Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 190 (2003) 42–51
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

New physics-based preconditioning of implicit methods for non-equilibrium radiation diffusion

V.A. Mousseau*, D.A. Knoll

Los Alamos National Laboratory, M.S. B216, Los Alamos, NM 87545, USA

Received 15 August 2002; received in revised form 21 February 2003; accepted 30 April 2003

Abstract

This note presents an extension of previous work on physics-based preconditioning of the non-equilibrium radiation diffusion equations. The new physics-based preconditioner presented in this manuscript is a minor modification to the operator-split preconditioner presented previously. Results show that the new preconditioner is more effective on test problems that are more nonlinear.

Keywords: Radiation diffusion; Preconditioning; Implicit; Nonlinear

In this note we continue our investigation of physics-based preconditioning of radiation diffusion [1]. Recent results show the importance of an implicit and nonlinearly consistent solution method [2,3] and the importance of preconditioning [4]. In this study we demonstrate that slight modifications to the base operator split preconditioner, presented in [1], significantly enhance its performance on a challenging problem.

The physics model is the same as in [1]. The partial differential equations are:

$$\frac{\partial E}{\partial t} - \nabla \cdot (D_{\mathrm{r}} \nabla E) = \sigma_{\mathrm{a}}(T^4 - E), \tag{1}$$

$$\frac{\partial T}{\partial t} - \nabla \cdot (D_{\mathrm{T}} \nabla T) = -\sigma_{\mathrm{a}}(T^4 - E). \tag{2}$$

Here, E is radiation energy and T is material temperature. The photon absorption cross-section, σ_{a}, is defined by

$$\sigma_{\mathrm{a}} = \frac{z^3}{T^3}, \tag{3}$$

* Corresponding author. Tel.: +1-505-665-5891; Fax: +1-505-665-5926.
E-mail addresses: vmss@lanl.gov (V.A. Mousseau), nol@lanl.gov (D.A. Knoll).

0021-9991/$ - see front matter
doi:10.1016/S0021-9991(03)00252-3

where z is the atomic mass number. The radiation diffusion coefficient, D_r, is given by

$$D_r(T,E) = \frac{1}{(3\sigma_a + (|\nabla E|/E))}. \tag{4}$$

The flux limiter, $|\nabla E|/E$, has been modified as described in [5]. Now ∇E is computed at the cell centers and is then averaged to the cell face. This new flux limiter increases the size of the discrete stencil and improves symmetry preservation. The material conduction coefficient, D_T, has the following form:

$$D_T(T) = 1.0 \times 10^{-2} T^{5/2}. \tag{5}$$

The nonlinear system of equations represented by Eqs. (1) and (2) are solved using the Jacobian-Free Newton–Krylov method (see [1, Section 1.1, p. 744]). The Krylov solver used to solve the linear systems was a non-restarted version of Saad's GMRES [6]. The linear system solved by Newton's method is

$$\mathbf{J}\begin{bmatrix}\delta E \\ \delta T\end{bmatrix} = \begin{bmatrix}-R_E(E,T) \\ -R_T(E,T)\end{bmatrix}, \tag{6}$$

where $\mathbf{J}$ is the Jacobian matrix and $[\delta E, \delta T]^{\mathrm{T}}$ is the vector of unknowns, and $[-R_E(E,T), -R_T(E,T)]^{\mathrm{T}}$ is the negative of the residual vector.

The residuals in semi-discrete (discrete only in time) form are:

$$R_E(E^{n+1}, T^{n+1}) = \frac{E^{n+1} - E^n}{\Delta t} - \nabla \cdot (D_r^{n+1} \nabla E^{n+1}) - \sigma_a^{n+1}[(T^{n+1})^4 - E^{n+1}], \tag{7}$$

$$R_T(E^{n+1}, T^{n+1}) = \frac{T^{n+1} - T^n}{\Delta t} - \nabla \cdot (D_T^{n+1} \nabla T^{n+1}) + \sigma_a^{n+1}[(T^{n+1})^4 - E^{n+1}]. \tag{8}$$

To develop our time split algorithm we will first linearize the transport coefficients at old time, and then linearize $(T^{n+1})^4$ as $(T^n)^3 T^{n+1}$. Making these changes in Eqs. (7) and (8) results in

$$\frac{E^{n+1} - E^n}{\Delta t} - \nabla \cdot (D_r^n \nabla E^{n+1}) - \sigma_a^n[(T^n)^3 T^{n+1} - E^{n+1}] = 0, \tag{9}$$

$$\frac{T^{n+1} - T^n}{\Delta t} - \nabla \cdot (D_T^n \nabla T^{n+1}) + \sigma_a^n[(T^n)^3 T^{n+1} - E^{n+1}] = 0. \tag{10}$$

We now define $\delta E = E^{n+1} - E^n$ and $\delta T = T^{n+1} - T^n$ and substitute these definitions into Eqs. (9) and (10) which gives

$$\frac{\delta E}{\Delta t} - \nabla \cdot (D_r^n \nabla \delta E) - \sigma_a^n[(T^n)^3 \delta T - \delta E] = -\Big[- \nabla \cdot (D_r^n \nabla E^n) - \sigma_a^n[(T^n)^4 - E^n]\Big] = -R_E(E^n, T^n), \tag{11}$$

$$\frac{\delta T}{\Delta t} - \nabla \cdot (D_T^n \nabla \delta T) + \sigma_a^n[(T^n)^3 \delta T - \delta E] = -\Big[- \nabla \cdot (D_T^n \nabla T^n) + \sigma_a^n[(T^n)^4 - E^n]\Big] = -R_T(E^n, T^n). \tag{12}$$

The second equal sign in Eqs. (11) and (12) result from recalling the definition of R_E and R_E in Eqs. (7) and (8) and noting that the first right-hand side values are the residuals evaluated at old time level n. If we define E^* and T^* to be intermediate time levels between E^n and E^{n+1}, and T^n and T^{n+1} and also define $\delta E^* = E^* - E^n$ and $\delta T^* = T^* - T^n$, we can use operator splitting to break Eqs. (11) and (12) into the following four equations:

$$\frac{\delta E^*}{\Delta t} - \nabla \cdot (D_r^n \nabla \delta E^*) = -R_E(E^n, T^n), \tag{13}$$

$$\frac{\delta T^*}{\Delta t} - \nabla \cdot (D_{\mathrm{T}}^n \nabla \delta T^*) = -R_T(E^n, T^n), \tag{14}$$

$$\frac{\delta E}{\Delta t} - \sigma_{\mathrm{a}}^n[(T^n)^3 \delta T - \delta E] = \frac{\delta E^*}{\Delta t}, \tag{15}$$

$$\frac{\delta T}{\Delta t} + \sigma_{\mathrm{a}}^n[(T^n)^3 \delta T - \delta E] = \frac{\delta T^*}{\Delta t}. \tag{16}$$

To advance the operator split (OS) solution from time level n to time level $n+1$, Eqs. (13) and (14) are solved for δE^* and δT^* with a simple multigrid linear solver which employs a piecewise constant restriction and prolongation operator [1]. Then δE and δT are computed by inverting the block diagonal 2×2 matrix which comes from Eqs. (15) and (16). The final step is to add δE and δT to E^n and T^n to get the new time values. To use this algorithm as a preconditioner one simply replaces the vector $[-R_E(E^n, T^n), -R_T(E^n, T^n)]^{\mathrm{T}}$, which is the right-hand side of Eqs. (13) and (14), with the vector to be preconditioned and the vector, $[\delta E, \delta T]^{\mathrm{T}}$, is the vector which represents the preconditioner times the vector to be preconditioned. This is the original preconditioner presented in [1].

There are two modifications made to this operator split (OS) preconditioner which yield the new preconditioner presented in this manuscript. The first change is to implement a defect correction step after the two diffusion operators (Eqs. (13) and (14)) are solved. The motivation for this is to divide the preconditioner into two steps, diffusion then reaction. After the diffusion step is completed, a new linear residual is computed. This new linear residual is now used as the right-hand side of the reaction part of the operator (Eqs. (15) and (16)). Therefore, some of the error related to diffusion has been relaxed from the equations before the reaction step takes place.

The second modification to the preconditioner makes it a nonlinear preconditioner. The basic idea is to replace the linear block 2×2 (Eqs. (15) and (16)) with a block Jacobi iteration which includes more of the nonlinearity in Eqs. (7) and (8). The motivation for this change is to allow the preconditioner to change between Newton iterations rather than changing only once per time step, such as in the linear operator split preconditioner of [1] represented by Eqs. (13)–(16). For a test problem where a radiation front interacts with a rapid change in photon absorption cross-section, σ_{a} (which is represented by the test problem of this manuscript) it is possible that the part of the Jacobian matrix which represents the physics of this interaction may change significantly from one Newton iteration to the next. Therefore for this type of problem a preconditioner which changes on each Newton iteration may be advantageous.

To implement the defect correction change, a new linear residual vector $[RL_E(E^*, T^*), RL_T(E^*, T^*)]^{\mathrm{T}}$ is computed after Eqs. (13) and (14) are solved. This vector is computed by subtracting the Jacobian matrix, $\mathbf{J}$, times the solution vector, $[\delta E^*, \delta T^*]^{\mathrm{T}}$, from the initial right-hand side vector $[-R_E(E^n, T^n), -R_T(E^n, T^n)]^{\mathrm{T}}$ of Eqs. (13) and (14). In equation form this is,

$$\begin{bmatrix} RL_E(E^*, T^*) \\ RL_T(E^*, T^*) \end{bmatrix} = \mathbf{J} \begin{bmatrix} \delta E^* \\ \delta T^* \end{bmatrix} - \begin{bmatrix} -R_E(E^n, T^n) \\ -R_T(E^n, T^n) \end{bmatrix}. \tag{17}$$

Here the Jacobian matrix, $\mathbf{J}$, times the vector $[\delta E^*, \delta T^*]^{\mathrm{T}}$ is approximated by

$$\mathbf{J} \begin{bmatrix} \delta E^* \\ \delta T^* \end{bmatrix} \approx \begin{bmatrix} \frac{R_E(E^* + \epsilon \delta E^*, T^* + \epsilon \delta T^*) - R_E(E^*, T^*)}{\epsilon} \\ \frac{R_T(E^* + \epsilon \delta E^*, T^* + \epsilon \delta T^*) - R_T(E^*, T^*)}{\epsilon} \end{bmatrix}, \tag{18}$$

which is the standard Jacobian-free approximation to the action of the Jacobian matrix [7] (ϵ is set according to Eq. (8) of [1, p. 745]). The new linear residual will be used in the second operator split step so this new preconditioner is a defect correction method.

The second change replaces the block 2×2 solution of Eqs. (15) and (16) with a "Matrix-Lite" [8] (block Jacobi) iteration of the complete Jacobian matrix. In the operator split preconditioner there were two time steps, the diffusion time step went from time level n to the intermediate time level "*" and then the reaction time step went from time level n to time level $n+1$ including the "*" level information. Because of the defect correction, the new preconditioner will now advance diffusion from time level n to time level "*", and then the "Matrix Lite" will advance both reaction and diffusion from time level "*" to time level $n+1$. Because of this new approach we need to define $\delta E^{**} = E^{n+1} - E^{*}$ and $\delta T^{**} = T^{n+1} - T^{*}$. Using these new definitions, the "Matrix-Lite" iteration approximates the solution to the Jacobian matrix, $\mathbf{J}$, times the vector $[\delta E^{**}, \delta T^{**}]^{\mathrm{T}}$, equals the new right-hand side which is the new linear residual,

$$\mathbf{J}\begin{bmatrix}\delta E^{**}\\ \delta T^{**}\end{bmatrix} = \begin{bmatrix}-RL_E(E^{*}, T^{*})\\ -RL_T(E^{*}, T^{*})\end{bmatrix}. \tag{19}$$

To solve Eq. (19) for δE^{**} and δT^{**}, we will use a "Matrix-Lite" iteration. The first step in constructing the "Matrix-Lite" iteration is to partition the Jacobian matrix into two components, the diagonal 2×2 blocks $\mathbf{D}$, and the off diagonal blocks $(\mathbf{L} + \mathbf{U})$.

$$\mathbf{J} = \mathbf{D} + (\mathbf{L} + \mathbf{U}). \tag{20}$$

The multiplication of the off-diagonal blocks $(\mathbf{L} + \mathbf{U})$ times a vector δ^{**} can be calculated from Eq. (20) by

$$(\mathbf{L} + \mathbf{U})\delta^{**} = \mathbf{J}\delta^{**} - \mathbf{D}\delta^{**}. \tag{21}$$

The computation of the Jacobian matrix, $\mathbf{J}$, times the vector δ^{**} is computed using the same Jacobian-free approximation shown in Eq. (18). Therefore the block Jacobi iteration can be used without ever having to construct the off diagonal blocks $(\mathbf{L} + \mathbf{U})$. Only the 2×2 diagonal blocks, $\mathbf{D}$, of the Jacobian matrix need to be computed and stored. The elements of the 2×2 diagonal blocks are computed using finite differences and Eqs. (7) and (8). Since the block 2×2 diagonal matrix needs to be stored, the algorithm is referred to as "Matrix-Lite" not matrix-free. Therefore, the "Matrix-Lite" iteration to solve $\mathbf{J}\delta = -\mathrm{res}$ can be written as

$$\delta^{k+1} = \mathbf{D}^{-1}(-\mathrm{res} - \mathbf{J}\delta^{k} + \mathbf{D}\delta^{k}), \tag{22}$$

where the superscript k is the "Matrix-Lite" iteration index and $\delta^{0} = 0$.

Therefore the new Matrix-Lite defect correction (MLDC) solution can be represented in equation form as

$$\frac{\delta E^{*}}{\Delta t} - \nabla \cdot (D_{\mathrm{r}}^{n} \nabla \delta E^{*}) = -R_E(E^{n}, T^{n}), \tag{23}$$

$$\frac{\delta T^{*}}{\Delta t} - \nabla \cdot (D_{\mathrm{T}}^{n} \nabla \delta T^{*}) = -R_T(E^{n}, T^{n}), \tag{24}$$

$$\begin{bmatrix}RL_E(E^{*}, T^{*})\\ RL_T(E^{*}, T^{*})\end{bmatrix} = \mathbf{J}\begin{bmatrix}\delta E^{*}\\ \delta T^{*}\end{bmatrix} - \begin{bmatrix}-R_E(E^{n}, T^{n})\\ -R_T(E^{n}, T^{n})\end{bmatrix}, \tag{25}$$

$$\mathbf{J}\begin{bmatrix}\delta E^{**}\\ \delta T^{**}\end{bmatrix} = \begin{bmatrix}-RL_E(E^{*}, T^{*})\\ -RL_T(E^{*}, T^{*})\end{bmatrix}, \tag{26}$$

$$\delta E = \delta E^{*} + \delta E^{**}, \tag{27}$$

$$\delta T = \delta T^{*} + \delta T^{**}. \tag{28}$$

In the first step of the new Matrix-Lite defect correction (MLDC) solution (23) and (24) are solved for δE^* and δT^* using the same simple multigrid algorithm used to solve Eqs. (13) and (14) in the operator split preconditioner. The additional work required by the new preconditioner comes from the fact that Eqs. (25) and (26) change with every Newton iteration. In Eq. (25) the Jacobian-free approximation is used to compute the new linear residuals, $RL_E(E^*, T^*)$ and $RL_T(E^*, T^*)$. Using the new linear residuals computed in Eq. (25), Eq. (26) is now solved using a "Matrix-Lite" iteration for the variables δE^{**} and δT^{**}. The change from time level n to time level $n+1$ is computed from Eqs. (27) and (28). If the MLDC algorithm is used as a preconditioner, one simply replaces the vector $[-R_E(E^n, T^n), -R_T(E^n, T^n)]^{\mathrm{T}}$ in Eqs. (23)–(25) with the vector to be preconditioned. Again the vector that represents the preconditioner times the vector to be preconditioned is $[\delta E, \delta T]^{\mathrm{T}}$. Note δE and δT are computed from Eqs. (27) and (28).

The residual function evaluations, Eqs. (7) and (8), required by the Jacobian-free approximation, which is employed both in computing the new linear residual in Eq. (25) and in the "Matrix-Lite" iteration, Eq. (22) make the new preconditioner very expensive relative to the original operator split preconditioner.

The justification for this additional expense required by the Matrix-Lite defect correction (MLDC) preconditioner is based on the nonlinearity of the problem being solved. If the problem is highly nonlinear then the Jacobian matrix will change from the first to the last Newton iteration in a time step. This change in the Jacobian matrix manifests itself as a loss in effectiveness of the preconditioner. In the early Newton iterations, a Picard linearized preconditioner will still be close to the Jacobian matrix and the preconditioner will work well. However, as the Newton iterations continue and the Jacobian matrix changes, a preconditioner based on a Picard linearization may actually be worse than no preconditioning. In contrast, if one has a preconditioner which accurately matches the current Jacobian matrix, this loss of effectiveness will not occur.

In the results that follow these points will demonstrated. In a problem that is more nonlinear the expense of the new Matrix-Lite defect correction preconditioner is compensated by the fact that it is still very effective at large time steps. The results also demonstrate that there are two ways to lower the nonlinearity of a problem. The first is to simply lower the time step, and the second is to lower the nonlinearity of the physics being simulated. Note, all results presented in this study are on a 64×64 grid.

A new test problem is employed in this study which is different than the one used in [1]. The new test problem has more of a two-dimensional effect and is more difficult to solve. The reason for this difficulty is the larger range in the dynamical time scale of the problem. The new problem is a thermal blast wave similar in nature to the third problem considered in [2]. In the early part of the blast the dynamical time scale is very fast but as the energy spreads out from the blast the dynamical time scale becomes slower. Because of this wide range of time scales, a time step control algorithm [9] has been employed.

The test problem consists of a unit square with insulated boundaries for both radiation energy (E) and material temperature (T). The domain has a constant $z = 1$ with two regions of $z = z_{\mathrm{high}}$ defined by

$$\left(\frac{3}{16} < x < \frac{7}{16},\ \frac{9}{16} < y < \frac{13}{16}\right) \text{ and } \left(\frac{9}{16} < x < \frac{13}{16},\ \frac{3}{16} < y < \frac{7}{16}\right).$$

The initial energy is given by

$$E(r) = 0.001 + E_{\mathrm{amp}} \exp\left[-\left(\frac{r}{0.1}\right)^2\right], \tag{29}$$

where r is the distance from the lower left corner. Note z_{high} and E_{amp} are defined by the problem description. For this test problem the material temperature is initially in equilibrium ($T = E^{1/4}$). The transient

starts with $\Delta t = 5.0 \times 10^{-4}$ and Δt is allowed to grow 25% per time step until it reaches the set radiation diffusion CFL limit [9] which is computed by

$$\Delta t^{n+1} = \left\{ \mathrm{CFL} \times \sqrt{\Delta x^2 + \Delta y^2} \times \sum_{i=2,j=2}^{63,63} \left| \nabla E_{i,j}^n \right| \right\} \Bigg/ \left\{ \sum_{i=2,j=2}^{63,63} \frac{\left| E_{i,j}^n - E_{i,j}^{n-1} \right|}{\Delta t^n} \right\}. \tag{30}$$

Note that the boundary cells ($i = 1$, $j = 1$, $i = 64$, and $j = 64$) have been omitted from the computation of the time step. The aggressive time step growth of 25% is chosen to force the time step to reach the radiation CFL limit quickly. This rapid time step growth was chosen to demonstrate the robustness of the preconditioner. A less aggressive time step growth would have been chosen if the goal had been to demonstrate time convergence of the solution algorithm. It should be noted that this 25% growth rate only controls the time step size as it ramps up from the initial value to the fixed CFL value and it does not effect the time step size in the rest of the transient.

The initial temperature (radiation and material) as well as the high z obstacles are shown in Fig. 1 for the more nonlinear problem ($E_{\mathrm{amp}} = 100$ and $z_{\mathrm{high}} = 10$). The same information is presented in Fig. 2 for the less nonlinear problem ($E_{\mathrm{amp}} = 25$ and $z_{\mathrm{high}} = 2.5$). By comparing Figs. 1 and 2, one can see that the initial energy pulse is both higher and steeper in the more nonlinear test problem. In Fig. 3 contours of radiation temperature ($T_{\mathrm{r}} = E^{1/4}$) are presented for the more nonlinear problem at its final time of $t = 3$ for both the CFL = 0.1 case of Table 1 (thin solid contour lines) and the CFL = 0.02 case of Table 2 (thick dashed contour lines). One can see that these two solutions are very close and quantitatively, the average relative error between the two solutions is 0.35%. Fig. 4 shows the radiation temperature at $t = 20$ for the less nonlinear problem. Two observations can be made about the comparison of Figs. 3 and 4. First, the

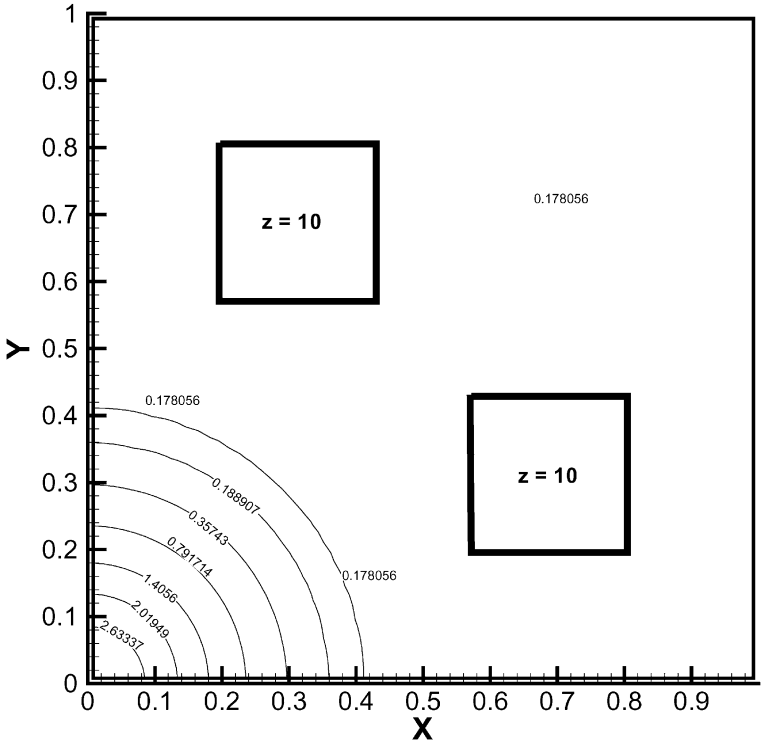

Fig. 1. More nonlinear test problem initial conditions.

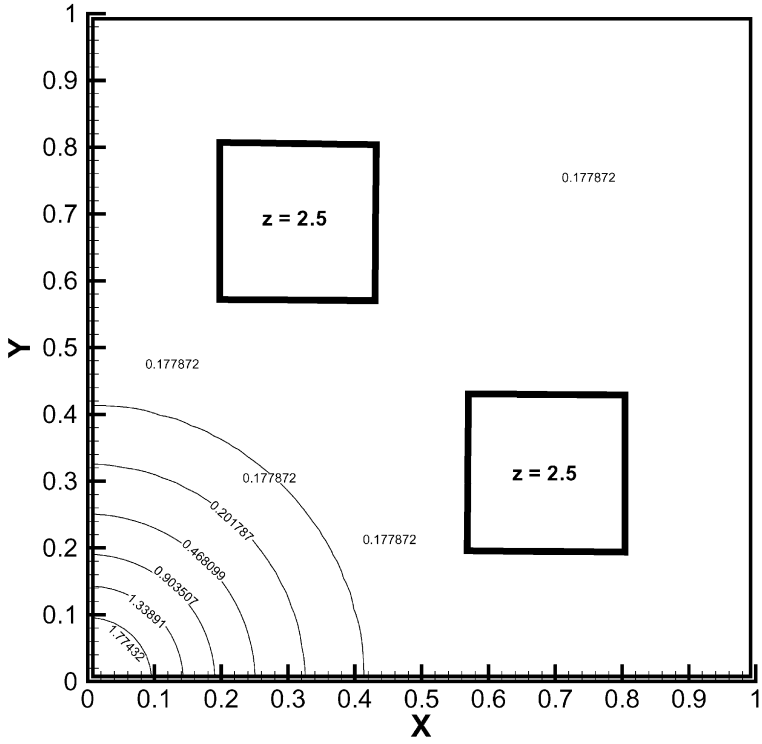

Fig. 2. Less nonlinear test problem initial conditions.

gradients of radiation temperature are much steeper in Fig. 3 than they are in Fig. 4. Second, in the less nonlinear problem (Fig. 4) the obstacles have little impact on the solution.

In Table 1, results are presented for the more nonlinear problem. The nonlinearity comes from the high energy pulse (see Eq. (29) when $E_{\text{amp}} = 100$) and from the effect of a high ratio in z values ($z_{\text{high}} = 10$) on the photon absorption cross-section (see Eq. (3)). These rapid changes in T and z cause very rapid changes in D_{r}, D_{T}, and σ_{a} when the time step is chosen to be roughly equal to the time scale that the temperature front is moving (CFL = 0.1). In Table 1, one can see that the single pass "MLDC 1" preconditioner is very effective, averaging less than four GMRES iterations per Newton iteration, and only requires the total storage of five Krylov vectors. We can also see here that the additional cost of the second Matrix-Lite iteration ("MLDC 2") did not reduce the amount of work or storage significantly and the CPU time is higher (note that all of the CPU times have been normalized to emphasize the differences in timing). The Operator Split preconditioner failed to converge in 500 GMRES iterations which was worse than no preconditioning at all which averaged 60 GMRES iterations per Newton and required the storage of 162 Krylov vectors. These results show that the old operator split preconditioner actually increased the amount of work for this highly nonlinear problem.

Table 2 presents results for the same test problem as Table 1 except the time step is made smaller (CFL = 0.02). In Table 1, where the CFL number is 0.1, the thermal front crosses a cell in about 10 time steps. In Table 2, where the CFL number is 0.02, it take approximately 50 time step for the thermal front to cross a cell. Therefore, the changes in D_{r}, D_{T}, and σ_{a} are not as fast as they were in Table 1. At this level of nonlinearity the old operator split (OS) preconditioner is competitive with the new MLDC preconditioner although it requires significantly more memory (114 Krylov vectors verses four Krylov vectors for MLDC). It should also be noted that even though the average GMRES iterations per Newton iteration is smaller for

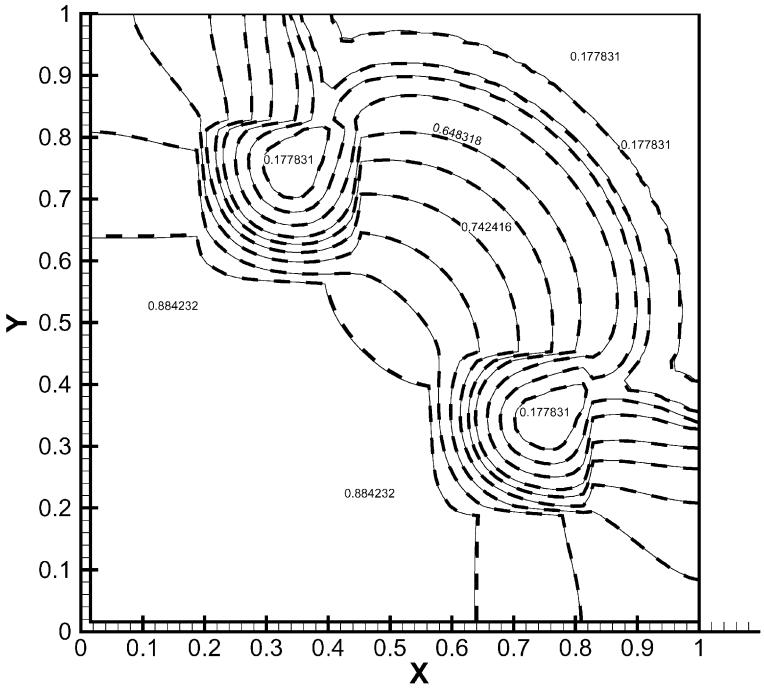

Fig. 3. More nonlinear test problem final radiation temperature: thin solid CFL = 0.1, thick dashed CFL = 0.02.

Table 1
$E_{amp} = 100$, $z_{high} = 10$, CFL = 0.1, and $t_{final} = 3$

Preconditioner	Average (GMRES/Newton)	Maximum GMRES	CPU
MLDC 2	2.67	4	1.09
MLDC 1	3.69	5	1.00
OS	–	500+	–
None	60.15	162	4.05

Table 2
$E_{amp} = 100$, $z_{high} = 10$, CFL = 0.02, and $t_{final} = 3$

Preconditioner	Average (GMRES/Newton)	Maximum GMRES	CPU
MLDC 2	1.80	3	3.97
MLDC 1	2.40	4	3.50
OS	8.11	114	3.60
None	20.91	75	5.27

the operator split preconditioner than no preconditioning (8 vs. 21), the maximum number of Krylov vectors is larger for the operator split than for no preconditioning (114 vs. 75). This indicates that even at this time step the operator split preconditioner may be doing more harm than good on some time steps. Even though the operator split preconditioner is competitive with the MLDC preconditioner for this time step (3.60 vs. 3.50), one has to recall that the physical transient in Tables 1 and 2 are the same. Therefore the "MLDC 1" preconditioner allowed the Newton–Krylov iteration to reach the final time of the transient (in

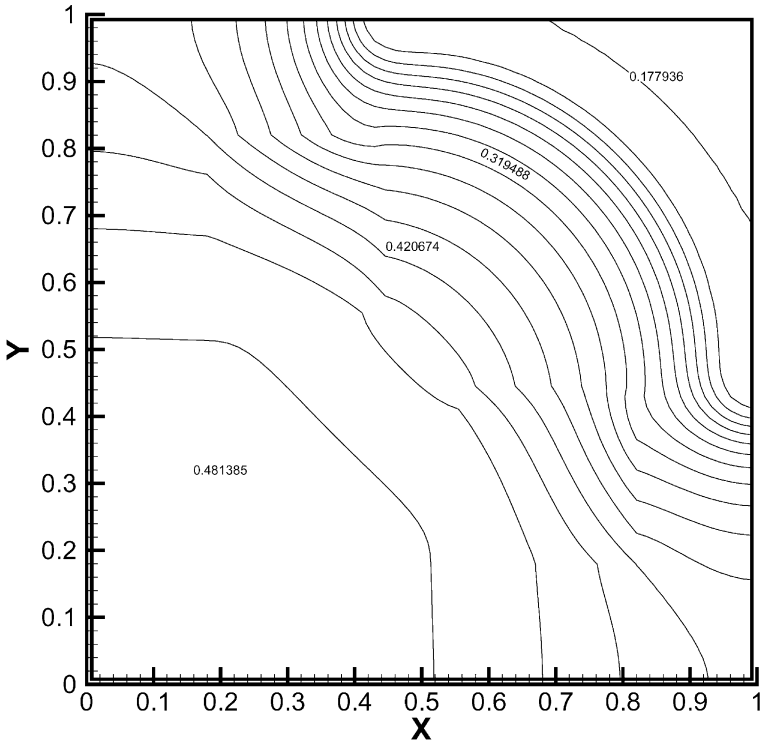

Fig. 4. Less nonlinear test problem final radiation temperature.

Table 1) 3.6 times faster than the operator split preconditioner (in Table 2) by allowing the solution algorithm to work efficiently at a much larger time step.

Table 3 shows results for a test problem that is significantly less nonlinear than the test problem presented in Tables 1 and 2 (note that the initial energy pulse and the ratio of z values have both been lowered by a factor of four). This table presents a simulation that is closer to the ones used in the previous work [1]. One can see that for this test problem the operator split preconditioner is significantly better than no preconditioning and it is competitive in CPU time to the MLDC preconditioner. It should be noted however that the MLDC preconditioner requires less Krylov vectors to be stored. It is interesting to see that at these physical conditions the operator split preconditioner can run at large time steps (CFL = 0.2) which allow the thermal front to cross a cell in five time steps. The reason for this is that with a smaller front (since E_{amp} is four times smaller) and with smaller z ratio between the objects (since z_{high} is four times smaller) the changes in D_{r}, D_{T}, and σ_{a} are much smaller.

Table 3
$E_{\text{amp}} = 25$, $z_{\text{high}} = 2.5$, CFL = 0.2, and $t_{\text{final}} = 20$

Preconditioner	Average (GMRES/Newton)	Maximum GMRES	CPU
MLDC 2	3.30	8	0.76
MLDC 1	4.49	10	0.69
OS	11.61	40	0.69
None	34.46	63	1.33

In this study it has been shown that with minor modifications to an operator split preconditioner, a new preconditioner can be constructed which is significantly more effective at lowering the number of iterations required by the Krylov solver. It should be noted that all of these modifications are readily available when a Jacobian-free Newton–Krylov solution method is employed. In addition, it has been shown for problems that are very nonlinear this preconditioner may be more efficient at lowering the CPU time than a operator split preconditioner. It should be noted that for all of the test problems the new preconditioner had the smallest maximum number of GMRES iterations. For a non-restarted version of GMRES, this maximum number determines the memory usage. Therefore, this minimization of storage may be reason enough to investigate this type of preconditioner.

Acknowledgements

This work was supported under the auspices of the US Department of Energy under DOE contract W-7405-ENG-36 at Los Alamos National Laboratory. LA-UR-02-172.

References

[1] V.A. Mousseau, D.A. Knoll, W.J. Rider, Physics-based preconditioning and the Newton–Krylov method for non-equilibrium radiation diffusion, J. Comput. Phys. 160 (2000) 743–765.
[2] D.A. Knoll, W.J. Rider, G.L. Olson, Nonlinear convergence, accuracy, and time step control in non-equilibrium radiation diffusion, J. Quant. Spectrosc. Radiat. Transfer 70 (2001) 25–36.
[3] D.A. Knoll, L. Chacon, L.G. Margolin, V.A. Mousseau, Nonlinearly consistent approximations for multiple time scale problems, J. Comput. Phys. 185 (2003) 583–611.
[4] P.N. Brown, C.S. Woodward, Preconditioning strategies for fully implicit radiation diffusion with material-energy transfer, SIAM. J. Sci. Comput. 23 (2001) 499–516.
[5] G.L. Olson, J.E. Morel, Solution of the radiation diffusion equation on an AMR Eulerian mesh with material interfaces, LANL Report LA-UR-99-2949, 1999.
[6] Y. Saad, M. Schultz, GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems, SIAM. J. Sci. Stat. Comput. 7 (1986) 856–869.
[7] P.N. Brown, A.C. Hindmarsh, Matrix-free methods for stiff systems of ODE's, SIAM. J. Num. Anal. 23 (1986) 610–638.
[8] W.J. Rider, personal communication, 1998.
[9] W.J. Rider, D.A. Knoll, Time step size selection for radiation diffusion calculations, J. Comput. Phys. 152 (1999) 790–795.