005-5066

Grady, Dennis E.
Applied Research Associates, Inc.
4300 San Mateo Blvd. NE
Suite A-220
Albuquerque, NM 87110

Haque, Aamer
U. S. Deptartment of Energy
MS DP-153
1000 Independence Ave. S. W.
Washington, DC 20585

Huang, Y.
AMSRL-WT-TD
APG, MD 21005-5066

Johnson, Gordon R.
Alliant Techsystems, Inc.
600 2nd Street NE (MN11-1614)
Hopkins, MN 55343

Kee, A. L.
Indian Head Division, NSWC
101 Strauss Ave.
Bldg 301
Indian Head, MD 20640-5035

Kimsey, Kent D.
AMSRL-WT-TC
APG, MD 21005-5066

Kingman, Pat
AMSRL-WT-TD
APG, MD 21005-5066

Landsberg, Alexandra M.
Indian Head Division, NSWC
101 Strauss Ave.
Bldg 301
Indian Head, MD 20640-5035

Littlefield, David
The University of Texas at Austin
Institute for Advanced Technology
4030-2 Braker Lane
Austin, TX 78759-5329

Mair, Hans U.
Institute for Defense Analysis
Operational Evaluation Division
1801 North Beauregard Street
Alexandria, VA 22311-1772

Mello, Frank
OSD/OT&E/LFT
1700 Defense Pentagon 1C730
Washington, DC 20301-1700

Moore, Timothy W.
PO Box 12273
Huntsville, AL 35815

Namburu, Raju
US ARL
AMSRL-CI-H
APG, MD 21005-5067

Poy, Stephen
Naval Surface Warfare Center
Carderock Division
Code 673
9500 MacArthur Blvd.
West Bethesda, MD 20817-5700

Rajendran, AP
U. S. ARL
SLCBR-TR-AR
APG, MD 21005-5066

Schoenfield, Scott
USARL
AMSRL-WM-TD
APG, MD 21005

Schraml, Stephen J.
U.S. Army Research Laboratory
AMSRL-WT-TC
APG, MD 21005-5066

Stecher, Frederick
Alliant TechSystems
MS: MN07-MW44
4700 Nathan Lane, North
Plymouth, MN 55442-2512

Starbuck, J. Michael
Oak Ridge National Laboratory
P.O. Box 2009,
MS-8048
Oak Ridge, TN 37831

Stults, Allen
ITT Industires Advanced Engineering & Sciences
600 Boulevard South, Suite 208
Huntsville, AL 35802-2104

Taleyarkhan, Rusi
Oak Ridge National Laboratory
MS 8058, P. O. Box 2009
Oak Ridge, TN 37831-8058

Vanden, K. J.
AFRL/MNAC
101 West Eglin Blvd.
Eglin AFB, FL 32542-6810

Walker, James
Southwest Research Institute
P. O. Drawer 28510
San Antonio, TX 78228-0510

Wilson, Leonard T.
NSWC
17320 Dahlgren Road
Dahlgren, VA 22448-5100

Los Alamos National Laboratory
Mail Station 5000
P.O. Box 1663
Los Alamos, NM 87545

Attn: L. G. Margolin, MS D413
Attn: J. E. Morel, MS D409
Attn: S. Rojas, MS P946

University of California
Lawrence Livermore National
Laboratory
7000 East Ave.
P.O. Box 808
Livermore, CA 94550

Attn: R. Christensen, MS L-035
Attn: E. Dube, MS-L-035
Attn: C. McMillan, MS L-035
Attn: M. Murphy, MS L-368
Attn: R. Tipton, MS L-035

Sandia Internal

MS:	Attention:
0819	E. A. Boucheron, 9231 (5)
0819	D. E. Carroll, 9231 (25)
0899	Technical Library,9616 (2)
0612	Review and Approval Desk, 9612 (1) for DOE/OSTI
9018	Central Technical Files, 8945-1 (1)