SPATIAL MULTIGRID FOR ISOTROPIC NEUTRON TRANSPORT*

B. CHANG†, T. MANTEUFFEL‡, S. MCCORMICK‡,
J. RUGE‡, S. AND B. SHEEHAN‡

Abstract. A spatial multigrid algorithm for isotropic neutron transport is presented in x-y geometry. The linear system is obtained using discrete ordinates in angle and corner balance finite differencing in space. Spatial smoothing is accomplished by a four-color block Jacobi relaxation, where the diagonal blocks correspond to four cell blocks on the spatial grid. A bilinear interpolation operator and its transpose are used for the grid transfer operators. Good convergence factors were observed for homogeneous material properties. Heterogeneous material properties prove more difficult, especially the case of a vacuum region surrounded by a thick, diffusive region. In this case, a small amount of absorption, or "effective absorption" in a time-dependent problem, restores good convergence. Numerical results are presented.

Key words. Neutral particle transport, multigrid, S_N approximation, finite difference discretization

AMS subject classifications. 65N06, 65N30, 65N55

1. Introduction.

The purpose of this paper is to present a spatial multigrid algorithm for isotropic neutron transport. The algorithm described here is an extention to two spatial dimensions of the work presented by Manteuffel et al. in [13] and [14]. The partial differential equation (PDE) to be solved is:

$$\Omega \cdot \nabla\psi + \sigma_t\psi - \frac{\sigma_s}{4\pi}\int_\Omega \psi \, d\Omega = q \qquad (x,y,\Omega) \in D \times S^2 \tag{1.1}$$

$$\psi = g \qquad (x,y) \in \partial D, \quad \mathbf{n}\cdot\Omega < 0. \tag{1.2}$$

$\psi = \psi(x,y,\Omega)$	the angular flux of neutrons
$\Omega \in S^2$	direction of neutron travel on unit sphere
$\sigma_t = \sigma_t(x,y)$	total macroscopic cross section
$\sigma_s = \sigma_s(x,y)$	scattering macroscopic cross section
$q = q(x,y,\Omega)$	interior source of neutrons
$g = g(x,y,\Omega)$	boundary source of neutrons
D, ∂D	unit square, boundary of the unit square
$\mathbf{n}$	outward unit normal

The primary numerical methodology currently in use for solving this PDE is source iteration combined with diffusion synthetic acceleration (DSA) [2, 4, 10, 12]. While DSA has achieved a great deal of success, it also suffers from certain limitations. One of these limitations is the sensitivity of the method to the spatial discretization [2]. In particular, it is difficult to implement DSA with the discontinuous finite difference

*Submitted to the SIAM Journal on Scientific Computing, May 30, 2006.

†Center for Applied Scientific Computation, Lawrence Livermore National Lab, Post Office Box 808 L-561, Livermore, CA, 94551. *email*: bchang@llnl.gov.

‡Department of Applied Mathematics, Campus Box 526, University of Colorado at Boulder, Boulder, CO, 80309–0526. *email*: tmanteuf@colorado.edu, stevem@colorado.edu, jruge@colorado.edu, and brendan.sheehan@colorado.edu. This work was sponsored by the Department of Energy under grant numbers DE-FC02-01ER25479 and DE-FG02-03ER25574, Lawrence Livermore National Laboratory under contract number B533502, Sandia National Laboratory under contract number 15268, and the National Science Foundation under DMS-0410318.

method known as *corner balance*. [1] Fast transport algorithms for the corner balance discretization are an area of active research interest. Other limitations of DSA stem from source iteration, the relaxation step on which it is based [12]. This relaxation is not ideally suited to parallel computing and does not allow for acceleration by spatial multigrid [13, 8, 15]. In contrast, the method outlined below employs the corner balance discretization, acceleration by spatial multigrid, and is well suited to parallel computing.

To explain these concepts further, it is necessary to first discretize the PDE in angle. The angular discretization used in this algorithm is known as discrete ordinates, or S_N [12]. The idea is simply that the PDE is enforced at only a finite number of angles, $\Omega_k \quad k = 1...d$. The angles are chosen so that the angular integral can be replaced by an appropriate quadrature rule, and (1.1)-(1.2) is then written as the following system of coupled PDE's in space. For $k = 1...d$:

$$\Omega_k \cdot \nabla \psi_k + \sigma_t \psi_k - \frac{\sigma_s}{4\pi} \sum_{m=1}^{d} w_m \psi_m = q_k \qquad (x,y)\ \epsilon\ D \tag{1.3}$$

$$\psi_k = g_k \qquad (x,y)\ \epsilon\ \partial D, \quad \mathbf{n} \cdot \Omega_k < 0. \tag{1.4}$$

$\Omega_k\ ,\ w_k$	given direction, quadrature weight
$\psi_k\ ,\ q_k\ ,\ g_k$	$\psi(x,y,\Omega_k)\ ,\ q(x,y,\Omega_k)\ ,\ g(x,y,\Omega_k)$

The spatial discretization (1.3)-(1.4) is discussed in the next section. For now, assume the problem has been spatially discretized, and write the discrete problem as

$$[\mathbf{D} + \mathbf{\Sigma} - \mathbf{S}]\, \mathbf{\Psi} = \mathbf{Q}, \tag{1.5}$$

where the three terms in brackets represent the discrete versions of the three terms in (1.3). Notice that $\mathbf{D} + \mathbf{\Sigma}$ contains local spatial coupling of the unknowns, and $\mathbf{S}$ contains the angular coupling. Since problems of interest are too large to be solved directly, an iterative approach based on a matrix splitting is used. In the case of transport sweeps, $\mathbf{S}$ is split from the rest of the matrix and the iteration becomes

$$[\mathbf{D} + \mathbf{\Sigma}]\, \mathbf{\Psi^{i+1}} = [\mathbf{S}]\, \mathbf{\Psi^{i}} + \mathbf{Q}, \tag{1.6}$$

which implies that the error evolves according to

$$\mathbf{e^{i+1}} = [\mathbf{D} + \mathbf{\Sigma}]^{-1}\, [\mathbf{S}]\, \mathbf{e^{i}}. \tag{1.7}$$

This step is computationally inexpensive because $[\mathbf{D} + \mathbf{\Sigma}]$ is triangular, or block triangular, depending on the discretization. However, calculating the action of the inverse of a triangular matrix on a vector is an essentially sequential operation, and this is

[1] DSA is implemented in [18] for a discretization similar to corner balance. The authors indicate that a fully consistent implementation is computationally expensive, but that similar results can be obtained when an inconsistent DSA is used as a preconditioner for a Krylov method

why transport sweeps are not ideally suited to parallel computing. [2] To see why transport sweeps do not allow for acceleration by spatial multigrid, consider the case where $\sigma_s \approx \sigma_t$ and $\sigma_t h \gg 1$, where h is the cell width. In this parameter regime, which is common in real world problems, $[\mathbf{D}+\mathbf{\Sigma}]^{-1} \approx [\mathbf{\Sigma}]^{-1}$. Now imagine an error, $\mathbf{e^i}$, that is independent of angle, so that $\mathbf{Se^i} \approx \mathbf{\Sigma e^i}$. Equation (1.7) now shows $\mathbf{e^{i+1}} \approx \mathbf{e^i}$. Errors that are constant in angle are not significantly changed by transport sweeps in this parameter regime, regardless of their spatial form. Since these errors have no predictable form in space after relaxation, a spatial coarse-grid correction cannot be used to accelerate convergence.

The algorithm described below is based on a different relaxation step, which is well suited to parallel computation, and leaves error with a predictable spatial form. The remainder of this paper is organized as follows. Section two describes the spatial discretization, and section three gives the details of the relaxation step. Sections four outlines the multigrid algorithm. Sections five and six give numerical results for homogeneous and heterogeneous domains. Section seven examines the computaional cost of the method. Sections 8 and 9 compare the parallel scalability and performance of the method to that of DSA, and section 10 contains a brief conclusion.

2. Spatial Discretization. As mentioned above, the spatial discretization used in this algorithm is the corner balance finite difference method [3]. The derivation of corner balance is contained in [1]. In this context, it suffices to say that in the case of square cells the solution has four spatial degrees of freedom per cell. Figures 2.1 and 2.2 illustrate the location of the spatial unknowns and a random example solution on a four cell grid. It is important to note that these figures represent the grid and solution in only one of the directions.

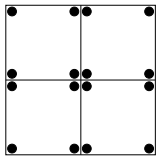


FIG. 2.1. *Location of spatial unknowns*

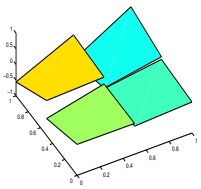
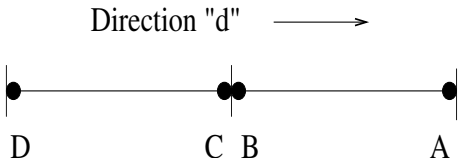


FIG. 2.2. *Bilinear discontinuous example solution*

[2] A domain decomposition method for transport sweeps is described in [19]. This method performs sweeps on local subdomains and avoids the global sequential solve. While this improves the parallel scalability of sweeps, it also significantly changes the performance of the associated DSA algorithm. For the duration of this paper, all references to the performance and parallel scalability of DSA assume an algorithm based on global sweeps.

[3] The spatial discretization used in this algorithm is *simple* corner balance, not *upwind* corner balance, both of which are derived in [1].

To describe the relaxation step in section 3, it is necessary to examine the corner balance approximation of a first derivative. The nature of the corner balance approximation of the derivative terms in (1.3) is best explained with a one-dimensional, two-cell example:

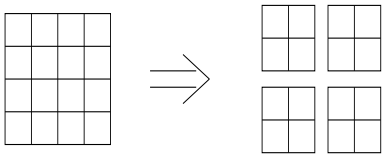


$$\text{At point A in direction d:} \qquad \frac{d\psi}{dx} \approx \frac{\psi_A - \psi_B}{h} \tag{2.1}$$

$$\text{At point B in direction d:} \qquad \frac{d\psi}{dx} \approx \frac{\frac{\psi_A+\psi_B}{2} - \psi_C}{\frac{h}{2}} = \frac{\psi_A + \psi_B - 2\psi_C}{h} \tag{2.2}$$

Equations (2.1) and (2.2) contain two kinds of spatial coupling. Equation (2.1) contains *within-cell* coupling (A to B), while (2.2) contains both within-cell and *between-cell* coupling (A and B to C). The distinction between these two categories of spatial coupling is the basis for the matrix splitting described below.

3. Relaxation Method. Imagine splitting the two-dimensional domain into 4-cell blocks:



The matrix $\mathbf{D}$ is split as follows: $\mathbf{D} = [\mathbf{D_{wb}} + \mathbf{D_{bb}}]$, where $\mathbf{D_{wb}}$ contains the *within block* coupling (elements of $\mathbf{D}$ which couple unknowns in a given block to other unknowns in that same block) and $\mathbf{D_{bb}}$ contains the *between block* coupling (elements of $\mathbf{D}$ which couple unknowns in a given block to unknowns in a separate block). This splitting gives the new relaxation step

$$[\mathbf{D_{wb}} + \mathbf{\Sigma} - \mathbf{S}]\, \mathbf{\Psi^{i+1}} = [-\mathbf{D_{bb}}]\, \mathbf{\Psi^{i}} + \mathbf{Q}. \tag{3.1}$$

The inversion of $[\mathbf{D_{wb}} + \mathbf{\Sigma} - \mathbf{S}]$ is straightforward because, with the appropriate ordering of the unknowns, it is block diagonal and represents a set of unrelated 4-cell transport problems. From this perspective, it is clear that this relaxation is a form of block-Jacobi. Since each 4-cell problem can be handled by a different processor, the step is well suited to parallel computing. Equation (3.1) implies that the unknowns corresponding to all the directions within a 4-cell block are solved for simultaneously. This means that the size of the matrix corresponding to each 4-cell block grows with the size of the quadrature set being used. However, since the scattering operator is

a low rank perturbation of the streaming-collision operator, the Sherman-Morrison-Woodbury formula allows the 4-cell inversions to be accomplished at a cost that rises only linearly with the number of angles. This formula, and a more detailed discussion of computational cost, are included in section five.

This relaxation step is designed to spatially smooth the error in the thick diffusive regime where $\sigma_s \approx \sigma_t$ and $\sigma_t h \gg 1$. In order to see how the smoothing is accomplished, consider applying equation (3.1) in a diffusive material, with a zero right-hand-side and random initial guess. The first iteration calls for the solution of many unrelated 4-cell problems, each with a zero interior source and random inflow boundary conditions. Intuition suggests that in the continuous case, the solution would be random near the boundary, but smooth away from the boundary. Therefore, in the discrete 4-cell case, we expect the solution to have the four spatial unknowns at the center of the domain roughly equal. Figures 3.1 and 3.2 show the corner balance solution to a problem with zero interior source, random inflow, and $\sigma t = \sigma_s = 1e4$. Figure 3.1 is a domain with 64 cells, and is intended to be an approximation to the continuous case. Figure 3.2 is a 4 cell domain, and shows that the center variables are roughly equal. Once again, these plots show the solution in only one of the directions.

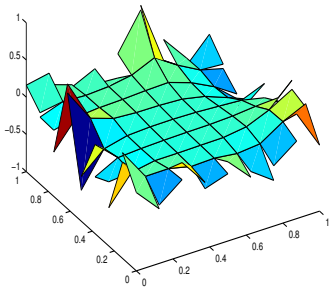


FIG. 3.1. *Zero source, random inflow on 64 cells with* $\sigma t = \sigma_s = 1e4$

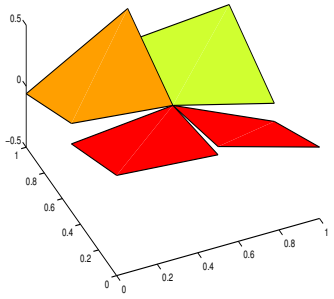


FIG. 3.2. *Zero source, random inflow on 4 cells with* $\sigma t = \sigma_s = 1e4$

After one relaxation, the error is no longer random. At the center of each 4-cell block, the four spatial errors are roughly equal in any given direction. Therefore,

the error is now "partially continuous". This relaxation step is the extension to two spatial dimensions of the relaxation used by Manteuffel et al. [13, 14], and has been presented and analyzed in two spatial dimensions by Lansrud [9]. However, the error is not spatially smoothed unless the 4-cell blocks are shifted, and the block-Jacobi step is repeated, a total of four times. If the arrangement of the 4-cell blocks is now shifted by one cell, a second relaxation step makes the error continuous at the centers of the new 4-cell blocks. This process of shifting the arrangement of the four cell blocks is repeated, and a total of four different arrangements are used. The different four cell block patterns are illustrated in figure 3.3. The iterate is updated before the blocks are shifted, so each of the arrangements below represents a separate block-Jacobi iteration.

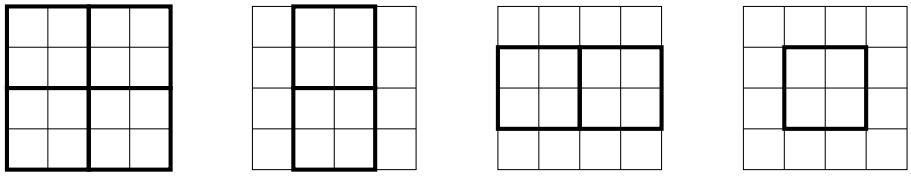


FIG. 3.3. *4-cell block pattern*

At the end of the complete Red-Black-Orange-White (RBOW) pass, every vertex on the original grid has been at the center of a 4-cell solve once, and the error is nearly continuous over the entire domain. In addition to becoming nearly continuous, the error also tends to become smooth over the entire domain and bilinear over the four cell blocks defined by the last color pass. This happens because the centerlines of the 4-cell blocks in a given color pass become the boundary conditions for the blocks in the next color pass. Continuous centers become continuous boundary conditions, and continuous boundary conditions produce smoother solutions. Smoother solutions become smoother boundary conditions, and the cycle pushes the error into a form that is not only continuous, but is increasingly smooth and bilinear over the blocks of the last pass.

In practice, it is not necessary to include all spatial points within a 4-cell block in the block solve. The computational cost can be reduced by including only the points which are spatially coupled to the points at the center of the block. Figure 3.4 shows the *full block* on the left and the *reduced block* on the right. Only the 12 points along the"plus sign" of the 4-cell block are included in the reduced block [4]. For the remainder of this paper, the term "4-cell inversion" is taken to mean that the unknowns at the spatial locations of the reduced block, and in all the directions, are solved for simultaneously. Furthermore, the term "RBOW relaxation" implies that the reduced block is being used.

Figure 3.5 shows the evolution of random error on a 256 cell grid, where $\sigma_t = \sigma_s = 10^4$, over 5 RBOW relaxations. The important feature of the RBOW step is not that it reduces the size of the error, but that it puts the error in a predictable spatial form. The desired spatial form is bilinear over the "red" blocks. This allows

[4]In three spatial dimensions, there would be 64 spatial locations in an 8-cell block. However, only 32 of these would lie along the "three dimensional plus sign", making the cost savings of using reduced blocks more dramatic than it is in two spatial dimensions.

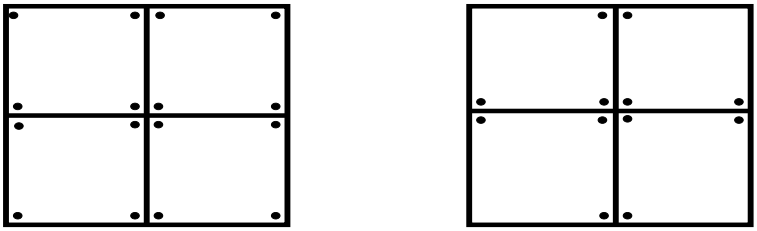


FIG. 3.4. *Spatial unknowns in the full and reduced block*

a coarse-grid correction based on bilinear interpolation. To leave the error bilinear over the red blocks, the order of the color pass is rearranged so that the red pass is last. In figure 3.5, it is easy to see that the error is nearly bilinear over the red blocks after 5 RBOW relaxations and a coarse-grid correction would be very effective at this point. Experiments have shown, however, that the coarse-grid correction is surprisingly effective after just one relaxation.

4. Multigrid Algorithm. Since the error after RBOW relaxation is close to being bilinear over the red blocks, it can be approximately represented on a coarse grid. To accomplish this, it is necessary to define the coarse grid, the fine grid, and grid transfer operators that map vectors from one grid to the other. Consider a fine grid with a total of N unknowns, and a coarse grid on which the fine-grid red blocks are single cells. This coarse grid has a total of $\frac{N}{4}$ unknowns. The *interpolation* operator, $\mathbf{P}$, is an $N \times \frac{N}{4}$ matrix that maps coarse vectors, denoted by $\mathbf{v^{2h}}$, to fine vectors, denoted by $\mathbf{v^h}$ (the notation h and 2h represent a doubling of the cell width). The interpolation is bilinear, meaning the spatial unknowns on any coarse cell, which are bilinear by definition, are mapped to the same bilinear over the corresponding fine-grid red block. The *restriction* operator $\mathbf{R}$ is simply the transpose of $\mathbf{P}$ up to a constant. This restriction operator is often called *full weighting*.

For the purpose of illustrating the concept of coarse-grid correction, assume that relaxation leaves the error exactly bilinear over the red blocks. This means the fine error lies in the range of bilinear interpolation, and we can write $\mathbf{e^h} = \mathbf{P e^{2h}}$. The coarse-grid correction then proceeds as follows:

- After relaxation, calculate the fine residual, $\mathbf{r^h} = \mathbf{b} - \mathbf{A\Psi^i}$
  ($\mathbf{A}, \mathbf{b}, \mathbf{\Psi^i}$ represent the entire discrete transport operator, the discrete right-hand-side, and the current iterate, respectively).
- Write the residual equation $\mathbf{A e^h} = \mathbf{r^h}$
  and use the assumption above to get $\mathbf{APe^{2h}} = \mathbf{r^h}$.
- Solve the $\frac{N}{4} \times \frac{N}{4}$ system $[\mathbf{RAP}]\,\mathbf{e^{2h}} = \mathbf{Rr^h}$.
- Correct the current iterate: $\mathbf{\Psi^{i+1}} = \mathbf{\Psi^i} + \mathbf{Pe^{2h}}$.

In the ideal situation outlined above, the coarse-grid correction would be perfect and the problem would be solved exactly. In reality, the coarse-grid correction is not perfect because the fine error is not exactly in the range of interpolation. The idea is that if the fine error is close to the range of interpolation, then the coarse-grid correction improves the current iterate at a lower computational cost. There are, of course, several other important differences between the ideal example above and the actual algorithm. Since an $\frac{N}{4} \times \frac{N}{4}$ system is still far too large to be solved directly,

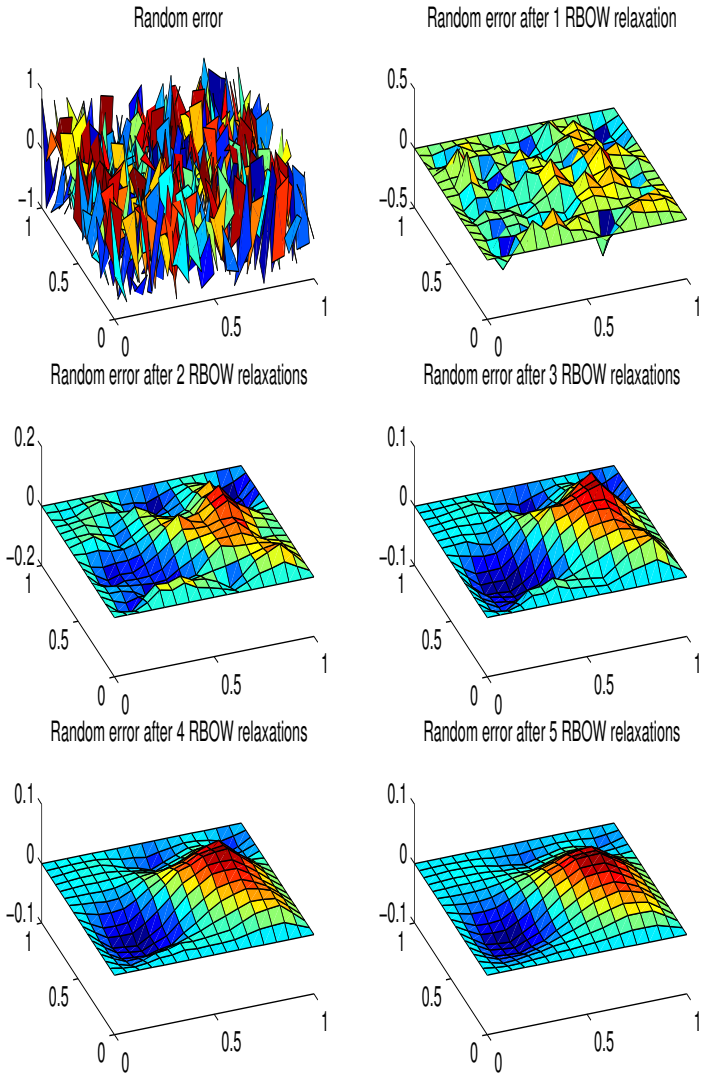


FIG. 3.5. *Evolution of random error*

the procedure is applied recursively. The exact solve does not take place until the coarsest grid, which can be as small as four cells total. The sequence of relaxations and coarse-grid corrections is called a V-cycle [7]. A V-cycle can feature any given number of relaxation steps, called pre-relaxations, on each grid level as the grid is being coarsened. It is also common to include some number of additional relaxations, called post-relaxations, after each coarse-grid correction. The optimal number of pre and post-relaxations to use is often determined experimentally. The notation V(n,m) indicates a V-cycle with n pre-relaxations and m post-relaxations. In summary, the spatial multigrid algorithm is a V-cycle with the following features:

- RBOW relaxation
- Bilinear interpolation operator ($\mathbf{P_{BL}}$)
- Restriction is full weighting ($\mathbf{R_{FW}}$)
- Coarse-grid operator is $\mathbf{R_{FW}AP_{BL}}$

Notice that the operator for the coarse-grid problem is $\mathbf{R_{FW}AP_{BL}}$. When the corner balance transport operator is multiplied on the left and right by $\mathbf{R_{FW}}$ and $\mathbf{P_{BL}}$, respectively, its stencil is altered. The evolution of this stencil on coarser and coarser grids is fairly easy to track, and heads rapidly to a limit that is equivalent to the bilinear discontinuous Galerkin finite element discretization. The bilinear discontinuous Galerkin discretization is very similar to corner balance [1, 2, 3, 16], and the sequence of intermediate stencils also have a similar character. Since the evolution of the stencil is not dramatic, RBOW relaxation is an appropriate smoother for the coarse-grid problems [5].

5. Results For Problems With Homogeneous Material. The method described above has been coded and tested in two spatial dimensions. In this section, convergence factors are given for the zero right-hand-side problem, with zero boundary conditions and random initial guess. The convergence factor is measured by taking a ratio of the vector two-norm of the residual before and after a given V-cycle:

$$\rho^{i+1} = \frac{\|\mathbf{r^{i+1}}\|_2}{\|\mathbf{r^i}\|_2}. \tag{5.1}$$

In most cases, the convergence factor settles to a nearly constant value after a few V-cycles. For consistency, the factors given below were calculated by running fifteen V-cycles and taking the geometric mean of the last five.

$$\rho = \left(\prod_{i=11}^{15} \rho^i \right)^{\frac{1}{5}}. \tag{5.2}$$

All cycles are V(1,1) cycles. In all cases, the domain is the unit square, which is divided into 4096 square cells. Four directions are used, giving a total of 65,536 unknowns. Table 5.1 gives results for homogeneous, pure scattering domains. In all cases, the results improve with absorption ($\sigma_s < \sigma_t$).

TABLE 5.1
Convergence factors for domains with homogeneous material and pure scattering.

$\sigma_t h$	ρ
10^4	.06
1	.03
10^{-4}	.1

[5] The bilinear discontinuous Galerkin (BLDG) stencil is not altered by the coarsening process, i.e. $\mathbf{R_{FW}A^h_{BLDG}P_{BL}} = \mathbf{A^{2h}_{BLDG}}$. The algorithm described here can be applied to BLDG instead of corner balance with very similar results.

In the case of the thick and medium problem (top two lines), the convergence factors in table 5.1 do not depend on problem size; using more cells or more directions has no effect on the convergence. The same cannot be said for the thin problem (bottom line). Due to the nature of the thin operator, successive V-cycles create a "marching-style" near exact solve in a pure thin domain. Once the marching process crosses the domain, there is a large drop in the residual. For this reason, thin convergence factors on domains with few cells are better than they would be on domains with many cells. Also, increasing the number of directions degrades the thin convergence factor because, as directions become more grid-aligned, bilinear interpolation is less appropriate for thin regions. While the pure thin problem by itself is not of much interest, it is important to understand how the method might behave in thin regions of domains with heterogeneous material. Domains of this type are discussed further in the next section.

6. Results For Problems With Heterogeneous Material. This method has been tested on a variety of heterogeneous domains. The domains and coarse grids are set up in such a way that there is no "mixed coarsening". In other words, the case of a 4-cell block composed of multiple materials being coarsened into a single cell does not arise. The idea is to deal with the difficulties of a heterogeneous domain without the added complication of mixed coarsening [6].

In some cases, the convergence factors for heterogeneous domains are no worse than those for homogeneous domains. The most notable exception to this rule is the case of a thin region surrounded by a thick region, as indicated in figure 6.1. Table 6.1 and 6.2 give the convergence factors for this domain with several variations to the cross sections. In table 6.1, both regions are pure scattering and the values of $\sigma_t h$ are listed for the thick(thin) regions, respectively. While the convergence is still good for fairly large jumps in cross section, it is clear that performance deteriorates completely as the thin region approaches a true vacuum. In table 6.2, the thin region is always a true vacuum, while the thick region always has $\sigma_t h = 10^4$, with some absorption, as indicated by the scattering ratio ($scat.\ ratio = \frac{\sigma_s}{\sigma_t}$). Clearly, the convergence is improved with the introduction of a small amount of absorption.

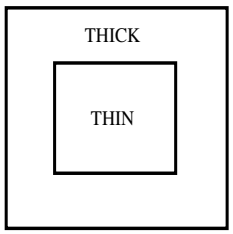


FIG. 6.1. *Heterogeneous model domain*

It is important to note that a small amount of "effective absorption" will always be present if the steady state problem is being solved at each time step of a time dependent-problem with an implicit time discretization. To see this, let c be the particle speed, and consider the time-dependent problem

[6] Preliminary experiments with mixed coarsening show that the convergence is not adversely affected, provided the bilinear interpolation is replaced with a kinked-linear interpolation similar to that used by Manteuffel et al. in [14].

TABLE 6.1

Convergence factors for the heterogeneous model domain with a thin pure scattering center and a thick pure scattering perimeter.

$\sigma_t h$ thick (thin)	ρ
10^4 (10^4)	.06
10^4 (10^{-1})	.11
10^4 ($5 * 10^{-2}$)	.23
10^4 (10^{-2})	.61
10^4 (0)	.97

TABLE 6.2

Convergence factors for the heterogeneous model domain with vacuum center and thick perimeter. The thick region always has $\sigma_t h = 10^4$ with some absorption, as indicated by the scattering ratio.

$scat.\ ratio = \frac{\sigma_s}{\sigma_t}$	ρ
.999	.09
.9999	.24
.99999	.63
1	.97

$$\frac{1}{c}\frac{\partial \psi}{\partial t} + \Omega \cdot \nabla \psi + \sigma_t \psi - \frac{\sigma_s}{4\pi}\int_\Omega \psi \, d\Omega = q. \tag{6.1}$$

A backward Euler approximation in time yields

$$\frac{1}{c}\frac{\psi^{n+1} - \psi^n}{\Delta t} + \Omega \cdot \nabla \psi^{n+1} + \sigma_t \psi^{n+1} - \frac{\sigma_s}{4\pi}\int_\Omega \psi^{n+1} \, d\Omega = q^{n+1}, \tag{6.2}$$

or

$$\Omega \cdot \nabla \psi^{n+1} + (\sigma_t + \frac{1}{c\Delta t})\psi^{n+1} - \frac{\sigma_s}{4\pi}\int_\Omega \psi^{n+1} \, d\Omega = q^{n+1} + \frac{1}{c\Delta t}\psi^n. \tag{6.3}$$

The effective absorption cross section is $\sigma_a^* \geq (\frac{1}{c\Delta t})$. This parameter prevents any region from being either pure vacuum or pure scattering. Larsen [11] details a method for time and energy-dependent problems that requires the solution of an equation similar to (6.3) in the inner iteration. The algorithm described here has been applied to (6.3) for the model domain with pure vacuum and pure scattering ($\sigma_t h = 10^4$) materials. Naturally, convergence improves with larger values of $\frac{1}{c\Delta t}$. Table 6.3 shows the convergence factors for various values of $\frac{1}{c\Delta t}$ from 0 to nx, where nx is the number of cells in a single dimension (i.e. in 2-D on a 64x64 grid nx=64). The value of $\frac{1}{c\Delta t}$ must be proportional to nx for the results to be independent of grid size. This is because it is the amount of absorption *per cell width* that affects the convergence of the method. The values listed in parentheses are for a W-cycle, which requires slightly more work than a V-cycle [7].

TABLE 6.3
Convergence factors for the heterogeneous model domain with vacuum center, thick pure scattering perimeter and time-step-based effective absorption. The thick region always has $\sigma_t h = 10^4$. Parentheses indicate convergence factor for a W-cycle instead of a V-cycle.

$\frac{1}{c\Delta t}$	ρ
nx	$7 * 10^{-4}$
$\frac{nx}{10}$	.12
$\frac{nx}{100}$	.56 (.38)
$\frac{nx}{1000}$	.88 (.69)

7. Computational Cost. To put the performance of the method in perspective, it is necessary to discuss computational cost. As mentioned previously, all 4-cell inversions in the relaxation step are accomplished via the Sherman-Morrison-Woodbury formula:

$$(\mathbf{A} + \mathbf{U}\mathbf{V}^{\mathbf{T}})^{-1} = \mathbf{A}^{-1} - \left[\mathbf{A}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\right]. \tag{7.1}$$

To see how this formula applies to the discrete transport operator, let $\mathbf{A} = \mathbf{D} + \mathbf{\Sigma}$ be the streaming-collision operator and let $-\mathbf{S} = \mathbf{U}\mathbf{V}^{\mathbf{T}}$ be the scattering operator, where

$$\mathbf{U} = -\begin{bmatrix} \mathbf{\Sigma_s} \\ \mathbf{\Sigma_s} \\ \vdots \\ \mathbf{\Sigma_s} \end{bmatrix}, \tag{7.2}$$

and

$$\mathbf{V}^{\mathbf{T}} = \frac{1}{4\pi}\begin{bmatrix} w_1\mathbf{I} & w_2\mathbf{I} & \cdots & w_d\mathbf{I} \end{bmatrix}. \tag{7.3}$$

In equation (7.3), w_i is a quadrature weight and $\mathbf{I}$ is an $N_{sp} \times N_{sp}$ identity, where N_{sp} is the number of spatial unknowns in the domain. In equation (7.2), $\mathbf{\Sigma_s}$ is a diagonal matrix of size $N_{sp} \times N_{sp}$, with the values of σ_s at each spatial point in the domain on the diagonal. As before, d is the number of directions, so the total number of unknowns in the problem is $N = N_{sp} * d$, and the sizes of $\mathbf{U}$ and $\mathbf{V}^{\mathbf{T}}$ are $N \times N_{sp}$ and $N_{sp} \times N$, respectively. Notice that $\mathbf{V}^{\mathbf{T}}$ is the discrete integration operator, and $\mathbf{V}^{\mathbf{T}}\mathbf{\Psi} = \mathbf{\Phi}$, where $\mathbf{\Phi}$ is the scalar flux. Now that the discrete system can be written as $(\mathbf{A} + \mathbf{U}\mathbf{V}^{\mathbf{T}})\mathbf{\Psi} = \mathbf{q}$, consider the following algebraic manipulations:

- Multiply $(\mathbf{A} + \mathbf{U}\mathbf{V}^{\mathbf{T}})\mathbf{\Psi} = \mathbf{q}$ by $\mathbf{A}^{-1}$ and rearrange to get (7.4).
- Multiply (7.4) by $\mathbf{V}^{\mathbf{T}}$ and rearrange to get (7.5).

$$\mathbf{\Psi} = \mathbf{A}^{-1}\left[\mathbf{q} - \mathbf{U}\mathbf{\Phi}\right] \tag{7.4}$$

$$(\mathbf{I} + \mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{U})\mathbf{\Phi} = \mathbf{V}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{q} \tag{7.5}$$

In practice, the Sherman-Morrison-Woodbury formula is applied by solving (7.5), and then solving (7.4). The cost of this process is twice the cost of calculating the action of $\mathbf{A}^{-1}$ on a vector, plus the cost of forming and inverting $(\mathbf{I} + \mathbf{V^T A^{-1} U})$. By calculating the cost of a single 4-cell inversion this way, the cost of a complete RBOW relaxation can easily be obtained. To minimize the cost per iteration, an LU factorization of the matrix $(\mathbf{I} + \mathbf{V^T A^{-1} U})$ corresponding to each 4-cell block is formed and stored in a separate setup phase. To facilitate the calculation of the action of $\mathbf{A}^{-1}$, LU factorizations of the diagonal blocks of $\mathbf{A}$ are also stored during the setup phase. Table 7.1 gives both the computational cost and storage requirement of one fine-grid RBOW relaxation, including the setup phase. One *computational unit* (CU) is the number of operations necessary to compute the fine-grid residual, and one *storage unit* (SU) is the memory required to store the current iterate. Notice the cost and storage requirement for the setup phase are given as upper bounds. The setup costs 36 CU only if every cell in the domain has a different size and/or cross section. Regions of identical cells significantly reduce the setup cost. Similarly, the setup storage is 11 SU only in the case of a minimal quadrature set (4 directions in 2-D). Using more directions reduces the setup storage requirement in terms of SU's. The range of 4-6 CU for the iteration cost is also a function of the quadrature set. Again, if a minimal set of 4 angles is chosen, the iteration cost is 6 CU. This value decreases to 4 CU as the number of angles rises [7].

TABLE 7.1
Computational cost of one RBOW relaxation including setup phase. One CU is the cost of computing the fine-grid residual. One SU is the memory required to store the current iterate.

phase	operations	storage
setup	$\leq 36 \quad CU$	$\leq 11 \quad SU$
iteration	$4-6 \quad CU$	$1 \quad SU$

The cost of the entire V-cycle can easily be obtained based on the cost of one fine-grid relaxation [7]. Let the cost of one fine-grid RBOW relaxation be one work unit (WU). Neglecting the cost of intergrid transfers (10% to 20% of total), the cost of a V(1,1) cycle is

$$2(1 + 2^{-2} + 2^{-2*2} + ...2^{-n*2}) < \frac{2}{1 - 2^{-2}} = \frac{8}{3}\, WU. \tag{7.6}$$

Using table 7.1 and equation (7.6), the approximate cost of a V(1,1) cycle in CU is $\frac{8}{3}(4 - 6\ CU) \approx (11 - 16\ CU)$.

8. Parallel Scalability. The purpose of this section is to compare the time required to complete one RBOW relaxation to the time required to complete one global transport sweep on a parallel machine. In terms of the computational units defined above, a transport sweep requires only 1 CU while the RBOW relaxation requires 4-6 CU. Therefore, on a serial machine, the RBOW relaxation takes 4-6 times as long as a sweep. However, the picture is different on a parallel machine. In

[7] The cost and storage requirements of the algorithm have a component that is independent of the number of angles, which is trivialized as the number of angles rises. This is why the values in the table are higher for fewer angles, in terms of CU/SU.

order to make a time-to-completion comparison, it is necessary to outline a model for the way in which information propagates through the processor grid during a sweep. First, imagine conducting a sweep in only one of the S_N directions, on a domain with only one spatial dimension, using a machine with P processors. The first processor, starting with the inflow boundary conditions, must perform its calculations and send the results to the next processor. Define the time required to complete this number of operations to be one single processor time (SPT). The time required to complete the sweep in this single S_N direction is simply P single processor times. In two spatial dimensions, the time required to complete a sweep in a single S_N direction is $2P^{\frac{1}{2}}$ SPT. This is because, if one imagines the P processors to be arranged in a $P^{\frac{1}{2}}$ by $P^{\frac{1}{2}}$ array, the information can be passed through the processor grid on"diagonal bands" in a total of approximately $2P^{\frac{1}{2}}$ steps, as shown in figure 8.1.

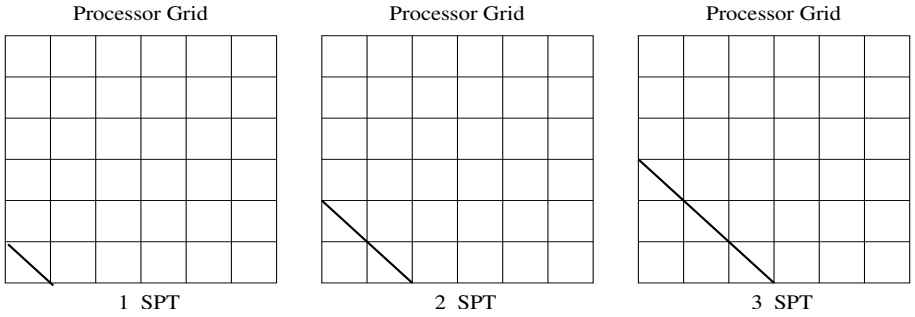


FIG. 8.1. *Transport sweep in one S_N direction and two spatial dimensions through a processor grid. One Single processor time (SPT) is the time required for one processor to complete its computations.*

In general, if i is the number of spatial dimensions, the time required to complete the sweep in a single S_N direction is $iP^{\frac{1}{i}}$ SPT. Now consider performing the sweep in all the S_N directions instead of just one. Returning to the 1-D example, once the first processor has completed its calculations and passed the results on to the second processor, it is free to begin a sweep in a second S_N direction. Naturally, this is true with more spatial dimensions as well, and the sweeps for the various S_N directions can follow each other through the processor grid like ducks in a row, as shown in figure 8.2. If it were possible to begin all of the directional sweeps from the same starting

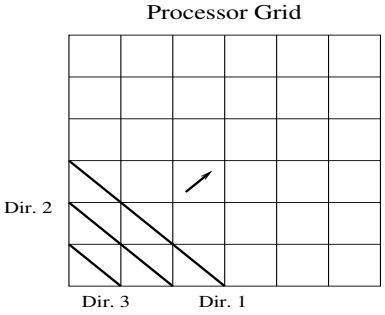


FIG. 8.2. *Transport sweep in multiple S_N directions and two spatial dimensions. The directional sweeps follow each other through the processor grid.*

point in the processor grid, it is easy to see that the total time required to complete

a sweep in M directions is $[iP^{\frac{1}{i}} + M]$ SPT. In reality, in 1-D, half the directional sweeps must begin from the other side of the processor grid and in 2-D, one quarter of the directional sweeps begin in each of the four corners of the processor grid, as shown in figure 8.3. This reduces the number of "ducks" in each row, but also creates

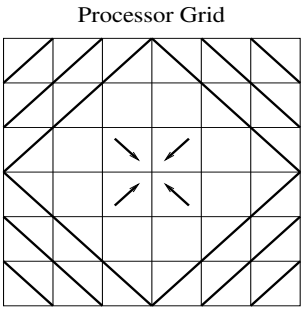


FIG. 8.3. *Transport sweep in multiple S_N directions and two spatial dimensions. Directional sweeps are initiated from the appropriate place in the processor grid.*

a delay when processors begin to receive information from multiple directions. A careful analysis reveals that the total time-to-completion is still $[iP^{\frac{1}{i}} + M]$ SPT even when the various directional sweeps are initiated from the appropriate place in the processor grid.

The number of SPT's required to complete one RBOW relaxation can be obtained more easily by using the serial result given above. In parallel, the number of operations each processor must do to complete its share of the RBOW step is simply (4-6) CU_{local}, where one CU_{local} is the number of operations required to compute the portion of the residual that one processor is responsible for. This is roughly the same number of operations that would be required to do a sweep in ALL directions on that processor's portion of the domain, and therefore requires M single processor times, where again, M is the total number of S_N directions. Therefore, the time to complete the RBOW step in parallel is (4-6)M SPT, regardless of the number of processors being used. [8]

The time-to-completion models described above consider only the time required to complete the necessary operations. However, the time required to complete the processor-to-processor communications might also be significant, and the nature of that communication is quite different for the RBOW step than it is for a transport sweep. While numerical tests on a parallel machine are beyond the scope of this paper, it is worthwhile to give a brief outline of the different communication requirements. Basic models of processor communication time describe the total time required to pass a message containing k units of data as [17]:

$$t_l + kt_b \tag{8.1}$$

The time t_l is called the message *latency* and the reciprocal of t_b is called the *bandwidth*. In the case of a transport sweep, many small communications are required. Each time information is passed to the next group of processors, the message size is

[8] This model for the time-to-completion for the RBOW step assumes no overlap in the spatial regions for which each processor is responsible. In reality, the shifting block process requires that a small overlap exist to facilitate the inversion of four cell blocks that straddle the line between two processors. If a significant number of cells are assigned to each processor, the effect of the overlap on time-to-completion is very small.

equal to the number of spatial unknowns on half the processor boundary in only one S_N direction. The total time for all communication in a complete sweep is therefore $[iP^{\frac{1}{i}}+M](t_l+k_{sw}t_b)$, where k_{sw} indicates the relatively small message size associated with sweeps. In the case of the RBOW step, communication only needs to occur four times, at the beginning of each color pass. However, the message size is larger, including the number of spatial unknowns on half the processor boundary in all of the inflow directions (half of all the S_N directions). The total time for all communication in a RBOW step is $4(t_l+k_{rbow}t_b)$, where k_{rbow} indicates the larger message size. Clearly, the parallel time-to-completion for a transport sweep or a RBOW relaxation depends on the number of processors and the number of S_N directions, as well as the nature of t_l and t_b on a given machine. Table 8.1 summarizes computation and communication time for the models described above.

TABLE 8.1
Summary of Computation time and Communication requirements for one RBOW step and one global sweep on a parallel machine.

	RBOW	sweep
Computation time	(4-6)M SPT	$[iP^{\frac{1}{i}}+M]$ SPT
Communication time	$4(t_l+k_{rbow}t_b)$	$[iP^{\frac{1}{i}}+M](t_l+k_{sw}t_b)$

While the above discussion sheds some light on the parallel scalability issue, it is only a first step towards a parallel performance comparison with DSA. In order to compare performance, one must consider two additional issues. First, the total iteration cost is a multiple of the relaxation cost. As discussed above, the V-cycle cost in 2-D is approximately $\frac{8}{3}$ times the cost of the RBOW step. In the context of parallel computing, this factor of $\frac{8}{3}$ assumes perfect parallel scalability of the coarse-grid work. In reality, the coarse-grid work does not scale perfectly. It is likely that at some point in the V-cycle, the number of unknowns in the coarse-grid problem will drop below the number of processors, and therefore certain processors will be idle. However, in most applications, the percentage of computational work that takes place while processors are idle is very small, and the affect on parallel scalability is insignificant. This issue is examined in [6]. In the case of an efficient DSA algorithm, the total iteration cost may in fact be close to the cost of the transport sweep. However, if the diffusion solve is accomplished via multigrid, the parallel scalability of the coarse-grid work affects the DSA algorithm just as it does the spatial multigrid algorithm described here. Furthermore, in the case of corner balance and other similar discretizations, a fully consistent DSA cannot be implemented at a computational cost near that of a single sweep. If an inconsistent DSA is used as a preconditioner for a Krylov method [18], the Krylov method itself represents additional computational cost. Second, the convergence factor obtained by the spatial multigrid method for a given problem is different than the convergence factor a DSA algorithm would obtain. Convergence factors for particular problems are discussed below.

9. Comparison to DSA. In order to properly compare spatial multigrid to DSA for a particular problem, one must consider not only the time required to complete an iteration, but also the convergence factor. In the case of a homogeneous thin domain, DSA gives excellent convergence due to the sweeps. For these domains,

spatial multigrid struggles, especially if the number of S_N directions is large. For a homogeneous thick domain dominated by scattering, both methods have impressive convergence factors. As indicated above, the spatial multigrid algorithm produces a convergence factor of about .06 in the thick diffusive case. While the exact convergence factor obtainable from a DSA algorithm for the thick diffusive case may depend on the particular implementation, it is likely that it would be near the theoretical value obtained from Fourier analysis of about .23. Finally, there is the heterogeneous case to consider, especially the case of the thin center. While tests have been done on thin center domains with DSA [18], convergence factors as defined by equations 5.1 and 5.2 are not given. Therefore, for the purpose of comparison, a DSA algorithm was coded and tested on the heterogeneous model domain described above.

In multigrid terms, DSA can be thought of as a 2-grid scheme in angle. The relaxation method is transport sweeps, and the coarse-grid problem is formed by taking a Galerkin P_1 [9] closure of the residual equation. The simplest way to implement a "fully consistent" DSA scheme is to simply solve the coarse grid system exactly, and this is what we have done. While this strategy is computationally expensive, the goal here is only to get an idea of what convergence factors DSA is capable of producing for certain heterogeneous problems. In [18], Warsa et. al. indicate that an inconsistent DSA used as a preconditioner for a Krylov method is a viable alternative to the fully consistent DSA. Therefore, for the purpose of this comparison, we do not consider the cost of the fully consistent DSA we have implemented, but only use it to get an idea of what convergence factors DSA is capable of producing for the thin center case. Table 9.1 shows the same tests as table 6.3. The convergence factors for the spatial multigrid method are listed again, and the convergence factors for our implementation of DSA are also given. Clearly the performance of DSA degrades in much the same way as the performance of the spatial multigrid method does for the thin center problem.

TABLE 9.1

Convergence factors for spatial multigrid and DSA on the heterogeneous model domain with vacuum center, thick pure scattering perimeter and time-step-based effective absorption. The thick region always has $\sigma_t h = 10^4$. Parentheses indicate convergence factor for a W-cycle instead of a V-cycle.

$\frac{1}{c\Delta t}$	$\rho_{sp.mg}$	ρ_{dsa}
nx	$7 * 10^{-4}$	.03
$\frac{nx}{10}$	.12	.41
$\frac{nx}{100}$	.56(.38)	.84
$\frac{nx}{1000}$	.88(.69)	.98

The reason for the degraded performance is the same in both cases: the error after relaxation is not sufficiently close to the range of interpolation. Our spatial multigrid method is based on the idea that RBOW relaxation leaves behind bilinear error. For the pure thick domain, this is true and convergence is excellent. However, in the case of the thin center problem, the error after the RBOW step is not as close to the range of bilinear interpolation, and convergence suffers. Similarly, DSA is based on the idea that a transport sweep leaves behind error that is P_1 in angle. Again, for a pure thick problem, this is true and convergence is excellent. In the thin center case, the error

[9] P_1 refers to a first order spherical harmonic expansion in angle. One forms the coarse-grid system by assuming the angular flux has a first order spherical harmonic expansion in angle, and then integrating the transport equation against the zeroth and first order spherical harmonics.

after the transport sweep in the thin region is much more complicated than P_1 in angle, and therefore the coarse-grid correction looses its effectiveness and convergence suffers.

In conclusion, while a parallel performance comparison of a state-of-the-art DSA algorithm with our spatial multigrid method is beyond the scope of this paper, the discussion above allows for a rough comparison from which one can identify the types of problems and machines on which one method would likely prove faster than the other. The time required to complete one iteration of either algorithm depends on the number of processors, the number of S_N angles being used, and the speed of processor to processor communication. The convergence factors obtain by the two methods are also different, and highly problem dependent.

10. Conclusions and Future Work. The multigrid algorithm presented here gives good convergence factors at a realistic cost for most homogeneous domains, and is appropriate for heterogeneous domains in time-dependent applications. Sequential operations associated with global transport sweeps are avoided, making the method suitable for large parallel machines. Future work will focus on improving the convergence factors for heterogeneous domains, reducing computational cost, and applying the methodology to other discretizations.

REFERENCES

[1] Marvin L. Adams, *Sub-cell balance methods for radiative transfer on arbitrary grids*, Trasp. Theory Stat. Phys., 26 (1997), pp.385-431.

[2] Marvin L. Adams and Edward W. Larsen, *Fast iterative methods for discrete-ordinates particle transport calculations*, Progress in Nuclear Energy, 40 (2002), pp.3-159.

[3] Marvin L. Adams, and William R. Martin, *Diffusion synthetic acceleration of discontinuous finite element transport iterations*, Nucl. Sci. Eng., 111 (1992), pp.145-167.

[4] S. F. Ashby, P. N. Brown, M. R. Dorr, and A. C. Hindmarsh, *A linear algebraic analysis of diffusion synthetic acceleration for the Boltzmann transport equation*, SIAM J. Numer. Anal., 32 (1995), pp.128-178.

[5] Allen Barnett, J. E. Morel, and D. R. Harris, *A multigrid acceleration method for the 1-D S_N equations with anisotropic scattering*, Nucl. Sci. Eng., 102 (1989), pp.1-21.

[6] W. L. Briggs, L. Hart, S. F. McCormick, and D. Quinlan, *"Multigrid Methods on a Hypercube", Multigrid Methods: Theory, Applications, and Supercomputing, S. F. McCormick, ed., in Lecture Notes in Pure and Applied Mathematics* Marcel Dekker, New York, 1988.

[7] William L. Briggs, Van Emden Henson, and Steve F. McCormick, *A Multigrid Tutorial, Second Edition*, SIAM, Philadelphia, Pa, 2001.

[8] A. Hoisie, O. Lubeck, and H. Wasserman, *Performance and scalability analysis of tera-flop scale parallel architectures using multidimensional wavefront applications*, Int. J. of High Performance Computing Applications, 14 (2000), pp.330-346.

[9] B. D. Lansrud, *A spatial multigrid iterative method for two-dimensional discrete-ordinates transport problems*, Ph.D. thesis, Department of Nuclear Engineering, Texas A&M University, College Station, TX, 2005.

[10] Edward Larsen, *Unconditionally stable diffusion synthetic acceleration methods for the slab geometry discrete-ordinates equations. part I: theory*, Nucl. Sci. Eng., 82 (1982), pp.47.

[11] Edward Larsen, *A grey transport acceleration method for time-dependent radiative transfer problems*, J. Comp. Phys., 78 (1988), pp.459-480.

[12] E. E. Lewis and W. F. Miller, *Computational Methods of Neutron Transport*, American Nuclear Society, La Grange Park, IL, 1993.

[13] T. A. Manteuffel, S. F. McCormick, J. E. Morel, S. Oliveira, and G. Yang, *A fast multigrid algorithm for isotropic neutron transport problems I: pure scattering*, SIAM J. Comp., 16 (1995), pp.601-635.

[14] T. A. Manteuffel, S. F. McCormick, J. E. Morel, and G. Yang, *A fast multigrid algorithm for isotropic neutron transport problems II: with absorption*, SIAM J. Comp., 17 (1996), pp.1449-1474.

[15] M. M. Mathis, N. M. Amato, and M. L. Adams, *A general performance model for parallel sweeps on orthogonal grids for particle transport calculations*, in Proc. ACM Int. Conf. Supercomputing (ICS), pp.255-263, Santa Fe, NM (2000).

[16] J. E. Morel, J. E. Dendy Jr., and T. A. Wareing, *Diffusion-accelerated solution of the two-dimensional S_N equations with bilinear-discontinuous differencing*, Nucl. Sci. Eng., 115 (1993), pp.304-319.

[17] Peter S. Pacheco, *Parallel Programming with MPI*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1997.

[18] J. S. Warsa, T. A. Wareing, and J. E. Morel, *Krylov iterative methods and the degraded effectiveness of diffusion synthetic acceleration for multidimensional S_N calculations in problems with material discontinuities*, Nucl. Sci. Eng., 147 (2004), pp.218-248.

[19] Musa Yavuz and Edward W. Larsen, *Iterative methods for solving x-y geometry S_N problems on parallel architecture computers*, Nucl. Sci. Eng., 112 (1992), pp.32-42.