Journal of Computational Physics **160,** 743–765 (2000)
doi:10.1006/jcph.2000.6488, available online at http://www.idealibrary.com on IDEAL®

Physics-Based Preconditioning and the Newton–Krylov Method for Non-equilibrium Radiation Diffusion

V. A. Mousseau, D. A. Knoll, and W. J. Rider

Los Alamos National Laboratory, M.S. D413, Los Alamos, New Mexico 87545
E-mail: vmss@lanl.gov, nol@lanl.gov, wjr@lanl.gov

Received August 25, 1999; revised February 28, 2000

An algorithm is presented for the solution of the time dependent reaction-diffusion systems which arise in non-equilibrium radiation diffusion applications. This system of nonlinear equations is solved by coupling three numerical methods, Jacobian-free Newton–Krylov, operator splitting, and multigrid linear solvers. An inexact Newton's method is used to solve the system of nonlinear equations. Since building the Jacobian matrix for problems of interest can be challenging, we employ a Jacobian–free implementation of Newton's method, where the action of the Jacobian matrix on a vector is approximated by a first order Taylor series expansion. *P*reconditioned *g*eneralized *m*inimal *res*idual (PGMRES) is the Krylov method used to solve the linear systems that come from the iterations of Newton's method. The preconditioner in this solution method is constructed using a physics-based divide and conquer approach, often referred to as operator splitting. This solution procedure inverts the scalar elliptic systems that make up the preconditioner using simple multigrid methods. The preconditioner also addresses the strong coupling between equations with local 2×2 block solves. The intra-cell coupling is applied after the inter-cell coupling has already been addressed by the elliptic solves. Results are presented using this solution procedure that demonstrate its efficiency while incurring minimal memory requirements.

Key Words: Jacobian-free Newton Krylov; radiation diffusion; multigrid.

1. INTRODUCTION

A solution technique for multidimensional non-equilibrium radiation diffusion is presented. The proposed algorithm couples three methods together to form one solution method. The three methods are Jacobian-free Newton–Krylov [1], operator splitting [2], and multigrid solvers [3, 4]. Each will be discussed in detail below. Each of these methods has their strengths and weaknesses and this solution method tries to employ the strengths and to counter the weaknesses.

0021-9991/00 $35.00

This research builds upon earlier studies of Newton–Krylov solutions of the radiation diffusion equations [5–8]. Developing nonlinearly convergent time integration methods for non-equilibrium radiation diffusion is a recent endeavor. In [5] a comparison of the Jacobian-free Newton–Krylov method with a Picard nonlinear iteration was done. It was demonstrated that the Jacobian–free Newton–Krylov method had superior convergence properties. In [8] a detailed time step convergence study was done comparing Jacobian–free Newton–Krylov with a method which does not converge the nonlinearity within a time step (the status quo in the application field [9]). The results clearly indicate that converging nonlinearities produced a superior algorithm in terms of efficiency and accuracy. We also mention that in addition to our work on equilibrium radiation diffusion [6], there has been other recent work on equilibrium radiation diffusion using multigrid and converging nonlinearities [25].

Although we focus on radiation diffusion, the proposed solution method should also be applicable to other physical systems where operator splitting is currently used as a solver such as the Navier–Stokes equations [10].

1.1. Jacobian-free Newton–Krylov method. Newton's method is used to solve the coupled system of nonlinear equations. In its exact form, Newton's method provides quadratic convergence; however, due to approximations that are employed by the solution method in this paper, super-linear convergence (nearly quadratic) is realized in practice. Historically, there have been two main obstacles which have prevented people from using Newton's method for large scale multi-physics applications. In the following we will try to show how each obstacle is overcome.

First, an initial guess inside of the radius of convergence is required for Newton's method to converge. For steady state problems obtaining a good initial guess can require a significant investment of work. In transient problems, however, the initial guess is simply the converged solution from the last time step. If Newton's method does not converge, by lowering the time step one can always get the initial guess as close as necessary to the solution at the next time level. Because of this sensitivity to the initial guess Newton's method provides an automatic measurement of the nonlinearity of the problem. In general one would like the Newton iterations to converge in a small number of iterations. If Newton's method is taking a large number of iterations the time step may be too large for accuracy. This automatic error estimate provides one method to control the time step size in a transient problem. Therefore, in transient problems getting an initial guess inside of the radius of convergence of Newton's method is not a significant concern.

The second obstacle to using Newton's method for large scale multi-physics simulations is the formation of the Jacobian matrix. The Jacobian matrix can be a large matrix (rank equals the number of control volumes times the number of equations and the bandwidth can get very large depending on the coupling between variables) which may be difficult to form. Even though forming the Jacobian matrix may be difficult, this approach has been used successfully for large scale multi-physics steady-state simulations [11, 12]. Depending on the amount of coupling between equations it may be very difficult to even determine the structure of the Jacobian matrix. Once you do know the structure, you are then faced with the task of computing the coefficients. To clarify discussion a brief overview of Newton's method will be provided next.

Newton's method solves the nonlinear system of equations,

$$\mathbf{F}(\mathbf{x}) = 0, \tag{1}$$

where $\mathbf{F}$ is the nonlinear residual function (the discretized system of partial differential

equations) and $\mathbf{x}$ is the state vector. To solve this system, Newton's method requires the solution of a series of linear systems of the form

$$\mathbf{J}^k \delta\mathbf{x}^k = -\mathbf{F}(\mathbf{x}^k). \tag{2}$$

Here $\mathbf{J}$ is the Jacobian matrix, $\delta\mathbf{x}^k$ is the update to the state vector for the kth nonlinear iteration, and k is the nonlinear iteration index. Upon the solution of each of these linear problems the nonlinear iteration is advanced by

$$\mathbf{x}^{k+1} = \mathbf{x}^k + d\,\delta\mathbf{x}^k, \tag{3}$$

where d is a damping parameter used to expand the Newton radius of convergence. This iteration is continued until

$$\|\mathbf{F}(\mathbf{x}^k)\|_2 < tol_1, \tag{4}$$

where tol_1 is the nonlinear convergence criteria and in these studies $tol_1 = 1.0 \times 10^{-5}$ unless otherwise noted.

The coefficients of the Jacobian matrix are derivatives of the residual with respect to the dependent variables,

$$J^k_{i,j} = \frac{\partial F_i(\mathbf{x}^k)}{\partial x^k_j}. \tag{5}$$

For large nonlinear multi-physics systems, computing these coefficients can be difficult. These derivatives can be computed analytically, or using a algebraic symbolic manipulator, or numerically. All of these options present challenges.

The solution technique we use provides a way around this problem. In a Krylov linear solver the solution is built from a linear combination of matrix vector products,

$$\delta\mathbf{x}^k = \sum_{j=0}^{l-1} \alpha_j \mathbf{J}^j \mathbf{r}_0, \tag{6}$$

where $\mathbf{r}_0$ is the initial residual to the linear problem, the α_j's are the coefficients constructed by the Krylov method, and l is the number of Krylov iterations. The important thing to note is that the Jacobian matrix itself is never needed for the Krylov solution. The only Jacobian information required in the Krylov solution is the product of the Jacobian matrix and a vector. This Jacobian-matrix-vector product can be approximated using a first order Taylor series expansion [5], which results in the approximation [1, 13]

$$\mathbf{J}\mathbf{v} \approx \frac{\mathbf{F}(\mathbf{x} + \epsilon\mathbf{v}) - \mathbf{F}(\mathbf{x})}{\epsilon}, \tag{7}$$

where $\mathbf{v}$ is some Krylov vector and ϵ is a small perturbation computed from the equation

$$\epsilon = \frac{b \sum_{i=1}^{N} x_i}{N \|\mathbf{v}\|_2}, \tag{8}$$

where $b = 5.0 \times 10^{-8}$ is approximately the square root of machine round-off. The use of Eq. (7) requires a complete nonlinear residual evaluation $\mathbf{F}(\mathbf{x} + \epsilon\mathbf{v})$ on each Krylov iteration

(note, $\mathbf{F}(\mathbf{x})$ is already known). Therefore, it is important to keep the number of Krylov iterations per Newton iteration small for computational efficiency. If the number of Krylov iterations gets large enough, it will be more efficient to build the Jacobian matrix. As will be shown under Results, for the transient radiation diffusion, the number of Krylov iterations per Newton iteration is approximately ten or less.

The two main historical obstacles to using Newton's method have been addressed by the proposed algorithm. To get an initial guess inside of the radius of convergence, one has to simply lower the size of the time step. Instead of building the $N \times N$ Jacobian matrix (where N is the total number of unknowns) simply approximate its action with N function evaluations using the Jacobian-free approximation. This approximation of the action of the Jacobian matrix must be applied on each Krylov iteration.

Additionally, to improve the efficiency of Newton's method, we use an inexact Newton's method [14]. When the Newton iteration is "far" from convergence (i.e., the residual is "big") there is no reason to spend a large amount of computer time solving the linear system accurately. However, when the Newton iteration is "close" (i.e., the residual is "small") the convergence rate of Newton's method is tightly coupled to the accuracy of the linear solution. To adjust the amount of work done in the linear solve (convergence tolerance) we employ an inexact Newton's method. In the inexact Newton's approach the convergence criteria for the linear solver is proportional to the residual in the nonlinear iteration. In equation form this is

$$\|\mathbf{J}^k \boldsymbol{\delta}\mathbf{x}^k + \mathbf{F}(\mathbf{x}^k)\| < tol_2 \|\mathbf{F}(\mathbf{x}^k)\|, \tag{9}$$

where $tol_2 = 1.0 \times 10^{-2}$ is the value used in this study unless otherwise noted.

The Krylov solver used in this solution technique is the *p*reconditioned *g*eneralized *m*inimal *res*idual (PGMRES) method [15]. PGMRES is a non-symmetric solver whose convergence rate depends on the eigenvalues of the matrix not the matrix structure. Since we never form the Jacobian matrix, we cannot employ a solver which depends on a matrix structure that we may not know. Another advantage of PGMRES is that its residuals are monotonically decreasing. This means that on each iteration the error gets smaller (measured in an appropriate norm). A third advantage of PGMRES, which is important to the use of the Jacobian-free approximation, is that it normalizes the size of the Krylov vectors which it uses. Since the error in the Taylor series expansion (Eq. (7)) is proportional to the size of the Krylov vector ($\mathbf{v}$), the Jacobian-free approximation works well with PGMRES where the Krylov vectors are of size unity. Another motivation for keeping the number of PGMRES iterations small is evident in Eq. (6). Since each new Krylov vector is orthogonal to all of the previous Krylov vectors, the amount of work required to find a new vector increases with the number of Krylov iterations.

The drawback to using PGMRES is that it requires the storage of one additional Krylov vector (which is the same size as a state vector) per Krylov iteration. It is this property which enables PGMRES to obtain its monotonic error reduction for non-symmetric systems. The impact on the solution algorithm is that the storage requirement is MN where M is the maximum number of Krylov iterations required to converge.

There is an approach called restarting which tries to keep the amount of storage for PGMRES constant. In this strategy the number of Krylov vectors is fixed and the linear iteration is restarted when the fixed dimension of the Krylov subspace has been reached. This approach minimizes the memory requirements, but does not ensure the monotonically decreasing residuals or even convergence.

If the restarting strategy is not employed, to keep the Krylov vector storage requirement from becoming impractical, one has to keep the number of Krylov iterations low. This is accomplished by preconditioning the system. Preconditioning is a process which approximates the inverse of the Jacobian matrix ($\mathbf{JP}^{-1} \approx \mathbf{I}$). Now instead of solving

$$\mathbf{J}\delta\mathbf{x} = -\mathbf{F}(\mathbf{x}), \tag{10}$$

one solves the system

$$\mathbf{JP}^{-1}\mathbf{P}\delta\mathbf{x} = -\mathbf{F}(\mathbf{x}). \tag{11}$$

Since PGMRES works on the number of unique eigenvalues of the system, if the new system $\mathbf{JP}^{-1}$ has most of it eigenvalues clustered around one, then PGMRES will need very few iterations to reach convergence. Therefore, the problem of using PGMRES, which is the storage of the Krylov vectors, can be minimized by using an effective preconditioner.

1.2. Operator split (fractional time step, alternating block factorization) preconditioning. If one had the Jacobian matrix built, an obvious choice for a preconditioner is an approximate inverse of the Jacobian. Since we do not wish to build the Jacobian matrix, we can be more creative in our construction of the preconditioner. One of the motivations for preconditioning the system is to save memory by reducing the number of Krylov vectors that need to be stored. Therefore, it is important that the preconditioning process itself is not memory intensive.

Before further discussion, one needs to define two terms for clarity. In the following discussions "coupled systems" will refer to matrices which have an order equal to the number of equations per control volume times the number of control volumes. "Scalar equations" will refer to systems which have an order equal to the number of control volumes.

One approach to producing a preconditioner that is not memory intensive is to employ a numerical technique which was developed when computer memories were relatively small and computers were relatively slow. In the early days of computational physics the numerical technique of operator splitting [2] (or fractional time step methods [16] or alternate-block factorization [17]) was the main work horse for the solution of coupled systems. This approach reduces the memory requirements and the computational complexity of solving a system of equations. This is accomplished by solving each scalar equation independently on the entire grid and then coupling these solutions back together to get the solution to the larger coupled system.

For example, if one has a system with n control volumes and m equations per control volume ($N = mn$) the total system size is $mn \times mn$. The storage required for this system is mnL where L is the number of nonzero diagonals for the large $mn \times mn$ system of equations. Assuming a lexicographical ordering of the unknowns, the number of nonzero diagonals is the number of variables touched by the stencil (i.e., approximately equal to the number of variables per cell times the number of cells in the differencing stencil). However, if you solve the systems one equation at a time your storage requirements are nl where $l \leq L$ is the number of nonzero diagonals in each of the m smaller $n \times n$ systems. This memory reduction is realized since the same memory locations can be used to solve each of the m equations. One can see that the memory savings is $m(L/l)$.

Since the amount of work required to solve a system scales greater than linearly with the size of the system, one can clearly see that the work for one large system is proportional to $(mnL)^{\alpha}$ where $\alpha > 1$ is the exponential power for solving systems. The work for solving

the m smaller systems is proportional to $m(nl)^{\alpha}$. This results in a computation complexity reduction of $m^{(\alpha-1)}(L/l)^{\alpha}$. For the one-dimensional studies presented later $m=2$ and $n=50$ or $n=200$.

The exact details of how to construct the operator split preconditioner will be shown in a later section. The basic idea however is to:

1. Employ a simple Picard linearization to linearize the the nonlinear discretized system of partial differential equations.
2. Order the equations so that there is adequate coupling between them.
3. Solve the equations one at a time using the most recent value of variables available.

Physics-based preconditioning is the application of this idea as a preconditioner in a Newton–Krylov method and refers to the splitting of the solution process based on different types of physics (e.g., transport-diffusion physics versus equilibration-reaction physics).

As solution techniques, these *single-step* operator split methods are often robust but are only as accurate as the linearization approximations. When used as a solver, the *single-step* operator split algorithms must take time steps small enough to keep the linearization approximations "accurate." Since the nonlinear residuals $[(\mathbf{F}(\mathbf{x})]$ are never formed, the true accuracy of the solution is never measured as part of the solution procedure.

It should be noted that operator split algorithms can also be applied in an iterative, *multi-step* fashion as a solver or as a multigrid smoother. When employed in this fashion, operator splitting may not be robust since convergence is not guaranteed. As a preconditioner, however, the outer Newton iteration handles the nonlinear error so the operator split algorithm can be used at larger time steps. By using operator splitting as a preconditioner, we take advantage of its reduction in memory requirements and its reduction in computational complexity, but we are not forced to take the small time steps required for accuracy since the Newton iteration significantly improves the accuracy of the problem for a given time step [5]. Therefore, when an operator split algorithm is used as a preconditioner for Newton–Krylov, Newton–Krylov can be considered to be an accelerator of the convergence of the operator split method.

1.3. Multigrid. In the current application, the operator split algorithm produces a set of linear, elliptic, scalar equations that need to be solved. The multigrid technique was initially designed for use on scalar elliptic equations so it is an obvious choice to use as a solver. However, because the system is so small in the one-dimensional problem, a simple symmetric Gauss Seidel iteration is used. For the two-dimensional problems the multigrid method is used as a solver. The algorithmic scaling of the multigrid method makes it an attractive choice for a preconditioner since one can get solutions to large problems quickly.

The multigrid preconditioner used in this study was developed in [18]: The restriction and prolongation operators employed here are piecewise constant and a variational coarse grid operator is used. Although these choices may not be optimal for multigrid as a solver, we have found it to be an acceptable approach when using multigrid as a preconditioner. There are three advantages to choosing piecewise constant prolongation and restriction operators.

1. They are the easiest to implement.
2. They do not have difficulties with irregular meshes.
3. They are simple to implement as a two-level solver in parallel [19] since they minimize communication costs.

The multigrid preconditioner uses a simple "V" cycle and incorporates symmetric Gauss Seidel as a smoother. From our experience, we have not seen the advantage in CPU time

reduction from using more advanced multigrid strategies. Since PGMRES is the linear equation solver and the equations used in the preconditioner are not the same as the equations being solved, it is not clear that there is an advantage to using more accurate linear solvers in the preconditioner. Implementing more sophisticated multigrid strategies and measuring their effects on CPU time and convergence is an area of future research.

In this solution technique, Newton's method handles the nonlinearities and the Krylov solver handles the coupled systems. We are applying the multigrid algorithm in the preconditioner where the problem has already been broken up into linear, elliptic, scalar pieces by operator splitting. Therefore, in this solution technique, multigrid is applied in situations where its algorithm is near optimal.

2. PHYSICS MODEL

The following coupled system for radiation energy, E, and material temperature, T, will be solved using the solution technique of this paper. These equations represent an idealization of non-equilibrium radiation diffusion in a material [2, 5, 6, 9, 20, 21],

radiation diffusion (gray approximation),

$$\frac{\partial E}{\partial t} - \nabla \cdot (D_r \nabla E) = \sigma_a (T^4 - E); \tag{12}$$

material energy balance,

$$\frac{\partial T}{\partial t} - \nabla \cdot (D_t \nabla T) = -\sigma_a (T^4 - E). \tag{13}$$

Here σ_a is the photon absorption cross-section. In thermal equilibrium we have $E = T^4$, and for the non-equilibrium case one can define a radiation temperature as $T_r = (E)^{0.25}$. We will choose

$$\sigma_a = \frac{z^3}{T^3}, \tag{14}$$

where z is the atomic mass number and we use the following form for the radiation diffusion coefficient,

$$D_r(T) = \frac{1}{3\sigma_a}. \tag{15}$$

However, in regions of strong gradients, simple diffusion theory can fail, resulting in a flux of energy moving faster than the speed of light. To prevent this artificial behavior, the diffusion coefficient is augmented in the following heuristic fashion, referred to as flux limiting [2],

$$D_r(T, E) = \frac{1}{(3\sigma_a + (1/E)|\partial E/\partial x|)}. \tag{16}$$

The following form of the material (plasma) conduction diffusion coefficient from Spitzer and Harm [22] is used,

$$D_t(T) = kT^{5/2}, \tag{17}$$

where k is a constant.

3. DEVELOPMENT OF AN OPERATOR SPLIT (PHYSICS BASED) PRECONDITIONER

Since the operator split method is a proven solver, it seems obvious that it might be a good preconditioner. To describe the operator split solution method we will work with the one-dimensional version of the problem,

$$\frac{\partial E}{\partial t} - \frac{\partial}{\partial x}\left(D_r \frac{\partial E}{\partial x}\right) = \sigma_a (T^4 - E) \tag{18}$$

$$\frac{\partial T}{\partial t} - \frac{\partial}{\partial x}\left(D_t \frac{\partial T}{\partial x}\right) = -\sigma_a (T^4 - E). \tag{19}$$

For simplicity we assume constant spacing and define the following dependencies between the diffusion coefficients and the photon absorption cross section,

$$D_{r,i+1/2}^{n+1} = D_r\left(E_{i+1/2}^{n+1}, T_{i+1/2}^{n+1}\right) \tag{20}$$

$$D_{t,i+1/2}^{n+1} = D_t\left(T_{i+1/2}^{n+1}\right) \tag{21}$$

$$\sigma_{a,i}^{n+1} = \sigma_a\left(T_i^{n+1}\right). \tag{22}$$

Here the superscript $n+1$ indicates a new time or implicit value and the subscript i denotes a control volume centered quantity and the subscript $i+1/2$ denotes a control volume face quantity. Then the fully implicit, fully nonlinear, finite difference forms of Eqs. (18) and (19) are

$$\frac{E_i^{n+1} - E_i^n}{\Delta t} - \left[D_{r,i+1/2}^{n+1}\left(\frac{E_{i+1}^{n+1} - E_i^{n+1}}{\Delta x^2}\right) - D_{r,i-1/2}^{n+1}\left(\frac{E_i^{n+1} - E_{i-1}^{n+1}}{\Delta x^2}\right)\right] - \sigma_{a,i}^{n+1}\left[\left(T_i^{n+1}\right)^4 - E_i^{n+1}\right] = 0 \tag{23}$$

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} - \left[D_{t,i+1/2}^{n+1}\left(\frac{T_{i+1}^{n+1} - T_i^{n+1}}{\Delta x^2}\right) - D_{t,i-1/2}^{n+1}\left(\frac{T_i^{n+1} - T_{i-1}^{n+1}}{\Delta x^2}\right)\right] + \sigma_{a,i}^{n+1}\left[\left(T_i^{n+1}\right)^4 - E_i^{n+1}\right] = 0. \tag{24}$$

The first step in our operator split method is to apply a Picard linearization to the nonlinear equations. We linearize Eqs. (23) and (24) by evaluating D_r, D_t, and σ_a at old time values and by linearizing $(T^{n+1})^4$ as $(T^n)^3 T^{n+1}$ to get

$$\frac{E_i^{n+1} - E_i^n}{\Delta t} - \left[D_{r,i+1/2}^{n}\left(\frac{E_{i+1}^{n+1} - E_i^{n+1}}{\Delta x^2}\right) - D_{r,i-1/2}^{n}\left(\frac{E_i^{n+1} - E_{i-1}^{n+1}}{\Delta x^2}\right)\right] - \sigma_{a,i}^{n}\left[\left(T_i^{n}\right)^3 T_i^{n+1} - E_i^{n+1}\right] = 0 \tag{25}$$

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} - \left[D_{t,i+1/2}^{n}\left(\frac{T_{i+1}^{n+1} - T_i^{n+1}}{\Delta x^2}\right) - D_{t,i-1/2}^{n}\left(\frac{T_i^{n+1} - T_{i-1}^{n+1}}{\Delta x^2}\right)\right] + \sigma_{a,i}^{n}\left[\left(T_i^{n}\right)^3 T_i^{n+1} - E_i^{n+1}\right] = 0. \tag{26}$$

These are the linearly implicit, discrete equations. Under Results the operator split preconditioner will be compared to a Picard preconditioner. The Picard preconditioner solves the couple system of Eqs. (25) and (26) with a block symmetric Gauss Seidel iteration. These

equations will next be used to develop the operator splitting approach as a solver and as a preconditioner.

3.1. Operator splitting as a solver. To demonstrate how to use operator splitting as a preconditioner, we will first show how to use it as a solver. This step is done first since many readers have already used operator splitting as a solver. Operator splitting solves the coupled system in a two step process. First the system is linearized using the Picard linearization yielding a coupled linear system. Then this coupled system is broken up into three separate pieces (radiation diffusion, material heat conduction, and energy exchange coupling).

The first step in our physics-based operator split algorithm is to solve the radiation transport (diffusion) physics of Eq. (18),

$$\frac{\partial E}{\partial t} - \frac{\partial}{\partial x}\left(D_r \frac{\partial E}{\partial x}\right) = 0. \tag{27}$$

The resulting finite difference form of Eq. (27) is taken from Eq. (25),

$$\frac{E_i^* - E_i^n}{\Delta t} - \left\{\frac{D_{r,i+1/2}^n \frac{E_{i+1}^* - E_i^*}{\Delta x} - D_{r,i-1/2}^n \frac{E_i^* - E_{i-1}^*}{\Delta x}}{\Delta x}\right\} = 0. \tag{28}$$

Here we have defined an intermediate energy level E^*.

The second step in our physics-based operator split algorithm is to solve the material thermal transport (diffusion) physics of Eq. (19),

$$\frac{\partial T}{\partial t} - \frac{\partial}{\partial x}\left(D_t \frac{\partial T}{\partial x}\right) = 0. \tag{29}$$

The discretized form of Eq. (29) comes from Eq. (26),

$$\frac{T_i^* - T_i^n}{\Delta t} - \left\{\frac{D_{t,i+1/2}^n((T_{i+1}^* - T_i^*)/\Delta x) - D_{t,i-1/2}^n((T_i^* - T_{i-1}^*)/\Delta x)}{\Delta x}\right\} = 0. \tag{30}$$

Here T^* is an intermediate material temperature level.

The third and final step in our physics-based operator split algorithm is to solve the equilibration radiation coupling (reaction) physics from Eqs. (18) and (19),

$$\frac{\partial E}{\partial t} = \sigma_a(T^4 - E) \tag{31}$$

$$\frac{\partial T}{\partial t} = -\sigma_a(T^4 - E). \tag{32}$$

This results in a coupled 2×2 matrix for each control volume. The discretized equations which make up the 2×2 systems are parts of Eqs. (25) and (26). The discrete equations are

$$\frac{E_i^{n+1} - E_i^*}{\Delta t} = \sigma_{a,i}^n\left(\left(T_i^n\right)^3 T_i^{n+1} - E_i^{n+1}\right) \tag{33}$$

$$\frac{T_i^{n+1} - T_i^*}{\Delta t} = -\sigma_{a,i}^n\left(\left(T_i^n\right)^3 T_i^{n+1} - E_i^{n+1}\right). \tag{34}$$

As a solver, our operator split solution algorithm follows:

1. Solve Eq. (28) for $\mathbf{E}^*$.
2. Solve Eq. (30) for $\mathbf{T}^*$.

3. Substitute the values for $\mathbf{E}^*$ and $\mathbf{T}^*$ into Eqs. (33) and (34).
4. Solve the resulting 2×2 system on each control volume to get $\mathbf{E}^{n+1}$ and $\mathbf{T}^{n+1}$.

Next we will express the operator split solution algorithm in matrix notation. This will be necessary when discussing the operator split procedure as a preconditioner. We can rewrite Eq. (28), which is step one in our solution scheme, in matrix form as

$$\mathbf{P_1}\mathbf{E}^* = \frac{\mathbf{E}^n}{\Delta t}. \tag{35}$$

It is important to note that the linear operator $\mathbf{P_1}$ is of order n. Equation (30) (the second step in the solution algorithm) can be expressed in matrix form as

$$\mathbf{P_2}\mathbf{T}^* = \frac{\mathbf{T}^n}{\Delta t}. \tag{36}$$

The same as for $\mathbf{P_1}$ the linear operator $\mathbf{P_2}$ is of order n. Equations (33) and (34) (steps 3 and 4 of the solution scheme) can be represented as the following block 2×2 diagonal matrix equation,

$$\mathbf{P_3}\begin{pmatrix}\mathbf{E}^{n+1}\\ \mathbf{T}^{n+1}\end{pmatrix} = \begin{pmatrix}\frac{\mathbf{E}^*}{\Delta t}\\ \frac{\mathbf{T}^*}{\Delta t}\end{pmatrix}. \tag{37}$$

As opposed to the scalar operators $\mathbf{P_1}$ and $\mathbf{P_2}$ the coupled linear operator $\mathbf{P_3}$ is of order $2n$. The operator $\mathbf{P_3}$ is applied cumulatively to correct the error associated with the strong coupling within a cell which $\mathbf{P_1}$ and $\mathbf{P_2}$ did not address. By solving the $\mathbf{P_1}$ system (Eq. (35)) for $\mathbf{E}^*$ and the $\mathbf{P_2}$ system (Eq. (36)) for $\mathbf{T}^*$ and substituting into the $\mathbf{P_3}$ system (Eq. (37)) we get

$$\begin{bmatrix}\mathbf{P_1} & 0\\ 0 & \mathbf{P_2}\end{bmatrix}\Delta t\mathbf{P_3}\begin{pmatrix}\mathbf{E}^{n+1}\\ \mathbf{T}^{n+1}\end{pmatrix} = \begin{pmatrix}\frac{\mathbf{E}^n}{\Delta t}\\ \frac{\mathbf{T}^n}{\Delta t}\end{pmatrix}. \tag{38}$$

If we define $\mathbf{O}$ by

$$\mathbf{O} = \begin{bmatrix}\mathbf{P_1} & 0\\ 0 & \mathbf{P_2}\end{bmatrix}\Delta t\mathbf{P_3}, \tag{39}$$

we can then write our operator split method in matrix form as

$$\mathbf{O}\begin{pmatrix}\mathbf{E}^{n+1}\\ \mathbf{T}^{n+1}\end{pmatrix} = \begin{pmatrix}\frac{\mathbf{E}^n}{\Delta t}\\ \frac{\mathbf{T}^n}{\Delta t}\end{pmatrix}. \tag{40}$$

3.2. Operator split in incremental form. We will now explain how to get our operator split solution method into the correct form for use as a preconditioner. Recall the base equation for Newton's method,

$$\mathbf{J}^k\delta\mathbf{x}^k = -\mathbf{F}(\mathbf{x}^k). \tag{41}$$

This equation is solved for $\delta\mathbf{x}^k$ and then the solution is advanced by

$$\mathbf{x}^{k+1} = \mathbf{x}^k + d\,\delta\mathbf{x}^k. \tag{42}$$

Therefore, one can see that Newton's method solves for the change in x from nonlinear iteration k to $k+1$. However, when used as a solver, our operator split method solved for the new time value ($\mathbf{x}^{n+1}$), not the change in $\mathbf{x}$ from time level n to $n+1$. Therefore, one has to rework the normal operator split method into incremental form so it can be used as a preconditioner for Newton's method.

The first step in converting our operator split algorithm into incremental form is to define the residual functions for use in Newton's method. From the fully implicit, fully nonlinear, finite difference form (Eqs. (23) and (24)) we can define the residual functions

$$\mathbf{F_E}(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{T}}) = \frac{\mathcal{E}_i - E_i^n}{\Delta t} - \left[D_r\left(\mathcal{E}_{i+1/2}, \mathcal{T}_{i+1/2}\right)\left(\frac{\mathcal{E}_{i+1} - \mathcal{E}_i}{\Delta x^2}\right) - D_r\left(\mathcal{E}_{i-1/2}, \mathcal{T}_{i-1/2}\right)\left(\frac{\mathcal{E}_i - \mathcal{E}_{i-1}}{\Delta x^2}\right)\right] - \sigma_a(\mathcal{T}_i)\left[(\mathcal{T}_i)^4 - \mathcal{E}_i^{n+1}\right] \tag{43}$$

$$\mathbf{F_T}(\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{T}}) = \frac{\mathcal{T}_i - T_i^n}{\Delta t} - \left[D_t\left(\mathcal{T}_{i+1/2}\right)\left(\frac{\mathcal{T}_{i+1} - \mathcal{T}_i}{\Delta x^2}\right) - D_t\left(\mathcal{T}_{i-1/2}\right)\left(\frac{\mathcal{T}_i - \mathcal{T}_{i-1}}{\Delta x^2}\right)\right] + \sigma_a(\mathcal{T}_i)[(\mathcal{T}_i)^4 - \mathcal{E}_i], \tag{44}$$

where $\boldsymbol{\mathcal{E}}$ and $\boldsymbol{\mathcal{T}}$ are dummy radiation energy and dummy material temperature, respectively. It is now important to note that

$$\mathbf{F_E}(\mathbf{E}^n, \mathbf{T}^n) = -\left[D_{r,i+1/2}^n\left(\frac{E_{i+1}^n - E_i^n}{\Delta x^2}\right) - D_{r,i-1/2}^n\left(\frac{E_i^n - E_{i-1}^n}{\Delta x^2}\right)\right] - \sigma_{a,i}^n\left[\left(T_i^n\right)^4 - E_i^n\right] \tag{45}$$

$$\mathbf{F_T}(\mathbf{E}^n, \mathbf{T}^n) = -\left[D_{t,i+1/2}^n\left(\frac{T_{i+1}^n - T_i^n}{\Delta x^2}\right) - D_{t,i-1/2}^n\left(\frac{T_i^n - T_{i-1}^n}{\Delta x^2}\right)\right] + \sigma_{a,i}^n\left[\left(T_i^n\right)^4 - E_i^n\right]. \tag{46}$$

These residuals, $\mathbf{F}_{\mathrm{E}}(\mathbf{E}^n, \mathbf{T}^n)$ and $\mathbf{F_T}(\mathbf{E}^n, \mathbf{T}^n)$, will be important and will come up in the future.

First, we will make the definitions

$$\delta E_i = E_i^{n+1} - E_i^n \tag{47}$$

$$\delta T_i = T_i^{n+1} - T_i^n. \tag{48}$$

Then, recalling the definitions of the residuals (Eqs. (45) and (46)), we can rewrite Eqs. (25) and (26) in incremental form to get

$$\frac{\delta E_i}{\Delta t} - \left[D_{r,i+1/2}^n\left(\frac{\delta E_{i+1} - \delta E_i}{\Delta x^2}\right) - D_{r,i-1/2}^n\left(\frac{\delta E_i - \delta E_{i-1}}{\Delta x^2}\right)\right] - \sigma_{a,i}^n\left[\left(T_i^n\right)^3 \delta T_i - \delta E_i\right] = -\mathbf{F_E}(\mathbf{E}^n, \mathbf{T}^n) \tag{49}$$

$$\frac{\delta T_i}{\Delta t} - \left[D_{t,i+1/2}^n\left(\frac{\delta T_{i+1} - \delta T_i}{\Delta x^2}\right) - D_{t,i-1/2}^n\left(\frac{\delta T_i - \delta T_{i-1}}{\Delta x^2}\right)\right] + \sigma_{a,i}^n\left[\left(T_i^n\right)^3 \delta T_i - \delta E_i\right] = -\mathbf{F_T}(\mathbf{E}^n, \mathbf{T}^n). \tag{50}$$

From the definition of $\mathbf{O}$ in Eq. (39), it is easy to see that Eqs. (49) and (50) can be rewritten in matrix form as

$$\mathbf{O}\begin{pmatrix}\delta\mathbf{E}\\ \delta\mathbf{T}\end{pmatrix} = \begin{pmatrix}-\mathbf{F}_{\mathbf{E}}(\mathbf{E}^n, \mathbf{T}^n)\\ -\mathbf{F}_{\mathbf{T}}(\mathbf{E}^n, \mathbf{T}^n)\end{pmatrix}. \tag{51}$$

Also, it is clear that we now have the operator split preconditioner in the same form as the Newton iteration matrix,

$$\mathbf{J}\delta\mathbf{x} = -\mathbf{F}(\mathbf{x}), \tag{52}$$

which makes it trivial to apply as a preconditioner as is demonstrated below.

3.3. Operator split preconditioner implementation. Now that we have the operator split solver in the same form as the Newton iteration, we can proceed to use the operator split algorithm as a preconditioner. There are slight modifications that need to be implemented to use the algorithm as a preconditioner. First, the right hand side (i.e., the residuals evaluated at old time values) is not constructed. The right hand side is now provided by the Krylov solver and is a Krylov vector instead of a residual vector. Second, instead of using the old time values ($\mathbf{E}^n$ and $\mathbf{T}^n$) to evaluate the matrix elements in $\mathbf{P_1}$, $\mathbf{P_2}$, and $\mathbf{P_3}$, these coefficients are now constructed from the latest nonlinear iterate value ($\mathbf{E}^k$ and $\mathbf{T}^k$). The following describes in detail how the preconditioner is implemented.

Recall the definition of right preconditioning (Eq. (11)), but now replace the generic preconditioner $\mathbf{P}$ with the operator split preconditioner $\mathbf{O}$ to get

$$\mathbf{J}\mathbf{O}^{-1}\mathbf{O}\delta\mathbf{x} = -\mathbf{F}(\mathbf{x}). \tag{53}$$

Define $\mathbf{z} = \mathbf{O}\delta\mathbf{x}$ and we can rewrite Eq. (53) as

$$\mathbf{J}\mathbf{O}^{-1}\mathbf{z} = -\mathbf{F}(\mathbf{x}). \tag{54}$$

To solve the above system, we need to compute the action of the matrix $\mathbf{J}\mathbf{O}^{-1}$ on a Krylov vector $\mathbf{v}$,

$$\mathbf{w} = \mathbf{J}\mathbf{O}^{-1}\mathbf{v}. \tag{55}$$

Computationally this is done in a two step procedure.

1. Solve $\mathbf{O}\mathbf{q} = \mathbf{v}$ for $\mathbf{q}$ (using Eq. (51)).
2. Compute $\mathbf{w} = \mathbf{J}\mathbf{q}$ using the Jacobian-free approximation,

$$\mathbf{J}\mathbf{q} \approx \frac{\mathbf{F}(\mathbf{x} + \epsilon\mathbf{q}) - \mathbf{F}(\mathbf{x})}{\epsilon}. \tag{56}$$

This procedure is used each time the action of the Jacobian matrix is needed. After the Krylov method has converged on a solution for $\mathbf{z}$, the last step is to solve

$$\mathbf{O}\delta\mathbf{x} = \mathbf{z}, \tag{57}$$

for $\delta\mathbf{x}$. Now that we have described how to construct and implement the operator split preconditioner, we will demonstrate its performance.

4. RESULTS

The operator split preconditioner will now be demonstrated on three separate test problems.

1. The one-dimensional problem that was used to demonstrate the construction of the operator split preconditioner. In this section the operator split preconditioner will be compared to the coupled Picard based preconditioner. Results will show that the two approaches behave similarly.

2. A two-dimensional problem with the same one-dimensional physics as the one-dimensional problem. This problem will be used to demonstrate the scalability of the operator split preconditioner. Results will show that the method scales very well with problem size.

3. A two-dimensional problem with two-dimensional physics. This problem was chosen to demonstrate the two-dimensional capability of the solution method. By making the atomic number, z, a function of x and y we were able to create a region with a lower radiation diffusion coefficient. By modifying the ratio of high to low atomic number in the two dimensional "blockage" we were also able to demonstrate how the method behaves when faced with a strong spatial dependence of the radiation diffusion coefficient. Results show that the method works just as well in two dimensions as it did in one dimension. Results also demonstrate that as the discontinuities in the matrix coefficients become large (caused by the blockage), the effectiveness of the preconditioner is negatively impacted.

It is of some interest to compare the solution based on the Picard linearized coupled system to the nonlinear Newton–Krylov approach. In general, as the problem becomes more nonlinear and as the required accuracy is tightened, the Newton–Krylov method outperforms the Picard method. For more detailed results see [5].

4.1. Problem 1: One-dimensional. The model problem considered in this study is taken from [5, 23] and consists of a unit radiation flux impinging on an initially cold slab of unit depth. This results in mixed, or Robin, boundary conditions for the radiation equations at $x = 0$ and $x = 1$. Following [24], at $x = 0$ we use

$$\frac{1}{4}E - \frac{1}{6\sigma_a}\frac{\partial E}{\partial x} = 1, \tag{58}$$

and at $x = 1$ we use

$$\frac{1}{4}E + \frac{1}{6\sigma_a}\frac{\partial E}{\partial x} = 0. \tag{59}$$

For the material diffusion equation the boundary conditions are simply $\frac{\partial T}{\partial x} = 0$ at $x = 0$ and $x = 1$. The initial energy is $E_0 = 1.0 \times 10^{-5}$ and the initial material temperature is $T_0 = (E_0)^{1/4} \approx 5.62 \times 10^{-2}$. The atomic number z is constant and equal to 1.0 everywhere. The simulation runs for three units of time ($time_f = 3.0$).

The first set of results show the effect of the diffusion coefficient on the solution to the problem (see Fig. 1). Here the constant k is varied in Eq. (17) to show the effect that material conduction has on the solution. It is clear that an increase in the material diffusion coefficient causes the thermal front to move faster. Note that the material temperature never exceeds the radiation temperature.

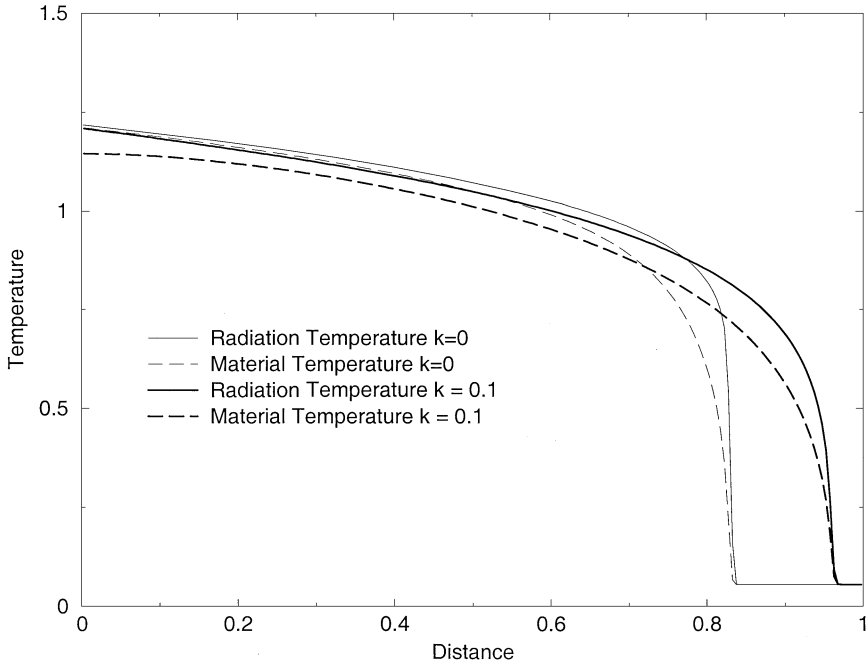

FIG. 1. Effects of thermal diffusion.

In the following, a comparison is made between the Picard preconditioner and the operator split preconditioner for different thermal wave CFL numbers [7]. The basic idea of a thermal wave CFL number is to assign a wave speed to the thermal wave and then base a CFL number on this wave speed. For example, if the CFL number is one, the thermal wave crosses a computational grid cell in one time step. For a thermal wave CFL number of 0.1, the thermal wave takes ten time steps to cross a computational cell. We are using a fixed time step plus a time step ramp which brings the initial time step up to the fixed value in a small number of time steps. The ramp helps the calculation through the initial startup of the simulation. Because of this ramp and the changing velocity of the thermal wave front, the statement of CFL is an asymptotic approximation.

To compare different approaches we partition the computational work into the pieces.

$$WORK \propto N \times \frac{time_f}{\Delta t} \times time\_step \times \frac{Newton}{time\_step} \times \frac{PGMRES}{Newton}. \tag{60}$$

In this equation, final time ($time_f$) is usually fixed by the problem definition so the numbers of interest are the number of unknowns (recall $N = nm$, where n is the number of cells and m is the number of equations per control volume), time step size (Δt), Newton iterations per time step ($\frac{Newton}{time\_step}$), and PGMRES iterations per Newton iteration ($\frac{PGMRES}{Newton}$). In this paper all CPU times are presented as seconds of silicon graphics incorporated, octane computer time.

Table I shows a comparison of the operator split preconditioner to the Picard based preconditioner for a problem with fifty control volumes. In the Picard based preconditioner the linearized coupled system (Eqs. (26) and (25)) is solved using a 2×2 block symmetric Gauss Seidel iteration. The complexity and the storage requirement for the Picard linearized preconditioner (which is already smaller and simpler that a true Jacobian based preconditioner) is higher than the operator split based preconditioner, but one can see from Table I

TABLE I
Operator Split vs Picard for $n = 50$

Problem	Picard			Operator split		
	$\frac{Newton}{time\_step}$	$\frac{PGMRES}{Newton}$	CPU	$\frac{Newton}{time\_step}$	$\frac{PGMRES}{Newton}$	CPU
k = 0.0 CFL = 0.1	2.379	2.845	0.9	2.403	3.456	1.1
k = 0.0 CFL = 1.0	4.678	6.725	0.4	4.750	7.390	0.4
k = 0.1 CFL = 0.1	2.088	2.586	1.0	2.117	3.337	1.1
k = 0.1 CFL = 1.0	4.160	5.403	0.3	4.160	5.978	0.4

that even as the thermal diffusion coefficient is varied and the CFL numbered is varied the two methods perform almost identically.

In Table I, one can clearly see that increasing the nonlinearity of the problem, by raising the time step size an order of magnitude (CFL from 0.1 to 1.0), approximately doubles the number of Newton iterations per time step and also approximately doubles the number of PGMRES iterations per Newton iteration. This indicates that both the nonlinear and the linear problems are more difficult when the time step is increased.

It should be noted here that the predicted run time speed-up of the operator split preconditioner was not realized in this study. This may be simply because the run time analysis was asymptotic and the one-dimensional runs done here were too small to show the asymptotic behavior. It also may be that the second step of the preconditioner implementation (the action of the Jacobian $\mathbf{w} = \mathbf{Jq}$) may be using a significant amount of CPU time relative to the CPU time required to evaluate the action of the preconditioner ($\mathbf{Oq} = \mathbf{v}$).

The effect of changing the material diffusion coefficient in Table I is similar. Here increasing the material diffusion coefficient both lowers the number of nonlinear iterations ($\frac{Newton}{time\_step}$) and the number of linear iterations ($\frac{PGMRES}{Newton}$).

In Table II the same study is repeated for a grid which is four times as large. One observes that the performance of the methods (measured by the number of Newton iterations per time step and the number of PGMRES iterations per Newton iteration) is very similar. Once again increasing the CFL number increases both the linear and nonlinear iterations. However, increasing the material diffusion clearly lowers the nonlinear iterations, but there is now no longer a clear trend in the linear iterations.

4.2. Problem 2: Two-dimensional with one-dimensional simulation. The test problem for this subsection is similar to the one-dimensional problem. The computational domain ranges from $0 \le x \le 1$ and $0 \le y \le 1$. The boundary condition for the radiation equation at

TABLE II
Operator Split vs Picard for $n = 200$

Problem	Picard			Operator split		
	$\frac{Newton}{time\_step}$	$\frac{PGMRES}{Newton}$	CPU	$\frac{Newton}{time\_step}$	$\frac{PGMRES}{Newton}$	CPU
k = 0.0 CFL = 0.1	2.057	3.825	14.4	2.053	4.115	16.0
k = 0.0 CFL = 1.0	5.136	10.341	7.5	5.499	9.990	8.4
k = 0.1 CFL = 0.1	2.009	4.225	18.5	2.008	4.603	20.4
k = 0.1 CFL = 1.0	3.727	9.708	6.2	3.687	9.018	6.1

the left boundary ($x = 0$, $y \in [0, 1]$) is

$$\frac{1}{4}E - \frac{1}{6\sigma_a}\frac{\partial E}{\partial x} = 1. \tag{61}$$

At the right boundary ($x = 1$, $y \in [0, 1]$) we use

$$\frac{1}{4}E + \frac{1}{6\sigma_a}\frac{\partial E}{\partial x} = 0. \tag{62}$$

At the top and bottom boundaries ($y = 0$ or $y = 1$, $x \in [0, 1]$) the boundary conditions are

$$\frac{\partial E}{\partial y} = 0. \tag{63}$$

For the material conduction equation the top and bottom boundaries are

$$\frac{\partial T}{\partial y} = 0, \tag{64}$$

and the left and right boundaries are

$$\frac{\partial T}{\partial x} = 0. \tag{65}$$

The initial conditions are the same as in the one-dimensional runs, the initial energy is $E_0 = 1.0 \times 10^{-5}$, and the initial material temperature is $T_0 = (E_0)^{\frac{1}{4}} \approx 5.62 \times 10^{-2}$. The atomic number z is constant and $z = 1.0$ everywhere. The two-dimensional simulation also runs for three units of time ($time_f = 3.0$).

Figure 2 shows a contour plot of the material temperature at time equal to three. From this plot one can see the step gradient near $x = 0.8$. Also, the zero gradient boundary conditions in both the radiation and material conduction equations is evident in the straight vertical lines of constant temperature.

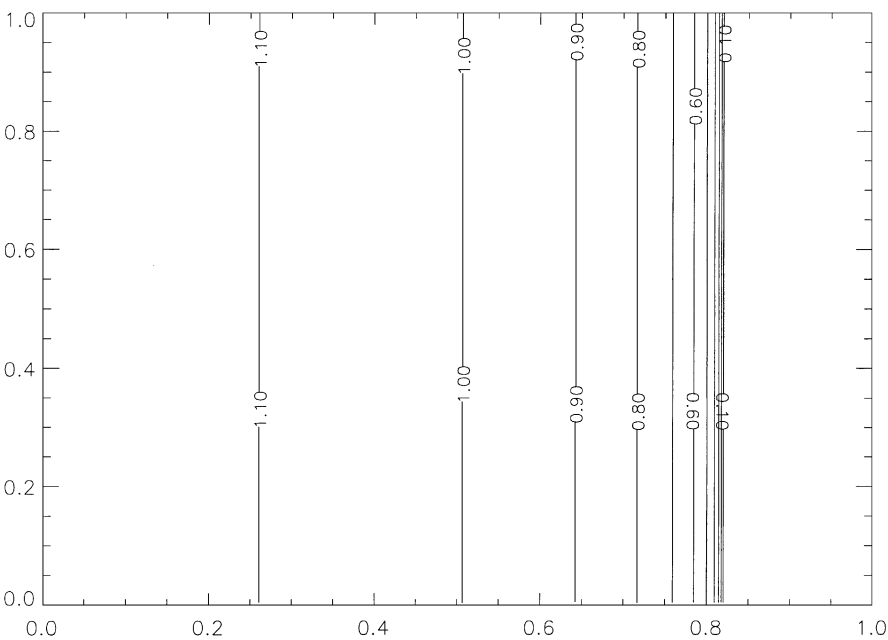

FIG. 2. Material temperature.

TABLE III
Grid Convergence Study CFL $\approx$ 0.1

Grid	Δt	$\frac{Newton}{time\_step}$	$\frac{PGMRES}{Newton}$	CPU
32×32	$1.2e^{-2}$	2.008	4.691	29.1
64×64	$6.0e^{-3}$	1.981	4.819	366.5
128×128	$3.0e^{-3}$	1.392	4.948	2356.5
256×256	$1.5e^{-3}$	1.016	4.966	15917.0

Table III shows a mesh convergence study for this one-dimensional problem solved in two dimensions. In this study the thermal wave CFL number is held approximately equal to 0.1 by cutting the time step (Δt) in half each time the mesh spacing is cut in half. In Table III one can see that the number of Newton iterations per time step falls throughout the mesh refinement. Simultaneously, the number of PGMRES iterations per Newton iteration is approximately constant (slightly increasing). This results in a total run time which scales by slightly less than eight. (Note: this scaling does not hold for the coarsest mesh case. This may be because of cache effects which artificially speed up the smaller problem.) If the amount of work is constant (i.e., the same number of Newton iterations per time step and the same number of PGMRES iterations per Newton iteration) the CPU time will scale roughly by eight (two for the increase in the x direction, two for the increase in the y direction, and two for the halving of the time step).

Table IV shows similar results for the CFL ≈ 0.5 case. Comparing Tables III and IV one can see that increasing the time step by a factor of five increased the nonlinearity (as evidenced by the increase in the number of Newton iterations per time step). Also the number of PGMRES iterations per Newton iteration increased from approximately five to approximately nine. However, the trends in Table IV are identical to Table III. The PGMRES iterations per Newton do not increase with grid size and the total amount of CPU time scales slightly less than a theoretical value of eight (again ignoring the first grid).

Figure 3 shows the Mesh convergence results for a one-dimensional slice taken down the middle of the two-dimensional domain. The plot shows the Radiation temperature at time 3.0 for four different grids. Two observations can be made from this plot. First, it is clear that the solution is converging upon mesh refinement. Second, one can see that the gradient around $x = 0.8$ gets steeper with the mesh refinement. For the 32×32 grid problem, the simulation has difficulties resolving the steep gradient at the thermal wave front. This and cache effects may explain why the CPU time scaling does not hold for the first grid. In Table V, one can see the L-2 norm of the distance between the material temperatures for the coarser grids and the 256×256 grid. Here the ratio of the error terms is slightly less than the theoretical value of four (second order in space and half the grid size). This may be

TABLE IV
Grid Convergence Study CFL $\approx$ 0.5

Grid	Δt	$\frac{Newton}{time\_step}$	$\frac{PGMRES}{Newton}$	CPU
32×32	$6.0e^{-2}$	3.352	8.921	20.3
64×64	$3.0e^{-2}$	3.214	9.092	282.5
128×128	$1.5e^{-2}$	2.755	9.092	1973.3
256×256	$7.5e^{-3}$	2.151	9.094	13893.3

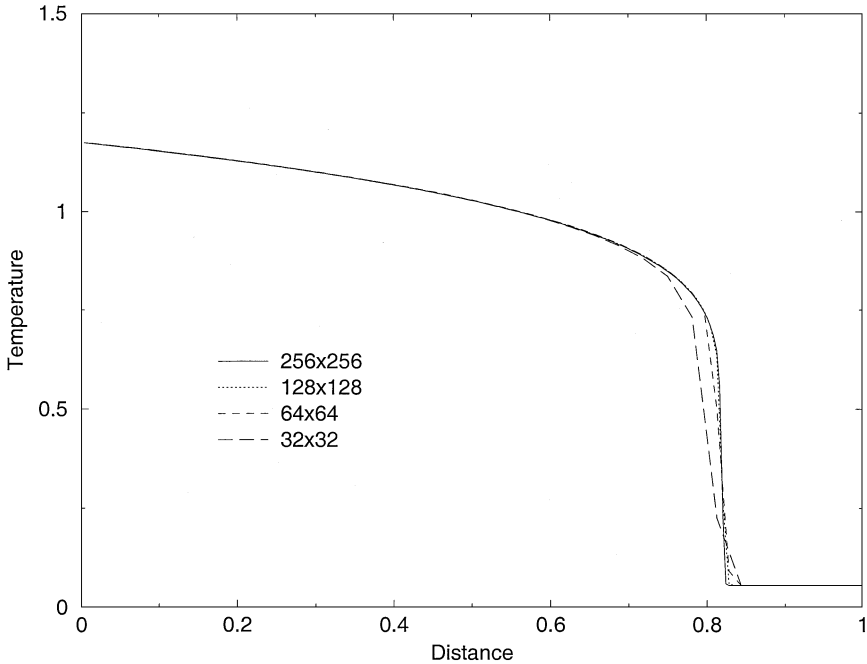

FIG. 3. Radiation temperature mesh convergence.

related to the temporal error term which is first order so the overall theoretical value would be two. The convergence analysis of the data is done for the material temperature instead of the radiation temperature since the material temperature has a smoother profile.

4.3. Problem 3: Two-dimensional with two-dimensional simulation. To clearly demonstrate the two-dimensional capability of this solution scheme, a second two-dimensional problem is constructed. In this problem, the initial conditions, the boundary conditions, and the total run time are exactly the same as in the first two-dimensional run. However, there is now a region in the simulation where the atomic number z is not equal to one. More precisely,

$$z = \begin{cases} 5.0 & \text{if } \frac{1}{3} \le x \le \frac{2}{3} \text{ and } \frac{1}{3} \le y \le \frac{2}{3} \\ 1.0 & \text{otherwise.} \end{cases} \tag{66}$$

Table VI shows the solution algorithm performance under mesh refinement for the base two-dimensional geometry problem. In Table VI one can see that the number of Newton iterations per time step decreases the same as in the one-dimensional test problem. However, instead of the flat number of PGMRES per Newton iterations that was clear in the one-dimensional test problem, the number of PGMRES iterations per Newton iteration actually

TABLE V
Material Temperature Mesh Convergence

Grids	Error $\|\Delta T\|_2$	Ratio
32-256	0.6992	NA
64-256	0.1997	3.50
128-256	0.0524	3.81

TABLE VI
Grid Convergence Study CFL ≈ 0.1

Grid	Δt	$\frac{Newton}{time\_step}$	$\frac{PGMRES}{Newton}$	CPU
32×32	$1.2e^{-2}$	1.991	4.684	22.3
64×64	$6.0e^{-3}$	1.185	4.339	152.0
128×128	$3.0e^{-3}$	1.001	3.890	1074.9
256×256	$1.5e^{-3}$	1.000	3.720	11550.6

decreases. It should be noted here that there is a slight discrepancy in the timing data. Because of the initial strength of the transient (full energy being added to a cold material), the time steps for these simulations start out small and are then ramped up to their maximum time step size. This ramp takes place over approximately fifty time steps. The statistics for the average Newton iterations per time step and the average PGMRES iterations per Newton iteration do not include the work that is done on the time step ramp. However, the CPU time information includes the work done during the ramping of the time step. Looking at the CPU time for the 256×256 run in Table VI, one can see that even though the number of Newton iterations per time step goes down as does the number of PGMRES iterations per Newton iteration, the increase in run time is greater than the theoretical value of eight. This anomaly appears to be associated with the amount of work necessary to get the simulation through the initial transient.

To understand this data one must consider two important gradients that need to be resolved in this problem. The first one has no physical meaning and is simply an effect of the discretization. In the boundary conditions for this problem, the energy flux into the left wall is constant. The initial condition for this problem is the material is initially cold. Therefore, under mesh refinement the initial constant energy flux is being imparted into a smaller and smaller initially cold cell. This makes the start of the problem more difficult as the mesh spacing is decreased.

The second gradient which is important to consider is the gradient in the diffusion coefficient. From Eqs. (14) and (15) one can see that D_r is proportional to T and z or in equation form,

$$D_r \propto \frac{T^3}{z^3}. \tag{67}$$

The temperature across the interface at the thermal wave front drops by about a factor of ten. If the atomic number varies by a factor of ten between two different materials (as is the case in Fig. 4) the total change in diffusion coefficient (if these two differences occur at the same location) will be $(10^3)^2 = 10^6$. This is clearly evident in Fig. 4 where the contours of $\log_{10}$ of constant diffusion coefficient vary from -1 in the "hot" low z area to -7 in the "cold" high z area. Here the diffusion coefficient varies over six orders of magnitude in a very small spatial distance. Therefore, as the mesh is refined the ability to resolve this six order of magnitude drop improves. This may be the explanation for why the number of PGMRES iterations per Newton iteration decreases for this problem.

To investigate this phenomenon further the data in Table VII were generated. For this study the effects of grid spacing are removed by fixing the mesh at 128×128. In Table VII one can now see clearly that as the gradient in diffusion coefficients steepens (the z ratio increases) the amount of work that PGMRES requires to solve the linear system increases. This is illustrated in Figs. 5 and 6.

TABLE VII

Atomic Number Study CFL $\approx$ 0.1 Grid 128×128 $\Delta t = 3.0 \times 10^{-3}$

Z number	Z^3	$\frac{Newton}{time\_step}$	$\frac{PGMRES}{Newton}$	CPU
1.25	1.95	1.001	3.316	1063.4
2.5	15.62	1.001	3.368	1084.6
5.0	125.00	1.001	3.890	1072.5
7.5	421.87	1.001	5.157	1760.6
10.0	1000.00	1.001	11.846	3933.2

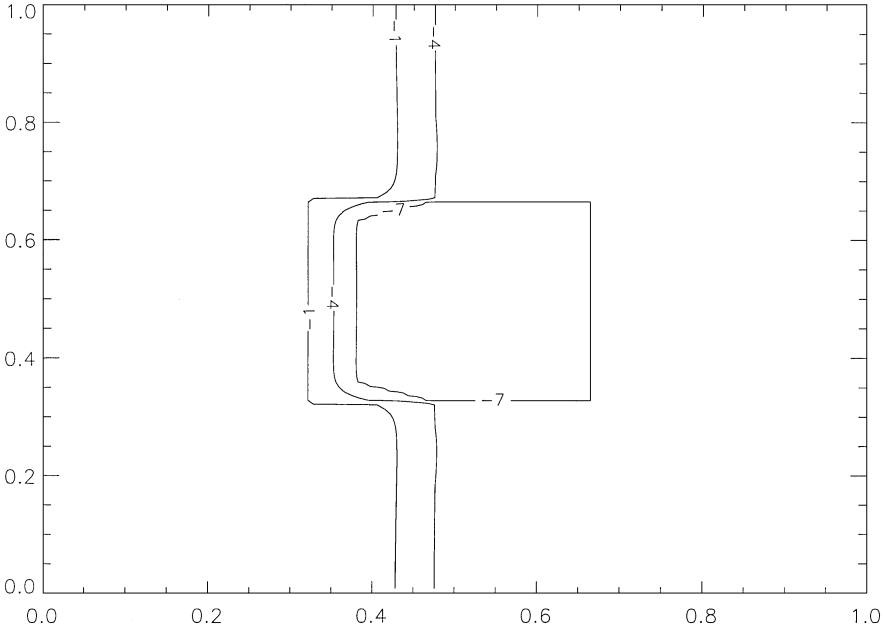

FIG. 4. The $\log_{10}$ of the diffusion coefficient; z number ratio = 10.0.

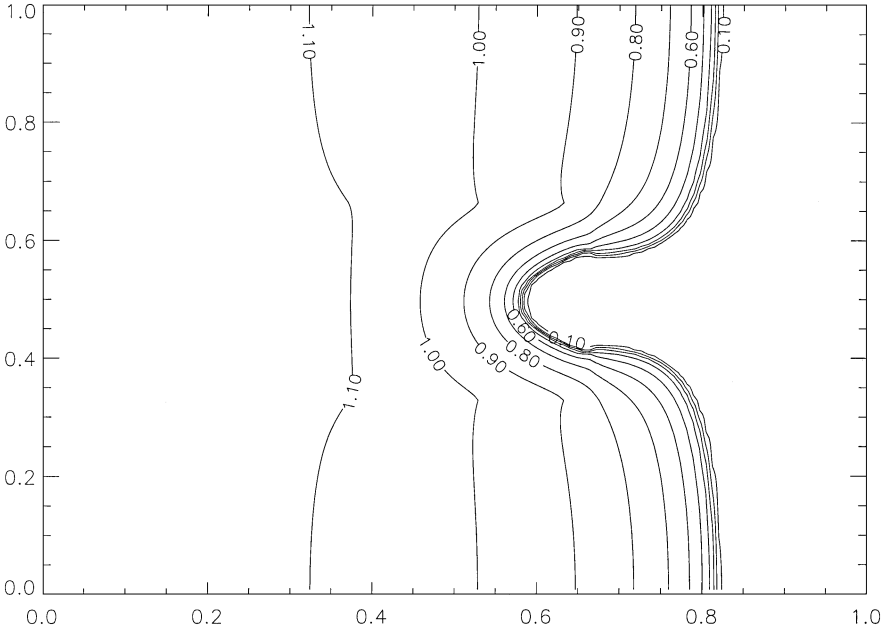

FIG. 5. Material temperature; z number ratio = 2.5.

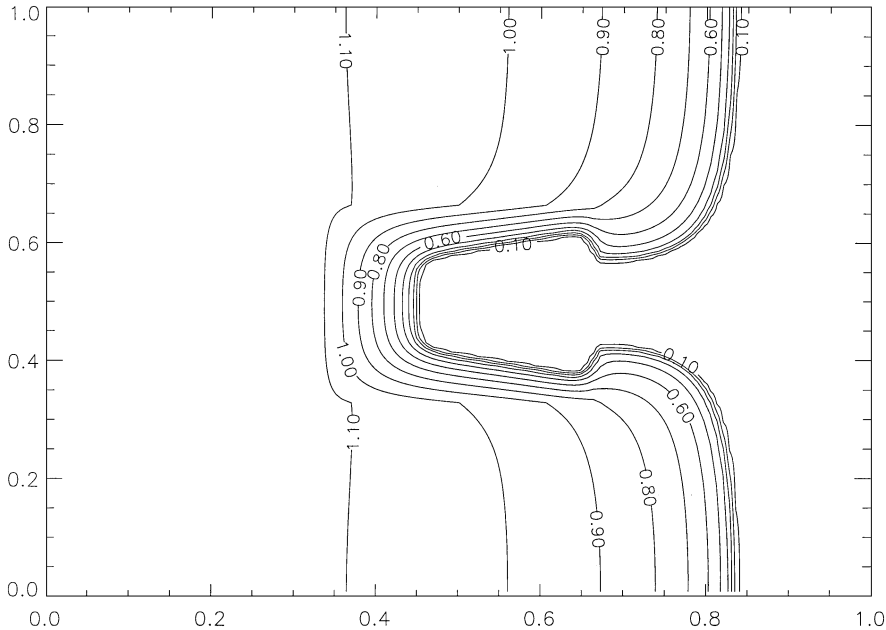

FIG. 6. Material temperature; z number ratio = 10.0.

In Fig. 5 one can see the material temperature contours for the z number ratio of 2.5. A z ratio of 2.5 results in a diffusion coefficient difference of $(2.5)^3 = 15.625$. In this run, the high z material slows the the thermal wave but the actual shape of the high z square is not evident.

In Fig. 6 the z ratio is increased to 10. The difference in the diffusion coefficient for this case is $(10)^3 = 1000$. Here the thermal wave front is radically slowed which causes an additional steepening of the diffusion coefficients due to the sharp thermal gradients.

5. CONCLUSIONS

A physics-based preconditioning approach based on operator splitting has been described. This approach allows one to use well developed technology to construct a preconditioner for a fully coupled, fully implicit, solution. Because of the physics-based operator split technology, the preconditioner is easy to construct, uses very little memory, and allows one to employ physics insight at each of the operator split steps.

In one dimension it was demonstrated that the scalar based operator split preconditioner was as efficient as the fully coupled Picard based preconditioner. In two dimensions it was demonstrated that the multigrid operator split preconditioning provided a linear solving capability that scales independent of the grid size. Finally results from a true two-dimensional problem with interesting physics showed that this solution algorithm behaves well under difficult conditions (i.e., diffusion coefficients that vary by six orders of magnitude).

It should be noted that in this paper the operator-split preconditioned Newton–Krylov method has only been demonstrated for the non-equilibrium radiation diffusion problem. In another paper [10], we demonstrate this approach on the incompressible Navier–Stokes equations in stream-function vorticity form. Currently we are adapting this approach to

the shallow water wave equations, the radiation hydrodynamics equations, and the magnetohydrodynamics equations. Progress on this work will be reported in future publications.

ACKNOWLEDGMENTS

This work was supported under the auspices of the U.S. Department of Energy under DOE Contract W-7405-ENG-36 at Los Alamos National Laboratory.

REFERENCES

1. P. N. Brown and Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, *SIAM J. Sci. Statist. Comput.* **11**, 450 (1990).
2. R. L Bowers and J. R. Wilson, *Numerical Modeling in Applied Physics and Astrophysics* (Jones & Bartlett, Boston, 1991).
3. P. Wesseling, *An Introduction to Multigrid Methods* (Wiley, Chichester, England, 1992).
4. W. L. Briggs, *A Multigrid Tutorial* (Soc. for Industr. & Appl. Math., Philadelphia, 1987).
5. D. A. Knoll, W. J. Rider, and G. L. Olson, An efficient nonlinear solution method for non-equilibrium radiation diffusion, *J. Quantitative Spectrosc. Radiat. Transfer* **63**, 15 (1999).
6. W. J. Rider, D. A. Knoll, and G. L. Olson, A multigrid Newton–Krylov method for multidimensional equilibrium radiation diffusion, *J. Comput. Phys.* **152**, 164 (1999).
7. W. J. Rider and D. A. Knoll, Time step size selection for radiation diffusion calculations, *J. Comput. Phys.* **152**, 790 (1999).
8. D. A. Knoll, W. J. Rider, and G. L. Olson, Nonlinear convergence, accuracy, and time step control in nonequilibrium radiation diffusion, submitted to *J. Quant. Spectrosc. Radiat. Transfer*, LA-UR-98-4649.
9. C. Baldwin, P. Brown, R. Falgout, F. Graziani, and J. Jones, Iterative linear solvers in a 2D radiation-hydrodynamics code: Methods and performance, *J. Comput. Phys.* **154**, 1 (1999).
10. D. A. Knoll and V. A. Mousseau, On Newton–Krylov-multigrid methods for the incompressible Navier–Stokes equations, *J. Comput. Phys.*, in review, LA-UR-99-5100.
11. D. A. Knoll and P. R. McHugh, Enhanced nonlinear iterative techniques applied to a nonequilibrium plasma flow, *SIAM J. Sci. Comput.* **19**(1), (1998).
12. D. A. Knoll, An improved convection scheme applied to recombining diverter plasma flows, *J. Comput. Phys.* **142**, 473 (1998).
13. P. N. Brown and A. C. Hindmarsh, Matrix-free methods for stiff systems of ODE's, *SIAM J. Numer. Anal.* **23**, 610 (1986).
14. R. Dembo, S. Eisenstat, and T. Steihaug, Inexact Newton methods, *SIAM J. Numer. Anal.* **19**, 400 (1982).
15. Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving non-symetric linear systems, *SIAM J. Sci. Statist. Comput.* **7**, 856 (1986).
16. P. J. Roach, *Computational Fluid Dynamics* (Hermosa, Albuquerque, NM, 1982).
17. R. E. Bank, T. F. Chan, W. M. Coughran, and R. K. Smith, *The Alternate-Block-Factorization Procedure for Systems of Partial Differential Equations*, Technical Report, AT & T Bell Laboratories, Numerical Analysis Manuscript 89-5, July 1989.
18. D. A. Knoll, G. Lapenta, and J. U. Brackbill, A multilevel iterative field solver for implicit, kinetic, plasma simulation, *J. Comput. Phys.* **149**, 377 (1999).
19. B. Smith, P. Bjorstad, and W. Gropp, *Domain Decomposition Parallel Multilevel Methods for Elliptic Partial Differential Equations* (Cambridge Univ. Press, Cambridge, UK, 1996).
20. D. L. Peterson, R. L. Bowers, A. E. Greene, K. D. McLenithan, T. A. Oliphant, N. F. Roderick, and A. J. Scannapieco, Two-dimensional modeling of magnetically driven Rayleigh–Taylor instabilities in cylindrical Z pinches, *Phys. Plasmas* **3**(1), 368 (1996).

21. A. M. Winslow, Multifrequency-Gray method for radiation diffusion with Compton scattering, *J. Comput. Phys.* **117**, 262 (1995).
22. L. Spitzer and R. Harm, Transport phenomena in a completely ionized gas, *Phys. Rev.* **89**(5), 977 (1953).
23. M. D'Amico and G. C. Pomraning, Treatment of nonlinearities in flux-limited diffusion calculations, *J. Quant. Spectrosc. Radiat. Transfer* **49**, 457 (1993).
24. R. H. Szilard and G. C. Pomraning, Numerical transport and diffusion methods in radiative transfer, *Nucl. Sci. Eng.* **112**, 256 (1992).
25. W. Dai and P. R. Woodward, Numerical Simulations for Nonlinear Heat Transfer in a System of Multimaterials, *J. Comput. Phys.* **139**, 58 (1998).