PARALLEL MULTIGRID METHODS FOR TRANSPORT EQUATIONS: THE ANISOTROPIC CASE*

S. OLIVEIRA†

Abstract. A efficient parallel multilevel algorithm is developed for solving the transport equations on parallel computers for one-dimensional anisotropic scattering. The parallel algorithm is developed by using a multigrid in angle scheme that is known to attenuate both rapidly and slowly varying errors in angle. The spatial discretization scheme used is the modified linear discontinuous finite element method, which represents a lumped version of the standard linear discontinuous scheme. The angular discretization is accomplished by expanding the angular dependence in Legendre polynomials and is known as the S_N approximation when the first N Legendre polynomials are used. Legendre transforms of complexity $O(N)$ and a anisotropic parallel algorithm of complexity $O(N \log_2 m \log_2 N)$ are developed.

Key Words. Transport equations, Legendre tranforms, cyclic reduction, multigrid, SIMD and MIMD computers, Connection Machine.

1. Introduction. The transport of neutrally charged particles can be modeled by the linear Boltzmann transport equation. For steady-state problems, within the same energy group this equation can be written as

$$\underline{\Omega} \cdot \nabla \psi(\underline{u}, \underline{\Omega}) + \sigma_t \psi(\underline{u}, \underline{\Omega}) = S(\underline{u}, \underline{\Omega}) + Q(\underline{u}, \underline{\Omega}), \tag{1.1}$$

where $\psi(\underline{u}, \underline{\Omega})$ is the particle flux at position $\underline{u}$ moving in direction $\underline{\Omega}$ (unit vector) and $\sigma_t\, d\underline{u}$ represents the expected number of interactions (absorptive or scattering) that a particle will have in traveling a distance $d\underline{u}$. The first term on the left-hand side represents particle loss due to streaming, the second term represents loss due to scattering or absorption, the first term on the right-hand side represent sources due to rescattering, and the second term represents source due to external sources. The source term $S(\underline{u}, \underline{\Omega})$ contains the particles that are scattered from every other

* This work was supported by AFOSR under grant AFOSR 86-0126, the NSF under grant DMS-8704169,the DOE under grant DE-FG02-90ER25086 and Los Alamos National Laboratory.

† Computer Science Department, Texas A&M University, College Station, TX 77843-3112.

direction $\underline{\Omega}'$ to direction $\underline{\Omega}$ and can be modeled as

$$S(\underline{u}, \underline{\Omega}) = \sigma_s \int d\underline{\Omega}' \Sigma_s(\underline{u}, \underline{\Omega}' \to \underline{\Omega}) \psi(\underline{x}, \underline{\Omega}'),$$

where Σ_s is the probability density that a particle will be scattered from direction $\underline{\Omega}'$ into direction $\underline{\Omega}$ and $\sigma_s d\underline{u}$ represents the expected number of collisions that result in a scattering along a path $d\underline{u}$.

When $\sigma_s \to \infty$ and $\sigma_s/\sigma_t \to 1$, equation (1.1) will be singularly perturbed. In fact, it is often the case that this equation is nearly singular in all spatial frequencies of interest. This fact causes difficulties for numerical solution techniques, and multigrid in particular.

Consider equation (1.1) in a semi-infinite slab of width b in the x-dimension. Assuming that the solution is constant in y and z, then $\frac{\partial \psi}{\partial y} = \frac{\partial \psi}{\partial z} = 0$. Equation (1.1) becomes

$$\mu \cdot \frac{\partial \psi}{\partial x} + \sigma_t \psi = S(x, u) + q(x, \mu), \tag{1.2}$$

Here, $\psi(x, \mu)$ represents the flux of particles at position x traveling at an angle $\theta = \arccos(\mu)$ from the x-axis. The quantity σ_t was already defined. The source term in one dimension is given by

$$S(x, \mu) = \sigma_s \int_{-1}^{1} d\mu' \Sigma_s(x, \mu' \to \mu) \psi(x, \mu')$$

The boundary conditions prescribing particles entering the slab are

$$\psi(0, \mu) = g_0(\mu), \qquad \psi(b, -\mu) = g_1(\mu), \qquad \mu \in (0, 1). \tag{1.3}$$

For the isotropic scattering model, Σ_s is constant in angle, that is the particle is equally likely to scatter in any direction. The anisotropic scattering is modeled under the assumption that the probability of scattering from direction $\underline{\Omega}'$ into direction $\underline{\Omega}$ is a function of the angle between these directions $(\underline{\Omega} \cdot \underline{\Omega}')$. In this paper we consider the anisotropic scattering case. The scattering source term

$$S(\underline{u}, \underline{\Omega}) = \int d\underline{\Omega}' \Sigma_s(\underline{u}, \underline{\Omega}' \to \underline{\Omega}) \psi(\underline{x}, \underline{\Omega}'),$$

are integral operators with spherical harmonics as their singular vectors. We assume that the variation in angle can be described by a finite collection of singular vectors, resulting in an S_N discretization. In one-dimensional problems, the singular vectors for the integral operators are Legendre polynomials. Let $p_0(\mu), \ldots, p_{N-1}(\mu)$ be the Legendre polynomials and $\sigma_0, \ldots, \sigma_{N-1}$ be the associated singular values. Expressing the flux $\psi(x, \mu)$ as a finite summation of the first N Legendre polynomials we have

$$\psi(x,\mu) = \sum_{l=0}^{N-1} (2l+1)\phi_l(x)p_l(\mu), \tag{1.4}$$

where $(2l+1)$ is a normalization factor and the moments ϕ_l are

$$\phi_l(x) = \frac{1}{2}\int_{-1}^{1} d\mu' p_l(\mu')\psi(x,\mu'). \tag{1.5}$$

which can be written exactly as

$$\phi_l(x) = \sum_{k=1}^{N} w_k p_l(\mu_k)\psi(x,\mu_k), \tag{1.6}$$

for $l = 0, \ldots, N-1$, where the variables $\mu_1, \mu_2, \ldots, \mu_N$ are the Gauss quadrature points and $w_1, w_2, \ldots, w_N$ are the Gauss quadrature weights of degree N. The scattering term in one dimension then becomes

$$S_s(x,\mu) = \sigma_s \int_{-1}^{1} d\mu' \Sigma_s(x, \mu' \to \mu)\psi(x,\mu') = \sum_{l=0}^{N-1}(2l+1)\sigma_l\phi_l(x)p_l(\mu), \tag{1.7}$$

Thus the anisotropic transport equation in 1-D can be written as

$$\mu \cdot \frac{\partial \psi}{\partial x} + \sigma_t \psi = \sum_{l=0}^{N-1}(2l+1)\sigma_l\phi_l(x)p_l(\mu) + q(x,\mu) \tag{1.8}$$

In this paper we develop an efficient parallel algorithm for solving the anisotropic neutron transport equation on parallel computers. We use an angular multigrid algorithm that requires a shifted source iteration at each level. Performing this iteration with a shifted transport sweep would take $2m$ sequential steps for a grid with m cells. We develop a data parallel block cyclic reduction scheme based on [14] for this step of the angular multigrid method, which takes $\log_2 m$ steps. One issue dealt with is the stability of the parallel block cyclic reduction. We can get a stable algorithm if we choose to write the system of equations in such a way that the right-hand side remains bounded at all levels of cyclic reduction. The stability of this method is proved. The coarsest level of angular multigrid involves an isotropic S_2 equation. We use our parallel multigrid algorithm for the isotropic model to solve this equation. There are other aspects of the angular multigrid that need attention in the development of a parallel angular multigrid, such as calculation of the scattering term. This step is completely parallelized in the spatial direction and partially parallelized in the angular direction. In addition, we attempted to minimize the number of communications in the angular direction. At diverse stages of the algorithm, we need to perform Legendre transforms. An algorithm for these transforms is developed that takes $O(N)$ steps. We compare two kinds of angular multigrid schemes, a $V(1,1)$-cycle and a $V(1,0)$-cycle, and conclude that $V(1,1)$ is superior on Connection Machine architectures. The algorithms are also compared to sequential implementations on the Cray Y-MP.

An outline of the remainder of this paper is as follows. In Section 3.2 we describe the angular multigrid algorithm in the flux notation. In Section 3.3 we describe the spatial discretization used. In Section 3.4 we show how to perform the shifted source iteration as a cyclic reduction for the MLD discretization. In Section 3.5 we develop the parallel angular multigrid algorithm. In Section 3.6 we develop the parallel cyclic reduction algorithm. In Section 3.7 we analyze the stability of our cyclic reduction. In Section 3.8 we report on convergence rates and timings tests. Finally, in Section 3.9 we make a few conclusions.

For the most part, the following notational conventions are used in this paper. Particular usage and exceptions should be clear from context.

$H_k, \ldots$	–	Operator or Matrix at level k of angular multigrid cycle
B_i^k, F_i^k, $D_i^k, \ldots$	–	Sub-matrices at level k of cyclic reduction, on row i of a bigger matrix (corresponding to spatial cell i).
N	–	Number of Gauss quadrature points ($n = \frac{N}{2}$).
N_k	–	Number of Gauss quadrature points at level k.
p_i	–	Legendre polynomial of degree i ($i = 0, \ldots, N-1$).

P_i	–	Processor i $(i = 1, \ldots, N)$.
σ_i	–	Singular values associated with the Legendre polynomial p_i (degree i) $(i = 0, \ldots, N-1)$.
$\underline{z}$, $\underline{r}$, ...	–	Vectors.
a, ϵ, α, ...	–	Scalars or functions.
SIMD (MIMD)	–	Single (multiple) Instruction Multiple Data computer

2. Angular Multigrid. Notice that equations (1.4), (1.6), and (1.7) can be written, for all the quadrature points, in the matrix notation

$$\underline{\psi}(x) = T^{-1}\underline{\phi}(x), \qquad \underline{\phi}(x) = T\underline{\psi}(x), and \qquad S(x, \underline{\mu}) = T^{-1}DT\underline{\psi}(x),$$

respectively. The vector $\underline{\psi}(x)$ is equal to $(\psi(x,\mu_1), \psi(x,\mu_2), \ldots, \psi(x,\mu_N))^T$, and the vector $\underline{\phi}(x) = (\phi_0(x), \phi_1(x), \ldots, \phi_{N-1}(x))^T$ represents the Legendre moments. The $N \times N$ matrices T and T^{-1}, are defined by their respective elements

$$T_{i,j} = w_j p_{i-1}(\mu_j), \qquad T^{-1}_{i,j} = (2j-1)p_{j-1}(\mu_i),$$

and the $N \times N$ diagonal matrix D has diagonal elements

$$D_{i,i} = \sigma_{i-1}.$$

The requirement that (1.8) holds at each quadrature point yields a coupled system of ODE's:

$$M\underline{\psi}'(x) + \sigma_t\underline{\psi}(x) = T^{-1}DT\underline{\psi}(x) + \underline{q}(x), \tag{2.9}$$

where M=diag$(\cdots, \mu_j, \cdots)$ and D = diag$(\cdots, \sigma_j, \cdots)$. This is the flux representation of the S_N equations. If the right-hand side of (2.9) is known, it becomes an uncoupled system of ODE's . This leads to the iteration

$$M\underline{\psi}'^{l+1}(x) + (\sigma_t - \alpha)\underline{\psi}^{l+1}(x) = T^{-1}(D - \alpha I)T\underline{\psi}^l(x) + \underline{q}(x), \tag{2.10}$$

which is called a shifted source iteration. The variable α is a shift parameter. The system of ODE's will be stable for $\alpha \le \sigma_t$. Suppose $\sigma_t = \sigma_0 \ge \sigma_1 \ge \ldots \ge \sigma_{N-1} \ge 0$

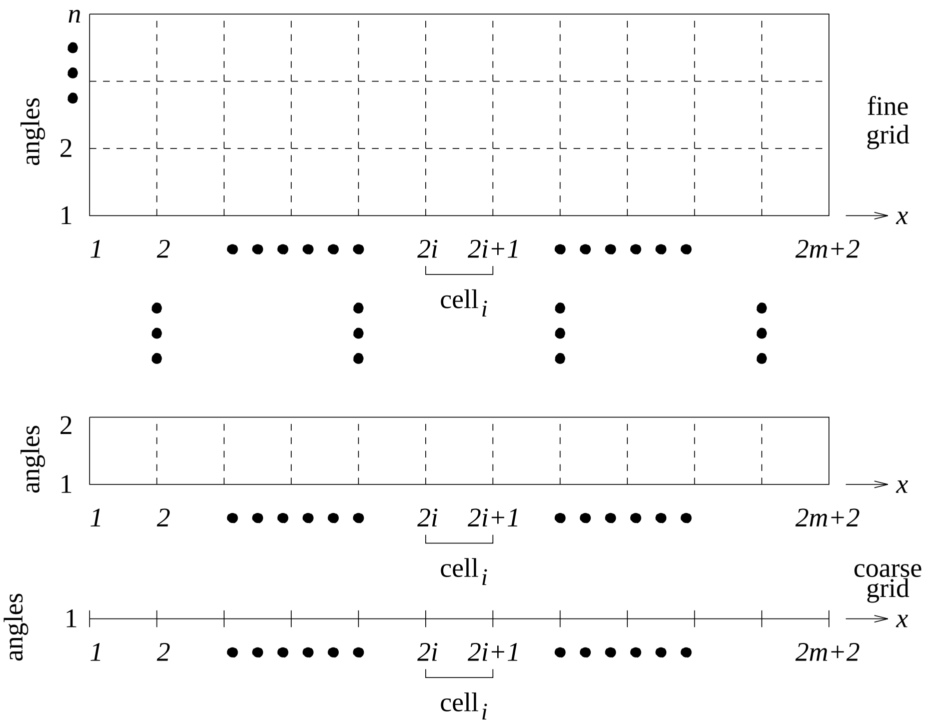

FIG. 1. *Computational grid levels angular multigrid.*

and $\sigma_t \gg 1$. If α is chosen to be $(\sigma_{N/2} + \sigma_{N-1})/2$, a shifted source iteration will attenuate all the errors in the upper half of the flux moments ($p_{N/2}$ through p_{N-1}). In other words, it will attenuate the errors that vary rapidly in angle, but poorly attenuate those that vary slowly in angle. This suggests that multigrid in angle be used together with a shifted source iteration at each grid level k as Morel and Manteuffel developed in [13]. Figure 1 illustrates the first two grid levels for the angular multilevel scheme where n is the number of scattering angles. In order to solve (2.10) at level k, we use the same spatial discretization used for the isotropic model in [11] and [12]: the modified linear discontinous (MLD) scheme. When this discretization is used we obtain

$$H_k \underline{\psi}_k^{l+1} = S_k \underline{\psi}_k^l + \underline{q}_k,$$

where H_k and S_k both contain the shift α_k at angular level k. Matrix H_k is described in Section 3.4 and

$$S_k = T_k^{-1}(D_k - \alpha_k I_k)T_k. \tag{2.11}$$

The transport sweep has been used before to perform the shifted source iteration [9]. Letting m be the number of cells ($2m + 2$ spatial points) used in the MLD spatial discretization, the transport sweep is a sequential algorithm that takes $2m$ steps, corresponding to solving a block lower bidiagonal system by block forward substitution and a block upper bidiagonal system by block backward substitution. To make the source iteration economical on SIMD architectures such as the CM200, we perform it at each angular level by using cyclic reduction, which takes $\log_2 m$ parallel steps. Throughout the angular multigrid cycle, multiplications by T and T^{-1}, which represent Legendre transforms, will be performed many times. Our storage design and algorithm for these multiplications allow Legendre transforms to be performed in O(N) parallel steps.

The angular multigrid method was designed to accelerate problems with highly forward-peaked scattering [13], where the probability that a neutron is scattered through a small angle is higher than for a large angle. It takes $\log_2 N$ steps. Before we describe the implementation details of this algorithm, we give a general description of the $V(\eta, \nu)$ angular multigrid cycle. Again, consider the discretized shifted source iteration

$$H_k \underline{\psi}_k^{l+1} = S_k \underline{\psi}_k^l + \underline{q}_k.$$

The angular multigrid method proceeds as follows

Set $k = 1$ and $N_k = N$ for the first angular grid level.

1. Perform η shifted source iterations on the fine grid:

$$H_k \underline{\psi}_k^{l+1} = S_k \underline{\psi}_k^l + \underline{q}_k,$$

$l = 0, \cdots, \eta - 1$. Set $\underline{\psi}_k = \underline{\psi}_k^\eta$. Compute the residual $\underline{r}_k = S_k(\underline{\psi}_k^{l+1} - \underline{\psi}_k^l)$, where $l = \eta - 1$. Calculate the moments for this residual: $\Delta\phi_k = (\underline{\phi}^{l+1} - \underline{\phi}^l)_k = T_k \underline{r}_k$. Disregard the high moments: $\Delta\phi_{k+1} = [I, 0]\Delta\phi_k$

2. Set $N_{k+1} = N_k/2$ and $k = k+1$. Perform a source iteration on the coarse grid with $\underline{q}_k$ equal to $T_k^{-1}\Delta\phi_k$

$$H_k \underline{\hat{\psi}}_k^{l+1} = S_k \underline{\hat{\psi}}_k^{l} + \underline{q}_k,$$

$l = 0, \cdots, \eta - 1$. Set $\underline{\hat{\psi}} = \underline{\hat{\psi}}^{\eta}$. Repeat Steps 1 and 2 until N_k is equal to 2.

3. On the coarsest grid ($N_k = 2$), this problem will be reduced to the isotropic model. Solve it exactly using the parallel spatial multigrid algorithm developed in [10] and [11].

4. Add the corrections from the coarser grid to the flux solution on the finer grid:

$$\underline{\hat{\psi}}_k^{0} = \underline{\hat{\psi}}_k + T_k^{-1} \begin{bmatrix} I_{k+1} \\ 0 \end{bmatrix} T_{k+1} \underline{\hat{\psi}}_{k+1}.$$

5. Perform ν shifted source iterations using the newest calculated solution for level k:

$$H_k \underline{\hat{\psi}}_k^{l+1} = S_k \underline{\hat{\psi}}_k^{l} + \underline{q}_k,$$

$l = 0, \cdots, \nu - 1$. Set $\underline{\hat{\psi}}_k = \underline{\hat{\psi}}_k^{\nu}$. Set $N_{k-1} = 2N_k$ and $k = k-1$. Repeat steps 4 and 5 until $k = 1$ and $N_k = N$.

In the next two subsections we detail our $V(1,0)$ and $V(1,1)$ implementations. The $V(1,0)$-cycle, in general, does not need a special description, but in order to develop an algorithm that takes advantage of the greater simplicity of the $V(1,0)$-cycle over the $V(1,1)$-cycle and to avoid redundant calculations, we write the algorithms in a slightly different way for the interpolation steps (Steps 5–6 in the next section).

2.1. The $V(1,0)$ cycle. Consider writing the transport equation in its discretized form

$$H_k \underline{\psi}_k^{l+1} = S_k \underline{\psi}_k^{l} + \underline{q}_k. \tag{2.12}$$

• *Step 0. Initialization.*

Set $k = 1$ and $N_k = N$.

•*Step 1. Shifted source iteration in parallel.*

The shifted source iteration is given by

$$H_k \underline{\psi}_k^{l+1} = \underline{f}_k^l, \qquad \text{where} \qquad \underline{f}_k^l = S_k \underline{\psi}_k^l + \underline{q}_k. \tag{2.13}$$

At level k, the number of angles is half the number in the previous grid. This indicates that the Gauss quadrature points are different at each angular grid level and, consequently, the matrices H_k, S_k, and M_k will depend on the grid level. This step is performed using cyclic reduction (Section 4). The initial guess for the flux on the right-hand side of the matrix equation is $\underline{\psi}_k^0 = 0$. Here and in the next steps we assume $\eta = 1$. Set $\underline{\psi}_k = \underline{\psi}_k^\eta$.

• *Step 2. Calculating the residual.*

The residual is given by

$$\underline{r}_k = S_k(\underline{\psi}_k^{l+1} - \underline{\psi}_k^l) = T_k^{-1}(D_k - \alpha_k I_k)T_k(\underline{\psi}_k^{l+1} - \underline{\psi}_k^l). \tag{2.14}$$

Multiplication by matrix S_k is applied in conjunction with the restriction operation as described in the next step.

• *Step 3. Restriction of the residual.*

The external source term on the next grid level ($\underline{q}_{k+1}$) is equal to the restricted residual. The restriction operator is given by

$$I_N^{N/2} = T_{k+1}^{-1}[I_{k+1}, \quad 0]T_k. \tag{2.15}$$

Putting (2.14) and (2.15) together, we have

$$\underline{q}_{k+1} = T_{k+1}^{-1}[I_{k+1}, \quad 0](D_k - \alpha_k I_k)T_k(\underline{\psi}_k^{l+1} - \underline{\psi}_k^l). \tag{2.16}$$

To compute this efficiently, we proceed as follows

- Calculate $\Delta\underline{\psi}_k$: $\Delta\underline{\psi}_k = \underline{\psi}_k^{l+1} - \underline{\psi}_k^l$.
- Multiply $\Delta\underline{\psi}_k$ by T_k. $\Delta\underline{\phi}_k = T_k \Delta\underline{\psi}_k$.

  This represents a change from flux representation to moment representation.

The matrix T_k is given by $T_k =$

$$\begin{bmatrix} w_1p_0(\mu_1) & w_2p_0(\mu_2) & w_3p_0(\mu_3) & \cdots & w_{N-1}p_0(\mu_{N-1}) & w_Np_0(\mu_N) \\ w_1p_1(\mu_1) & w_2p_1(\mu_2) & w_3p_1(\mu_3) & \cdots & w_{N-1}p_1(\mu_{N-1}) & w_Np_1(\mu_N) \\ w_1p_2(\mu_1) & w_2p_2(\mu_2) & w_3p_2(\mu_3) & \cdots & w_{N-1}p_2(\mu_{N-1}) & w_Np_2(\mu_N) \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ w_1p_{N-1}(\mu_1) & w_2p_{N-1}(\mu_2) & w_3p_{N-1}(\mu_3) & \cdots & w_{N-1}p_{N-1}(\mu_{N-1}) & w_Np_{N-1}(\mu_N) \end{bmatrix}_{\mathrm{N\times N}} . \tag{2.17}$$

We should notice here that whenever we present a matrix with its elements shown explicitly we omit the k index for simplicity ($N = N_k, \mu_j = \mu_j^k, w_j = w_j^k$ and so on).

- Multiplications by $[I_{k+1}, \quad 0]$ and by $(D_k - \alpha_k I)$ are interchangeable. We multiply by $[I_{k+1}, \quad 0]$ first, thus disregarding the second half of the moments (higher moments) and considering only the first half of the moments (lower moments) on the coarser grid $(k+1)$. This can be written as

$$\Delta\underline{\phi}_{k+1} = \begin{bmatrix} I & 0 \end{bmatrix}_{\frac{N}{2}\times N} \Delta\underline{\phi}_k$$

- At angular level k we multiply by the scattering matrix $(D_k - \alpha_k I)$ truncated to $N_k/2$ rows and columns. Notice that D_k is a truncation of the scattering matrix used with the P_{N-1} expansion on the fine grid (k) and is given by

$$D_k = \begin{bmatrix} \sigma_0 & & & & \\ & \sigma_1 & & & \\ & & \sigma_2 & & \\ & & & \ddots & \\ & & & & \sigma_{\frac{N}{2}-1} \end{bmatrix}_{\frac{N}{2}\times\frac{N}{2}} ,$$

and α_k is the shift on the fine grid (k) and is equal to $(\sigma_{N/2} + \sigma_{N-1})/2$.

- Finally, multiply by T_{k+1}^{-1} $(N_{k+1} = N_k/2)$, which is calculated using the Gauss quadrature points of level $k+1$. Recall that the N_{k+1} Gauss quadrature points are not a subset of the N_k points. This matrix is given by $T_{k+1}^{-1} =$

$$\begin{bmatrix} p_0(\mu_1) & 3p_1(\mu_1) & 5p_2(\mu_1) & \cdots & (2N-3)p_{N-2}(\mu_1) & (2N-1)p_{N-1}(\mu_1) \\ p_0(\mu_2) & 3p_1(\mu_2) & 5p_2(\mu_2) & \cdots & (2N-3)p_{N-2}(\mu_2) & (2N-1)p_{N-1}(\mu_2) \\ p_0(\mu_3) & 3p_1(\mu_3) & 5p_2(\mu_3) & \cdots & (2N-3)p_{N-2}(\mu_3) & (2N-1)p_{N-1}(\mu_3) \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ p_0(\mu_N) & 3p_1(\mu_N) & 5p_2(\mu_N) & \cdots & (2N-3)p_{N-2}(\mu_N) & (2N-1)p_{N-1}(\mu_N) \end{bmatrix}_{N\times N} . \tag{2.18}$$

Using $\underline{q}_{k+1}$ as the new source term, repeat Steps 1 through 3 until we have a P_1 expansion on the right-hand-side (when $N_k/2 = 2$).

• Step 4. *Calculation of the error at the coarsest level.*

With the appropriate shift ($\alpha = \sigma_1$) at this level ($k = \log_2 n$), we have an isotropic transport equation:

$$H_k \hat{\underline{\psi}}_k^{l+1} = S_k \hat{\underline{\psi}}_k^{l} + \underline{q}_k$$

where

$$\underline{q}_k = T_k^{-1}[I_{k+1} \ \ 0](D_{k-1} - \alpha I)T_{k-1}(\underline{\psi}_{k-1}^{l+1} - \underline{\psi}_{k-1}^{l}),$$

$$T_k^{-1} = \begin{bmatrix} \frac{1}{2} & \frac{3}{2}\mu_1 \\ \frac{1}{2} & \frac{3}{2}\mu_2 \end{bmatrix}_{2\times 2},$$

$$T_k = \begin{bmatrix} w_1 & w_2 \\ w_1\mu_1 & w_2\mu_2 \end{bmatrix}_{2\times 2},$$

and

$$D_k = \begin{bmatrix} \sigma_0 - \sigma_1 & 0 \\ 0 & 0 \end{bmatrix}_{2\times 2}.$$

At this level, instead of performing the source iteration through cyclic reduction as was done in finer angular grid levels, we use the parallel multigrid in space algorithm for solving the isotropic equations. For $N = 2$, multigrid in space solves this system of equations exactly in one V-cycle.

• Step 5. *Interpolation of the correction.*

Starting with $k = \log_2 n$, transform each correction from the flux domain to the moment domain:

$$\underline{\varepsilon}_k = \begin{bmatrix} I_k \\ 0 \end{bmatrix} T_k \underline{\psi}_k. \tag{2.19}$$

The padding is not actually performed in the CM codes, since the entries (or the processors) where $\underline{\varepsilon}$ is not calculated were previously initialized to zero. Set $k = k - 1$ and calculate a new $\underline{\varepsilon}_k$. Repeat this process until $k = 2$, summing the moments for levels $l = 2, \cdots, \log_2 n$ as

$$\underline{\varepsilon}_1 = \underline{\varepsilon}_2 + \underline{\varepsilon}_3 + \cdots + \underline{\varepsilon}_{\log_2 n}. \tag{2.20}$$

• Step 6. *Addition of the corrections.*

At the finest angular grid, the summation of the errors at all angular levels is transformed from moment to flux representation and this flux-correction is then added to the latest flux solution at this level:

$$\underline{\psi}_1 = \underline{\psi}_1 + T_1^{-1} \underline{\varepsilon}_1. \tag{2.21}$$

Recall that, at this level, T^{-1} will be an $N \times N$ matrix generated by using the Gauss quadrature points for all N scattering angles.

2.2. The $V(1,1)$ cycle. For the $V(1,1)$ cycle, we modify the interpolation steps (Steps 5 through 6). The general interpolation operation at level k can be described as

$$\underline{\hat{\psi}}_k^0 = \underline{\hat{\psi}}_k + T_k^{-1} \begin{bmatrix} I_{k+1} \\ 0 \end{bmatrix} T_{k+1} \underline{\hat{\psi}}_{k+1}. \tag{2.22}$$

After we calculate $\underline{\psi}_k^0$, we perform shifted source iteration:

$$H_k \hat{\underline{\psi}}_k^{l+1} = S_k \hat{\underline{\psi}}_k^{l} + \underline{q}_k, \tag{2.23}$$

where $\underline{q}_k$ is the restricted residual used at level k when going down the cycle. Notice that we assume $\nu = 1$ (one iteration when ascending). Set $k = k - 1$ and repeat this process until $k = 1$.

3. Discretization. The spatial discretization is accomplished by the MLD scheme, which uses elements that are linear across each cell and discontinuous in the upwind direction In our grid representation, the variable $\psi_{ij}^{+(-)}$ denotes the flux of particles at position x_i in the direction μ_j $(-\mu_j)$.

For the 1-D anisotropic scattering model the source term can be written as

$$S(x, \mu) = \sigma_s \int_{-1}^{1} d\mu' \Sigma_s(x, \mu' \to \mu)\psi(x, \mu') = \sum_{\ell=0}^{N-1} (2\ell + 1)\sigma_\ell \phi_\ell p_\ell(\mu), \tag{3.24}$$

Assuming σ_t/σ_s=1, the nodal equations using variables at the edges are

$$\mu_j(\psi_{i,l,j}^{+} + \psi_{i,r,j}^{+}) - 2\mu_j \psi_{i-1,r,j}^{+} + \sigma_t h \psi_{i,l,j}^{+} = h \sum_{\ell=0}^{N-1} (2\ell + 1)\sigma_\ell \phi_{i,l,\ell} p_\ell(\mu_j), \tag{3.25}$$

$$\mu_j(\psi_{i,r,j}^{+} - \psi_{i,l,j}^{+}) + \sigma_t h \psi_{i,r,j}^{+} = h \sum_{\ell=0}^{N-1} (2\ell + 1)\sigma_\ell \phi_{i,r,\ell} p_\ell(\mu_j), \tag{3.26}$$

$i = 1, \ldots, m$, $j = 1, \ldots, n$ and the negative angle equations are

$$\mu_j(\psi_{i,l,j}^{-} + \psi_{i,r,j}^{-}) - 2\mu_j \psi_{i+1,l,j}^{-} + \sigma_t h \psi_{i,r,j}^{-} = h \sum_{\ell=0}^{N-1} (2\ell + 1)\sigma_\ell \phi_{i,r,\ell} p_\ell(\mu_j), \tag{3.27}$$

$$\mu_j(\psi_{i,l,j}^{-} - \psi_{i,r,j}^{-}) + \sigma_t h \psi_{i,l,j}^{-} = h \sum_{\ell=0}^{N-1} (2\ell + 1)\sigma_\ell \phi_{i,l,\ell} p_\ell(\mu_j), \tag{3.28}$$

$i = 1, \ldots, m$, $j = 1, \ldots, n$.

Using the vector notation such that $\underline{\psi}_i^{+(-)}$ is an n-vector ($\underline{\psi}_i^{+(-)} = (\psi_{i1}^{+(-)}, ..., \psi_{in}^{+(-)})^T$), we can rewrite these equations in matrix form as

$$M_i(\underline{\psi}_{i,l}^{+} + \underline{\psi}_{i,r}^{+}) - 2M_i \underline{\psi}_{i-1,r}^{+} + \sigma_t \underline{\psi}_{i,l}^{+} = S^{+}(\underline{\psi}_{i,l}^{+}, \underline{\psi}_{i,l}^{-}) \tag{3.29}$$

$$M_i(\underline{\psi}_{i,r}^{+} - \underline{\psi}_{i,l}^{+}) + \sigma_t \underline{\psi}_{i,r}^{+} = S^{+}(\underline{\psi}_{i,r}^{+}, \underline{\psi}_{i,r}^{-}) \tag{3.30}$$

$i = 1, \ldots, m$ and the negative angle equations are

$$M_i(\underline{\psi}_{i,l}^- + \underline{\psi}_{i,r}^-) - 2M_i\underline{\psi}_{i+1,l}^- + \sigma_t\underline{\psi}_{i,r}^- = S^-(\underline{\psi}_{i,r}^+, \underline{\psi}_{i,r}^-) \tag{3.31}$$

$$M_i(\underline{\psi}_{i,l}^- - \underline{\psi}_{i,r}^-) + \sigma_t\underline{\psi}_{i,l}^- = S^-(\underline{\psi}_{i,l}^+, \underline{\psi}_{i,l}^-) \tag{3.32}$$

$i = 1, \ldots, m$ where the matrices S^+ and S^- represent the upper and lower part of matrix $S = T^{-1}DT$, respectively, and matrix $M_i = \mathrm{diag}(\frac{\mu_1}{h_i}, \cdots, \frac{\mu_n}{h_i})$. If we use shifted source iteration to solve these matrix equations, the right-hand sides are assumed known, the matrix $S = T^{-1}(D - \alpha I)T$ and the σ_t's are replaced by $\epsilon = \sigma_t - \alpha$ in the above equations.

4. Source Iteration in Parallel. The source iteration is performed using cyclic reduction [14]. Using equations (3.29–3.32), the source iteration for the entire domain can be written as $H\underline{\psi}^{l+1} = \underline{f}^l$, where the matrices H and the vector $\underline{f}$ already contain the shifts (α). The matrix H is $4 \times m \times n$. A sub matrix of H, corresponding to 2 cells of a spatial domain with m cells (an $8n \times 8n$ matrix), is

$$\begin{bmatrix} \epsilon I + M_i & 0 & -M_i & 0 & 0 & 0 & 0 & 0 \\ 0 & \epsilon I + M_i & 0 & M_i & 0 & 0 & 0 & 0 \\ M_i & 0 & \epsilon I + M_i & 0 & -2M_i & 0 & 0 & 0 \\ 0 & -M_i & 0 & \epsilon I + M_i & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \epsilon I + M_{i+1} & 0 & -M_{i+1} & 0 \\ 0 & 0 & 0 & -2M_{i+1} & 0 & \epsilon I + M_{i+1} & 0 & M_{i+1} \\ 0 & 0 & 0 & 0 & M_{i+1} & 0 & \epsilon I + M_{i+1} & 0 \\ 0 & 0 & 0 & 0 & 0 & -M_{i+1} & 0 & \epsilon I + M_{i+1} \end{bmatrix}_{8n \times 8n} \tag{4.1}$$

the variables for this matrix are

$$\underline{\psi} = (\underline{\psi}_{i,l}^-, \underline{\psi}_{i,l}^+, \underline{\psi}_{i,r}^-, \underline{\psi}_{i,r}^+, \underline{\psi}_{i+1,l}^-, \underline{\psi}_{i+1,l}^+, \underline{\psi}_{i+1,r}^-, \underline{\psi}_{i+1,r}^+)^T. \tag{4.2}$$

The right-hand side for this two-cell system can be represented as

$$\underline{f} = (\underline{f}_{i,l}^-, \underline{f}_{i,l}^+, \underline{f}_{i,r}^-, \underline{f}_{i,r}^+, \underline{f}_{i+1,l}^-, \underline{f}_{i+1,l}^+, \underline{f}_{i+1,r}^-, \underline{f}_{i+1,r}^+)^T. \tag{4.3}$$

Notice that this relaxation can be performed separately for the negative and positive variables, since the above system of equations can be decoupled into two systems, one for the positive and another for the negative variables.

• **Positive variable cyclic reduction**

Writing the above system for the positive variables over the entire domain generates a block lower bidiagonal (BLB) system with block matrices of size $2n \times 2n$. The associated matrix is

$$\begin{bmatrix} D_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -F_2 & D_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -F_3 & D_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -F_i & D_i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -F_{i+1} & D_{i+1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -F_m & D_m \end{bmatrix}_{2mn \times 2mn} . \tag{4.4}$$

The variables for this matrix are

$$\underline{\psi} = (\underline{\psi}_1^+, \underline{\psi}_2^+, \cdots, \underline{\psi}_i^+, \underline{\psi}_{i+1}^+, \cdots, \underline{\psi}_m^+)^T, \tag{4.5}$$

where $\underline{\psi}_i^+ = (\underline{\psi}_{i,l}^+, \underline{\psi}_{i,r}^+)$. The right-hand side for this smaller system is

$$\underline{f} = (\underline{f}_1^+, \underline{f}_2^+, \cdots, \underline{f}_i^+, \underline{f}_{i+1}^+, \cdots, \underline{f}_m^+)^T \tag{4.6}$$

where $\underline{f}_i^+ = (\underline{f}_{i,l}^+, \underline{f}_{i,r}^+)$,

$$D_i = \begin{bmatrix} \epsilon I + M_i & M_i \\ -M_i & \epsilon I + M_i \end{bmatrix},$$

and

$$F_i = \begin{bmatrix} 0 & 2M_i \\ 0 & 0 \end{bmatrix}. \tag{4.7}$$

Notice that index i in this notation represents the variables evaluated for cell i: $(\underline{\psi}^+_{i,l}, \underline{\psi}^+_{i,r})$. If we multiply the above system by

$$
\begin{bmatrix}
I & 0 & 0 & 0 & 0 & 0 & 0 \\
F_2 D_1^{-1} & I & 0 & 0 & 0 & 0 & 0 \\
0 & F_3 D_2^{-1} & I & 0 & 0 & 0 & 0 \\
0 & 0 & F_4 D_3^{-1} & I & 0 & 0 & 0 \\
0 & 0 & 0 & \ddots & \ddots & 0 & 0 \\
0 & 0 & 0 & 0 & F_{m-1} D_{m-2}^{-1} & I & 0 \\
0 & 0 & 0 & 0 & 0 & F_m D_{m-1}^{-1} & I
\end{bmatrix}_{mn \times mn} .
$$

we have

$$
\begin{bmatrix}
D_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & D_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-F_3^2 & 0 & D_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -F_4^2 & 0 & D_4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -F_i^2 & 0 & D_i & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -F_{i+1}^2 & 0 & D_{i+1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -F_m^2 & 0 & D_m
\end{bmatrix}_{mn \times mn} , \tag{4.8}
$$

where

$$
F_i^2 = \begin{bmatrix} 0 & \frac{4M_i M_{i-1}^2}{\Delta_{i-1}} \\ 0 & 0 \end{bmatrix},
$$

and $\Delta_i = 2M_i^2 + 2\epsilon M_i + \epsilon^2 I$. Matrices D_i's are unchanged. If this process is performed recursively, at each level k a matrix is obtained with the same block diagonal matrices as in the initial stage and with off-diagonal matrices whose elements depend on k. The multiplying matrix to go from level $k-1$ to level k is

$$
(4.9)\quad \begin{bmatrix}
I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & I & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & I & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\
F_{\delta+1}^{k-1} D_1^{-1} & 0 & 0 & 0 & 0 & I & 0 & 0 \\
0 & \ddots & 0 & 0 & 0 & 0 & \ddots & 0 \\
0 & 0 & F_m^{k-1} D_\delta^{-1} & 0 & 0 & 0 & 0 & I
\end{bmatrix}_{mn\times mn},
$$

where δ is the stride at level $k-1$ ($\delta = 2^{(k-2)}$). Also,

$$
(4.10)\qquad F_i^k = \begin{bmatrix} 0 & 2^{2^{(k-1)}} M_i \frac{M_{i-1}^2}{\Delta_{i-1}} \frac{M_{i-2}^2}{\Delta_{i-2}} \cdots \frac{M_{i-(2^{(k-1)}-1)}^2}{\Delta_{i-(2^{(k-1)}-1)}} \\ 0 & 0 \end{bmatrix}.
$$

To obtain $\underline{f}^k$, the restriction operator is applied repeatedly to the right-hand side of the original matrix equation until level k is reached. At level k, matrix H_k assumes the following form:

$$
(4.11)\qquad H_k = \begin{bmatrix}
D_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & D_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & D_3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \ddots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & D_\delta & 0 & 0 & 0 \\
F_{\delta+1}^k & 0 & 0 & 0 & 0 & D_{\delta+1} & 0 & 0 \\
0 & \ddots & 0 & 0 & 0 & \ddots & \ddots & 0 \\
0 & 0 & F_m^k & 0 & 0 & 0 & 0 & D_m
\end{bmatrix},
$$

where $\delta = 2^{(k-1)}$. At level $k = \log_2 m$ we have a block diagonal matrix which can be solved for each diagonal block

$$
(4.12)\qquad D_i \underline{\psi}_i = \underline{f}_i^k, \quad i = 1, \cdots, m,
$$

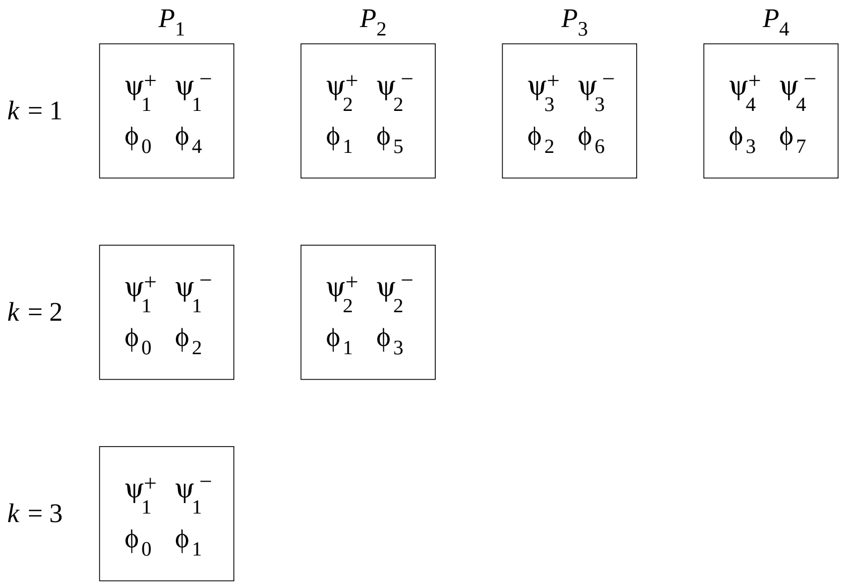

FIG. 2. *Data structure for a particular spacial grid point and for different angular levels, showing the fluxes and the Legendre moments.*

where

$$D_i^{-1} = \begin{bmatrix} \frac{\epsilon I + M_i}{\Delta_i} & \frac{-M_i}{\Delta_i} \\ \frac{M_i}{\Delta_i} & \frac{\epsilon I + M_i}{\Delta_i} \end{bmatrix}. \tag{4.13}$$

The n-vector $\underline{f}_m^k$ is obtained through multiplication through the same matrices used on the left-hand side of the matrix equations,

• **Negative variable cyclic reduction.**

The same process can be applied to the negative variable relaxation matrix. in which case we have a block upper bidiagonal (BUB) system with block matrices of size $2n \times 2n$.

5. Angular Multigrid in Parallel. The parallel algorithms described here work for both kind of architectures SIMD (CM200) and MIMD (CM5). As expected, the codes have demonstrated the same behavior on both computers due to the ability of MIMD computers to simulate SIMD architectures.

5.1. Storage. We assign one processor to each point (i.e., each (x_i, μ_j) pair). At different angular levels we have different Gauss quadrature points. Consider a

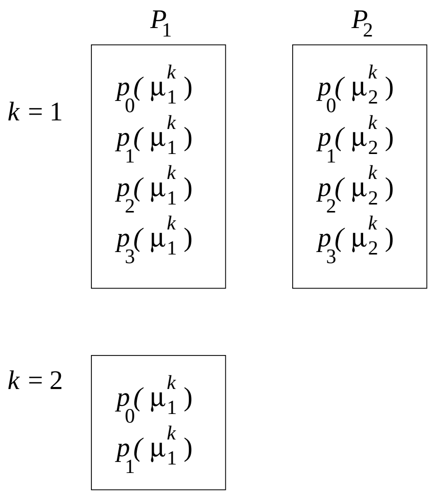

FIG. 3. *Data structure for Legendre polynomials for a particular spatial grid point and for two angular levels starting with $N = 4$.*

set of processors that contains the same spatial grid point and different angular grid points. Each one of these processors will contain one angle from each angular level. For example, processor P_1 will store μ_1 for level 1 and all the variables calculated at this point. It will also contain μ_1 for level 2. Figure 2 illustrates the distribution of data for $N = 8$ ($n = 4$) in the CM200. In our implementation we will always have two variables associated with the same quantity (one for the positive angles, and another for the negative angles), for example, $\psi^+(x_i, \mu_i)$ and $\psi^-(x_i, \mu_i)$, $q^+(x_i, \mu_i)$ and $q^-(x_i, \mu_i)$, and so on. We use some duplication of data to avoid communication. For example, variables w_j (a one-dimensional array) are stored as a two-dimensional array ($w_{i,j}$). The diagonal matrices M_i are stored as full matrices, where each column is a replication of the diagonal. At each processor we store the Legendre polynomials of all degrees evaluated for the angle associated with that processor. Figure 3 illustrates this for the case where the starting number of scattering angles is four.

5.2. Algorithm. Now we describe the steps to perform angular multigrid on the CM200.

1. Initialization.

This is performed completely with in-place calculations.

2. Calculation of the right-hand side for the shifted source iteration.

The right-hand side variables for the shifted source iteration step are assumed known. They are calculated through the following expression: $\underline{f}^l = S\underline{\psi}^l + \underline{q}$, where $S = T^{-1}DT$ (the matrix D contains the shift α). The initial guess for the flux on the right hand side of the matrix equation may be the zero vector ($\underline{\psi}^0 = 0$) or may be a nonzero vector. For a nonzero vector we notice that multiplication of the $\underline{\psi}^l$ vector by the matrix S can be performed in three stages:

- Multiplication by matrix T.
- Multiplication by the diagonal matrix D.
- Multiplication by matrix T^{-1}.

The above three steps will be performed at different points of the algorithm and will be described next. After multiplication by the matrix S is performed, we have a resultant vector in which entries are stored conformable to the entries of vector $\underline{q}$. Consequently, the addition of this vector to $\underline{q}$ takes only one parallel step. Details about the implementation of the parallel cyclic reduction are discussed in the following section.

<u>Multiplication of a vector by the $N \times N$ matrix T_k: $\underline{z} = T_k \underline{y}$</u>

Every row of matrix T , given in (2.17), will update an entry of the n-vector $\underline{z}$ for all the spatial grid points. For example, processor $P_{i,j}$ representing angle μ_j and a spatial point x_i will use row j of matrix T to update its entry (z_j) of vector $\underline{z}$. In particular, the processors that contain the entries of angle μ_1 will use the first row of matrix T for updating variables at μ_1 (z_1). Looking back at the data structure, we can observe that the multiplications of $w_j p_1(\mu_j) y_j$ will all be performed in place and are done simultaneously for all the points in the spatial and angular domain. The next step is to perform a summation across the angles. This summation is done with the CM200/CM5 intrinsic function *sum*, and is also done for all the spatial points simultaneously. The only sequential part of this matrix multiplication is due to the single instruction mode of the CM200. The different processors will use *different* polynomials (because they are updated with different rows of matrix T) to update the entries of vector $\underline{z}$, and this has to be done sequentially for this kind of data structure on an SIMD computer. This step will be executed in N different stages for the entire angular and spatial domain. We can express this as

$$
\begin{aligned}
&do \quad k = 1, N \\
&\qquad t_j = w_j p_{k-1}(\mu_j) y_j \quad for\ all\ procs_j\ (j = 1, \ldots, N) \\
&\qquad z_k = sum(t_j, 2) \\
&cont
\end{aligned}
$$

for all $i = 1, \ldots, 2m+2$ simultaneously. In MIMD computers, e.g. CM5, different processors can perform different kind of instructions simultaneously. Thus an algorithm like this could take one parallel step on MIMD computers with the appropriate data structure (each processor would be evaluating a different degree Legendre polynomial). In future work, the author will address MIMD archtectures.

Multiplication of a vector by an $N \times N$ matrix T_k^{-1}: $\underline{z} = T_k^{-1} \underline{y}$.

Multiplication by T^{-1}, given in (2.18), is performed in a manner similar to that for T. To illustrate, consider updating for the processors that store variables μ_1 and any of the $2m+2$ spatial grid points. These updates are done using the first row of matrix T^{-1}. Here we notice that the entries of this row are all stored at the processor that contains μ_1 (as is shown in Figure 3), but will be multiplied by array elements of $\underline{y}$ distributed across different processors throughout the CM200. Furthermore, it is not convenient to take advantage of the CM200 shifts since the shifts for different rows will be different for the same elements of $\underline{y}$. Consider rows 1 and 2 in (2.18): variable y_3, for example, needs a shift of 2 for row 1 and a shift of 1 for row 2. If we create a new vector that stores the partial summations for all spatial and angular points, the multiplication will be completed in N steps, where $N-1$ is the maximum degree of the polynomials at each processor. The partial summations are calculated as

$$
\begin{aligned}
&do \quad d = 1, N \\
&\qquad sp_j = sp_j + (2d-1) p_{d-1}(\mu_j) y_d \quad for\ all\ procs_j (j = 1, \ldots, n.) \\
&cont
\end{aligned}
$$

for $i = 1, \ldots, 2m+1$ simultaneously.

Multiplication of a vector by the $N \times N$ matrix D_k: $\underline{z} = D_k \underline{y}$

This multiplication is done in place since each processor has the appropriate entry of the matrix and only multiplies its own vector entry:

$$z = \sigma y$$

for $i = 1, \ldots, 2m+1$, $j = 1, \ldots, n$ simultaneously.

3. Calculation of the residual.

This step is performed completely in parallel and takes only one step for the whole spatial and angular domain.

4. Restriction of the residual (coarsening).

First, we change from the flux to moment domain. This is accomplished through the multiplication by matrix T_k and was previously described for a generic vector. Second, we perform the truncation of moments. This is obtained automatically by just disregarding the appropriate processors that store the high moments at that level. Finally, we transform our moments into fluxes through multiplication by T_{k+1}^{-1} as was described previously for a generic vector.

5. Calculation of the error on the finest grid.

This is performed by using the parallel isotropic scattering solver developed in Chapter 1.

6. Interpolation of the error.

Multiplication by matrix T is performed as described previously. The addition of the vectors $\underline{\varepsilon}_2, \underline{\varepsilon}_3, \ldots, \underline{\varepsilon}_{\log_2 n}$ are performed with in-place calculations in one parallel step.

7. Correction of the flux at the first angular level.

Multiplication by matrix T^{-1} is as previously described. The addition of the resultant vector to the flux at the finest angular level is done in-place.

6. Parallel Cyclic Reduction.

6.1. Storage and Initialization. Each column of the matrix in (4.1) will multiply data belonging to a unique cell (for each level of the cyclic reduction algorithm). Each F_i^k (4.10) has a single nonzero submatrix, that we call e_i^k, which depends on the grid level k and spatial cell i. For nonuniform grids, the matrices F_i^k are different for distinct cells. The level index for this variable is stored sequentially for each processor, as is the case for all the variables that contain a level index. The storage is done this way since the different calculation levels of the cyclic reduction have to be performed sequentially (it takes $\log_2 m$ steps and each level uses results from the preceding level). Throughout the transference of data between levels, this storage minimizes communication.

In this section we omit the angular index, since all the steps for the parallel cyclic reduction are performed for all angles simultaneously. Most of the steps for the spatial grid points will also be performed in parallel, even in the case where we have variable mesh sizes. For example, all processors will have their own definition of e_i^k and perform their calculation simultaneously. Notice that e_i^k has more product terms on coarser grid levels (larger k). The algorithm for the calculation of e_i^k on the CM200 is shown below. In the following description $shift(v, c, d)$ is a CM operation which shifts the contents of the array v either to the right ($d = 1$) or to the left ($d = -1$) along coordinate c.

$$
\begin{array}{lll}
 & do\ 2\ \ k = 1, ml & \\
 & \quad t = M & \\
 & \quad t1 = M & \\
 & \quad t2 = M & \\
 & \quad do\ 1\ \ ii = 1, 2**(k-1)-1 & \\
 & \qquad t1 = shift(t1, 2, -1) & \\
 & \qquad t2 = shift(t2, 2, -1) & \\
 & \qquad t = t1 **2 * t/t2 & \\
1 & \quad continue & \\
 & \quad e(k) = t * (2**(2**(k-1))) & \\
2 & continue &
\end{array}
$$

for all $i = 1, \ldots, 2m+2, j = 1, \ldots, n$ simultaneously, ml is the number of levels in the cyclic reduction ($ml = \log_2 m$).

6.2. Algorithm for Parallel Cyclic Reduction. Here we describe the algorithm for the *positive variable cyclic reduction*. The algorithm for the negative variable cyclic reduction is performed in a similar way.

1. Descending.

Descending for cyclic reduction at each level k is performed through the multiplication of the matrix in (4.11) by the matrix in (4.9). Note that the only nonzero elements for the latter matrix belong to the identity and block matrices like $F_i^k D_{i+\delta}^{-1}$. Matrix D_i^{-1} is calculated analytically in (4.13) and can be represented here as

$$D_i^{-1} = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix}. \tag{6.1}$$

Consequently matrix $F_i^k D_{i+\delta}^{-1}$ is represented by

$$F_i^k D_{i+\delta}^{-1} = \begin{bmatrix} e_i^k d_{21} & e_i^k d_{12} \\ 0 & 0 \end{bmatrix}, \tag{6.2}$$

Notice that this matrix has only 2 $n \times n$ nonzero submatrices: $e_i^k d_{21}$ and $e_i^k d_{12}$, which are henceforth denoted as r_1 and r_2 respectively. These submatrices will multiply q_{2i} and q_{2i+1} (source terms at the left and right sides of cell i, respectively). The first step of the coarsening algorithm on the CM200 is

$$t_1 = r_1 q \quad \text{and} \quad t_2 = r_2 q.$$

At cell i, these multiplications are required from its left neighbor (cell $i - \delta$). Thus, we need to shift the two arrays (t_1 and t_2) from the left by δ:

$$t_3 = shift(t_1, 2, -\delta) + shift(t_2, 2, -\delta),$$

where δ is the stride at step $k-1$ ($\delta = 2^{k-1}$). The algorithm to update the variables in the descending levels in the CM200 is

$$q^{k+1} = q^k$$

$$q^{k+1} = q^{k+1} + t_3$$

Notice that during this step there is no communication between the processors.

2. Solving the block diagonal system.

When we get to level $k = \log_2 m$, we have a block diagonal system as shown in (4.12). The inverse of matrix D_m is calculated analytically in (4.13) and will be represented in the following algorithm as (6.1). We can update the right and left side variables for each cell through the use of two masks that differentiate the right and left side of each spatial cell (in the matrix, the left and right sides of a cell correspond to the first n rows and the second n rows of each $2n$ block row, respectively).

$maske$ - represents all the left side variables for all the cells.

$masko$ - represents all the right side variables for all the cells.

Since the first n rows of each system $\underline{\psi}_i = D_i^{-1} \underline{f}_m^k$ represent the variables at the left side of cell m, we use $maske$ to update these variables. Recall that vector $\underline{f}_i^k$ represents the right hand side of the matrix system at level $k = \log_2 m$ for block row i. The algorithm for the left side variables update of each cell is

$t = shift(f, 1, 1)$

$where(maske)$

$\psi \ = \ d_{11} f \ + \ d_{12} t$

$endwhere$

The algorithm for the right side variable update of each cell is

$t = shift(f, 1, -1)$

$where(masko)$

$\psi \ = \ d_{21} f \ + \ d_{22} t.$

$endwhere$

Notice that the shifts at this level are both unitary shifts.

7. Stability of Cyclic Reduction. The transport sweep is performed by cyclic reduction, as was shown in Section 4. In proceeding from level k to level $k+1$, subvectors of the right-hand side vector are multiplied by $F_{i-\delta}^{k-1} D_i^{-1}$. To prove stability, it is necessary to prove that the elements of matrices F_i^k at all levels k are bounded.

For positive variables, we have shown that

(7.3) $$F_i^k = \begin{bmatrix} 0 & 2^{2^{(k-1)}} M_i \frac{M_{i-1}^2}{\Delta_{i-1}} \frac{M_{i-2}^2}{\Delta_{i-2}} \cdots \frac{M^2_{i-(2^{(k-1)}-1)}}{\Delta_{i-(2^{(k-1)}-1)}} \\ 0 & 0 \end{bmatrix},$$

where

(7.4) $$\Delta_i = 2M_i^2 + 2\epsilon M_i + \epsilon^2 I$$

and

(7.5) $$D_i^{-1} = \begin{bmatrix} \frac{\epsilon I + M_i}{\Delta_i} & \frac{-M_i}{\Delta_i} \\ \frac{M_i}{\Delta_i} & \frac{\epsilon I + M_i}{\Delta_i} \end{bmatrix}.$$

We want to prove that $\|F_i^k\| \le \|F_i^1\|$ for $k = 1, \cdots, \log_2 m$.

Lemma 7.1. *Let M_i and Δ_i be diagonal matrices, where Δ_i is given by (7.4). Then $\|M_i^2\Delta_i^{-1}\| \le \frac{1}{2}$.*

Proof. From the definition of matrix norm we have

$$\|M_i^2\Delta_i^{-1}\| = \|M_i^2(2M_i^2 + 2\epsilon M_i + \epsilon^2 I)^{-1}\| = \max_i (\frac{\mu_i}{h})^2 (\frac{h^2}{2\mu_i^2 + 2\epsilon h \mu_i + \epsilon^2 h^2}) \le \frac{1}{2}$$

■

THEOREM 7.1. *(Stability of the positive variables cyclic reduction). Let F_i^k and F_i^1 be given by (4.10) and (4.7), respectively. Then $\|F_i^k\| \le \|F_i^1\|$ for $k = 1, \ldots, \log_2 m$.*

Proof. From the definition of F_i^k, we can see that its norm is equal to its nonzero $n \times n$ block sub-matrix norm:

$$\begin{aligned} \|F_i^k\| &= \|2^{2^{(k-1)}} M_i M_{i-1}^2 {\Delta_{i-1}}^{-1} M_{i-2}^2 \Delta_{i-2}^{-1} \ldots M^2_{i-(2^{(k-1)}-1)} \Delta^{-1}_{i-(2^{(k-1)}-1)}\| \\ &= \|2^{2^{(k-1)}-1} 2M_i M_{i-1}^2 \Delta_{i-1}^{-1} M_{i-2}^2 \Delta_{i-2}^{-1} \ldots M^2_{i-(2^{(k-1)}-1)} \Delta^{-1}_{i-(2^{(k-1)}-1)}\| \end{aligned}$$

We thus have

$$\begin{aligned} \|F_i^k\| &\le 2^{2^{(k-1)}-1} 2\|M_i\| \|M_{i-1}^2 \Delta_{i-1}^{-1}\| \|M_{i-2}^2 \Delta_{i-2}^{-1}\| \|M_{i-3}^2 \Delta_{i-3}^{-1}\| \ldots \|M^2_{i-(2^{k-1}-1)} \Delta^{-1}_{i-(2^{k-1}-1)}\| \\ &\le 2^{2^{k-1}-1} 2\|M_i\| (\tfrac{1}{2})^{2^{k-1}-1} = 2\|M_i\| = \|F_i^1\|. \end{aligned}$$

■

The stability theorem for the negative variable cyclic reduction follows in a corresponding way.

TABLE 1

Convergence rates for the V(1,1) and V(1,0) angular multigrid cycles for constant m ($m = 4$) and various values of N (S_N cases).

n	2	4	8	16	32	64	128
$V(1,0)$	0.32	0.47	0.54	0.56	0.57	0.57	0.57
$V(1,1)$	0.12	0.22	0.29	0.33	0.34	0.34	0.34

8. Experimental Results. In this section we first compare the convergence factors for the two kinds of angular multigrid cycles used in this paper: $V(1,0)$ and $V(1,1)$. Then we compare the timings between one $V(1,0)$-cycle and one $V(1,1)$-cycle to show which one gives the best convergence factor considering the difference in time. The convergence factor is defined as the limit ratio between consecutive error norms. Our convergent factor estimates were these ratios after 5 angular multigrid V cycles. We also show how these algorithms behave when m is increased and when n is increased. The convergence factors are measured for the forward peaked Fokker-Planck scattering model.

Table 1 shows the convergence factors for $m = 4$ and various values of n. The convergence factors for $V(1,0)$ ($\rho_{1,0}$) and $V(1,1)$ ($\rho_{1,1}$) increase as n increases, but seem to reach an upper bound for $n \geq 32$ ($\rho_{1,0} = .57$ and $\rho_{1,1} = .34$). This agrees with the Fourier analysis in [13] which predicts $\rho \leq .36$ for the $V(1,1)$ cycle. Since we have twice as many smoothing steps in the $V(1,1)$ cycle compared with $V(1,0)$, it was not surprising that the convergence factors for $V(1,1)$ are approximately the squares of the convergence factors of $V(1,0)$ (i.e. $\rho_{1,1} \approx {\rho_{1,0}}^2$) for the various values of N. The two methods would be equivalent if a $V(1,1)$ cycle took twice as long as a $V(1,0)$ cycle. The parallel timings presented in this section were measured on a

TABLE 2

Timings in seconds for the $V(1,1)$ and $V(1,0)$ angular multigrid cycles for constant m ($m = 4$) and various values of n (S_N cases, $N = 2n$) on the CM5 and Cray Y-MP.

n	2	4	8	16	32	64	128
$V(1,0) - Cray$	0.023	0.062	0.180	0.665	2.872	14.780	87.920
$V(1,1) - Cray$	0.024	0.065	0.188	0.694	2.973	15.102	88.593
$V(1,0) - CM$	0.185	0.232	0.315	0.47	0.773	1.341	2.533
$V(1,1) - CM$	0.227	0.302	0.425	0.676	1.145	2.131	3.923

TABLE 3

Timings in seconds for the $V(1,1)$ and $V(1,0)$ angular multigrid cycles for constant m ($m = 64$) and various values of n (S_N cases, $N = 2n$) on the CM5 and Cray Y-MP.

n	2	4	8	16	32	64	128
$V(1,0) - Cray$	0.335	0.590	0.922	1.943	5.768	22.993	114.980
$V(1,1) - Cray$	0.363	0.671	1.093	2.320	6.790	26.010	125.210
$V(1,0) - CM$	0.699	0.875	1.046	1.457	2.110	3.799	7.389
$V(1,1) - CM$	0.917	1.203	1.603	2.771	3.371	6.134	11.800

TABLE 4

Comparison between $V(1,1)$ & $V(1,0)$ where (ρ_{11}, t_{11}) and (ρ_{10}, t_{10}) were the convergence factors and timings on the CM5, respectively ($m = 4$ and varying $N = 2n$).

n	2	4	8	16	32	64	128
$\rho_{1,0}^{t_{11}/t_{10}}$	0.25	0.37	0.43	0.43	0.43	0.41	0.42
$\rho_{1,1}$	0.12	0.22	0.29	0.33	0.34	0.34	0.34

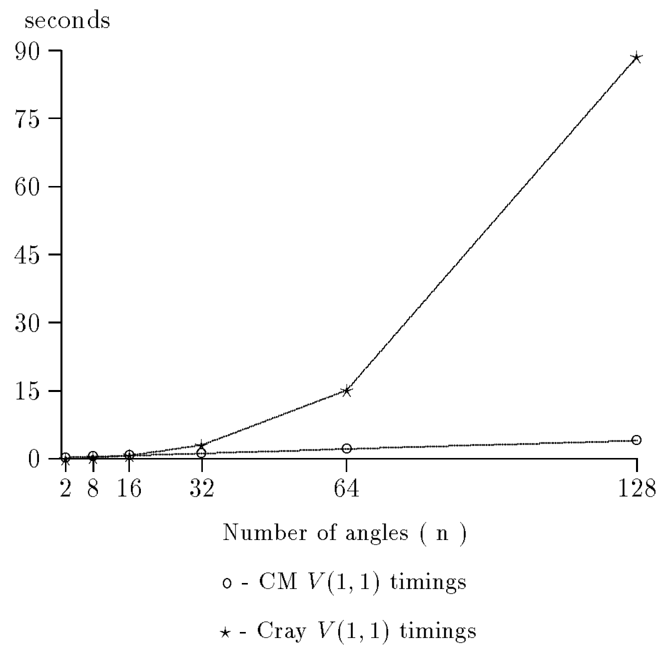

FIG. 4. *Timings for a $V(1,1)$ cycle varying (n) and $m = 4$.*

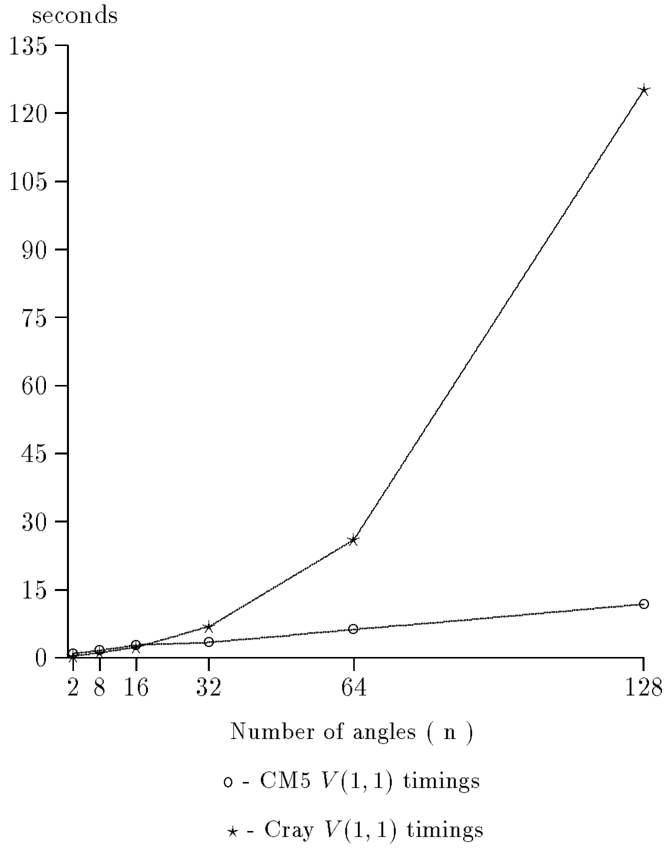

FIG. 5. *Timings for a $V(1,1)$ cycle varying (n) and $m = 64$.*

32-node CM5. The CM2 timings are in general slower than the CM5 but follow the same behaviour and conclusions. Tables 2 and 3 show the timings on the Cray Y-MP and CM5 for $m = 4$ and $m = 64$, respectively. If we consider the parallel timings on the CM5 for $V(1,1)$ (t_{11}) and $V(1,0)$ (t_{10}), we notice that using the time for $V(1,1)$ as the standard unit we can compare the convergence factors for the two algorithms (Table 4). It is obvious from this table that $V(1,1)$ is superior to $V(1,0)$ on the CM5. The same conclusions are true for our sequential codes on the Cray Y-MP. This computer has 64 vector registers of length 64 bits. For our timings we used only one of the 8 processors available on the Cray Y-MP. Graphs 4 and 5 show how much faster the CM5 $V(1,1)$ codes are than the Cray-YMP sequential version.

In general, our algorithms behaved as expected. When N increases, we notice that, even though the number of levels for the angular multigrid scheme is $\log_2 n$, the timings for the whole algorithm are proportional to N. This is caused mostly by the T^{-1} and T multiplications, which were shown to take N steps in Section 5.

Table 5

Convergence rates for the V(1,0) and V(1,1) angular multigrid cycles for $N = 16$ and various values of m.

m	4	16	32	64
$V(1,0)$	0.53	0.53	0.53	0.53
$V(1,1)$	0.22	0.28	0.28	0.28

Table 6

Timings on the CM5 for the V(1,0) and V(1,1) angular multigrid cycles for $N = 8$ and various values of m.

m	4	8	16	32	64
$V(1,0)$	.32	0.43	0.59	0.80	1.05
$V(1,1)$	0.42	0.61	0.83	1.15	1.61

This means that whenever N is doubled, the timings on the CM5 will at least double. This is exactly what is shown on Tables 2 and 3. Of course, if we consider the same algorithm sequentially, we should expect the time spent on this algorithm to be at least proportional to N^2 due to the matrix multiplications. This can be observed on the same table for the sequential timings on the Cray Y-MP.

Still for varying N, Figures 4 and 5 show the improvement of our parallel algorithms. The differences in timings are even bigger for a large number of angles. This may be significant for many problems, since a larger number of Gauss quadrature points may be needed to obtain a satisfactory approximation for the solution.

For two dimensional problems, the number of angles is even larger. Consequently an extension of these parallel algorithms for two dimensions should be very useful.

For the same number of angles, as m increases the convergence factors do not change considerably (Table 5) and the timings will increase, due mainly to the cyclic reduction step. These increases are proportional to $\log_2 m$ for the $V(1,0)$ cycle and $2\log_2 m$ for $V(1,1)$ cycle. The same happens at the coarsest level when the parallel multigrid isotropic code is used, as shown earlier in [11]. The behavior when we increase m is shown in Table 6.

9. Conclusions. An efficient parallel algorithm for solving the anisotropic neutron transport equations was developed. The sequential angular multigrid algorithm was developed before by Morel and Manteuffel [13] and contains a step that is traditionally sequential. In this paper this step was parallelized by using cyclic reduction. Depending on the way we write the matrix equations, this process can be stable or not. We wrote these equations in a convenient way, and proved their stability. Cyclic reduction is a well known algorithm for solving linear systems of equations and is available in some software libraries. These libraries, however, are written for generic matrix equations and do not exploit the specific matrix structure that we have. This makes our development of parallel cyclic reduction an important example of optional approaches for parallel computers, where paying special attention to the structure of the problem pays off. The steps we used to create an efficient parallel cyclic reduction algorithm can also be applied when cyclic reduction appears in other contexts. We improved our sequential versions, with a few compiler directives and small changes, towards more vectorizable codes, but we do not claim to have used a completely vectorized implementation in our measurings. From our experimental results we conclude that the $V(1,1)$-cycle is, in general, preferable to the $V(1,0)$-cycle on the CM5. Notice that both of these algorithms are faster than their sequential versions; our parallel timings behaved like $O(N)$, while our sequential codes were like $O(N^2)$. This behavior was dictated by the transformations from flux domain to moment domain and vice-versa, which are accomplished through multiplications by $N \times N$ matrices. These matrix multiplications represent the numerical tools used to obtain a Legendre expansion of a function f (multiplication by T) and the evaluation of a function at N nodes, given an N-term Legendre expansion (multiplication by T^{-1}). The sequential steps for these algorithms are known to be $O(N^2)$ and recently a *fast algorithm* was developed in [1], which takes $O(N \log N)$ sequential steps. However, this algorithm requires evaluation at the Chebyshev points in $[-1, 1]$ rather than Gauss quadrature which would imply that the transformation between the moment and flux representation is not exact. Our data parallel algorithm for the Legendre transforms is $O(N)$. It represents an improvement that can be used in many applications including algorithms involving quadratures, approximation theory, and the solution of PDE's. In this paper we dealt with non-absorption scattering cases ($\sigma_s/\sigma_t = 1$). The work developed here was for the one-dimensional transport equations, but the same kind of algorithms can be applied for two-dimensional and three-dimensional transport equations. The improvement from using cyclic reduction in higher dimensions will not be as straighforward as here, unless further parallel algorithms are developed. There will still be some improvement, due to (in the worst case) the replacement of a linear factor with a logarithmic one in the complexity estimate [15]. In addition to choosing very accurate methods when using parallel computers, we should search for algorithms that take advantage of the kind of architecture we are using. The best algorithms thus will represent a compromise between accuracy and time efficiency, will be architecture dependent, and will evolve as new architectures are designed.

Acknowledgment. I would like to thank my PhD advisor Tom Manteuffel for his support while I was at the University of Colorado at Denver. Also I would like to thank Jim Morel for his help while I was at Los Alamos National Laboratory. Without these two interactions this research would not have succeeded.

REFERENCES

[1] Bradley K. Alpert and Vladimir Rokhlin, "A Fast Algorithm for the Evaluation of Legendre Expansions", *SIAM Journal of Scientific and Statistical Computing*, 12, No.1, pp. 158-179, January 1991.

[2] Kenneth M. Case and P. F. Zweifel, "Linear Transport Theory", Reading, Mass.: Addison-

Wesley, 1967.

[3] Carlo Cercignani, "The Boltzman Equation and Its Applications", New York, NY: Springer-Verlag, 1988.

[4] J. J. Duderstadt and W. R. Martin, "Transport Theory", New York, NY: John Wiley and Sons, 1979.

[5] V. Faber and T. A. Manteuffel, "Neutron Transport from the Viewpoint of Linear Algebra", Invariant Imbedding and Equations (Nelson, Faber, Manteuffel, Seth, and White, eds.) *Lecture Notes in Pure and Applied Mathematics*, 115 pp. 37–61, Marcel-Dekker, April, 1989.

[6] Gene H. Golub and C. F. Van Loan, "Matrix Computations", Baltimore and London: The Johns Hopkins University Press, 1989.

[7] M. S. Lazo and J. E. Morel, "A Linear Discontinuous Galerkin Approximation for the Continuous Slowing Down Operator", *Nucl. Sci. Eng.*, 92, pp. 98, 1986.

[8] E. W. Larsen and J. E. Morel, "Asymptotic Solutions of Numerical Transport Problems in Optically Thick Diffusive Regimes II", *J. Comp. Phys.* 83, p. 212, 1989.

[9] E. E. Lewis and W. F. Miller, "Computational Methods of Neutron Transport", New York, NY: John Wiley and Sons, 1984.

[10] T. Manteuffel, S. McCormick, J. Morel, S. Oliveira, and G. Yang, "Fast Multigrid Solver for Transport Problems I: Pure Scattering", to appear in *SIAM J. of Sci. Computing*, vol. 16, No. 3, May 1995.

[11] T. Manteuffel, S. McCormick, J. Morel, S. Oliveira, and G. Yang, "A Parallel Version of A Multigrid Algorithm for Isotropic Transport Equations", published by *SIAM Journal of Scientific and Statistical Computing*, vol. 15 no. 2, pp. 474–493, March 1994.

[12] J. E. Morel and E. W. Larson, "A New Class of S_N Spatial Differencing Schemes", *Nucl. Sci. Eng.*, 1988.

[13] J. E. Morel and T. A. Manteuffel, "An Angular Multigrid Acceleration Technique for the S_n Equations with Highly Forwarded-Peaked Scattering", *Nucl. Sci. Eng.*, 107, pp. 330–342, 1991.

[14] R. Sweet, "A Generalized Cyclic-Reduction Algorithm", *SIAM Journal of Numer. Anal.*, Vol. 11, pp. 506–520, 1974.

[15] R. Sweet, "A Cyclic Reduction Algorihtm For Solving Block Tridiagonal Systems Of Arbritary Dimension" *SIAM Journal of Numer. Anal.*, Vol. 14, pp. 706–, 1977.

[16] P. Wesseling "An Introduction to Multigrid Methods", Chichester, England: John Wiley and Sons, 1992.

[17] G. Milton Wing. "An Introduction to Transport Theory", New York, NY: John Wiley and Sons, 1962.