Iterative Linear Solvers in a 2D Radiation-Hydrodynamics Code: Methods and Performance*

Chuck Baldwin Peter N. Brown Robert Falgout Jim Jones†
Frank Graziani‡

December 8, 1998

Abstract

Computer codes containing both hydrodynamics and radiation play a central role in simulating both astrophysical and inertial confinement fusion (ICF) phenomena. A crucial aspect of these codes is that they require an implicit solution of the radiation diffusion equations. We present in this paper the results of a comparison of five different linear solvers on a range of complex radiation and radiation-hydrodynamics problems. The linear solvers used are diagonally scaled conjugate gradient, GMRES with incomplete LU preconditioning, conjugate gradient with incomplete Cholesky preconditioning, multigrid, and multigrid-preconditioned conjugate gradient. These problems involve shock propagation, opacities varying over 5-6 orders of magnitude, tabular equations of state, and dynamic ALE (Arbitrary Lagrangian Eulerian) meshes. We perform a problem size scalability study by comparing linear solver performance over a wide range of problem sizes from 1000 to 100,000 zones. The fundamental question we address in this paper is: is it more efficient to invert the matrix in many inexpensive steps (like diagonally scaled conjugate gradient) or in fewer expensive steps (like multigrid)? In addition, what is the answer to this question as a function of problem size and is the answer problem dependent? We find that the diagonally scaled conjugate gradient method performs poorly with the growth of problem size, increasing in both iteration count and overall CPU time with the size of the problem and also increasing for larger time steps. For all problems considered, the multigrid algorithms scale almost perfectly (i.e. the iteration count is approximately independent of problem size and problem time step). For pure radiation flow problems (i.e. no hydrodynamics), we see speedups in CPU time of factors of ≈15-30 for the largest problems, when comparing the multigrid solvers relative to diagonal scaled conjugate gradient. For the incomplete

*This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

†Center for Applied Scientific Computing, L-561, Lawrence Livermore National Laboratory, Livermore, California 94550.

‡Low Energy Density Physics, L-170, Lawrence Livermore National Laboratory, Livermore, California 94550.

factorization preconditioners, we see a weak dependence of iteration count on problem size. The speedups observed for pure radiation flow are typically on the order of 10 relative to diagonal scaled conjugate gradient. For radiation hydrodynamics problems, we again see multigrid scaling perfectly. However, for the problems considered, we see speedups relative to diagonal scaled conjugate gradient of no more than ≈10, with incomplete Cholesky in fact either equaling or outperforming multigrid. We trace these observations to the time step control and the feature of ALE to relax distorted zones.

1 Introduction

Computer codes containing both hydrodynamics and radiation play a central role in simulating both astrophysical and inertial confinement fusion (ICF) phenomena [1-2]. With increasing experimental data coming from observational astronomy [3] and laser experiments [4], there is a need for performing spatially and temporally resolved numerical calculations of such physical processes as convective instabilities in a supernova or radiatively driven Richtmeyer-Meshov instabilities in an ICF capsule. These problems require accurately simulating not only fluid motion (including shock propagation) but also the transport of radiation energy density in both optically thick and thin materials in as computationally expedient a method as possible.

Typically, a multiphysics code also means multiple time scales, and radiation-hydrodynamics (RHD) codes are no exception. For problems of interest in this paper, namely radiation transport coupled to shock propagation, the hydrodynamic time scale is determined by the speed of sound and a "zone width," and the radiation time scale is determined by $1/(\kappa\rho c)$, where κ is the frequency dependent opacity, ρ is the material density, and c is the speed of light. The need to know transient shock behavior implies that an explicit formulation of the hydrodynamics equations is required. An RHD code can certainly run at the smallest time step as determined by the radiation Courant condition

$$\frac{c\Delta t}{\kappa\rho\Delta x^2} < \frac{1}{2},$$

where Δt is the time step and Δx is the zone width, but this can be extremely inefficient causing the code to have to run millions of cycles in order to capture a radiatively driven implosion. The other option is to run at the largest time step possible within the limits of stability and accuracy. This means running at time steps much larger than the time step demanded by the radiation Courant condition. Therefore, it is imperative from a stability standpoint that the radiation be run implicitly. It is this fact that necessitates the use of matrix solvers.

Performing spatially resolved numerical calculations of an ICF implosion can require hundreds of thousands to millions of zones in 2D and millions to tens of millions of zones in 3D. In addition, astrophysical and ICF applications can give rise to a wide range of density and temperature scales coupled to complicated flows that imply a highly anisotropic distribution of opacities covering a wide range of values (typically five orders of magnitude across an interface). Consequently, the matrices that need to be inverted in a real world application of an RHD code are not only extremely large, but are difficult to invert due primarily to the wide range in values of the matrix entries. The purpose of this paper is to compare and contrast the performance of five linear iterative solvers over a wide range of RHD problems which cover a wide range of zone counts. Typically, we will consider small (≈1000 zones), medium (≈10,000 zones), and large (≈100,000 zones) problems. The results presented in this paper are for a DEC-Alpha computer. All calculations are run serially although comments are made concerning future work on multiple processor platforms.

The five linear solvers chosen are:

1. diagonally scaled preconditioned conjugate gradient (DSCG) [15],

2. generalized minimal residual method with incomplete LU thresholding preconditioning (ILUT+GMRES) [15],

3. conjugate gradient method with incomplete Cholesky thresholding preconditioning (ICT+CG) [5],

4. semicoarsening multigrid (SMG) method [16], and

5. semicoarsening multigrid preconditioned conjugate gradient method (SMG+CG) [16].

The accuracy and stability of the code is enforced via time step controls which cause the time step to evolve as a function of time in a complicated fashion. This implies that the linear solvers are presented each time step with a changing matrix that might or might not be diagonally dominant. Hence, there is an intimate connection between time step behavior and linear solver performance. We perform a problem size scalability study by comparing linear solver performance over a wide range of problem sizes. The fundamental question we address in this paper is: is it more efficient to invert the matrix in many inexpensive steps (like diagonally scaled conjugate gradient) or in fewer expensive steps (like multigrid)? In addition, what is the answer to this question as a function of problem size and is the answer problem dependent?

The focus of this paper is scalability of algorithms. In the current parlance, the word scalable usually refers to the number of processors. However, our definition is more general. A code is scalable if it can effectively use additional computational resources to solve larger problems. More precisely, the total work, storage, and communication per process should not depend on overall problem size. As such, a specific factor that contributes to iterative numerical scalability is the convergence rates of iterative linear solvers. We stress that linear solver convergence can be discussed independent of parallel computing and is often overlooked as a key scalability issue. The scalability problem of linear iterative solvers should be analyzed in a multidimensional space where one degree of freedom is the problem size and the other degree of freedom is the number of processors. The purpose of this paper is to present results of the first phase of a two phase project. That is solver performance as a function of problem size. In a subsequent publication, we will present results of solver performance as a function of the number of processors for a given problem size and extend the results to 3D.

The rest of the paper is organized as follows. In §2, we give an overview of the radiation-hydrodynamics equations and code that we use in our testing, and then briefly describe the underlying discretization methods in the code. In §3, we discuss the above linear iterative methods. In §4, we present the suite of test problems, and in §5 we present our numerical results. Finally, a summary discussion is presented in §6.

2 The Radiation-Hydrodynamics Code

2.1 Physics

2.1.1 Hydrodynamics

We assume a nonrelativistic formulation of hydrodynamics whose governing equations are given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \tag{2.1}$$

$$\frac{\partial}{\partial t}\left(\rho \vec{u} + \vec{P}_r\right) + \nabla \cdot (\rho \vec{u}\vec{u} + \Phi_r) + \nabla p = 0 \tag{2.2}$$

$$\frac{\partial}{\partial t}\left(\frac{1}{2}\rho u^2 + E_m + E_r\right) + \nabla \cdot \left[\left(\frac{1}{2}\rho u^2 + E_m + p\right)\vec{u} + \vec{F}_r\right] = 0 \tag{2.3}$$

In these equations,

$$\begin{aligned}
\rho &= \text{fluid density,} \\
\vec{u} &= \text{fluid velocity,} \\
p &= \text{fluid pressure,} \\
E_m &= \text{fluid internal energy density,} \\
\vec{P}_r &= \text{radiation momentum density,} \\
\Phi_r &= \text{radiation momentum flux tensor,} \\
E_r &= \text{radiation energy density, and} \\
\vec{F}_r &= \text{radiation energy flux.}
\end{aligned}$$

In general, $\vec{F}_r = c^2 \vec{P}_r$. The radiation quantities appearing in the fluid dynamics equations will be defined in the next section within the context of diffusion theory.

2.1.2 Radiation

The RHD code we use in this study models the radiation transport as multigroup diffusion with flux limiting [14, 13]. Multigroup diffusion is an isotropic approximation to the radiation transport equation. There is no assumption made concerning the distribution of photons in frequency, only that the radiation field is approximately isotropic in space. Causality is enforced via the Wilson flux limiter [14, 13]. In addition, we assume local thermodynamic equilibrium so that the emission function is simply proportional to the Planck function (Kirchoff's Law) [14, 13]. We also assume that the electrons and ions carry their own temperatures. Implicit in this assumption is that locally, the electrons and ions can be represented by Fermi-Dirac and Maxwell-Boltzmann distributions respectively. We allow for both electron and ion conduction by using the form due to Lee and More [11] which incorporates both degeneracy as well as partial ionization effects. For this study however, we have turned off electron and ion conduction. The electrons and ions are coupled together through the Brysk-Spitzer coupling [6] which describes the rate of energy transfer between Maxwellian distributions of particles (allowing for a partial degeneracy of the electron gas). The electrons themselves are coupled to the radiation field through the opacity which is corrected for stimulated emission. We use tabular forms for the opacity which come from the code XSNQ [12]. This code uses an average atom approximation to compute bound-bound, bound-free, and free-free contributions to the opacity coming from line absorption, photoionization, and inverse brehmstrahlung. We do not include the effects of Compton scattering. The governing equations are

$$\frac{1}{c}\frac{\partial \varepsilon_\nu}{\partial t} = \nabla \cdot \left(\frac{1}{3\sigma_\nu}\nabla \varepsilon_\nu\right) + \sigma_\nu \left(B_\nu(T_e) - \varepsilon_\nu\right) \tag{2.4}$$

$$\frac{\partial}{\partial t}\left(\rho C_{V_e} T_e\right) = \nabla \cdot \left[D_e \nabla T_e\right] + \rho C_{V_e} \Omega_{ie}(T_i - T_e) - \int_0^\infty \sigma_\nu \left(B_\nu(T_e) - \varepsilon_\nu\right) d\nu \tag{2.5}$$

$$\frac{\partial}{\partial t}\left(\rho C_{V_i} T_i\right) = \nabla \cdot \left[D_i \nabla T_i\right] - \rho C_{V_i} \Omega_{ie}(T_i - T_e) \tag{2.6}$$

In these equations

$$\begin{aligned}
\varepsilon_\nu &= \text{radiation spectrum,} \\
T_e &= \text{electron temperature,} \\
T_i &= \text{ion temperature,} \\
\sigma_\nu &= \kappa_\nu \rho = \text{absorption inverse mean free path corrected for stimulated emission,} \\
D_e &= \text{electron conduction coefficient,} \\
D_i &= \text{ion conduction coefficient,} \\
C_{V_e} &= \text{electron heat capacity,} \\
C_{V_i} &= \text{ion heat capacity,} \\
\rho &= \text{material density,} \\
\Omega_{ie} &= \text{Brysk-Spitzer electron ion coupling coefficient, and} \\
B_\nu(T_e) &= \frac{8\pi h \nu^3}{c^3}\left(e^{h\nu/kT} - 1\right)^{-1} = \text{Planck function.}
\end{aligned}$$

The material motion and radiation transport are coupled via $\vec{P}_r$, Φ_r, E_r and $\vec{F}_r$. In the diffusion approximation, the radiation energy density E_r is simply $\int_0^\infty \varepsilon_\nu d\nu$, which defines the radiation temperature since $E_r = aT_r^4$ (a is the radiation density constant). The radiation momentum flux tensor is diagonal and is proportional to one-third of the radiation energy density. The radiation energy flux is related to the spectrum via Fick's law and is a direct consequence of the near isotropy of the radiation. For specific details regarding radiation transport and the diffusion approximation, we reference the works of Pomraning [14] and Mihalas and Mihalas [13].

2.2 Numerical Solution Procedures

2.2.1 Hydrodynamics

The RHD system (2.1)–(2.6) is solved on an ALE (Arbitrary Lagrangian Eulerian) mesh [1]. ALE is a technique that makes use of the ability of Lagrangian methods to let zones track material motion and at the same time avoid the mesh entanglements that Lagrangian codes inevitably get into by allowing the mesh to relax once a zone becomes too distorted. This latter step is called remap. We refer the interested reader to papers in [1] for the details of ALE hydrodynamics. The zones that make up the mesh always remain quadrilateral in 2D or hexahedral in 3D. Hence, the mesh is structured and logically rectangular. In addition, unlike AMR (adaptive mesh refinement) the number of zones in the problem remains fixed in time. The solution scheme is predictor-corrector and is fully second order accurate in space and time. There is a monotonic artificial viscosity (Q) [2]. The hydrodynamics step is governed by an explicit time step control determined by the sound speed and the zone size. The hydrodynamics is operator split from the radiation and is solved for first in any given cycle. In any given cycle, the hydrodynamic and radiation steps are performed only once. This implies that the coupled RHD problem can only be first order accurate in time. Figures 27–30 show a typical time snapshot sequence of an ALE mesh created during a radiatively driven implosion.

The radiation equations (2.4)–(2.6) are solved on the hydrodynamic grid. It is this fact and the fact that ALE relaxes a mesh that becomes too distorted that implies that ALE has an effect on the matrix solvers themselves. We will comment on this observation in subsequent sections.

2.2.2 Radiation

The multigroup radiation equations are solved via the partial temperature method [2]. A linear continuous finite element representation based on triangular elements is used for the "div-grad" operator [8, 18]. In 2D this means that we have a nine point stencil while in 3D the stencil is twenty seven. Integrating the "div-grad" over the volume of the j-th zone yields

$$\begin{aligned}\int_{j\text{-th zone}} \left(\nabla\cdot\left(\frac{1}{3\sigma_\nu}\nabla\varepsilon_\nu\right)\right) dV \;=\; & -a_{1,j+1}\left[\varepsilon_{\nu,j+1}-\varepsilon_{\nu,j}\right]+a_{1,j}\left[\varepsilon_{\nu,j}-\varepsilon_{\nu,j-1}\right] \\ & -b_{0,j+s}\left[\varepsilon_{\nu,j+s}-\varepsilon_{\nu,j}\right]+b_{0,j}\left[\varepsilon_{\nu,j}-\varepsilon_{\nu,j-s}\right] \\ & -b_{1,j+s+1}\left[\varepsilon_{\nu,j+s+1}-\varepsilon_{\nu,j}\right]+b_{1,j}\left[\varepsilon_{\nu,j}-\varepsilon_{\nu,j-s-1}\right] \\ & +bm_{1,j+1}\left[\varepsilon_{\nu,j-s+1}-\varepsilon_{\nu,j}\right]+bm_{1,j+s}\left[\varepsilon_{\nu,j}-\varepsilon_{\nu,j+s-1}\right].\end{aligned} \tag{2.7}$$

The terms in equation (2.7) represent eight fluxes: four for the face fluxes (proportional to a_1 and b_0) and four for the fluxes at the corners (proportional to b_1 and bm_1). These latter fluxes go to zero in the limit of an orthogonal mesh. In addition, the spectrum ε_ν, is evaluated at the new time step. Solving the radiation equations on an ALE mesh which is logically rectangular but nonorthogonal in terms physical space yields a matrix which has a simple striped structure. In Figures 1 and 2, we show the generic form of the matrix generated by the radiation equation on an ALE mesh and the corresponding zonal couplings of equation (2.7) generated by the finite element representation of the "div-grad" operator. Flux conservation implies that the matrix is symmetric. In addition, the matrix is also positive definite. Besides containing incoming and outgoing flux information, the main diagonal contains information regarding local coupling physics and the old time step radiation spectrum. These terms are order $1 + O(\Delta t)$. Couplings to neighboring zones are represented in the matrix by the off-diagonal terms and come from incoming and outgoing fluxes. These terms are order $O(\Delta t)$. For an orthogonal mesh, the matrix structure simplifies considerably with the matrix consisting of nonzero entries on the main diagonal, directly above and below the main diagonal and nonzero entries one stride length away also forming diagonals. Once the mesh becomes nonorthogonal, nonzero entries start appearing above and below the diagonals that are one stride length away from the main diagonal. Running pure radiation problems implies for our code a fixed mesh. However, running hydrodynamics implies ALE and the subsequent relaxation of distorted zones. This implies for radiation that the corner couplings which appear above and below the diagonals that appear one stride length away from the main diagonal, can be relatively small compared to diagonal entries.

The equations (2.4)–(2.6) are solved implicitly with the understanding that the opacity, conduction coefficients, heat capacity, and electron-ion coupling are evaluated at the start of the radiation step and hence contain only updated information coming from the hydrodynamics step. In addition, the Planck function is linearized about the old time stamp value for the electron temperature. The full set of coupled radiation, electron, and ion equations are operator split from each other and therefore solved in several steps. These are outlined as follows:

1. Solve the radiation diffusion, 1/3 of the electron-ion coupling, and all of the radiation-electron coupling.

2. Solve another 1/3 of the electron-ion coupling.

3. Solve the electron conduction and the remaining 1/3 of electron-ion coupling.

4. Solve the ion conduction.

2.2.3 Time Step Controls

The stability of the hydrodynamics demands a Courant condition based on the sound speed and zone size. Accuracy on the other hand restricts the time step for the radiation. Since the radiation solver step is split from the hydrodynamic solution step, the solution to a coupled RHD problem is only first order accurate in time. In addition, the Planck function is linearized about the old electron temperature, the various couplings between the photon, electron, and ion fields are operator split, and quantities such as the specific heat use old time stamp values all leading to inaccuracies if the time step is too large. For these reasons the code restricts the fractional change in the radiation temperature in any given zone to be less than 20%. By running analytic test problems, we have found that this restriction yields reasonable accuracy. The splitting of the hydrodynamic and radiation steps also means that it is important to limit the impact of a particular physics package on any other package. For this reason we introduce a limit on the radiation acceleration. Radiation acceleration arises

$$\begin{bmatrix} D & x & 0 & 0 & \dots & x & x & 0 & \dots & \dots \\ x & D & x & 0 & \dots & x & x & x & 0 & \dots \\ 0 & x & D & x & 0 & \dots & x & x & x & 0 \\ 0 & 0 & x & D & x & 0 & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ x & x & 0 & \dots & x & D & x & 0 & \dots & \dots \\ x & x & x & 0 & \dots & x & D & x & \dots & \dots \\ 0 & x & x & x & 0 & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

Figure 1: Generic matrix nonzero pattern

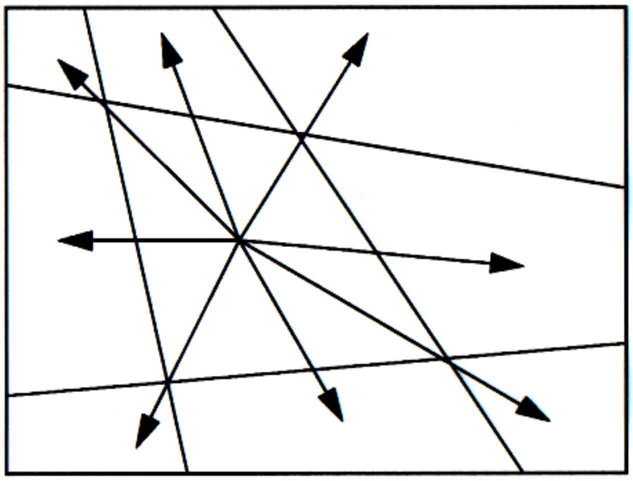

Figure 2: Generic zonal couplings

due to the transfer of momentum from the photons to the fluid which in turn induces a force or acceleration on the fluid. In diffusion, this acceleration is essentially just the gradient of the radiation pressure. Hence the time step

$$\Delta t \approx \sqrt{\frac{\rho}{\text{radiation pressure}}} \times \text{zone size}$$

defines a causality condition whereby the signal produced by the radiation pressure accelerating the fluid cannot exceed an effective sound speed given by $\sqrt{\frac{\text{radiation pressure}}{\rho}}$.

At this point we have several time scale restrictions governing the accuracy and stability of the code. A single time step governs both the radiation and hydrodynamic packages and it evolves dynamically from some initial value. The choice of the initial value is problem dependent and depends on the material properties and geometry of the system under study. A general rule of thumb is that for problems involving optically thick materials an initial time step of $\Delta t = 10^{-4}\mu$sec works well, while for optically thin materials an initial time step of $\Delta t = 10^{-8}\mu$sec is needed. A point worth mentioning is that the RHD code used in this paper is postdictive, in the sense that if any one of a series of time step restrictions is violated the code only decreases the time step on the following cycle and not the current cycle. Consequently, if any given time step satisfies the controls listed, then the time step is allowed to increase by a factor of 1.2 in the next cycle. If however, the time step violates any of the restrictions (i.e Courant, $\Delta T_r/T_r \leq .2$, or radiation acceleration) the code operates on the next cycle at a time step dictated by the largest of the three controls.

The time step control, as we will see has a bearing on the efficiency at which a particular solver can invert the matrix. Hence, there is an intimate connection between solver convergence rates and evolving time steps values.

3 Iterative Linear Solvers with Preconditioning

3.1 Diagonally Scaled Conjugate Gradient

The first preconditioned iterative method we consider is the diagonally scaled conjugate gradient method (DSCG). By this we mean the preconditioner for the conjugate gradient method is simply the inverse of the main diagonal of the matrix. This has the virtue of ease of coding, and for the time-dependent simulations of interest works quite well when the timestep size is very small. This follows since the matrix A has the form $A = I - \Delta t J$, where J is the discretized spatial differential operator. For very small Δt, A is diagonally dominant, and so the inverse of its main diagonal should be a good approximate inverse. However, the performance of DSCG is highly dependent on the problem size, degrading quickly as the size increases.

3.2 Incomplete Factorizations as Preconditioners

The use of incomplete factorizations as a technique for generating preconditioners has been extensive in the literature. We refer the reader to the excellent book by Saad [15] for a comprehensive development of these techniques. The major advantage of using incomplete factorizations as preconditioners is their easy application to a variety of problems. These techniques only require a matrix, no specific knowledge of the problem under consideration is required, unlike the case with structured multigrid methods. However, the standard approaches to using these methods typically do not scale well with problem size, and our numerical results below demonstrate this. For comparison purposes, we consider two incomplete factorization methods, both developed by Saad and his coworkers. Specifically, we use ILUT (Saad,[15]) and a modified version of ILUT designed for symmetric positive definite problems, called ICT. We only briefly describe these techniques, and refer the reader to the above references for more detail on ILUT, and a recent report describing ICT [5].

3.2.1 ILUT

The ILUT algorithm was conceived as a combination of two earlier techniques, one being a *level-of-fill* concept and the other a *threshold dropping tolerance*. Both techniques were effective by themselves on certain classes of problems, but were plagued with inherent difficulties. The level-of-fill concept did not take into account actual numerical values in the matrix, and hence could perform poorly on some problems where fill-in was important, while with the drop-tolerance approach it was difficult to estimate the needed storage and work to accomplish the factorization. ILUT was the first incomplete *LU* algorithm to successfully combine the two approaches. The following basic approach is as follows:

Generic Incomplete LU Factorization with Thresholding, `ILUT(lfil,droptol)`:

```
0    row(1:n) = 0, U(1,1:n)=A(1,1:n)
1    do i = 2,n
2      row(1:n) = A(i,1:n)
3      do k = 1,i-1 (and where row(k) .ne. 0)
4         row(k) := row(k) / U(k,k)
5         apply a dropping rule to row(k)
6         if (row(k) .ne. 0) then
7           row(k+1:n)=row(k+1:n)-row(k)*U(k,k+1:n)
8         endif
9         apply a dropping rule to row(1:n)
10     enddo
11     L(i,1:i-1) = row(1:i-1)
12     U(i,i:n) = row(i:n)
13     row(1:n) = 0
14   enddo
```

The dropping rules in lines 5 and 9 are based on an input relative tolerance `droptol` and a sparsity dropping tolerance `lfil`. In `ILUT(lfil,droptol)`, the following rules are used:

- In line 5, if $|\mathrm{row}(k)| \leq \tau_i \equiv$ `droptol` $\cdot$ 2-norm of row i, then $\mathrm{row}(k) := 0$.

- In line 9, first any element in the row with magnitude less than τ_i is dropped. Then only the `lfil` largest elements in the L part of the row and the `lfil` largest elements in the U part of the row are kept, plus the diagonal element.

The second step controls the number of elements per row. Note that no pivoting is performed. An ILUTP variant performs pivoting. The advantages of ILUT over earlier ILU techniques are two-fold:

- Taking `droptol` $= 0$ and `lfil` $= n$ gives an exact sparse LU factorization with no pivoting. Thus, the user can control the quality of the preconditioner.

- The user can determine how much storage is needed beforehand.

Since ILUT is formulated for nonsymmetric matrices, we use ILUT as a preconditioner for the Generalized Minimal Residual (GMRES) linear iterative method.

3.2.2 ICT

For symmetric positive definite (SPD) problems, ILUT is too costly, both in terms of the storage and computational work involved. Generally, the Cholesky factorization for SPD matrices is used as the basis for generating a preconditioner based on incomplete factorization. The ICT algorithm is based on an LDL^T decomposition of an SPD matrix A. Briefly, consider the sequence of matrices

$$A_{k+1} = \begin{pmatrix} A_k & w_{k+1} \\ w_{k+1}^T & \alpha_{k+1} \end{pmatrix},$$

where $A_n = A$. If A_k is nonsingular and its LDL^T factorization

$$A_k = L_k D_k L_k^T$$

is already available, then the LDL^T factorization of A_{k+1} is

$$A_{k+1} = \begin{pmatrix} L_k & 0 \\ y_{k+1}^T & 1 \end{pmatrix} \begin{pmatrix} D_k & 0 \\ 0 & d_{k+1} \end{pmatrix} \begin{pmatrix} L_k^T & y_{k+1} \\ 0 & 1 \end{pmatrix}$$

in which

$$\begin{aligned} y_{k+1} &= D_k^{-1} L_k^{-1} w_{k+1} \\ d_{k+1} &= \alpha_{k+1} - y_{k+1}^T D_k y_{k+1}. \end{aligned}$$

Hence, the last row and column of the factorization can be obtained by solving one unit lower triangular system and computing a scaled dot product. The ICT incomplete factorization based on this factorization is given as follows:

Generic Incomplete LDL^T Factorization with Thresholding, `ICT(lfil,droptol)`:

```
D(1,1) = A(1,1), L(1,1) = 1;
do k = 2,n
  row = A(k,1:k-1)
  do i = 1,k-1 (and where row(i) .ne. 0)
     if ( abs(row(i)/D(i,i)) .le. droptol ) row(i) = 0
     if (row(i) .ne. 0) then
       do j = i+1,k-1
         row(j) = row(j) - row(i)*L(j,i)
       enddo
       row(i) = row(i) / D(i,i)
     endif
  enddo
  drop all but the lfil largest elements in row
  D(k,k) = A(k,k) - row*D(1:k-1,1:k-1)*row'
  L(k,1:k-1) = row
  L(k,k) = 1
enddo
```

(Note that in the algorithm, `row` represents y_{k+1}^T.) If A is SPD, then the diagonal matrix D has all positive entries in its LDL^T factorization. For the incomplete factorization, this may no longer be true, and likely signals a poor approximation to the original matrix. In the testing described below, we use the ICT algorithm as a preconditioner in a PCG (Preconditioned Conjugate Gradient) linear iteration.

3.2.3 Ordering Strategies

When using incomplete factorization techniques to generate preconditioners, the ordering of the rows and columns of the matrix can have a dramatic effect on the amount of fill-in that occurs. For direct solvers, some version of the minimum degree algorithm is a good generic choice for a reordering strategy as it produces the least amount of fill-in. However, a reordering strategy that generates the most effective preconditioner based on an incomplete factorization typically is not the one with the least fill-in. Some strategy based on minimizing the bandwidth, such as the reverse Cuthill-McKee (RCM) reordering strategy, often generates more effective preconditioners. This fact is not well understood, and has been the subject of much research [15]. We use the RCM reordering algorithm in all of the problems considered below, as it is crucial for good performance on the larger test problems.

3.3 Multigrid solvers and preconditioners

Multigrid methods can be very efficient solvers for the linear systems arising from discretized elliptic partial differential equations. Multigrid's chief advantage is that it is a scalable algorithm in that, when properly designed, the solver's convergence rate is independent of the size of the discretized system. Standard multigrid methods combine simple relaxation (which quickly reduces high-frequency error components) with error correction from a coarser grid (which can accurately represent low-frequency error components). For our problem, the multigrid solver must be able to efficiently deal with anisotropies and widely variable coefficients. The semi-coarsening algorithm used is based on the work by Schaffer [16] (see also [7] and [17]), and we will briefly discuss this particular multigrid algorithm. We focus on the 2D algorithm (commenting on the 3D extension) and on those features that differentiate it from standard multigrid methods. For more general multigrid references, see [3], [4], [9], [19]. For current information on the multigrid field, including an extensive bibliography, a repository of papers and codes, and current events, access the World Wide Web server MGNet at http://casper.cs.yale.edu/mgnet/www/mgnet.html.

3.3.1 SMG: semi-coarsening multigrid

Let $AU = F$ be the given linear system to solve, where the unknown U and right-hand side F are vectors defined on a logically rectangular grid. We will use an h superscript to denote quantities defined on the given grid. The matrix A is symmetric, positive definite and connections have the standard "nearest-neighbor" 9-point stencil form. The multigrid algorithm of Schaffer uses a combination of semi-coarsening, line-relaxation, and operator-based interpolation. The resulting algorithm is efficient and robust with respect to anisotropic and widely variable coefficients in the matrix A.

As the grid is logically rectangular, there is a unique index (i, j) for each point on the grid, and the grid can be given a "red/black" line coloring. All unknowns $\{(i, j), j \text{ odd}\}$ are considered "red" and will be used for the coarse grid. We will use a $2h$ superscript to denote quantities defined on the coarse grid. This is called semi-coarsening (as opposed to full or standard coarsening) as the coarse grid is only coarser in one of the dimensions. Red/black line relaxation involves updating the solution at all red lines to satisfy their equations (a tridiagonal solve for each red line) followed by a similar update for the black lines. Because of the 9-point stencil, there is no dependence between lines of the same color and they could be updated in parallel.

An important, unique feature of the SMG algorithm is the definition of the interpolation operator I_{2h}^h used to transfer an error correction from the coarse to the fine grid. The definition is motivated by the relationship between error on red and black lines after a black line relaxation sweep. To briefly describe the approach, let

$$A_{J,J-1}U_{J-1} + A_{J,J}U_J + A_{J,J+1}U_{J+1} = F_J \tag{3.8}$$

be the equations for the J^{th} line. Here $U_J = (U_{i,J}, i = 1, \ldots, n_x)$ and similarly for $U_{J\pm1}$. After relaxing this line, the error equation is

$$A_{J,J-1}e_{J-1} + A_{J,J}e_J + A_{J,J+1}e_{J+1} = 0, \tag{3.9}$$

so

$$e_J = -A_{J,J}^{-1}A_{J,J-1}e_{J-1} - A_{J,J}^{-1}A_{J,J+1}e_{J+1}. \tag{3.10}$$

After black line relaxation this relationship describes how the error at black lines is related to the error at red (coarse) lines; it gives the "ideal" interpolation formula. However, using equation 3.10 leads to non sparse interpolation operators. In the SMG algorithm, sparse approximations to these ideal interpolation operators are used. The matrices $-A_{J,J}^{-1}A_{J,J-1}$

and $-A_{J,J}^{-1}A_{J,J+1}$ are approximated by diagonal matrices with the same action on constant vectors. The computation of these interpolation operators involves a tridiagonal solve for each black grid line.

With this definition for the interpolation operator I_{2h}^{h}, its transpose is used for the restriction operator I_{h}^{2h} (used in transferring residuals from the fine to the coarse grid), and the coarse grid versions of A are defined by the Galerkin condition, i.e. $A^{2h} = I_h^{2h} A^h I_{2h}^h$. These components are then used in a standard multigrid V-cycle as outlined below.

$V(\nu_1, \nu_2)$-cycle

1. Pre-relaxation on $A^h U^h = F^h$. Perform ν_1 sweeps of red/black line relaxation.
2. Set $F^{2h} = I_h^{2h}(F^h - A^h U^h)$.
3. "Solve" $A^{2h}U^{2h} = F^{2h}$ by recursion.
4. Correct $U^h \leftarrow U^h + I_{2h}^h U^{2h}$.
5. Post-relaxation on $A^h U^h = F^h$. Perform ν_2 sweeps of black/red line relaxation.

The equation to be solved in step 3 has the same form as the original grid h problem. It is solved by applying the same algorithm using a still coarser grid $4h$. Eventually, a coarse grid is reached that has a single grid line and line relaxation is a direct solver.

3.3.2 SMG+CG: multigrid as a preconditioner

As will be shown in the numerical results, the SMG algorithm alone can be an efficient solver for our linear systems. However, using it as a preconditioner in a PCG (Preconditioned Conjugate Gradient) iteration is generally a more robust strategy and, depending on the problem, can be more efficient as well. In the preconditioning step of PCG we apply a single V-cycle of SMG, and the V-cycle must be constructed so as to yield a symmetric preconditioner. The reference [10] provides conditions that guarantee symmetry of a multigrid V-cycle, and the SMG algorithm meets these provided that the number of pre-relaxations, ν_1 is equal to the number of post-relaxations, ν_2.

In all our numerical tests, the SMG runs used a $V(1,0)$-cycle and the SMG+CG runs used a $V(1,1)$-cycle as the preconditioner. The $V(1,0)$-cycle is generally the most efficient stand-alone solver, so the SMG+CG carries the computational overhead of the PCG algorithm plus the additional relaxation sweeps needed to guarantee symmetry. In comparing the SMG+CG runs to the runs using SMG alone, SMG+CG runs have greater computational work per iteration, but require fewer iterations.

4 The Multi-Physics Test Suite

The purpose of the muti-physics test suite is to present to the code and in particular the linear solvers, a wide spectrum of problems. In this way, an accurate and fair assessment of the speed of the solvers can be made. The first part of the test suite covers radiation flow alone without the effects of hydrodynamic motion. The second part tests radiation flow in the presence of material motion. Two test problems that fit into the pure radiation flow category are (1) radiation flow on a highly distorted mesh in 2D (Kershaw mesh problem, see Figure 3), and (2) radiation flow in a spherical geometry (see Figure 11). In both cases, the lack of material motion means that the mesh is fixed in time. The last two problems run in the suite test the RHD capabilities of the code. More importantly, from the standpoint of this paper, it tests the linear solvers on a mesh that is changing with time. Because of the complicated mesh pattern that arises from shocks and ordinary material motion, the time step is a complicated function of time. The test suite is intended to test the ability of the linear solvers to solve the radiation equations on a dynamically changing mesh with a time step control determined by both hydrodynamic and radiative processes. The net result as far as the linear solvers are concerned is that the matrix itself is changing both in the values of its elements and in its structure. The solver performance tests presented in this paper are therefore more severe and realistic than are test matrices which are typically used to judge solver performance. We do not wish to imply that our test suite is exhaustive, only that it presents for what we believe the first time, a realistic assessment of linear solver performance. Figures 3, 11, 19 and 27 show the initial geometry and mesh for each of the problems in the test suite.

All problems presented here were run without electron or ion conduction. In addition, the number of frequency groups was taken to be one. In this way, the statistics presented in this paper are for only one matrix solve per cycle. The advantage of doing this is that the performance figures for radiation diffusion are not confused with those of material conduction. Although calculations without conduction or with one frequency group may not give the best representation of reality, this is not the purpose of this paper. We are primarily interested in linear solver performance in complex RHD flows. The importance however, of the linear solver timing results presented here become magnified when a full multigroup calculation involving say 50 groups is performed. For example, the results presented for CPU time spent in the radiation package would be multiplied by approximately 50 times thereby dominating other physics packages such as hydrodynamics. This means, that a slow inefficient solver becomes a tremendous sink of time in any multiphysics code since the solver has to perform the inversion for each frequency group. This fact should be kept in mind.

4.1 Kershaw Problem

The problem consists of a slab of CH foam heated at one end with a constant temperature source of 300 eV. The slab measures 4.0 cm in the vertical direction and 4.5 cm in the horizontal direction. The CH foam is at a constant density of 1.05 gm/cc. The mesh is shown in Figure 3. The small problems consists of 40×50 zones, the medium 80×100 zones, and the large 320×400 zones. The boundary conditions are reflecting at the right, top, and bottom boundaries and open at the left boundary. The problem describes Marshak wave propagation. The problem is run to 10^{-3} microseconds.

4.2 Spherical Diffusion without Hydrodynamics

This problem consists of a ball of DT ice at a density of .25 gm/cc at a radius of .04 cm, a shell of CH foam at density 1.05 gm/cc and a radius of .07 cm, a shell of sourced He at a density of .0005 gm/cc, temperature of 300 eV and a radius of .24 cm, and finally a shell of Au at a density of 19.3 gm/cc and a radius of .3 cm (see Figure 11). This problem represents an ICF capsule where the DT ice is the fuel, the CH foam is the ablator, and the He the gas inside the Au hohlraum. The source temperature is set at 300 eV. This problem is run without material motion and hence merely tests the diffusion of radiation on a fixed polar geometry. This problem was run with 1000 zones (10 angular$\times$100 radial), 10000 (10 angular$\times$1000 radial), and 100000 zones (100 angular$\times$1000 radial). The problem is taken to be rotationally symmetric about the x axis and is run to 10^{-3} microseconds.

4.3 Radiatively Driven Symmetric Implosion

This problem is identical in principle to the previous problem with hydrodynamics turned on (see Figure 19). This problem describes the ablation of the CH foam followed by the subsequent implosion of the DT ice capsule. The DT capsule continues to implode with corresponding increases in temperature and density until the shocks converge on the center and bounce, whereupon the DT capsule explodes. This problem was run on an initially uniform rectilinear mesh in the small (30$\times$30 zones), medium (100$\times$100 zones), and large (300$\times$300 zones) categories.

4.4 Radiatively Driven Asymmetric Implosion

This is a problem identical to the radiatively driven symmetric implosion except the DT ice capsule has been shimmed so that it is an ellipse (since this problem is also rotationally symmetric about the x axis the capsule really is an ellipsoid). The major axis is .06 cm while the minor axis is .04 cm (see Figure 27). In this problem, the CH foam is heated and ablates in an asymmetrical fashion. This causes an asymmetrical implosion of the capsule. This problem was run with 1,000 zones (10 angular $\times$ 100 radial), 10,000 (100 angular $\times$ 100 radial), and 100,000 zones (100 angular $\times$ 1,000 radial). This problem tests the linear solver performance on a mesh, though initially symmetric, becomes skewed in time due to off center convergence of incoming shocks.

5 Results

The Kershaw calculations were performed on a DEC-Alpha with a 300Mhz Alpha chip with 8 GB of main memory. The other problems were performed on a 625 Mhz DEC-Alpha chip with 8 GB of main memory. Before a given result was considered satisfactory, we had to make sure that a given problem run with a variety of linear solvers was giving identical answers. In order to do this we compared time dependent data at selected zones in a specific problem and also the time step as a function of time for each solver. This method proved useful in locating several bugs in the linear solvers. A given problem run on a variety of linear solvers was not considered acceptable unless all time dependent data for a given problem size agreed to one part in 10^6.

For all of the runs using ILUT and ICT, we used `droptol` = 0.0001 and `lfil` = 20. Additionally, in all of the figures showing iteration and timestep counts, the labels A, B, C, D, and E refer to the methods DS+CG, ILUT+GMRES, ICT+CG, MG and MG+CG, respectively.

5.1 Kershaw Problem

This problem was run to a time slightly past steady state. Figures 3, 4, 5 and 6 show the evolution of the radiation front from time zero to steady state. The transient profile shown in Figure 4 shows some mesh imprinting. The steady state profile however shows uniform contours; a benefit of the finite element representation of the "div-grad" operator. For the small, medium, and large Kershaw mesh problems, Figures 7, 8 and 9 show iteration count per cycle as a function of problem time in microseconds. In addition, in Figure 10 we show the time step in microseconds versus problem time, also in microseconds, for the medium size problem. (For all of the tests, we show only the time step history for the medium size problem, as this is representative of the other cases.) We have started the time step at 10^{-10} microseconds. This time has been made artificially small so that the DSCG algorithm solves the matrix in one iteration. As mentioned in §2.2.3, the code allows the time step to grow by a factor of 1.2 per cycle unless a given accuracy or stability criterion is violated. The nature of this particular problem (i.e. the mesh) and its corresponding time step controls imply that the matrix is diagonally dominant for a period of time beginning at $t = 10^{-10}$ microseconds. As the radiation front moves through the distorted mesh, zones are becoming more non-orthogonal thus giving rise to large corner couplings. This effect in the matrix means that off diagonal elements are being populated with non-trivial values. At the same time, as the radiation front is travelling through the CH foam, the time step is increasing. This also leads to off diagonal elements in the matrix becoming more important. The result of these two effects can be seen when comparing iteration count versus time for the various solvers. Although all the solvers scale with problem time, DSCG is by far the most sensitive. When we look at iteration count for the medium and large Kershaw problems, we observe a sensitivity (albeit weak) of both the ILUT+GMRES and ICT+CG solvers. By far, the MG and MG+CG solvers show the best scalability as evidenced by the fact that their iteration count as a function of time is almost independent of problem size.

Although iteration count is interesting, the bottom line is CPU time. Tables 1, 2 and 3 give the CPU times for the parts of the simulation that we are interested in : total code execution time, execution time for the radiation transport, and execution time for the linear solvers. For the small Kershaw problem the ILUT+GMRES and ICT+CG are competitive with the multigrid solvers (MG and MG+CG) each giving rise to a speedup of ≈3.5 compared to DSCG. As the problem size is increased to 8,000 and then 128,000 zones, we observe several interesting features of the solvers (see Table 4). The ILUT+GMRES seems to reach an asymptotic speedup value of ≈4 relative to DSCG. The ICT+CG solver is able to yield a speedup of a factor of ≈11 by the time we reach problem sizes on the Kershaw mesh of 128,000. But, by far, the biggest winner is MG and MG+CG. The medium Kershaw mesh shows a slight separation in speedups between ICT+CG and the MG solvers. However, the

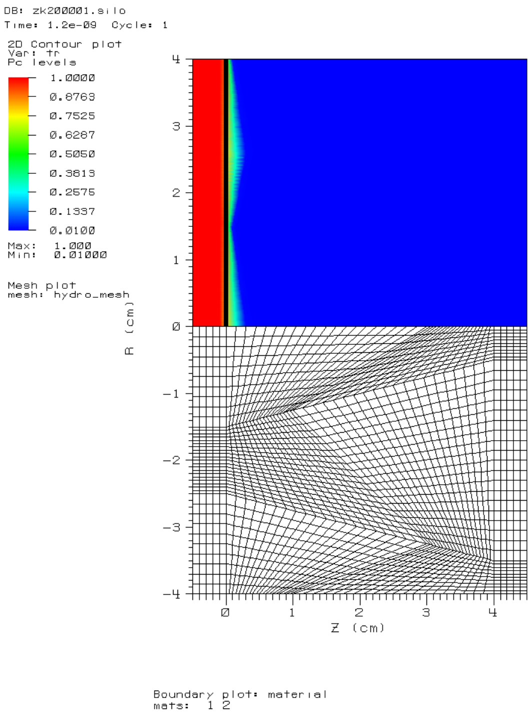

Figure 3: Initial geometry for Kershaw Problem

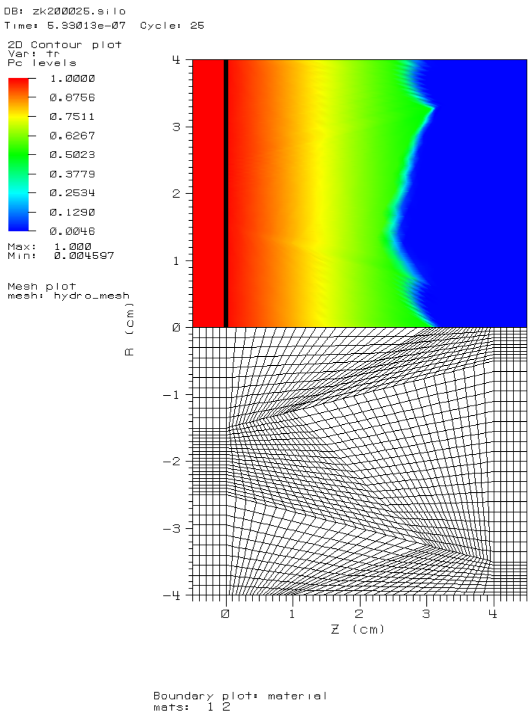

Figure 4: Snapshot 2

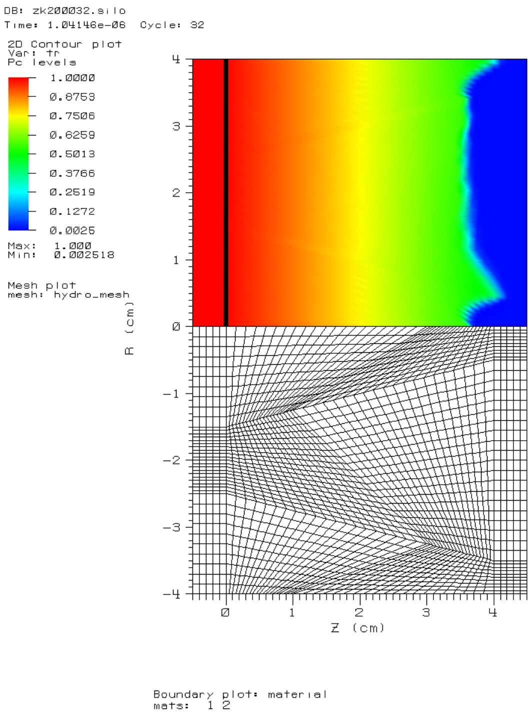

Figure 5: Snapshot 3

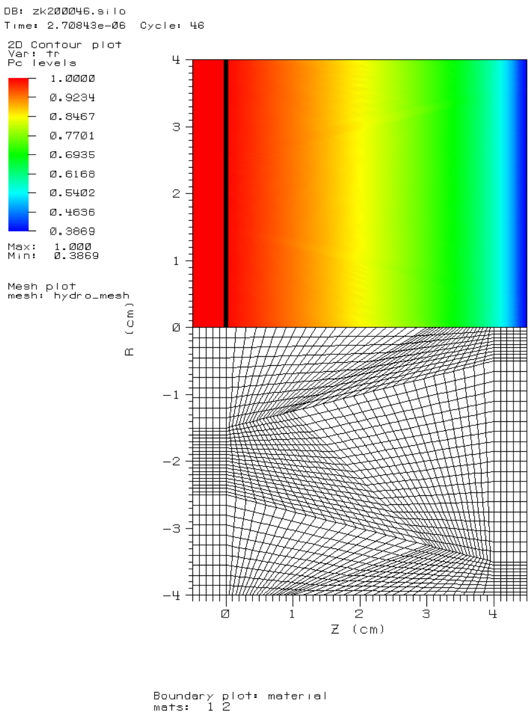

Figure 6: Snapshot 4

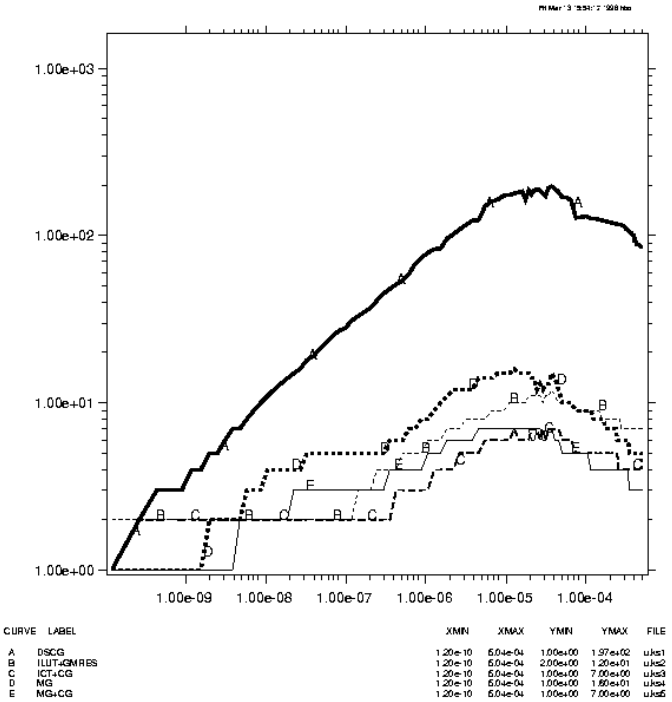

Figure 7: Iteration Counts for Small Kershaw Problem

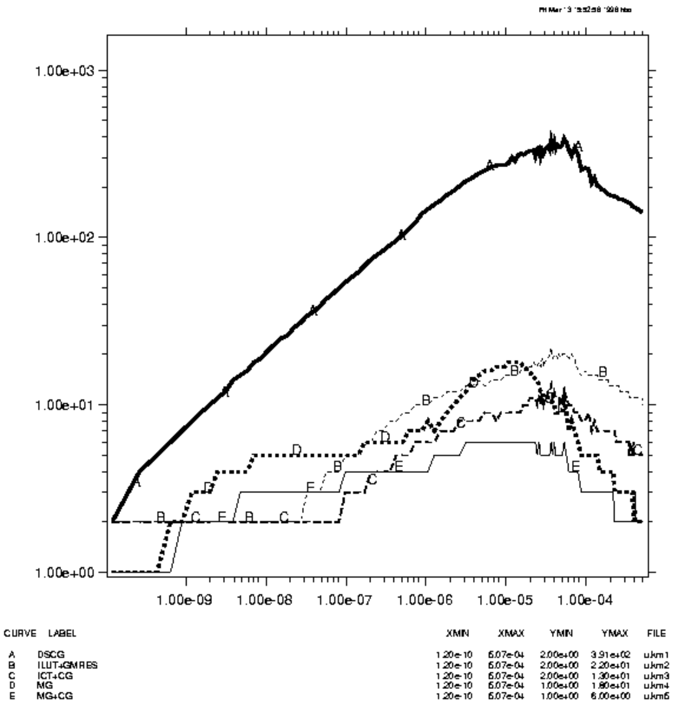

Figure 8: Iteration Counts for Medium Kershaw Problem

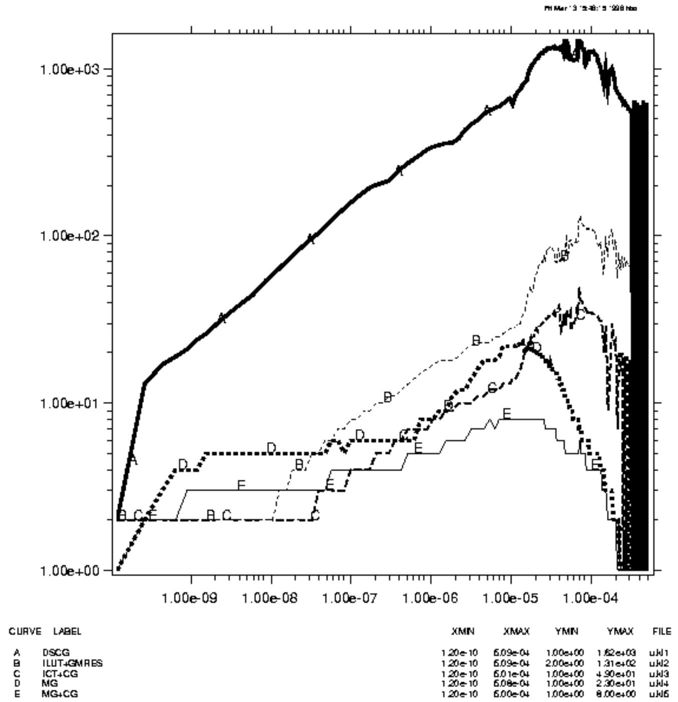

Figure 9: Iteration Counts for Large Kershaw Problem

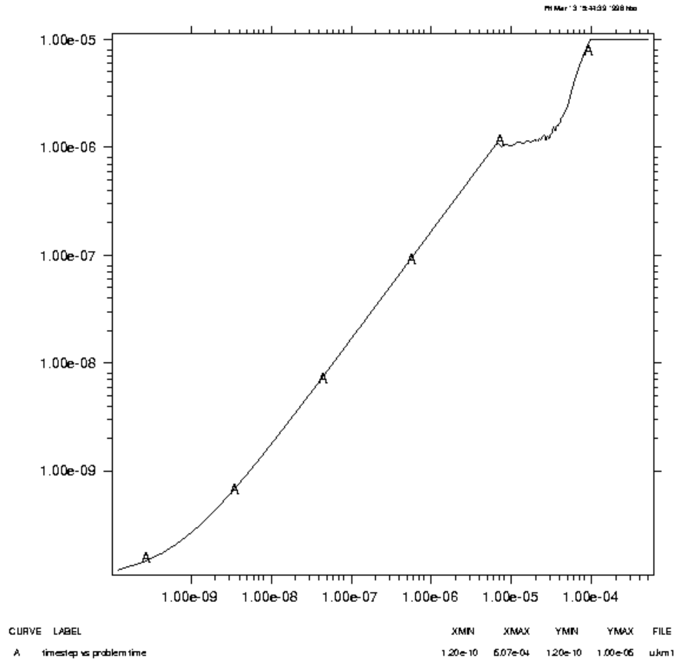

Figure 10: Time Steps for Medium Kershaw Problem

real strong separation occurs for the large problem where both MG solvers beat ICT+CG by a factor of 2. This problem is evidence of the fact that running a scalable algorithm like MG or MG+CG, although expensive per iteration, more than makes up for its overhead when large matrices (order ≈100,000 by 100,000) need to be inverted.

5.2 Spherical Diffusion without Hydrodynamics

In this problem, the mesh is again fixed in time. This problem tests the linear solver convergence rate on a mesh more typical of what appears in ICF calculations (at least initially before instabilities and shocks set in and destroying the symmetry of the mesh). Figures 11, 12, 13 and 14 show the evolution of the radiation temperature and mesh for the problem. In Figures 15, 16 and 17 we see iteration count as a function of time for the small, medium and large size meshes. Figure 18 shows the time step as a function of time. As mentioned previously, the time step started at 10^{-6} microseconds and was allowed to increase by the factor of 1.2 per cycle unless the fractional change in the radiation temperature exceeded .2. The time step increases, and if the accuracy criterion is violated, then the reduced time step is applied at the next cycle. We note some of the same features that were observed in the Kershaw mesh. First, the obvious scaling of DSCG iteration count with both time step and problem size. Although, the growth of iteration count with time step and problem size is not as severe as the Kershaw case. The reason for this is primarily due to the smooth, almost orthogonal nature of the mesh compared to the Kershaw problem which implies a matrix more sparsely populated in off diagonal entries. This explains for example why the DSCG iteration count for this problem barely gets past 100 for the medium size problem while for the Kershaw case, the iteration count was at several hundred. Tables 5, 6 and 7 show the CPU time spent in the whole code, radiation package, and linear solver. In addition, the relative speedups compared to DSCG are shown. The speedups for all solvers relative to DSCG is impressive even for the small problems with a maximum speedup of 3.58 for the MG solver (see Table 8). For the medium mesh results, again MG by itself is the clear winner (speedup factor of 10.54), closely followed by MG+CG (speedup factor of 7.76), ICT+CG (speedup factor of 6.25), and finally ILUT+GMRES (speedup factor of 4.58). Similarly to the Kershaw problem we again see MG (either MG alone or as preconditioner to CG) a clear winner. The large mesh problem shows even more impressive results, with MG alone a clear winner at 16.75 speedup over DSCG.

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	179.14	118.54	102.72	105.33	103.46
Radiation	156.06	93.75	78.60	82.26	80.18
Linear Solve	105.70	40.87	30.33	31.56	29.12

Table 1: Runtimes for Small Kershaw Problem (in seconds)

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	1255.81	592.61	496.61	456.99	439.55
Radiation	1161.53	491.89	397.14	362.63	345.54
Linear Solve	942.80	270.00	181.56	143.79	127.20

Table 2: Runtimes for Medium Kershaw Problem (in seconds)

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	103294.39	31096.20	15124.77	10932.68	10430.30
Radiation	101238.93	28890.89	12929.50	8866.40	8373.00
Linear Solve	96403.21	23910.35	8105.20	3993.01	3520.13

Table 3: Runtimes for Large Kershaw Problem (in seconds)

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Small Problem	1.00	2.59	3.48	3.35	3.63
Medium Problem	1.00	3.43	5.09	6.56	7.41
Large Problem	1.00	2.43	11.89	24.14	27.38

Table 4: Speedup (over DS+CG) for Kershaw Problem

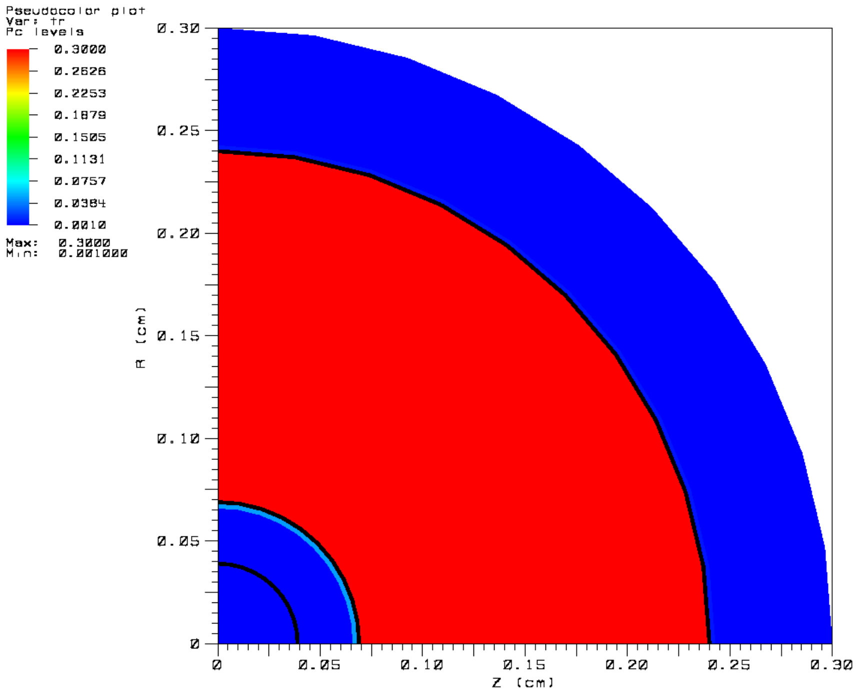

Figure 11: Initial geometry for Spherical Diffusion Problem

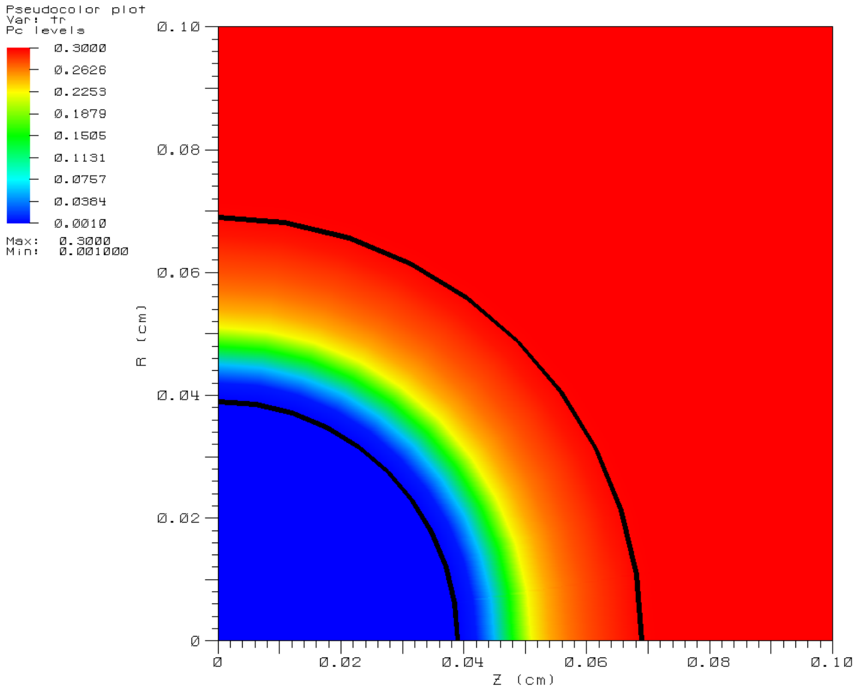

Figure 12: Snapshot 2

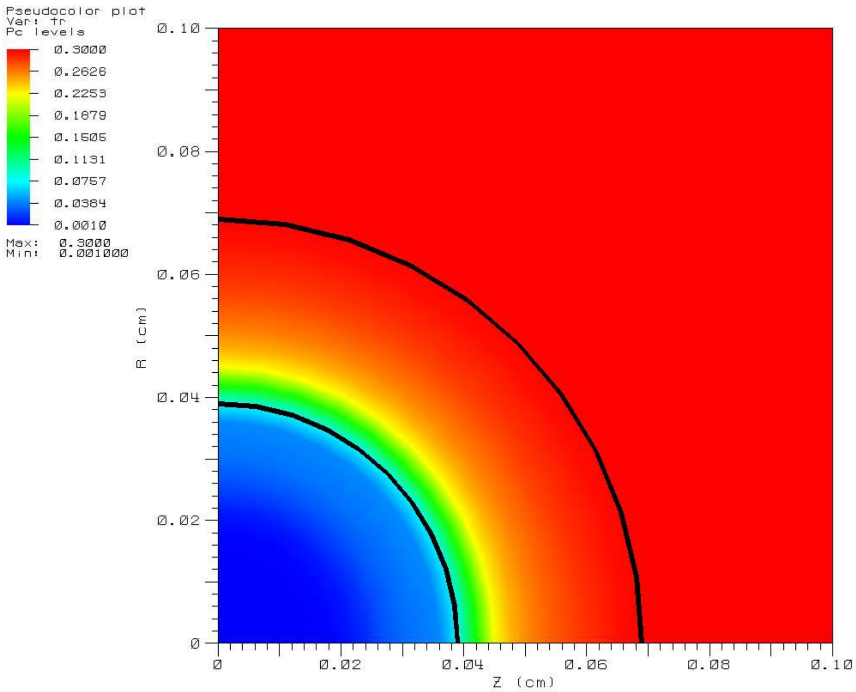

Figure 13: Snapshot 3

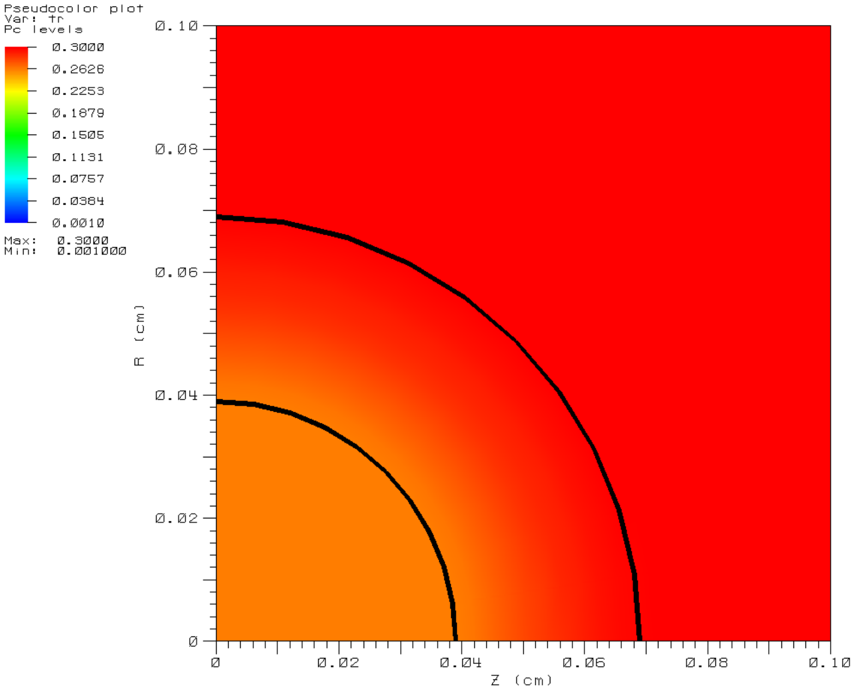

Figure 14: Snapshot 4

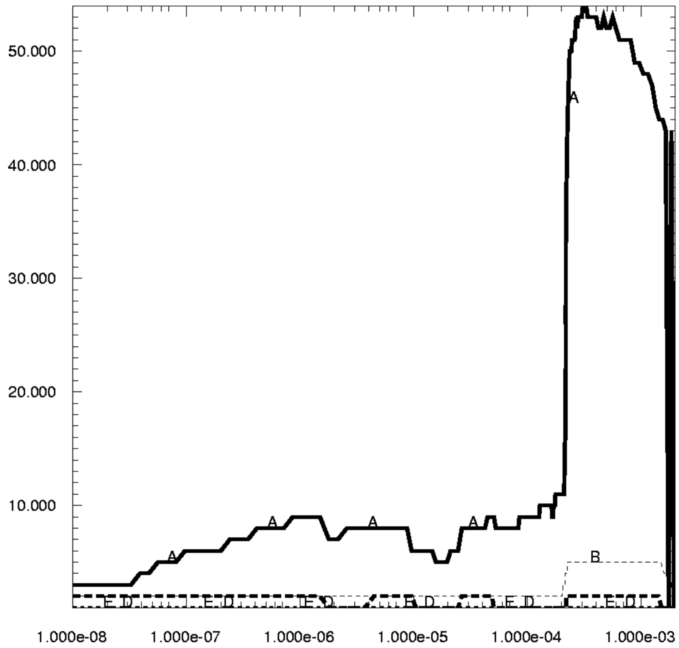

Figure 15: Iteration Counts for Small Spherical Diffusion Problem

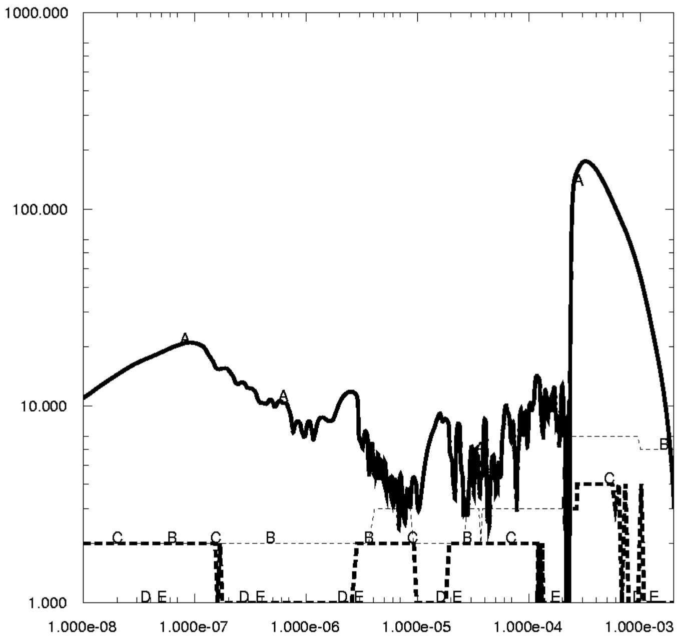

Figure 16: Iteration Counts for Medium Spherical Diffusion Problem

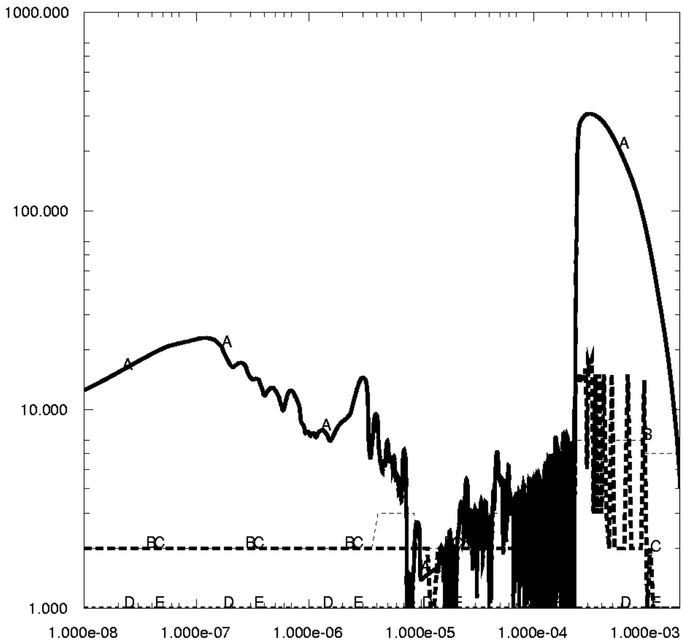

Figure 17: Iteration Counts for Large Spherical Diffusion Problem

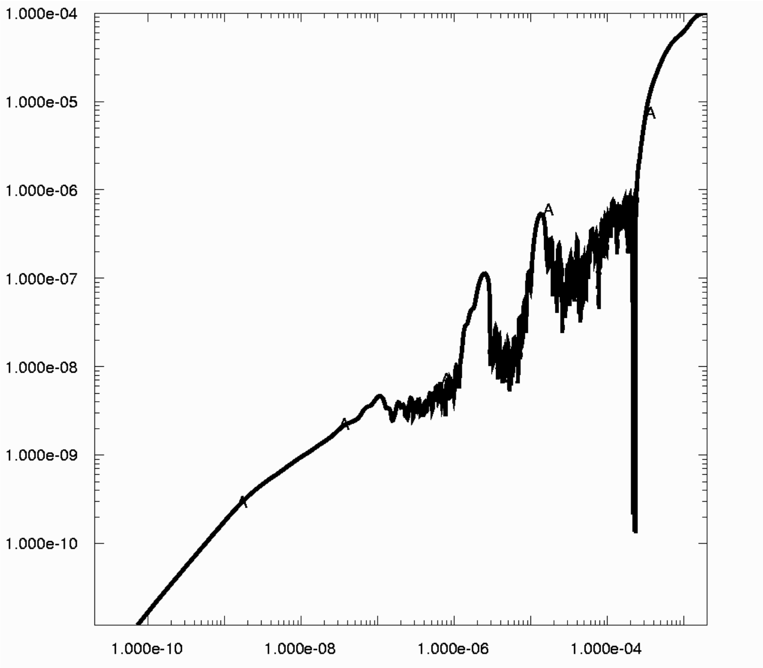

Figure 18: Time Step for Medium Spherical Diffusion Problem

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	76.95	68.42	65.42	64.52	66.18
Radiation	59.09	50.47	47.36	46.23	48.18
Linear Solve	21.22	11.78	8.92	5.93	7.95

Table 5: Runtimes for Small Spherical Diffusion Problem (in seconds)

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	6074.63	1633.89	1509.51	1393.90	1445.30
Radiation	4761.21	1267.50	1141.31	1029.40	1079.23
Linear Solve	1415.69	308.95	226.55	134.31	182.40

Table 6: Runtimes for Medium Spherical Diffusion Problem (in seconds)

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	104472.08	20361.25	12745.62	11405.62	12001.11
Radiation	82519.81	17662.52	10054.89	8746.91	9292.76
Linear Solve	23395.36	10146.93	2704.00	1397.07	1925.50

Table 7: Runtimes for Large Spherical Diffusion Problem (in seconds)

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Small Problem	1.00	1.80	2.38	3.58	2.67
Medium Problem	1.00	4.58	6.25	10.54	7.76
Large Problem	1.00	2.31	8.65	16.75	12.15

Table 8: Speedups for Spherical Diffusion Problem

5.3 Radiatively Driven Symmetric Implosion

This problem tests the radiation-hydrodynamics on a rectilinear ALE mesh in a configuration relevant to ICF calculations. The important fact concerning this problem and the next problem is that the mesh is dynamic; responding to material motion and relaxing when zone distortion has become too extreme. Figures 19, 20, 21 and 22 show time snap shots of the radiation temperature and mesh for the medium size problem. The figures show the initial 300 eV He source, the subsequent ablation of the CH foam, and finally the compression of the DT ice. Figures 23, 24 and 25 show the iteration count as a function of time for all solvers. The DSCG again shows the most iterations, with the characteristic scaling of iteration count with problem size and time step. For the small problem, the MG shows the next highest number of iterations to convergence followed by ILUT+GMRES, ICT+CG and MG+CG (the latter two show comparable iteration counts). For the medium and large problems, we see the characteristic weak scaling of ILUT+GMRES and ICT+CG with problem size and the MG+CG scheme being almost completely independent of problem size. It is interesting to note that the MG scheme, although algorithmically scalable, requires approximately 100 iterations to converge over a large portion of the problem time. This fact illustrates the utility of using MG as a preconditioner for CG to give a more robust algorithm overall.

One important feature of the time step in this type of problem (i.e. implosions), which is different from the radiation only cases discussed earlier, is the rapid decrease in the time step due to the shrinking of zones (see Figures 21 and 26. This time step drop is primarily due to the Courant condition placed on the hydrodynamics. Tables 9, 10 and 11 show the CPU time spent in the whole code, radiation package, and linear solver. In addition, the relative speedups compared to DSCG are shown in Table 12. What was surprising was the relative speedups, although significant, were not as large as were seen earlier in the pure radiation diffusion problems. For example, for the small problem, the speedups never exceeded a factor of two while the large mesh problems dont quite reach speedup factors of 10. This fact is due to two factors. One is the time step control coming from the explicit hydrodynamics (i.e Courant condition) and the other is the property of ALE to smooth out distorted meshes. In fact, if allowed to do so, ALE would try to smooth out the Kershaw mesh if the hydrodynamics were turned on. Lower time steps keep the matrix diagonally dominant and a smooth mesh, as discussed earlier, means smaller corner couplings and a more sparsely populated matrix. These two effects combine to make a matrix that is easier to invert than the Kershaw case and a matrix that is more diagonally dominant than the spherical diffusion without hydrodynamics. This latter fact is because of the rectilinear nature of the mesh which keeps the corner coupling terms small.

What is a little misleading when one looks at overall code CPU time, is the fact that

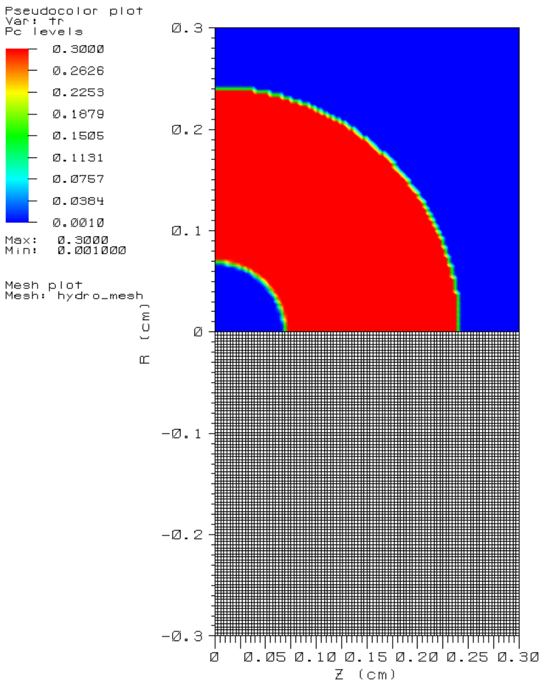

Figure 19: Initial geometry for Symmetric Implosion Problem

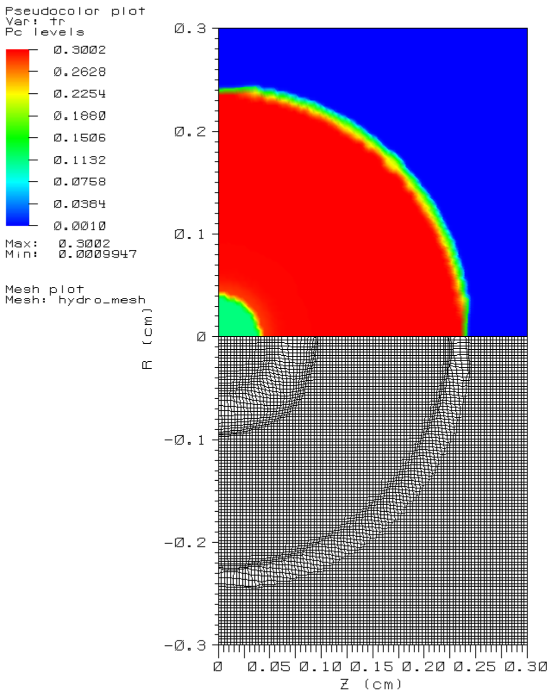

Figure 20: Snapshot 2

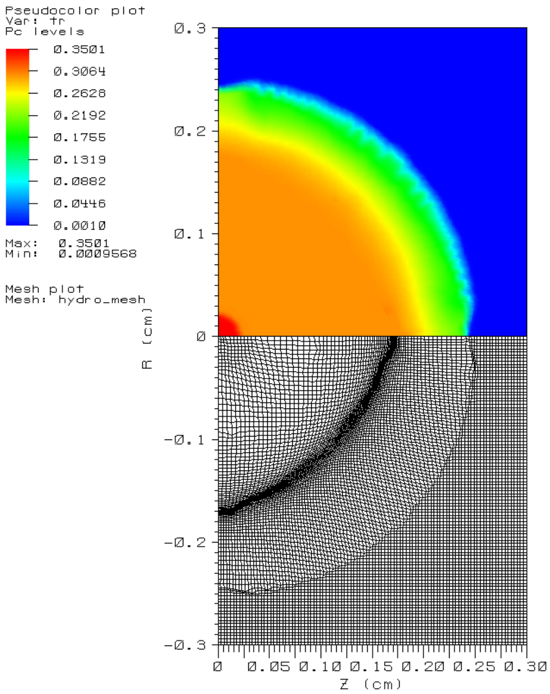

Figure 21: Snapshot 3

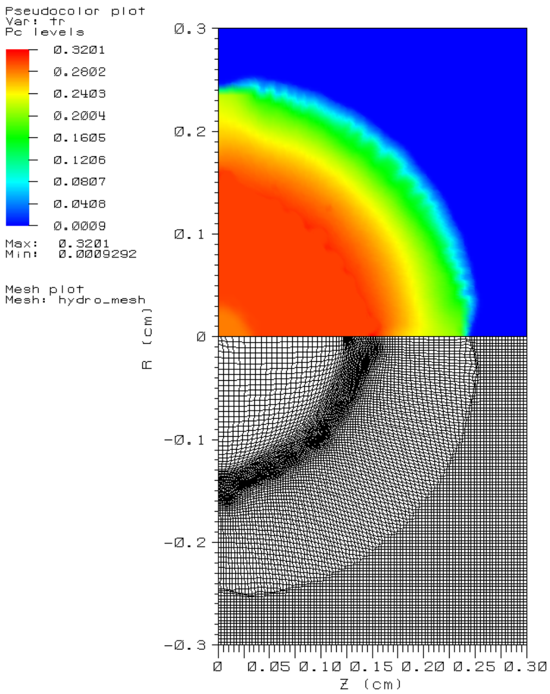

Figure 22: Snapshot 4

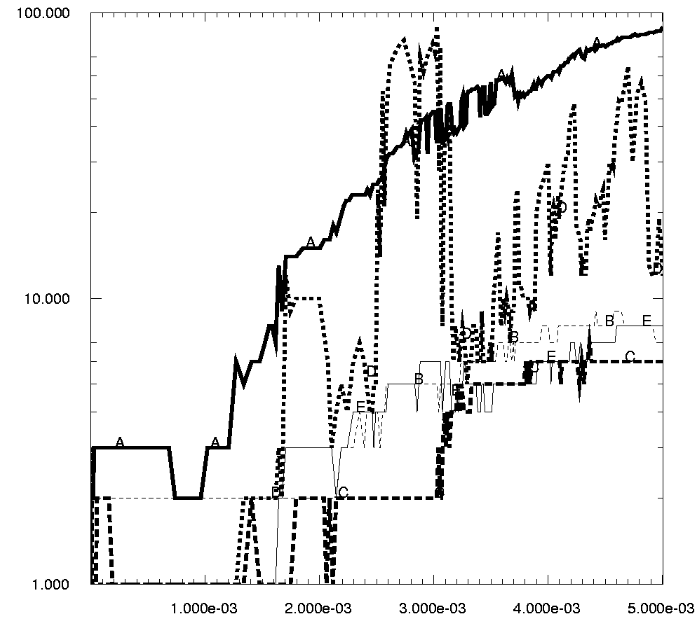

Figure 23: Iteration Counts for Small Symmetric Implosion Problem

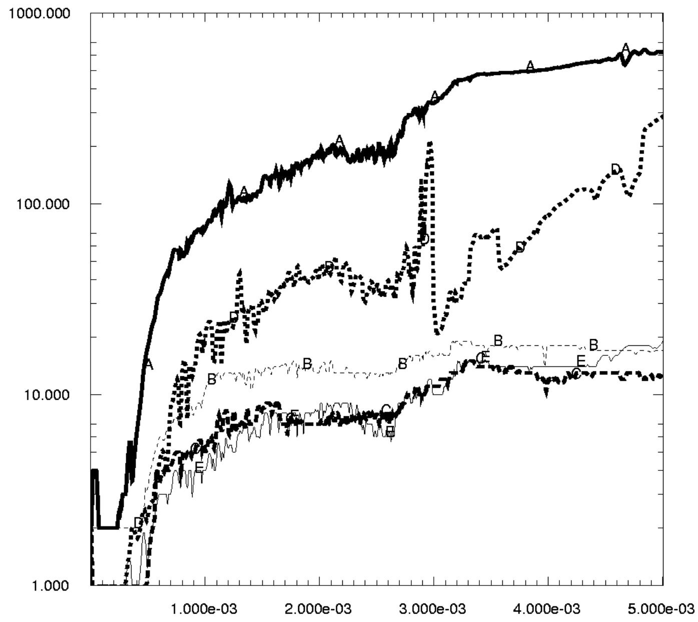

Figure 24: Iteration Counts for Medium Symmetric Implosion Problem

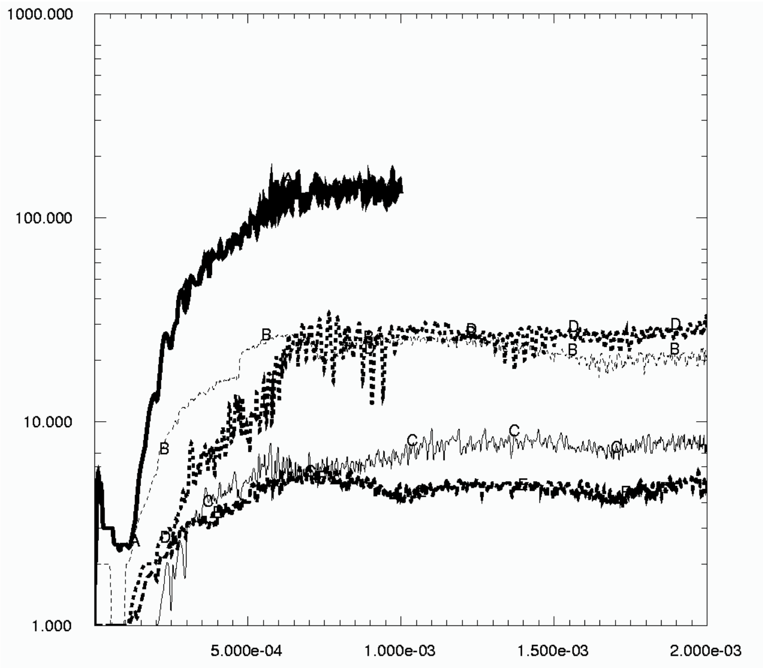

Figure 25: Iteration Counts for Large Symmetric Implosion Problem

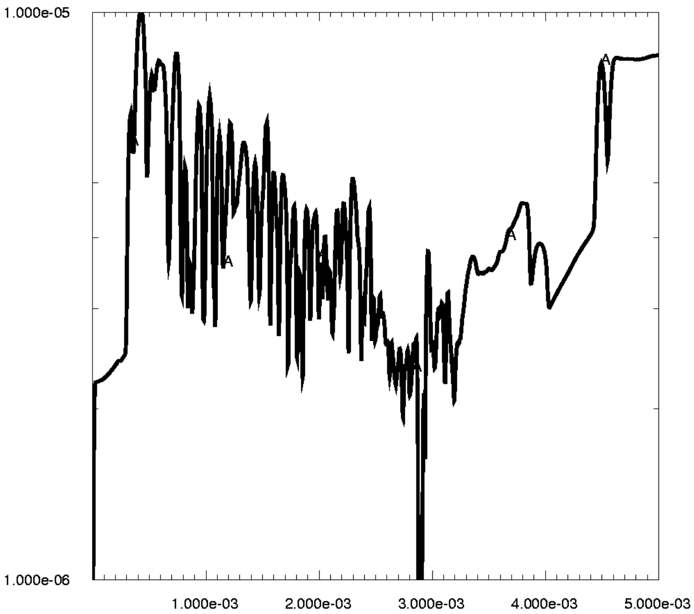

Figure 26: Time Steps for Medium Symmetric Implosion Problem

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	191.52	162.52	173.47	199.92	167.39
Radiation	79.45	58.11	60.62	91.53	63.03
Linear Solve	34.58	15.36	12.44	45.22	20.85

Table 9: Runtimes for Small Symmetric Implosion Problem (in seconds)

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	18044.61	9060.15	8289.29	14447.06	9519.38
Radiation	13397.81	4365.24	3380.86	9761.04	4662.40
Linear Solve	11415.53	2301.53	1251.66	7754.48	2571.45

Table 10: Runtimes for Medium Symmetric Implosion Problem (in seconds)

hydrodynamics looks like a bottleneck to overall code performance at least as far as the linear solvers are concerned. It should be stressed again, that these calculations were done for one group and a multigroup calculation would imply a factor of the number of groups be applied to the CPU time spent in the linear solver. Therefore, a multigroup calculation with more than 10 groups would in fact show that radiation diffusion in fact dominates the overall CPU time of the code. The importance therefore, of any speedup of a factor of 2 or more can mean big savings in CPU time for a multigroup calculation.

5.4 Radiatively Driven Asymmetric Implosion

This problem tests the radiation-hydrodynamics on an ALE mesh in a spherical geometry with a non-symmetric compression of the DT fuel. Although not particularly realistic (one would not design a capsule with this much asymmetry), this problem tests the linear solvers on a mesh which is distorted both in radial and angular directions. Figures 27, 28, 29 and 30 show time snap shots of the pressure in megabars and the mesh. The figures show the initial 300 eV He source, the subsequent ablation of the CH foam, and finally the compression of the DT ice. Note that the compression does not satisfy spherical symmetry. Figures 31, 32 and 33 show the iteration count as a function of time for all solvers. The DSCG again shows the most iterations, with the characteristic scaling of iteration count with problem size and time step. For the small problem, the MG and ILUT+GMRES show the next highest number of iterations to convergence followed by MG+CG and ICT+CG. For the medium problem, we

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	268261.96	195932.93	182955.08	225516.22	199480.29
Radiation	185486.13	100980.08	76225.40	128005.89	89666.50
Linear Solve	149820.31	58636.32	29410.26	85417.22	41950.78

Table 11: Runtimes for Large Symmetric Implosion Problem (in seconds)

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Small Problem	1.00	2.25	2.78	0.76	1.65
Medium Problem	1.00	4.96	9.12	1.47	4.44
Large Problem	1.00	2.56	5.09	1.75	3.57

Table 12: Speedups for Symmetric Implosion Problem

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	311.58	249.89	249.12	251.54	250.62
Radiation	47.63	31.12	30.66	32.37	31.82
Linear Solve	23.68	7.18	5.73	7.51	6.90

Table 13: Runtimes for Small Asymmetric Implosion Problem (in seconds)

see the characteristic weak scaling of ILUT+GMRES and ICT+CG with problem size and the MG schemes being almost completely independent of problem size. The large problem iteration count versus time shows the characteristic scalings with time step and problem size. We see both MG and MG+CG are algorithmically scalable, although the more robuse MC+CG has smaller iteration counts. Again, we see a limit to the time step growth due primarily to the Courant condition placed on the hydrodynamics, Figure 34. Tables 13, 14 and 15 show the CPU time spent in the whole code, radiation package, and linear solver. In addition, the relative speedups compared to DSCG are shown in Table 16. The results are similar to the radiatively driven symmetric implosion. That is, the results are not quite as impressive as Kershaw for the same reasons mentioned in §5.3. However, the results are better than the symmetric implosion on a rectilinear ALE mesh. The reason for this is that the highly asymmetric nature of the implosion is causing the mesh to distort giving rise to a matrix with a larger number of corner coupling terms. Note that again it is ICT+CG beating multigrid, albeit weakly for the small, medium, and large problems.

We repeat the comments of §5.3 for emphasis. These calculations were done for one group and a multigroup calculation would imply a factor of the number of groups be applied to the CPU time spent in the linear solver. Therefore, a multigroup calculation with more than 10 groups would in fact show that radiation diffusion in fact dominates the overall CPU time of the code. The importance therefore, of any speedup of a factor of 2 or more can mean big savings in CPU time for a multigroup calculation.

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	5043.90	4126.60	4532.59	4385.09	4102.50
Radiation	810.44	324.74	277.17	259.27	283.96
Linear Solve	600.55	112.16	67.11	52.16	76.76

Table 14: Runtimes for Medium Asymmetric Implosion Problem (in seconds)

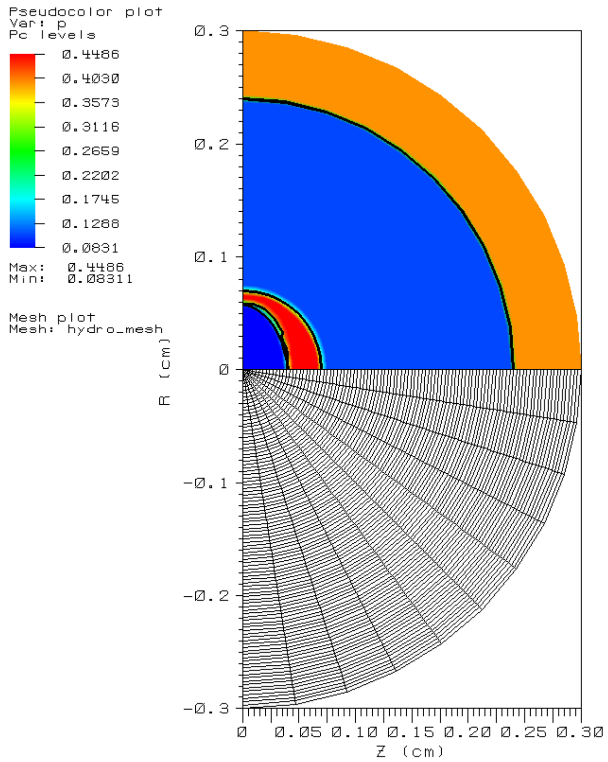

Figure 27: Initial geometry for Asymmetric Implosion Problem

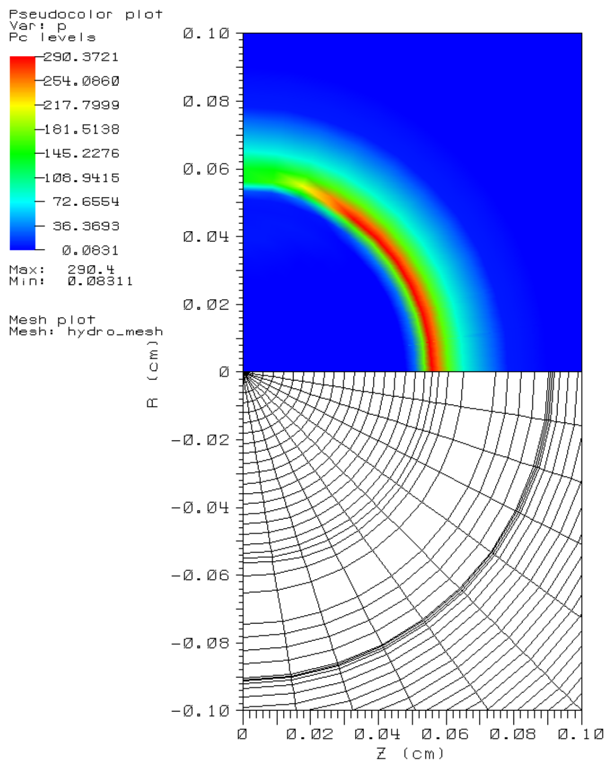

Figure 28: Snapshot 2

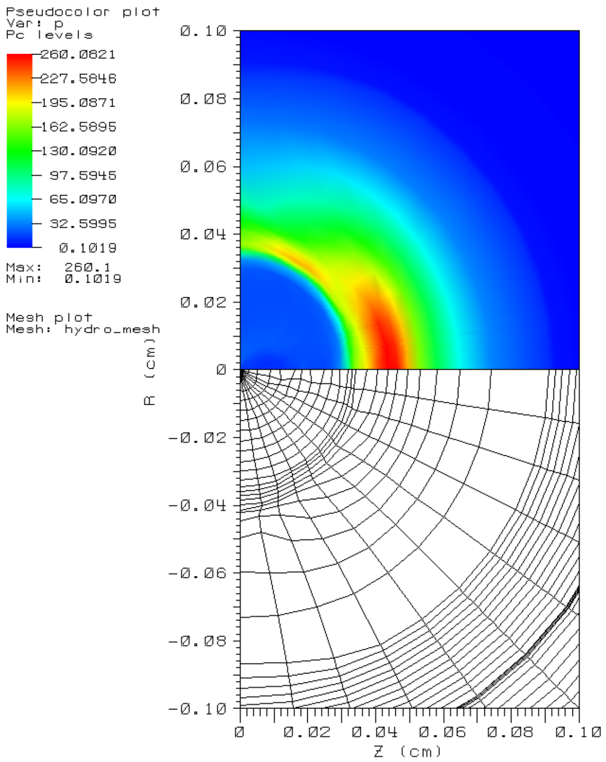

Figure 29: Snapshot 3

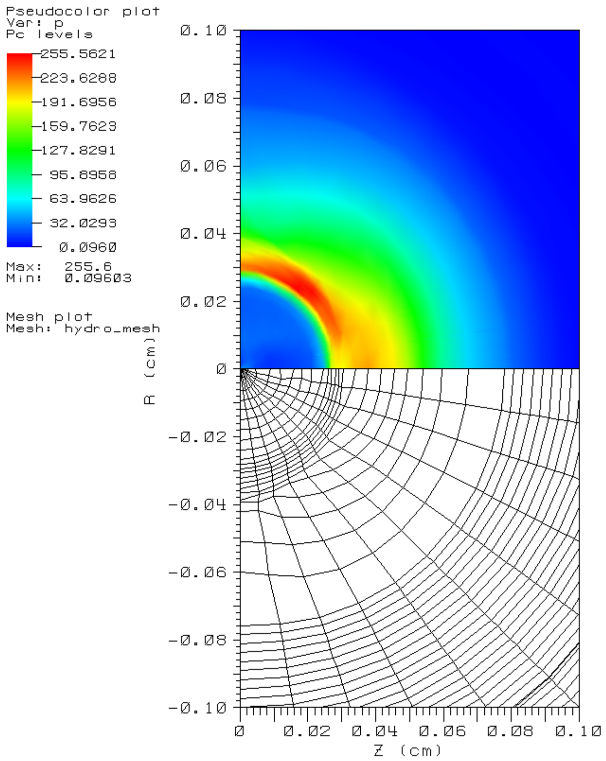

Figure 30: Snapshot 4

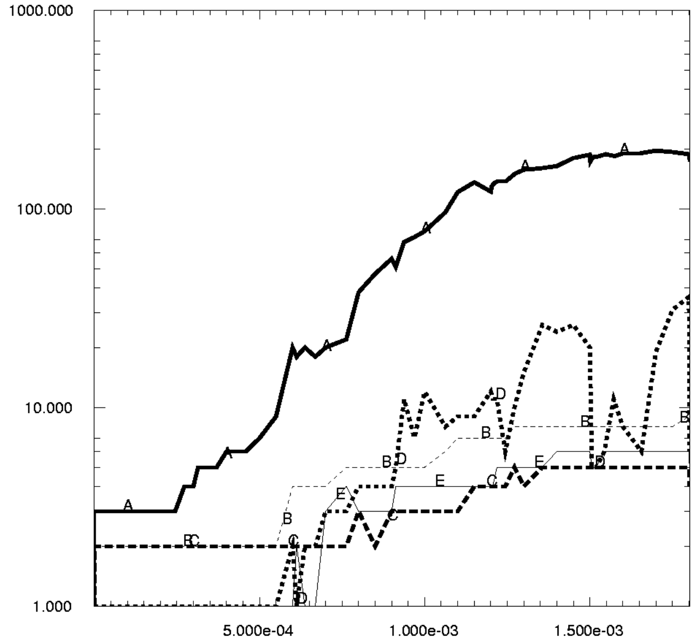

Figure 31: Iteration Counts for Small Asymmetric Implosion Problem

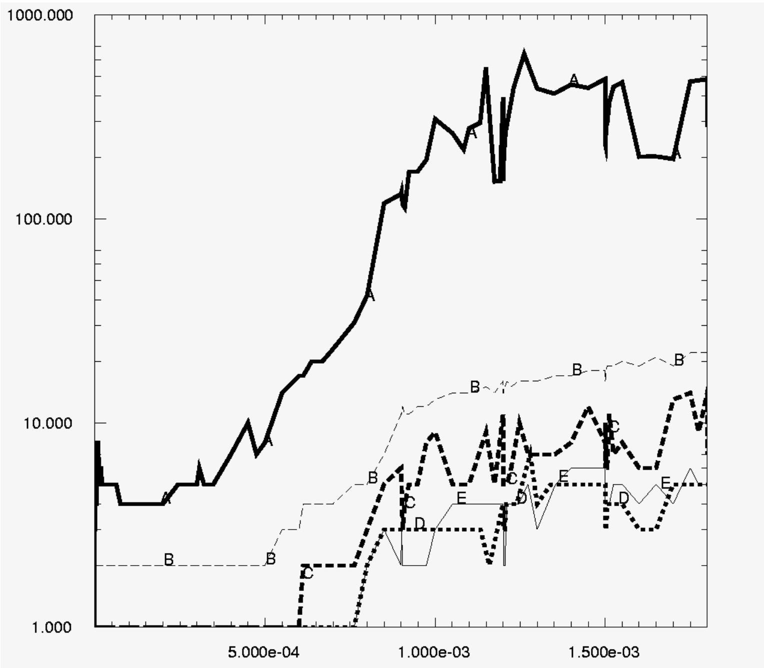

Figure 32: Iteration Counts for Medium Asymmetric Implosion Problem

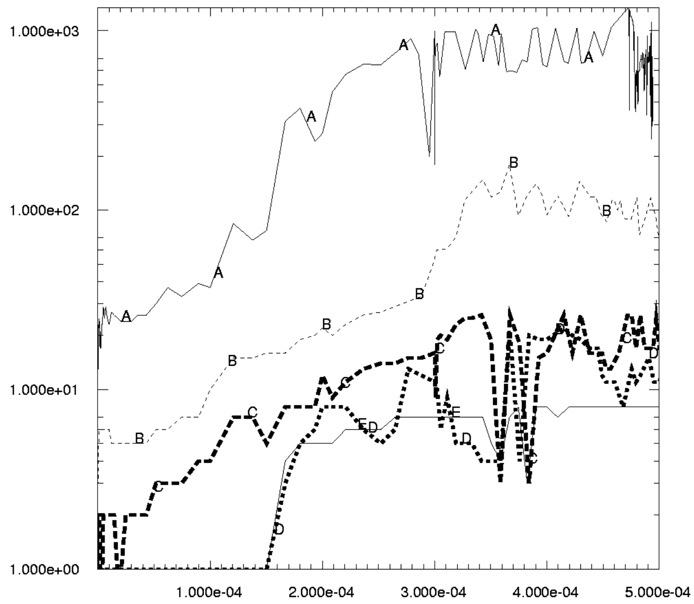

Figure 33: Iteration Counts for Large Asymmetric Implosion Problem

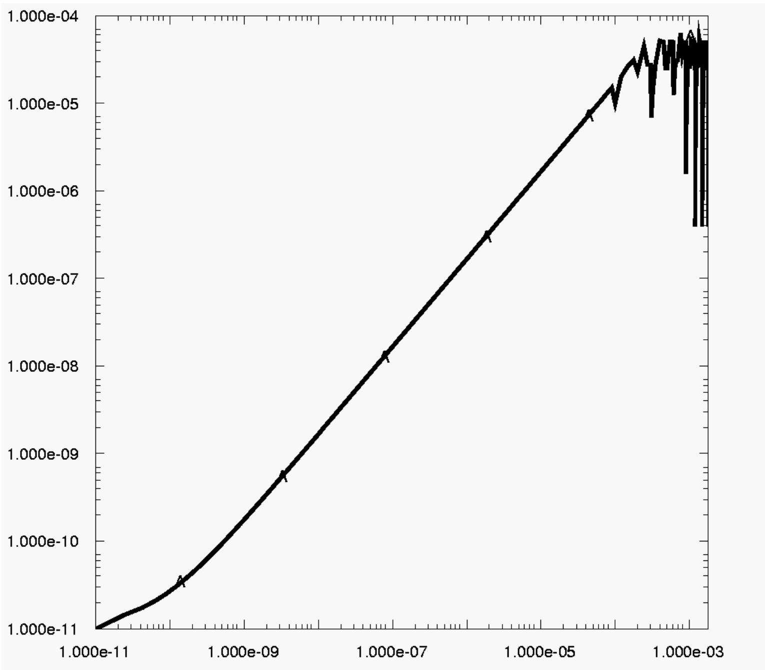

Figure 34: Time Steps for Medium Asymmetric Implosion Problem

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Whole Code	445876.04	1165554.15	1150199.61	0.0	1102108.65
Radiation	194675.51	174912.03	105895.58	0.0	75831.09
Linear Solve	185606.67	122919.69	31889.33	0.0	23107.11

Table 15: Runtimes for Large Asymmetric Implosion Problem (in seconds)

	DS+CG	ILUT+GMRES	ICT+CG	MG	MG+CG
Small Problem	1.00	3.30	4.13	3.15	3.43
Medium Problem	1.00	5.35	8.95	11.51	7.82
Large Problem	1.00	1.51	5.82	0.00	8.03

Table 16: Speedup (over DS+CG) for Asymmetric Implosion Problem

6 Conclusions

Computer codes containing both hydrodynamics and radiation play a central role in simulating both astrophysical and inertial confinement fusion (ICF) phenomena. A crucial aspect of these codes is that they require an implicit solution of the radiation diffusion equations. We have shown in this paper the results of a comparison of five different linear solvers (diagonally scaled conjugate gradient (DSCG), GMRES with incomplete LU preconditioning (ILUT+GMRES), conjugate gradient with incomplete Cholesky preconditioning (ICT+CG), multigrid (SMG), and multigrid-preconditioned conjugate gradient (SMG+CG)) over a range of complex radiation and radiation-hydrodynamics problems, and over a range of problem sizes. The importance of scalable linear solvers is clearly manifest when timing comparisons are performed between DSCG, ILUT+GMRES, ICT+CG and SMG or SMG+CG for pure radiation flow problems. The large size pure radiation flow problems show a speed-up factor of $\approx$30 when SMG or SMG+CG is compared to DSCG. The timing differences between the linear solvers becomes all the more important when multigroup calculations are performed. For problems involving radiation-hydrodynamic flows, the situation is more complex. The results of our scalability study clearly show the inadequecies of DSCG as both problem size and time step grows for these type of problems. However, ICT+CG is comparable and even slightly better than SMG or SMG+CG. For the large symmetric and non-symmetric radiatively driven implosions, ICT+CG, SMG, and SMG+CG all show speed-up factor of $\approx$10 compared to DSCG. This observation is tied to several facts. In a multiphysics code such as the code discussed in this paper, time step controls and mesh relaxation plays an important role in determining the nature and structure of the matrix. The performance figures presented here for the linear solvers, are in fact "integral" quantities. What we observe is a close correlation between time step, the diagonal dominance of the matrix, and the subsequent iteration count of the linear solver. Since the problems presented in this study all start at a relatively small time step, DSCG beats all solvers up to a point where DSCG is taking $\approx$100 iterations to converge. At this point, ILUT+GMRES, ICT+CG, SMG, and SMG+CG all become cost effective and outperform DSCG. For a large enough time step, SMG and SMG+CG beat all solvers. This is what we observe in the Kershaw and spherical diffusion problem. However, in cases where hydrodynamic flows and time scales are involved, the code may lower the time step due to converging shocks thus making the matrix more diagonally dominant and therby making it easier for ILUT+GMRES or ICT+CG to solve than either SMG or SMG+CG. This behavior is observed in the implosion problems. The above discussion leads to the fact that a code with some measure of adaptivity in regards to linear solver choice will run optimally. How and when this choice is to be made in a radiation-hydrodynamics code is an interesting issue and we leave this topic to future discussion.

References

[1] R. Barton, *Development of a multimaterial 2d arbitrary lagrangian-eulerian mesh computer program*, in Numerical Astrophysics, L. Centrella and Wheeler, eds., Jones and Bartlett Publishers, Boston, 1985.

[2] R. Bowers and J. Wilson, *Numerical Modeling in Applied Physics and Astrophysics*, Jones and Bartlett Publishers, Boston, 1991.

[3] A. Brandt, *Multi–level adaptive solutions to boundary–value problems*, Math. Comp., 31 (1977), pp. 333–390.

[4] W. L. Briggs, *A Multigrid Tutorial*, SIAM Books, Philadelphia, 1987.

[5] P. N. Brown, E. Chow, and Y. Saad, *ICT: a dual threshold incomplete LDL^T factorization*, Tech. Rep. LLNL UCRL Tech Report, Lawrence Livermore National Laboratory, 1998. In preparation.

[6] H. Brysk, Plasma Physics, 16 (1974).

[7] J. E. Dendy, M. P. Ida, and J. M. Rutledge, *A semicoarsening multigrid algorithm for SIMD machines*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 1460–1469.

[8] C. A. J. Fletcher, *Computational Techniques for Fluid Dynamics*, Springer-Verlag, New York, 1991.

[9] W. Hackbusch, *Multigrid Methods and Applications*, vol. 4 of Computational Mathematics, Springer–Verlag, Berlin, 1985.

[10] M. J. Holst and S. Vandewalle, *Schwarz methods: To symmetrize or not to symmetrize*, SIAM J. Numer. Anal., 34 (1997), pp. 699–722.

[11] Y. Lee and R. More, Physics of Fluids, 27 (1984).

[12] W. A. Lokke and W. H. Grasberger, *Xsnq-u, a non-ite emission and absorption coefficient subroutine*, Tech. Rep. UCRL-52276, Lawrence Livermore National Laboratory, 1977.

[13] D. Mihalas and B. Weibel-Mihalas, *Foundations of Radiation Hydrodynamics*, Oxford University Press, New York, 1984.

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

[15] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Comapny, Boston, 1996.

[16] S. Schaffer, *A semi-coarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients*, SIAM J. Sci. Stat. Comput. to appear.

[17] R. A. Smith and A. Weiser, *Semicoarsening multigrid on a hypercube*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 1314–1329.

[18] R. E. Tipton. Personal Communication.

[19] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley & Sons, Chichester, 1992.