NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2004; **11**:in press (DOI: 10.1002/nla.386)

One-level Newton–Krylov–Schwarz algorithm for unsteady non-linear radiation diffusion problem

Serguei Ovtchinnikov and Xiao-Chuan Cai*,†

Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, U.S.A.

SUMMARY

In this paper, we present a parallel Newton–Krylov–Schwarz (NKS)-based non-linearly implicit algorithm for the numerical solution of the unsteady non-linear multimaterial radiation diffusion problem in two-dimensional space. A robust solver technology is required for handling the high non-linearity and large jumps in material coefficients typically associated with simulations of radiation diffusion phenomena. We show numerically that NKS converges well even with rather large inflow flux boundary conditions. We observe that the approach is non-linearly scalable, but not linearly scalable in terms of iteration numbers. However, CPU time is more important than the iteration numbers, and our numerical experiments show that the algorithm is CPU-time-scalable even without a coarse space given that the mesh is fine enough. This makes the algorithm potentially more attractive than multilevel methods, especially on unstructured grids, where course grids are often not easy to construct. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: implicit method; Newton–Krylov–Schwarz; additive Schwarz; non-linear radiation diffusion equation; parallel processing

1. INTRODUCTION

Radiation transport plays an important role in many physical phenomena. Radiation diffusion, being a special case of the radiation transport, is obtained by assuming that the medium is isotropic and optically thick. Utilizing the black-body radiation model and integrating over all radiation frequencies with an assumption of radiation-media equilibrium, an equilibrium radiation diffusion description is found [1]. Radiation diffusion is a highly non-linear phenomenon. In addition to the non-linear behaviour resulting from the governing equations, the flux-limited form of the diffusion coefficient introduces an extra degree of non-linearity to the system [2, 3]. Several successful attempts have been made to numerically solve the radiation

*Correspondence to: X.-C. Cai, Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, U.S.A.
†E-mail: cai@cs.colorado.edu

Contract/grant sponsor: Department of Energy; contract/grant number: DE-FC02-01ER25479
Contract/grant sponsor: National Science Foundation; contract/grant number: ACI-0072089
Contract/grant sponsor: National Science Foundation; contract/grant number: ACI-0305666

Received 6 May 2003
Revised 18 November 2003
Accepted 7 December 2003

diffusion equations [1, 4–9]. A typical implementation takes advantage of the multigrid approach combined with some variant of Newton's method. The time derivative is often dealt with by using either an implicit backward Euler or a Crank–Nicolson scheme. In this paper, we study a one-level Newton–Krylov–Schwarz (NKS) [10] method and speculate that the single mesh method has certain advantages over methods that require multiple meshes in order to achieve the optimal convergence and scalability when the number of processors is large. As is known, optimal choices of the courser meshes may not be easy to obtain for an unstructured mesh. In addition, multiple meshes may not be easy to implement on parallel computers.

Integration of the radiation diffusion equations is a computationally intensive task. Parallel processing has to be applied in order to attain the computational practicality in realistic settings. In this work, we use an overlapping additive Schwarz domain decomposition [18] to divide the computational domain and partition the work-load among multiple processors. Our approach is based on a backward Euler time discretization combined with an inexact Newton method for solving the non-linear algebraic systems. The Jacobian linear systems are solved with an additive Schwarz preconditioned restarted GMRES. We study the scalability of our implementation and its parallel performance. In the parallel implementation we use the Portable Extensible Toolkit for Scientific computation (PETSc) package available as an open source software from the Argonne National Laboratory [11].

The rest of the paper is organized as follows. In Section 2, we discuss a model radiation diffusion equation and its discretization. In Section 3, we introduce a parallel one-level Newton–Krylov–Schwarz method. Section 4 is devoted to numerical experiments and parallel performance of the NKS method. Finally, in Section 5, we conclude the paper with several remarks.

2. A MODEL PROBLEM

In a wide variety of applications, radiation diffusion is essential to our understanding of the underlying physics, for example, in the modelling of the interior of the Sun and in the modelling of the processes taking place during a nuclear blast. Radiation diffusion can be posed in several forms. One of the possible approaches [1] is to start with the equations of non-equilibrium radiation diffusion:

$$\begin{aligned} \frac{\partial E}{\partial t} &= \nabla \cdot \left(\frac{c}{3\kappa} \nabla E \right) + c\kappa(aT^4 - E) \\ \frac{\partial C_v T}{\partial t} &= c\kappa(E - aT^4) \end{aligned} \tag{1}$$

where κ is the opacity, a is the Stefan–Boltzmann constant, c is the speed of light, E is the radiation energy, T is the material temperature and C_v is the heat capacity of the medium. The first equation in (1) is the energy equation, and the second one provides an expression for the material temperature. For the sake of simplicity, we assume that $C_v = a = 1$ and $c = 3$. Furthermore, assuming that the material temperature is in equilibrium with the radiation energy

$$E = aT^4$$

and that the opacity has the form

$$\kappa = T^{-3}$$

we can simplify the system of equations (1) to a scalar equation

$$\frac{\partial(\alpha + (1-\alpha)E^{-3/4})E}{\partial t} = \nabla \cdot (D(E)\nabla E) \tag{2}$$

where

$$D(E) = Z^{\beta}E^{3/4}$$

Equation (2) signifies that there are two important limits: (a) the energy of the system is dominated by the material energy; and (b) the energy of the system is dominated by the radiation energy. In our study we set $\alpha = 1$ considering only the case when the energy of the system is dominated by the radiation energy. We experiment with a multimaterial model, where the opacity depends on the atomic number Z of the medium. Therefore, the diffusion coefficient $D(E)$ is also a function of the atomic number. In our experiments we set $\beta = -3$. The spatial distribution of Z will be introduced later in the paper.

In the above formulation the diffusion coefficient may lose its physical meaning because there is nothing in the functional form of the coefficient to prevent transport phenomena occurring faster than the maximum speed in the media (the speed of light). Following [1], we use a flux-limited diffusion in the following form:

$$D_L(E) = \frac{1}{1/D(E) + |\nabla E|/E}$$

Now, replacing $D(E)$ in (2) by $D_L(E)$, we obtain

$$\frac{\partial E}{\partial t} = \nabla \cdot \left(\frac{1}{Z^{-\beta}E^{-3/4} + |\nabla E|/E} \cdot \nabla E \right) \tag{3}$$

Solving (3) is the main focus of the paper. Let $E = E(x, y, t)$ with $(x, y) \in \Omega$ and $t \in [0, T]$. We only consider a simple two-dimensional case, where $\Omega = [0, 1] \times [0, 1]$. The initial and boundary conditions are given as follows. On the top and the bottom boundaries of the domain, we assume

$$\frac{\partial E}{\partial n} = 0 \quad \text{when } y = 0 \text{ or } y = 1 \tag{4}$$

and on the left and the right boundaries of the domain, we assume

$$\frac{1}{4}E + \frac{1}{2}D_L(E)\frac{\partial E}{\partial x} = E_{\text{flux}} \quad \text{when } x = 0 \text{ or } x = 1 \tag{5}$$

We also assume the initial condition

$$E(x, y, 0) = 1 \quad \text{for any } (x, y) \in \Omega$$

In the rest of this section, we briefly describe the time and space discretization schemes. Based on the results of Reference [1], explicit schemes are not practical because of the

restrictive nature of the stability-based time-step control. Therefore, we use an implicit backward Euler scheme for the time integration with a fixed time step size Δt,

$$\frac{E^{n+1}-E^n}{\Delta t} = \nabla \cdot (D_L(E^{n+1})\nabla E^{n+1}), \quad n = 0, 1, \ldots \tag{6}$$

Here $E^0 = E(\cdot, \cdot, 0)$. We discretize the spatial part of the radiation diffusion equation using a standard five-point finite difference method on a uniform grid of size h in both x and y directions. More precisely, at a given interior mesh point (i, j), we take

$$\begin{aligned}(\nabla(D_L(E)\nabla E))_{i,j} \approx & \frac{1}{h}\left(D_L^{i+1/2,j}\,\frac{E_{i+1,j}-E_{i,j}}{h} - D_L^{i-1/2,j}\,\frac{E_{i,j}-E_{i-1,j}}{h} - \right) \\ & + \frac{1}{h}\left(D_L^{i,j+1/2}\,\frac{E_{i,j+1}-E_{i,j}}{h} - D_L^{i,j-1/2}\,\frac{E_{i,j}-E_{i,j-1}}{h}\right)\end{aligned} \tag{7}$$

Here the 'half-point' $D_L^{i+1/2,j}$ is defined as follows:

$$D_L^{i+1/2,j} = \frac{1}{1/D^{i+1/2,j} + (|E_{i+1,j}-E_{i,j}|/h)/((E_{i+1,j}+E_{i,j})/2)} \tag{8}$$

where

$$D^{i+1/2,j} = \frac{2D(E_{i+1,j})D(E_{i,j})}{D(E_{i+1,j})+D(E_{i,j})}$$

is taken as the harmonic mean value. The schemes at other half-points are similar. Let M_x and M_y be the numbers of mesh points in x and y directions. The boundary conditions are discretized using

$$\begin{aligned}\frac{1}{4}E_{0,j} + \frac{1}{2}D_L(E_{0,j})\,\frac{E_{1,j}-E_{0,j}}{h} &= -E_{\text{influx}} \\ \frac{1}{4}E_{M_x-1,j} + \frac{1}{2}D_L(E_{M_x-1,j})\,\frac{E_{M_x-2,j}-E_{M_x-1,j}}{h} &= E_{\text{outflux}} \\ E_{i,0}-E_{i,1} &= 0 \\ E_{i,M_y-1}-E_{i,M_y-2} &= 0\end{aligned}$$

for the left, right, bottom and top boundaries, respectively. The corner points are ignored in the computation. $D_L(E_{0,j})$ and $D_L(E_{M_x-1,j})$ are given by

$$D_L(E_{0,j}) = \frac{1}{1/D^{0,j} + |E_{1,j}-E_{0,j}|/h/E_{0,j}}$$

and

$$D_L(E_{M_x-1,j}) = \frac{1}{1/D^{M_x-1,j} + |E_{M_x-2,j}-E_{M_x-1,j}|/h/E_{M_x-1,j}}$$

respectively. Combining time and space discretizations, and using a natural ordering of the mesh points, we obtain a non-linear system of equations that needs to be solved at every time step:

$$(E^{n+1} - \Delta t \nabla \cdot D_L(E^{n+1}) \nabla E^{n+1} - E^n)_{i,j} = 0 \tag{9}$$

In the remaining part of the paper, we will denote such a non-linear system of equations as

$$F(E) = 0 \tag{10}$$

Note that the absolute value function that appears in (8) and the discontinuous atomic coefficient Z, to be specified explicitly later in the paper, make the non-linear algebraic system non-differentiable and difficult to solve using a Newton-type method in which some kind of derivative information is necessary. In this paper, we construct the Jacobian matrix of F using a multi-coloured finite difference method introduced in Reference [12].

3. ONE-LEVEL NEWTON–KRYLOV–SCHWARZ METHOD

The family of Newton–Krylov–Schwarz (NKS) methods is a general-purpose parallel algorithm for solving systems of non-linear algebraic equations. NKS, as its name suggests, has three main components: (1) an inexact Newton method for the non-linear systems; (2) a Krylov subspace linear solver for the Jacobian equations (restarted GMRES [13]); and (3) a Schwarz-type preconditioner. We carry out the Newton iterations as following:

$$E_{k+1} = E_k - \lambda_k J(E_k)^{-1} F(E_k), \quad k = 0, 1, \ldots \tag{11}$$

where E_0 is an initial approximation to the solution and $J(E_k) = F'(E_k)$ is the Jacobian at E_k, and λ_k is the steplength determined by a linesearch procedure [14, 15]. The inexactness of Newton's method is reflected in the fact that we do not solve the Jacobian system exactly. The accuracy of the Jacobian solver is determined by some $\eta_k \in [0, 1)$ and the condition

$$\|F(E_k) + J(E_k)s_k\| \leqslant \eta_k \|F(E_k)\| \tag{12}$$

The overall algorithm can be described as follows:

1. Inexactly solve the linear system $J(E_k)s_k = -F(E_k)$ for s_k using a preconditioned GMRES(30).
2. Perform a full Newton step with $\lambda_0 = 1$ in the direction s_k.
3. If the full Newton step is unacceptable, we backtrack λ_0 using the cubic backtracking procedure until a new λ is obtained that makes the $E_+ = E_k + \lambda_k s_k$ an acceptable step.
4. Set $E_{k+1} = E_+$ and return to step 1 unless a stopping condition has been met.

In step 1 above we use a left-preconditioned GMRES to solve the linear system; i.e. the vector s_k is obtained by approximately solving the linear Jacobian system

$$M_k^{-1} J(E_k) s_k = -M_k^{-1} F(E_k)$$

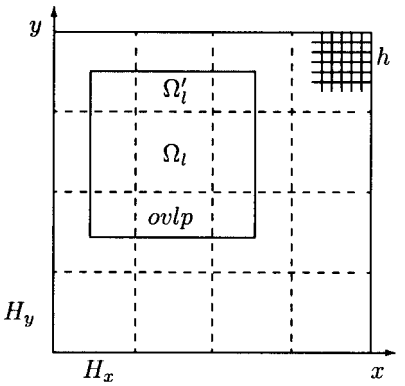

Figure 1. Decomposition of domain Ω with an overlap ovlp. The dashed lines indicate the partition of the domain into non-overlapping substructures Ω_l of size $H_x \times H_y$, and the innermost solid rectangle indicates an overlapping subdomain Ω_l'. The incomplete fine mesh of solid lines illustrates underlying uniform subintervals with mesh size h.

where M_k^{-1} is a one-level additive Schwarz preconditioner [18]. To formally define M_k^{-1}, we need to introduce a partition of Ω. We first partition the domain into non-overlapping substructures Ω_l, $l = 1, \ldots, N$, as shown in Figure 1. In order to obtain an overlapping decomposition of the domain, we extend each subregion Ω_l to a larger region Ω_l', i.e. $\Omega_l \subset \Omega_l'$. Only simple box decomposition is considered in this paper—all the subdomains Ω_l and Ω_l' are rectangular and made up of integral numbers of fine mesh cells. The size of Ω_l is $H_x \times H_y$ and the size of Ω_l' is $H_x' \times H_y'$, where the H's are chosen so that the overlap, ovlp, is uniform in the number of fine grid cells all around the perimeter, i.e.

$$\text{ovlp} = (H_x' - H_x)/2 = (H_y' - H_y)/2$$

for interior subdomains. For boundary subdomains, we simply cut off the part that is outside Ω. Figure 1 illustrates a decomposition with an overlap of four fine mesh cells. On each extended subdomain Ω_l', we construct a subdomain preconditioner B_l, whose elements are $B_l^{i,j} = \{J_{ij}\}$, where the node indexed by (i, j) belongs to the interior of Ω_l'. The entry J_{ij} is calculated with finite differences $J_{ij} = 1/\delta(F_i(E_j + \delta) - F_i(E_j))$, where $0 < \delta \ll 1$ is a constant. Homogeneous Dirichlet boundary conditions are used on the internal subdomain boundary $\partial\Omega_l' \cap \Omega$, and the original boundary conditions are used on the physical boundary, if present. The additive Schwarz preconditioner can be written as

$$M_k^{-1} = I_1 B_1^{-1} (I_1)^{\mathrm{T}} + \cdots + I_N B_N^{-1} (I_N)^{\mathrm{T}} \tag{13}$$

Let n be the total number of mesh points, and n_l' the total number of mesh points in Ω_l', then I_l is an $n \times n_l'$ extension matrix that extends each vector defined on Ω_l' to a vector defined on the entire fine mesh by padding an $n_l' \times n_l'$ identity matrix with zero rows. Various inexact additive Schwarz preconditioners can be constructed by replacing the matrices B_l in (13) with convenient and inexpensive to compute matrices, such as those obtained with incomplete factorizations. In this paper we employ ILU factorizations.

4. PARALLEL IMPLEMENTATION AND NUMERICAL RESULTS

4.1. Software

We use Portable Extensible Toolkit for Scientific computation (PETSc), developed at Argonne National Laboratory [11], in our implementation of the NKS algorithm discussed in the previous section. In a nutshell, PETSc is a suite of data structures and functions encapsulating necessary components for building large-scale parallel and serial scientific applications. The code is written in a hostless manner and allows easy switching between different numbers of processors. Each processor is assigned one subdomain, and the information pertaining to the interior of a subdomain is uniquely owned by that processor. The processor stores subvectors and a block of the Jacobian matrix associated with an extended subdomain. At the beginning of every non-linear iteration, the E-dependent local blocks of the Jacobian, as well as the preconditioning matrices, are computed. The preconditioning matrices are factored, and the upper and lower triangular parts are stored. After the solution of each subproblem is obtained, those portions that lie within the overlapping regions are sent to neighbouring processors to complete collective operations. Participating processors communicate with each other by message passing using MPI.

4.2. Test case and problem parameters

In our numerical tests Ω is a unit square, uniformly partitioned into rectangular meshes up to 2048×2048 in size. We assume the following boundary conditions, see Figure 2:

- On Γ_2 and Γ_4, we assume $\partial E/\partial n = 0$, where $n = (n_x, n_y)$ is the unit outward normal.
- On Γ_1, we impose the influx condition

$$\frac{1}{4}E = \frac{1}{2}D_L(E)\frac{\partial E}{\partial x} = -E_{\text{influx}}$$

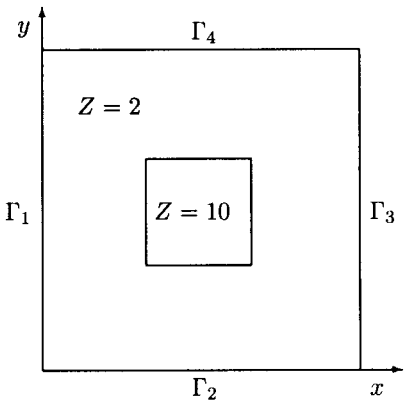

Figure 2. Physical domain with a middle inset. The inset is positioned in $\left(\frac{1}{3}, \frac{2}{3}\right)$ in both x and y directions. Inside the inset the atomic number $Z = 10$, and $Z = 2$ outside the inset. Γ_1 is the influx boundary and Γ_3 is the outflux boundary. No energy transport is allowed across Γ_2 and Γ_4 boundaries.

where E_{influx} is the energy influx in the direction of the outward normal. In our tests, we set $E_{\text{influx}} = 2500.0$ which is, according to Reference [1], in the range of large influx values.

- On Γ_3, we impose the outflux condition

$$\frac{1}{4}E + \frac{1}{2}D_L(E)\frac{\partial E}{\partial x} = E_{\text{outflux}}$$

where $E_{\text{outflux}} = 0.25$ is the outflux of energy in the direction of the outward normal.

The E_{influx} parameter plays an important role. If this parameter is small, then the equation is easy to integrate, and the solution is smooth. As E_{influx} increases, the solution develops a sharp front, and the integration of the system becomes increasingly difficult. For large values of the influx parameter the non-linear iterative process may break down thus preventing the algorithm from obtaining a meaningful solution. The break-down exhibits itself by ushering the system into a stagnation mode, when the non-linear residual either stops decreasing, or decreases by a very small fraction. In our experiments we use $E_{\text{influx}} = 2500.0$ which is already in the high value range.

Note that the material in Ω is inhomogeneous. Ω is partitioned into two sub-regions with different atomic numbers Z, as shown in Figure 2. Therefore, the value of the diffusion coefficient D_L depends on the spatial co-ordinates x and y.

A number of parameters needs to be specified for a successful run of the NKS algorithm. Below we mention the values of each parameter that we use in numerical experiments:

- Finite-differencing parameter, δ. We compute the Jacobian system at each Newton iteration by a multicoloured finite difference method with the parameter δ. We have observed that the δ-parameter is important for the convergence of Newton iterations. The choice of a numerical value for the δ-parameter depends on both the fine mesh size h and the time step. For the mesh sizes of 128×128, 256×256, 512×512, 1024×1024 and the time step $\Delta t = 0.001$ we set $\delta = 10^{-10}$. For the 2048×2048 mesh and time step $\Delta t = 0.001$ δ is reduced to 10^{-12}.
- Non-linear solver. The initial guess is a uniform energy distribution $E_{i,j} = 1$. We declare the non-linear convergence if the condition $\|F(E_k)\| \leqslant 10^{-8}\|F(E_0)\|$ is satisfied.
- Linear solver. The convergence tolerance for the linear iterative solver at each Newton iteration, η_k, is 10^{-6}. We restart GMRES at every 30th iteration.
- Decomposition of Ω. We always set the number of subdomains to be equal to the number of processors. Depending on the mesh size, we employ 1–256 processors in our numerical experiments.
- Overlap size, ovlp. In this paper, we assume that the same number of fine mesh cells, $\text{ovlp} \in [1, 20]$, is extended in both x and y directions.

4.3. Accuracy of the numerical solution

In general, the solution of the two-dimensional equilibrium radiation diffusion results in a energy/temperature surface. Frequently, the analogy of a temperature wave, originating at one side of the domain and propagating to the other side, is used to describe the solution. The shape of this wave depends on the boundary conditions and the material properties of the interior domain points. Figure 3 shows some sample solution contour plots on a 128×128

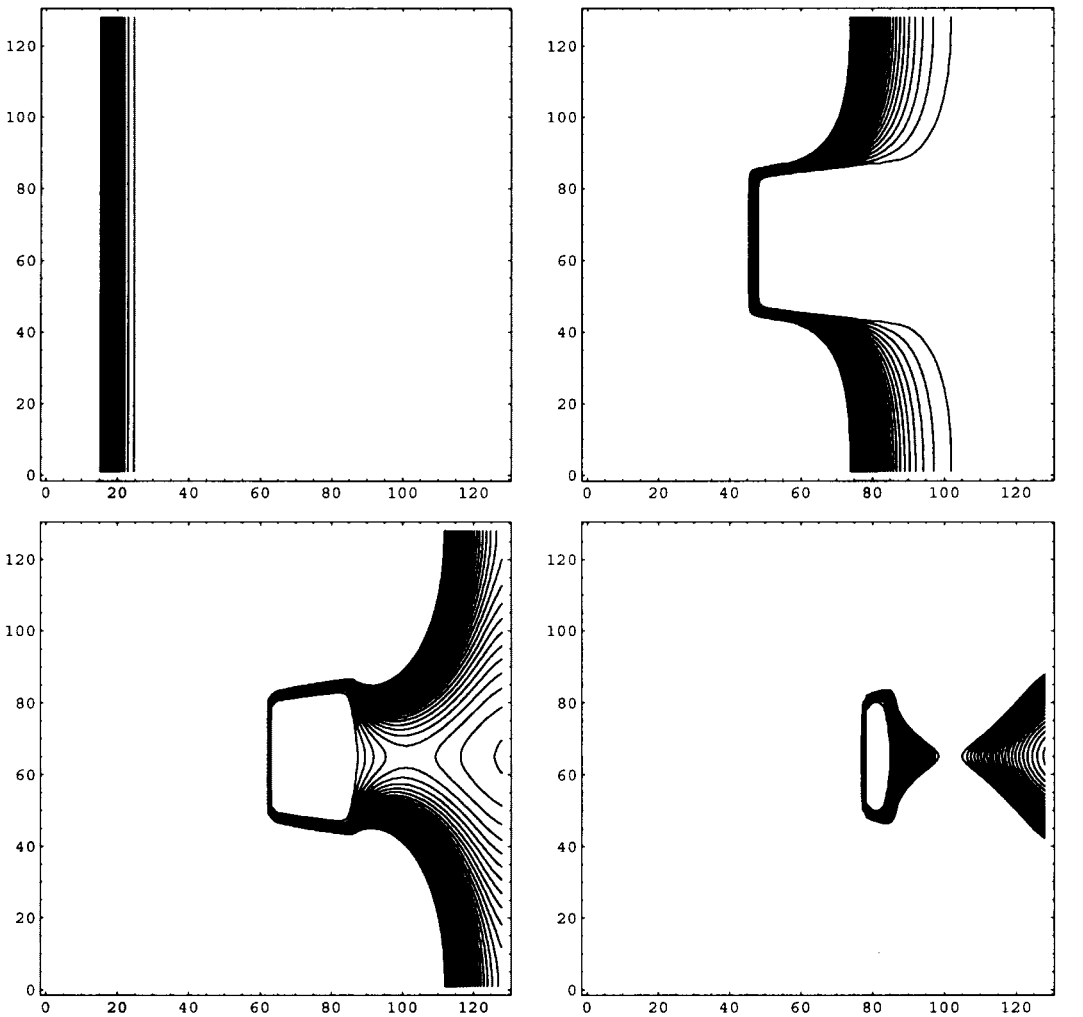

Figure 3. Contour plots of the solution on a 128×128 mesh with $\Delta t = 0.001$. The plots are positioned in left to right, top to bottom order.

mesh with time step $\Delta t = 0.001$. The contour plots are shown in the left–right, top–bottom order. We can observe a formation of the temperature wave on the left-hand side of the domain that travels to the right as the time-integration proceeds. Note that the multimaterial geometry of the domain effectively slows down the temperature wave propagation. The last contour plot presents the solution after 520 time steps. In our experiments, we investigate solutions obtained after up to 5000 time steps with a typical time step of $\Delta t = 0.001$.

Time accuracy is an important concern of our numerical simulation. Figure 4 demonstrates the high-profile plots of the solution when the elapsed time equals to 4.0. The 'high-profile' plot shows a solution profile at $y = 0.5$. The time integration is accomplished with $\Delta t = 0.01$,

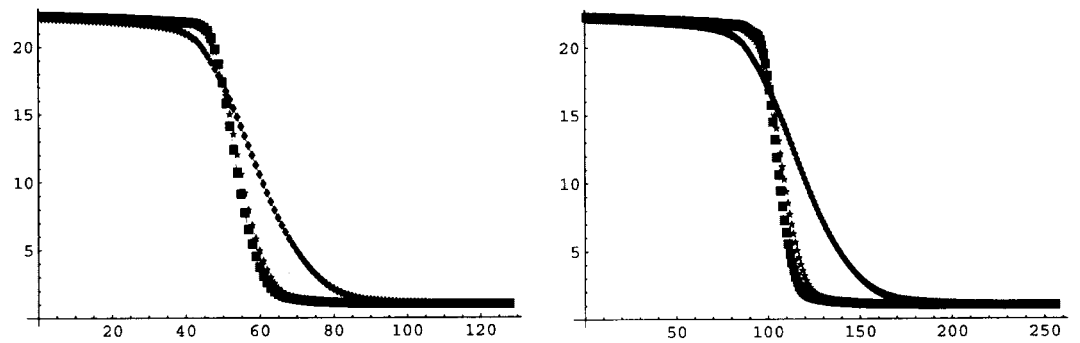

Figure 4. Comparative solution accuracy plots. Solution profiles at $y = 0.5$ are shown at $t = 4.0$. Obtained with $\Delta t = 0.01$ ($*$), 0.001($\bigstar$), and 0.0001(). The left figure uses a 128×128 mesh and the right figure uses a 256×256 mesh.

0.001 and 0.0001 on two meshes of size 128×128 and 256×256. Solutions obtained with the $\Delta t = 0.001$ and 0.0001 are very close to each other. Here we assume that a solution obtained with the smallest time step is the most accurate one. Our results show that time step of 0.01 is too inaccurate for our algorithm, and we choose $\Delta t = 0.001$ as the working time step size for our experiments. The solution is non-convergent in the sense that the starting point of the temperature wave (after a certain number of time iterations) is not spatially fixed. As we refine the mesh, the location of starting point changes and the shape of the wave becomes better resolved.

4.4. Parallel performance and scalability studies

In this subsection, we look at some machine-dependent properties of the algorithm. Our main concern is the scalability, which is the most important quality in evaluating a parallel algorithm. Indeed, if the scalability is poor, then the problems of realistic sizes may not be solved using a massively parallel approach. All CPU times reported here are obtained on an IBM SP2.

In Table I, we first study the scalability of the algorithm with respect to the number of processors. This kind of scalability shows how the domain partitioning (or the problem partitioning) affects the efficiency of the underlying solution methods. For this test, we use a fixed 1024×1024 mesh, and 4–256 processors. The overlapping size is fixed at $\text{ovlp} = 9$ for all partitions. It is clear that the non-linear iterations are completely independent of the number of subdomains, or the number of processors.

Throughout the experiments, we use box partitions. For example, in the case of 4 processors, all subdomains are squares with side lengths equal to 0.5. In the case of 8 processors, we allocate 2 processors in the x direction, and 4 in the y direction. Resulting rectangular subdomains have side lengths 0.5 and 0.25. It turns out the shortest side length has a great impact on the number of GMRES iterations. For instance, the shortest side length is reduced from 0.5 to 0.25, when we go from 4 to 8 processors, and the average number of GMRES iterations increases from about 60 to 108. However, when the number of processors goes from 8 to 16, the shortest side length is not changed. In this case, the average number of

Table I. Scalability with respect to the number of processors. 1024×1024 mesh. ILU factorization for all subproblems. The Schwarz preconditioner overlap ovlp $= 9$. Time step $\Delta t = 0.001$, 20 time steps. The problem is solved with 256, 128, 64, 32, 16, 8 and 4 processors. Average non-linear iterations are taken per time step. Average linear iterations are taken per non-linear iteration.

np	Total linear iterations	Average linear iterations	Total nonlinear iterations	Average nonlinear iterations
4	7123	59.9	139	7.1
8	12867	108.1	139	7.1
16	12812	107.7	139	7.1
32	20396	172.9	138	7.0
64	20392	172.8	138	7.0
128	31957	268.6	139	7.1
256	32201	272.9	138	7.0

GMRES iterations stays near a constant 108. This indicates that the shape of the subdomain has an impact on the number of linear iterations, but not at all on the non-linear iterations.

Ignoring the shape factor, if we look at the cases corresponding to 4, 16, 64 and 256 processors in Table I, we observe that the number of linear GMRES iterations increases as we increase the number of processors.

To further understand the linear scalability, we present the number of GMRES iterations with three different mesh sizes and ten different overlapping sizes in Table II. For a fixed mesh size and a fixed number of processors, the smallest number of iterations often corresponds to the large overlap case. This observation is quite different from the results obtained by solving Poisson's equation when small overlap is often as good as large overlap [16]. Let H be the diameter of the subdomain, from Table II, we can tell that even if we fix the ratio

$$\frac{H}{\text{ovlp}}$$

and refine the mesh from 1/256 to 1/512 to 1/1024, the number of GMRES iterations is not close to a constant. This is again very different from the estimates obtained with the usual additive Schwarz theory for elliptic equations [16] and for parabolic equations [14]. However, in practice, what really matters is the CPU time. Below we look at the CPU-time scalability of the algorithm.

Table III shows our results for the total number of non-linear iterations and the CPU-time as a function of the number of subdomains and the overlapping size. We fix the mesh to 1024×1024 and the time step size to $\Delta t = 0.001$. The total number of time steps is 20. Here again, we observe that the total number of non-linear iterations is practically independent of the number of processors and the overlapping size ovlp. Unlike the number of non-linear iterations, the CPU-time depends on both the subdomain size and the overlap. The most apparent observation with respect to the CPU-time data is that the optimal overlap size is not small. Indeed, for cases with 128, 64, 32, 16, 8 and 4 processors the best CPU-time results correspond to the overlap of 10. Table IV summarizes the results on how the total numbers of non-linear iterations change across meshes of different sizes. We see that the total number of non-linear iterations is nearly independent of the fine mesh size.

Table II. Average numbers of GMRES iterations. Three different meshes 256×256, 512×512 and 1024×1024. Overlapping size, ovlp varies from 1 to 10. The problem is solved with 256, 128, 64, 32, 16, 8 and 4 processors.

ovlp	np = 256	np = 128	np = 64	np = 32	np = 16	np = 8	np = 4
			256×256 mesh				
1	136.9	133.3	83.1	81.7	50.3	49.9	29.1
2	116.4	114.2	74.1	73.0	47.5	47.1	25.3
3	100.5	98.6	66.1	264.8	43.1	42.9	23.2
4	89.8	89.2	60.6	59.3	39.3	38.9	21.5
5	81.7	79.5	55.7	54.1	36.3	36.4	20.5
6	75.0	73.6	50.9	50.0	34.3	33.9	19.6
7	71.0	68.0	47.9	46.9	32.1	31.6	18.9
8	63.2	60.9	45.0	44.0	30.6	30.0	18.1
9	58.3	56.7	42.6	41.6	29.2	28.6	17.9
10	56.5	56.0	40.4	39.5	27.7	27.3	17.6
			512×512 mesh				
1	264.5	256.6	151.7	151.2	93.5	92.7	52.8
2	224.7	220.7	139.2	137.7	87.7	87.1	45.6
3	192.3	187.7	125.4	124.3	80.0	79.8	41.3
4	171.1	169.7	114.3	113.9	72.5	72.4	38.1
5	158.2	156.3	104.8	104.1	67.9	67.6	35.3
6	148.3	144.7	98.9	98.1	64.8	64.4	33.4
7	139.6	136.5	93.0	92.3	61.3	61.0	32.1
8	132.8	129.5	87.4	86.9	58.5	58.6	31.1
9	126.9	122.9	82.9	81.4	56.5	56.2	30.3
10	121.1	116.3	77.9	77.6	54.3	54.0	29.6
			1024×1024 mesh				
1	538.1	522.2	324.5	321.6	199.6	197.0	101.1
2	458.6	461.6	294.6	294.8	181.7	183.0	84.5
3	395.5	393.0	266.0	263.8	167.9	167.1	76.0
4	357.7	352.2	241.8	242.2	151.8	149.2	69.2
5	332.5	316.5	228.8	226.7	137.8	137.4	64.7
6	311.6	307.3	214.6	212.5	127.7	126.3	62.2
7	294.3	290.6	196.6	197.5	118.8	117.6	61.5
8	280.9	279.3	185.1	184.0	114.2	113.0	60.7
9	272.9	266.6	173.5	175.7	109.5	108.1	59.9
10	261.4	257.9	165.0	164.4	105.4	104.3	58.9

In Figure 5, we present some fixed mesh scalability results using two mesh sizes. In the left figure, we use the usual scalability factor

$$t(8)/t(\mathrm{np}) \tag{14}$$

where $t(8)$ denotes the CPU-time obtained with 8 processors and $t(\mathrm{np})$ denotes the CPU-time obtained with np processors. In the left figure we solve the problem on a 1024×1024 mesh with $\Delta t = 0.001$ and the overlap $\mathrm{ovlp} = 10$. The heights of the bars are calculated according to formula (14). In the figures the straight line indicates a linear dependence, or a linear speed-up. We observe that with up to 64 processors the results exhibit a superlinear speed-up.

Table III. Total number of non-linear iterations and CPU time in second 1024×1024 mesh. ILU factorization for all subproblems. The Schwarz preconditioner overlap ovlp varies from 2 to 10. The problem is solved with 256, 128, 64, 32, 16, 8 and 4 processors. Time step size is $\Delta t = 0.001$.

ovlp	np = 256	np = 128	np = 64	np = 32	np = 16	np = 8	np = 4
			Total number of non-linear iterations				
2	138	138	137	139	140	138	139
4	138	140	138	138	136	137	141
6	139	139	136	138	139	138	138
8	138	137	137	137	138	139	139
10	138	138	137	138	136	137	137
			CPU time (s)				
2	393	511	618	1540	3060	7325	7724
4	317	422	536	1331	2582	6192	6895
6	299	400	504	1207	2317	5478	6250
8	299	376	467	1103	2140	5047	6135
10	302	368	441	1048	2004	4736	5933

Table IV. Total number of non-linear iterations as a function of the subdomain and fine mesh sizes. $\Delta t = 0.001$, 20 time steps, ovlp = 10.

Mesh	np = 256	np = 128	np = 64	np = 32	np = 16	np = 8
256×256	123	123	123	123	123	123
512×512	150	150	150	150	150	150
1024×1024	138	138	137	138	136	137
2048×2048	123	123	123	123	123	123

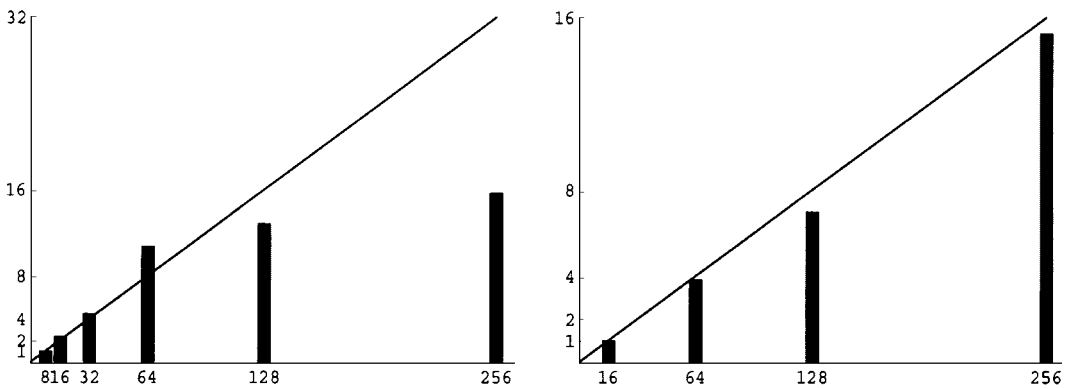

Figure 5. CPU time scalability. $\Delta t = 0.001$, 20 time steps, ovlp = 10. Numbers on the x-axis are the numbers of processors. Left figure: 1024×1024 mesh, bar height $= t(8)/t(\text{np})$. Right figure: 2048×2048 mesh, bar height $= t(16)/t(\text{np})$.

Table V. Parallel efficiency results with $\Delta t = 0.001$, 20 time steps, ovlp $= 10$, 1024×1024 mesh, speed-up $= t(8)/t(\text{np})$.

np	Speed-up	Parallel efficiency
8	1.00	100
16	2.36	118
32	4.52	113
64	10.75	134
128	12.86	80
256	15.69	49

Table VI. Parallel efficiency results with $\Delta t = 0.001$, 20 time steps, ovlp $= 10$, 2048×2048 mesh, speed-up $= t(16)/t(\text{np})$.

np	Speed-up	Parallel efficiency
16	1.00	100
64	3.84	96
128	7.00	87
256	15.27	95

However, starting with 128 processors, the algorithm no longer has its speed-up advantage. Corresponding parallel efficiency results are reported in Table V. Our working hypothesis is that the 1024×1024 mesh is not fine enough for the high number of processors. Thus, if we refine the mesh, the linear speed-up behaviour, will be restored. In the right figure we show results obtained on a 2048×2048 mesh. In this figure, the bar height is calculated using

$$t(16)/t(\text{np})$$

since the problem is harder to solve with 8 processors due to the memory limitation. We observe, that the linear speed-up is, indeed, restored on the finer mesh.

The right figure in Figure 5 shows that the CPU time scalability of the NKS-based solver is quite good, even though the number of linear iterations based scalability is disappointing according to Tables I and II. Table VI shows corresponding parallel efficiency measurements.

5. CONCLUSIONS

A parallel Newton–Krylov–Schwarz-based non-linearly implicit algorithm is developed and tested for a multimaterial radiation diffusion problem with a rather large influx boundary condition in a two-dimensional domain. The parallel software is constructed using PETSc,

and parallel results are obtained on an IBM SP system with up to 256 processors. The main findings of our study can be summarized as follows:

- The number of non-linear iterations is not sensitive (within 20%) to the mesh size. The number of non-linear iterations does not depend on the number of processors and the overlapping size.
- The number of GMRES iterations for solving the Jacobian systems increases as the number of subdomains increases or as the mesh is refined, no matter what the overlap is. This seems to indicate that the usual additive Schwarz theory does not apply to the radiation diffusion problem.
- The optimal overlapping size is not small.
- The algorithm/software is CPU-time scalable if the mesh is fine enough. This is a very important observation. It makes the method and the implementation suitable for solving problems on large meshes and with large number of processors.
- It does not look like a coarse grid is necessary to achieve the CPU time scalability. This suggests that the one-level NKS algorithm is potentially more attractive than multilevel methods especially in the case of unstructured grids since obtaining a coarse one is often difficult.

REFERENCES

1. Rider W, Knoll DA, Olson GL. A multigrid Newton–Krylov method for multimaterial equilibrium radiation diffusion. *Journal of Computational Physics* 1999; **152**:164–191.
2. Bowers RL, Wilson JR. *Numerical Modeling in Applied Physics and Astrophysics*. Jones and Bartlett: Boston, 1991.
3. Mihalas D, Mihalas BW. *Foundation of Radiation Hydrodynamics*. Oxford University Press: Oxford, 1984.
4. Brown PN, Woodward CS. Preconditioning strategies for fully implicit radiation diffusion with material-energy transfer. *SIAM Journal of Scientific Computing* 2001; **23**:499–516.
5. Knoll DA, Rider WJ, Olson GL. An efficient nonlinear solution method for non-equilibrium radiation diffusion. *Journal of Quantitative Spectroscopy and Radiative Transfer* 1999; **63**:15–29.
6. Knoll DA, Rider WJ, Olson GL. Nonlinear convergence, accuracy and time step control in non-equilibrium radiation diffusion. *Journal of Quantitative Spectroscopy and Radiative Transfer* 2001; **70**:25–36.
7. Mavriplis DJ. Multigrid approaches to non-linear diffusion problems on unstructured meshes. NASA/CR-2001-210660. *ICASE Report No*. 2001-3, NASA Langley Research Center.
8. Mousseau VA, Knoll DA, Rider WJ. Physics-based preconditioning and the Newton–Krylov methods for non-equilibrium radiation diffusion. *Journal of Computational Physics* 2000; **160**:743–765.
9. Stals L. Comparison of nonlinear solvers for the solution of radiation transport equations. *Electronic Transactions on Numerical Analysis* (*ETNAL*) 2003; **15**:78–93.
10. Cai X-C. Gropp WD, Keyes DE, Melvin RG, Young DP. Parallel Newton–Krylov–Schwarz algorithms for the transonic full potential equation. *SIAM Journal on Scientific Computing* 1998; **19**:246–265.
11. Balay S, Buschelman K, Gropp WD, Kaushik D, Knepley M, McInnes LC, Smith BF, Zhang H. *PETSc Users manual*, ANL-95/11—Revision 2.1.5, Argonne National Laboratory, 2002.
12. Coleman TF, Moré JJ. Estimation of sparse Jacobian matrices and graph coloring problem. *SIAM Journal on Numerical Analysis* 1983; **20**:187–209.
13. Saad Y. *Iterative Methods for Sparse Linear Systems*. PWS, Boston, MA, 1995.
14. Dennis JE, Schnabel RB. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM: Philadelphia, 1996.
15. Eisenstat S, Walker H. Globally convergent inexact Newton method. *SIAM Journal on Optimization* 1994; **1**:393–422.
16. Dryia M, Widlund O. Domain decomposition algorithms with small overlap. *SIAM Journal on Scientific Computing* 1994; **15**:604–620.
17. Cai X-C, Additive Schwarz algorithms for parabolic convection–diffusion equation. *Numerische Mathematik* 1991; **60**:41–62.
18. Smith B, Bjørstad P, Gropp W. *Domain Decomposition*: *Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press: Cambridge, MA, 1996.

