**SAND REPORT**

SAND2004-5996
Unlimited Release
Printed February 2005

# ALEGRA-HEDP: Version 4.6

Thomas A. Brunner, Kyle R. Cochrane, Christopher J. Garasi, Thomas A. Haill, Thomas A. Mehlhorn, Allen C. Robinson, and Randall M. Summers

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's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Sandia National Laboratories

SAND2004-5996
Unlimited Release
Printed February 2005

# ALEGRA-HEDP: Version 4.6

Thomas A. Brunner, Christopher J. Garasi, Thomas A. Haill,
Thomas A. Mehlhorn, and Kyle R. Cochrane
HEDP Theory and ICF Target Design

Allen C. Robinson and Randall M. Summers
Computational Physics Research & Development

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1186

**Abstract**

ALEGRA is an arbitrary Lagrangian-Eulerian finite element code that emphasizes large distortion and shock propagation in inviscid fluids and solids. This document describes user options for modeling resistive magnetohydrodynamics, thermal conduction, and radiation transport effects, and two material temperature physics.

# Acknowledgment

The authors wish to acknowledge the assistance of Bob Campbell, Dan Carroll, Mary Chen, Mike Desjarlais, Jeff Lawrence, Ray Lemke, Bryan Oliver, Steve Rosenthal, Erik Wemlinger, and especially Tim (”The Enchanter”) Trucano in inspecting and testing the HEDP physics options for the ALEGRA code and its predecessors (primarily SPARTAN, RHALE++, and RHALE).

Pavel Bochev has made important contributions to the description and understanding of the edge element discretization for the 3D magnetics. Jonathan Hu and Ray Tuminaro have been instrumental in the development and implementation of the algebraic multigrid software.

We would also like to acknowledge the support of the members of the NEVADA framework development team and the rest of the ALEGRA development team. Their perpetual assistance in areas of team leadership, architecture, configuration management, regression testing, Diatom insertion, remap and advection, and more, have been invaluable.

We would also like to acknowledge the support of the managers of the HEDP Theory & ICF Target Design and the Computational Physics Research & Development departments, both past and present. Their constancy of vision over many years of development has allowed this project to arrive at full fruition.

# Contents

# Figures

## Tables

## Nomenclature

| Term | Definition |
|---|---|
| `ALE` | Arbitrary Lagrangian Eulerian |
| `ALEGRA` | Arbitrary Lagrangian Eulerian General Research Application |
| `FLD` | Flux Limited Diffusion |
| `HEDP` | High Energy Density Physics |
| `ICF` | Inertial Confinement Fusion |
| `MHD` | MagnetoHydroDynamics |
| `MMALE` | Multi-Material ALE |
| `QMD` | Quantum Molecular Dynamics |
| `RHALE` | Robust Hydrodynamic Arbitrary Lagrangian Eulerian |
| `SMALE` | Single-Material ALE |
| `1T` | One Material Temperature |
| `2T` | Two Material (Ion and Electron) Temperatures |

# ALEGRA-HEDP: Version 4.6

## 1 Introduction to ALEGRA HEDP

This manual provides a supplement to the ALEGRA User's Manual [12] and follows the pattern set therein. This manual describes the additional features that support high energy density physics (HEDP) applications. These applications include Z pinch and inertial confinement fusion (ICF) simulations. This manual represents a prioritized summary of the ALEGRA HEDP input options. The manual is intended as a reference guide and not as a tutorial or self-contained instruction manual.

HEDP applications require a wide range of physics phenomena to be modeled. This includes hydrodynamics, solid mechanics, transient magnetics, thermal conduction, radiation transfer, and under extreme conditions, decoupled ion and electron temperatures (two temperatures).

The 2D version of ALEGRA HEDP supports 2D Cartesian (XY) MHD modeling [17, 18] in which the magnetic field $\vec{B} = (B_x, B_y)$ may be in the plane of the mesh and the current density $J_z$ is orthogonal to the plane, or else in which the magnetic field $B_z$ may be orthogonal to the plane of the mesh and the current density $\vec{J} = (J_x, J_y)$ is in the plane. It also supports 2D cylindrically symmetric (RZ) MHD modeling in which the magnetic field $B_\theta$ may be orthogonal to the plane of the mesh and the current density $\vec{J} = (J_r, J_z)$ is in the plane. The 2D code supports unstructured quadrilateral grids.

The 3D version of ALEGRA HEDP supports 3D Cartesian (XYZ) MHD modeling on fully unstructured grids. The 3D code implements a magnetic diffusion solution based on edge and face elements [6]. A discretely divergence free property is maintained both in the magnetics solve and during a constrained transport algorithm in the remap phase. The magnetic flux density is represented in terms of face elements with the element magnetic flux on faces as degrees of freedom. The 3D code supports unstructured hexahedral grids.

Self-consistent transfer of energy to and from external lumped element circuit equations may be modeled. Coupled circuit equations allow a set of differential-algebraic equations to be coupled to the transient magnetic or MHD simulations. This is useful in situations where the simulation is driven by a known voltage source (most magnetic boundary conditions require knowledge of the current) or where the dynamics of the system being modeled feeds-back into the electrical response of the external circuit. This feature is fully described in Section 6.6 on page 70.

The `TRANSIENT MAGNETICS` package uses the Aztec library [46] to solve the discrete partial differential equation that governs its physics. Aztec is an iterative library that greatly simplifies solving linear systems of equations. Aztec includes a number of Krylov iterative methods such as conjugate gradient (CG), generalized minimum residual (GMRES), and stabilized biconjugate gradient (BiCGSATB). An algebraic multigrid option [44] for the edge element diffusion formulation in 3D and node-centered 2D formulations are available. These features are fully described in Section 11 on page 105.

An implicit thermal transport modeling capability is available. The current discretization is based on a support operator technique implemented by Kent Budge which has fundamental unknowns as thermal fluxes on faces. The `THERMAL CONDUCTION` package also uses the Aztec library. No multigrid capability is available in this case, but it is generally unnecessary.

Significant progress has been made at Sandia with respect to conductivity modeling in the solid-liquid-vapor transitions regions. Mike Desjarlais has been the principal theoretical lead in this effort. The LMD models are currently seen as the preferred capabilities. The user is advised to stay abreast of these rapidly advancing development as results are highly dependent on proper models [13, 14].

Applications which require true `RADIATION` transport are supported by the ALEGRA HEDP code [10, 9, 8]. Several radiation transport options are available including single and multi-group `LINEARIZED DIFFUSION`, and implicit Monte Carlo (`KULL IMC`) transport.

A simple radiation emission model is included, as opposed to full radiation transport, for purposes of radiating excess energy in cases where the medium is optically thin and radiated energy immediately escapes the system. In this case, full radiation transport may not be necessary. The emission model accounts for reabsorption by reducing the emission rate by a factor equal to one minus the probability of escape.

Opacity models are available to support both the emission and radiation transport models. For one group emission, the XSN and tabular opacity models can be made more accurate at lower temperatures through the use of the `DYNAMIC INTEGRATION` option for determining group bounds and the `HAGEN RUBENS` option. See Section 12.7.2 on page 147 for details.

Finally for highly ionized plasma simulation with long ion-electron equilibration time, a two-material temperature option is available.

## 1.1 New for Version 4.6

This is the first official release of this manual.

# 2 General Input

## 2.1 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. This section repeats some of the syntax rules from the primary ALEGRA User's Manual [12]. For additional input syntax, such as common parameter constructs, see the ALEGRA User's Manual.

### 2.1.1 Keywords

A keyword is a short sequence of English words denoting some action or quantity. For example,

```
TITLE
RADIATION MAGNETOHYDRODYNAMICS CONDUCTION
LORENTZ FORCE DENSITY FLOOR
MATERIAL FRACTION FORCE LIMIT
```

are all examples of keywords.

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 `UPPER CASE`, while common grammatical constructs and numerical parameters whose values are supplied by the user will be shown in `lower case`. Optional keywords, constructs, or parameters will be enclosed in [square brackets]. Alternative choices for a keyword may be enclosed {curly braces} and separated by an `OR` symbol ( | ), as in {`ABC | DEF`}, meaning `ABC` or `DEF`.

### 2.1.2 Delimiters

Keywords may or may not require a numeric field or other grammatical construct to follow it. *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, colons, semicolons, equal signs, or newlines 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.

### 2.1.3 Comments

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

```
$$$$$$$$$$$$$$$$$$$$ material models $$$$$$$$$$$$$$$$$$$$

  material 1  "W"
    model = 11               $ EOS
    model = 12               $ ionization
    model = 13               $ electrical conductivity
    model = 14               $ opacity

    density             1.6230    $ kg/m^3
    ion temperature     1.0 [ev]  $ K (1 ev)
    electron temperature 1.0 [ev]  $ K (1 ev)
  end
```

# 3 Execution Control

Execution control refers to keywords that appear outside the `PHYSICS ... END` block. As such they are not specific to any particular physics such as `TRANSIENT MAGNETICS` or `THERMAL CONDUCTION`. Typically these include the `TITLE` and `UNITS` keywords, `START TIME` and `TERMINATION TIME`, various `EMIT PLOT` and `EMIT OUTPUT` commands, the `PLOT VARIABLES ... END` block, as well as `MATERIAL` and material `MODEL` input. This section provides a supplement to the ALEGRA User's Manual [12].

## 3.1 Job Initiation and Termination

### 3.1.1 Units

```
UNITS, {CONSISTENT | SI | CGS | CGSEV | GAUSSIAN}
```

Default units for ALEGRA are `CGS` units. However, the `UNITS` keyword may be used to change the default units to `SI`, `CGSEV`, or `GAUSSIAN` units. Systems of units can be a subject of some confusion. The source of the confusion is that the number of fundamental units and of the dimension of physical quantities is arbitrary [24, 25].

For mechanical quantities (density, velocity, acceleration, force, energy, etc.), all quantities can be expressed in terms of three fundamental units (time, length, and mass). Any simulations involving only these basic mechanical quantities can be run using any `CONSISTENT` set of units, as well as the other explicit sets of units.

Default temperatures are always Kelvin unless `CGSEV` units are specified. For high temperature applications where the temperature is above the thousands of Kelvin range, it is common to express the temperature in units of energy by combining Boltzmann's constant, $k_B$, with the temperature ($k_B T$ has dimensions of energy and 1 eV is equivalent to 11605.67 K). In the `CGSEV` units case, eV are the temperature units.

For `THERMAL CONDUCTION`, quantities such as the specific heat and the thermal conductivity are defined in terms of the fundamental mechanical units (time, length, and mass). Thus `THERMAL CONDUCTION` problems may be run using any `CONSISTENT` set of units, as well as the other explicit sets of units.

For `TRANSIENT MAGNETICS` a variety of electrical units may be associated with input and output quantities. This is accomplished by defining values for six units-dependent constants, $\kappa_1$ to $\kappa_6$, at simulation start up. These constants are defined in Table 10 on page 38.

For SI units, only one choice is available because the Ampere is defined to be a fundamental unit and the electrical quantities default to the standard SI units. Only SI is fully implemented at this time for 3D simulations. SI, CGS, and GAUSSIAN are in general supported in 2D TRANSIENT MAGNETICS simulations.

For CGS-related units, many choices are available in principle, however in ALEGRA MHD only two of these choices have been made available to the user. The default CGS units are also called Practical CGS units. Practical CGS units are a combination of SI and CGS units as found in Knoepfel's monograph [25]. Densities, lengths and velocities are expressed in CGS-like units of g/cm$^3$, cm and cm/s. Charge, currents and voltages are expressed in SI-like units of coulombs, amperes and volts. Current densities, electric fields and conductivities are expressed in mixed units of amperes/cm$^2$, volts/cm and ohm-cm. Magnetic fields are in gauss. The other CGS-related set of units is the GAUSSIAN system.

For RADIATION, quantities such as the speed of light, Planck's constant, Boltzmann's constant, and the absorption and scattering opacities are defined in terms of the fundamental mechanical units (time, length, and mass). Thus RADIATION problems may be run using either SI or CGS units.

The units associated with common variables are outlined in Table 1. Additional systems of units may be found in many textbooks such as Knoepfel's monograph [25] or the NRL Plasma Formulary [22].

## 3.2 I/O Control

### 3.2.1 Plot Variables

```
PLOT VARIABLES
   name [, conversion] [, AS "string"]
   name [, conversion] [, AS "string"]
END
```

This section describes additional PLOT VARIABLES available with this version of ALEGRA. Other basic plot variables for hydrodynamics, solid dynamics, and materials are listed in the ALEGRA User's Guide [12].

In Tables 2 through 4 the variable types are listed as being scalar, vector, or material.

**Table 1.** Dimensions associated with variables for various systems of units.

| Variable | SI | CGS | CGSEV | GAUSSIAN |
|---|---|---|---|---|
| $t$ , time | s | s | s | s |
| $\vec{x}$, position | m | cm | cm | cm |
| $\vec{v}$, velocity | m/s | cm/s | cm/s | cm/s |
| $m$ , mass | kg | g | g | g |
| $\rho$ , mass density | kg/m$^3$ | g/cm$^3$ | g/cm$^3$ | g/cm$^3$ |
| $\vec{F}$, force | N (Newton) | dyne | dyne | dyne |
| $E$ , energy | J (Joule) | erg | erg | erg |
| $e$ , specific internal energy | J/kg | erg/g | erg/g | erg/g |
| $T$ , temperature | K (Kelvin) | K | eV | K |
| $C_v$ , specific heat | J/(kg*K) | erg/(g*K) | erg/(g*eV) | erg/(g*K) |
| $q$ , particle charge | C (Coulomb) | C | C | esu |
| $\vec{B}$, magnetic induction | T (Tesla) | G (Gauss) | G (Gauss) | G (Gauss) |
| $\vec{J}$, current density | A/m$^2$ | A/cm$^2$ | A/cm$^2$ | statampere/cm$^2$ |
| $\vec{E}$, electric field | V/m | V/cm | V/cm | statvolt/cm |
| $\sigma$ , electrical conductivity | (Ohm-m)$^{-1}$ | (Ohm-cm)$^{-1}$ | (Ohm-cm)$^{-1}$ | (statohm-cm)$^{-1}$ |
| $k$ , thermal conductivity | J/(m*s*K) | erg/(cm*s*K) | erg/(cm*s*eV) | erg/(cm*s*K) |
| $E_r$ , rad energy density | J/m$^3$ | erg/cm$^3$ | erg/cm$^3$ | erg/cm$^3$ |
| $T_r$ , rad temperature | K | K | eV | K |
| $D$ , rad diffusion coeff | m | cm | cm | cm |
| $\tau_a$ , absorption opacity | m$^{-1}$ | cm$^{-1}$ | cm$^{-1}$ | cm$^{-1}$ |
| $\tau_s$ , scattering opacity | m$^{-1}$ | cm$^{-1}$ | cm$^{-1}$ | cm$^{-1}$ |

Scalar variables have only one component and may be located at the mesh cell (or element) centers or located at the mesh nodes (or vertices).

Vector variables have two or three components depending on dimensionality and may be located at the mesh cell (or element) centers or located at the mesh nodes (or vertices). In 3D Cartesian, vectors have three components. Each component appears individually in the output files and is labeled with a suffix of _X, _Y, or _Z. In 2D, vectors have only two components. Each component appears individually and is labeled with a suffix of _X or _Y in Cartesian geometry, or with a suffix of _R or _Z in cylindrical geometry. Vector components orthogonal to the mesh are still necessary for 2D simulations. These variables are assigned to their own scalar variable and are listed as such in Table 2.

Material variables are associated with material properties and generally are located at the mesh cell (or element) centers. In general, all `MATERIAL` variables for each `MODEL` may be plotted by listing the variable name (with or without underscores) within the `PLOT VARIABLES` keyword construct. These variables are listed in the material model ”Registered Plot Variables” tables for each model (see Table 25, Table 27, Table 31, Table 45, up through Table 49, for example.)

Table 2: Plot Variables for `TRANSIENT MAGNETICS`.

| Variable Name | Type | Description |
|---|---|---|
| `AZ` or `ATHETA` | scalar (2D) | $A_z$ or $A_\theta$ component of the vector potential orthogonal to the mesh for 2D magnetics calculations |
| `AJXB` | vector | Lorentz force for 3D/2D MHD calculations |
| `B` | vector | Node-based magnetic field for 3D/2D magnetics calculations |
| `BE` | vector | Element-centered magnetic field for 3D/2D magnetics calculations |
| `BZ` or `BTHETA` | scalar (2D) | Node-based $B_z$ or $B_\theta$ component of the magnetic field orthogonal to the mesh for 2D magnetics simulations |
| `BEZ` or `BETHETA` | scalar (2D) | Element-centered $B_z$ or $B_\theta$ component of the magnetic field orthogonal to the mesh for 2D magnetics calculations |
| `DIVB` | scalar (3D) | Divergence of the magnetic field for 3D magnetics calculations. Should always be exactly zero to round-off. |
| `PHI` | scalar (2D) | Node-based product of $rA_\theta$ for 2D cylindrical simulations using the `FIFE R SCALED` formulation |
| `PSI` | scalar (2D) | Node-based product of $rB_\theta$ for 2D cylindrical simulations using the `FIFE R SCALED` formulation |
| `E` | vector | Node-based electric field for 3D/2D magnetics calculations |
| `EZ` or `ETHETA` | scalar (2D) | Node-based $E_z$ or $E_\theta$ component of the electric field orthogonal to the mesh for 2D magnetics calculations |
| `J` | vector | Node-based current density for 3D/2D magnetics calculations |
| `JE` | vector | Element-centered current density for 3D/2D magnetics calculations |
| `JZ` or `JTHETA` | scalar (2D) | Node-based $J_z$ or $J_\theta$ component of the current density orthogonal to the mesh for 2D magnetics calculations |
| `JEZ` or `JETHETA` | scalar (2D) | Element-centered averaged current density orthogonal to the mesh for 2D magnetics calculations |
| `VVOL` | scalar | Vertex volume for magnetics calculations |
| `ECON` | material | Electrical conductivity |

Additional `TRANSIENT MAGNETICS` plot variables other than those listed in Table 2 are available. They are not listed in the table because they typically are not of use to the user.

These variables are particular to the details of solution formulations and are primarily of use in debugging the solution method.

**Table 3.** Plot Variables for `THERMAL CONDUCTION`.

| Variable Name | Type | Description |
|---|---|---|
| `THERMAL CON` | material | Thermal conductivity |
| `THERMAL_CON_PAR` | material | Thermal conductivity component parallel to B field |
| `THERMAL_CON_PERP` | material | Thermal conductivity component perpendicular to B field |

**Table 4.** Plot Variables for `EMISSION`.

| Variable Name | Type | Description |
|---|---|---|
| `EMITTED POWER` | scalar array | Radiation power (energy per time) emitted from each mesh element. Tallied by radiation energy group (gg identifier) and material (mm identifier). Tallies are also summed over energy groups and material, if there is more than one group or material. |
| `EMITTED_POWER` | | total power |
| `EMITTED_POWER_Mmm_Ggg` | | by group and material |
| `EMITTED_POWER_Ggg` | | summed over materials |
| `EMITTED_POWER_Mmm` | | summed over groups |

`TWO TEMPERATURE` doubles or triples the number of material variables available for several of the material properties. Unless electrons and ions are in equilibrium, there is no single material `TEMPERATURE`. Instead, this variable is replaced by separate `ELECTRON TEMPERATURE` and `ION TEMPERATURE`. For other variables, ion and electron components are available in addition to the standard variables for single temperature physics, For example, the material `PRESSURE` also has the components `ELECTRON PRESSURE` and `ION PRESSURE`. Table 6 lists the additional material variables present when `TWO TEMPERATURE` physics is enabled.

**Table 5.** Plot Variables for `RADIATION`

| Variable Name | Type | Description |
|---|---|---|
| `RAD ENERGY DENSITY` | scalar array | Node-centered radiation energy density for linearized diffusion. Element-centered radiation energy density for IMC. The energy density is an array dimensioned to the number of energy groups. |
| `RAD TEMPERATURE` | scalar | Element-centered radiation temperature in Kelvin. When the radiation is in equilibrium with the material, this should be the same as the material temperature. The radiation temperature is related to the energy density through $T_r = (\frac{1}{a}\sum_g E_{rg}]^{1/4}$ where $a$ is the black body constant. |
| `RAD DIFFUSION COEFFICIENT` | scalar array | Element-centered radiation diffusion coefficient. The diffusion coefficient is an array dimensioned to the number of energy groups. |
| `RAD ETA` | scalar | Element-centered value of $\eta$ used for time step control. See the radiation user's guide [10] for details. |
| `OPACITY_A` | material array | Element-centered mean absorption opacity. This array is dimensioned to the number of materials and energy groups (e.g., `OPACITY_A_mm_gg`). |
| `OPACITY_R` | material array | Element-centered Rosseland mean opacity. This array is dimensioned to the number of materials and energy groups (e.g., `OPACITY_R_mm_gg`). |

**Table 6.** Plot Variables for `TWO TEMPERATURE`

| Variable Name | Type | Description |
|---|---|---|
| `ENERGY CHANGE ELECTRONS` | material | Electron energy change for each material |
| `ENERGY CHANGE IONS` | material | Ion energy change for each material |
| `ELECTRON ENERGY` | material | Electron specific internal energy |
| `ION ENERGY` | material | Ion specific internal energy<br>When added together the result is the total specific internal energy. |
| `ELECTRON PRESSURE` | material | Electron pressure |
| `ION PRESSURE` | material | Ion pressure<br>When added together the result is the total material pressure. |
| `ELECTRON SPECIFIC HEAT VOL` | material | Electron specific heat at constant volume |
| `ION SPECIFIC HEAT VOL` | material | Ion specific heat at constant volume |
| `ELECTRON TEMPERATURE` | material | Electron temperature |
| `ION TEMPERATURE` | material | Ion temperature |
| `ELECTRON THERMAL CON` | material | Electron thermal conductivity |
| `ION THERMAL CON` | material | Ion thermal conductivity |

# 4 General Physics Input

The physics option keywords specify the physics module(s) to use for a calculation. HEDP related options are listed in Table 7. Many options are actually combinations of the physics modules implemented in ALEGRA. The preferred syntax is to use the name of the desired physics module, followed by nested blocks of keywords specific to the parent physics modules, and terminated with a corresponding `END` keyword. For example:

```
radiation magnetohydrodynamics conduction
  $ radmhdcon is a child class formed by combining
  $ the following 4 parent classes
  hydrodynamics
    no displacement, x, nodeset 1
    [other hydrodynamics keywords]
    ...
  end

  radiation: linearized diffusion
    [linearized diffusion keywords]
    ...
  end:end

  transient magnetics
    [transient magnetics keywords]
    ...
  end

  thermal conduction
    [thermal conduction keywords]
    ...
  end

  [radmhdcon specific keywords]
  ...
  [block keywords]
  ...
  [function keywords]
  ...
end
```

```
[other non-physics keywords]
...
```

Keywords pertaining to parent modules of a particular physics module are included in the problem specification by placing the keywords in an `END`-block introduced by the name of the parent module. Thus, a `NO DISPLACEMENT` keyword, which is specific to `HYDRODYNAMICS` (actually `DYNAMICS`), appears immediately within the `HYDRODYNAMICS ... END`-block.

This new syntax offers certain advantages to ALEGRA code development teams. For users of ALEGRA, it offers the advantage of grouping related keywords with the particular physics module to which they are relevant.

## 4.1 Physics Choices

Physics modules included in ALEGRA HEDP, along with their keywords and parent classes, are listed in Table 7. The parent classes that are listed refer to the specific input sections that are contained in the base ALEGRA User's Manual [12].

Table 7: Physics Options and Parent Classes.

| Physics Option | Parent Classes | Description |
|---|---|---|
| `HYDRODYNAMICS` | `ENERGETICS`<br>`MECHANICS`<br>`DYNAMICS` | Deformation with zero stress deviators. |
| `SOLID DYNAMICS` | `ENERGETICS`<br>`MECHANICS`<br>`DYNAMICS` | Deformation with nonzero stress deviators. |
| `TRANSIENT MAGNETICS` | `ENERGETICS` | Transient magnetics diffusion in non-moving media. |
| `THERMAL CONDUCTION` | `ENERGETICS` | Thermal conduction in non-moving media. |
| `RADIATION` | `ENERGETICS` | Radiation transport in non-moving media. |
| `MAGNETOHYDRODYNAMICS` | `TRANSIENT MAGNETICS`<br>`SOLID DYNAMICS` | Transient magnetics diffusion coupled with solid dynamics. |

*continued on next page*

*continued from previous page*

| Physics Option | Parent Classes | Description |
|---|---|---|
| `MAGNETOHYDRODYNAMICS CONDUCTION` | `TRANSIENT MAGNETICS` `SOLID DYNAMICS` `THERMAL CONDUCTION` | Transient magnetics diffusion coupled with solid dynamics and thermal conduction. |
| `HYDRODYNAMICS CONDUCTION` | `THERMAL CONDUCTION` `HYDRODYNAMICS` | Thermal conduction coupled to hydrodynamics. |
| `SOLID DYNAMICS CONDUCTION` | `THERMAL CONDUCTION` `SOLID DYNAMICS` | Thermal conduction coupled to solid dynamics. |
| `RADIATION HYDRODYNAMICS` | `RADIATION` `SOLID DYNAMICS` | Radiation transport coupled to solid dynamics. |
| `RADIATION MAGNETOHYDRODYNAMICS` | `RADIATION` `TRANSIENT MAGNETICS` `SOLID DYNAMICS` | Radiation transport coupled to transient magnetics and solid dynamics. |
| `RADIATION HYDRODYNAMICS CONDUCTION` | `RADIATION` `THERMAL CONDUCTION` `SOLID DYNAMICS` | Radiation transport and thermal conduction coupled to solid dynamics. |
| `RADIATION MAGNETOHYDRODYNAMICS CONDUCTION` | `RADIATION` `THERMAL CONDUCTION` `TRANSIENT MAGNETICS` `SOLID DYNAMICS` | Radiation transport and thermal conduction coupled to transient magnetics and solid dynamics. |

In addition to the various physics options, there are options available that modify or enhance the physics options. These modifiers are listed in Table 8.

**Table 8.** Physics Option Modifiers.

| Physics Option | Parent Class Modified | Description |
|---|---|---|
| `EMISSION` | `ENERGETICS` | Uses a radiation emission model, in lieu of full radiation transport. Applicable to all above physics options. |
| `TWO TEMPERATURE` | `ENERGETICS` | Enables separate ion and electron material temperatures. Should be used in conjunction with two temperature material models. Applicable to all above physics options. |

## 4.2 Block Input

```
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 specification of LAGRANGIAN, MMALE, SMALE, or EULERIAN initially sets the mesh movement for the block to these types. In particular for MMALE and SMALE meshes, additional physics based subkeywords have been defined to weight the node positions when remapping a particular block. MHD specific ALEGRA BLOCK subkeywords are described in Table 9. These subkeywords supplement the BLOCK subkeywords found in the primary ALEGRA User's Manual [12].

**Table 9.** BLOCK Remap Sub-Keywords.

| Sub-Keyword | Type | Description |
|---|---|---|
| B MAG TRIGGER<br>J MAG TRIGGER<br>B MAG GRADIENT TRIGGER<br>J MAG GRADIENT TRIGGER | real | Nodes will be flagged for remesh TRANSIENT MAGNETICS quantities.<br>Triggers for remesh controlled by when the magnitude of the vector quantity at the node exceeds the input threshold value. |
| B MAG WEIGHT<br>J MAG WEIGHT<br>B MAG GRADIENT WEIGHT<br>J MAG GRADIENT WEIGHT | real | Weights for remesh controlled by TRANSIENT MAGNETICS quantities.<br>The remesh algorithm will use the maximum of the nodal value or the user specified value to bias the Tipton remesh method. The algorithm will tend to pull mesh nodes into regions where the selected physics parameter (or its gradient) are greatest. |

## 4.3 Aztec Set

```
AZTEC SET int
```

This keyword may appear as many times as desired, but if more that one type of physics is specified, then this keyword MUST appear within the specific physics input sections. For example, `TRANSIENT MAGNETICS` and `THERMAL CONDUCTION` solvers typically use different `AZTEC` control sets, therefore one must specify different `AZTEC SET int` keywords in the specific physics input sections.

The default `AZTEC SET` is 0 which defaults to the conjugate gradient algorithm with symmetric diagonal scaling and other default options. Omission of the `AZTEC SET id` keyword from a physics input block that requires the use of Aztec will cause that physics to use the default set. The settings for this default set may be changed by including an `AZTEC SET 0 ... END` block in the input file. See Section 11 for a descriptions of possible options.

Example:

```
magnetohydrodynamics conduction
   conduction
      ...
      scale, 1.0e10  $ do not use explicit conduction time step
      $ no explicit AZTEC SET keyword
   end
   transient magnetics
      ...
      aztec set, 1
   end
end

aztec 0       $ do NOT use ML for thermal conduction control
   solver,    cg
   scaling,   sym_diag
   conv norm, rhs
   precond,   none
   output,    none
   tol,       1.e-8
   max iter,  5000
end

aztec 1       $ 2D Mag example for AZTEC/ML solver options
   solver,    cg
   scaling,   sym_diag $ default = sym_diag
   conv norm, rhs      $ default = r0
   tol,       1.e-8
   output,    none
   multilevel          $ These settings may be overkill
      fine sweeps      = 1
      fine smoother    = GAUSS SEIDEL
      coarse smoother  = lu
      interp algorithm = AGGREGATION
    end
end
```

# 5 Magnetohydrodynamics Input

This section provides a list of supplemental keywords relevant to the magnetohydrodynamics section of the ALEGRA User's Manual [12]. The set of single-fluid hydrodynamics equations solved by ALEGRA-MHD are:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \tag{1}$$

$$\rho \left( \frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla) \vec{v} \right) = -\nabla p + \left[ \rho_q \kappa_1 \vec{E} \right] + \kappa_2 \vec{J} \times \vec{B} \tag{2}$$

$$\rho \left( \frac{\partial e}{\partial t} + (\vec{v} \cdot \nabla) e \right) = -p (\nabla \cdot \vec{v}) - \nabla \cdot \vec{Q} - \left[ \kappa_1 \rho_q \vec{v} \cdot \vec{E} \right] + \kappa_1 \eta \vec{J} \cdot \vec{J} \tag{3}$$

where $\rho$ is the mass density, $\rho_q$ is the charge density, $\vec{v}$ is the velocity, $p$ is the scalar pressure (for simplicity we ignore tensor pressures), $e$ is the specific internal energy, $\vec{B}$ is the magnetic field, $\vec{E}$ is the electric field, $\vec{J}$ is the current density, $\vec{Q}$ is the heat flux, and $\eta$ is the resistivity.

Equations 1 to 3 utilize two of a set of unit dependent constants, $\kappa_1$ and $\kappa_2$. The sole purpose of these constants is to allow the equations to be valid for a variety of sets of units. These are described further in Section 6 on page 37. The equations show the coupling to magnetics through the Lorentz force, $\kappa_2 \vec{J} \times \vec{B}$, and Joule heating, $\kappa_1 \eta \vec{J} \cdot \vec{J}$, and to thermal conduction through the heat flux, $-\nabla \cdot \vec{Q}$. In the MHD limit, it is assumed that the plasma is charge neutral ($\rho_q \approx 0$), and high-frequency phenomena are ignored ($\frac{\partial \vec{D}}{\partial t}$ is neglected), therefore the terms and equations in [square brackets] are ignored.

## 5.1 Algorithm Control

```
MAGNETOHYDRODYNAMICS
   ...
   ... [hydrodynamics keywords] ...
   ...
   [DENSITY FLOOR real]
   [HYDRO FORCE DENSITY FLOOR real]
   [LORENTZ FORCE DENSITY FLOOR real]
   [MATERIAL FRACTION FORCE LIMIT real]
   ...
END
```

Due to numerical approximation, some physics options may compute external forces that are inordinately large compared to the material present in a mixed material/void cell, hence the user may want to limit the forces in this case. Limiting only the forces associated with that physics may introduce its own set of problems. The options listed here turn off all forces, both internal and external, and accelerations.

### 5.1.1 Density Floor

```
DENSITY FLOOR real (0.0)]
```

When using various density floors to limit hydrodynamic forces, Lorentz forces, etc., it is possible to reduce the material density far below any specified density floor. This can occur because one side of a cell may zero the forces, whereas the other side may not, and material may then be advected out of the cell. Having the density fall too far below a density floor may be detrimental, especially when it is desired to have the limited feature turn back on as densities would rise back above the density floor. This keyword forces ALEGRA to boost cell densities up to the specified `DENSITY FLOOR` when they fall below this value. If this option is employed, the user should monitor the global mass tallies `MASSTOT`, `NODEMASS`, and `MASSERR`, to ensure that too much mass is not added to the simulation.

### 5.1.2 Hydrodynamic Force Density Floor

```
HYDRO FORCE DENSITY FLOOR real (0.0) [POWER real (0.0)]
```

Problems can arise in the calculation of forces on vertices with very little mass. The effective accelerations can become much too large causing mesh elements to become inverted and the calculation to terminate. The `HYDRO FORCE DENSITY FLOOR` keyword conditionally or gradually turns off the nodal forces.

As an alternative to the `MATERIAL FRACTION FORCE LIMIT`, the `HYDRO FORCE DENSITY FLOOR` keyword allows the user to limit the force acting on materials based upon the nodal mass density. The densities of attached elements are averaged. If the average density is less than the specified density floor, the nodal force is limited, otherwise the force is not changed. The multiplier ($hfdf$) of the nodal force used when the average density ($\rho$) is less than the specified $\rho_{floor}$ is given by the following expression where $p$ is the specified `POWER`.

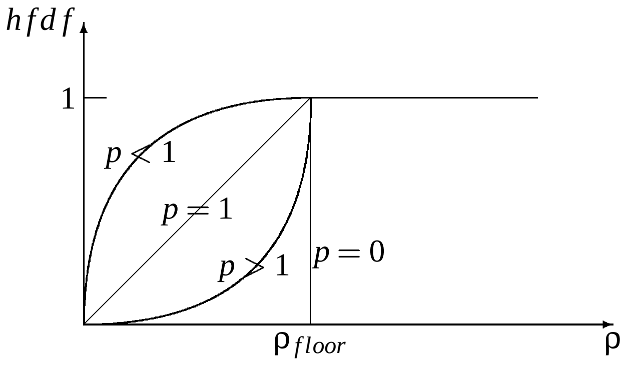

**Figure 1.** Hydrodynamic & Lorentz Force Density Floor Multiplier.

$$hfdf = \begin{cases} 0, & p = 0 \\ \left(\frac{\rho}{\rho_{floor}}\right)^p, & p > 0 \end{cases} \tag{4}$$

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 drops rapidly near 0. Graphically the multiplier can be displayed as in Figure 1.

Both the `HYDRO FORCE DENSITY FLOOR` and the `MATERIAL FRACTION FORCE LIMIT` may be specified simultaneously, but note that the `HYDRO FORCE DENSITY FLOOR` will supersede the `MATERIAL FRACTION FORCE LIMIT`. The `MATERIAL FRACTION FORCE LIMIT` will be ignored. Like `MATERIAL FRACTION FORCE LIMIT`, this keyword limits all forces, both internal (e.g., pressure) and external (e.g., gravity and Lorentz).

### 5.1.3 Lorentz Force Density Floor

```
LORENTZ FORCE DENSITY FLOOR real (0.0) [POWER real (0.0)]
```

Problems can arise in the calculation of Lorentz forces on vertices with very little mass. The effective accelerations can become much too large causing mesh elements to become inverted and the calculation to terminate. The `LORENTZ FORCE DENSITY FLOOR` keyword conditionally or gradually turns off the Lorentz forces.

As an alternative to the `MATERIAL FRACTION FORCE LIMIT`, the `LORENTZ FORCE DENSITY`

`FLOOR` keyword allows the user to limit the Lorentz force acting on nodes based upon the nodal mass density. The densities of attached elements are averaged. If the average density is less than the specified density floor, the nodal force is limited, otherwise the force is not changed. The multiplier ($lfdf$) of the nodal force used when the average density ($\rho$) is less than the specified $\rho_{floor}$ uses the same expression found for the hydrodynamic force limiter.

Both the `LORENTZ FORCE DENSITY FLOOR` and the `MATERIAL FRACTION FORCE LIMIT` may be specified simultaneously, but note that the `HYDRO FORCE DENSITY FLOOR` will supersede the `MATERIAL FRACTION FORCE LIMIT`. The `MATERIAL FRACTION FORCE LIMIT` will be ignored.

### 5.1.4 Material Fraction Force Limit

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

Problems can arise in the calculation of forces on vertices with very little mass. The effective accelerations can become much too large causing mesh elements to become inverted and the calculation to terminate. The `MATERIAL FRACTION FORCE LIMIT` keyword allows the user to limit the force acting on mixed material/void elements. This difficulty arises because 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., $matfrac = 1 - voidfrc$) 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.

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 the average material fraction ($matfrac$) is less than the specified $limit$ is given by the following expression where $p$ is the specified `POWER`.

$$mffl = \begin{cases} 0, & p = 0 \\ \left(\frac{matfrac}{limit}\right)^p, & p > 0 \end{cases} \tag{5}$$

For $p = 0$ the limiter is a step function, for $p = 1$ the limiter is linear, for $p > 1$ the

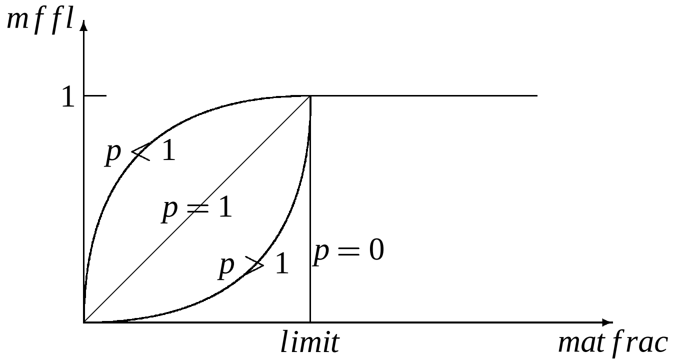

**Figure 2.** Material Fraction Force Limiter for MHD.

limiter drops rapidly near the threshold, and for $p < 1$ the limiter drops rapidly near 0. Graphically the multiplier can be displayed as in Figure 2.

Note that `TRANSIENT MAGNETICS` has the same `MATERIAL FRACTION FORCE LIMIT` keyword. The difference in usage is that in `TRANSIENT MAGNETICS` this keyword limits the Lorentz force only. In `HYDRODYNAMICS` this keyword limits all forces, both internal (e.g., pressure) and external (e.g., gravity and Lorentz).

# 6 Transient Magnetics Input

```
TRANSIENT MAGNETICS
   ...
   ... [transient magnetics keywords] ...
   ...
END
```

The `TRANSIENT MAGNETICS` input controls the magnetics capability in this version of ALEGRA. This allows for modeling electromagnetic effects in the quasi-neutral, quasi-static magnetic field approximation. The set of Maxwell's equations in the single-fluid limit are:

$$\nabla \times \vec{E} = -\kappa_3 \frac{\partial \vec{B}}{\partial t} \tag{6}$$

$$\nabla \times \vec{H} = \left[\kappa_4 \frac{\partial \vec{D}}{\partial t}\right] + \kappa_5 \vec{J} \tag{7}$$

$$\eta \vec{J} = \vec{E} + \kappa_3 \vec{v} \times \vec{B} \tag{8}$$

$$\left[\nabla \cdot \vec{D} = \kappa_6 \rho_q\right] \tag{9}$$

$$\nabla \cdot \vec{B} = 0 \tag{10}$$

$$\left[\vec{D} = \varepsilon \vec{E}\right] \tag{11}$$

$$\vec{B} = \mu \vec{H} \quad \text{or} \quad \vec{H} = \nu \vec{B} \tag{12}$$

where $\rho_q$ is the charge density, $\vec{v}$ is the velocity, $\vec{B}$ is the magnetic field, $\vec{E}$ is the electric field, $\vec{J}$ is the current density, $\eta$ is the resistivity, and $\varepsilon$ is the permittivity, and $\mu$ is the permeability.

Equations 6 to 12 utilize a set of unit dependent constants, $\kappa_1$ to $\kappa_6$. The sole purpose of these constants is to allow the equations to be valid for a variety of sets of units. Units in the context of electromagnetics is a topic of considerable discussion [24, 25]. While six constants are used for convenience, we note that the constants are not all independent. It can be shown that $\kappa_2 = \kappa_1\kappa_3$ by comparing the dimensionality of the Lorentz force with the dimensionality of Faraday's Law (Eq. 6). From the conservation of charge it follows that $\kappa_5 = \kappa_4\kappa_6$. Finally, from the electromagnetic wave equation in a vacuum one has $c^2 = (\kappa_3\kappa_4\varepsilon\mu)^{-1}$. Table 10 defines the $\kappa$ factors for five systems of units.

**Table 10.** Values of Constants for Various Systems of Units

| Constant | SI | Practical CGS | CGS Gaussian | CGS EMU | CGS ESU |
|---|---|---|---|---|---|
| $\kappa_1$ | 1 | $10^7$ | 1 | 1 | 1 |
| $\kappa_2 = \kappa_1\kappa_3$ | 1 | $\frac{1}{10}$ | $\frac{1}{c}$ | 1 | 1 |
| $\kappa_3$ | 1 | $10^{-8}$ | $\frac{1}{c}$ | 1 | 1 |
| $\kappa_4$ | 1 | $\frac{10^8}{c^2}$ | $\frac{1}{c}$ | 1 | 1 |
| $\kappa_5 = \kappa_4\kappa_6$ | 1 | $\frac{4\pi}{10}$ | $\frac{4\pi}{c}$ | $4\pi$ | $4\pi$ |
| $\kappa_6$ | 1 | $\frac{4\pi c^2}{10^9}$ | $4\pi$ | $4\pi$ | $4\pi$ |
| $\varepsilon$ (permittivity) | $\frac{10^7}{4\pi c^2}$ F/m | 1 | 1 | $c^{-2}$ | 1 |
| $\mu$ (permeability) | $4\pi 10^{-7}$ H/m | 1 | 1 | 1 | $c^{-2}$ |

In the MHD limit, it is assumed that the plasma is charge neutral ($\rho_q \approx 0$), and high-frequency phenomena are ignored ($\frac{\partial \vec{D}}{\partial t}$ is neglected), therefore the terms and equations in [square brackets] are ignored. Additional discussion can be found in several references [39, 17, 18].

While most input keywords are similar in 3D and 2D, there are some differences. These are noted in the following keyword descriptions. Various magnetic units are supported. See the `UNITS` keyword in Section 3.1.1 for more information.

3D Example:

```
transient magnetics
   formulation, whitney              $ select algorithm
   delta time = 5.e-6                $ implicit timestep

$ boundary conditions
   e tangent bc, sideset 10, 0., x 1. y 0. z 0.

   cyl axial slot bc, sideset 11, 1.0e6,
      x 0. y 0. z 0.,
      x 0. y 0. z 1.,
      0.00075, 0.001, 0.002, 0.00225

   uniform h bc, sideset 20, 0., x 0. y 0. z 1.
   uniform h bc, sideset 30, 0., x 1. y 0. z 0.
   void conductivity, 1.            $set void conductivity
   current density, projected       $method for current density computation
end
```

2D Example:

```
transient magnetics
   start = 0.
   stop  = 100.e-9
   delta time = 0.15e-9

   rz cyl radial slot bc, sideset 1, function 1, scale 1.,
      r 0.0, z 0.0, -2.0, -1.0, 1.0, 2.0
   centerline bc, sideset 3, 0.0,

   aztec set, 1
end
```

## 6.1 Formulation

```
FORMULATION, WHITNEY                                                     (3D)
FORMULATION, {FIFE | FIFE R SCALED}                                      (2D)
```

This keyword defines the solution formulation. In 3D, it is unnecessary as there is only one option.

In 2D, default value is geometry dependent. In 2D Cartesian geometry, the two methods are equivalent, hence `FIFE` (Fully Integrated Finite Element) is chosen automatically as the default. The difference is manifested in 2D cylindrical geometry, where `FIFE R SCALED` is the preferred method and is chosen automatically as the default. `FIFE` solves for $B_\theta$ or $A_\theta$ as the fundamental variable (depending on boundary conditions) and `FIFE R SCALED` solves for $\Psi = rB_\theta$ or $\Phi = rA_\theta$ as the fundamental variable (depending on boundary conditions).

## 6.2 Time Step Control

### 6.2.1 Start

```
START real (−∞)                                              (2D only)
```

This keyword indicates the simulation time to start the magnetic field solution, if different from the problem `START TIME`. Field values are set equal to zero prior to this time. It has no effect in 3D.

### 6.2.2 Stop

```
STOP real (+∞)                                               (2D only)
```

This keyword indicates the simulation time to stop the magnetic field solution, if different from the problem `TERMINATION TIME`. Field values are set equal to zero after this time. It has no effect in 3D.

### 6.2.3 Delta Time

```
DELTA TIME real (1.e-6) [FIXED | MAGNETIC DIFFUSION | MAGDIFF]     (3D)

DELTA TIME real (1.e-6) [FIXED | MAGNETIC DIFFUSION | MAGDIFF |
         AUTOSTEP, TOLERANCE real, BMIN real ]                     (2D)
```

The minimum of the mechanics time step and this `DELTA TIME` value is used as the actual magnetohydrodynamics time step. Otherwise this is the time step for the transient magnetics physics. The specified value provides an upper bound on the time step when using the non-fixed options described below. This value may also be used to give a time scale for the differential algebraic equation solver at `CIRCUIT SOLVER` initialization. The overall magnetics time step is reported in the `DT_MAG` global variable. This is the net result after considering all options described below.

The optional `FIXED` parameter causes transient magnetics to choose a fixed time step. This is the default.

The optional `MAGNETIC DIFFUSION` or `MAGDIFF` parameter adjusts the time step based upon the magnetic diffusion coefficient, $D$. Mesh element that are less than 1% material are excluded from consideration.

$$\Delta t \leq \frac{h^2}{4D} \tag{13}$$

where $D = 1/(\kappa_2 \kappa_5 \sigma \mu_0)$, $\sigma$ is the electrical conductivity, $\mu_0$ is the permeability, and $h$ is a characteristic cell size. The individual result of the magnetic diffusion time step is reported in the `DT_MAGDIFF` global variable.

The optional `AUTOSTEP` parameter is valid only in 2D and adjusts the time step by estimating a relative truncation error and adjusting the time step to keep the error below the specified `TOLERANCE`. The `BMIN` parameter omits mesh cells with a magnetic field value below this value from consideration.

$$\Delta t_{new} = \Delta t_{old} \sqrt{\frac{tolerance}{relerr}} \tag{14}$$

where $relerr = |\frac{\Delta B}{B}|$. Both the `MAGNETIC DIFFUSION` and `AUTOSTEP` options may be specified simultaneously.

Another factor, the Alfven speed, may control the time step when hydrodynamics is enabled. In this case, the Alfven speed is combined with a material wave speed, e.g., the sound speed, and used in a Courant condition. Keywords affecting the Alfven speed are described below. The individual result of the Alfven speed time step is reported in the `DT_ALFVEN` global variable.

$$\Delta t \leq \frac{h}{\sqrt{V_{hydro}^2 + V_{Alfven}^2}} \tag{15}$$

where $V_{Alfven} = \frac{B}{\sqrt{\mu_0 \rho}}$ in SI units.

Example 1:

```
transient magnetics
   ...
   delta time 0.5e-6, magnetic diffusion
   ...
end
```

Example 2:

```
transient magnetics
   ...
   delta time 1., autostep, tol 0.01, bmin 1.e-4, magdiff
   ...
end
```

### 6.2.4 Alfven Density Floor

`ALFVEN DENSITY FLOOR real (0.0)`

If this keyword is present then the Alfven speed is artificially set to zero in any element with a density below `ALFVEN DENSITY FLOOR`. This avoids the problem of very small time steps due to anomalously large Alfven speeds in low-density regions.

Example:

```
transient magnetics
   ...
   alfven density floor 0.01    $ gm/cmˆ3
   ...
end
```

### 6.2.5 Alfven Velocity Maximum

`ALFVEN VELOCITY MAXIMUM real (0.)`

If this keyword is present then the Alfven speed is artificially set to zero in any element with a Alfven wave speed greater than `ALFVEN VELOCITY MAXIMUM`. This avoids the problem of very small time steps due to anomalously large Alfven speeds.

Example:

```
transient magnetics
   ...
   alfven velocity max 1.0e7   $ m/s
   ...
```

```
end
```

## 6.3 Algorithm Control

### 6.3.1 A Rezone

```
A REZONE, {CT VAN LEER (default) | CT DONOR | CT HARMONIC |
           CT MONOTONIC | MONOA (2D only)}                      (2D,3D)
```

The `CT` options indicate the constrained transport reconstruction option for the magnetic flux in 3D or the vector potential in 2D Cartesian. The `CT` options take an optional `ISO` and `EH` arguments for use by developers. `CT VAN LEER` is the default in 2D, but `MONOA` is still available as an example of a not so good algorithm. This keyword can also be used in 3D, but `MONOA` is not permitted.

### 6.3.2 Alpha

```
ALPHA real (0.)                                                (2D only)
```

Variable weight of lumped versus consistent mass matrix. A value of 1.0 implies use of a fully consistent mass matrix and the default value of zero implies use of a lumped mass matrix. This is only active in 2D. In 3D a fully consistent vector edge element mass matrix is used.

The effect of this option can be seen in certain magnetic diffusion simulations. Use of the consistent mass matrix can result in small negative magnetic field values where none are expected. Use of the lumped avoids this behavior, thus it is the default.

### 6.3.3 Aztec Set

```
AZTEC SET int (0)
```

This defines the id-number of the Aztec parameter set which the magnetic solver will use.

For 2D MAG or MHD, the equations solved are formulated to always produce symmetric positive definite matrices, hence the recommended settings for Aztec parameters are:

```
AZTEC int
   SOLVER,    CG
   SCALING,   SYM_DIAG  $ (or SYM_ROW_SUM)
   PRECOND,   SYM_GS    $ (used if no MULTILEVEL)
   CONV NORM, RHS       $ (or ANORM or NOSCALED)
   OUTPUT,    NONE
   TOL,       1.E-12    $ (< 1.E-8)
   MAX ITER,  1000      $ (>500)
   MULTILEVEL
      FINE SWEEPS      = 5
      FINE SMOOTHER    = GAUSS SEIDEL
      COARSE SWEEPS    = 1     $ (because it is LU)
      COARSE SMOOTHER  = LU
      MULTIGRID LEVELS = 10
      INTERP ALGORITHM = AGGREGATION
   END
END
```

These settings for the solver, scaling and preconditioner are appropriate for symmetric positive definite matrices. A convergence norm setting of `R0` is not recommended because 2D `TRANSIENT MAGNETICS` uses the old solution as the initial guess and this norm can easily remain greater than the tolerance, especially if the solution approaches steady state. Tighter tolerances typically produce a more accurate answer and should be the first parameter changed if the answer is not approaching the expected solution, especially for high resolution meshes. Note however that tight tolerances may cause Aztec to report convergence warnings. These appear because Aztec converges for the scaled matrices, but not necessarily for the unscaled matrices or for the convergence norm. To eliminate these messages the user can try setting the `SCALING` and the `PRECONDITIONER` to `NONE` to force convergence of the unscaled matrices. Finally, the iterative solver typically should not require an exceptional number of iterations. One case where high iteration counts may be necessary is if the applied boundary conditions cause a strong step function change to the solution during the initial time step.

### 6.3.4 Conserve Memory

```
CONSERVE MEMORY
```

If this keyword is found then the magnetic solver code releases back to the system temporary memory utilized in setting up and solving the linear equations. Otherwise, this memory is only allocated once.

### 6.3.5 Magnetic Field

```
MAGNETIC FIELD, {PROJECTED | CONSISTENT}
```

Variations on implementations of deriving the magnetic field from the vector potential by solving the equation $\vec{B} = \nabla \times \vec{A}$. The default is `PROJECTED` in 3D and `CONSISTENT` in 2D. The `CONSISTENT` option is available in 2D only.

`PROJECTED` solves the curl equation by treating the LHS as a lumped diagonal matrix and evaluating the RHS by a single point integration using the value of the curl at the cell center.

`CONSISTENT` employs a finite element approach to calculate the nodal magnetic field directly from the nodal vector potential while skipping the element-centered magnetic field as an intermediary. The LHS is treated as a lumped diagonal matrix and the RHS is evaluated by Gaussian integration of the curl using finite element basis functions.

### 6.3.6 Magnetic Force

```
MAGNETIC FORCE, {TENSOR | PJXBV | PJXPBPV | NOBFORCE}
```

Variations on implementations of the magnetic force. Default is `PJXPBPV` in 3D and `TENSOR` in 2D. Both the `PJXPBPV` and `PJXBV` options first compute the magnetic field and current density at the mesh vertices and then form the cross product to determine the magnetic force. The `TENSOR` option computes the magnetic stress tensor at the element centers and then computes a surface integral around each mesh vertex to determine the magnetic force. `NOBFORCE` turns off the magnetic force term in MHD simulations. Primarily for developers.

### 6.3.7 Current Density

```
CURRENT DENSITY, {PROJECTED | CONSISTENT}
```

Various implementations of deriving the current density from either the magnetic field by solving $\vec{J} = \nabla \times \frac{1}{\mu}\vec{B}$ or from the vector potential by solving $\vec{J} = \nabla \times \frac{1}{\mu}\nabla \times \vec{A}$. The default is `CONSISTENT` in 2D. Only `PROJECTED` is available in 3D.

`PROJECTED` solves $\vec{J} = \nabla \times \frac{1}{\mu}\vec{B}$ by treating the LHS as a lumped diagonal matrix and evaluating the RHS by a single point integration using the value of the curl at the cell center.

`CONSISTENT` solves $\vec{J} = \nabla \times \frac{1}{\mu}\nabla \times \vec{A}$ by treating the LHS as a lumped diagonal matrix and evaluating the RHS by Gaussian integration using finite element basis functions. The RHS also integrates the double curl once to incorporate the boundary conditions.

### 6.3.8 Joule Heat

```
JOULE HEATING, {STANDARD | MAXSIGMA | NOHEAT}
```

Default is `STANDARD`. Joule heating is based on a fully-integrated, finite-element-based rate consistent with the discrete magnetic diffusion equations. In multi-material mesh cells, the Joule heat is partitioned among materials according to the conductivity fraction. For a simple steady state problem this gives the correct Joule heating in mixed cells.

`MAXSIGMA` limits Joule heating in the multi-material and mixed-void-material elements by effectively limiting the electric field by the ratio of the average conductivity to the maximum conductivity in the element. This prevents overheating of materials in mostly void cells and may allow calculations to run longer than using the `STANDARD` option.

$$\dot{Q}_{Joule} = \sigma_{ave}|E|^2 \approx \sigma_{ave}\left|\frac{\sigma_{ave}}{\sigma_{max}}E\right|^2 \tag{16}$$

`NOHEAT` turns off the Joule heating term.

Another feature that is built into ALEGRA is the ability to selectively turn off Joule heating for specific materials. Occasionally simulations are run with non-conducting materials, i.e., insulators. The electrical conductivity model should return a small, but finite, conductivity so that very little current flows in the insulator. Even though the conductivity and current density are small, the Joule heating, $J^2/\sigma$, may be non-negligible. Joule heating

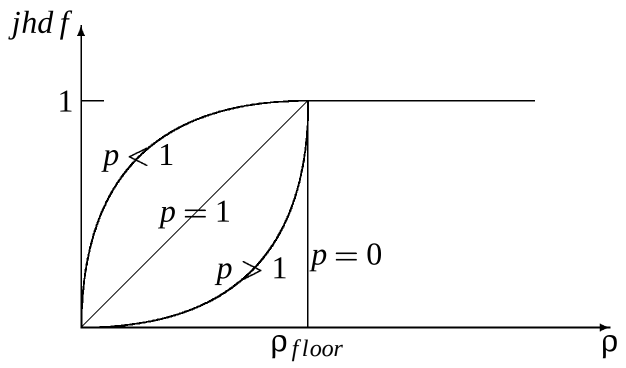

**Figure 3.** Joule Heat Density Floor.

in these materials may be disabled if the conductivity of the material is below the `VOID CONDUCTIVITY`. This feature also disables Joule heating of materials with no conductivity model.

### 6.3.9 Joule Heat Density Floor

```
JOULE HEAT DENSITY FLOOR real (0.0) [POWER real (0.0)]
```

Problems can arise in the calculation of the Joule heating in elements with very little mass. The effective heating can become much too large causing the material temperatures to become too high and the calculation to terminate. `JOULE HEAT DENSITY FLOOR` conditionally or gradually turns off the Joule heating term.

If the element mass density is less than the specified density floor, the Joule heating is limited, otherwise the Joule heating not changed. The multiplier ($jhdf$) of the Joule heating used when the average density ($\rho$) is less than the specified $\rho_{floor}$ is given by the following expression where $p$ is the specified `POWER`.

$$jhdf = \begin{cases} 0, & p = 0 \\ \left(\frac{\rho}{\rho_{floor}}\right)^{p}, & p > 0 \end{cases} \tag{17}$$

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 drops rapidly near 0. Graphically the multiplier can be displayed as in Figure 3.

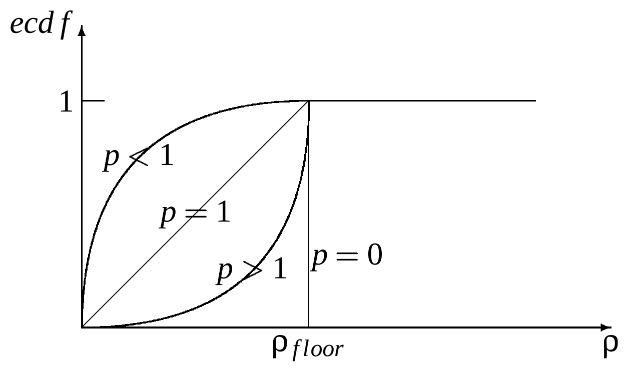

**Figure 4.** Electrical Conductivity Density Floor.

### 6.3.10 Electrical Conductivity Density Floor

```
ELECTRICAL CONDUCTIVITY DENSITY FLOOR real (0.0) [POWER real (0.0)]
```

Problems can arise in the diffusion of the magnetic field and hence the calculation of Lorentz forces in regions of the mesh with very little mass. `ELECTRICAL CONDUCTIVITY DENSITY FLOOR` conditionally or gradually reduced down to the `VOID CONDUCTIVITY`. This feature allows these regions to behave akin to void.

If the element mass density is less than the specified density floor, the electrical conductivity is limited, otherwise the electrical conductivity is not changed. The multiplier ($ecdf$) of the electrical conductivity used when the average density ($\rho$) is less than the specified $\rho_{floor}$ is given by the following expression where $p$ is the specified `POWER`. In all cases, the electrical conductivity will be greater than or equal to the `VOID CONDUCTIVITY`.

$$ecdf = \begin{cases} 0, & p = 0 \\ \left(\frac{\rho}{\rho_{floor}}\right)^p, & p > 0 \end{cases} \tag{18}$$

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 drops rapidly near 0. Graphically the multiplier can be displayed as in Figure 4.

### 6.3.11 Void Conductivity

```
VOID CONDUCTIVITY real (default 1.e-6)
```

This keyword defines the isotropic void conductivity as well as the default for materials if a conductivity model is missing. It is not a required input. The default is set to 1.E-6 $(\Omega\text{-m})^{-1}$, except for the 2D vector potential formulation, in which case the default is zero.

Under certain circumstances, the value of the void conductivity can be used to disable Joule heating in select materials (see Section 6.3.8 on page 46). This feature also disables Joule heating of materials with no conductivity model.

### 6.3.12 Void Reluctivity

```
VOID RELUCTIVITY real (default 7.957747e+5)
```

This keyword defines the isotropic void reluctivity, i.e. the inverse of the permeability, as the default for materials if a reluctivity models is not defined. It is not a required input. The default is set to $1/(4\pi{*}10^{-7}$ H/m).

## 6.4 Boundary Conditions

Magnetic field distributions (and subsequently current density) are generated using proper magnetic field initial conditions as well magnetic diffusion from boundary conditions. If not set properly, non-physical results may occur. Users are highly encouraged to first create a sketch of their experiment and identify the computational domain and relevant boundary conditions before setting up the grid. Default boundary conditions for the 2D RZ code are zero current density tangent to the boundary (current inflow or outflow). Default boundary conditions for the 2D XY and 3D code are zero tangent magnetic field.

### 6.4.1 Uniform H BC

```
UNIFORM H BC, sideset, function-set, vector, [LENGTH real]
```

Apply $\vec{H} = \vec{B}/\mu_0$ boundary conditions to the given `sideset` assuming the `vector` is tangential to the `sideset`. If it is not then the tangential component will be imposed. The `function-set` specifies the magnitude of the magnetic field H when no `LENGTH` keyword is present. If a `LENGTH` is given then the `function-set` is assumed to be a current rather than a magnetic field and `LENGTH` is used to change current to magnetic field units according to:

$$B(t) = \mu_0 H(t) = \frac{\mu_0 I(t)}{L} \tag{19}$$

In 2D, a vector is specified as:

```
[X real, Y real,] [Z real]                          (Cartesian geometry)
[R real, Z real,] [THETA real]                    (cylindrical geometry)
```

depending upon geometry. The first two components or third component may be optional.

It is often useful to specify `SCALE` = 7.957747e5 in the `function-set`, especially when working in SI units when the magnetic induction $B$ is specified. This `SCALE` is $1/\mu_0$ and scales $H = B/\mu_0$.

3D Examples:

```
uniform h bc, sideset 10, 1.0, scale 7.957747e5, x 0. y 0. z -1.
uniform h bc, sideset 11, 1.0, scale 7.957747e5, x 0. y 0. z 1.
```

2D Example (1 Tesla magnetic field in the boundary plane of a Cartesian mesh):

```
uniform h bc, sideset 10, 1.0, scale 7.957747e5, x 0. y -1.
```

2D Example (1 Tesla magnetic field orthogonal to the boundary plane of a Cartesian mesh):

```
uniform h bc, sideset 20, 1.0, scale 7.957747e5, z 1.
```

### 6.4.2 E Tangent BC

```
E TANGENT BC, sideset, function-set, vector
```

Apply tangential electric field boundary conditions to set the tangential components of the electric field.

`vector` is the electric field direction which the code normalizes to unit magnitude and `function-set` gives the magnitude.

3D Example:

```
e tangent bc, sideset 100, 0.0, x 1. y 0. z 0.
```

2D Example:

```
e tangent bc, sideset 2, 200.0, r 0. z 1.
```

Typically, E tangent boundary conditions of zero are used to specify that current density is traveling orthogonal (into or out of) the specified boundary.

### 6.4.3 Cylindrical Axial Slot BC

```
CYLINDRICAL AXIAL SLOT BC, sideset, {function-set | CIRCUIT},
     vector1, vector2,   r0, r1, r2, r3                          (3D)

{XY|RZ} CYLINDRICAL AXIAL SLOT BC, sideset, {function-set | CIRCUIT},
     [vector1,]   r0, r1, r2, r3                                 (2D)
```

This boundary condition can be utilized to approximate current inflow and outflow through a boundary in a manner similar to current flow into a device via an anode-cathode (AK) gap or COAX cable feeds. The user must apply the axial slot boundary condition to a `sideset` which must be a planar sideset cutting a cylindrical region. The prefix `XY` or `RZ` is needed in 2D to distinguish between Cartesian and cylindrical geometries.

The user begins by sketching the computational domain and identifying key boundary locations (see Figure 5). A coordinate system must first be established, thereby defining a cylindrical axis and radial points from it (the cylinder does not necessarily have to be aligned with the coordinate axes of the mesh).

The `vector1` defines a point on the axis of the cylinder and may displace the cylinder from the origin and also defines the $r = 0$ location. In 2D, it is used only in Cartesian geometry and is omitted in cylindrical geometry. The `vector2` is used only in 3D and defines the direction of the cylindrical axis. Given a point $\vec{P}$ on the `sideset`, $z = (\vec{P} - \vec{V}_1) \cdot$

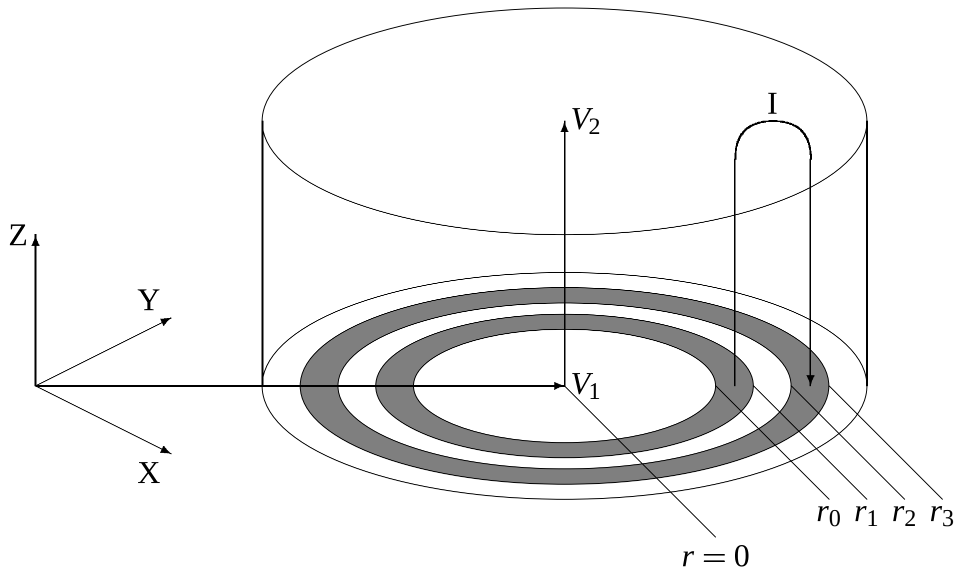

**Figure 5.** Geometry for the Cylindrical Axial Slot boundary condition.

$\vec{V}_2$ and $r = |\vec{P} - \vec{V}_1 - (z \cdot \vec{V}_2)|$, assuming `vector2` is normalized to unity (ALEGRA will do this).

The real parameters `r0`, `r1`, `r2`, and `r3` specify the radial range over which this boundary condition applies. The total current is assumed to flow uniformly into the mesh between `r0` and `r1` and out of the mesh between `r2` and `r3` so that there is no net accumulation of electrical charge. In other words, the axial current density is assumed constant between `r0` and `r1`, constant and of the opposite sign between `r2` and `r3`, and zero elsewhere on the sideset.

The `function-set` keyword may be used to drive the simulation with a prescribed current source. The direction (polarity) of the current flow can be reversed by changing the sign of the `function-set` (i.e., setting `SCALE` = -1.0). The `CIRCUIT` keyword may be used instead of a `function-set` to couple the simulation to an external circuit.

Relative to the axis of the cylinder the magnetic field in the sideset is specified by

$$H_\theta = f(r)\frac{I(t)}{2\pi r} \tag{20}$$

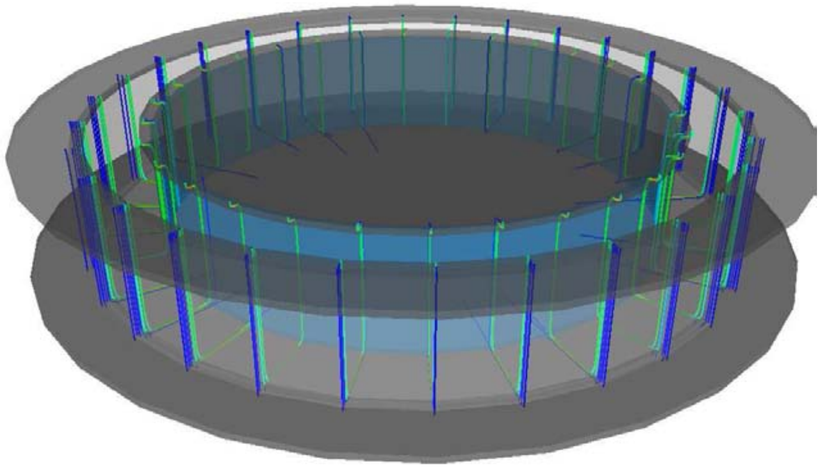

**Figure 6.** Current density streamlines from 3D cylindrical axial slot application. A large anode-cathode (AK) gap has been placed in the top plate where the boundary condition was applied. Using the cylindrical axial slot BC, the current can now flow down the side of the can, along the bottom, back up the inner (blue) wall and back out the domain as if the slot was connected to a much larger experimental apparatus which did not need to be incorporated into the simulation.

where

$$f(r) = \begin{cases} 0, & r \leq r_0 \\ \frac{r^2 - r_0^2}{r_1^2 - r_0^2}, & r_0 \leq r \leq r_1 \\ 1, & r_1 \leq r \leq r_2 \\ \frac{r_3^2 - r^2}{r_3^2 - r_2^2}, & r_2 \leq r \leq r_3 \\ 0, & r_3 \leq r. \end{cases} \tag{21}$$

In many instances the user may wish to allow the current to diffuse naturally into the inner and outer conductors. This may be accomplished by letting the slot (or gap) between $r_1$ and $r_2$ be its own sideset. $r_0$ and $r_1$ may be equal to each other and less than the inner radius of the gap. $r_2$ and $r_3$ may be equal to each other and greater than the outer radius of the gap. Because the extent of the sideset is limited to the gap, the boundary condition will be applied only in the gap. ALEGRA will then naturally diffuse the magnetic field and

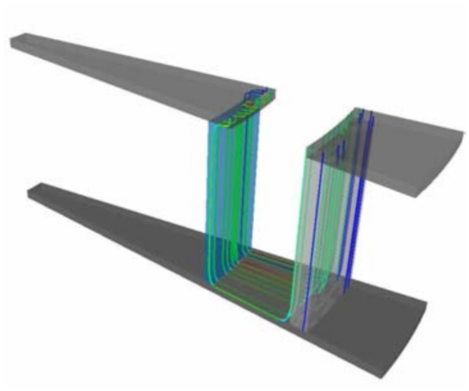

**Figure 7.** Current density streamlines from a periodic 3D cylindrical axial slot application in a wedge domain similar to that in Figure 6. In this example, the core of the domain has been removed.

current into the inner and outer conductors.

3D Example (this example specifies a constant 1 MA current flow through the mesh):

```
cyl axial slot bc, sideset 11, 1.e6,
   x 0. y 0. z 0.,
   x 0. y 0. z 1.,
   0.00075, 0.001, 0.002, 0.00225
```

2D Cylindrical Example:

```
rz cyl axial slot bc, sideset 1, function 2, scale 0.8,
   0.0, 0.0, 1.0, 2.0
```

Note that ”`scale`” in this example is actually part of the ”`function-set`” keyword.

2D Cartesian Example:

```
xy cyl axial slot bc, sideset 300, circuit,
   x 0.0 y 0.0,    1.0, 1.0, 2.0, 2.0
```

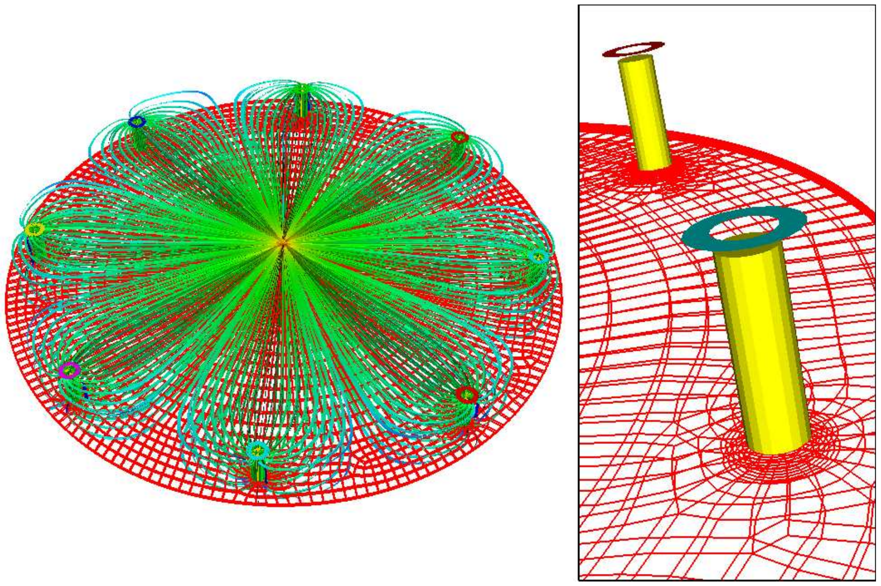

**Figure 8.** Current density streamlines from 3D cylindrical axial slot application. Individual axial slot boundary conditions are applied to each post, similar to COAX cable feeds. The top mesh plate has been removed in order to see the interior of the computational domain. The scale option can be used to appropriately scale the current at each post to a prescribed total current function.

### 6.4.4 Cylindrical Radial Slot BC

```
CYLINDRICAL RADIAL SLOT BC, sideset, {function-set | CIRCUIT},
     vector1, vector2,   z0, z1, z2, z3                            (3D)

{XY|RZ} CYLINDRICAL RADIAL SLOT BC, sideset, {function-set | CIRCUIT},
     vector1,   [z0, z1, z2, z3]                                   (2D)
```

Apply a current boundary condition to the given `sideset` which must be of cylindrical shape. The prefix `XY` or `RZ` is needed in 2D to distinguish between Cartesian and cylindrical geometries.

The `function-set` keyword may be used to drive the simulation with a known current source. The direction of the current flow can be reversed by changing the sign of the

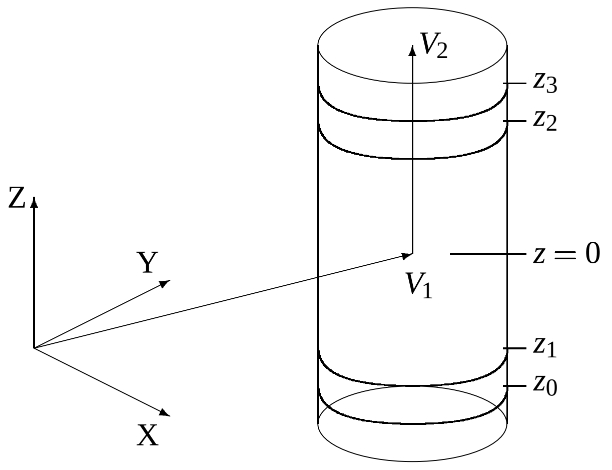

**Figure 9.** Geometry for the Cylindrical Radial Slot boundary condition.

`function-set` (i.e., setting `SCALE` = -1.0). The `CIRCUIT` keyword may be used instead of a `function-set` to couple the simulation to a set of circuit equations.

The `vector1` defines a point on the axis of the cylinder and may displace the cylinder from the origin and also defines the $z = 0$ location. In 2D cylindrical geometry, it only defines the $z = 0$ location. The `vector2` is used only in 3D and defines the direction of the cylindrical axis. Given a point $\vec{P}$ on the `sideset`, $z = (\vec{P} - \vec{V}_1) \cdot \vec{V}_2$ and $r = \left|\vec{P} - \vec{V}_1 - (z \cdot \vec{V}_2)\right|$, assuming `vector2` is normalized to unity (ALEGRA will do this).

The real parameters `z0`, `z1`, `z2`, and `z3` specify the axial range over which this boundary condition applies. The total current is assumed to flow uniformly into the mesh between `z0` and `z1` and out of the mesh between `z2` and `z3` so that there is no net accumulation of electrical charge. In other words, the radial current density is assumed constant between `z0` and `z1`, constant and of the opposite sign between `z2` and `z3`, and zero elsewhere on the `sideset`.

Relative to the axis of the cylinder the magnetic field in the sideset is specified by

$$H_\theta = f(z)\frac{I(t)}{2\pi r}, \tag{22}$$

where

$$f(z) = \begin{cases} 0, & z \le z_0 \\ \frac{z-z_0}{z_1-z_0}, & z_0 \le z \le z_1 \\ 1, & z_1 \le z \le z_2 \\ \frac{z_3-z}{z_3-z_2}, & z_2 \le z \le z_3 \\ 0, & z_3 \le z. \end{cases} \tag{23}$$

In many instances the user may wish to allow the current to diffuse naturally into the lower and upper conductors. This may be accomplished by letting the slot (or gap) between `z1` and `z2` be its own sideset. `z0` and `z1` may be equal to each other and less than the lower height of the gap. `z2` and `z3` may be equal and greater than the upper height of the gap. Because the extent of the sideset is limited to the gap, the boundary condition will be applied only in the gap. ALEGRA will then naturally diffuse the magnetic field and current into the lower and upper conductors.

In situations where the upper and lower conductors are not explicitly modeled, `z0` and `z1` may be below the mesh and `z2` and `z3` may be above the mesh to model a current that effectively flows into the bottom and out of the top of the mesh.

3D Example:

```
cyl radial slot bc, sideset 30, function 1,
   x 0.0 y 0.0 z 0.0,
   x 0.0 y 0.0 z 1.0,
   -10.0, -1.0, 1.0, 10.0
```

2D Cylindrical Example:

```
rz cyl radial slot bc, sideset 1, function 2, scale 0.8,
   r 0.0 z 0.0,    -2.0, -1.0, 1.0, 2.0
```

Note that ”`scale`” in this example is actually part of the ”`function-set`” keyword.

2D Cartesian Example:

```
xy cyl radial slot bc, sideset 300, circuit,
   r 0.0 z 0.0
```

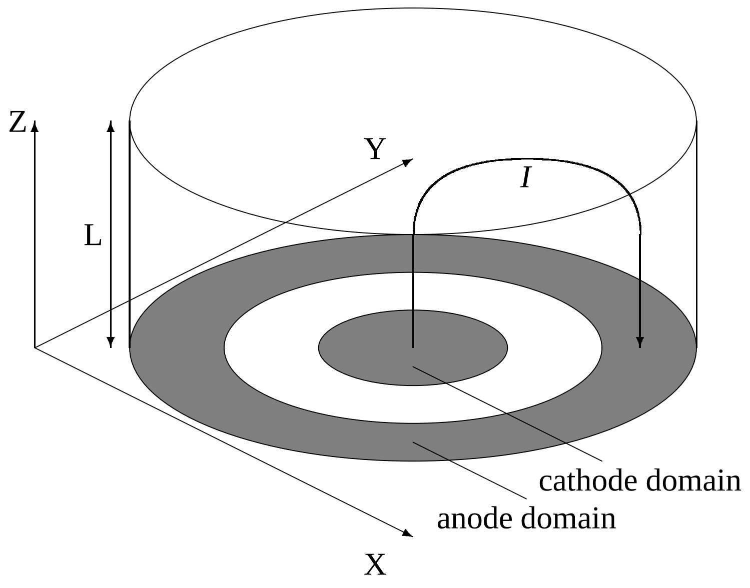

**Figure 10.** Geometry for the E Normal Current Return boundary condition.

### 6.4.5 E Normal Return Current BC

```
E NORMAL RETURN CURRENT BC, real,
     CATHODE BLOCK id-list, ANODE BLOCK id-list,
     {function-set | CIRCUIT}                              (2D XY only)

E NORMAL RETURN CURRENT BC, real,
     CATHODE MATERIAL id-list, ANODE MATERIAL id-list,
     {function-set | CIRCUIT}                              (2D XY only)
```

This boundary condition allows the modeling of a cross-section of a cathode-anode configuration where the normal current in the cathode is balanced by the return current in the anode. Specification of the cathode and anode may be accomplished using either `block-ids` or `material-ids`. Block and material specifications may not be mixed in a single simulation, however.

The first `real` is the length of the return current in the Z direction orthogonal to the plane of the mesh. IMPORTANT: The region keyword `VOLUMETRIC SCALE FACTOR` must also be set to the product of this length and any planar geometrical symmetries. This is because the code divides the volumetric scale factor by this normal length in order to compute the planar symmetry for scaling the current.

If the block specification is used, the cathode and anode `id-lists` need to include all blocks which may contain cathode or anode material during the course of the calculation. It is desirable to have void (blocks not specified in the cathode or anode lists and having no material) separating the two regions. The code will give warning messages if any material enters these void regions. The code cannot otherwise determine if the anode and cathode regions physically touch, causing an electrical short and thus invalidating the modeling assumptions.

If the material specification is used, the cathode and anode `id-lists` must be distinct. If the cathode and anode materials are the same, two `MATERIAL` keyword groups must still be defined, but may reference the same set of `MODEL` subkeywords. The number of mesh blocks is immaterial. No separating void blocks are needed. Cathode and anode materials may coexist in the same mesh block. Other materials may be present in addition to the anode and cathode materials. These materials may represent insulators or perhaps electrically isolated conductors. The code will abort should cathode and anode materials coexist in the same mesh cell. This situation indicates an electrical short and again invalidates the modeling assumptions.

The `function-set` is representative of a voltage drop driver. Alternatively the mesh may be coupled to an external `CIRCUIT` model.

Finally, this boundary condition is incompatible with H-tangent boundary conditions such as the `CYLINDRICAL AXIAL SLOT BC`, `CYLINDRICAL RADIAL SLOT BC`, or `UNIFORM H BC`.

Example 1:

```
e normal return current bc, 1.0,
     cathode block 5, anode block 1 2 3 4, function 1
```

Example 2:

```
e normal return current bc, 0.036,
     cathode material 11 12, anode material 13, circuit
```

The equations which describe this boundary conditions and the coupling to an external circuit equation have been documented [27].

### 6.4.6 Centerline BC

```
CENTERLINE BC, {sideset | NONE}                              (2D RZ only)
```

Zero the magnetic field or the vector potential along the given `sideset`. In 2D cylindrical geometry, the value of the magnetic field $B_\theta$, the vector potential $A_\theta$, and their products with the radius $r$ are identically zero along the centerline of the mesh. Multiple occurrences of this command are allowed when the centerline is comprised of more than one `sideset`.

This is the only boundary condition for magnetic field simulations that can be known *a priori* and was historically provided for the user. The method devised to implement the boundary condition, however, is rather inefficient for large meshes, requiring a loop over the entire mesh every time step. The historic method is still used in cylindrically symmetric problems when no `CENTERLINE BC`s are specified. Use of this boundary condition suppresses the historic method.

This boundary condition is valid along the inner edge of cylindrical meshes that do not extend to the centerline, but still have the field equal to zero because there is no enclosed current. It is also valid along the outer edge of cylindrical meshes where there is zero net current (i.e., there may be oppositely directed currents that cancel one another).

The keyword `NONE` specifies that there is no such applicable boundary condition. Such may be the case if there is an interior current-carrying conductor that is not explicitly modeled. The intent of the `NONE` keyword is to suppress the historic method in cases where there is no zero-field inner boundary.

2D RZ Examples:

```
centerline bc, sideset 11
centerline bc, sideset 12
```

## 6.5 Magnetic Flux Density Initial Conditions

Generally speaking magnetic flux density initial conditions must be used with care and consideration. In most real problems the interior magnetic field should be generated as a consequence of boundary conditions. However, sometimes it is convenient to input initial conditions and then view the evolution of the field from that point on. The `DIFFUSE INITIAL FIELD` option can also be used to smooth the initial conditions if desired before the actual start of the simulation. The general format for initial magnetic flux density initial conditions is

```
   INITIAL B FIELD, [block-id,] {option}
```

This specifies the application of an initial condition by `block-id`. The `block-id` determines the mesh block to which the initial condition applies. If omitted, the initial condition will apply to all mesh blocks. `block-ids` are required to be positive.

The `option` keyword allows the user to choose from the following initial conditions. If no initial conditions are specified, then the initial magnetic field is set equal to zero.

If multiple initial conditions are specified, then the fields are added according to the superposition principle.

### 6.5.1 Diffuse Initial Field

```
DIFFUSE INITIAL FIELD, [real]
```

This options invokes a transient magnetics solve before startup whereby the relevant magnetic field distribution may be computed for a given span of time based on the boundary conditions implemented. This option is particularly useful in establishing a potentially non-trivial magnetic field distribution before the simulation start time.

The `real` number represents the time interval for the magnetic field diffusion. The size of the time interval relative to the magnetic diffusion time, $\tau_d = l^2/4D$, where $D = 1/(\kappa_2\kappa_5\sigma\mu_0)$, determines the amount of diffusion that takes place. Currently this option performs only one magnetic solve. The user should be aware that setting the time interval too high may result in lack of convergence of the iterative solver.

### 6.5.2 Uniform Initial Field

```
INITIAL B FIELD, [block-id,] UNIFORM, vector1,
     [symtensor], [CONSTANT vector2]                          (3D)

INITIAL B FIELD, [block-id,] UNIFORM, vector1,
     [CONSTANT real]                                          (2D)
```

Uniform magnetic flux density value. `block-id` specifies the input block to which the field applies. If `block-id` is missing the field applies to all blocks.

The value, $\vec{B}$, `vector1`, specifies the magnitude and direction of the initial magnetic field. Units for the magnitude are Tesla in `SI` units, or Gauss in practical `CGS` units, or Gauss in `GAUSSIAN` units.

In 3D the `symtensor`, $\vec{\vec{S}}$, contributes an extra term to the vector potential to give certain continuity properties to the vector potential at block boundaries. It does not affect the magnetic field density except at block boundaries. The `vector2`, $\vec{C}$, is a constant vector offset to the vector potential available to match the vector potential across block boundaries.

$$\vec{A} = \vec{C} + .5(\vec{\vec{S}}\vec{x} + \vec{B} \times \vec{x}) \tag{24}$$

In 2D, a vector is specified as

```
[X real, Y real,] [Z real]                             (Cartesian geometry)
[R real, Z real,] [THETA real]                       (cylindrical geometry)
```

depending upon geometry. The first two components or third component may be optional. The `CONSTANT` is an offset to the vector potential component to match the vector potential across block boundaries.

### 6.5.3 Uniform Planar Current

```
INITIAL B FIELD, UNIFORM PLANAR CURRENT, vector1, vector2, vector3
                                                          (Cartesian only)
```

This keyword defines a planar magnetic flux density generated by a planar sheath of uniform current density. The magnetic field is zero on one side of the sheath (called the negative side), is uniform on the other side of the sheath (called the positive side), and varies linearly within the sheath. This initial condition applies to all mesh blocks and is not meaningful in 2D cylindrical geometry.

`vector1` specifies a point on the negative face of the sheath. `vector2` orients the sheath normal. Its magnitude equals the thickness of the sheath. Its direction points toward the positive side of the sheath where the magnetic field is non-zero. `vector3` defines the direction and magnitude of the magnetic field.

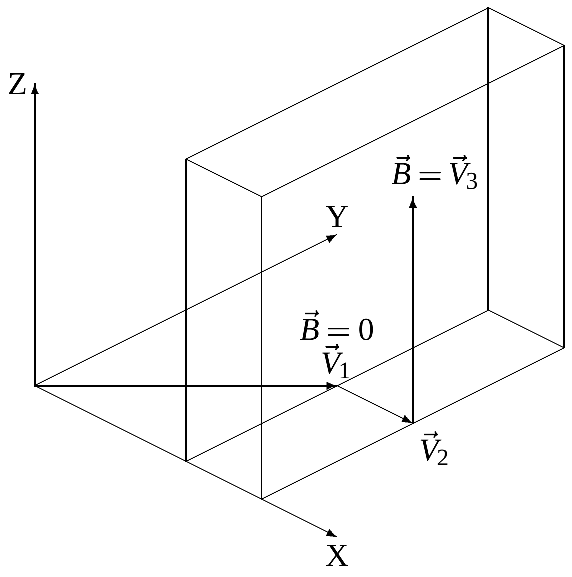

**Figure 11.** Geometry for the Uniform Planar Current initial condition.

In 2D, `vector1` and `vector2` have only (X,Y) components, while `vector3` may have all three components and is specified as:

```
[X real, Y real,] [Z real]
```

The analytic formulas for this initial condition are as follows. Given a point $\vec{P}$ in the mesh, the coordinate relative to the negative face of the sheath is $x = (\vec{P} - \vec{V}_1) \cdot \vec{V}_2 / |\vec{V}_2|$. The magnetic field and vector potential are:

$$\vec{B}(x) = B_0 \frac{\vec{V}_3}{|\vec{V}_3|} \begin{cases} 0, & x \leq 0 \\ x/|\vec{V}_2|, & 0 \leq x \leq |\vec{V}_2| \\ 1, & |\vec{V}_2| \leq x \end{cases} \tag{25}$$

$$\vec{A}(x) = B_0 \frac{\vec{V}_2 \times \vec{V}_3}{|\vec{V}_2 \times \vec{V}_3|} \begin{cases} 0, & x \leq 0 \\ -x^2/(2|\vec{V}_2|), & 0 \leq x \leq |\vec{V}_2| \\ -\frac{1}{2}|\vec{V}_2| - (x - |\vec{V}_2|), & |\vec{V}_2| \leq x \end{cases} \tag{26}$$

where $B_0 = |\vec{V}_3|$ has been called out for clarity.

For example in 2D Cartesian geometry, a 0.5 mm sheath perpendicular to the x axis and centered at 2.0 cm and with a 200 T magnetic field on its positive side is specified using SI units as:

```
initial b field, uniform planar current,
   x 0.01975 y 0.,  $ sheath face
   x 0.0005  y 0.,  $ sheath normal/thickness
   z 200.           $ magnetic field
```

### 6.5.4 Uniform Axial Current

```
INITIAL B FIELD, [block-id,] AXIAL CURRENT,
     [R0 real] [R1 real] [B0 real] [B1 real] [CONSTANT vector,]
     vector1, vector2,   z0, z1, z2, z3                            (3D)

INITIAL B FIELD, [block-id,] AXIAL CURRENT,
     [R0 real] [R1 real] [B0 real] [B1 real] [CONSTANT real,]
     vector1,   [z0, z1, z2, z3]                                   (2D)
```

This keyword defines an azimuthal magnetic flux density generated by a uniform axial current. All optional parameters default to zero if they are not specified.

`block-id` specifies the input block to which the field applies. If `block-id` is missing the field applies to all blocks.

`R0` and `R1` specify the inner and outer radii of the annular region where the axial current flows. The current density is assumed to be uniform for $R_0 \leq r \leq R_1$ and zero elsewhere. `B0` and `B1` specify the magnitude of the azimuthal magnetic field at the radii `R0` and `R1`, respectively.

The optional `CONSTANT` keyword is added to the vector potential to allow continuity of the vector potential across block boundaries.

`vector1` defines a point on the axis of the cylinder which displaces the annular region from the origin and defines the $z = 0$ point along the axis. `vector2` defines the direction of the axis of the region in 3D (ALEGRA will normalize `vector2`). These parameters are similar to the `CYLINDRICAL RADIAL SLOT BC` boundary condition (see Figure 12).

The real numbers `z0`, `z1`, `z2`, and `z3` define the axial extent of the initial field. The magnetic field is modified by a factor $f(z)$ that increases linearly from zero to one between

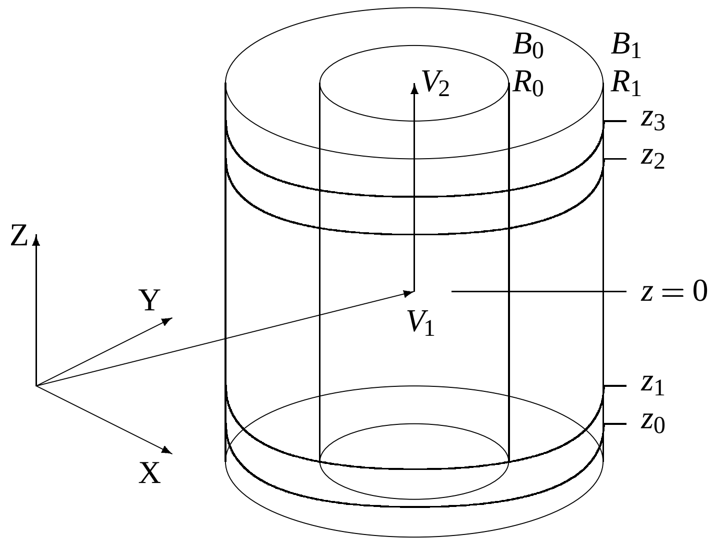

**Figure 12.** Geometry for the Axial Current initial condition.

`z0` and `z1`, is unity between `z1` and `z2`, and decreases linearly back to zero between `z2` and `z3`. It is zero outside this range. This models a positive uniform current density entering the mesh through a cylindrical slot between `z0` and `z1` and exiting the mesh through a cylindrical slot between `z2` and `z3`. `z0` and `z1` can be below the mesh and `z2` and `z3` above the mesh to model a current that flows into the bottom and out of the top of the mesh.

$$f(z) = \begin{cases} 0, & z \leq z_0 \\ \frac{z-z_0}{z_1-z_0}, & z_0 \leq z \leq z_1 \\ 1, & z_1 \leq z \leq z_2 \\ \frac{z_3-z}{z_3-z_2}, & z_2 \leq z \leq z_3 \\ 0, & z_3 \leq z. \end{cases} \tag{27}$$

In 2D, `vector1` is a point on the axis of the cylindrical region. In Cartesian geometry it can be used to displace the center of the region from the origin. In cylindrical geometry, it defines the $z = 0$ point along the axis. No second vector is needed because the direction of the cylindrical axis can only be parallel to the z axis. The real numbers `z0`, `z1`, `z2`, and `z3` are only needed in 2D for cylindrical geometry and have the same meaning as in 3D.

The analytic formulas for this initial condition are as follows. Given a point $\vec{P}$ in the mesh, the location along the axis is $z = (\vec{P} - \vec{V}_1) \cdot \vec{V}_2$ and the radial distance from the cylindrical axis is $r = |\vec{P} - \vec{V}_1 - (z \cdot \vec{V}_2)|$. The azimuthal magnetic field and vector potential are:

$$B_\theta(r) = \begin{cases} 0, & r < R_0 \\ \frac{1}{r}\left(R_0 B_0 \frac{R_1^2 - r^2}{R_1^2 - R_0^2} + R_1 B_1 \frac{r^2 - R_0^2}{R_1^2 - R_0^2}\right), & R_0 \le r \le R_1 \\ B_1 \frac{R_1}{r}, & R_1 \le r \end{cases} \tag{28}$$

$$\vec{A}(r) = \vec{A}_0 + \vec{n} \begin{cases} 0, & r < R_0 \\ \frac{R_0 B_0 - R_1 B_1}{2}\left(\frac{r^2 - R_0^2}{R_1^2 - R_0^2}\right) + \frac{(R_0 B_1 - R_1 B_0) R_0 R_1}{R_1^2 - R_0^2} \ln\left(\frac{r}{R_0}\right), & R_0 \le r \le R_1 \\ \frac{R_0 B_0 - R_1 B_1}{2} + \frac{(R_0 B_1 - R_1 B_0) R_0 R_1}{R_1^2 - R_0^2} \ln\left(\frac{R_1}{R_0}\right) - R_1 B_1 \ln\left(\frac{r}{R_1}\right), & R_1 \le r \end{cases} \tag{29}$$

where $\vec{A}_0$ is the optional `CONSTANT` and $\vec{n}$ is the unit vector in the direction of `vector2`. $\vec{A}_0$ is added to the vector potential to allow continuity of the vector potential across block boundaries. Quite often, $R_0 = 0$ and/or $B_0 = 0$ and the above expressions simplify tremendously. In 2D, $\vec{A}_0$ and $\vec{n}$ have only a z component.

For example, if block 3 is an annular region located between two perfect cylindrical conductors at radii 0.002 meters and 0.02 meters and 1000 Amperes of current flows on the inner conductor and -1000 Amperes flows on the outer conductor, the magnetic field in block 3 is described in 3D by:

```
$ calculate A as a superposition of fields
$ block 1 (inner conductor) defaults to zero
$ block 2 (outer conductor)

  initial b field, block 2, axial current, r1 0.02, b1 0.0,
     constant, x 0., y 0., z -5.605e-4,
     x 0., y 0., z 0.,  x 0., y 0., z 1.,   -2., -1., 1., 2.

$ block 3 (annular region)

  initial b field, block 3, axial current, r1 0.002, b1 0.1,
     x 0., y 0., z 0.,  x 0., y 0., z 1.,   -2., -1., 1., 2.
```

or in 2D cylindrical by:

```
$ Btheta (not Atheta) is the fundamental variable
$ blocks 1 and 2 (inner and outer conductors) default to zero
$ block 3 (annular region)

  initial b field, block 3, axial current, b 0.1, r 0.002,
     r 0., z 0.,   -2., -1., 1., 2.
```

### 6.5.5 Mean Initial Field

```
INITIAL B FIELD, MEAN, vector
```

This keyword is intended to be used for vector potential solutions on periodic meshes. One cannot represent a constant magnetic field with a periodic vector potential. In 2D, this means that the magnetic field must be in the plane of the mesh, orthogonal components are not allowed. The `vector` is a magnetic field vector that is constant in both space and time. There is no `block-id` subkeyword with this command. To be constant in space, this command must apply to all blocks. This keyword is typically used in conjunction with other initial fields and/or boundary conditions to represent the total magnetic field.

In 3D, the `vector` is specified as:

```
X real, Y real, Z real                            (Cartesian geometry)
```

In 2D, the `vector` is specified as either:

```
X real, Y real                                    (Cartesian geometry)
R real, Z real                                  (cylindrical geometry)
```

Orthogonal components to the mesh are not allowed. Only a non-zero `Z` component makes physical sense in cylindrical geometry.

Example:

```
   initial b field, block 1, uniform, x 0.0 y  0.001120998243  $ left
   initial b field, block 2, uniform, x 0.0 y -0.001120998243  $ right

   initial b field, mean, x 0.0008407486825 y 0.0
   uniform h bc, sideset 21,  0.001120998243, scale 7.957747e5,
                 x 0.0 y 1.0                                  $ left
   uniform h bc, sideset 22, -0.001120998243, scale 7.957747e5,
                 x 0.0 y 1.0                                  $ right
```

### 6.5.6 User Defined

```
INITIAL B FIELD, [block-id] USER DEFINED, (A or B)

[END]
```

This option provides the user with substantial flexibility in establishing a magnetic field based upon an analytic function using basic functions. The user may specify the vector potential field (**A**) in 3D or the scalar potential ($A_z$) for 2D xy simulations. The magnetic flux density is given by the curl of **A**. For 2D xy or rz simulations the field value is the scalar magnetic flux density ($B_z$ or $B_\theta$). This option is the preferred method of initializing magnetic field configurations that are not covered by other options. The C code which is implemented must abide by the special rules given in the file README.txt in the alegra/tools/rtcompiler directory of the ALEGRA distribution. Three examples are provided:

```
INITIAL B FIELD, [block-id,] USER DEFINED, A,
   $ quoted C code with:
   $    coord[0..1] as the input point in the field and
   $    field[0] as the output vector potential scalar field.
   $ For example, specify a 2D xy vector potential field
   "field[0] = -3.*coord[0]*cos(coord[1]);"
[END]

INITIAL B FIELD, [block-id,] USER DEFINED, B,
   $ quoted C code with:
   $    coord[0..1] as the input point in the field and
   $    field[0] as the output magnetic flux density field.
   $ For example, specifies the magnetic flux density field in 2D rz.
   "field[0] = -3.*coord[0]*cos(coord[1]);"
[END]

INITIAL B FIELD, [block-id,] USER DEFINED, A,
   $ quoted C code with:
```

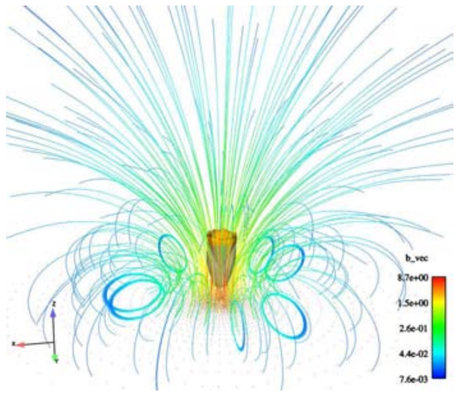

**Figure 13.** Magnetic field streamlines from 3D user defined magnetic field example.

```
   $    coord[0..2] as the input point in the field and
   $    field[0..2] as the output vector potential field.
   $ For example, specify a 3D xyz vector potential field for a coil of radius L1
   $ with magnitude B1.
      "
        double pi     = acos(-1.0);
        double radius = sqrt( coord[0]*coord[0] + coord[1]*coord[1] +
                        coord[2]*coord[2] );
        double theta  = atan2( sqrt( coord[0]*coord[0] +
                               coord[1]*coord[1] ), coord[2] );
        double phi    = atan2( coord[1], coord[0] );
        double aphi   = {B1*L1*L1}*radius*sin(theta) /
                        ( ({L1*L1}+radius*radius+{2.0*L1}*radius*sin(theta))*
                        sqrt(({L1*L1}+radius*radius+{2.0*L1}*radius*sin(theta)))
                        );

        field[0] =  -aphi*sin(phi)/2.0;
        field[1] =   aphi*cos(phi)/2.0;
        field[2] =   0.;
      "
[END]
```

## 6.6 Coupled Circuit Equations

If required, the user can setup an external circuit (represented as a set of differential-algebraic equations) to be coupled to the transient magnetic or MHD simulations. This is useful in situations where the simulation is driven by a known voltage source (most magnetic boundary conditions require knowledge of the current) or where the dynamics of the system being modeled feeds back into the electrical response of the external circuit. The circuit solver may be used in both 2D and 3D, and in either `SI` or `CGS` (Practical CGS) units. An explanation of the global output variables associated with the circuit solver is given in Table 54 on page 163.

The circuit equations are solved using DASPK [29, 15, 36]. DASPK is describe further in Section 6.6.1 below.

Coupled circuit equations are fairly simple to set up and the method is outlined using the following steps.

1. Sketch a diagram of the equivalent circuit (see Figure 14).
2. Label the junctions between the circuit components with numbers. Each junction becomes a `CIRCUIT NODE` in the input deck. The numerical labels become the node numbers in the input deck. Circuit nodes can connect more than two circuit elements, that is, parallel circuits are allowed.
3. Each circuit component becomes a `CIRCUIT ELEMENT` in the input deck. Most circuit elements connect two nodes and these are designated as the input and output nodes. A few circuit elements connect three nodes. Positive currents flow from the input node to the output node. If the order of the nodes happen to be mislabeled, no matter, ALEGRA will report a negative current flowing in the element.
4. ALEGRA will automatically construct a set of equations consistent with Kirchhoff's laws for the equivalent circuit, namely that the sum of the voltages around a loop is zero, $\sum V = 0$, and that the sum of the currents into a node is zero, $\sum I = 0$.

For example, the Z accelerator in the Pulsed Power Center at Sandia National Laboratories can be modeled using an equivalent circuit. The simplest such model has only a time-dependent voltage source, an external resistance and an external inductance. The Z pinch load is modeled using the magnetohydrodynamics features of ALEGRA. As the Z pinch plasma (load) collapses and stagnates onto its axis, the inductance of the load rises dramatically causing a current drop in the external circuit. Voltage monitors along the accelerator reliably measure the voltage at a significant distance from the load. The actual

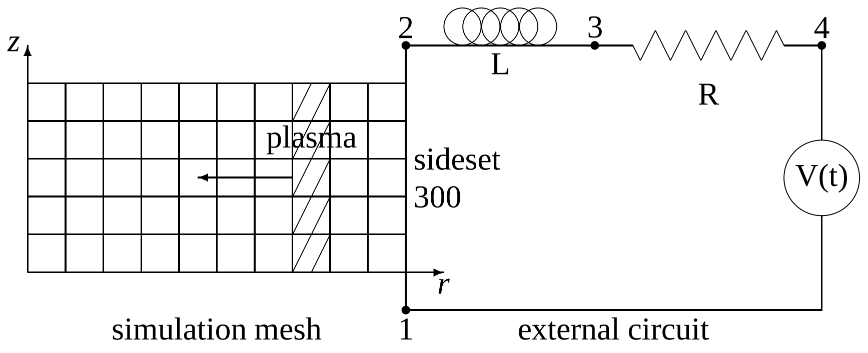

**Figure 14.** Simple circuit model for the Z accelerator.

current through the load is not known to the same accuracy. Both of these aspects can be modeled using a coupled circuit equation.

Example:

```
$ external circuit loop
    circuit solver, relerr 1.e-4, abserr 1.e-4

    circuit node, 1, fixedv 0. $ ground
    circuit node, 2
    circuit node, 3
    circuit node, 4, function 1 $ source voltage

  $ circuit element, 1 4  $ represented by source voltage
    circuit element, 4 3, resistor 0.120    $ ohms
    circuit element, 3 2, inductor 12.05e-9 $ henrys
    circuit element, 2 1, mesh

    rz cyl radial slot bc, sideset 300, circuit,
       r 0. z 0.,   -10., -1., 1., 10.
```

### 6.6.1 Circuit Solver

```
CIRCUIT SOLVER, [RELERR real (0.01)] [ABSERR real (0.01)]
```

Circuit solver error criteria passed to DASPK. The error values typically used range from `1.E-4` to `1.E-6`. The DASPK solution procedure is somewhat sensitive. The DASPK solver is an adaptive differencing type of code which uses divided difference to estimate accuracy. It is the contention of the authors that the user cannot push the tolerances too low or else one will end up with roundoff problems. If the user gives some reasonable happy medium tolerance the package will be reasonably robust. It is recommended that the user adjust these error parameters to gain a feel for solution response and carefully monitor the actual solutions obtained.

### 6.6.2 Circuit Node

```
CIRCUIT NODE node_#
   [sideset]
   [ STARTV real |
     FIXEDV real |
     function-set |
     DAMPED SINUSOID amplitude damping_rate frequency phase |
     DOUBLE EXPONENTIAL A_amplitude a_exponent B_amplitude b_exponent]
```

This input defines a circuit node and associated voltage. A node may be attached to one or more `CIRCUIT ELEMENT`s. According to Kirchhoff's law the sum of the currents flowing from each circuit element into the node must sum to zero. If a node connects to only one circuit element then that node must have its voltage specified.

The `sideset`, if specified, is constant potential surface in the ALEGRA mesh. The voltage at this surface is defined by one of the following voltage options. The potential is linked to the ALEGRA simulation through an `E TANGENT BC` boundary condition.

The remaining set of optional keywords allow a choice of the voltage behavior of the node. Note that the user is not required to specify a voltage option. In that case, the voltage is allow to vary and the values are computed by the circuit model.

`STARTV` sets the initial voltage of the node. Subsequent voltage values are computed by the circuit model. This option is useful for specifying an initial voltage across a capacitor element.

`FIXEDV` fixes the voltage of the node at a constant value. At least one node must have a constant zero voltage. This node provides the ground for the circuit model.

`function-set` or `FUNCTIONV` specifies a tabular voltage input for the node. The `table_#`

refers to a standard ALEGRA `FUNCTION` and the optional `SCALE` parameter is a voltage multiplier.

`DAMPED SINUSOID` requires four real values and specifies a time dependent voltage of the form

$$V(t) = A \cdot e^{-a \cdot t} \cdot \sin(\omega \cdot t + \theta) \quad , \tag{30}$$

where $A$ is the voltage `amplitude`, $a$ is the `damping_rate`, $\omega$ is the `frequency` (radians/second) and $\theta$ is the `phase` (radians).

`DOUBLE EXPONENTIAL` requires four real values and specifies a time dependent voltage of the form

$$V(t) = A \cdot e^{-a \cdot t} - B \cdot e^{-b \cdot t} \quad . \tag{31}$$

### 6.6.3 Circuit Element

```
CIRCUIT ELEMENT input_node output_node
   { MESH |
     CAPACITOR real_capacitance |
     INDUCTOR  real_inductance |
     RESISTOR  real_resistance |
     SOURCE,   function-set |
     SWITCH,   function-set |
     VARISTOR  real_vref real_iref real_alpha |
     ZFLOW,    {function-set} real_gmin real_gmax }
```

This line defines each circuit element and its characteristics. Many circuit element input values are order dependent and have no keywords.

The `input_node` and the `output_node` are integer numbers correspond to nodes defined by the `CIRCUIT NODE` keyword. A positive current flows through the circuit element from the `input_node` to the `output_node`.

One of the remaining keywords is required to specify the type of circuit element. Valid options are specified in the following list.

`MESH` specifies that this circuit element represents the mesh. This circuit element couples to a `TRANSIENT MAGNETICS` boundary condition that specifies the `CIRCUIT` keyword such as the `CYLINDRICAL RADIAL SLOT BC`.

`CAPACITOR` specifies a standard electrical capacitor with a capacitance value of `real_capacitance`.

`INDUCTOR` specifies a standard electrical inductor with a inductance value of `real_inductance`.

`RESISTOR` specifies a standard electrical resistor with a resistance value of `real_resistance`.

`SOURCE` specifies a prescribed current source defined by the `function-set`.

`SWITCH` specifies a resistor with a time dependent resistance value given by a `function-set`.

`ZFLOW` specifies a Zflow resistor with a Zflow value given by the `function-set`. Minimum and maximum conductance values are also required. The first node in the element node list specifies the Zflow loss node. The model requires that the Zflow loss node have a circuit element connectivity of 3. The conductance for the Zflow element is given by

$$\frac{I_{upstream} - I_{downstream}}{Z_{flow}\sqrt{I^2_{upstream} - I^2_{downstream}}} \tag{32}$$

Example:

```
circuit solver, relerr=1.e-4, abserr 1.e-4

$ mesh loop
$ capacitor-mesh $ mesh has inductance and resistance

  circuit node 1 fixedv 0.
  circuit node 2 startv 1.  $ initial capacitor voltage

  circuit element, 1 2, capacitor 5.e-6    $ Farads
  circuit element, 2 1, mesh

  xy cyl radial slot bc, sideset 300, circuit,
     r 0. z 0.

$ test loop - independent of mesh
$ capacitor-inductor-resistor
```

```
  circuit node 3 fixedv 0.
  circuit node 4 startv 1.  $ initial capacitor voltage
  circuit node 5

  circuit element, 3 4, capacitor 5.e-6    $ Farads
  circuit element, 4 5, inductor 18.64e-9  $ Henrys
  circuit element, 5 3, resistor  5.09e-2  $ Ohms
```

This example is from a problem that computes the behavior of a section of coaxial cable (the mesh) driven by the voltage stored in a capacitor. This is represented by nodes 1 and 2 and the associated circuit elements. For comparison a simple test RLC circuit is also modeled by node 3, 4, and 5, and the associated elements. Note that this test circuit is completely independent of the mesh demonstrating the flexibility of the circuit model.

## 6.7 Current Tally

```
CURRENT TALLY, int,
     sideset1 [SYMMETRY FACTOR real] [sideset2 ...]
END                                                        (3D/2D)

CURRENT TALLY, int,
     block-id1 [SYMMETRY FACTOR real] [block-id2 ...]
END                                                      (2D only)
```

This option allows the user to tally the total current flowing through a collection of sidesets or blocks as a function of time. Multiple sidesets may be included in a single tally. The `int` is used to identify the tally in the EXODUS and HISPLT output files. In 2D, the user may specify a set of blocks to tally the total current flow perpendicular to the mesh. `sidesets` and `block-ids` may not be mixed in the same tally, although both may appear in separate `CURRENT TALLY` commands in the same input deck. Both the incoming and outgoing currents are reported for each tally. The names of the tallies in the output files are `I_IN`*int* and `I_OUT`*int*.

The optional `SYMMETRY FACTOR` keyword allows the user to individually scale the current tally of each `sideset` or `block` in situations where a symmetry is present and a reduced simulation is modeled. This keyword is necessary because sidesets (which are areas) may not scale in the same manner as volumes (e.g., the `VOLUMETRIC SCALE FACTOR` keyword).

Coding internal to ALEGRA attempts to force the user to specify a completed set of tallies. The code checks to see that the incoming and outgoing currents are equal, or else that one is zero in the case of a partial tally. If the two values differ by more than some tolerance (currently 1.E-5), a warning message is issued.

Assuming the specified tallies represent the total current, ALEGRA also reports the total current in the variable `CURRENT`, the total mesh inductance in the variable `INDUCTANCE`, and the total mesh resistance in the variable `RESISTANCE`.

Examples:

```
current tally, 1, sideset 3, sideset 4, end
current tally, 2, block 12, end
```

# 7 Thermal Conduction Input

```
THERMAL CONDUCTION
   ...
   ... [thermal conduction keywords] ...
   ...
END
```

The `THERMAL CONDUCTION` keyword group specifies controls for the implicit thermal solver routines.

$$\rho C_v \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) \tag{33}$$

where $T$ is the temperature, $\rho$ is the density, $C_v$ is the specific heat, and $k$ is the thermal conductivity.

## 7.1 Time Step Control

The thermal conductivity time step is given by

$$\Delta t_{con} = f \min(\Delta t_k, \Delta t_c) \tag{34}$$

where

$$\Delta t_k = \frac{\rho C_v h^2}{k} \tag{35}$$

where $h$ is a characteristic cell size and

$$\Delta t_c = \Delta t(.05)|T + T_{min}|/|T - T_{old}| \tag{36}$$

### 7.1.1 Time Step Scale, $f$

```
TIME STEP SCALE real (default 10.)
```

The keyword `TIME STEP SCALE` can be set to control the scale, $f$, applied to the time step. The time step is calculated using an explicit criteria, but can be increased by any desired factor $f$ since the actual calculation is implicit, by use of the `TIME STEP SCALE` keyword. The default value is 10.

The thermal conduction time step is reported in the `DT_CON` global variable. Note that this time step is reported in the output file without the `TIME STEP SCALE` factor.

### 7.1.2 Minimum Temperature, $T_{min}$

`MINIMUM TEMPERATURE real (default 1000 K)` $T_{min}$ is the cutoff temperature for temperature change control.

### 7.1.3 Minimum Density , $\rho_{min}$

`MINIMUM DENSITY real (default 0.)` $\rho_{min}$ is the cutoff thermal conductivity density. This keyword is needed when the thermal solves are taking too many iterations due to a stiffness rather than mass dominated matrix corresponding to extremely high thermal diffusion speeds. This can happen for very small densities occurring in the problem. No multigrid is available for the face centered discretization employed so to make progress this keyword can be employed to essentially turn off diffusion in very low density regimes. This is accomplished internally in the code by setting the thermal conductivity $k$ to very small values.

## 7.2 Algorithm Control

### 7.2.1 Aztec Set

```
AZTEC SET int
```

This keyword sets the identification number of the Aztec parameter set which the implicit thermal solver will use. The actual parameters used are specified by the `AZTEC` keyword group. The default value is 0.

### 7.2.2 Conserve Memory

```
CONSERVE MEMORY
```

Requests that all solver memory for conduction be returned to the system and reallocated the next cycle. By default the memory is retained across cycles. WARNING: It is possible that excessive allocations and deallocations can have a detrimental effect on code performance.

### 7.2.3 Thermal Conductivity Density Floor

```
THERMAL CONDUCTIVITY DENSITY FLOOR real (0.0) [POWER real (0.0)]
```

Problems can arise in the diffusion of the material temperature in regions of the mesh with very little mass. `THERMAL CONDUCTIVITY DENSITY FLOOR` conditionally or gradually reduces the thermal conductivity down to zero. This feature allows low density regions to behave akin to void.

If the element mass density is less than the specified density floor, the thermal conductivity is limited, otherwise the thermal conductivity is not changed. The multiplier ($ecdf$) of the thermal conductivity used when the average density ($\rho$) is less than the specified $\rho_{floor}$ is given by the following expression where $p$ is the specified `POWER`.

$$tcdf = \begin{cases} 0, & p = 0 \\ \left(\frac{\rho}{\rho_{floor}}\right)^p, & p > 0 \end{cases} \tag{37}$$

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 drops rapidly near 0. Graphically the multiplier can be displayed as in Figure 15.

## 7.3 Boundary Conditions

### 7.3.1 No Heat Flux

```
NO HEAT FLUX, sideset
```

This keyword allows the user to specify zero heat flux through a surface defined by the `sideset` argument.

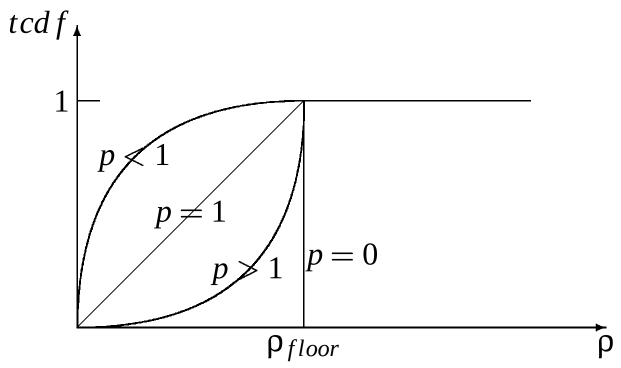

**Figure 15.** Thermal Conductivity Density Floor.

### 7.3.2 Prescribed Heat Flux

`PRESCRIBED HEAT FLUX, sideset, function-set`

This keyword allows the user to prescribe the heat flux on a surface defined by the `sideset` keyword as a function of time.

### 7.3.3 Temperature BC

`TEMPERATURE BC, sideset, function-set`

This keyword allows the user to specify the temperature on a surface defined by the `sideset` keyword as a function of time.

# 8 Radiation Transport Input

```
RADIATION
   {LINEARIZED DIFFUSION | KULL IMC}
      ...
      ... [radiation transport model keywords] ...
      ...
   END
END
```

The section only serves to make the user aware of the radiation transport capability of ALEGRA and to provide a brief summary of some of the keywords. Details of the implementation can be found in various references [10, 9, 8].

The `RADIATION` input controls the radiation transport capability in this version of ALEGRA. This allows for modeling radiation transport effects in either the linearized diffusion or the implicit Monte Carlo approximations. Note that if the radiation transport block is contained within a more comprehensive physics block such as `RADIATION HYDRODYNAMICS`, then the outermost `RADIATION ... END` keyword pair in the above syntax can be omitted.

It is extremely important to always specify all your radiation boundary conditions. Different transport methods may have different types of boundary conditions that are naturally default. (Monte Carlo has a different default than nodal diffusion, for instance.) If you change transport methods, your answer will change unexpectedly if you rely on the default boundary conditions.

## 8.1 The Transport Equation

The Boltzmann transport equation describes how a variety of different types of particles travel through a material. It is generally considered the most accurate description of the statistical average density of particles in a system. The Boltzmann equation is very general, and every discipline has a different way to write it that fits their needs best. The intensity $I(\vec{x},\hat{\Omega},\varepsilon,t)$ is the primary variable having units of (energy/area/time/solid-angle/energy-interval). $\varepsilon = h\nu$ is the photon energy, $\nu$ is the photon frequency, $h$ is Planck's constant, and $c$ is the speed of light.

The intensity and Boltzmann equation may be integrated with respect to energy to yield the single or multi-group transport equation.

$$I_g(\vec{x},\hat{\Omega},t) = \int_{\varepsilon_g}^{\varepsilon_{g+1}} I(\vec{x},\hat{\Omega},\varepsilon',t)\,d\varepsilon' \tag{38}$$

The single-group, energy-integrated Boltzmann equation for radiation transport is

$$\frac{1}{c}\frac{\partial I}{\partial t} + \hat{\Omega}\cdot\nabla I = -\tau_t I + \frac{\tau_s}{4\pi}\int_{4\pi} I\,d\Omega' + \tau_a \frac{c}{4\pi}B(T_m) + S \tag{39}$$

In addition to Eq. 39, which describes the evolution of the energy density of the photons, there is another equation that describes the change in the energy content of the material. This equation is

$$\frac{\partial u_m}{\partial t} = -\int_{4\pi} \tau_a \left(\frac{c}{4\pi}B(T_m) - I\right) d\Omega + Q_m \tag{40}$$

In Eq. 39 and Eq. 40, $\hat{\Omega}$ is the unit angle vector, $u_m$ is the material energy density, $\tau_t = \tau_s + \tau_a$[1] are the total, scattering, and absorption opacities with units of inverse length, $S$ is an external source of photon energy, $Q_m$ is an external source of material heating, and $B(T_m)$ is the black body function, which is defined as

$$B(T_m) = \frac{8\pi^5 k_B^4}{15h^3c^3}T_m^4 = aT_m^4 \tag{41}$$

where $T_m$ is the material temperature, $a$ is the black body constant, and $k_B$ is Boltzmann's constant.

## 8.2 Useful Quantities

Not only is Eq. 39 difficult to solve, but the intensity $I$ contains much more detailed information than is frequently needed to solve a particular problem. In fact, the coupling with the material in Eq. 40 is only through the radiation energy density $E_r$, which is the integral of $I$ over all angles, or

$$E_r = \frac{1}{c}\int_{4\pi} I\,d\Omega \tag{42}$$

[1]The total opacity is the sum of the other opacities only in the energy dependent case. Depending on the method used to calculate group averages, this may no longer be true.

Another useful quantity is the net flux of energy $\vec{F}_r$, which is the first angular moment of $I$, or

$$\vec{F}_r = \frac{1}{c}\int_{4\pi} \hat{\Omega} I d\Omega \tag{43}$$

The flux $\vec{F}_r$ can be used to compute the net power through a surface using

$$P = \int_S c\vec{F}_r \cdot \hat{n} dA \tag{44}$$

where $\hat{n}$ is the outward unit normal of the surface $S$.

## 8.3 Group Bounds

```
GROUP BOUNDS
  [LOG real TO real BY int, [SCALE real]]
  [LINEAR real TO real BY int, [SCALE real]]
  ...
END
```

Photon energy group bounds in Kelvin, unless otherwise specified. The photon energy spectrum is divided into regions. The package further subdivides regions into the actual photon groups, using either linear or logarithmic intervals within the region. `SCALE` is used as an opacity multiplier.

Photon energy `GROUP BOUNDS` are determined by dividing the energy spectrum into regions that the package further subdivides into the actual groups. The user can specify that a particular spectrum region is subdivided using linear or logarithmic intervals. For example, a specification

```
group bounds
   log 1. [ev] to 32. [ev] by 5
end
```

will cause the spectral region from 1.0 to 32.0 eV to be divided into five photon groups whose bounds are 1.0 to 2.0, 2.0 to 4.0, 4.0 to 8.0, 8.0 to 16.0, and 16.0 to 32.0 eV.

Recommendations for minimum and maximum energy bounds to be used in radiation hydrodynamics calculations may be determined by examining a few properties of the

Planck function. The Planck function [38] is given by

$$I(\vec{x},E,\hat{\Omega},t)dEd\Omega = \frac{2}{h^3c^3}\frac{E^3}{e^{E/kT}-1}dEd\Omega = aT^4\frac{15}{\pi^4}\frac{x^3}{e^x-1}dx\frac{d\Omega}{4\pi} \tag{45}$$

where

$$a = \frac{8\pi^5 k_B^4}{15h^3c^3} = \frac{4\sigma}{c} \tag{46}$$

and $h$ = Planck's constant (6.6252e-27 erg-s), $c$ = speed of light (2.99793e10 cm/s), $k_B$ = Boltzmann's constant (1.38042e-16 erg/K), and $\sigma$ = the Stefan-Boltzmann constant (5.6686e-5 erg/(cm$^2$-s-K$^4$)). The Planck function is written in a form that separates the value of the energy density, $aT^4$, from two factors that integrate to unity.

Reasonable values for the minimum and maximum energy boundaries can be estimated from the properties of the normalized Planck function:

$$g(x) = \frac{15}{\pi^4}\frac{x^3}{e^x-1} \tag{47}$$

which describes the shape of the equilibrium radiation energy spectrum for a given temperature $T$. Here $x = E/kT$. A plot of this function is shown in Figure 16.

The maximum of the Planck function is found by setting the first derivative of $g(x)$ to zero and solving for $x$.

$$\frac{dg(x)}{dx} = 0 = \frac{15}{\pi^4}\frac{x^2[(3-x)e^x-3]}{(e^x-1)^2} \tag{48}$$

or

$$3e^{-x} = 3-x \tag{49}$$

The maximum occurs at $x_{max}$ = 2.82144 and has the value $g_{max}$ = 0.21889. Any set of group bounds should include the value $E = 2.82k_BT$ within one of the groups.

An estimate for the lowest and highest group bound depends on the fraction of the energy spectrum one wishes to capture. Consider the integration domain bounded by the solution of

$$f \cdot g_{max} = g(x) = \frac{15}{\pi^4}\frac{x^3}{e^x-1} \tag{50}$$

The FWHM domain ($f = 0.5$) lies above $x = 1$ and this domain contains 75% of the energy density. More than 80% of the energy density lies between $x = 0.98$ and $x = 5.9$. Additionally, approximately 99% of the energy density lies between $x = 0.2$ and $x = 11$. A summary of group bound estimates is presented in Table 11.

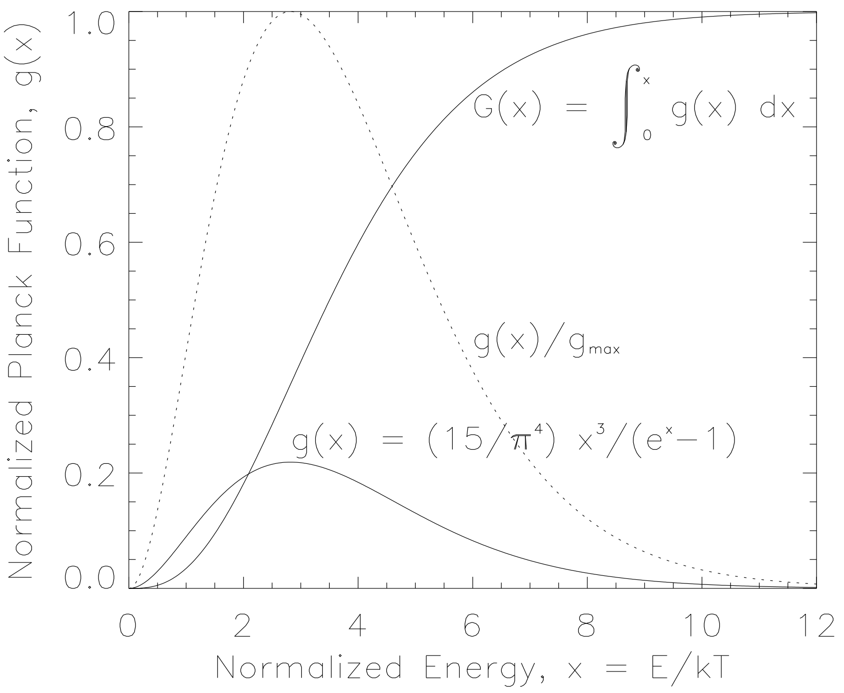

**Figure 16.** Plots of the normalized Planck function and its integral. Plot of the Planck function also scaled to 1 for ease of graphically determining lowest and highest group bounds.

Thus, if one has a notion of the range of radiation temperatures in a given simulation, e.g., a prescribed radiation temperature boundary condition or a radiation field in equilibrium with a material temperature, the lowest group bound should be approximately 0.1 to 0.2 times the lowest temperature and the highest group bound should be 10 to 12 times the highest temperature.

Non-equilibrium conditions, which this note does not address, may modify the above considerations. Also, intermediate group boundaries can be determined by absorption edges, emission lines, diagnostic filters, and so forth.

**Table 11.** Minimum and Maximum Energy Group Boundaries

| $f$ | Lowest group boundary $x_{low}$ | Highest group boundary $x_{high}$ | Minimum value of $g(x)$ $g_{min}$ | Fraction of energy density in integral |
|---|---|---|---|---|
| 0.50 | 1.15746 | 5.41158 | 0.10944 | 0.75357 |
| 0.40 | 0.98376 | 5.88425 | 0.08755 | 0.81714 |
| 0.30 | 0.81070 | 6.44278 | 0.06567 | 0.87297 |
| 0.20 | 0.62916 | 7.16701 | 0.04378 | 0.92208 |
| 0.10 | 0.42034 | 8.29990 | 0.02189 | 0.96482 |
| 0.05 | 0.28691 | 9.35041 | 0.01094 | 0.98363 |
| 0.01 | 0.12299 | 11.60880 | 0.00219 | 0.99705 |
| 0.001 | 0.03806 | 14.59893 | 0.00022 | 0.99973 |
|  | $x_{low} \approx \pi^2 \sqrt{\frac{f g_{max}}{15}}$ | $x_{high} \approx 6 - \ln(\frac{\pi^4 f g_{max}}{15})$ | $g_{min} = f \cdot g_{max}$ |  |

## 8.4 Linearized Diffusion

```
LINEARIZED DIFFUSION
   FLUX LIMITER = {LARSON [int] | SIMPLIFIED LEVERMORE POMRANING |
                   MINERBO | NONE}
   PARTIAL GROUP SOLVE [real]
   PURE EULERIAN
   AZTEC SET int

   GROUP BOUNDS
     [LOG real TO real BY int, [SCALE real]]
     [LINEAR real TO real BY int, [SCALE real]]
     ...
   END

   INITIAL CONDITIONS, [block-id,] UNIFORM TEMPERATURE real
   STEADY STATE INITIALIZATION

   VACUUM BOUNDARY, sideset
   REFLECTIVE BOUNDARY, sideset
   ALBEDO BOUNDARY, sideset, function-set
   TEMPERATURE SOURCE BOUNDARY, sideset, function-set-1, function-set-2
   VOLUME SOURCE, [block-id,] UNIFORM TEMPERATURE, real, function-set

   FLUX TALLY, "string-id", int, sideset, [sideset, ...,] [SCALE real]

   ... [other LINEARIZED DIFFUSION model keywords] ...
END
```

One of the main transport models in ALEGRA in the linearized flux limited diffusion. This model solves for the radiation energy density at the nodes of the mesh. The user must specify a `LINEARIZED DIFFUSION` block in the `RADIATION` block to select this method. All common keywords, such as boundary conditions, should be inside the `LINEARIZED DIFFUSION` block.

In the diffusion approximation the intensity $I$ is assumed to have the form

$$I(\vec{x},\hat{\Omega},t) = \frac{1}{4\pi}E_r + \frac{3}{4\pi}\hat{\Omega}\cdot\vec{F}_r \tag{51}$$

where $\vec{F}_r$ is the radiative flux defined by

$$\vec{F} = \int_{4\pi}\hat{\Omega}I d\Omega \approx -\frac{1}{3\tau_t}\nabla E \tag{52}$$

Eq. 51 and Eq. 52 lead to the diffusion equation, namely

$$\frac{1}{c}\frac{\partial E}{\partial t} - \nabla\cdot D\nabla E = \tau_a\left(4\pi B(T_m) - E\right) + S_E \tag{53}$$

where $D = \frac{1}{3\tau_t}$ is the diffusion coefficient.

The coupled diffusion and material energy equations form a nonlinear system. This method linearizes the system in a particular way so that a simple linear solve is possible. It is possible to get time steps that are too large for this method to handle. Always check to make sure that reducing the time step does not change the solution.

The diffusion equation is very easy to solve but is inaccurate in optically thin regions and where the gradient of the energy density is large. Flux limited diffusion is an improvement to fix these deficiencies at the cost of making the equations nonlinear. In flux limited diffusion, the equations are modified such that the factor of $1/3\tau_t$ in Eq. 52 and Eq. 53 is replaced with a nonlinear function of $E$. For several flux limiters, this nonlinear function is chosen to get the exact transport solution for a particular problem [33, 28]. There has been a fundamental change in the form of the equations; the transport equation (Eq. 39) is hyperbolic, implying that particles (and energy) travels at finite speeds. The time dependent diffusion equation is parabolic, allowing the particles to travel at infinite speed; a small change in one part of the problem immediately affects every other part of the problem.

## 8.5 Implicit Monte Carlo

```
KULL IMC
   {NUMBER OF PHOTONS int | PHOTONS PER ELEMENT int}

   GROUP BOUNDS
     [LOG real TO real BY int, [SCALE real]]
     [LINEAR real TO real BY int, [SCALE real]]
     ...
   END

   INITIAL CONDITIONS, [block-id,] UNIFORM TEMPERATURE real
   STEADY STATE INITIALIZATION

   VACUUM BOUNDARY, sideset
   REFLECTIVE BOUNDARY, sideset
   ALBEDO BOUNDARY, sideset, function-set
   TEMPERATURE SOURCE BOUNDARY, sideset, function-set-1, function-set-2

   VOLUME SOURCE, [block-id,] UNIFORM TEMPERATURE, real, function-set

   FLUX TALLY, "string-id", int, sideset, [sideset, ...,] [SCALE real]

   ... [other KULL IMC model keywords] ...
END
```

The implicit Monte Carlo (IMC) package in ALEGRA is from LLNL's KULL code. The user must specify a `KULL IMC` block in the `RADIATION` block to select this method. All common keywords, such as boundary conditions, should be inside the `KULL IMC` block.

Currently IMC only works on Eulerian Hex8 and Quad4 meshes, and for Cartesian geometries.

# 9 Emission Input

```
EMISSION
   {BLACKBODY | PLANCK | BREMSSTRAHLUNG}
   GROUP BOUNDS
      ...
   END
   [EMISSION ENERGY FLOOR, 0.0]
   [MAXIMUM EMISSION DENSITY, 0.0]
   [MINIMUM TEMPERATURE, 0.0]
   [NEWTON | BISECT]
   [MAXIMUM NEWTON ITERATIONS, 1]
   [TOLERANCE, 1.0e-6]
END
```

The emission model is a way to keep material heating under control by radiating photon energy using an approximate radiation emission model. The net radiation emission is typically a balance between energy emitted and energy absorbed,

$$\frac{de_m}{dt} = -\sum_g \left[R(T_m, g) - A(g)\right] \cdot f_\rho \cdot f_T \tag{54}$$

for each group $g$ and each material $m$. Here $\frac{de_m}{dt}$ is the time rate of change of the material specific energy, $T_m$ is the material temperature, $g$ is the group number, $R(T_m, g)$ is a radiation emission rate function, and $A(g)$ is a radiation absorption rate function. $f_\rho$ and $f_T$ are density and temperature dependent multipliers described below. Simple one-dimensional considerations allow us to approximate the probability of escape after a distance $x$ as $e^{-\tau_{ag} d}$ where $\tau_{ag} d$ is known as the optical depth, $\tau_{ag}$ is the absorption opacity, and the distance $d$ is the mean distance to void. We approximate the absorption rate as the emission rate times the probability of reabsorption (i.e., not escaping).

$$A(g) = \left(1 - e^{-\tau_{ag} d}\right) R(T_m, g) \tag{55}$$

Thus, the net emission rate becomes the total emission rate reduced by the probability of escape.

$$\frac{de_m}{dt} = -\sum_g e^{-\tau_{ag} d} R(T_m, g) \cdot f_\rho \cdot f_T \tag{56}$$

Because ALEGRA is an energy-based code, the actual discrete algorithm approximates the material temperature variation in terms of the specific energy.

$$T_m^{n+1} = T_m^n + \frac{(e_m^{n+1} - e_m^n)}{C_v^n} \tag{57}$$

where $C_v$ is the material heat capacity. The first Newton iterate is used to calculate the change in $e_m$. The derivative of $R(T_m, g)$ with respect to temperature is then required. The Newton correction is intended to provide a mechanism for avoiding excessive cooling.

## 9.1 Emission Model Options

There are currently two emission options within ALEGRA:

- `BLACKBODY` or `PLANCK` - Blackbody emission model using the Planck function
- `BREMSSTRAHLUNG` - thermal Bremsstrahlung emission model

One or the other emission model must be specified.

### 9.1.1 Planck Emission

{`BLACKBODY` | `PLANCK`}

These two keywords both refer to the same emission model. The `BLACKBODY` or `PLANCK` emission model defines the radiation emission rate function in units of *energy/mass/time* as

$$R(T_m, g) = \sigma_{ag} B(T_m, g) = \frac{\tau_{ag}}{\rho} B(T_m, g) \tag{58}$$

where $\sigma_{ag}$ is the group absorption cross section in units of *area/mass*, $\tau_{ag} = \rho\sigma_{ag}$ is the group absorption opacity in units of $1/length$ ($\tau_{ag}$ is the variable that ALEGRA stores), $\rho$ is the mass density, and $B(T_m, g)$ is the Planck function integral in units of $energy/area/time$ given by

$$B(T_m, g) = 4\pi \int_{\varepsilon_g}^{\varepsilon_{g+1}} B(\varepsilon, T_m) d\varepsilon = 4\pi \int_{\varepsilon_g}^{\varepsilon_{g+1}} \frac{2}{h^3c^2} \frac{\varepsilon^3}{\exp\left(\frac{\varepsilon}{kT_m}\right) - 1} d\varepsilon \qquad (59)$$

This emission model requires that all `MATERIAL`s have an opacity model specified. Scattering opacity values are ignored. See ”Opacity Models” in Section 12.7.

### 9.1.2 Bremsstrahlung Emission

`BREMSSTRAHLUNG`

The Bremsstrahlung emission model is pertinent to astrophysical X-ray cluster cooling flows and is defined by the White and Sarazin prescription for a half-solar abundance cooling function [23]. The radiation emission rate function is given by

$$R(T_m, g) = \frac{10^{-35}\rho}{(\mu m_p)^2} \begin{cases} 2.35 \cdot e^{-\left(\frac{T_m}{3.5\cdot 10^5}\right)^{4.5}} + \frac{5. \cdot e^{-\left(\frac{T_m}{2.8\cdot 10^6}\right)^{4.4}}}{T_m^{0.17}} \\ \quad + \frac{0.05 \cdot e^{-\left(\frac{T_m}{1.55\cdot 10^7}\right)^{4}}}{T_m^{0.08}} + 2.4 \cdot 10^{-5}\sqrt{T_m} & , \quad T_m > 10^5 \\ 3.18 \left(\frac{T_m}{10^5}\right)^{1.6} - \frac{150}{T_m}, & \text{otherwise} \end{cases} \qquad (60)$$

It is recommended that the `BISECT` option be enabled for this emission model since the cooling function is not monotonic with temperature.

`BREMSSTRAHLUNG` does not require any opacity models and should only be called with a single energy group. The limits of this group are irrelevant. Use of the probability of escape does require an opacity model however, unless the `OPTICAL MULTIPLIER` is set to zero.

## 9.2 Group Boundaries

```
GROUP BOUNDS
   LOG real TO real BY int, [SCALE real (1.0)]
   LINEAR real TO real BY int, [SCALE real (1.0)]
   ...
END
```

This required keyword defines the photon energy group bounds in Kelvin, unless otherwise specified. The photon energy spectrum is divided into regions. The package further subdivides regions into the actual photon groups using either `LINEAR` or logarithmic (`LOG`) intervals within the region. The optional `SCALE` keyword is used as an opacity multiplier. Its default value is 1.0

For example, a specification of

```
group bounds
   log 1. [ev] to 32. [ev] by 5
end
```

will cause the spectral region from 1.0 to 32.0 eV to be divided into five photon groups whose bounds are 1.0 to 2.0, 2.0 to 4.0, 4.0 to 8.0, 8.0 to 16.0, and 16.0 to 32.0 eV.

The `GROUP BOUNDS` input is required for both models.

## 9.3 Algorithm Control

### 9.3.1 Emission Multiplier

```
EMISSION MULTIPLIER real (1.0)
```

If this optional keyword is present then the net emission rate is scaled by this factor. The main purpose of this factor is to allow the user to perform sensitivity studies relative to the natural emission rate.

### 9.3.2 Optical Multiplier

```
OPTICAL MULTIPLIER real (1.0)
```

If this optional keyword is present then the optical depth $\tau_{ag}x$ in the probability of escape is scaled by this factor. This factor may be used for sensitivity studies or to adjust the estimate of the mean-distance-to-void $x$ should there be a systematic error.

### 9.3.3 Mean Distance to Void

```
MDTV, {RMS (default) | MIN MAX}
```

This optional keyword selects the method used to estimate of the mean-distance-to-void $d$. The default method is `RMS`.

The `RMS` method computes the root-mean-square size of the material and the mesh about the mean material and mesh locations for each orthogonal coordinate, respectively. The RMS mesh sizes are used as weights for the RMS material sizes in the standard distance formula to account for one-dimensional simulations.

$$\begin{aligned} d &= 2\sqrt{w_x^2 x_{rms}^2 + w_y^2 y_{rms}^2 + w_z^2 z_{rms}^2} \\ w_x &= X_{rms}/\sqrt{X_{rms}^2 + Y_{rms}^2 + Z_{rms}^2} \\ w_y &= Y_{rms}/\sqrt{X_{rms}^2 + Y_{rms}^2 + Z_{rms}^2} \\ w_z &= Z_{rms}/\sqrt{X_{rms}^2 + Y_{rms}^2 + Z_{rms}^2} \end{aligned} \tag{61}$$

where $x_{rms}$, $y_{rms}$, and $z_{rms}$ are the root-mean-square size of the material, and $X_{rms}$, $Y_{rms}$, and $Z_{rms}$ are the root-mean-square size of the mesh.

The `MIN MAX` method uses the difference between the maximum and minimum material coordinates and the difference between the maximum and minimum mesh coordinates as the size of the material and mesh respectively. The min-max mesh sizes are used as weights for the min-max material sizes in the standard distance formula to account for one-dimensional simulations.

$$\begin{aligned}
d &= \sqrt{w_x^2(x_{max}-x_{min})^2+w_y^2(y_{max}-y_{min})^2+w_z^2(z_{max}-z_{min})^2} \qquad (62)\\
w_x &= (X_{max}-X_{min})/\sqrt{(X_{max}-X_{min})^2+(Y_{max}-Y_{min})^2+(Z_{max}-Z_{min})^2}\\
w_y &= (Y_{max}-Y_{min})/\sqrt{(X_{max}-X_{min})^2+(Y_{max}-Y_{min})^2+(Z_{max}-Z_{min})^2}\\
w_z &= (Z_{max}-Z_{min})/\sqrt{(X_{max}-X_{min})^2+(Y_{max}-Y_{min})^2+(Z_{max}-Z_{min})^2}
\end{aligned}$$

where $x_{min}$, $y_{min}$, $z_{min}$, $x_{max}$, $y_{max}$, and $z_{max}$ are the minimum and maximum values where material is present, and $X_{min}$, $Y_{min}$, $Z_{min}$, $X_{max}$, $Y_{max}$, and $Z_{max}$ are the minimum and maximum values of the mesh.

For uniform meshes the `RMS` method yields a smaller mean-distance-to-void than the `MIN MAX` method. The `MIN MAX` method is better for highly non-uniform meshes.

### 9.3.4 Emission Energy Floor

```
EMISSION ENERGY FLOOR real (0.0)
```

If this optional keyword is present then the available energy in an element will not drop below the percentage given by the `EMISSION ENERGY FLOOR`. This floor may help to avoid the problem of an element obtaining a negative temperature by emitting more energy than is available. The available energy is defined by a linear extrapolation of the material specific internal energy down to zero temperature.

$$e_{avail} = e_m(T_m) - e_m(0) = C_v(T_m - 0) = C_v T_m \qquad (63)$$

For example, to keep the energy available in an element from dropping below 1% of its starting value, one would use the `EMISSION ENERGY FLOOR` as follows:

```
emission
   ...
   emission energy floor 0.01
end
```

Emission down to this energy floor is allowed.

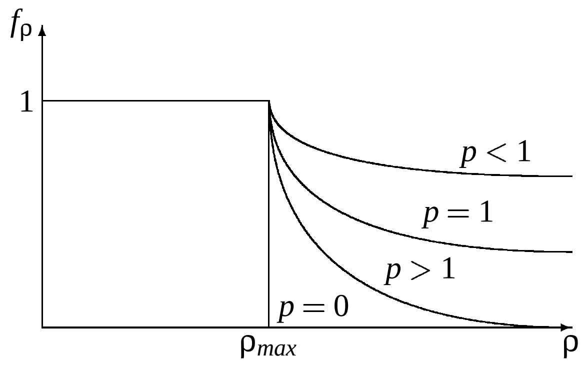

**Figure 17.** Maximum Emission Density Multiplier.

### 9.3.5 Maximum Emission Density

```
MAXIMUM EMISSION DENSITY real (0.0) [POWER real (0.0)]
```

If this optional keyword is present and the density value is non-zero, then only material densities less than the input value will emit at the prescribed rate. A zero value implies emission at all densities. The assumption is that above this density the material is optically thick and some portion of the emitted energy is immediately reabsorbed. The multiplier of the net emission rate when the density is greater than the specified limit is given by the following expression where $p$ is the specified `POWER`.

$$f_\rho = \begin{cases} 1, & \rho \le \rho_{max} \\ 0, & \rho > \rho_{max}, p = 0 \\ \left(\frac{\rho_{max}}{\rho}\right)^p, & \rho > \rho_{max}, p > 0 \end{cases} \tag{64}$$

Graphically the multiplier can be displayed as in Figure 17. This option should only be used when the probability of escape factor is insufficient to limit the net emission rate at high densities.

Example:

```
emission
   ...
   max emission density 0.1
end
```

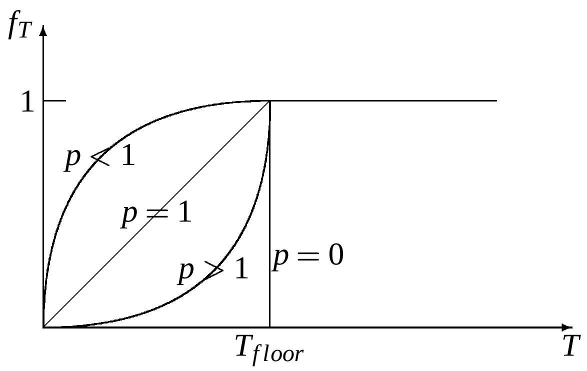

**Figure 18.** Minimum Emission Temperature Multiplier.

### 9.3.6 Minimum Emission Temperature

```
MINIMUM TEMPERATURE real (0.0) [POWER real (0.0)]
```

The optional `MINIMUM TEMPERATURE` specification is such that if emission model would cool an element below this temperature, the emission model turns itself off. Emission down to this minimum temperature is allowed. If the temperature value is non-zero, then only material temperatures greater than the input value will emit at the prescribed rate. A zero value implies emission at all temperatures. The assumption is that the material is too cold to emit. The multiplier of the net emission rate when the temperature is less than the specified limit is given by the following expression where $p$ is the specified `POWER`.

$$f_T = \begin{cases} 1, & T \geq T_{floor} \\ 0, & T < T_{floor}, p = 0 \\ \left(\frac{T}{T_{floor}}\right)^p, & T < T_{floor}, p > 0 \end{cases} \tag{65}$$

Graphically the multiplier can be displayed as in Figure 18. This option should only be used when the probability of escape factor is insufficient to limit the net emission rate at low temperatures.

Example:

```
emission
   ...
   min temperature 933. $ Kelvin
end
```

### 9.3.7 Maximum Energy Change

```
MAXIMUM ENERGY CHANGE real (0.90)
```

This optional keyword allows the user to limit the maximum change in the material specific internal energy due to emission. This is accomplished by limiting the simulation time step so that the amount of energy radiated in a given simulation cycle complies with this constraint. The hope is that by limiting the energy change, other material properties can also change sufficiently rapidly, thereby resulting in a more accurate simulation. The default value allows a 90% change in the specific internal energy. The time step computed by this feature is reported in the global tally `DT_EMISSION`.

Example:

```
emission
   ...
   max energy change 0.40  $ fraction
end
```

### 9.3.8 Solution Method

```
{NEWTON (default) | BISECT}
```

If the `NEWTON` keyword is present, it explicitly forces use of the default Newton iteration method.

If the `BISECT` keyword is present then the Newton algorithm will incorporate a bisection step into it. Use this for the `BREMSSTRAHLUNG` emission option as the cooling function is non-monotonic. This option does not need to be specified for the `BLACKBODY` or `PLANCK` option.

### 9.3.9 Maximum Newton Iterations

```
MAXIMUM NEWTON ITERATIONS int (1)
```

Optionally specifies the maximum number of Newton iterations. A warning will occur if the maximum number of Newton iterations is exceeded and the solution tolerance remains less than the tolerance specified by the `TOLERANCE` keyword. Quite often it is sufficient to give this parameter a value of just 2.

### 9.3.10 Newton Iteration Tolerance

```
TOLERANCE real (1.0e-6)
```

Optional accuracy level setting for Newton iteration solution of emission energy equation. The default value is 1.0e-6.

# 10 Two Temperature Input

```
TWO TEMPERATURE
   [COUPLING (ON/OFF)]
   [MAXIMUM ENERGY CHANGE real]
   [MAXIMUM TEMPERATURE CHANGE real]
END
```

The `TWO TEMPERATURE` keyword group specifies control of the electron/ion energy equilibration routine and also flags the various physics models, e.g., Joule heating or radiation emission and absorption, to interact energetically with the electrons in the system. Other physics models must solve two equations instead of only one equation to describe physical processes, e.g., electrons and ions each have a $PdV$ work term contributing to separate internal energy equations. There is still only one artificial viscosity work term and it couples to the ions. In addition, separate electron and ion thermal conduction equations are solved. Some of the extra variables needed to implement `TWO TEMPERATURE` physics are listed in Table 6 on page 25. The use of this keyword group presently requires specification of a 2T equation of state. These are described in Section 12.3 on page 114.

The electron/ion coupling is based upon the Spitzer [41] electron-ion equilibration time given by

$$\tau_{ei} = \frac{3 m_i m_e k_B^{3/2}}{8(2\pi)^{1/2} n_i Z^2 e^4 \log(\Lambda)} \left( \frac{T_e}{m_e} + \frac{T_i}{m_i} \right)^{3/2} \tag{66}$$

therefore, *a strong warning is issued to users that 2T options are only valid in the Spitzer regime when materials are ionized*.

Separate electron and ion energy transfer equations are solved. The coupling terms transfer energy between the electrons and the ions and are given by

$$\rho \frac{\partial e_e}{\partial t} = -Q_{ei} = -\rho C_{ve} \frac{T_e - T_i}{\tau_{ei}} \tag{67}$$

$$\rho \frac{\partial e_i}{\partial t} = Q_{ei} = \rho C_{ve} \frac{T_e - T_i}{\tau_{ei}} \tag{68}$$

These equations are written in terms of energy loss to the electrons. $e_e$ and $e_i$ are the electron and ion specific internal energies and $C_{ve}$ and $C_{vi}$ are the electron and ion specific heats, respectively.

In terms of the electron and ion temperatures, these equations may be written

$$\frac{\partial T_e}{\partial t} = -\frac{T_e - T_i}{\tau_{ei}} \tag{69}$$

$$\frac{\partial T_i}{\partial t} = -\frac{C_{ve}}{C_{vi}}\frac{T_i - T_e}{\tau_{ei}} = -\frac{T_i - T_e}{\tau_{ie}} \tag{70}$$

Typically in the Spitzer regime, $C_{ve}/C_{vi} \approx \bar{Z}$. This leads to the result that $\tau_{ie} = \tau_{ei}/\bar{Z}$

## 10.1 General Initial Conditions

### 10.1.1 Diatom

```
DIATOM
   PACKAGE name
      MATERIAL int
      INSERT shape
         insert subkeywords
      ENDI
      ...
      [additional package subkeywords]
      ...
   ENDP
   ...
   [additional diatom packages]
   ...
ENDDIATOM
```

The `DIATOM` capability is a method of inserting material into a mesh. For complete details see the primary ALEGRA User's Manual [12]. The basic diatom capability allows the user to enter initial material density and temperature profiles using the `DEN*SITY` and `T*EMPERATURE` package subkeywords. The `DEN*SITY` subkeyword is still used to specify density profiles for the `TWO TEMPERATURE` case. The `T*EMPERATURE` subkeyword may be used to specify temperature profiles provided the initial electron and ion temperature profiles are the same. Separate electron and and ion temperature profiles may also specified. Table 12 lists the additional PACKAGE subkeywords available with TWO TEMPERATURE physics.

Table 12: PACKAGE Subkeywords for DIATOM Input

| Subkeyword | Argument | Meaning |
|---|---|---|
| ELE_T*EMPERATURE | real or T int | Initial electron temperature of the material in the insertion set and all subsequent insertions sets. Must reset the electron temperature of subsequent sets to desired the value or zero to obtain the initial ALEGRA values. A table (function) number can also be input to specify the electron temperature as a time varying function or for use with a graded option. |
| ION_T*EMPERATURE | real or T int | Initial ion temperature of the material in the insertion set and all subsequent insertions sets. Must reset the ion temperature of subsequent sets to desired the value or zero to obtain the initial ALEGRA values. A table (function) number can also be input to specify the ion temperature as a time varying function or for use with a graded option. |

Example:

```
diatoms
  package "left"
    material = 1
    ion_t    = t1      $ insert using a function
    ele_t    = t1
    agraded, p1 0. 0., p2 100. 0.
    numsub   = 10.
    insert box,
      p1 =    0.0  2.25
      p2 =   50.0  7.25
    endinsert
  endpackage
  package "right"
```

```
    material = 2
    ion_t    = 600.    $ insert using a number
    ele_t    = 600.
    numsub   = 10.
    insert box,
      p1 =   50.0  2.25
      p2 =  100.0  7.25
    endinsert
  endpackage
enddiatoms

function 1
    0. 300.
   51. 300.
  100. 600.
end
```

## 10.2 Algorithm Control

### 10.2.1 Coupling

```
COUPLING [ON | OFF]
```

The `COUPLING` keyword can be used to disable the electron/ion equilibration. The default is `ON`.

### 10.2.2 Density Floor

```
DENSITY FLOOR real
```

The `DENSITY FLOOR` keyword can be used to enhance the fraction of the electron or ion energy that is transferred to the opposite species. This is accomplished by restricting the density used in determining the equilibration time to be the maximum of the actual density and the `DENSITY FLOOR`. Since the density appears in the denominator of the equilibration time, this keyword effectively shortens the equilibration time.

### 10.2.3 Maximum Energy Change

```
MAXIMUM ENERGY CHANGE real
```

The `MAXIMUM ENERGY CHANGE` keyword can be used to limit the fraction of the electron or ion energy that is transferred to the opposite species. This is accomplished by limiting the simulation time step so that the amount of energy radiated in a given simulation cycle complies with this constraint. The hope is that by limiting the energy change, other material properties can also change sufficiently rapidly, thereby resulting in a more accurate simulation. The default is up to 90% of a species' energy can be transferred to the opposite species. The time step computed by this feature is reported in the global tally `DT_TWO_T`.

### 10.2.4 Maximum Temperature Change

```
MAXIMUM TEMPERATURE CHANGE real
```

The `MAXIMUM TEMPERATURE CHANGE` keyword can be used to limit the rate of change per simulation cycle of the electron or ion temperature. The rate of change is limited by decreasing the time step. The default is 0.0 implying unrestricted temperature change. The time step computed by this feature is reported in the global tally `DT_TWO_T`.

## 10.3 Thermal Conduction Boundary Conditions

`TWO TEMPERATURE` physics uses separate heat conduction equations for the electrons and ions. These two equations may have similar or different boundary conditions depending on the situation at hand. Therefore, the `THERMAL CONDUCTION` boundary conditions are modified to allow for application of the various boundary conditions to either electrons or ions or both.

### 10.3.1 No Heat Flux

```
NO [ION | ELECTRON] HEAT FLUX, sideset
```

This keyword allows the user to specify zero heat flux through a surface defined by the `sideset` argument.

If `TWO TEMPERATURE` physics is enabled, then the optional words `ION` or `ELECTRON` can be included to restrict which species the boundary condition affects. If no species is explicitly specified, then the boundary condition applies to both species.

### 10.3.2 Prescribed Heat Flux

```
PRESCRIBED [ION | ELECTRON] HEAT FLUX, sideset, function-set
```

This keyword allows the user to prescribe the heat flux on a surface defined by the `sideset` keyword as a function of time.

If `TWO TEMPERATURE` physics is enabled, then the optional words `ION` or `ELECTRON` can be included to restrict which species the boundary condition affects. If no species is explicitly specified, then the boundary condition applies to both species.

### 10.3.3 Temperature BC

```
[ION | ELECTRON] TEMPERATURE BC, sideset, function-set
```

This keyword allows the user to specify the temperature on a surface defined by the `sideset` keyword as a function of time.

If `TWO TEMPERATURE` physics is enabled, then the optional words `ION` or `ELECTRON` can be included to restrict which species the boundary condition affects. If no species is explicitly specified, then the boundary condition applies to both species.

# 11 Aztec Input

```
AZTEC [int]
   ...
END
```

Both the TRANSIENT MAGNETICS and the THERMAL CONDUCTION packages use the Aztec library [46] to solve the discrete partial differential equations that govern their physics. Aztec is an iterative solver library with many options for solving linear systems of equations. Aztec includes a number of Krylov iterative methods such as conjugate gradient (CG), generalized minimum residual (GMRES), and stabilized biconjugate gradient (BiCGSATB).

The AZTEC keyword set may appear as many times as desired. Different physics packages may utilize different AZTEC control sets as specified by using

```
AZTEC SET id(integer)
```

in the specific physics input section. The default AZTEC SET id-number is 0 which defaults to the conjugate gradient algorithm with symmetric diagonal scaling and other default options described below. The developers do not recommend that you blindly use the default settings as they are most likely not optimal for your particular simulations. Multigrid technology found in the associated ML package is required to achieve scalable solve times for very large problems. The Aztec [46] and ML [44] documentation give a descriptions of options. The three-dimensional magnetic field discretization and multigrid solution research and development can be reviewed in several references [5, 6, 7, 21, 39].

Most but not all AZTEC options have been implemented in the ALEGRA interface. However only the most important and useful options are listed in this section.

Example:

```
aztec 1      $ 3D simulation ONLY, mag control
   solver    = cg
   scaling   = none
   conv norm = rhs
   output    = none
   max iter  = 1000    $ something is wrong after a few hundred iterations
   tol       = 1.e-10  $ this may need to be reduced (problem specific)
   multilevel
```

```
      fine sweeps             = 1
      fine smoother           = Hiptmair
      coarse smoother         = LU
      multigrid levels        = 10
      interpolation algorithm = AGGREGATION
      hiptmair subsmoother    = GAUSS SEIDEL  $ for serial runs
    $ hiptmair subsmoother    = MLS           $ for parallel runs > 4PE
      SMOOTH PROLONGATOR
   end
end
```

In the descriptions below the first listed option is the ALEGRA default. These are not necessarily the same as the AZTEC defaults.

## 11.1 Basic Aztec Input

```
AZTEC [int]
   [basic aztec keyword]
   ...
END
```

This section describes the basic (non-multilevel) Aztec control options.

### Solver

```
SOLVER, {CG (default) | GMRES | CGS | TFQMR | BICGSTAB | LU |
         GMRESR | FIXED PT | SYMMLQ}
```

Note that CG (conjugate gradient) is applicable only to physics packages which are based on symmetric operators. These include 2D and 3D TRANSIENT MAGNETICS, and the THERMAL CONDUCTION package. The LU option uses the familiar LU-decomposition direct solution method, which is only available in serial mode and will likely quickly exhaust memory for sizable problems. For small serial test simulations, LU is a good option. CG by itself will not work effectively for 3D TRANSIENT MAGNETICS solutions.

### Scaling

```
SCALING, {SYM_DIAG (default) | NONE | JACOBI | BJACOBI | ROW_SUM |
          SYM_ROW_SUM}
```

The SYM_DIAG option is only available for physics packages based on symmetric operators, such as 2D TRANSIENT MAGNETICS and the THERMAL CONDUCTION package. It is highly recommended for these.

### Preconditioner

```
PRECONDITIONER, {NONE (default) | JACOBI | NEUMANN | LS | SYM_GS |
                 DOM_DECOMP}
```

The SYM_GS option has proven quite effective for those physics packages that are based on symmetric operators. However, it has been discovered that it is possible for the actual solution accuracy to vary significantly with the preconditioner option for a given fixed tolerance value. In particular the JACOBI and SYM_GS preconditioners used with conjugate gradient may cause a significantly less accurate answer than would be obtained without the preconditioner for the same tolerance value. See the ignore scaling option.

### Subdomain Solver

```
SUBDOMAIN SOLVER, {LU | ILUT (default) | ILU | RILU | BILU | ICC}
```

ICC can be used with symmetric operators.

### Convergence Norm

```
CONVERGENCE NORM, {R0 (default) | RHS | ANORM | NOSCALE | SOL |
                   expected values}
```

Various norm options are available. Users of the code MUST ensure that their solutions are accurate relative to the chosen convergence norm and the chosen tolerance value. (The AZTEC weighted option is not supported in ALEGRA).

### Output

```
OUTPUT, {NONE (default) | ALL | WARNINGS | LAST | int>0}
```

Note that no Aztec output is the default for ALEGRA. You must specifically request Aztec output if you want Aztec to tell you about the iterative solve. Transient magnetics will output Aztec information to the HISPLT file.

### Maximum Iterations

```
MAXIMUM ITERATIONS, int>0 (500)
```

### Tolerance

```
TOLERANCE, real (1.e-12)
```

WARNING! This default may not be appropriate for some `CONVERGENCE NORM` options and physics problems. The user is responsible to convince himself that the tolerance specified is sufficient for an otherwise FIXED set of AZTEC options. See the discussion under preconditioners.

## 11.2 Multilevel Input

```
AZTEC [int]
   [basic aztec keyword]
   ...
   MULTILEVEL
      [multilevel keyword]
      ...
   END
END
```

This section describes the multilevel Aztec control options.

**Multigrid Levels**

```
MULTIGRID LEVELS, int (10)
```

**Fine Sweeps**

```
FINE SWEEPS, int (2)
```

The number of pre and post smoother sweeps to apply at each level.

**Fine smoother**

```
FINE SMOOTHER, {GAUSS SEIDEL | JACOBI | LU | AZTEC int (0) |
                HIPTMAIR (Hcurl only) | MLS}
```

Default is automatically determined depending on matrix type. This is the type of smoother applied on each of the fine levels. The `AZTEC int` is the `AZTEC SET int` id to be used by the fine smoother.

**Coarse Sweeps**

```
COARSE SWEEPS, int (2)
```

Number of solver sweeps to apply at the coarse level if the coarse smoother is an iterative solver.

**Coarse smoother**

```
COARSE SMOOTHER, {GAUSS SEIDEL | JACOBI | LU (default) | AZTEC int (0) |
                  HIPTMAIR (Hcurl only) | MLS}
```

The `AZTEC int` is the `AZTEC SET int` id to be used by the coarse smoother.

### Interpolation Algorithm

```
INTERPOLATION ALGORITHM, {UC AGGREGATION | MIS AGGREGATION |
                          UC MIS AGGREGATION (aka AGGREGATION, default)}
```

### Verbose

```
VERBOSE, int (0)
```

Positive integer gives level of multigrid verbose output. Value ranges from 0 to 10. Used primarily to debug ML.

### Smooth Prolongator

```
SMOOTH PROLONGATOR (off)
```

Smooth Reitzinger/Schoeberl prolongator for Hiptmair. A useful option for 3D magnetics.

### Hiptmair Subsmoother

```
HIPTMAIR SUBSMOOTHER, {GAUSS SEIDEL (default) | MLS}
```

This is the subsmoother uses by Hiptmair smoothing.

# 12 Materials and Material Models Input

## 12.1 Materials

The MATERIAL keyword is explained fully the basic ALEGRA manual [12]. In this version of ALEGRA, some of the material models get some of their information from the MATERIAL keyword block, namely the atomic composition as defined by the NUMBER OF ELEMENTS keyword. This allows each material to be treated consistently and with a minimum of input required from the user. Currently, QEOS requires the material composition to be entered here. The XSN material composition may be entered here or under the XSN model. The material input format is:

```
MATERIAL int [string]
   MODEL int
   [MODEL int]
   variable_name real
   [variable_name real]
   [NUMBER OF ELEMENTS int
      ELEMENT int, MASS real, FRACTION real
      ...
    END]
END
```

Example of material input:

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

material 201 "Air"
   model 11      $ EOS
   model 12      $ XSN
   number of elements 3
      element  7, mass 14.007, fraction 0.7809
      element  8, mass 16.000, fraction 0.2095
      element 18, mass 39.948, fraction 0.0096
   end
end
```

## 12.2 Material Models

```
MODEL int model_name
   parameter [int|real]
   parameter = [int|real]
   ...
END
```

The following sections describe the various material models that support HEDP applications. In each of the material model descriptions that follow, two tables are presented for each `MODEL`. The first table lists and defines the user specifiable input parameters. These parameters are to be listed withing the `MODEL ... END` keyword block in the input deck. The second table lists the registered code variables input to and output from the various models. These variables are available for plotting and are to be listed within the `PLOT VARIABLES ... END` keyword block in the input deck.

A list of the currently available models is provided in Table 13. The models are categorized into general model types.

**Table 13.** Material Model Types and Model Names.

| General Material Model Type | Model Name |
|---|---|
| Two-Temperature Equation of State | `IDEAL GAS TWOT`<br>`LANL SESAME`<br>`LLNL QEOS` |
| Combined Electrical/Thermal Conductivity | `LMD`<br>`SPITZER` |
| Electrical Conductivity | `EC ANOMALOUS`<br>`EC KNOEPFEL` |
| Thermal Conductivity | `CONSTANT THERMAL CONDUCTIVITY`<br>`PLASMA`<br>`POLYNOMIAL` |
| Opacity | `KRAMERS FITTED`<br>`XSN`<br>`TABULAR OPACITY` |
| Ionization State | `SAHA IONIZATION` |

The material models listed in Table 14 are considered to be obsolete and may be deleted from ALEGRA in the near future.

**Table 14.** Obsolete Material Model Types and Model Names.

| General Material Model Type | Model Name |
|---|---|
| Electrical Conductivity | `KEC SESAME` |
| Thermal Conductivity | `KTC SESAME` |
| Ionization State | `KZB SESAME` |

## 12.3 Two-Temperature Equation of State Models

Materials can support `TWO TEMPERATURE` physics based upon their equation of state models. The equation of state must be able to accept separate ion and electron specific internal energies and return separate ion and electron temperatures $T_i$, and $T_e$. Both the ion and electron specific internal energies are normalized relative to the total material mass density. In this way the two specific internal energies may be summed to determine the total material specific internal energy, $e = e_i + e_e$. There is no single material temperature unless $T_i = T_e$.

`TWO TEMPERATURE` equations of state compute the ion and electron pressures, as well as the total material pressure, $P = P_i + P_e$. The total pressure is used to compute the hydrodynamic response of the single fluid due to any pressure gradients and also to determine the material sound speed used in numerical Courant conditions. The individual ion and electron pressures are used to compute the $PdV$ work for the respective species.

Finally, `TWO TEMPERATURE` equations of state compute separate ion and electron specific heats, $C_{vi}$ and $C_{ve}$, for use in the separate ion and electron `THERMAL CONDUCTION` equations and for use with the ion-electron equilibration model and the radiation emission model.

### 12.3.1 Ideal Gas TwoT

```
MODEL model_id_number IDEAL GAS TWOT
   [parameter = value]
   ...
END
```

The `IDEAL GAS TWOT` model computes the pressure, temperature and the sound speed for the material using standard ideal gas equations for the ions and electrons.

$$P_i = \rho(\gamma - 1)(e_i - e_0) \quad \text{and} \quad P_e = \rho(\gamma - 1)(e_e - e_0) \tag{71}$$

$$T_i = \frac{e_i - e_0}{C_{vi}} \quad \text{and} \quad T_e = \frac{e_e - e_0}{C_{ve}} \tag{72}$$

$$C_s = \sqrt{\frac{\gamma P}{\rho}} = \sqrt{\frac{\gamma(P_i + P_e)}{\rho}} \tag{73}$$

$$C_{ve} = \bar{Z} C_{vi} \tag{74}$$

- Modules: material_libs/standard_models
  - matmod_idlgas2.h
  - matmod_idlgas2.C
- Physics:
  - all physics options that specify `TWO TEMPERATURE`

**Table 15.** Input Parameters for `IDEAL GAS TWOT`

| Parameter Name | Type | Description |
|---|---|---|
| `RHO REF` | real | Density at the reference state, $\rho_0$, used to set the density if not defined by the `MATERIAL` keyword. (optional) |
| `TREF` | real | Absolute temperature at the reference state, $T_{ref}$, used to set the temperature if not defined by the `MATERIAL` keyword. (optional) |
| `CV` | real | Specific heat at constant volume, $C_v$, computed from the initial $\rho$, $T$, and $\gamma$, if not defined. (optional) |
| `FIXED ZBAR` | real | Fixed average ionization state, $\bar{Z}$, will override any ionization model, if set. (optional) |
| `GAMMA` | real | Ratio of specific heats, $\gamma = C_p/C_v$. (required) |
| `E SHIFT` | real | Arbitrary shift of reference energy. (optional) |

**Table 16.** Registered Plot Variables for `IDEAL GAS TWOT`

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | real | `INPUT` | Material density, $\rho$ |
| `ENERGY` | real | `OUTPUT` | Total specific energy, $e = e_i + e_e$ |
| `PRESSURE` | real | `OUTPUT` | Total pressure, $P = P_i + P_e$ |
| `SOUND_SPEED` | real | `OUTPUT` | Sound speed, $C_s$, based on total pressure and density |
| `ZBAR` | real | `IOPUT` | Average ionization state, output only if `FIXED ZBAR` defined. |
| `ION_ENERGY` | real | `IOPUT` | Ion specific energy, $e_i$ |
| `ION_PRESSURE` | real | `OUTPUT` | Ion pressure, $P_i$ |
| `ION_TEMPERATURE` | real | `OUTPUT` | Ion temperature, $T_i$ |
| `ION_SPECIFIC_HEAT_VOL` | real | `OUTPUT` | Ion specific heat, $C_{vi} = C_v$ |
| `ELECTRON_ENERGY` | real | `IOPUT` | Electron specific energy, $e_e$ |
| `ELECTRON_PRESSURE` | real | `OUTPUT` | Electron pressure, $P_e$ |
| `ELECTRON_TEMPERATURE` | real | `OUTPUT` | Electron temperature, $T_e$ |
| `ELECTRON_SPECIFIC_HEAT_VOL` | real | `OUTPUT` | Electron specific heat, $C_{ve} = \bar{Z} C_v$ |

### 12.3.2 LANL Sesame

```
MODEL model_id_number LANL SESAME
   [parameter = value]
   ...
END
```

The SESAME tables produced by Los Alamos National Laboratories [31, 20] contain tabulated values of energy and pressure information as functions of temperature and density. This information is available with ion and electron information combined (series 301 tables), or in many instances, separate (series 303 through 306 tables). Single temperature physics makes use of the 301 tables, while two-temperature physics requires the 303/304 tables. As an alternative to the 303 table, the user may individually call both the 305 and 306 tables, provided they are available. Table 17 list the table types that are available to `LANL SESAME`. `LANL SESAME` is designed to operate in either regime and uses a modified ”divided difference/rational interpolation” scheme developed by Gerry Kerley for Sesame_Mig (also available in ALEGRA as `KEOS SESAME`) to find the requested values.

**Table 17.** Table Types for `LANL SESAME`

| Table Number | Description |
|---|---|
| 301 | Total pressure and energy tables. (301 = 304 + 305 + 306 ) |
| 303 | Ionic EOS pressure and energy tables (including cold curve and zero point contributions). (303 = 305 + 306) |
| 304 | Thermal electron pressure and energy tables. |
| 305 | Thermal ion pressure and energy tables (including zero point contribution). |
| 306 | Cold curve. |
| 601 | Average ionization state ($\bar{Z}$) table. |
| 602 | Electrical conductivity table. |
| 603 | Thermal conductivity table. |

If no temperature is defined in the `MATERIAL` section of the input deck, `LANL SESAME` will define a default temperature by reverse interpolation from the reference density and 1 atmosphere pressure. The reference temperature is usually NOT 298 K.

The energy table is relative while the pressure table is absolute, therefore, the zero reference energy of the specific internal energy table can be adjusted at simulation startup. For example, if the user wants the minimum energy in the table to be zero, then `ENERGY`

ZERO is specified. A user defined shift would use ENERGY SHIFT = real. To be similar to the KEOS SESAME model (see Reference [12]), ENERGY CTH uses $e = e + (\partial e/\partial T) \cdot T$ where $\partial e/\partial T$ is calculated at the reference density and temperature point of the table and $T$ is the reference temperature. This is done to ensure that the reference energy is always positive regardless of how the table is modified.

There are several flags in ALEGRA to prevent users from falling into possible traps in LANL SESAME. The first does not allow the 301 tables to be called in 2T physics or in combination with either the 303 or 304 tables. The reason is that calling a total EOS table and combining it with a table it has already been combined would incorrectly represent the values in either table. An attempt to do this will abort the code. Another flag currently present requires both the 303 and 304 tables to be present. If they are not both requested, ALEGRA will automatically add the missing table. Finally, 2T tables are not available for all NMATs and ALEGRA will abort if the tables are not found.

Features that are not yet available in LANL SESAME include opacities, ionization/free electrons, melt curves, vaporization curves, and Young's modulus. These features will be added as time permits and on user's request (if possible).

- Modules: material_libs/lanl_sesame
  - lanl_sesame.h
  - lanl_sesame.C
  - sesame_*.h
  - sesame_*.C
- Physics:
  - all physics options (including those that specify TWO TEMPERATURE)

Table 18: Input Parameters for LANL SESAME.

| Parameter Name | Type | Description |
|---|---|---|
| NMAT | int | Sesame material call number (required) |
| TABLE | int | Sesame table call number (required) |
| FEOS | string | Use personal Sesame Ship file (optional) |

*continued on next page*

*continued from previous page*

| | | |
|---|---|---|
| CLIP | real | Clips the temperature by this $\Delta T$ value (in Kelvin) above the bottom and below the top of table. Energy will be set to either $e(\rho, T_{min} + T_{clip})$ or $e(\rho, T_{max} - T_{clip})$. |
| ICLIP<br>ECLIP | real<br>real | Clips the ion temperature.<br>Clips the electron temperature.<br>If CLIP $> 0$, and ICLIP and ECLIP not set, ICLIP = ECLIP = CLIP. |
| INITIAL PRESSURE | real | Initial temperature will be set based on $T(\rho, P_{init})$. This is useful for setting $P = 0$. (default = 1.0 ATM) |
| REFERENCE PRESSURE | real | The pressure that is used in the temperature initialization. (INITIAL PRESSURE overrides REFERENCE PRESSURE) (default = 1.0 ATM) |
| ENERGY SHIFT or<br>ENERGY ZERO or<br>ENERGY CTH | [real] | Sets the zero energy reference of the specific internal energy table. The real value only applies to the ENERGY SHIFT option. (The default is to not shift the table, *i.e.*, ENERGY SHIFT = 0.0) |
| PRIMARY INTERPOLATION | string | Sets the interpolation method to be used. If $\partial E/\partial T < 0$, secondary interpolation will be called to recalculate $\partial E/\partial T$. (default = RATIONAL) |
| SECONDARY INTERPOLATION | string | Sets the interpolation method to be used. (default = LINEAR) |
| MINIMUM TEMPERATURE<br>MAXIMUM TEMPERATURE | real<br>real | Sets a temperature floor.<br>Sets a temperature ceiling.<br>Values are not relative to the table edge. If CLIP is set and the floor is off the table, CLIP will override. |
| MINIMUM ION TEMPERATURE<br>MAXIMUM ION TEMPERATURE | real<br>real | Sets an ion temperature floor.<br>Sets an ion temperature ceiling.<br>Values are not relative to the table edge. If ICLIP is set and the floor is off the table, ICLIP will override. |

*continued on next page*

| *continued from previous page* | | |
|---|---|---|
| MINIMUM ELECTRON TEMPERATURE<br>MAXIMUM ELECTRON TEMPERATURE | real<br>real | Sets an electron temperature floor. Sets an electron temperature ceiling. Values are not relative to the table edge. If ECLIP is set and the floor is off the table, ECLIP will override. |
| R0<br>RF<br>T0<br>TF | real<br>real<br>real<br>real | Creates a phase space box, R0 $< \rho <$ RF and T0 $< T <$ TF, where the CVMULT and CVMAX parameters apply. (Defaults: T0 = REAL_MAX, TF = 0.0, R0 = REAL_MAX, RF = 0.0, so that this option is off.) |
| CVMULT | real | Specific heat multiplier (default = 1.0) |
| CVMAX | real | Specific heat maximum, $C_v \leq$ CVMAX (default = REAL_MAX) |
| TEMPERATURE SWITCH<br><br><br><br>ZBAR SWITCH | real<br><br><br><br>real | ALEGRA will run 1T until the material TEMPERATURE becomes greater than TEMPERATURE SWITCH, at which point it will switch to 2T. (default = 0.0)<br>ALEGRA will run 1T until the material ZBAR becomes greater than ZBAR SWITCH, at which point it will switch to 2T. The ZBAR used is calculated from the Saha model. (default = 0.0) |

Example of how to use LANL SESAME for 1T:

```
model 1 lanl sesame
   nmat  3715  $ aluminum
   table 301   $ total combined EOS table (303+304)
end
```

Example of how to use LANL SESAME for 2T:

```
model 1 lanl sesame
   nmat 3717   $ aluminum
   table 303   $ electron EOS
   table 304   $ ion EOS
```

**Table 19.** Registered Plot Variables for `LANL SESAME`

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | real | `IOPUT` | Material density, $\rho$ |
| `ENERGY` | real | `IOPUT` | Total specific energy, $e = e_i + e_e$ |
| `TEMPERATURE` | real | `IOPUT` | 1T material temperature, $T$ |
| `PRESSURE` | real | `OUTPUT` | Total pressure, $P = P_i + P_e$ |
| `SOUND_SPEED` | real | `OUTPUT` | $C_s = \left( \frac{\partial P}{\partial \rho} + \frac{T}{\rho} \left( \frac{\partial P}{\partial T} \right)^2 \left( \frac{\partial e}{\partial T} \right)^{-1} \right)^{1/2}$ |
| `SPECIFIC_HEAT_VOL` | real | `OUTPUT` | Specific heat, $C_v = \partial e / \partial T$ |
| `ION_ENERGY` | real | `IOPUT` | Ion specific energy, $e_i$ |
| `ION_TEMPERATURE` | real | `IOPUT` | Ion temperature, $T_i$ |
| `ION_PRESSURE` | real | `IOPUT` | Ion pressure, $P_i$ |
| `ION_SPECIFIC_HEAT_VOL` | real | `OUTPUT` | Ion specific heat, $C_{vi} = \partial e_i / \partial T_i$ |
| `ELECTRON_ENERGY` | real | `IOPUT` | Electron specific energy, $e_e$ |
| `ELECTRON_TEMPERATURE` | real | `IOPUT` | Electron temperature, $T_e$ |
| `ELECTRON_PRESSURE` | real | `OUTPUT` | Electron pressure, $P_e$ |
| `ELECTRON_SPECIFIC_HEAT_VOL` | real | `OUTPUT` | Electron specific heat, $C_{ve} = \partial e_e / \partial T_e$ |

```
   datafile 'myshipfile.dat'  $ modified file
end
```

### 12.3.3 LLNL QEOS

```
MODEL model_id_number LLNL QEOS
   [parameter = value]
   ...
END
```

`LLNL QEOS` [35] is a self contained theoretical model that requires no external data base. The inputs are the material composition (density) at room temperature (298.15 K). Outputs include energy, pressure, entropy, Helmholtz free energy, and their derivatives with respect to temperature and density.

It should be noted that composite materials are mixed on an atom fraction basis into a homogeneous element. For example, if carbon and hydrogen are mixed as polyethylene (CH), `LLNL QEOS` will see the equivalent element with charge of 3.5 and mass of 6.5. This is not correct, but the pressure balance element mixing functions are not complete yet.

Also note that the bulk modulus should be entered by the user if known. The bulk modulus corrects the calculated sound speed which affects the pressure.

- Modules: material_libs/llnl_eos
  - llnl_qeos.h
  - llnl_qeos.C
  - qeos*.[hC]
- Physics:
  - all physics options that specify `TWO TEMPERATURE`

Example of how to use QEOS for 1T:

```
model 1 llnl qeos
 $ density multiplier    = 1.0
 $ pressure multiplier   = 1.0
 $ reference temperature = 298.15  $ Kelvin
   reference density     = 2.7e3   $ kg/m^3
end
```

**Table 20.** Input Parameters for `LLNL QEOS`

| Parameter Name | Type | Description |
|---|---|---|
| `ENERGY MULTIPLIER` | real | Scales energy (default = 1.0) |
| `PRESSURE MULTIPLIER` | real | Scales pressure (default = 1.0) |
| `DENSITY MULTIPLIER` | real | Scales density (default = 1.0) |
| `REFERENCE DENSITY` | real | Usually density at room temperature |
| `REFERENCE TEMPERATURE` | real | Usually 298.15 K (default = 298.15 K) |
| `MINIMUM DENSITY` | real | Density floor (default = 0.0) |
| `MINIMUM PRESSURE` | real | Pressure floor (NO Default) |
| `MINIMUM TEMPERATURE` | real | Sets the minimum temperature (ion & electron) |
| `BULK MODULUS` | real | Sets the bulk modulus (Pa). User should use this whenever possible. |
| `TOLERANCE` | real | Temperature solve (default = 1.0e-6) |
| `ONE TEMPERATURE` | none | One temperature flag (default = 1T) |
| `TWO TEMPERATURE` | none | Two temperature flag (default = 1T) |

Example of how to use `QEOS` for 2T:

```
model 1 llnl qeos
   two temperature
   reference density 1.044e3 $ kg/m^3
end
```

**Table 21.** Registered Plot Variables for `LLNL QEOS`

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | real | `IOPUT` | Material density, $\rho$ |
| `ENERGY` | real | `IOPUT` | Total specific energy, $e = e_i + e_e$ |
| `TEMPERATURE` | real | `IOPUT` | 1T material temperature, $T$ |
| `PRESSURE` | real | `OUTPUT` | Total pressure, $P = P_i + P_e$ |
| `SOUND_SPEED` | real | `OUTPUT` | $C_s = \left(\frac{\partial P}{\partial \rho} + \frac{T}{\rho}\left(\frac{\partial P}{\partial T}\right)^2\left(\frac{\partial e}{\partial T}\right)^{-1}\right)^{1/2}$ |
| `SPECIFIC_HEAT_VOL` | real | `OUTPUT` | Specific heat, $C_v = \partial e/\partial T$ |
| `ION_ENERGY` | real | `IOPUT` | Ion specific energy, $e_i$ |
| `ION_TEMPERATURE` | real | `IOPUT` | Ion temperature, $T_i$ |
| `ION_PRESSURE` | real | `IOPUT` | Ion pressure, $P_i$ |
| `ION_SPECIFIC_HEAT_VOL` | real | `OUTPUT` | Ion specific heat, $C_{vi} = \partial e_i/\partial T_i$ |
| `ELECTRON_ENERGY` | real | `IOPUT` | Electron specific energy, $e_e$ |
| `ELECTRON_TEMPERATURE` | real | `IOPUT` | Electron temperature, $T_e$ |
| `ELECTRON_PRESSURE` | real | `OUTPUT` | Electron pressure, $P_e$ |
| `ELECTRON_SPECIFIC_HEAT_VOL` | real | `OUTPUT` | Electron specific heat, $C_{ve} = \partial e_e/\partial T_e$ |

## 12.4 Combined Electrical and Thermal Conductivity Models

Electrical conductivity models are used by TRANSIENT MAGNETICS to close the coupled set of hydrodynamics and Maxwell's equations by using Ohm's law which relates the current density $\vec{J}$ to the electric field $\vec{E}$ and the cross product of the velocity $\vec{v}$ and magnetic field $\vec{B}$,

$$\vec{J} = \sigma \left( \vec{E} + \kappa_3 \vec{v} \times \vec{B} \right). \tag{75}$$

where $\sigma$ is the electrical conductivity. This is the simplest form of Ohm's law.

THERMAL CONDUCTION physics solves the thermal heat conduction equation. The heat flux $\vec{q}$ is related to the temperature gradient by

$$\vec{q} = -k \nabla T, \tag{76}$$

where the constant of proportionality $k$ is the thermal conductivity. The heat flux $\vec{q}$ has dimensions of energy/area/time, the temperature has dimensions of Kelvin (K), so the thermal conductivity $k$ has dimensions energy/length/time/K.

Some models provide both electrical and thermal conductivities. Others provide only one or the other. The user must provide a set of models which will uniquely provide the required parameters for the physics which is requested. NOTE: If overlapping models are requested in the sense that two models provide the same output, then the code will use the last model listed to provide that given output.

### 12.4.1 LMD Conductivities

```
MODEL model_id_number LMD
   [parameter = value]
   ...
END
```

The Lee-More-Desjarlais (LMD) model [13, 14] is an extension of the Lee-More model [26] that provides much better agreement with measured electrical conductivities in the vicinity of the metal-insulator transition and yet smoothly blends with the Lee-More model away from this regime. The conductivities generated by the two models can differ by several orders of magnitude with the largest differences occurring for densities one to two orders of magnitude below solid density and temperatures below 2 eV. The model also outputs the thermal conductivities, as well as the ionization state, so that no separate thermal conductivity need be computed. This conductivity model also includes the effect of magnetic fields on the conductivity. The conductivities returned are those parallel and perpendicular to the magnetic field. The proper coordinate transformation between this magnetically oriented system and the computational coordinates needs to be ensured. The most significant changes in this model, relative to the Lee-More model, are an improved ionization model and a new expression for the minimum allowed collision time. The new ionization model blends the Thomas-Fermi ionization with a first-ionization Saha model. The Saha contribution is non-ideal in that a pressure ionization term is used to reduce the ionization energy near solid densities. The ionization state calculated with this model is a significant improvement over the Thomas-Fermi ionization in the vicinity of the metal-insulator transition, particularly for temperatures below 3 eV or so. The new expression for the minimum allowed collision time introduces a power law dependence on density through the `P2` parameter. This dependence is in agreement with the measured conductivities at low temperatures and densities below solid and effectively bridges the metal, liquid, and plasma conductivity regimes. It is anticipated that for most applications the user will make use of predefined parameter sets for each material of interest, linked to the `Z` for the material. These parameters may be overridden if the user wishes to experiment with modeling unsupported materials. The model is sufficiently robust that if the user supplies `Z`, `A`, `RHO SOLID`, `TMELT`, and `XIEV`, and accepts the defaults for the other parameters, reasonably good metallic conductivities will be generated.

In place of the `P2`, `P3`, and `P4` parameters of the generic Lee-More model, we now have

$$p_2 = \min\left[p_{2a} + \frac{p_{2b}}{p_T}, \frac{\bar{V_e}}{\sqrt{2E_F/m_e}}\left(\frac{n_a}{2\cdot 10^{22}}\right)^{2/p_T}\right], \tag{77}$$

$$p_3 = \frac{p_{3a}}{2}\left(1+\left(\frac{T_m}{T}\right)^{p_{3b}\cdot r_{fact}}\right), \text{ and} \tag{78}$$

$$p_4 = \frac{p_{4a}}{2}\left(1+\left(\frac{T_m}{T}\right)^{p_{3b}\cdot r_{fact}}\right) \tag{79}$$

where

$$p_T = 1+\exp\left(\frac{T-25000}{10000}\right), \text{ and} \tag{80}$$

$$r_{fact} = \sqrt{2/\left(\left(\frac{\rho}{\rho_s}\right)^2 + \left(\frac{\rho_s}{\rho}\right)^2\right)}. \tag{81}$$

$T_m$ is the melt temperature and $\rho_s$ is the material solid density. These constructions for P3 and P4 are used primarily for fine tuning the solid density conductivity as a function of temperature, including the drop in conductivity at melt which is determined by the relation:

$$\tau_{liq} = \frac{\tau_{BG}}{p_4}\frac{T}{T+p_5} \tag{82}$$

for $T \geq T_m$. The net effect of these changes to P3 and P4 are relatively minor. Note the new parameter P5 which is useful for matching the conductivity above melt for some materials, notably aluminum.

The correct parameters for some materials have been incorporated into the model. For these materials, one needs only to specify the atomic number, Z, or in the case of deuterium, also specify the atomic mass, A as 2. The code will set the remaining parameters correctly. Currently featured materials include those found in Table 22.

**Table 22.** Predefined Materials for LMD.

| Z | A | Symbol | Material |
|---|---|---|---|
| 1 | 2. | D | Deuterium |
| 13 | - | Al | Aluminum |
| 22 | - | Ti | Titanium |
| 29 | - | Cu | Copper |
| 47 | - | Ag | Silver |
| 73 | - | Ta | Tantalum |
| 74 | - | W | Tungsten |
| 79 | - | Au | Gold |
| - | - | - | Air |
| - | - | - | LXN (i.e., lexan) |

Certain materials have been researched through quantum molecular dynamics (QMD) simulations and the LMD model has been specifically ”tuned” for these materials. The tuned materials are listed in Table 23 and are accessed by using the ”TUNED” keyword listed in the table. Setting the Z parameter is optional.

**Table 23.** Tuned Materials for `LMD`.

| Z | A | Symbol | Keyword |
|---|---|---|---|
| 13 | - | Al | `TUNED ALUMINUM` |
| 74 | - | W | `TUNED TUNGSTEN` |

Note: there is no option for reverting to the original Lee More model for comparison purposes.

- Modules: material_libs/ec
  - ec_lmd.h
  - ec_lmd.C
  - lmd_setup.h
  - lmd.F
  - coulomb_logarithm.C
- Physics:
  - transient magnetics
  - all magnetohydrodynamics physics options

Table 24: Input Parameters for `LMD`.

| Parameter Name | Type | Description |
|---|---|---|
| `Z` | real | Atomic number (this is the only required parameter) |
| `A` | real | Atomic mass |
| `RHO SOLID` | real | Material solid density |
| `TMELT` | real | Specifies the solid density melt temperature. This scales the usual Cowan melt temperature calculation so that the melt temperature will be correct at solid density, but is still a function of density. Calculated if `TMELT` .le. 0. |
| `XIEV` | real | The energy of the first ionization in eV |
| | | *continued on next page* |

*continued from previous page*

| | | |
|---|---|---|
| `LOG LAMBDA MIN` | real | The minimum allowed value of the Coulomb logarithm |
| `G0, G1` | real | The spin level degeneracy factors $(2J+1)$ for the ground state and first ionization, respectively |
| `P1` | real | Multiplier on the average distance between ions $R_0 = (4\pi\rho/3Am_i)^{-1/3}$, used when comparing to the Debye-Huckel screening length and choosing the larger value to be the maximum impact parameter for the Coulomb logarithm |
| `P2A, P2B` | real | Used to construct the multiplier `P2` on the minimum electron relaxation time which is equal to the average distance between ion divided by the mean electron thermal velocity, $(3kT/m_e)^{1/2}$ |
| `P2C, P2D, P2E` | real | Default: `P2C = 25000.0`, `P2D = 2.0E22`, `P2E = 2.0` |
| `P3A, P3B` | real | Used to construct the multiplier `P3` on the electron relaxation time in the Bloch-Gruneisen regime (below the melt temperature). |
| `P4A, P4B` | real | Used to construct the multiplier `P4` on the electron relaxation time in the Bloch-Gruneisen regime (above the melt temperature) |
| `P5` | real | Used to fine tune the solid density conductivity above melt |
| `EC FLOOR` | real | Minimum electrical conductivity value (default = 0.) |
| `EC MULTIPLIER` | real | Electrical conductivity multiplier (default = 1.) |
| `TC FLOOR` | real | Minimum thermal conductivity value (default = 0.) |
| `TC MULTIPLIER` | real | Thermal conductivity multiplier (default = 1.) |

*continued on next page*

*continued from previous page*

| | | |
|---|---|---|
| `ANOMALOUS` | real | If greater than 0., will enable an anomalous collisionality in low density, high temperature regions. A value of 1.0 is nominal. See the `SPITZER` model for details. (default = 0.) |
| `TEMPERATURE CUTOFF` | real | The minimum temperature used to evaluate the conductivities (default = 0.) |
| `USE PARALLEL CONDUCTIVITY` | real | The `LMD` model returns the values of the conductivities perpendicular to the magnetic field by default. If not equal to 0., the `LMD` model will return the values of the conductivities parallel to the magnetic field. (default = 0.) |
| `PRESSURE IONIZATION PREFACTOR` | real | (default = 1.5) |
| `PRESSURE IONIZATION EXPONENT` | real | (default = 1.5) |
| `DIPOLE ALPHA` | real | (default = 50.0) |
| `EXTERNAL ZBAR` | none | Allows `LMD` to use a zbar other than the internally calculated one. NOTE: A zbar calculating model MUST be called before `LMD`. (default = off) |
| `TUNED ALUMINUM` | real | Allows the QMD aluminum specific coding to be called. This must be used in addition to `Z` = 13. (default = off) |
| `MATERIAL` | string | This can be used to call default mixed material settings. NOTE: This option will override the `Z` parameter, but none of the other parameters. |

Examples:

```
model 1 lmd
   z = 13.
   tuned aluminum
end
```

```
model 1 lmd
   z = 74.
end
```

**Table 25.** Registered Plot Variables for LMD.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| DENSITY | real | INPUT | Material density |
| TEMPERATURE | real | INPUT | Material temperature |
| ECON | real | OUTPUT | Electrical conductivity used |
| ECON_PAR | real | OUTPUT | Electrical conductivity parallel to the magnetic field |
| ECON_PERP | real | OUTPUT | Electrical conductivity transverse to the magnetic field |
| THERMAL_CON | real | OUTPUT | Thermal conductivity used |
| THERMAL_CON_PAR | real | OUTPUT | Thermal conductivity parallel to the magnetic field |
| THERMAL_CON_PERP | real | OUTPUT | Thermal conductivity transverse to the magnetic field |
| ZBAR | real | OUTPUT | Average ionization state |

```
model 1 lmd
   material = ’air’
end
```

### 12.4.2 Spitzer Conductivities

```
MODEL model_id_number SPITZER
   [parameter = value]
   ...
END
```

The SPITZER conductivity model [41] is valid for high temperature plasmas above 1 eV temperature. The model computes electrical conductivity components parallel and perpendicular to the magnetic field.

$$\sigma_{||}(\rho,T) = (\gamma_E)\frac{2(2kT)^{3/2}}{\pi^{3/2} m_e^{1/2} Z e^2 c^2 \ln\Lambda} \cdot \frac{1}{f_{anom}} \tag{83}$$

$$\sigma_{\perp}(\rho,T) = \left(\frac{3\pi}{32}\right)\frac{2(2kT)^{3/2}}{\pi^{3/2} m_e^{1/2} Z e^2 c^2 \ln\Lambda} \cdot \frac{1}{f_{anom}} \tag{84}$$

By default, the model returns the values of the perpendicular electric conductivity.

The model also computes thermal conductivity components parallel and perpendicular to the magnetic field.

$$\kappa_{||}(\rho,T) = (\delta_T)20\left(\frac{2}{\pi}\right)^{3/2}\frac{(kT)^{5/2}k}{m_e^{1/2} e^4 Z \ln\Lambda} \cdot \frac{1}{f_{anom}} \tag{85}$$

$$\begin{aligned}\kappa_{\perp}(\rho,T) &= \frac{8}{3}\frac{(\pi m_i k)^{1/2} n_i^2 Z^2 e^2 c^2 \ln\Lambda}{T^{1/2}B^2} \cdot f_{anom} \\ &= \frac{8}{3}(\pi m_{amu} k)^{1/2} N_A^2 e^2 c^2 \left(\frac{\rho^2 Z^2 \ln\Lambda}{A^{3/2} T^{1/2} B^2}\right) \cdot f_{anom}\end{aligned} \tag{86}$$

By default, the model returns the minimum of these two thermal conductivities.

In low-density, high-temperature plasmas in a strong magnetic field, the electrical conductivity may be reduced by an anomalous factor, $f_{anom} \geq 1$ [40]. This factor also enhances the value of the perpendicular thermal conductivity, and reduces the value of the parallel thermal conductivity as shown in the above formulas. These anomalous conductivities are enabled by setting the input parameter ANOMALOUS $> 0.$, where a value of 1.0 is nominal. A value greater than 1 increases this effect and a value less than 1 reduces this effect. In CGS units with the electron temperature expressed in eV, then the anomalous factor is:

$$\begin{aligned} f_{anom} &= 1 + P_{anomalous}\frac{\nu^*}{\nu_{ei}} \\ \nu^* &= \Omega_e\sqrt{(\bar{Z}m_e)/(Am_{amu})}\,(V_d/V_{th})^2 \\ \nu_{ei} &= 2.9\cdot 10^{-6}\,(\bar{Z}^2 N_i \log\Lambda)/(T_e^{3/2}) \\ V_d &= |\vec{J}|/(\bar{Z}eN_i) \\ V_{th} &= \sqrt{(2k_B T_i)/(Am_{amu})} \end{aligned} \tag{87}$$

where $P_{anomalous}$ is the ANOMALOUS input parameter, $\nu^*$ is an anomalous collision frequency, $\nu_{ei}$ is the electron-ion collision frequency, $\Omega_e = (e|\vec{B}|)/(m_e c)$ is the electron cyclotron frequency, $V_d$ is a drift velocity, $V_{th}$ is the ion thermal velocity, $\bar{Z}$ is the average ionization state, $A$ is the atomic mass, and $N_i$ is the ion number density.

The parameters COLD ECON and COLD TCON are a simplified attempt to extend the SPITZER model below the Spitzer regime. A better approach is to use the LMD model. If these parameters are non-zero and if the ionization state $\bar{Z}$ is less than 1, this model model will use $\bar{Z}$ to linearly interpolate between the cold value and the Spitzer value with $\bar{Z} = 1$.

$$\sigma = \begin{cases} (1-\bar{Z})\sigma_{cold} + \bar{Z}\sigma_{Spitzer}, & \bar{Z} < 1 \\ \sigma_{Spitzer}, & \bar{Z} \geq 1 \end{cases} \tag{88}$$

$$\kappa = \begin{cases} (1-\bar{Z})\kappa_{cold} + \bar{Z}\kappa_{Spitzer}, & \bar{Z} < 1 \\ \kappa_{Spitzer}, & \bar{Z} \geq 1 \end{cases} \tag{89}$$

- Modules: material_libs/ec
  - ec_spitzer.h
  - ec_spitzer.C
  - coulomb_logarithm.C
- Physics:

- transient magnetics
- all magnetohydrodynamics physics options

Table 26: Input Parameters for SPITZER.

| Parameter Name | Type | Description |
|---|---|---|
| A | real | Atomic mass |
| Z | real | Atomic number |
| FIXED ZBAR | real | Overrides any ionization model with a constant value for the ionization state (optional) |
| FIXED COULOG | real | Overrides the coulomb logarithm calculation with a constant value (optional) |
| EC MULTIPLIER | real | May be used to scale the electrical conductivity (optional) |
| TC MULTIPLIER | real | May be used to scale the thermal conductivity (optional) |
| COLD ECON | real | Electrical conductivity value when ZBAR$<$1.0 (optional) |
| COLD TCON | real | Thermal conductivity value when ZBAR$<$1.0 (optional) |
| ANOMALOUS | real | If greater than 0., will enable an anomalous collisionality in low density, high temperature regions. A value of 1.0 is nominal. (default = 0.) |
| TEMPERATURE CUTOFF | real | Model is called with the maximum of the material temperature or the cutoff temperature. (optional) |
| USE PARALLEL CONDUCTIVITY | real | Setting this parameter to be non-zero causes the parallel value of the conductivity will be used. (optional) |
| USE PERPENDICULAR CONDUCTIVITY | real | Setting this parameter to be non-zero causes the perpendicular value of the conductivity will be used. (optional) |

**Table 27.** Registered Plot Variables for SPITZER.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| DENSITY | real | INPUT | Material density |
| TEMPERATURE | real | INPUT | Material temperature |
| ECON<br>ECON_PAR<br>ECON_PERP | real | OUTPUT | Electrical conductivity |
| THERMAL_CON<br>THERMAL_CON_PAR<br>THERMAL_CON_PERP | real | OUTPUT | Thermal conductivity |
| COULOG | real | OUTPUT | Coulomb logarithm |

## 12.5 Electrical Conductivity Models

### 12.5.1 EC Anomalous Conductivity

```
MODEL model_id_number EC ANOMALOUS
   [parameter = value]
   ...
END
```

This is a density and temperature independent model that returns one of two user specified values for the conductivity depending upon the value of the cell-centered current density, $J_E$. This model is given by

$$\sigma(\rho, T) = \begin{cases} \sigma_0, & |\vec{J}_E| \leq J_{anom} \\ \sigma_1, & |\vec{J}_E| > J_{anom} \end{cases} \tag{90}$$

The model is parametrized by the constants $\sigma_0$, $\sigma_1$, and the magnitude of the current density $J_{anom}$. The orientation of the current density is inconsequential.

- Modules: material_libs/ec
  - ec_anomalous.h
  - ec_anomalous.C
- Physics:
  - transient magnetics
  - all magnetohydrodynamics physics options

**Table 28.** Input Parameters for `EC ANOMALOUS`.

| Parameter Name | Type | Description |
|---|---|---|
| `SIGMA0` | real | Reference conductivity, $\sigma_0$ |
| `SIGMA1` | real | Reference conductivity, $\sigma_1$ |
| `JANOM` | real | Threshold current density , $J_{anom}$ |

**Table 29.** Registered Plot Variables for `EC ANOMALOUS`.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | real | `INPUT` | Material density |
| `TEMPERATURE` | real | `INPUT` | Material temperature |
| `ECON` | real | `OUTPUT` | Electrical conductivity |

### 12.5.2 EC Knoepfel Conductivity

```
MODEL model_id_number EC KNOEPFEL
   [parameter = value]
   ...
END
```

A low temperature model called the Knoepfel model [25] is reasonable for metals from about 100 K up to the melting point. This model is given by

$$\sigma(\rho, T) = \max\left(\frac{\sigma_0}{1+\beta C_v(T-T_0)}\left(\frac{\rho}{\rho_0}\right)^{\alpha}, \sigma_{min}\right) \tag{91}$$

The model is parametrized by the constants $\sigma_0$, $\rho_0$, $\alpha$, and $\beta C_v$. A constant conductivity may be obtained by setting $\alpha = 0$ and $\beta C_v = 0$. A reasonable guess for $\alpha$ given no other information is twice the Gruneisen coefficient for the material. At a minimum this model can achieve a drop in conductivity with temperature. It cannot deal with the melt transition however. The conductivity floor is not part of the model given by Knoepfel, but is added for additional flexibility.

- Modules: material_libs/ec
  - ec_knoepfel.h
  - ec_knoepfel.C
- Physics:
  - transient magnetics
  - all magnetohydrodynamics physics options

**Table 30.** Input Parameters for `EC KNOEPFEL`.

| Parameter Name | Type | Description |
|---|---|---|
| `SIGMA0` | real | Reference conductivity, $\sigma_0$ |
| `RHO0` | real | Model reference density, $\rho_0$ |
| `T0` | real | Model reference temperature, $T_0$ |
| `ALPHA` | real | Density exponent, $\alpha$ |
| `BETACV` | real | Temperature coefficient, $\beta C_v$ |
| `SIGMAMIN` | real | Minimum conductivity, $\sigma_{min}$ |
| `TEMPERATURE CUTOFF` | real | Model is called with the maximum of the material temperature or the cutoff temperature |

**Table 31.** Registered Plot Variables for `EC KNOEPFEL`.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | real | `INPUT` | Material density |
| `TEMPERATURE` | real | `INPUT` | Material temperature |
| `ECON` | real | `OUTPUT` | Electrical conductivity |

### 12.5.3 KEC Sesame Conductivity

```
MODEL model_id_number KEC SESAME
   [parameter = value]
   ...
END
```

Electrical conductivity as found in tabular form in Kerley Sesame format [19]. Internally this is the 602 Sesame table. This is not really a model. It is a software interface. The origins of any table used here must be carefully researched and understood.

- Modules: material_libs/kerley_eos/sesame
  - sec_mig.h
  - sec_mig.C
  - secmig.F
- Physics:

- transient magnetics
- all magnetohydrodynamics physics options

**Table 32.** Input Parameters for `KEC SESAME`.

| Parameter Name | Type | Description |
|---|---|---|
| `FEC` | string | Table file name |
| `NEC` | integer | Table Number |
| `TEMPERATURE CUTOFF` | real | Model is called with the maximum of the material temperature or the cutoff temperature. |

**Table 33.** Registered Plot Variables for `KEC SESAME`.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | real | `INPUT` | Material density |
| `TEMPERATURE` | real | `INPUT` | Material temperature |
| `ECON` | real | `OUTPUT` | Electrical conductivity |

Example:

```
model, 2, kec sesame
   fec = 'ses600'
   nec = 3109
end
```

## 12.6 Thermal Conductivity Models

### 12.6.1 Constant Thermal Conductivity

```
MODEL model_id_number CONSTANT THERMAL CONDUCTIVITY
   [parameter = value]
   ...
END
```

The `CONSTANT THERMAL CONDUCTIVITY` model allows the user to specify a simple constant conductivity.

$$\kappa(\rho, T) = \kappa_0 \tag{92}$$

- Modules: material_libs/thermal_con_models
  - constant_con.h
  - constant_con.C
- Physics: thermal conduction and its derivatives

**Table 34.** Input Parameters for `CONSTANT THERMAL CONDUCTIVITY`.

| Parameter Name | Type | Description |
|---|---|---|
| `THERMAL CONSTANT` | real | Constant thermal conductivity, $\kappa_0$ |

**Table 35.** Registered Plot Variables of `CONSTANT THERMAL CONDUCTIVITY`.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `TEMPERATURE` | real | `INPUT` | Material temperature |
| `DENSITY` | real | `INPUT` | Material density |
| `THERMAL_CON` | real | `OUTPUT` | Thermal conductivity |

### 12.6.2 Plasma Thermal Conductivity

```
MODEL model_id_number PLASMA
   [parameter = value]
   ...
END
```

The PLASMA thermal conductivity model is a simple power law of the form:

$$\kappa(\rho, T) = A\rho^B T^C. \tag{93}$$

Similar to the `SPITZER` conductivity model if $B = 0$ and $C = 5/2$, it can be used to study deviations from pure Spitzer-like behavior. It can also be used to specify a constant conductivity.

- Modules: material_libs/thermal_con_models
  - plasma.h
  - plasma.C
- Physics: thermal conduction and its derivatives

**Table 36.** Input Parameters for `PLASMA`.

| Parameter Name | Type | Description |
|---|---|---|
| `A` | real | Thermal conductivity coefficient (note that conductivities obtained from tables may need to be scaled if `B` or `C` are nonzero) |
| `B` | real | Density exponent |
| `C` | real | Temperature exponent |

### 12.6.3 Polynomial Thermal Conductivity

```
MODEL model_id_number POLYNOMIAL
   [parameter = value]
   ...
END
```

**Table 37.** Registered Plot Variables of `PLASMA`.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | real | `INPUT` | Material density |
| `TEMPERATURE` | real | `INPUT` | Material temperature |
| `THERMAL_CON` | real | `OUTPUT` | Thermal conductivity |

The POLYNOMIAL thermal conductivity model is a simple power law model that depends only on temperature:

$$\kappa(\rho, T) = A_0 + A_1 \cdot T^{N_1} + A_2 \cdot T^{N_2} + A_3 \cdot T^{N_3}. \tag{94}$$

- Modules: material_libs/thermal_con_models
  - polynomial_con.h
  - polynomial_con.C
- Physics: thermal conduction and its derivatives

**Table 38.** Input Parameters for `POLYNOMIAL`.

| Parameter Name | Type | Description |
|---|---|---|
| `A0` | real | Constant coefficient |
| `A1` | real | Coefficient of first temperature term |
| `A2` | real | Coefficient of second temperature term |
| `A3` | real | Coefficient of third temperature term |
| `N1` | real | Power of first temperature term |
| `N2` | real | Power of second temperature term |
| `N3` | real | Power of third temperature term |

### 12.6.4 KTC Sesame Thermal Conductivity

```
MODEL model_id_number KTC SESAME
   [parameter = value]
   ...
END
```

**Table 39.** Registered Plot Variables of `POLYNOMIAL`.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | real | `INPUT` | Material density |
| `TEMPERATURE` | real | `INPUT` | Material temperature |
| `THERMAL_CON` | real | `OUTPUT` | Thermal conductivity |

Thermal conductivity as found in tabular form in Kerley Sesame format [19]. Internally this is the 603 Sesame table. This is not really a model. It is a software interface. The origins of any table used here must be carefully researched and understood.

- Modules: material_libs/kerley_eos/sesame
  - stc_mig.h
  - stc_mig.C
  - stcmig.F
- Physics: thermal conduction and its derivatives

**Table 40.** Input Parameters for `KTC SESAME`.

| Parameter Name | Type | Description |
|---|---|---|
| `FTC` | String | Table file name |
| `NTC` | int | Table Number |
| `TEMPERATURE CUTOFF` | real | Minimum temperature |

**Table 41.** Registered Plot Variables of `KTC SESAME`.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | real | `INPUT` | Material density |
| `TEMPERATURE` | real | `INPUT` | Material temperature |
| `THERMAL_CON` | real | `OUTPUT` | Electrical conductivity |

Example:

```
model, 4, ktc sesame
   ftc = ’ses600’
   ntc = 3109
end
```

## 12.7 Opacity Models

Opacity models calculate radiation absorption and emission coefficients for radiation and radiation hydrodynamics simulations. Currently recognized opacity model names are listed in Table 13 on page 112.

If possible and if needed or requested, ALEGRA's opacity models compute both the average Planck absorption opacity $\tau_{ag}$ and average Rosseland opacity $\tau_{rg}$ in units of (1/length). These opacities are defined according to

$$\tau_{ag} = \frac{\rho \int_{E_g}^{E_{g+1}} \sigma_a(E) B(T_m, E) dE}{\int_{E_g}^{E_{g+1}} B(T_m, E) dE} \tag{95}$$

and

$$\frac{1}{\tau_{rg}} = \frac{\int_{E_g}^{E_{g+1}} \frac{1}{\sigma_t(E)} \frac{\partial B(T_m,E)}{\partial T} dE}{\rho \int_{E_g}^{E_{g+1}} \frac{\partial B(T_m,E)}{\partial T} dE} \tag{96}$$

where $B(T_m, E)$ is the blackbody function, $\sigma_a(E)$ is the absorption cross section, $\sigma_t(E) = \sigma_a(E) + \sigma_s(E)$ is the total cross section (both in units of (length$^2$/mass)), and integrations are computed over the group boundaries. The Planck average involves only the absorption cross section is used for the radiation emission and absorption rates of materials. The Rosseland average involves the total cross section and is used for the diffusion coefficient of radiation transport methods.

### 12.7.1 Kramers Fitted Opacity

```
MODEL model_id_number KRAMERS FITTED
   [parameter = value]
   ...
END
```

The Kramers fitted opacity [49] uses the simple analytic expressions

$$\tau_a = \rho \sigma_a = C_a \rho^{a+1} T^b \nu^c \tag{97}$$

$$\tau_s = \rho \sigma_s = C_s \rho^{d+1} T^e \nu^f \tag{98}$$

to calculate the true absorption $\sigma_a$ and scattering $\sigma_s$ mass coefficients (units are area per mass). For multi-group cases, the frequency (actually energy) is set to the lower group bound.

For Kramers fitted opacity, the Rosseland opacity is set equal to the sum of the absorption and scattering opacities.

$$\tau_r = \tau_a + \tau_s \tag{99}$$

- Modules: material_libs/opac_models
  - kramers_fitted.h
  - kramers_fitted.C
- Physics:
  - emission
  - radiation and all derived physics classes

**Table 42.** Input Parameters for `KRAMERS FITTED`.

| Parameter Name | Type | Description |
|---|---|---|
| `KAP COEF` | real | $C_a$ - Coefficient of true absorption |
| `KAP RHO EXP` | real | $a$ - Density exponent for absorption |
| `KAP T EXP` | real | $b$ - Temperature exponent for absorption |
| `KAP NU EXP` | real | $c$ - Photon energy exponent for absorption |
| `SIG COEF` | real | $C_s$ - Coefficient of scattering |
| `SIG RHO EXP` | real | $d$ - Density exponent for scattering |
| `SIG T EXP` | real | $e$ - Temperature exponent for scattering |
| `SIG NU EXP` | real | $f$ - Photon energy exponent for scattering |

For example, free-free absorption and Thompson scattering in a completely ionized plasma could be specified as:

```
model 1 kramers fitted
   kap coef    =  2.22e32
   kap rho exp =  1.0
   kap t exp   = -0.5
   kap nu exp  = -3.0
```

**Table 43.** Registered Plot Variables for KRAMERS FITTED.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| DENSITY | Scalar | INPUT | Density |
| TEMPERATURE | Scalar | INPUT | Temperature |
| NU(j) | Scalar | INPUT | Left frequency bound in j-th group |
| OLD_DENSITY | Scalar | OUTPUT | Previous density (not used) |
| OLD_TEMP | Scalar | OUTPUT | Previous temperature (not used) |
| OPACITY_A | Opacity | OUTPUT | $\tau_a$ - Absorption opacity - units of $\frac{1}{length}$ |
| OPACITY_R | Opacity | OUTPUT | $\tau_r$ - Rosseland opacity - units of $\frac{1}{length}$ |
| OPACITY_S | Opacity | OUTPUT | $\tau_s$ - Scattering opacity - units of $\frac{1}{length}$ |

```
   sig coef    =  3.99
   sig rho exp =  0.0
   sig t exp   =  0.0
   sig nu exp  =  0.0
end
```

This model is particularly useful for calculations where the opacity in the regime of interest can be approximated reasonably well by the above analytical form and computational speed is desired.

### 12.7.2 XSN Opacity

```
MODEL model_id_number XSN
   NUMBER OF ELEMENTS int
      ELEMENT int, MASS real, FRACTION real
   END
   [parameter = value]
   ...
END
```

XSN [30, 34, 37] provides frequency-dependent emission and absorption coefficients for a material that may or may not be in LTE (local thermodynamic equilibrium). The model assumes an average ion approximation and can handle the case of multiple elements in a composite material. The XSN package is highly compute intensive compared to other opacity models.

In LTE, the required inputs are the mass density ρ, the atomic molar fraction $f_a$, and the electron temperature $T_e$.

In non-LTE, additional inputs are needed (see below).

Material composition information listed under the `XSN` model will override the composition information listed under the `MATERIAL` model.

- Modules: material_libs/opac_models
  - xsn.h
  - xsn.C
  - xsn_interface.h
  - xsn_interface.C
- Physics:
  - emission
  - radiation and all derived physics classes

Table 44: Input Parameters for `XSN`.

| Parameter Name | Type | Description |
|---|---|---|
| Commonly Used XSN Options | | |
| `NUMBER OF ELEMENTS`<br>`  ELEMENT int, MASS real,`<br>`      FRACTION real`<br>`END` | int | Specification of the atomic composition (optional). `XSN` will use the composition specified in the `MATERIAL` keyword block if omitted here, otherwise the `XSN` will override the `MATERIAL` specification for this model if present here. (default = 0, maximum = 5) |
| `ATOMIC VOLUME MULTIPLIER` | real | Fudge factor for atomic volumes (default = 1.0) |
| `DYNAMIC INTEGRATION` | none | Changes the group bounds to $g_0 = 0.12299T_m$ and $g_f = 11.6088T_m$ and uses these bounds to calculate the opacity. *The user must set `NUMBER OF GROUPS` = 1* |

*continued on next page*

*continued from previous page*

| | | |
|---|---|---|
| `HAGEN RUBENS` | none | This is NOT an ice cream sandwich with corned beef. Modifies the low photon energy opacities using the value of the electrical conductivity and is recommended primarily for metals [16]. *Note! The user must specify the electrical conductivity model before the XSN model in the MATERIAL ... END block of the input deck!* |
| `DENSITY CHANGE` | real | Fractional density change necessary before recomputing opacities. If the fractional density change, $\Delta\rho/\rho_{old}$, is less than this value, then the old opacity value is used again. $\rho_{old}$ is the density at which the opacity was last computed. (default = 0.0) |
| `DENSITY FLOOR` | real | Minimum density used in a call to XSN (default = 0.0, code units) |
| `TEMPERATURE CHANGE` | real | Fractional temperature change necessary before recomputing opacities. If the fractional temperature change, $\Delta T/T_{old}$, is less than this value, then the old opacity value is used again. $T_{old}$ is the temperature at which the opacity was last computed. (default = 0.0) |
| `TEMPERATURE CUTOFF` | real | Minimum temperature used in a call to XSN (default = 0.0, code units) |
| `COLD OPACITY` | real | Value of the cold opacity to be used below the `COLD OPACITY THRESHOLD` temperature. This option allows the user to specify an opacity value at lower temperatures where `XSN` may not compute accurate values, and when the `HAGEN RUBENS` option is not usable. The same value is used for both the Planck and Rosseland opacities. *The user must set `NUMBER OF GROUPS` = 1* (default = 0.0, code units, i.e., $\frac{1}{length}$) |
| `COLD OPACITY THRESHOLD` | real | Temperature value below which the `COLD OPACITY` will be used. (default = 0.0, code units) |

*continued on next page*

*continued from previous page*

| Other XSN Options | | |
|---|---|---|
| `BOUND BOUND MULTIPLIER` | real | Fudge factor for bound-bound opacity (default = 1.0) |
| `CONTINUUM LOWERING MULTIPLIER` | real | Fudge factor for continuum lowering (default = 1.0) |
| `CONTINUUM LOWERING TYPE` | int | Continuum lowering model to use:<br>1 = use no continuum lowering (default)<br>2 = ion-sphere model |
| `DOPPLER WIDTH OPTION` | int | Use Doppler line width model (default = 0) |
| `ENERGY LEVEL TYPE` | int | Energy level model to use:<br>1 = Bohr with Pauli corrections (default)<br>2 = Dirac energy levels |
| `FREE BOUND MULTIPLIER` | real | Fudge factor for free-bound emission (default = 1.0) |
| `LINE OPTION` | int | (default = 0) |
| `LINE WIDTH MULTIPLIER` | real | Fudge factor for line widths (default = 1.0) |
| `MINIMUM NUCLEAR Z` | real | (default = 10.0) |
| `MAXIMUM DENSITY` | real | Cutoff density for bound-bound opacity fudge factor (default = 100.0) |
| `NUMBER OF LEVELS` | int | (default = 7, maximum = 10) |
| `NUMBER OF SUBGROUPS PER GROUP` | int | (default = 100, maximum = 560) |
| `OPACITY AVG LIMIT` | int | (default = 1) |
| `PLANCK OPACITY OPTION` | int | (default = 1) |
| `STRAIGHT OPACITY AVERAGE` | int | (default = 1) |
| `TWO BAND OPTION` | int | (default = 0) |
| `EXTRA ELECTRONS PER NUCLEUS` | real | (default = 0.0) |
| `EXTRA Z SQUARED PER NUCLEUS` | real | (default = 0.0) |
| `OPACITY MULTIPLIER` | real | Multiplies all groups by a single constant (default=1.0) |

`DYNAMIC INTEGRATION` uses the fact that 99% of the energy in the black body function is within the bounds of $0.12299\,T < T < 11.6088\,T$. When calculating the opacities in `XSN`, there are a maximum of only 100 integration points, and this is an attempt to ensure a greater probability of having those points land on transition lines. This is especially true for lower temperatures where the black body function is very narrow.

**Table 45.** Registered Plot Variables for XSN.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| DENSITY | Scalar | INPUT | Material density |
| TEMPERATURE | Scalar | INPUT | Material temperature |
| OLD_DENSITY | Scalar | OUTPUT | Old density |
| OLD_TEMP | Scalar | OUTPUT | Old temperature |
| OPACITY_A | Opacity | OUTPUT | $\tau_a$ - Absorption opacity - units of $\frac{1}{length}$ |
| OPACITY_R | Opacity | OUTPUT | $\tau_r$ - Rosseland opacity - units of $\frac{1}{length}$ |
| OPACITY_S | Opacity | OUTPUT | $\tau_s$ - Scattering opacity - units of $\frac{1}{length}$ |

For example,

```
material 1 DDGAS
   ...
   model = 14
end

model 14 xsn  $ DD-GAS
   number of elements 1
      element 1, mass 2.0, fraction 1.0
   end
end

material 3 CH
   ...
   model = 36
end

model 36 xsn  $ CH + 5% O + 0.25% Br
   number of elements 4
      element  1, mass  1.0079, fraction 0.47375
      element  6, mass 12.0110, fraction 0.47375
      element  8, mass 15.9994, fraction 0.05
      element 35, mass 79.9040, fraction 0.0025
   end
end
```

### 12.7.3 Tabular Opacity

```
MODEL model_id_number TABULAR OPACITY
   NUMBER OF ELEMENTS int
      ELEMENT int, MASS real, FRACTION real
   END
   [parameter = value]
   ...
END
```

`TABULAR OPACITY` provides frequency-dependent emission and absorption coefficients for a material or mixture of materials using $\log_{10}$ bilinear interpolation. For materials with a $Z \leq 36$, the tables are built using Propaceos (Prism Inc.). Tables for materials with $Z > 36$ are built using `XSN`. Most of the tables range from $0.01 eV$ to $10^5 eV$ and $10^{-7}\rho_s$ to $10^2\rho_s$.

The simplest way to use this model is to specify a predefined material found in the `Opacity_Index.dat` file in the `$ALEGRA_MIGDATA` directory. Element materials may be specified by either the `MATERIAL` sub-keyword or else by the atomic number of the element in the `NUMBER OF ELEMENTS` sub-keyword group. For example, the following two inputs for iron are equivalent.

```
model 21 tabular opacity
   material = 'iron'
end

model 21 tabular opacity
   number of elements 1
     element 26, mass 55.847, frac 1.0
   end
end
```

For compounds the simplest way to specify the material is to request a predefined material (if available) found in the `Opacity_Index.dat` file in the `$ALEGRA_MIGDATA` directory, otherwise the user can let ALEGRA choose and mix the elements specified in `NUMBER OF ELEMENTS` sub-keyword group. In the latter case, the total opacities are computed by mixing the cross sections according to the mass fractions $f_m$ of the elements. Mass fractions are determined by the code from the input atomic fractions $f_a$ as specified `NUMBER OF ELEMENTS` keyword group. For example, polyethylene may be specified as:

```
model 21 tabular opacity
   material = 'ch'
end
```

or:

```
model 21 tabular opacity
   number of elements 2
     element 1, mass 1.0079, frac 0.5
     element 6, mass 12.011, frac 0.5
   end
end
```

There may be a significant difference between these two methods. Predefined compounds use Propaceos (or XSN) in a stand-alone mode to calculate the line to line transition probabilities in all materials before mixing, grouping, and writing data files to be read by ALEGRA. Using ALEGRA to mix grouped tables may be prone to inaccuracies. The farther the element numbers are from each other, the greater these inaccuracies will be.

Each `MATERIAL` defined in the `Opacity_Index.dat` file may have one or more opacity `TABLE`s associated with it. `TABLE` = 1 is usually the best table to use. This is the default. Other tables may be chosen if available.

The `TABULAR OPACITY` model regroups the cross-sections according to:

$$\sigma_{a_G} = \frac{\sum_G \sigma_{a_g} \int_g BB(T,g)dg}{\int_G BB(T)dG} \tag{100}$$

because the groups are assumed to constant across each sub-group (which is technically true once the original table has been compiled).

- Modules: material_libs/opac_models
  - tabular_opacity.h
  - tabular_opacity.C
- Data Files: `$ALEGRA_MIGDATA` directory
  - Opacity_Index.dat

- Physics:
  - emission
  - radiation and all derived physics classes

Table 46: Input Parameters for `TABULAR OPACITY`.

| Parameter Name | Type | Description |
|---|---|---|
| `MAXIMUM TEMPERATURE` | real | Max temperature to be used for interpolation (default = `REAL_MAX`/2.0) |
| `MINIMUM TEMPERATURE` | real | Min temperature to be used for interpolation (default = 0.0) |
| `MAXIMUM DENSITY` | real | Max density to be used for interpolation (default = `REAL_MAX`/2.0) |
| `MINIMUM DENSITY` | real | Min density to be used for interpolation (default = 0.0) |
| `TABLE` | int | Which material table, if more than one, to use (default = 1) |
| `RATIO` | real | Multiply all opacities in a table by a const. Useful for changing ”`CH`” into ”`CH2`” by `RATIO` = 0.928 (default = 1.0) |
| `DYNAMIC INTEGRATION` | none | Changes group bounds to $g_0 = 0.12299$ and $g_f = 11.6088$ and uses these bounds to calculate the opacity. *The user must set `NUMBER OF GROUPS` = 1* |
| `ZBAR` | none | Can return the `ZBAR` usually present in the opacity tables |
| `NUMBER OF ELEMENTS` | int | if different from `MATERIAL` elements |
| `DATAFILE` | string | Can point to a specific `DATAFILE` |
| `FILE` | string | Same as `DATAFILE` |
| `MATERIAL` | string | Call a specific material file such as ”`CH2`” |

Example to read a user supplied Propaceous file:

```
model 2 tabular opacity
   dynamic integration
   file = '../Titanium'
end
```

**Table 47.** Registered Plot Variables for `TABULAR OPACITY`.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | Scalar | `INPUT` | Material density |
| `TEMPERATURE` | Scalar | `INPUT` | Material temperature |
| `ZBAR` | Scalar | `OUTPUT` | material ionization (0<zbar<Z) |
| `OPACITY_A` | Opacity | `OUTPUT` | $\tau_a$ - Absorption opacity - units of $\frac{1}{length}$ |
| `OPACITY_R` | Opacity | `OUTPUT` | $\tau_t$ - Rosseland opacity - units of $\frac{1}{length}$ |
| `OPACITY_S` | Opacity | `OUTPUT` | $\tau_s$ - Scattering opacity - units of $\frac{1}{length}$ |

`DYNAMIC INTEGRATION` uses the fact that 99% of the energy in the black body function is within the bounds of $0.12299T < T < 11.6088T$. When calculating the opacities in `XSN`, there are a maximum of only 100 integration points, and this was an attempt to ensure a greater probability of having those points land on transition lines. This was especially true for lower temperatures where the black body function is very narrow. Because the `TABULAR OPACITY` input files have already been group averaged, this is only included for completeness.

More examples:

```
model 3 tabular opacity
   $ material defined in the MATERIAL keyword group
   table = 1                        $ default table
   minimum temperature = 3.0e3
   maximum temperature = 1.0e8
   minimum density     = 3.0e-3
   maximum density     = 1000.0e3
   ratio = 0.928                    $ modify opacity (CH2 = 0.928 CH)
end


model 2 tabular opacity
   material = 'air'
   number of elements 2
      element 1, mass  1.007, fraction 0.66666
      element 6, mass 12.011, fraction 0.33334
   end
   zbar
   minimum temperature = 3.0e3
   maximum temperature = 1.0e8
   minimum density     = 3.0e-3
   maximum density     = 1000.0e3
end
```

## 12.8 Ionization Models

### 12.8.1 Saha Ionization

```
MODEL model_id_number SAHA IONIZATION
   [parameter = value]
   ...
END
```

The `SAHA IONIZATION` model [49] solves for the relative abundance of nuclei in adjacent charge states and for the average ionization state, $\bar{Z}$, of a hot material. This model obtains its necessary information from the `NUMBER OF ELEMENTS ... END` block of the `MATERIAL` specification.

The ionization state is found by solving a nonlinear set of equations relating the fractional number of ions in each ionization state.

$$\frac{\alpha_{m+1}\alpha_e}{\alpha_m} = C_{Saha}\left(\frac{A}{\rho}\right)\left(\frac{u_{m+1}}{u_m}\right)\left(T^{3/2}\right)\exp\left(\frac{-I_{m+1}}{T}\right) \tag{101}$$

$$1 = \sum_{m=0}^{Z} \alpha_m \quad \text{and} \quad \alpha_e = \sum_{m=0}^{Z} m\alpha_m \tag{102}$$

where $\alpha_m = N_m/N$ represents the fraction of ions in the $m^{th}$ charge state and $N = \rho N_A/A$ is the number density of nuclei. $N_m$ represents the number density of nuclei in the $m^{th}$ charge state. $\alpha_e = N_e/N$ represents the ratio of free electrons to nuclei and also the average ionization state, $\bar{Z}$. $A$ is the atomic mass, $\rho$ is the density in g/cm$^3$, $u_m$ is essentially the electronic partition function (assumed to be equal to the statistical weight of the statistical weight of the ground state, i.e., 2 for the two electron spin states), $T$ is the temperature in keV, and $I_m$ is the ionization potential of the $m^{th}$ charge state also in keV.

$$C_{Saha} = \frac{2}{N_A}\left(\frac{2\pi m_e k_B}{h^2}\right)^{3/2} = 317.013 \tag{103}$$

where $N_A$ is the Avogadro's number, $m_e$ is the electron mass, $h$ is Planck's constant, and $k_B$ is essentially Boltzmann's constant converting the temperature from keV to ergs.

- Modules: material_libs/ec
  - matmod_saha.h
  - matmod_saha.C
  - saha_equation.F
- Physics: any physics requiring ionized materials

**Table 48.** Input Parameters for `SAHA IONIZATION`.

| Parameter Name | Type | Description |
|---|---|---|
| `TEMPERATURE CUTOFF` | real | Minimum temperature used to evaluate the ionization state |

**Table 49.** Registered Plot Variables of `SAHA IONIZATION`.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | real | `INPUT` | Material density |
| `TEMPERATURE` | real | `INPUT` | Material temperature |
| `ZBAR` | real | `OUTPUT` | Ionization state |

Example:

```
model, 5, saha ionization
  $ no parameters necessary
end
```

### 12.8.2 KZB Sesame Ionization

```
MODEL model_id_number KZB SESAME
   [parameter = value]
   ...
END
```

Ionization state as found in tabular form in Kerley Sesame format [19]. Internally this is the 601 Sesame table. This is not really a model. It is a software interface. The origins of any table used here must be carefully researched and understood.

- Modules: material_libs/kerley_eos/sesame
  - szb_mig.h
  - szb_mig.C
  - szbmig.F
- Physics: any physics requiring ionized materials

**Table 50.** Input Parameters for `KZB SESAME`.

| Parameter Name | Type | Description |
|---|---|---|
| `FZB` | String | Table file name |
| `NZB` | int | Table Number |

**Table 51.** Registered Plot Variables of `KZB SESAME`.

| Variable Name | Type | Mode | Description |
|---|---|---|---|
| `DENSITY` | real | `INPUT` | Material density |
| `TEMPERATURE` | real | `INPUT` | Material temperature |
| `ZBAR` | real | `OUTPUT` | Ionization state |

Example:

```
model, 5, kzb sesame
   fzb = 'ses600'
   nzb = 3109
end
```

# 13 Diagnostics

## 13.1 Global Diagnostic Variables

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 are the values of the kinetic, internal, magnetic, and radiation 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? A global energy balance can be constructed as follows.

$$E_{error}(t) = E_{tot}(t) - [E_{tot}(0) + E_{sources}(t) - E_{losses}(t)] = 0 \tag{104}$$

where

$$E_{tot}(t) = E_{int}(t) + E_{kin}(t) + E_{mag}(t) \tag{105}$$

Ideally the error in this energy balance should be zero, or at least small compared to most energy tallies. Deviations from zero are typically due to missing energy tallies, numerical inaccuracies, or perhaps too large of a time step.

**Table 52.** Global Tallies for All Physics Options.

| Global Variable Name | Explanation |
|---|---|
| ETOT<br>PTOT* | Total of the kinetic internal, magnetic, and radiation energies and the rate of total energy change. |

Table 53: Global Tallies for TRANSIENT MAGNETICS.

| Global Variable Name | Explanation |
|---|---|
| DT_MAG | Overall limiting time step for TRANSIENT MAGNETICS |
| DT_MAGDIFF | Suggested limiting time step for resolving magnetic diffusion. It is essentially one e-folding time for problems where the magnetic field varies exponentially in time. In SI units, $\tau_{magdiff} = (\mu\sigma h^2)/4$, where $\mu$ is the permeability, $\sigma$ is the conductivity, and $h$ is a characteristic cell size. |
| DT_ALFVEN | Limiting time step based upon Alfven wave speed. |
| CURRENT | Current flowing in the mesh as tallied by the CURRENT TALLY boundary conditions. Present only if CURRENT TALLY commands are specified. |

*continued on next page*

*continued from previous page*

| | |
|---|---|
| INDUCTANCE | Inductance of the mesh computed from the total magnetic field energy, EMAG, and the total CURRENT tally. Present only if CURRENT TALLY boundary conditions are specified. In SI units, $\frac{1}{2}LI^2 = E_{mag} = \frac{1}{2\mu_0}\int B^2 dV$. |
| RESISTANCE | Resistance of the mesh computed from the total joule heating rate, PJOULE, and the total CURRENT tally. Present only if CURRENT TALLY boundary conditions are specified. In SI units, $RI^2 = P_{Joule} = \int \eta J^2 dV$. |
| EMAG<br>PMAG | Magnetic field energy and rate of change. |
| EJOULE<br>PJOULE | Cumulative tally of the energy transferred to the material via Joule heating and rate of heating. |
| EJOULV<br>PJOULV | Cumulative tally of the energy transferred to the void via Joule heating and rate of heating as a consequence of small, but finite, void conductivities. This energy and power should be small compared to other energies and powers for a valid solution. |
| EUDJXB<br>PUDJXB | Cumulative tally of the energy transferred to the material via work done by $\vec{J} \times \vec{B}$ forces and rate of change. |
| EMAGHYDRO<br>PMAGHYDRO | Cumulative tally of the change in magnetic field energy due to Lagrangian motion of the mesh and rate of change. (2D only) |
| EMAGWORK<br>PMAGWORK | Cumulative tally of the energy transferred to the material via work done by forces and rate of change. Typically these tallies equal the sum of the EUDJXB and EMAGHYDRO tallies (2D only) |
| EPOYNT<br>PPOYNT | Cumulative tally of the energy transferred to the system via Poynting flux through the boundary of the mesh and rate of change. Since this tally typically represents a source of energy, it should be added to the energy balance equation. This tally is positive if energy is added to the mesh and negative if energy is lost from the mesh. |
| ETDERR<br>PTDERR | Time discretization energy and rate of change. This energy error originates naturally when constructing the fully-implicit numerical algorithm used to solve the MHD equations. It should be small compared to other energies and powers for a valid solution. (2D only) |

*continued on next page*

| *continued from previous page* | |
|---|---|
| EMAGRZERR | Error in the magnetic field energy as a result of advection, i.e., remap. It should be small compared to other energies for a valid solution. There is no corresponding power. |

Table 54 explains the key to decipher the output names for the circuit model as it is implemented. All circuit output variables are in SI units even though the circuit model may be used with a simulation that is in either SI or CGS units. Circuit model output names are composed of three parts: a prefix, a root, and a suffix. The root part designates the type of circuit component, either a circuit node or a circuit element. The prefix indicates the type of information represented by the tally. The suffix identifies the particular circuit node or circuit element for the tally.

The ”I_” current tallies are associated with circuit elements. Circuit elements are attached to circuit nodes. The sum of all currents coming into a circuit node should be zero (unless it is a voltage source or ground node). Kirchhoff’s law for currents is enforced. If a circuit node is attached to the output of a circuit element, then the element current is added to the sum. If a circuit node is attached to the input of a circuit element, then the element current is subtracted from the sum. Any discrepancies in the sum are stored in the ”IG_NODE_” variables, nominally referred to as a ”current to ground.” These ”IG_NODE_” variables can indicate a bad DASPK solve if they are not small compared to other currents. Ideally they would be zero.

The ”V_” voltage tallies are associated with circuit nodes. The difference in voltages between two adjacent circuit nodes should match the corresponding ”DV_” tally for the circuit element that connects the nodes. Kirchhoff’s law for zero voltage drop around a closed loop is enforced in this piecewise manner.

The ”E_” energy variables are the energy stored in circuit elements in the case of capacitors and inductors, the energy dissipated in the case of resistors, and the Poynting energy in the case of the mesh circuit element.

The ”_MQSMESH_” variables and the circuit solution depend upon the correctness of any symmetry factors should they be required, i.e., the VOLUMETRIC SCALE FACTOR input keyword.

External inductances, resistances, and capacitances are typically constants.

For example, the MQS element having an input node 2 and an output node 1 representing the mesh has the following variables present in the EXODUS and HISPLT output files: DV_MQSMESH_2_1, I_MQSMESH_2_1, E_MQSMESH_2_1, L_MQSMESH_2_1, and R_MQSMESH_2_1.

Table 54: Global Tallies for CIRCUIT SOLVER.

| Global Variable Name | Explanation |
|---|---|
| **Prefix** | |
| V_ | Circuit node voltage (Volts). |
| IG_ | Circuit node current to ground (Amperes). |
| I_ | Circuit element current (Amperes). |
| L_ | Circuit element inductance (Henrys). |
| R_ | Circuit element resistance (Ohms). |
| E_ | Circuit element energy (Joules). |
| DV_ | Circuit element voltage drop between input and output nodes (Volts). |
| **Root** | |
| _NODE_ | Circuit node. |
| _MQSMESH_ | Mesh element (MQS = magnetoquasistatics). |
| _RES_ | Resistor element. |
| _IND_ | Inductor element. |
| _CAP_ | Capacitor element. |
| _ZFLOW_ | Zflow resistor element. |
| _CS_ | Current source element. |
| _DET_ | Detonator resistor element. |
| _SW_ | Switch resistor element. |
| _TSW_ | Tracer switch element. |
| _VAR_ | Varistor resistor element. |
| **Suffix** | |
| _nn | Node identifier. |
| _ii_oo | Element identifier (ii = input node, oo = output node). |

**Table 55.** Global Tallies for THERMAL CONDUCTION.

| Global Variable Name | Explanation |
|---|---|
| DT_CON | Overall limiting time step for THERMAL CONDUCTION. The reported time step does not include scaling by the TIME STEP SCALE keyword so the user may readily see the time step that thermal conduction would normally employ. |

**Table 56.** Global Tallies for EMISSION.

| Global Variable Name | Explanation |
|---|---|
| DT_EMISSION | Limiting time step for EMISSION. This time step is factored into the DT_HYDRO time step. |
| ERADEMIT<br>PRADEMIT* | Cumulative tally of the radiation energy emitted by the material and the rate of emission due to the EMISSION model. Since this tally represents a gain to the radiation energy, it should be added to the initial radiation energy in energy balance equations. |
| DIST_TO_VOID | A report of the mean-distance-to-void used to compute the reabsorption factor in the emission model. |
| OPTICAL_DEPTH_MIN<br>OPTICAL_DEPTH_MAX | A report of the minimum and maximum optical depth used to compute the reabsorption factor in the emission model. The optical depth is defined to be the absorption opacity times the mean-distance-to-void. |

Table 57: Global Tallies for `RADIATION`

| Global Variable Name | Explanation |
|---|---|
| `DT_RAD` | Overall limiting time step for radiation physics. |
| `ERAD`<br>`PRAD` | Current value of the radiation field energy and rate of change. |
| `ERADEMIT`<br>`PRADEMIT` | Cumulative tally of the radiation energy emitted by the material and the rate of emission. Since this tally represents a gain to the radiation energy, it should be added to the initial radiation energy in energy balance equations.<br>The difference between the radiation energy absorbed and the radiation energy emitted equals the change in material internal energy (unless some other physics also modifies the internal energy). |
| `ERADBC`<br>`PRADBC` | Cumulative tally of the radiation energy emitted by boundary sources and the rate of emission. Since this tally represents a gain to the radiation energy, it should be added to the initial radiation energy in energy balance equations. |
| `ERADSRC`<br>`PRADSRC` | Cumulative tally of the radiation energy emitted by volumetric or point sources and the rate of emission. Since this tally represents a gain to the radiation energy, it should be added to the initial radiation energy in energy balance equations. |
| `ERADABS`<br>`PRADABS` | Cumulative tally of the radiation energy absorbed by the material and the rate of absorption. Since this tally represents a loss to the radiation energy, it should be subtracted from the initial radiation energy in energy balance equations.<br>The difference between the radiation energy absorbed and the radiation energy emitted equals the change in material internal energy (unless some other physics also modifies the internal energy). |
| `ERADLEAK`<br>`PRADLEAK` | Cumulative tally of the radiation energy leaked through the mesh boundaries and the rate of leakage. (Named analogously to the leakage term from neutron transport theory.) Since this tally represents a loss to the radiation energy, it should be subtracted from the initial radiation energy in energy balance equations.<br>Leakage through the boundary will soon be tallied by sideset. |

Table 58: Global Tallies for TWO TEMPERATURE

| Global Variable Name | Explanation |
|---|---|
| DT_TWO_T | Limiting time step for TWO TEMPERATURE. This time step is factored into the DT_HYDRO time step. |
| TAUEI_MIN<br>TAUEI_MAX | A report of the minimum and maximum ion-electron equilibration times. These tallies do not affect the time step control, but may be of interest to the user. |
| DT_CON_ION<br>DT_CON_ELE | Limiting time step for THERMAL CONDUCTION tallied separately for ions and electrons. The reported time steps do not include scaling by the TIME STEP SCALE keyword so the user may readily see the time step that thermal conduction would normally employ. |
| EINT_ION<br>PINT_ION | Internal energy and the rate of change associated with electrons if the TWO TEMPERATURE option is enabled. |
| EINT_ELE<br>PINT_ELE | Internal energy and the rate of change associated with ions if the TWO TEMPERATURE option is enabled. |
| EPDV_ION<br>PPDV_ION | Time-integrated PdV energy and instantaneous rate of change for ions. The PdV energy is the time-integrated work done by the ion pressure on the material. This work will manifest itself as ion internal energy. This energy is recoverable. If the rate of change is positive the ion internal energy is increasing, otherwise if the rate of change is negative, the ion internal energy is decreasing. The sum of the ion and electron contributions equals the total PdV work. |
| EPDV_ELE<br>PPDV_ELE | Time-integrated PdV energy and instantaneous rate of change for electrons. The PdV energy is the time-integrated work done by the electron pressure on the material. This work will manifest itself as electron internal energy. This energy is recoverable. If the rate of change is positive the electron internal energy is increasing, otherwise if the rate of change is negative, the electron internal energy is decreasing. The sum of the ion and electron contributions equals the total PdV work. |

*continued on next page*

| *continued from previous page* | |
|---|---|
| `ENONPDV`<br>`PNONPDV` | Time-integrated non-PdV energy and instantaneous rate of change. These are the same tallies as the one-temperature case, however the energy couples only to the ions since it is the ions that determine the hydrodynamic behavior of the fluid. The non-PdV energy is the time-integrated work done by the artificial viscosity and hourglass forces on the material. This work will manifest itself as ion internal energy. The rate of change should be positive since these are dissipative forces and the ion internal energy should be increasing. |
| `ETAUEI`<br>`PTAUEI` | Energy transferred from electrons to ion and the rate of energy transfer due to electron-ion collisions if the `TWO TEMPERATURE` option is enabled. A negative value indicates energy is being transferred from ions to electrons. |

# References

[1] G. O. Allshouse. 3-D code project. SNL internal memorandum (to J. P. VanDevender), November 1988.

[2] G. O. Allshouse. 3-D code status. SNL internal memorandum (to J. P. VanDevender), April 1988.

[3] G. O. Allshouse. 3-D code plan draft report, (U). SNL unpublished report (SRD), February 1989.

[4] R. L. Bell et al. CTH user's manual and input instructions, version 4.00. Technical report, Sandia National Laboratories, Albuquerque, NM, 87185, April 1999.

[5] P. B. Bochev, , C. J. Garasi, J. J. Hu, A. C. Robinson, and R. S. Tuminaro. An improved algebraic multigrid method for solving Maxwell's equations. *SIAM Journal of Scientific Computing*, 25(2):623–642, 2003.

[6] P. B. Bochev, J. J. Hu, A. C. Robinson, and R. S. Tuminaro. Towards robust 3D Z-pinch simulations: discretization and fast solvers for magnetic diffusion in heterogeneous conductors. *Electronic Transactions on Numerical Analysis (ETNA)*, 15:186–210, 2003. http://etna.mcs.kent.edu.

[7] P. B. Bochev and A. C. Robinson. Matching algorithms with physics: exact sequences of finite element spaces. In Donald Estep and Simon Tavener, editors, *Collected Lectures on the Preservation of Stability under Discretization*, chapter 8. SIAM, 2002.

[8] T. A. Brunner. Forms of approximate radiation transport. Technical report SAND2002-1778, Sandia National Laboratories, Albuquerque, NM, 87185, July 2002.

[9] T. A. Brunner. Nodal radiation diffusion in ALEGRA. Technical report, Sandia National Laboratories, Albuquerque, NM, 87185, November 2004. to be published.

[10] T. A. Brunner and T. A. Mehlhorn. A user's guide to radiation transport in ALEGRA-HEDP, version 4.6. Technical report SAND-2004-5799, Sandia National Laboratories, Albuquerque, NM, 87185, November 2004.

[11] K. G. Budge and J. S. Peery. RHALE-2D: An ALE shock code. unpublished report, 1990.

[12] S. K. Carroll et al. ALEGRA: Version 4.6. Technical report SAND2004-6541, Sandia National Laboratories, Albuquerque, NM 87185, December 2004.

[13] M. P. Desjarlais. Practical improvements to the Lee-More conductivity near the metal-insulator transition. *Contributions to Plasma Physics*, 41(2-3):267–270, March 2001.

[14] M. P. Desjarlais, J. D. Kress, and L. A. Collins. Electrical conductivity for warm, dense aluminum plasmas and liquids. *Physical Review E*, 66(2):025401(R), August 2002.

[15] C. W. Gear and L. R. Petzold. ODE methods for the solution of differential/algebraic systems. *SIAM Journal on Numerical Analysis*, 21(4):716–728, August 1984.

[16] Hagen and Rubens. title unknown. *Ann. d. Physic*, 11:873, 1903.

[17] T. A. Haill and A. C. Robinson. Two-dimensional finite-element magnetohydrodynamic equations in ALEGRA, part I: Magnetic field formulation. draft SAND report, Sandia National Laboratories, Albuquerque, NM 87185, 2005. to be published.

[18] T. A. Haill and A. C. Robinson. Two-dimensional finite-element magnetohydrodynamic equations in ALEGRA, part II: Vector potential formulation. draft SAND report, Sandia National Laboratories, Albuquerque, NM 87185, 2005. to be published.

[19] E. S. Hertel and G. I. Kerley. CTH reference manual: The equation of state package. Technical report SAND98-0947, Sandia National Laboratories, Albuquerque, NM 87185, 1998.

[20] K. S. Holian. T-4 handbook of material properties data bases, volume 1c: Equations of state. Technical report LA-10160-MS, Los Alamos National Laboratory, Los Alamos, NM, November 1984.

[21] J. J. Hu, R. S. Tuminaro, P. B. Bochev, C. J. Garasi, and A. C. Robinson. Toward an H-independent algebraic multigrid method for Maxwell's equations. *SIAM Journal on Scientific Computing*, July 2004. Accepted submit to revision.

[22] J. D. Huba. NRL plasma formulary. Technical report, Naval Research Laboratory, Washington, D.C., 1994.

[23] R. E. White III and C. L. Sarazin. Numerical models of star formation in x-ray cluster cooling flows. *Astrophysical Journal*, 318(2):629–644, July 1987.

[24] J. D. Jackson. *Classical Electrodynamics*. John Wiley & Sons, Inc., New York, second edition, 1975.

[25] H. Knoepfel. *Pulsed High Magnetic Fields*. North-Holland Publishing Company, Amsterdam, 1970.

[26] Y. T. Lee and R. M. More. An electron conductivity model for dense plasmas. *Physics of Fluids*, 27(5):1273, May 1984.

[27] R. W. Lemke et al. Self-consistent, two-dimensional, magnetohydrodynamic simulations of magnetically driven flyer plates. *Physics of Plasmas*, 10(5):1867–1874, May 2003.

[28] C. D. Levermore and G. C. Pomraning. A flux-limited diffusion theory. *The Astrophysical Journal*, 248:321–334, August 1981.

[29] S. Li and L. Petzold. Software and algorithms for sensitivity analysis of large-scale differential algebraic systems. *Journal of Computational and Applied Mathematics*, 125(1-2):131–145, December 2000.

[30] W. A. Lokke and W. H. Grasberger. XSNQ-U – a non-LTE emission and absorption coefficient subroutine. Technical report UCRL-52276, Lawrence Livermore National Laboratory, Livermore, CA, January 1977.

[31] S. P. Lyon and J. D. Johnson. SESAME: The Los Alamos National Laboratory equation of state database. Technical report LA-UR-92-3407, Los Alamos National Laboratory, Los Alamos, NM, 1992.

[32] J. M. McGlaun and T. G. Trucano. Proposal for development of a rad/hydro code. Sandia National Laboratories unpublished report, October 1989.

[33] G. N. Minerbo. Maximum entropy Eddington factors. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 20:541, 1978.

[34] R. M. More. Atomic physics in inertial confinement fusion. Technical report UCRL-84991 preprint, Lawrence Livermore National Laboratory, Livermore, CA, March 1981.

[35] R. M. More, K. H. Warren, D. A. Young, and G. B. Zimmerman. A new quotidian equation of state (QEOS) for hot dense matter. *Physics of Fluids*, 31(10):3059–3078, October 1988.

[36] L. Petzold. Differential/algebraic equations are not ODE's. *SIAM Journal on Scientific and Statistical Computing*, 3(3):367–384, September 1982.

[37] G. Pollak. Detailed physics of XSN-U opacity package. Technical report LA-UR-90-2423, Los Alamos National Laboratory, Los Alamos, NM, January 1990.

[38] G. C. Pomraning. *The Equations of Radiation Hydrodynamics*. Pergamon Press, Oxford, England, 1973.

[39] A. C. Robinson and C. J. Garasi. Three-dimensional z-pinch wire array modeling with ALEGRA-HEDP. *Computer Physics Communications*, 164, 2004.

[40] D. D. Ryutov, M. S. Derzon, and M. K. Matzen. The physics of fast Z pinches. *Reviews of Modern Physics*, 72(1):167–223, January 2000.

[41] L. Spitzer, Jr. *Physics of Fully Ionized Gases*. Interscience Publishers, second edition, 1962. 4th printing, January 1967.

[42] L. M. Taylor and D. P. Flanagan. PRONTO-2D, a two-dimensional transient solid dynamics program. Technical report SAND86-0594, Sandia National Laboratories, Albuquerque, NM, 87185, March 1987.

[43] L. M. Taylor and D. P. Flanagan. PRONTO-3D, a three-dimensional transient solid dynamics program. Technical report SAND87-1912, Sandia National Laboratories, Albuquerque, NM, 87185, March 1989.

[44] C. Tong and R. S. Tuminaro. ML 2.0 Smoothed Aggregation User's Guide. Technical report SAND2001-8028, Sandia National Laboratories, Albuquerque, NM 87185, December 2000.

[45] T. G. Trucano. Possible five year plan. SNL internal memorandum (to G. A. Allshouse), October 1988.

[46] R. S. Tuminaro et al. Official Aztec Users's Guide - Version 2.1. Technical report SAND99-8801J, Sandia National Laboratories, Albuquerque, NM 87185, November 1999.

[47] J. P. VanDevender. 3-D radiation hydrodynamics code development for inertial confinement fusion at Sandia. SNL internal memorandum (to distribution), May 1988.

[48] J. P. VanDevender. Three-dimensional radiation hydrodynamic code development at Sandia. SNL internal memorandum (to V. Narayanamurti), February 1989.

[49] Y. B. Zel'dovich and Y. P. Razier. *Physics of Shock Waves and High-Temperature Hydrodynamic Phenomena*. Academic Press, New York, 1966.

# A Historical Origins

The origins of the ALEGRA shock wave physics code project can be found in two internal Sandia memorandums that were written in the spring of 1988. In the first of these [2], G. O. Allshouse briefly stated needs for a 3-D radiation hydrodynamics code aimed primarily at light ion fusion capsule design. In the second memo [47], J. P. VanDevender, then director of the Sandia Light Ion Fusion and Pulsed Power Program, called for the planning of a code development project to develop a 3-D radiation hydrodynamics capability at Sandia to support the Sandia inertial confinement fusion program.

The latter memo initiated a two phase planning process for the code that would ultimately become ALEGRA. In the first phase, T. G. Trucano participated in a series of discussions intended to define a specific approach for a new 3-D radiation-hydrodynamics code at Sandia. These discussions included Trucano, Allshouse, A. Badruzzaman, and G. R. Montry, and meetings extended through the summer of 1988. By November 1988, Trucano had produced a skeleton summary with which to continue further planning [45]. Allshouse responded to the planning work up to this point with a firm recommendation to proceed with code development [1]. Further discussions of this very expensive and extensive project continued into February of 1989, without specific decisions, culminating in a stronger call for the start of such a project by VanDevender [48].

VanDevender's memo proved to be an important milestone for this project. In the second planning phase, Trucano and J. M. McGlaun were tasked for the remainder of Fiscal Year 1989 to prepare a specific and detailed code development proposal. Two key documents mark the planning process during this time. First, a draft detailed requirements document for the proposed radiation hydrodynamics code was written by Allshouse [3]. Second, a formal detailed written proposal was released in October of 1989 [32]. The major content of this proposal was orally presented for funding approval to a group of managers including VanDevender earlier in the year. Thus, the start of this project was the product of a very small group of people, with Allshouse playing the role of a true visionary, and McGlaun the major figure in translating that vision into a specific and viable code development proposal.

It is curious to point out also that during the planning of McGlaun and Trucano in the spring of 1989, M. K. Matzen was interviewed. At that time, Matzen specifically requested that the intended project be flexible enough to allow a development path that could include magneto-hydrodynamics physics necessary for computational modeling of fast Z-pinch physics, if deemed to be necessary in the future. The remarkable prescience of this request, before a single line of code had even been written, is now clear since the light ion program gave way to the Z pinch program in the 1996 time frame.

At this point, it is important to reiterate some of the main assumptions that entered into the formal proposal [32]. These points will help the reader to better understand some of the history further related below.

- The project was viewed as expensive and long term (five years at least).
- The project was intended to be an advanced development project for the computational shock wave physics effort as Sandia as well as for the ICF effort. Funding from both areas was to be applied to the project.
- The project was intended to start with an existing code.
- The new code would need to be capable of modeling large changes in length scales (capsule implosions) and turbulence to satisfy some of the requirements laid out in the requirements document [48], as well as complicated coupling between the hydrodynamics and other physical phenomena important to ICF. Complex 3-D geometries were expected to be of importance in the code applications.
- The code development strategy, as well as the software implementation, needed to be flexible and open-ended. (This ultimately was a major contributing factor to the decision to break new ground and use C++ as the major development language for the project by 1992. See below.)
- The code was to be developed for massively parallel computing platforms, although the machine architecture that this might include was not at all obvious in 1989.
- Pre- and post-processing tools were not to be developed under specific expenditures of this project, because of scarcity of resources, as well as the existence of additional projects intended to provide solutions in these areas.

The knowledgeable reader will understand the significance of most of these assumptions. It is remarkable to see how much of the current architecture and capabilities of ALEGRA can trace their origins back to these ideas. Of course, this is the kind of return one should get from a two year planning activity.

Based on these assumptions, it was concluded that the best way to meet the prescribed objectives was to write a block-structured, arbitrary connectivity multi-material arbitrary Lagrangian Eulerian (MMALE) code. The chosen development kernel with which to start was the PRONTO code [42, 43], a Lagrangian arbitrary (but fixed) connectivity (finite element) block-structured 3-D code in existence at Sandia at the time of the planning culmination. (See below.) The Lagrangian mode would accommodate changes in length scale with a conforming mesh, while the Eulerian mode (to be added) would accommodate the complex shearing and fluid distortion associated with turbulent flows.

The proposal also regarded the new code as a potential and logical next-generation extension of the CTH code [4], a production 3-D Eulerian code developed by Sandia and in wide use by the early 1990s. Though generally highly successful, CTH proved inadequate for certain classes of problems, such as capsule implosion simulations. Needless to say, a variety of new physics was required that was not implemented in either CTH or PRONTO: radiation transport, electron-ion two-temperature fluid approximations to plasma flows, electron thermal conduction, fusion burn physics, charged particle beam deposition, and potentially MHD physics associated with fast Z-pinches. We can also not overstate the novelty of applying ”standard” Eulerian shock hydrodynamics algorithms on an arbitrary connectivity finite element mesh in this project.

The code project which was born during the two year period of 1988 and 1989 finally started formally in March of 1990. An early 2D Fortran incarnation called RHALE was available later that same year. [11] In the late fall of 1990, the decision was made to recast RHALE into an object-oriented structure using the C++ programming language. The advantages of this change included: enhanced data structures and memory management, object-oriented programming constructs, excellent debugging and code development environments, and improved ability to use massively parallel computing hardware.

We are now more than ten years after the initial requests for exploring issues in developing a 3-D radiation hydrodynamics code, and there is no end in sight to the ALEGRA project. Two key elements of our original vision for this project remain quite firm in the current identity of ALEGRA. First, the code continues to serve as a platform for many computational shock wave physics R&D advances at Sandia. Second, the code continues to progress toward the goal of providing a primary theory tool for the Sandia Pulsed Power Program.

# Index

## DISTRIBUTION:

1 Musk, LTC Jeffrey H.
Department of Physics
MADN-PHYS
Building 753, Bartlett Hall
U.S. Military Academy
West Point, NY 10996-1790

1 MS 0370
T. G. Trucano, 9211

1 MS 0378
W. J. Bohnhoff, 9231

5 MS 0378
S. Carroll, 9231

1 MS 0378
R. R. Drake, 9231

1 MS 0378
D. M. Hensinger, 9231

1 MS 0378
D. A. Labreche, 9231

1 MS 0378
C. B. Luchini, 9231

1 MS 0378
S. J. Mosso, 9231

1 MS 0378
S. V. Petney, 9231

5 MS 0378
A. C. Robinson, 9231

1 MS 0378
J. Robbins, 9231

1 MS 0378
R. M. Summers, 9231

1 MS 0378
T. E. Voth, 9231

1 MS 0378
M. K. Wong, 9231

1 MS 0557
T. W. Simmermacher, 9124

1 MS 1110
P. B. Bochev, 9214

1 MS 1152
S. J. Chantrenne, 1642

1 MS 1168
C. Deeney, 1646

1 MS 1181
J. P. Davis, 1646

1 MS 1186
R. B. Campbell, 1674

1 MS 1186
P. J. Christenson, 1674

1 MS 1186
K. R. Cochrane, 1674

1 MS 1186
M. P. Desjarlais, 1674

1 MS 1186
C. J. Garasi, 1674

5 MS 1186
T. A. Haill, 1674

1 MS 1186
R. J. Lawrence, 1674

1 MS 1186
R. W. Lemke, 1674

1 MS 1186
T. K. Mattsson, 1674

1 MS 1186
T. A. Mehlhorn, 1674

1 MS 1186
B. V. Oliver, 1674

1 MS 1186
K. Peterson, 1674

1 MS 1186
S. A. Slutz, 1674

1 MS 1186
R. A. Vesey, 1674

1 MS 1186
E. C. Wemlinger, 1674

1 MS 1186
E. P. Yu, 1674

1 MS 1191
M. K. Matzen, 1600

1 MS 1191
M. A. Sweeney, 1670

1 MS 1194
S. E. Rosenthal, 1644

1 MS 1194
E. M. Waisman, 1644

1 MS 9159
J. J. Hu, 9214

1 MS 9159
R. S. Tuminaro, 9214

2 MS 0899
Technical Library, 9616

1 MS 9018
Central Technical Files, 8945-1