Electronic Transactions on Numerical Analysis.
Volume 15, pp. 78-93, 2003.
Copyright © 2003, Kent State University.
ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

COMPARISON OF NON-LINEAR SOLVERS FOR THE SOLUTION OF RADIATION TRANSPORT EQUATIONS*

LINDA STALS†

Abstract. We compare the performance of an inexact Newton-multigrid method and Full Approximation Storage multigrid when solving radiation transport equations. We also present an adaptive refinement algorithm and explore its impact on the solution of such equations.

Key words. FAS, Multigrid, Newton Method, Radiation Transport

AMS subject classifications. 35K55, 65M55, 65M60, 49M15

1. Introduction. Interest in the solution of radiation transport equations stems from the modeling of applications found in, for example, combustion, astrophysics and laser fusion. Features such as strong nonlinearities and large jumps in the coefficients make these equations difficult to solve and they can consume considerable computational resources if efficient solution techniques are not used. Two examples of efficient solution methods are the inexact Newton-multigrid method and the Full Approximation Storage (FAS) multigrid; both of which are nonlinear multigrid techniques.

The solution of radiation transport equations usually contains a wave front. Adaptively refined grids are well suited to capture the information along the front and thus give high resolution results.

In this paper we compare the performance of an inexact Newton-multigrid method and the FAS method for the solution of time-dependent radiation transport equations. We also explore the use of adaptive refinement techniques.

2. Physical Model. Under certain assumptions, such as isotropic radiation, optically thick material and temperature equilibrium, radiation transport may be modeled by the equation:

$$\frac{\partial E}{\partial t} - \nabla.(D(E)\nabla E) = 0 \qquad \text{on } \Omega \times I, \tag{2.1}$$

where E is the radiation energy.

More physically meaningful models of radiation transport are represented by systems of equations like those described in [4, 12, 14]; however, Equation (2.1) contains many of the features seen in the more general system, such as strong nonlinearities and large jumps in the coefficients, and therefore is a good place to start our investigation into different solution techniques.

One definition of the diffusivity term, $D(E)$, is:

$$D_1(E) = Z^{\alpha} E^{\beta}, \tag{2.2}$$

with $\alpha < 0$, $0 \leq \beta \leq 1$. Typically α is taken to be between -1 and -3 while β is taken to be $1/4$ or $3/4$. Z is the atomic mass number and may vary within the domain to reflect inhomogeneities in the material. The constant β controls the strength of the nonlinearity while α affects the size of the jumps in the coefficients.

*This research was supported by the U.S. Department of Energy under the ASCI ASAP program (subcontract B347882 from the Lawrence Livermore National Laboratory). Received May 22, 2001. Accepted for publication October 20, 2001. Recommended by Van Henson.

†Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162, USA AND ICASE, NASA Langley Res. Ctr, Hampton, VA 23681-2199, USA.

ETNA
Kent State University
etna@mcs.kent.edu

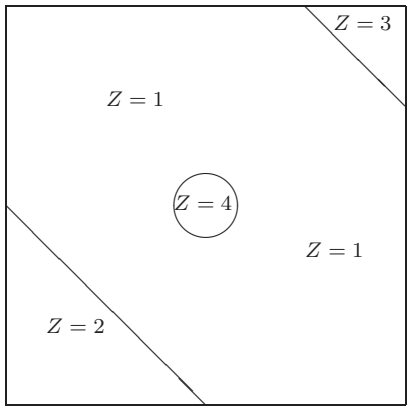

FIG. 2.1. *The values for the atomic mass number Z depend on the topology of the material. In our model problem, we define Z as shown above.*

The definition of $D(E)$ given in Equation (2.2) may produce results that shows the energy moving through the system at a rate faster than the speed of light. Consequently a flux limiter is included to slow down the movement and the diffusivity term is rewritten as:

$$D_2(E) = \left(\frac{1}{Z^\alpha E^\beta} + \frac{|\nabla E|}{E} \right)^{-1}. \tag{2.3}$$

The domain, Ω, is a square domain ($[0,1] \times [0,1]$) with the following mixture of Newton and Neumann boundary conditions;

$$\begin{array}{ll} \partial E/\partial n = 0 & \text{on } \Gamma_N \times I, \\ n^T D(E) \nabla E + E/2 = 2 & \text{on } \Gamma_{F_0} \times I, \\ n^T D(E) \nabla E + E/2 = 0 & \text{on } \Gamma_{F_1} \times I, \end{array}$$

where Γ_N represents the lines $y = 0$ and $y = 1$, Γ_{F_0} is the line $x = 0$ and Γ_{F_1} is the line $x = 1$, n is the outward unit normal and I is the time interval.

In our model problem we take Z to be 1.0 except in the following regions: $\sqrt{(x-0.5)^2 + (y-0.5)^2} \leq 0.125$, $y \leq 0.5 - x$ and $y \geq 1.75 - x$ where $Z = 4$, $Z = 2$ and $Z = 3$ respectively. See Figure 2.1.

Initially, the energy, E is set to be the constant value $E = 10^{-5}$.

Additional papers that look at these equations include [3, 4, 5, 7, 12, 16] and their accompanying references. The radiation transport model described here is very similar to that presented in [7, 12, 16], except that we use the finite element method instead of the finite difference method, as this more readily allows the use of adaptive refinement.

3. Discretization. To solve Equation (2.1), we use the following variational formulation, similar to that given in [6]: Find $u(t) \in V = H^1(\Omega)$, $t \in I$, such that

$$(\dot{u}(t), v)_\Omega + a(D(u(t)); u(t), v)_\Omega =< g, v >_{\partial\Omega} \qquad \forall v \in V, \tag{3.1}$$

where

$$a(D(u); v, w)_\Omega = \int_\Omega D(u) \nabla v \nabla w \, d\Omega + \int_{\Gamma_{F_0}} \frac{vw}{2} \, d\Gamma + \int_{\Gamma_{F_1}} \frac{vw}{2} \, d\Gamma,$$

ETNA
Kent State University
etna@mcs.kent.edu

$$< v, w >_{\partial\Omega} = \int_{\partial\Omega} vw \, d\Omega,$$

and

$$g = \begin{cases} 0 & \text{on } \Gamma_N \times I, \\ 2 & \text{on } \Gamma_{F_0} \times I, \\ 0 & \text{on } \Gamma_{F_1} \times I. \end{cases}$$

The time derivative is dealt with by either an implicit Euler or the Crank-Nicolson method.

4. Node-Edge Data Structure. A node-edge data structure similar to the one described in [15, 17, 18, 20] is used to store the finite element grid. In this data structure, a grid $\mathcal{M}^m$ is defined in terms of its geometrical, topological and algebraic attributes. The geometrical and topological attributes are simply the set of *nodes*, $\mathcal{N}^m$, and *edges*, $\mathcal{E}^m$. The stiffness matrix is associated with the algebraic attribute and is stored in the set of connections, $\mathcal{C}_A^m$, as a graph.

The node-edge data structure does not *explicitly* contain any information about the elements in the grid. Consequently, the same data structure can be used to store triangular, quadrilateral or tetrahedral grids. Information about the elements may be extracted by looping through the nodes and edges if necessary. The advantage of such a data structure is its flexibility. Although we concentrate on serial results in this paper, the code has been implemented in parallel and those results will be presented in a future paper.

We use a refinement algorithm to build the hierarchy of grids needed by our solution techniques (multigrid methods). In notational form this nested sequence of grids is represented by $\mathcal{M}^1 \subset \mathcal{M}^2 \subset ... \subset \mathcal{M}^n$. A more thorough description of this refinement algorithm is given in Section 6.

Information is moved from the coarse grid $\mathcal{M}^{m-1}$ to the fine grid $\mathcal{M}^m$ by the linear interpolation matrix $\mathbf{I}_{m-1}^m$. The restriction matrix $\mathbf{I}_m^{m-1}$ is defined to be the transpose of the interpolation matrix and moves the information from the fine grid $\mathcal{M}^m$ to the coarse grid $\mathcal{M}^{m-1}$. This extra algebraic information is stored in the table of connections. That is, if $\mathcal{C}_I^m$, $\mathcal{C}_R^m$ and $\mathcal{C}_A^m$ hold the interpolation, restriction and stiffness matrices, respectively, then the algebraic information for a multigrid grid is the set of *connections*, $\mathcal{C}^m$, where

$$\mathcal{C}^m = \mathcal{C}_A^m \cup \mathcal{C}_I^m \cup \mathcal{C}_R^m,$$

for $1 < m < n$, $\mathcal{C}^1 = \mathcal{C}_A^1 \cup \mathcal{C}_I^1$ and $\mathcal{C}^n = \mathcal{C}_A^n \cup \mathcal{C}_R^n$.

Finally, the grid $\mathcal{M}^m$ is given by $\mathcal{M}^m = \{\mathcal{N}^m, \mathcal{E}^m, \mathcal{C}^m\}$.

Conceptually the user views $\mathcal{E}^m$ as a set of edges, but internally they are stored as sets of neighbor endpoints. Given a node N_i it's set of neighbor endpoints is defined to be:

$$\mathcal{B}(\mathcal{E}, N_i) = \{N_j : (N_i, N_j) \in \mathcal{E}\}.$$

The neighbor endpoints readily allow us to find information about the triangles in the grid. For example, to build a table of triangles, $\mathcal{T}$, the following algorithm can be used:

```
for N_i ∈ N
    for N_j ∈ B(E, N_i)
        for N_k ∈ B(E, N_i)
            if (N_j, N_k) ∈ E
              T ← {N_i, N_j, N_k} ∪ T
```

ETNA
Kent State University
etna@mcs.kent.edu

5. Solution Techniques. We compare two different solution techniques: the inexact Newton-multigrid method and the Full Approximation Storage (FAS) multigrid. Note that Newton's method relies on a global linearisation sweep whereas the FAS scheme uses local linearisations.

5.1. Inexact Newton-Multigrid. Our implementation is a standard implementation of Newton's method, but we have included a brief description below to aid in the discussion of the numerical results.

Suppose we want to solve the nonlinear system $\mathbf{N}[\mathbf{x}] = \mathbf{b}$ where $\mathbf{N}$ is a nonlinear operator. Let $\mathbf{F}[\mathbf{x}] = \mathbf{b} - \mathbf{N}[\mathbf{x}]$ and take an initial guess $\mathbf{x}_0$. A high level algorithm for the inexact Newton-multigrid method is;

While $|\mathbf{F}[\mathbf{x}_k]|| >$ a given tolerance
  Calculate the Jacobian $\mathbf{F}'(\mathbf{x}_k)$
  Construct the coarse grid operator
  Solve the linear system $\mathbf{F}'(\mathbf{x}_k)\mathbf{y} = -\mathbf{F}[\mathbf{x}_k]$ by using the multigrid method
  Find the scaling factor γ
  Set $\mathbf{x}_{k+1} = \gamma\mathbf{y} + \mathbf{x}_k$

The method is inexact because the linear system is not solved exactly; we just require that

$$||\mathbf{F}[\mathbf{x}_k] + \mathbf{F}'(\mathbf{x}_k)\mathbf{y}|| < ||\mathbf{F}[\mathbf{x}_k]||/10.$$

The scaling factor is defined as

$$\gamma = \min\left\{1, 1/||\mathbf{c}||_2\right\}^{\frac{1}{4}}, \tag{5.1}$$

where $\mathbf{c}_i = \mathbf{y}_i/(\mathbf{x}_k)_i$. It is necessary to include a scaling factor when solving the time-dependent problems as the solution changes rapidly near wave fronts. This is similar to the scaling factor defined in [16].

When using the multigrid algorithm to solve a linear system $\mathbf{A}^n\mathbf{y} = \mathbf{f}^n$ given on a fine grid, the coarse grid operators are defined by the equation;

$$\mathbf{A}^{m-1} = \mathbf{I}_m^{m-1}\mathbf{A}^m\mathbf{I}_{m-1}^m, \tag{5.2}$$

where $\mathbf{A}^m$ $(1 < m \leq n)$ is the matrix defined on the grid $\mathcal{M}^m$, and $\mathbf{I}_m^{m-1}$ and $\mathbf{I}_{m-1}^m$ are the restriction and interpolation matrices introduced in Section 4.

A nested iteration scheme was used to obtain the initial guess $\mathbf{x}_0$.

5.2. FAS Scheme. A high level algorithm of the FAS scheme is given below. A more thorough description can be found in, for example, [1, 2, 8].

While $|\mathbf{F}[\mathbf{x}_k]|| >$ a given tolerance
  If not at coarsest grid
    Apply a nonlinear smoother μ_1 times to the system $\mathbf{N}^m[\mathbf{x}^m] = \mathbf{b}^m$
    Compute the defect $\mathbf{d}^m = \mathbf{b}^m - \mathbf{N}^m[\mathbf{x}^m]$
    Restrict the defect $\mathbf{d}^{m-1} = \mathbf{I}_m^{m-1}\mathbf{d}^m$
    Restrict the current approximation $\mathbf{x}^{m-1} = \widehat{\mathbf{I}}_m^{m-1}\mathbf{x}^m$
    Compute the approximation to
      $\mathbf{N}^{m-1}[\mathbf{y}^{m-1}] = \mathbf{d}^{m-1} + \mathbf{N}^{m-1}[\mathbf{x}^{m-1}]$
      by calling the FAS Scheme again using $\mathbf{x}^{m-1}$ as an initial guess

ETNA
Kent State University
etna@mcs.kent.edu

Calculate the correction $\widehat{\mathbf{x}}^{m-1} = \mathbf{y}^{m-1} - \mathbf{x}^{m-1}$
Interpolate the correction $\widehat{\mathbf{x}}^{m} = \mathbf{I}_{m-1}^{m}\widehat{\mathbf{x}}^{m-1}$
Correct the current approximation $\mathbf{x}^{m} = \mathbf{x}^{m} + \widehat{\mathbf{x}}^{m}$
Apply a nonlinear smoother μ_2 times to the system $\mathbf{N}^m[\mathbf{x}^m] = \mathbf{b}^m$
else
Solve the coarse grid problem $\mathbf{N}^1[\mathbf{x}^1] = \mathbf{b}^1$ by using Newton's method.

Notice that the Jacobian matrix is not formed, except on the coarsest grid, so the FAS scheme requires less storage than Newton's method.

The matrix $\widehat{\mathbf{I}}_m^{m-1}$ is a restriction operator, but not necessarily the same as $\mathbf{I}_m^{m-1}$. We used injection for this operation.

The nonlinear SOR method [13] was used as a smoother by the FAS scheme. The linearisation phase is incorporated in the smoother by applying a point-Newton method to a given grid node after 'fixing' the value at all of the other nodes. To apply the point Newton method, the diffusivity term (and its derivative) must be evaluated, which is expensive.

6. Grid Refinement. The refinement algorithm is based on the newest node bisection method. In this method, the triangles are subdivided by bisecting the edges that sit opposite the newest nodes. For example, if the dark points in Figure 6.1 are the newest nodes, then the resulting grid after one and two refinement sweeps are shown in Figure 6.2.

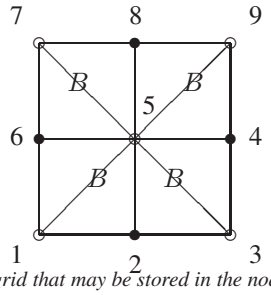

FIG. 6.1. *Example grid that may be stored in the node-edge data structure.*

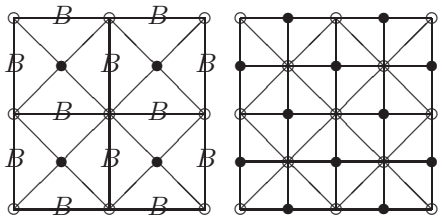

FIG. 6.2. *Resulting grid after two non-adaptive refinement sweeps of the grid in Figure 6.1. Note that the edges have been bisected along the base edges marked by a B.*

In terms of the node-edge data structure, it is easier to work with the base edges rather then the newest nodes, where the base edges are the edges that sit opposite the newest nodes, such as those marked by B in Figure 6.1.

6.1. Controlling the order of refinement. The most difficult part of the adaptive refinement routine is ensuring that the edges are bisected in the correct order to avoid long thin triangles. For example, suppose a triangle in Figure 6.1 is refined along one of the base edges as shown in Figure 6.3; then several of the triangles in the resulting grid will have two base edges. If the edges B_1 and B_2 are bisected during the next refinement sweep, then it is not

ETNA
Kent State University
etna@mcs.kent.edu

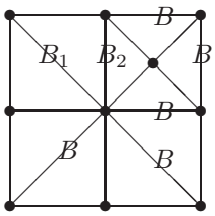

FIG. 6.3. *Result after bisecting the grid in Figure 6.1 along one of the base edges.*

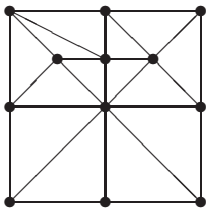

FIG. 6.4. *If the wrong edge (B_1) is bisected first, then the triangles can become long and thin.*

clear which base edge should be bisected first. If the wrong edge is chosen, as in Figure 6.4, we get long thin triangles that reduce the grids efficiency.

To determine the order in which to bisect the edges, we use a method similar to Mitchell's Compatibly Divisible Triangles [9, 10, 11].

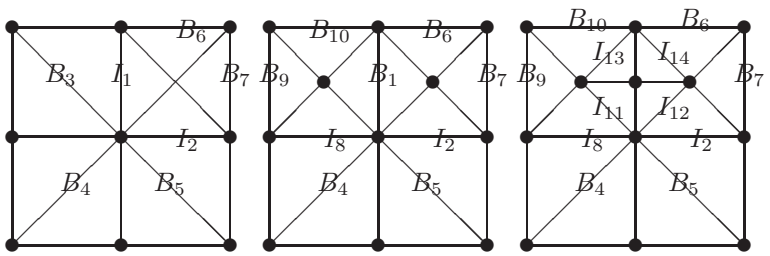

FIG. 6.5. *The interface-base edges marked by I sit between two different levels of refinement. Once an interface-base edge has been updated to a base edge, it may be bisected.*

We define an interface-base edge to be an edge that sits between two different levels of refinement. For example, in Figure 6.5, we have redrawn the grid from Figure 6.3 and marked the interface-base edges by an I. The neighboring coarse triangles must be refined before the interface-base edges are bisected. So B_3 in Figure 6.5 must be bisected before I_1. Note that it may be necessary to refine more then one neighboring triangle. For example, to bisect edge I_{11} in Figure 6.5, edges B_4 and I_8 should be bisected first.

6.2. Error Indicator (Stationary Problem). The idea behind our error indicator is to determine if the addition of a new node will significantly reduce the error.

Let $\mathbf{v}^m$ be the current approximation to the system of equations $\mathbf{N}^m[\mathbf{u}^m] = \mathbf{b}^m$, where $\mathbf{N}^m$ and $\mathbf{b}^m$ are the stiffness matrix and load vector defined on the current grid, $\mathcal{M}^m$. Then, each base edge and interface-base edge is assigned an error indicator that is equal to a weighted residual calculated at its midpoint. That is, if node i is the midpoint of a edge then the error indicator e^m assigned to that edge is:

ETNA
Kent State University
etna@mcs.kent.edu

$$e^m = \frac{\mathbf{r}_i^{m+1}}{\mathbf{N}_{i,i}^{m+1}[\mathbf{I}_m^{m+1}\mathbf{v}^m]}, \tag{6.1}$$

where

$$\mathbf{r}^{m+1} = \mathbf{b}^{m+1} - \mathbf{N}^{m+1}[\mathbf{I}_m^{m+1}\mathbf{v}^m].$$

$\mathbf{N}^{m+1}$ and $\mathbf{b}^{m+1}$ are the resulting stiffness matrix and load vector that would result if the edge was bisected at node i to form a new set of triangles. Note that it is not necessary to construct the whole stiffness matrix (or load vector); we only need the row corresponding to node i.

The motivation behind the error indicator is the question: how much will the error be reduced if we add node i? In regions of the domain where the solution is well approximated by the coarse grid we would expect the residual $\mathbf{r}^{m+1}$ to be small. In other regions where the solution is rapidly changing, and not well approximated by the coarse grid, the residual will be large. We divide by $\mathbf{N}_{i,i}^{m+1}[\mathbf{I}_m^{m+1}\mathbf{v}^m]$ to normalize the residual.

This error indicator is similar to the error indicator described by Mitchell [9, 10, 11], Rüde [15] or Villegas [21].

6.3. Moving Grids. When modeling non-stationary problems it is advantageous to adjust the shape of the grid to match the movement of the wave front. By noting that we store a sequence of nested grids, not just a single FEM grid, we are able to implement a very cheap de-refinement procedure. We simply shuffle the grids up one level, i.e. set $\bar{\mathcal{M}}^m$ to $\mathcal{M}^{m-1}$ where $\bar{\mathcal{M}}^m$ is the de-refined grid. The interpolation and restriction information at the coarsest and finest grids has to be updated, but otherwise this is a simple copy. Once all of the levels have been updated, we can apply the refinement algorithm described above.

6.4. Error Indicator (Non-Stationary Problem). The next issue is how to calculate the error indicator when taking the time derivative into account. To calculate the indicator at the next time-step, we need an approximation of the solution at that time-step. Applying an implicit algorithm to approximate the solution is cost-prohibitive, so we use an explicit method instead. We only use an explicit method to find the error indicator; once the grid has been refined, we use an implicit method to calculate the solution. Recall that the explicit Euler method is

$$\mathbf{M}^m\bar{\mathbf{v}}^m = \mathbf{M}^m\mathbf{v}^m + \Delta t(\mathbf{b}^m - \mathbf{N}^m[\mathbf{v}^m]),$$

where $\mathbf{M}^m$ is the mass matrix, $\mathbf{v}^m$ is the solution at the current time-step, $\bar{\mathbf{v}}^m$ is the solution at the next time-step and Δt is the step size. Based on this equation we then define the error indicator for non-stationary problems to be:

$$\begin{aligned} e^m &= \frac{\mathbf{r}_i^{m+1}}{\left(\mathbf{M}_{i,i}^{m+1}\right)^{1/4}}, \\ \mathbf{r}^{m+1} &= \mathbf{g}^{m+1} - \mathbf{M}^{m+1}\mathbf{I}_m^{m+1}\bar{\mathbf{v}}^m, \\ \mathbf{g}^{m+1} &= \mathbf{I}_m^{m+1}\left(\mathbf{M}^m\mathbf{v}^m + \Delta t(\mathbf{b}^m - \mathbf{N}^m[\mathbf{v}^m])\right) \end{aligned} \tag{6.2}$$

Note that this indicator only needs to evaluate $\mathbf{N}^m$ once on the original de-refined grid, to form the right-hand-side, and is thus a lot cheaper than the indicator used for the stationary problem.

ETNA
Kent State University
etna@mcs.kent.edu

Figures 6.6, 6.7 and 6.8 show examples of the moving grid, if $\alpha = -3$, $\beta = 3/4$. The values of Z are as given in Figure 2.1 and $D(E) = D_1(E)$. The step size is 0.5/1024, and the figures show the results at time step 110, 220 and 440 respectively. The indicator should pick up the regions where the solution is changing rapidly, which it does.

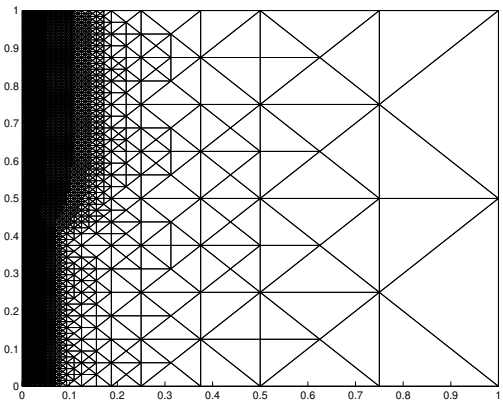
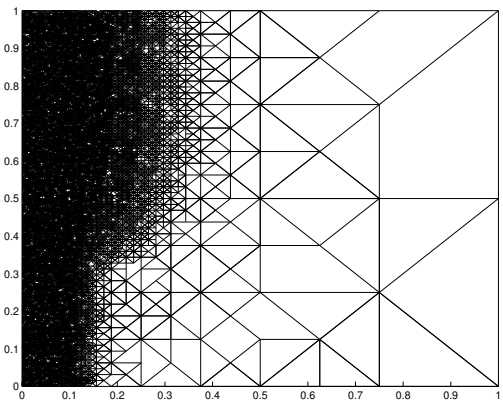
FIG. 6.6. *Example grid at time step* $110\Delta t$ *(*$\Delta t = 0.5/1024$*), which contains 5457 nodes. This grid was used to obtain the results given in Figure 7.6.*