SANDIA REPORT

SAND2004-5799
Unlimited Release
Printed January 2005

A User's Guide to Radiation Transport in ALEGRA-HEDP, Version 4.6

Thomas A. Brunner, Thomas A. Mehlhorn

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.

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

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

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.doe.gov/bridge

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

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online ordering:	http://www.ntis.gov/ordering.htm

SAND2004-5799
Unlimited Release
Printed January 2005

A User's Guide to Radiation Transport in ALEGRA-HEDP, Version 4.6

Thomas A. Brunner
Thomas A. Mehlhorn
HEDP Theory/ICF Target Design
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1186

Abstract

This manual describes the input syntax to the ALEGRA radiation transport package. All input and output variables are defined, as well as all algorithmic controls.

Intentionally left blank

Contents

Figures

Intentionally left blank

A User's Guide to Radiation Transport in ALEGRA-HEDP, Version 4.6

1 Introduction

This manual describes the radiation input syntax for ALEGRA-HEDP. The ALEGRA manual[2] describes how to run the code and general input syntax. The ALEGRA-HEDP manual[13] describes the input for other physics used in high energy density physics simulations, as well as the opacity models used by this radiation package. An emission model, which is the lowest order radiation transport approximation, is also described in the ALEGRA-HEDP manual. This document is meant to be used with these other manuals.

1.1 The Transport Equation

The Boltzmann transport equation describes how a variety of different types of particles travel through a material. The Boltzmann equation is very general, and every discipline has a different way to write it that fits their needs best. The intensity $I(\mathbf{r}, \Omega, \varepsilon, t) = ch\nu f$ is the primary variable, where $f(\mathbf{x}, \Omega, \varepsilon, t)$ is a phase space density, ν is the photon frequency with energy $\varepsilon = h\nu$, h is Planck's constant, and c is the speed of light. For simplicity, the one-group (energy integrated) Boltzmann equation for radiation transport is [7, 10, 5, 1]

$$\frac{1}{c}\frac{\partial I}{\partial t} + \Omega \cdot \nabla I = -\sigma_t I + \frac{\sigma_s}{4\pi}\int_{4\pi} I \, \mathrm{d}\Omega' + \sigma_a \frac{c}{4\pi} B(T_m) + S. \tag{1}$$

Eq. 1, which describes the evolution of the intensity of the photons, is coupled to an equation that describes the energy content of the material, namely

$$\frac{\partial u_m}{\partial t} = -\int_{4\pi} \sigma_a \left(\frac{c}{4\pi} B(T_m) - I \right) \, \mathrm{d}\Omega + Q_m. \tag{2}$$

In Eq. 1 and Eq. 2, Ω is the unit angle vector, u_m is the material energy density, $\sigma_t = \sigma_s + \sigma_a$[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

$$B(T_m) = \frac{8\pi^5 k^4}{15 h^3 c^3} T_m^4 = a T_m^4, \tag{3}$$

[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.

where a is the black body constant, and k is Boltzmann's constant.

1.2 Useful Quantities

Not only is Eq. 1 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. 2 is only through the radiation energy density, which is the integral of I over all angles, or

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

where E is radiation energy density.

Another useful quantity is the net flux of energy, which is the first angular moment of I, or

$$\mathbf{F} = \frac{1}{c}\int_{4\pi} \Omega I \, \mathrm{d}\Omega \tag{5}$$

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

$$P = \int_S c\mathbf{F}\cdot\hat{\mathbf{n}}\,\mathrm{d}A, \tag{6}$$

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

We can also define partial fluxes of energy flowing the the positive or negative direction as

$$\mathbf{F}^{\pm} = \frac{1}{c}\int_{\pm\Omega\cdot\hat{\mathbf{n}}>0} \Omega I \, \mathrm{d}\Omega \,, \tag{7}$$

where the integral over angle is only done over angles in the direction of $\hat{\mathbf{n}}$ for the positive partial flux and similarly for the negative flux. A power through the surface in a particular direction can be computed by using Eq. 6 and using $\mathbf{F}^{\pm}$ instead of $\mathbf{F}$, namely

$$P^{\pm} = \pm\int_S c\mathbf{F}^{\pm}\cdot\hat{\mathbf{n}}\,\mathrm{d}A, \tag{8}$$

where the extra $\pm$ in front is to ensure that $P^{\pm} \geq 0$. The net power is $P = P^{+} - P^{-}$.

2 General Input

The radiation transport physics specification keyword is *RADIATION*. The radiation block has all the keywords of the *ENERGETICS* block, as specified in the ALEGRA User's Manual[2]. Within the *RADIATION* block, there must be exactly one other block that selects the particular transport method. While the keywords to select specific transport methods will be discussed later, all transport methods share common keywords. These common keywords describe the physics parameters of the simulation. Algorithmic controls are left to the particular methods.

The *runid*.out file lists most variable and their units available for output. It also lists all user input and most defaults specified in input deck.

2.1 Algorithm Control

It is possible to turn the radiation transport calculation off after a user specified time with the command

SKIP SOLVE AFTER TIME = real

If the simulation time is after the real number specified here, the radiation solve will be skipped. All radiation energy and power tallies will be set to zero. This is a dangerous option to use unless you are certain that the radiation is not important after a given time. There will be a spike in the net leak energy tally as all remaining radiation energy leaves the mesh instantly.

2.2 Group Bounds

The energy dependent transport equation is discretized in energy into multiple energy groups. Even if the transport method itself doesn't discretize energy, the opacities are, so this section must always be specified. The energy boundaries of these groups are specified with the *GROUP BOUNDS* keyword. The energies are specified in temperature units of Kelvin. While this is not the most likely choice of users, it allows the use of unit designators to select between eV and keV.

GROUP BOUNDS
  LINEAR l_1 *TO* u_1 *by* n_1
  LOG l_2 *TO* u_2 *by* n_2
  ...
END

The two sub-keywords *LINEAR* and *LOG* may be repeated any number of times in the *GROUP BOUNDS* block. The ranges specified by *LINEAR* and *LOG* must increase and cannot overlap. The ending value of a range must the same as the starting value of the next range, otherwise an error is produced; overlapping or disjoint ranges are not allowed.

For *LINEAR* ranges, the group bounds are determined by

$$\Delta g = \frac{u_1 - l_1}{n_1} \tag{9}$$

$$g_{\text{next}} = l_1 + i\Delta g, \tag{10}$$

where Δg is the divisions in the group bounds, g_{next} is the next group bound added to the list, and $0 \leq i \leq n_1 + 1$. For *LOG* ranges, the group bounds are determined by

$$\Delta g' = \frac{\ln u_2 - \ln l_2}{n_2} \tag{11}$$

$$g_{\text{next}} = \mathrm{e}^{\ln l_2 + i\Delta g'}. \tag{12}$$

For example specifying

```
group bounds
   linear 0.0 [kev] to 10.0 [kev] by 5
   log 10.0 [kev] to 100000.0 [kev] by 4
end
```

results in group bounds at 0, 2, 4, 6, 8, 10, 10^2, 10^3, 10^4, and 10^5 keV. This is a really silly set of group bounds, it is meant as an example only.

The first and last bound will be ignored by the transport methods. They will always treat the lowest bound as zero and the highest bound as infinity so that the entire energy range of photons is covered. The specified lowest bound and the highest bound are used by the material models to determine the multigroup opacities.

If you have an idea of the temperatures that will occur in the simulation, the lowest bound should be approximately one tenth the lowest temperature and the highest bound should be about ten times the highest temperature. It is also helpful to put group boundaries near edges and other features in the opacity. Fewer, carefully chosen bounds can be as accurate as many, less carefully chosen bounds; run times can be significantly reduced if you choose the group bounds wisely.

2.3 Initial Conditions

For methods that support it, initial conditions can be specified. The *INITIAL CONDITIONS* keyword allows you to specify initial conditions on mesh blocks using

INITIAL CONDITIONS, [BLOCK id,] subtype

where the *[BLOCK id]* is optional; if omitted the condition applies to the entire mesh. Only one initial condition allowed per block of mesh.

The only current subtype of initial condition is *UNIFORM TEMPERATURE* which takes as a parameter a temperature in Kelvin,

INITIAL CONDITIONS, BLOCK n, UNIFORM TEMPERATURE real

Where the block specification *n* is a valid block id.

In some circumstances, it may be desirable to perform a steady state calculation to initialize the radiation field before the simulation starts. This is selected by specifying

STEADY STATE INITIALIZATION

Any conditions specified with *INITIAL CONDITIONS* are used as an initial guess to the linear solver for the steady state solution; good initial conditions can improve performance. The steady state solution is obtained by assuming that materials have a constant temperature, even though they can absorb and emit photons with (possibly) unequal rates. The specified boundary conditions are evaluated at the starting simulation time.

2.4 Boundary Conditions

All transport methods share a common set of boundary conditions. While each transport method has a natural default boundary condition, the default is not guaranteed to be the same for all transport methods. Therefore, it is very important that boundary conditions are specified on all boundaries. Assume that no default exists. Only one boundary condition may be specified for any part of the surface of the mesh. Not all transport methods support all boundary condition types; the transport method will warn and abort if a particular type is unsupported.

2.4.1 Albedo Boundaries

Albedo boundaries specify that a certain fraction, α, of the radiation that would otherwise leave the system is reflected back into the system. These are useful for approximating walls that heat up without actually simulating them. The syntax is

ALBEDO BOUNDARY, SIDESET id, function

The function lists the fraction α as a function of time. The values of the albedo must satisfy $0 \leq \alpha \leq 1$. All radiation energy groups have the same albedo. The scale factor of the function scales the albedo. Angular information is not preserved by albedo boundaries.

2.4.2 Reflecting Boundaries

Reflecting boundaries do not allow any radiation to leave the system. This is a zero flux boundary, and is useful for taking advantage of symmetry in the system. Angular information is preserved by reflecting boundaries. The syntax is

REFLECTIVE BOUNDARY, SIDESET id

2.4.3 Source Boundaries

Source boundaries specify how much radiation is entering the system. The amount of radiation leaving the system is not specified.

TEMPERATURE SOURCE BOUNDARY, SIDESET id, function1, function2

The source has a black body spectrum at the temperature in Kelvin, as specified in the first function. The scale factor of the function scales the temperature. The second function scales the power of the black body function as a function of time. This is useful if the black body emitter that is driving your system occupies less than full field of view, but for most simulations this should probably be set to one.

Alternatively, one can specify the temperature for the spectrum of the incoming radiation as well as the total power density, integrated over all energies. This is done with

TEMPERATURE SPECTRUM SOURCE BOUNDARY,
SIDESET id, function1, function2

The temperature of the spectrum is given by the first function, but the total input power density is given by the section function. Units of the temperature are in Kelvin, units for the power density are power per area.

2.4.4 Dirichlet Boundaries

The radiation energy density is specified at the boundary. This is mostly useful for verification tests of deterministic transport methods, and has little value for real simulations.

TEMPERATURE DIRICHLET BOUNDARY, SIDESET id, function

The energy density is a black body at the temperature in Kelvin specified in the function. The scale factor of the function scales the temperature.

2.4.5 Vacuum Boundaries

Specifies that no radiation enters the system. The amount of radiation that leaves the system is unspecified.

VACUUM BOUNDARY, SIDESET id

2.5 Volumetric Sources

The *VOLUMETRIC SOURCE* specifies an arbitrary source term for the radiation field. This can be used to model the effects of sources of radiation without actually modeling the source. The basic syntax is

VOLUMETRIC SOURCE, [BLOCK id,] subtype

The *BLOCK id* specifier is optional; if omitted, the source is applied to the entire mesh. Multiple sources are allowed on the same section of mesh; the contributions from each are summed together.

The *UNIFORM TEMPERATURE* subtype specifies that a uniform source over the block is applied. Two parameters are expected, a power density and a function with a temperature in Kelvin. The source will have a black body spectrum at the specified temperature, but normalized so that the total power per unit volume added to the system is given by the power density.

VOLUMETRIC SOURCE, [BLOCK id,]
UNIFORM TEMPERATURE, real, function

where the real value is the power density.

2.6 Flux Tallies

Flux tallies compute net power through an an arbitrary surface of the mesh using Eq. 6. You can specify flux tallies with

FLUX TALLY, "stringid", number_of_sidesets, sideset1, sideset2, ..., [SCALE s]

The *stringid* can be comprised of letters, numbers, and underscores. The string will be converted to all upper case. Multiple side sets can be used to make a flux tally. An integer number should specified for *number_of_sidesets*. Then specify *number_of_sidesets* side sets. The energy and power reported by the flux tally are scaled by the default scale factor used by the rest of the energy tallies. You can override the default scale factor by specifying your own scale with the optional *SCALE s* after all the sidesets.

The side must be a one sided side set, and can be interior to the mesh, and the orientation of the surface determines whether the net radiation flow will be positive or negative through the surface. Positive values of the flux tallies indicate a net flow "out" of the surface. In two dimensions, it is necessary in Cubit to specify extra information when creating a side set so that it is one sided, for example "`sideset 4 curve 3 wrt surface 1`". In three dimensions, Cubit by default creates a one sided side set.

Several output variables are created to tally the power and time integrated flow of power through the surface. They are

PFLUX_stringid is the net power evaluated at the current time step of radiation through the surface using Eq. 6.

PFLUXP_stringid is the power evaluated at the current time step of radiation through the surface in the positive direction using Eq. 8. *PFLUXP_stringid* is nonnegative.

```
flux tally, "TOP", 2, sideset 5, sideset 2
flux tally, "SIDE", 1, sideset 3
flux tally, "MIDPLANE", 3, sideset 5, sideset 1, sideset 4, scale 2.0
```

Code Sample 1. Sample usage of flux tallies.

PFLUXN_stringid is the power evaluated at the current time step of radiation through the surface in the negative direction using Eq. 8. *PFLUXN_stringid* is nonnegative.

EFLUX_stringid is the time integrated net energy that has crossed the surface, or

$$\hat{E}_{\text{flux}} = \int_{t_0}^{t} P_{\text{flux}} \, \mathrm{d}t' \tag{13}$$

EFLUXP_stringid is the time integrated energy that has crossed the surface integrated in the positive direction.

EFLUXN_stringid is the time integrated energy that has crossed the surface integrated in the negative direction.

The variables *PFLUXN_stringid* and *PFLUXP_stringid* will not be zero at a reflective boundary, but they will be equal to each other. Also, $P_{\text{flux}} = P_{\text{positive flux}} - P_{\text{negative flux}}$.

Code Sample 1 shows an example of how to use flux tallies. Note that the same side set can be specified in multiple flux tallies.

2.6.1 A Warning About Flux Tallies in Nodal Diffusion Methods

In the nodal diffusion methods, the flux tallies are calculated with a volume integral that is approximately equivalent to the surface integral. Side sets, as created by Cubit and other mesh programs, are a set of pairs of surfaces and elements which define the inside of the surface. The volume of integration is done over these elements in the side set. Figure 1 shows two possible integration volumes (the areas enclosed by the red and green lines) for the same side set, which is represented by the blue line.

Due to the details of the integration, there is a contribution to the tally of flux crossing the solid red and green lines that should not be included. This is an inaccuracy in the tally, and can be significant. The error can be reduced greatly if you choose your flux tally surfaces well. First, extend the side set so that it ends where there is no (or little) flux. In Figure 1, the side set goes all the way to the boundary;

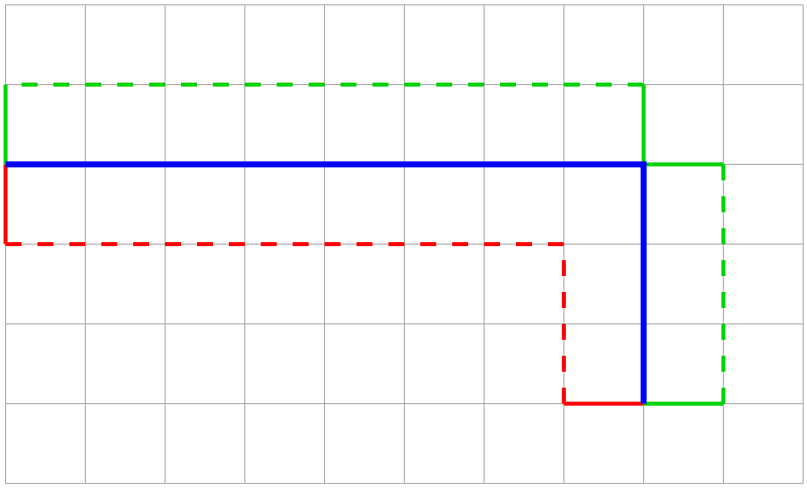

Figure 1. The blue line is the surface on which the flux tally is requested. The elements enclosed by the red or green lines define two different senses for the inside of the surface. In the nodal diffusion methods, the tally is performed with a volume integral on the elements that define the inside. There is an erroneous contribution to the tally from the flux crossing the solid red or green lines. While the results of the tally should be symmetric when choosing either side as the inside, they are not in this calculation. In this case, defining the red side to be the inside results in a smaller error.

if this boundary is reflective, there would be no error at this end. Second, which side is defined as the inside can be important as well. Defining the green side as the inside leads to two extra surfaces that can have error relative to the red side. While the flux tallies should be symmetric when computed with either side of the surface defined as the inside, in the diffusion methods, this will not be true.

2.7 Units

All values taken by parameters are in the unit system specified by the *UNITS* keyword. All temperatures are in Kelvin, lengths in meters or centimeters, etc. Because the official code units are not always convenient for the user, ALEGRA allows you to specify units for any real-valued parameter. Unit specification is is described in detail in the ALEGRA User's Manual[2]. Code Sample 2 shows how to specify temperatures in the input deck for radiation problems. The only caveat is that unit designators cannot be used within function, but they can be used to specify a scale factor for an entire function. All temperatures throughout the input

```
units, si

radiation
   linearized flux limited diffusion
      initial conditions, uniform temperature 0.001 [keV]
      group bounds
         linear 300.0 [K] to 5.0 [eV] by 12
      end
   end
end

material 1
   temperature 1.0 [eV]
end

material 2
   temperature 300.0 [K]
end
```

Code Sample 2. An example of how to use unit designators to specify temperature throughout a simple input deck.

deck should be specified with these commands; doing so will ensure that a consistent set of conversion factors have been used. If a unit designator is omitted, it is assumed that the number is already in code units.

2.8 Output Variables

To make any physics package useful, there must be some form of output. Various variables are available to you for output in the Exodus (or other) output files. For a complete list of available variables, see the file *runid*`.out`; this file also shows what units the variables have. Some output variables are common to all transport methods, and some are specific to the method. The common ones are

DT_RAD is the time step size chosen by the transport method. The ALEGRA time step is limited by the smallest *DT_x* calculated by each of the physics.

ERAD is total radiation energy energy in the mesh. It is calculated using

$$\hat{E}_{\text{rad}} = \int_{\Omega} \sum_{g} E_g \, \mathrm{d}V, \tag{14}$$

where E_g is the radiation energy density in group g, Ω is the problem domain, and $\hat{E}_{\text{rad}}$ is *ERAD*. Units of energy.

PRADEMIT is the radiation power emitted from the material evaluated over the current time step. It is

$$P_{\text{emit}} = \int_{\Omega} \sum_{g} \sum_{m} c\sigma_{a,m}^{g} B_g(T_m) \, \mathrm{d}V, \tag{15}$$

where c is the speed of light, $\sigma_{a,m}^{g}$ is the absorption opacity (with units of inverse length) for material m and group g, and $B_g(T_m)$ is the black body function for group g evaluated at the material temperature T_m. $P_{\text{emit}} > 0$. Units of power.

PRADABS is the radiation power absorbed by the material evaluated over the current time step. It is

$$P_{\text{abs}} = \int_{\Omega} \sum_{g} \sum_{m} c\sigma_{a,m}^{g} E_g \, \mathrm{d}V. \tag{16}$$

$P_{\text{abs}} > 0$. Units of power.

PRADNETLEAK is net power lost by the system through the boundaries, namely Eq. 6 integrated over the surface of the mesh,

$$P_{\text{net leak}} = \int_{S} c\mathbf{F} \cdot \hat{\mathbf{n}} \, \mathrm{d}A. \tag{17}$$

where S is the entire boundary of the mesh, $\mathbf{F}$ is the net flux through the surface defined in Eq. 5, and $\hat{\mathbf{n}}$ is the outward unit normal to the surface. $P_{\text{net leak}} > 0$ for net loss out of the system. Units of power.

PRADSRC is the power deposited by all of the volumetric sources in the mesh.

$$P_{\text{source}} = \int_{\Omega} \sum_{g} S_g \, \mathrm{d}V. \tag{18}$$

where S_g is the group-wise power density source. Units of power.

PRADBC is the power injected into the system by the source boundary condition. This quantity, while saved as output, is not correct for parallel IMC simulations. Units of power.

For every power tally, there is a corresponding energy tally that time integrates the power over simulation, or

$$\hat{E}_{\text{tally}} = \int_{t_0}^{t} P_{\text{tally}} \, \mathrm{d}t', \tag{19}$$

where t_0 is the simulation begin time, and t is the current simulation time.

ERADEMIT is the time integrated energy emitted by the material. Units of energy.

ERADABS is the time integrated energy absorbed by the material. Units of energy.

ERADNETLEAK is the time integrated, net energy lost by the system through the boundaries. Units of energy.

ERADSRC is the time integrated energy added by the volumetric source. Units of energy.

ERADBC is the time integrated energy added by source boundary conditions. This quantity, while saved as output, is not correct for parallel IMC simulations. Units of energy.

ERADERROR is an estimate of the energy over the current time step that was lost. Ideally this value should be zero. For some transport methods, independent tallies for all of the above variables are not available, so this term will be identically zero. Note that this is the only *ERAD...* variable that is not time integrated.

$$\hat{E}_{\text{rad error}} = \hat{E}_{\text{rad}}(t_n) - \hat{E}_{\text{rad}}(t_{n-1}) - \Delta t \left(P_{\text{emit}} - P_{\text{abs}} + P_{\text{source}} - P_{\text{net leak}} \right) \tag{20}$$

Units of energy.

ERADERRORSUM the time integrated value of *ERADERROR*. Units of energy.

DEDT_RAD is an element centered variable for each material. It is the net power given to each material over the time step by the radiation field. It is computed with an integral over each element, namely

$$\frac{\partial}{\partial t} E_{\text{radiation to material}} = f_m c \sum_g \int_{V_e} \sigma_{a,m}^g (E_g - B_g(T_m)) \, \mathrm{d}V, \tag{21}$$

where f_m is the volume fraction of material m. A positive value indicates there is a net flow of energy to the material from the radiation.

This is only stored if requested for output and is advected by the remap routine in Eulerian or ALE calculations.

RAD_SOLVE_TIME is the maximum amount of CPU time used by any single processor in a parallel run to advance the radiation to the next time step. This global variable is only active if *DETAILED SOLUTION STATISTICS* is specified in the input deck.

2.9 Extra Information

To get information about the solution statistics, like solve time, specify the keyword

DETAILED SOLUTIONS STATISTICS

The radiation transport methods have a debug mode. Turning this feature on will print more data than is useful for anybody. It is meant as a tool for developers only; its existence is included in the manual for completeness. If you are brave, this mode is selected with

DEBUG MODE: RADIATION

This is specified at the input deck "file scope", outside of all physics blocks.

3 Linearized diffusion

One of the main transport models in ALEGRA is the linearized flux limited diffusion. This model solves for the radiation energy density at the nodes of the mesh, and is described in detail in a separate theory document[12]. Create a *LINEARIZE DIFFUSIOND* block in the *RADIATION* block to select this method. All common keywords, such as boundary conditions, should be inside the *LINEARIZED DIFFUSION* block as well as the method specific keywords listed below.

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

$$I(\mathbf{r},\Omega,t) = \frac{1}{4\pi}E + \frac{3}{4\pi}\Omega\cdot\mathbf{F}, \tag{22}$$

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

$$\mathbf{F} = \int_{4\pi} \Omega I \, \mathrm{d}\Omega \approx -\frac{1}{3\sigma_t}\nabla E. \tag{23}$$

Eq. 22 and Eq. 23 lead to the diffusion equation, namely

$$\frac{1}{c}\frac{\partial E}{\partial t} - \nabla\cdot D\nabla E = \sigma_a\left(4\pi B(T_m) - E\right) + S_E. \tag{24}$$

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 $1/3\sigma_t$ in Eq. 23 and Eq. 24 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 [8, 6].

There has been a fundamental change in the form of the equations; the transport equation (Eq. 1) 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.

3.1 Algorithm Control

All the keywords of the general transport method are available to set boundary conditions, initial conditions and sources. Several extra keywords are available to control this method as well.

3.1.1 Flux Limiters

Several flux limiters[11, 9] are available to use; each is selected by using the *FLUX LIMITER* keyword. The flux limiter is calculated separately for every energy group. The current default flux limiter is the *LARSEN 2* flux limiter. While there are many flux limiters to choose from, the choice of a particular flux limiter does not matter much in practice. Results from all flux limiters are very similar when compared to results from the full transport equation.

All flux limiters limit to the standard diffusion approximation as $R \to 0$ and limit to free streaming as $R \to \infty$, where

$$R = \frac{|\nabla E|}{\sigma_t E}. \tag{25}$$

While the free streaming limit makes the most sense in one dimension, it seems to help in multiple dimensions as well. The free streaming limit is

$$D = \frac{E}{|\nabla E|} \tag{26}$$

so that in one dimension, the diffusion equation becomes

$$\frac{1}{c}\frac{\partial E}{\partial t} + \frac{\partial E}{\partial x} = S \tag{27}$$

where S is the stuff on the right hand side of the equation. All flux limiters use the free streaming D in Eq. 26 if $R > 1000$. Figure 2 shows the opacity normalized diffusion coefficient as a function of R for different flux limiters.

The Larsen flux limiter[9, 11], is selected with

```
FLUX LIMITER = LARSEN int
```

where the integer is the parameter n in Eq. 28 below. The special case where $n = 1$ is also known as the sum flux limiter. The Larsen limited diffusion coefficient is

$$D_{\mathrm{L}} = \frac{1}{\left[(3\sigma_t)^n + \left(\frac{|\nabla E|}{E}\right)^n\right]^{1/n}} \tag{28}$$

This is the current default flux limiter.

The simplified Levermore-Pomraning flux limiter[6] can be selected with

```
FLUX LIMITER = SIMPLIFIED LEVERMORE POMRANING
```

Using this flux limiter, the simplified Levermore-Pomraning diffusion coefficient is calculated using

$$D_{\mathrm{LP}} = \frac{1}{\sigma_t}\frac{1}{R}\left(\coth R - \frac{1}{R}\right) \approx \frac{1}{\sigma_t}\frac{2+R}{6+3R+R^2} \tag{29}$$

Because the hyperbolic functions in Eq. 29 are expensive to evaluate, it is approximated with with the rational approximation on the right side of the equation. Since flux limited diffusion is an approximation of the transport equation, it doesn't matter much if we're a little bit off the "exact" formulation.

Regular diffusion can be selected by using

```
FLUX LIMITER = NONE
```

The standard diffusion coefficient is

$$D = \frac{1}{3\sigma_t}, \tag{30}$$

where σ_t is the total (Rosseland mean) opacity with units of inverse length. The diffusion coefficient, D, has units of length.

The Minerbo flux limiter[3] can be selected with

```
FLUX LIMITER = MINERBO
```

This flux limiter is derived from the maximum entropy variable Eddington factor proposed by Minerbo[8]. The Minerbo diffusion coefficient is defined by

$$\frac{\coth Z - 1/Z}{1-\coth^2 Z + 1/Z^2} = R \tag{31}$$

$$D_{\mathrm{M}} = \frac{1}{\sigma_t}\left(1-\coth^2 Z + \frac{1}{Z^2}\right). \tag{32}$$

Given a value for R, Eq. 31 must be solved for Z before evaluating D_{M}. In practice, the rational approximation to D_{M} is used, namely

$$D_{\mathrm{M}} \approx \frac{1}{\sigma_t}\left(\frac{1}{3} - \frac{R^2}{15 - 6.248R^{0.5385} + 9R + 3R^2}\right) \tag{33}$$

The maximum error of this approximation is about 0.4%.

For all flux limiters, the diffusion coefficient is limited to be no larger than one hundred times the maximum length scale of the mesh. This was chosen to make the matrix well behaved for the linear solver while not changing the problem from a pure vacuum too much. This default factor can modified with

MAXIMUM DIFFUSION COEFFICIENT FACTOR = real

The default is one hundred. The length scale of the mesh is determined by ALEGRA. A two dimensional mesh that is one meter square will have a length scale of $\sqrt{2}$ meters, resulting in a maximum diffusion length of $100\sqrt{2}$ meters.

3.1.2 Time Step Control

The time step control for the linearized diffusion package can be turned off with the keyword

TURN OFF TIME STEP CONTROL [= integer]

The optional integer is used to specify for how many cycles the control is ignored. (For example, setting the integer to 3 will cause the control to start potentially limiting the time step for the third cycle.) If this keyword is not specified at all, the control is active for all time steps. If the optional integer is omitted, the time step control is deactivated for all time steps. A negative integer will also deactivate the control for all time steps. Even if the control is turned off, the time step is computed and stored in the global variable *DT_RAD*.

The time step control calculates the next time step size is based on the current time step and the change in the radiation field between time steps, so wave fronts can be captured. This control works in a vacuum, but can be very susceptible to noise from opacities, the linear solver, and other sources. Additionally, if a problem is nearing a steady state solution in the radiation field, the radiation time step will grow without bound.

The first step is to find the normalized change of the radiation energy density at each node using

$$\eta_i = \left| \frac{\sum_g E^n_{g,i} - \sum_g E^{n-1}_{g,i}}{\sum_g E^n_{g,i} + \sum_g E^{\min}_g} \right|, \tag{34}$$

where $E^n_{g,i}$ is the energy density for a particular node and group at time step n. The factor $E^{\min}_g$ ensures that a divide by zero will not occur, and also allows unimportant regions of the problem to not influence the time step. The maximum

$\eta_{max} = \max \eta_i$ on the mesh is found, and the new (potential) time step is computed using

$$\Delta t_\eta^{n+1} = \Delta t^n \sqrt{\frac{\eta_{target}}{\eta_{max}}}. \tag{35}$$

The default value is $\eta_{target} = 0.04$, but this can be changed with

MAXIMUM ENERGY DENSITY CHANGE = real

The value of Δt_η^{n+1} is stored in the global variable *DT_RAD*. The nodal variable *RAD_ETA* stores a spatial map of the relative change in the radiation field, which is useful to determine which part of the mesh is reducing the time step.

The factor E_g^{min} is computed for each group by finding the maximum energy density in the mesh for a particular group and multiplying by a factor. This factor can be set with

MINIMUM ENERGY DENSITY FACTOR = real

and has a default value of 10^{-14}. Increasing this value will effectively remove nodes with low energy densities, relative to the maximum energy density, from the time step computation. The default value is set low enough to include most nodes. If cold regions of the problem are controlling the time step unnecessarily, increase the value of *MINIMUM ENERGY DENSITY FACTOR* before increasing *MAXIMUM ENERGY DENSITY CHANGE*. It might be best to set this to the same value as the tolerance of the linear solve or higher, assuming the values mean the same thing.

3.1.3 Linear Solver Control

The linear solver control is set with

AZTEC SET int

where *int* refers to an Aztec control set defined outside of the *RADIATION* block. The details of controlling Aztec are described in the ALEGRA HEDP manual[13]. The multilevel preconditioner does very well with this method, and should be used for all but the smallest of problems (no more than a few hundred elements) where direct solver may be faster. If the multilevel preconditioner has trouble with a particular problem, please let the radiation developers know; the problem should be corrected. Use the conjugate gradient solver, and the symmetric row sum scaling. The tolerance should be set relatively low, 10^{-12} or smaller.

3.1.4 Partial Group Solve

Many problems start out cold, but will get hot later in time, so a large energy group range is needed. However, at early times, some of the higher energy groups may contribute no energy in the radiation field. By default, all energy groups are solved, even if there is no energy in them. With the *PARTIAL GROUP SOLVE* keyword, you can skip the solve for some groups. Estimates of the total source energy emitted during the next time step as well as the radiation energy in each group are used to turn off (or on) group solves. For the sources this is

$$E^g_{\text{all sources}} = \int \left[S + \sum_m c\sigma^g_{a,m} B_g(T_m) \right] \mathrm{d}V + \Delta t \int P_{\text{source boundary conditions}} \, \mathrm{d}A, \tag{36}$$

and for the radiation energy this is

$$E^g_{\text{radiation}} = \int E_g \, \mathrm{d}V. \tag{37}$$

If any group has more than a certain fraction of either the source energy or the radiation energy, the linear solve is performed. Otherwise, the energy density in that group is set to zero everywhere. For each group, if

$$\frac{E^g_{\text{all sources}}}{\sum_g E^g_{\text{all sources}}} > f \quad \text{or} \quad \frac{E^g_{\text{radiation}}}{\sum_g E^g_{\text{radiation}}} > f, \tag{38}$$

then the group is solved.

This option is turned on using

PARTIAL GROUP SOLVE [real]

where an optional real value sets the value of f. By default, $f = 0$, so only groups with exactly zero energy are skipped. Keep f as small as possible to allow your problems to run, otherwise unexpected problems may arise. For example, if you have a very narrow group that is intended to resolve a small detail of the the opacity, the group may be skipped because there is not much total energy in that narrow group.

This option will reduce the performance of the diffusion package if all the groups are active, so it should only be used when you expect groups to be skipped. However, even skipping just a few group solves during the entire simulation can save a considerable amount of run time.

3.1.5 Pure Eulerian Mode

ALEGRA typically operates by computing all physics in a Lagrangian frame during a time step, then remapping the solution variables back to the original mesh to

complete the Eulerian step. This can sometimes be problematic for the radiation, and specifying

PURE EULERIAN

will turn off all hydrodynamic motion terms in the diffusion solve.

3.2 Output Variables

Several variables are available for output.

RAD_ENERGY_DENSITY is a node centered radiation energy density E_g. It has units of energy per volume. The energy density is an array with the length the same as the number of energy groups.

RAD_ENERGY_DENSITY_OLD is a node centered radiation energy density from the last time step. You should never need this variable, but if you use tracers in a Hisplot file, you will get it anyway.

RAD_TEMPERATURE is an element centered value of the radiation temperature in Kelvin. When in equilibrium with the material, this should be the same as the material temperature. The radiation temperature in an element is related to the nodal energy density through

$$T_r = \left[\frac{1}{aV_e} \int_{V_e} \sum_g E_g \, \mathrm{d}V \right]^{1/4}, \tag{39}$$

where a is the black body constant and V_e is the volume of an element.

This is only computed and stored if requested for output.

RAD_ETA is the node value of η in Eq. 34 used for time step control. This is useful to find out what parts of the problem are controlling the time step. This variable has different meaning than variables with the same name in different transport methods.

This is only stored if requested for output.

RAD_DIFFUSION_COEFFICIENT is the element centered value of D in Eq. 24. There is one value for each group. It includes the flux limiter contributions; it should also be limited by the *MAXIMUM DIFFUSION COEFFICIENT FACTOR*.

This is only stored if requested for output.

RAD_AZ_TIME is the time reported by Aztec for the linear solve. This is summed over all groups and is only output if *DETAILED SOLUTION STATISTICS* was specified in the input deck.

RAD_SOLVED indicates if a particular group was solved for when *PARTIAL GROUP SOLVE* is active. Its value is zero if the group was skipped or one if the group was solved as normal. This variable is active only if *DETAILED SOLUTION STATISTICS* was specified in the input deck.

RAD_GROUP_ENERGY is the radiation energy in each group. It is only active if both *PARTIAL GROUP SOLVE* and *DETAILED SOLUTION STATISTICS* were specified in the input deck.

RAD_SOURCE_ENERGY is the estimated energy emitted by all sources for a particular group. It includes material emission, volumetric sources, and boundary conditions. It is only active if both *PARTIAL GROUP SOLVE* and *DETAILED SOLUTION STATISTICS* were specified in the input deck.

3.3 An Example

Code Sample 3 shows an example of a nearly complete input deck. Several important parts have been deleted, like material specification.

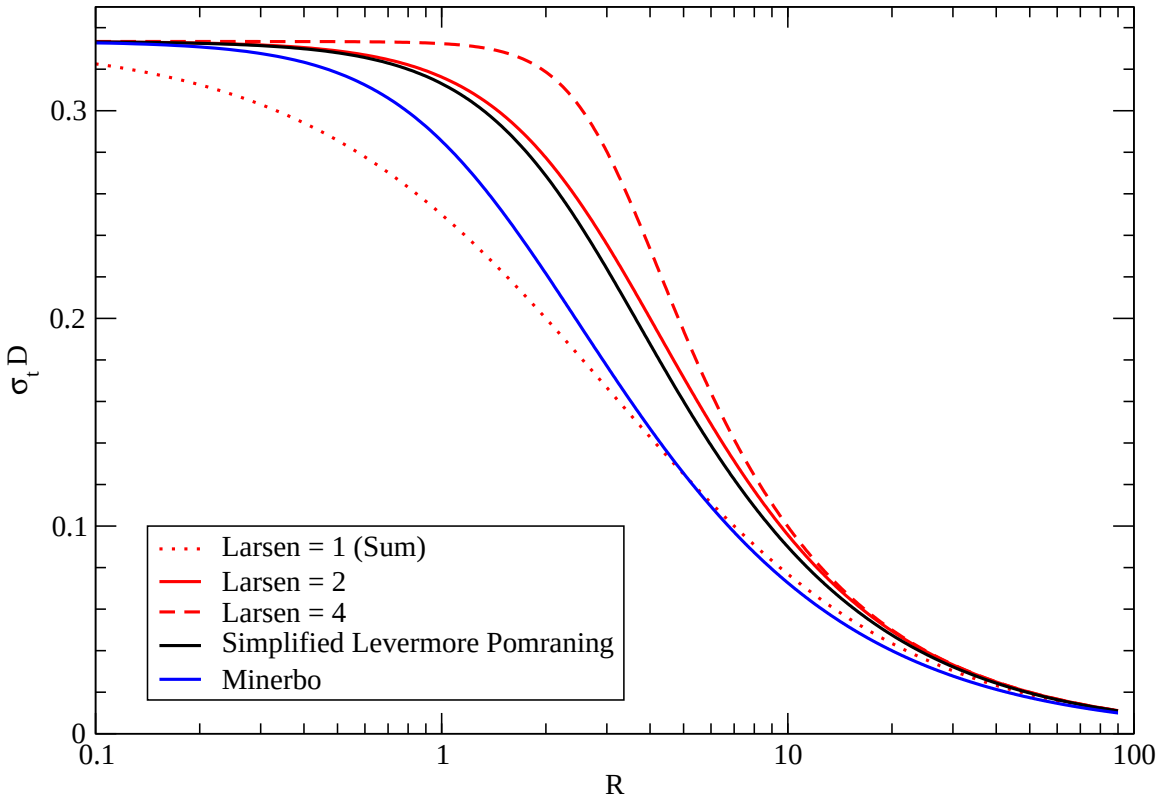

Figure 2. Opacity normalized diffusion coefficients as a function of the normalized gradient $R = |\nabla E|/\sigma_t E$ for various flux limiters. All limit to standard diffusion as $R \to 0$ and become the same (free streaming) value as $R \to \infty$. This plot does not indicate if one flux limiter is better than any other for a given simulation; it is simply a comparison of how much limiting each flux limiter does for a particular set of inputs. For example, the Larsen=1 flux limiter is generally the most restrictive, but this may not always be ideal.

```
radiation

   Cartesian 2d
   maximum initial time step 3.0e-14

   linearized diffusion

      group bounds
         log 0.01 [eV] to 10.  [keV] by 3
      end

      $ ALWAYS specify ALL boundaries
      temperature source boundary, sideset 1,
         function 1 scale 1.0 [kev] , 1.0
      vacuum boundary, sideset 2
      reflective boundary, sideset 3
      reflective boundary, sideset 4

      $ Some defaults
      flux limiter = larsen 2
      maximum diffusion coefficient factor = 100.0
      maximum energy density change = 0.2
      minimum energy density factor = 1.0e-14

   end

   $ Functions must be outside transport method
   function 1
      $seconds keV
      0.0 0.0
      3.0e-12 1.0
   end
end
```

Code Sample 3. Sample usage of the linearized diffusion method. Always specify all boundary conditions. Default values for some parameters are explicitly specified. Note that all radiation specific keywords are inside the `linearized diffusion` block.

4 Implicit Monte Carlo

The implicit Monte Carlo (IMC) package in ALEGRA is from LLNL's KULL code[4]. Create 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 as well as the method specific keywords listed below.

Currently IMC only works on Eulerian Hex8 and Quad4 meshes in Cartesian geometry.

4.1 Algorithm Control

All the keywords of the general transport method are available to set boundary conditions, initial conditions and sources. Several extra keywords are available to control this method as well.

4.1.1 Time Step Control

The calculation of the time step size for the next step is governed by the keyword

MAXIMUM MATERIAL TEMPERATURE CHANGE = real

At each element e, the averaged material temperature is calculated using

$$T_e = \frac{\sum_{m=1}^{M} \alpha_m T_m}{\sum_{m=1}^{M} \alpha_m}, \tag{40}$$

where α_m is the material volume fraction, and T_m is the material temperature. Only real materials are averaged; the sum does not include the void "material." A relative change in material temperature is calculated in each element using

$$\eta_e = \frac{|T_e^n - T_e^{n-1}|}{T_e^n + T_{\min}}, \tag{41}$$

The factor $T_g^{\min}$ ensures that a divide by zero will not occur, and also prohibits unimportant (cold) regions of the problem from influencing the time step and is set with

MINIMUM TEMPERATURE = 200.0 [K]

The default is $T_{\min} = 200$ K. The maximum η_e is found in the mesh and the new time step is computed using

$$\Delta t^{n+1} = \Delta t^n \frac{\eta_{\text{target}}}{\eta_{\max}}. \tag{42}$$

The default value is $\eta_{\text{target}} = 5.0$ and can be set using

MAXIMUM TEMPERATURE CHANGE = 5.0

The value of Δt^{n+1} is stored in the global variable *DT_RAD*.

4.1.2 Photon Population Control

The total number of photons on all processors used by the IMC package can be set with

NUMBER OF PHOTONS = int

where *int* is some positive integer. Alternatively, you may specify the average number of photons per element using

PHOTONS PER ELEMENT = int

where *int* is some positive integer. Using this keyword has the effect of multiplying the specified photons per element by the total number of elements in the entire mesh, and setting *NUMBER OF PHOTONS* to that number. Either *NUMBER OF PHOTONS* or *PHOTONS PER ELEMENT* must be specified.

The initial seed for the simulation can be set with

GLOBAL SEED = int

The *GLOBAL SEED* can be any integer. Changing this value will cause the simulation to evolve in a different way. By running multiple simulations with different seeds, it is possible to estimate the statistical error of the Monte Carlo simulation. (The distribution of any quantity in the simulation should be a Gaussian, and the width of the Gaussian is the error estimate.)

4.1.3 Planar Faces

Currently, the faces of a three dimensional mesh must be planar. This is one reason why only Eulerian meshes are supported. Meshes are checked for planar faces by first assuming the four nodes of a face are a tetrahedron. The volume of this tetrahedron should be zero if the four points are coplanar. The tetrahedron volume, which is normalized by the volume of the element, must be smaller than some threshold value, or

$$\frac{V_{\text{face tetrahedron}}}{V_{\text{element}}} < f \tag{43}$$

where f is the threshold set by

PLANAR FACE TOLERANCE = real

with a default value of 10^{-12}. This calculation can be susceptible to round off error. If a face fails the check, diagnostics are dumped to the screen, included the locations of the nodes on the face. These can be used to find the problematic face to see if it is really non-planar.

4.2 Output Variables

Several variables are available for output.

RAD_ENERGY_DENSITY is an element centered radiation energy density. It has units of energy per volume. The energy density is an array with the length the same as the number of energy groups.

This is only stored if requested for output.

RAD_TEMPERATURE is an element centered value of the radiation temperature in Kelvin. When the radiation is in equilibrium with the material, this should be be the same as the material temperature. The radiation temperature in an element is related to the energy density through

$$T_r = \left[\frac{1}{a} \sum_g E_g \right]^{1/4}, \tag{44}$$

where a is the black body constant.

This is only stored if requested for output.

RAD_ETA is the element value of η in Eq. 41 used for time step control. This is useful to find out what parts of the problem are controlling the time

```
radiation hydrodynamics

   hydrodynamics
      $ Some useful commands
   end

   $ radiation $ This is not needed
   linearized diffusion

      $ ALWAYS specify ALL boundaries
      vacuum boundary, sideset 2
      vacuum boundary, sideset 2
      reflective boundary, sideset 3
      reflective boundary, sideset 4

   end
   $ end $ This is not needed
end
```

Code Sample 4. A multi-physics input deck sample. Note that the interior radiation block is not necessary.

step. This has different meaning than the variables with the same name in other transport methods.

This is only stored if requested for output.

5 Multi-physics Packages

There are four multi-physics packages available with the radiation package. They are *RADIATION HYDRODYNAMICS*, *RADIATION HYDRODYNAMICS CONDUCTION*, *RADIATION MAGNETOHYDRODYNAMICS*, and *RADIATION MAGNETOHYDRODYNAMICS CONDUCTION*. The hydrodynamics portion is always calculated using the solid dynamics physics package in ALEGRA[2]. Keywords for the MHD and thermal conduction packages can be found in the ALEGRA HEDP manual[13].

Because the *RADIATION* block has no keywords other than the ones to select the transport method, the input deck in Code Sample 4 is perfectly valid.

References

[1] George I. Bell and Samuel Glasstone. *Nuclear Reactor Theory*. Krieger, 1970.

[2] S. K. Carroll, R. R. Drake, D. M. Hensinger, C. B. Luchini, S. V. Petney, J. Robbins, A. C. Robinson, R. M. Summers, T. E. Voth, , M. K. Wong T. A. Brunner, C. J. Garasi, T. A. Haill, and T. A. Mehlhorn. ALEGRA: Version 4.6. Technical Report SAND2004-6541, Sandia National Laboratories, Albuquerque, NM, December 2004.

[3] Chukai Yin and Bingjing Su. A nonlinear diffusion theory for particle transport in strong absorbers. *Annals of Nuclear Energy*, 29(12):1403–1419, August 2002.

[4] Nicholas A. Gentile, Noel Keen, and Jim Rathkopf. The KULL IMC package. Technical Report UCRL-JC-132743, Lawrence Livermore National Laboratory, Livermore, CA, 1998.

[5] Randall J. LeVeque, D. Mihalas, E. A. Dorfi, and E. Müller. *Computational Methods for Astrophysical Fluid Flow*. Springer, 1998.

[6] C. D. Levermore and G. C. Pomraning. A flux-limited diffusion theory. *The Astrophysical Journal*, 248:321–334, August 1981.

[7] Dimitri Mihalas and Barbara Weibel-Mihalas. *Foundations of Radiation Hydrodynamics*. Dover, 1999.

[8] Gerald N. Minerbo. Maximum entropy eddington factors. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 20:541, 1978.

[9] Gordon L. Olson, Lawrence H. Auer, and Michael L. Hall. Diffusion, P1, and other approximate forms of radiation transport. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 64(6):619–634, March 2000.

[10] G. C. Pomraning. *The Equations of Radiation Hydrodynamics*. Pergamon Press, 1973.

[11] Bingjing Su. Variable eddington factors and flux limiters in radiative transfer. *Nuclear Science and Engineering*, 137:281–297, March 2001.

[12] Thomas A. Brunner. Nodal Radiation Diffusion in ALEGRA. Working document to be published soon., 2004.

[13] Thomas A. Brunner, Christopher J. Garasi, Thomas A. Haill, Thomas A. Mehlhorn, Kyle R. Cochrane, Allen C. Robinson, and Randal M. Summers. ALEGRA-HEDP: Version 4.6. Technical Report SAND2004-5996, Sandia National Laboratories, Albuquerque, NM, December 2004.

Intentionally left blank

Index

DISTRIBUTION:

1 MS 1186
Thomas A. Brunner, 1674

1 MS 1186
Thomas A. Haill, 1674

1 MS 1186
Thomas A. Mehlhorn, 1674

3 MS 9018
Central Technical Files, 8945-1

2 MS 0899
Technical Library, 9616