

“Transpose Free” Alternating Direction Smoothers for Serial and Parallel Multigrid Methods

Craig C. Douglas^{*,‡} Sachit Malhotra[†] Martin H. Schultz[‡]

Abstract

Alternating Direction Implicit (ADI) methods are very good smoothers for multigrid. Like multigrid itself, ADI propagates information very quickly across a grid. On parallel processors, ADI is very inefficient due to the tridiagonal solves in each of the spatial directions. In one direction, the data typically resides in one processor. In the other directions, the data spans processor memories on distributed memory machines.

In this paper, a “transpose free” variant of ADI is considered which eliminates the drawback of ADI on parallel processors. In addition, it is quite useful on serial computers. We provide convergence rates for a model problem and numerical results for variable coefficient elliptic problems in two and three dimensions.

Key words: multigrid, alternating direction implicit methods, iterative methods, parallel computing, elliptic partial differential equations.

1 Introduction

Alternating Direction Implicit (ADI) methods were originally developed for the solution of multidimensional elliptic and parabolic differential equations. The first use of ADI methods was to reduce the solution of equations in higher dimensions to the solution of multiple tridiagonal systems [13, 16], for which efficient algorithms are available. ADI methods continue to be used as smoothers for multigrid methods [15, 20] in the serial case. However, in the case of computing on parallel computers, the solution of tridiagonal systems is no longer an inexpensive operation.

In the case of computing on parallel computers, the solution of tridiagonal systems is efficient only so long as the required data is wholly contained in a single processor. Typically, all the solves in one of the directions involves either a global transpose of the data (so that all the data for an individual solve is in a single processor) or a global communication step involving some form of a recursive doubling strategy e.g., [2]. In either case, this imposes an overhead which can dramatically reduce the parallel efficiency of the computation. This effect is particularly pronounced on parallel machines with high latency networks such as networks of workstations connected by an ethernet.

^{*}Department of Mathematics, University of Kentucky, 715 Patterson Office Tower, Lexington, KY 40506-0027, USA.

[†]Scientific Computing Associates, One Century Tower, 265 Church Street, New Haven, CT 06510-7010, USA.

[‡]Department of Computer Science, Yale University, P.O. Box 20-8285, New Haven, CT 06520-8285, USA.

The reduction of the parallel overhead in the solution the resulting tridiagonal systems has been a focus of much research [9, 10, 17, 18, 19, 21]. In this paper we discuss an approach to decrease the overhead imposed by the solution of the corresponding tridiagonal systems by approximating the tridiagonal solves in one direction with Gauss–Seidel iterations.

The tridiagonal matrices to be inverted for elliptic problems are strongly diagonally dominant. The Gauss–Seidel approximation converges to the solution of the original system quickly. This implies that a very small number of Gauss–Seidel sweeps are required to obtain an accurate approximation. Aggregation of the sweeps, as outlined in [14], can further improve the performance of this approximation on parallel machines.

We present numerical experiments on two platforms to validate our claim: an IBM SP2 with thin nodes and a network of SUN SPARC10 workstations connected by an ethernet. The methods of this section outperform conventional ADI methods on both serial and parallel computers.

In the single processor case, the approximation offers a significant speedup over ADI due to its lower arithmetic costs. Furthermore, the Gauss–Seidel method uses data within the cache far more effectively than conventional tridiagonal solvers. Since the solvers sweep over the data twice, the data within the cache is invalidated for all but the smallest problems. Red–black Gauss–Seidel, on the other hand, can be implemented so that only one pass over the data is required for a number of sweeps. On machines such as modern RISC processors, where the cost of a cache miss is very high, this provides a significant speedup over the use of tridiagonal solvers.

For the solution of large problems, where we might have to resort to the use of virtual memory during the course of the computation, the more efficient use of local data has an even greater effect on the total running time of the program.

On parallel processors, ADI methods suffer from the additional disadvantage of having to store the partial factors of the matrix during the substructuring phase. This imposes an increased overhead in both arithmetic costs and memory requirements. The approximate method, on the other hand, does not suffer from the same disadvantages and hence provides a significant speedup over the conventional ADI method. In fact, using the concept of redundant computation, larger number of Gauss–Seidel sweeps can be implemented with a single sweep over the data. This ensures that the locality of reference of the data can be maintained even on parallel processors, with the advantages of better utilization of the cache.

We shall investigate the use of ADI methods as smoothers for general elliptic problems. The tridiagonal matrices to be inverted for these problems are strongly diagonally dominant. As a result, a small number of Gauss–Seidel sweeps are required to generate an accurate approximation to the solution. This approximation can be parallelized with limited communication overhead. The aggregation of the sweeps, as outlined in [14], ensures that all the Gauss–Seidel iterations required can be completed with a single set of communication events between neighboring processors.

2 Classical ADI

2.1 A Simple Description

In this section we review the classical ADI algorithm which will motivate the main results of this paper. We shall limit our discussion to the solution of the inhomogeneous constant coefficient equation on a square

domain. The continuous problem is defined by

$$-u_{xx}(x, y) - u_{yy}(x, y) + \sigma u(x, y) = S(x, y), \quad (x, y) \in R, \quad (1)$$

on the unit square, $R: 0 < x, y < 1$ of the x - y plane, where $\sigma \geq 0$ and

$$u(x, y) = \gamma(x, y), \quad (x, y) \in \partial R,$$

where $\gamma(x, y)$ is a prescribed function on the boundary ∂R [22]. The problem is discretized by imposing a uniform $N \times N$ grid on the unit square with a grid spacing given by $h \equiv 1/(N + 1)$. The problem then reduces to the solution of the linear system

$$Au = k \quad (2)$$

where

$$A = H + V + \Sigma.$$

The matrices H and V are the discretized versions of the differential operators in the horizontal and vertical directions respectively. The matrix Σ is a non-negative diagonal matrix. Each of the matrices H and V are each *completely reducible* [22]. In particular, they are reducible to the direct sum of irreducible Stieltjes matrices [22]. For the rest of the discussion we shall ignore the matrix Σ , which can be incorporated into the discussion by defining the modified matrices [13]

$$H' = H + \Sigma/2$$

$$V' = V + \Sigma/2.$$

The ADI method proceeds by converting (2) to a pair of matrix equations

$$(H + \rho I)u = k - (V - \rho I)u \quad (3)$$

$$(V + \rho I)u = k - (H - \rho I)u \quad (4)$$

for some positive scalar ρ . The iterative method is then defined as

$$(H + \rho I)u^{m+\frac{1}{2}} = k - (V - \rho I)u^m, \quad (5)$$

$$(V + \rho I)u^{m+1} = k - (H - \rho I)u^{m+\frac{1}{2}}, \quad m \geq 0, \quad (6)$$

where u^0 is an arbitrary initial vector approximation. The matrices $H + \rho I$ and $V + \rho I$ can be converted to tridiagonal matrices after suitable permutations of the rows and columns and the systems (5) and (6) are easily solved.

If the exact solution of the discretized problem (2) is denoted by u^* and the error e^m is defined by

$$e^m \equiv u^m - u^*,$$

it follows that

$$e^{m+1} = T_\rho e^m$$

where

$$T_\rho = (V + \rho I)^{-1}(H - \rho I)(H + \rho I)^{-1}(V - \rho I). \quad (7)$$

The above method converges to the solution for any positive ρ [22, Theorem 7.1]. In fact, for the model problem (2), the rate of convergence for the optimal choice of ρ is the same as for SOR with the optimal overrelaxation parameter.

3 Gauss Seidel Iteration

We investigate the effect of replacing the tridiagonal solve for one of the ADI half steps with Gauss-Seidel sweeps. In this section, we restate some of the properties of the Gauss-Seidel method as applied to the matrices obtained by the discretization of H and V in (2).

The Gauss-Seidel method is preferred to the Jacobi method since it converges twice as fast as the Jacobi method [22]. Furthermore, it does not require any auxiliary storage unlike the Jacobi method.

The eigenvectors of the operators H and V in (2) are given by

$$v_{j,m} = \sin(\pi jy) \sin(\pi mx) \quad j, m = 1, \dots, N.$$

For (2), using Gauss–Seidel with red-black ordering, we state the following result:

Theorem 3.1 *Red-black Gauss–Seidel applied to the solution of $(H + \rho I) = r$ leaves the two-dimensional space spanned by the eigenvectors $v_{j,m}$ and $v_{j,m'}$ invariant, where*

$$m' = N + 1 - m.$$

Hence on V_m , the subspace spanned by $v_{j,m}$ and $v_{j,m'}$, red-black Gauss-Seidel has the matrix representation

$$M_\rho^1 = \begin{pmatrix} \delta \left(\frac{1+\delta}{2}\right) & \delta \left(\frac{1+\delta}{2}\right) \\ -\delta \left(\frac{1-\delta}{2}\right) & -\delta \left(\frac{1-\delta}{2}\right) \end{pmatrix} \quad (8)$$

with

$$\delta = \frac{2 - \lambda_m}{2 + \rho}. \quad (9)$$

k applications of the red-black point Gauss–Seidel method has a representation given by

$$M_\rho^k = \begin{pmatrix} \delta^{2k-1} \left(\frac{1+\delta}{2}\right) & \delta^{2k-1} \left(\frac{1+\delta}{2}\right) \\ -\delta^{2k-1} \left(\frac{1-\delta}{2}\right) & -\delta^{2k-1} \left(\frac{1-\delta}{2}\right) \end{pmatrix}. \quad (10)$$

Proof: Equation (8) is a restatement of a well known result [7]. Equation (10) is derived from repeated applications of the matrix in (8). ■

Corollary 3.1 *In the subspace $V_m = \text{span} \{v_{j,m}, v_{j,m'}\}$, let u^0 be an arbitrary initial iterate and u^* be exact solution to the problem*

$$(H + \rho I)u = r.$$

k applications of the Gauss–Seidel method generates the iterate u^k where

$$u^k = (I - M_\rho^k)u^* + M_\rho^k u^0. \quad (11)$$

As can be seen, the red-black Gauss-Seidel method reduces the error for eigenvectors with eigenvalues close to 2 (where δ is small) rapidly whereas the error along the extremal eigenvectors (with λ_m close to 0 or 4) dies down slowly. In particular, after k applications of the method, for the eigenspace defined by

$$V_{j,m} = \text{span} \{v_{j,m}, v_{j,m'}\} \quad (12)$$

we have

$$\max |v_{j,m}^k, v_{j,m'}^k| \leq \delta^{2k-1} (1 + \delta) \max |v_{j,m}^0, v_{j,m'}^0|.$$

The error in the eigenspace is uniformly bounded and decreases with the number of iterations.

4 The ADG(ρ, k) Method

In this section we analyze the properties of an ADI iteration with the tridiagonal solves for one half step replaced by k Gauss–Seidel sweeps. We denote this approximate method by ADG(ρ, k) where ρ is the ADI parameter used and k is the number of Gauss–Seidel iterations used to approximate the tridiagonal solve.

In Section 4.1 we apply the results of this section to the analysis of ADG(ρ, k) as a smoother for multigrid methods.

The convergence rate of any method which approximates ADI can be analyzed as a perturbation of the ADI method.

Lemma 4.1 *In the eigenspace $V_{j,m}$ defined in (12), let $u_{j,m}^*$ be the solution to the problem defined by (2), $u_{j,m}^0$ be an arbitrary initial iterate, $u_{j,m}^{ADI}$ be the iterate produced by a single iteration of ADI ((5) and (6)) and $u_{j,m}^{ADG}$ be the iterate produced by a single application of the ADG(ρ, k) method. If*

$$\begin{aligned} d_{j,m} &\equiv u_{j,m}^0 - u_{j,m}^*, \\ \Delta_{j,m} &\equiv u_{j,m}^{ADG} - u_{j,m}^{ADI} \end{aligned}$$

and

$$D_\rho \equiv \begin{pmatrix} \lambda_m - \rho & 0 \\ 0 & \lambda_{m'} - \rho \end{pmatrix},$$

then in $V_{j,m}$ we have

$$\Delta_{j,m} = \frac{1}{\lambda_j + \rho} D_\rho M_\rho^k \begin{pmatrix} d_{j,m} \\ d_{j,m'} \end{pmatrix}. \quad (13)$$

with M_ρ^k defined in (10).

Proof: The result can be proved by considering the effect of each half step of ADG(ρ, k) on the eigenvectors of the matrix A . The eigenvectors of A have the form

$$v_{j,m} = \sin(\pi m x) \sin(\pi j y).$$

The horizontal ADG(ρ, k) sweep, which is defined by

$$(H + \rho I)^{GS} u^{\frac{1}{2}} = b - (V - \rho I) u^0$$

(where the superscript GS refers to a Gauss–Seidel iteration being performed), produces an approximation to the ADI iterate. The iteration can be thought of producing the ADI iterate and an additional term $\Delta_{GS}^{\frac{1}{2}}$. The iterate is given by

$$u_{j,m,jm'}^{1/2} = u_{ADI}^{1/2} + \Delta_{GS}^{1/2} \quad (14)$$

where, using (10),

$$\Delta_{GS}^{1/2} = M_\rho^k \begin{pmatrix} d_{j,m} \\ d_{j,m'} \end{pmatrix}.$$

Application of (6) (i.e. the second half step) maps $v_{j,m}$ to

$$\frac{\lambda_m - \rho}{\lambda_j + \rho} v_{j,m}.$$

The operator, when applied to $u_{j,m,jm'}^{1/2}$ (14) maps the iterate to

$$u^{ADG} = u^{ADI} + \frac{1}{\lambda_j + \rho} \begin{pmatrix} \lambda_m - \rho & 0 \\ 0 & \lambda_{m'} - \rho \end{pmatrix} \Delta_{GS}^{1/2}.$$

The result of the lemma follows. \blacksquare

Theorem 4.1 *In the eigenspace $V_{j,m}$ defined in (12), let u^* be the solution to the problem defined by (2), u^0 be an arbitrary initial iterate and u^{ADG} be the iterate produced by a single application of the ADG(ρ, k) method. If*

$$e_{j,m}^0 \equiv u_{j,m}^0 - u_{j,m}^*$$

and

$$e_{j,m} \equiv u_{j,m}^{ADG} - u_{j,m}^*$$

we have

$$\|e_{j,m}\| \leq (\|T_\rho\| + \frac{\delta^{2k}}{2} \left| \frac{\lambda_m - \rho}{\lambda_j + \rho} \right|) \|e_{j,m}^0\|$$

with T_ρ defined in (7) and δ defined in (9).

Proof: Since

$$e_{j,m} = u_{j,m}^{ADG} - u_{j,m}^* = (u_{j,m}^{ADG} - u_{j,m}^{ADI}) + (u_{j,m}^{ADI} - u_{j,m}^*)$$

applying Lemma 4.1 and (7) we get

$$\|e_{j,m}\| \leq \|T_\rho\| \|e_{j,m}^0\| + \|\Delta_{j,m}\|,$$

where

$$\Delta_{j,m} = \frac{\lambda_m - \rho}{\lambda_j + \rho} M_\rho^k e_{j,m}^0.$$

Since

$$\|M_\rho^k\| = \frac{\delta^{2k}}{2},$$

where δ is defined in (9), we have

$$\|\Delta_{j,m}\| \leq \frac{\delta^{2k}}{2} \left| \frac{\lambda_m - \rho}{\lambda_j + \rho} \right| \|e_{j,m}^0\| \tag{15}$$

or

$$\|e_{j,m}\| \leq (\|T_\rho\| + \frac{\delta^{2k}}{2} \left| \frac{\lambda_m - \rho}{\lambda_j + \rho} \right|) \|e_{j,m}^0\|. \quad \blacksquare$$

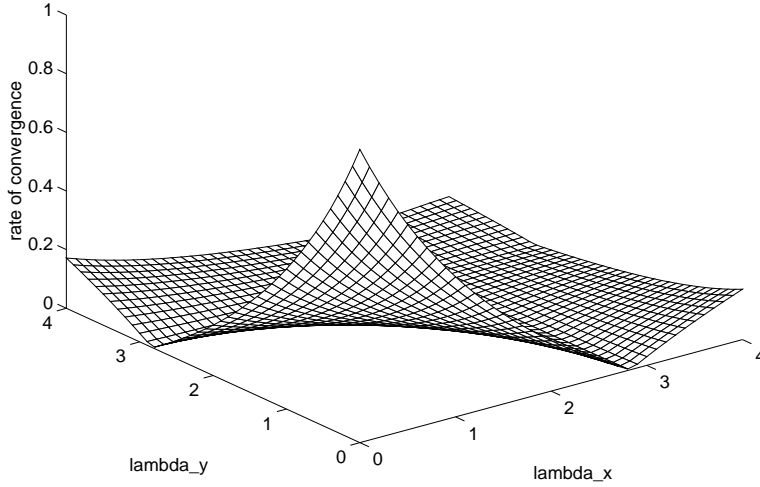


Figure 1: ADI in two dimensions: effect on different $v_{j,m}$

4.1 $ADG(\rho,k)$ as a Multigrid Smoother

As can be seen from Figure 1 ADI methods reduce the error for the high frequency components of the error substantially every iteration. In this section we investigate the use of $ADG(\rho,k)$ as a preconditioner for general elliptic problems. Although the analysis in this section is for the model problem (2) numerical experience suggests that the $ADG(\rho,k)$ method works well in practice as a preconditioner for non constant coefficient problems.

Theorem 4.2 *For the model problem defined by (2), the two grid method [1], using s smoothing steps of $ADG(\rho,k)$ at the finer level, converges with*

$$\|e^{n+1}\| \leq r_{s,\rho,k} \|e^n\|$$

with $r_{s,\rho,k}$ given in Table 1.

Proof: Assume that at any multigrid level l , the domain R for (1) is discretized uniformly using $N_l = 2^l - 1$ points in each direction. If the discretization of the operator A (2) at level l is denoted by A_l , the eigenvectors of A_l are given by

$$v_{\nu\mu}^l(i,j) = \sin(\nu\pi ih_l) \sin(\mu\pi jh_l) \quad \nu, \mu = 1, \dots, N_l.$$

The corresponding eigenvalues $\lambda_{\nu\mu}^l$ are given by

$$\lambda_{\nu\mu}(A_l) = \frac{4}{h_l^2} \left[\sin^2 \left(\frac{\nu\pi h_l}{2} \right) + \sin^2 \left(\frac{\mu\pi h_l}{2} \right) \right]$$

s	$ADI (\equiv ADG(\sqrt{8}, \infty))$	$ADG(\sqrt{8}, 1)$	$ADG(\sqrt{8}, 2)$	$ADG(\sqrt{8}, 3)$
1	0.1728	0.2422	0.1725	0.1715
2	0.0923	0.1039	0.0898	0.0920
3	0.0627	0.0677	0.0572	0.0602

Table 1: $r_{s,\rho,k}$ for multigrid using $ADG(\rho,k)$ as a smoother

where $h_l = 1/(N_l + 1) = 2^{-l}$ [8]. In the eigenspace $\alpha^{\nu\mu}$,

$$\alpha_{\nu\mu} = \text{span} \{v_{\nu\mu}, v_{\nu'\mu}, v_{\nu\mu'}, v_{\nu'\mu'}\}$$

where

$$\nu' = N_l - \nu + 1 \quad \text{and} \quad \mu' = N_l - \mu + 1$$

the operator has the representation

$$L_l^{\nu\mu} = \frac{4}{h_l^2} \begin{pmatrix} s_\nu^2 + s_\mu^2 & & & \\ & c_\nu^2 + s_\mu^2 & & \\ & & s_\nu^2 + c_\mu^2 & \\ & & & c_\nu^2 + c_\mu^2 \end{pmatrix},$$

where $s_\nu = \sin(\frac{\nu\pi h_l}{2})$ and $c_\nu = \cos(\frac{\nu\pi h_l}{2})$ [8]. The standard nine point restriction operator $\mathcal{R} : l \rightarrow l - 1$ has a representation

$$\mathcal{R} = [c_\nu^2 c_\mu^2, -s_\nu^2 c_\mu^2, -c_\nu^2 s_\mu^2, s_\nu^2 s_\mu^2]$$

on $\alpha_{\nu,\mu}$ [1, 8]. The standard nine point prolongation operator $\mathcal{I} : l - 1 \rightarrow l$ has the representation [1, 8]

$$\mathcal{I} = \begin{bmatrix} c_\nu^2 c_\mu^2 \\ -s_\nu^2 c_\mu^2 \\ -c_\nu^2 s_\mu^2 \\ s_\nu^2 s_\mu^2 \end{bmatrix} = \mathcal{R}^t.$$

For s applications of an operator G as a smoother, the error for the two grid method propagates as [11]

$$e^{n+1} = (I - \mathcal{I}(A_{l-1})^{-1} \mathcal{R} A_l) G^s e^n$$

For $ADG(\rho,k)$, using Lemma 4.1 and (11), we have

$$G_{ADG(\rho,k)} = (I - M_\rho^{ik})(V + \rho I)^{-1}(H - \rho I)(H + \rho I)^{-1}(V - \rho I) - M_\rho^{ik}(H + \rho I)^{-1}(V - \rho I)$$

where M_ρ^k is a block diagonal matrix with 2×2 blocks given by

$$M_\rho' = \begin{pmatrix} M_\rho(\delta(\nu)) & \\ & M_\rho(\delta(\nu')) \end{pmatrix},$$

with $M_\rho(\nu)$ defined in (8) with

$$\delta = \frac{2 - 4s_\nu^2}{2 + \rho}.$$

The analysis of the two grid method now reduces to the analysis of 4×4 matrices corresponding to the different eigenspaces $\alpha_{\nu\mu}$. The optimum rate of convergence is achieved at $\rho = \sqrt{8}$. ■

As can be seen from Table 1, a very small number of Gauss–Seidel steps (one or two) are required to maintain the rate of convergence of the multigrid method using ADI as a smoother. Although the analysis in Theorem 4.2 is for the two grid method for Poisson’s equation, we have found that one Gauss–Seidel sweep (i.e. an application of $ADG(\rho, 1)$) is sufficient in practice for more general problems. The spectral norm of the iteration operator (see Table 1) converges very rapidly to that for ADI. Although for the model problem, the spectral norm for $ADG(\rho, 2)$ is smaller than that for $ADG(\rho, 3)$ we cannot expect this to be true for all problems in general. However, the conclusions about $ADG(\rho, k)$ being an effective approximation for ADI hold for more general problems.

Since the application of ADG has lower arithmetic cost than ADI (approximately $17n^2$ flops for $ADG(\rho, 1)$ compared to $22n^2$ flops for ADI on a $n \times n$ grid on a single processor) we recommend the use of this method in situations where ADI is used as a smoother for multigrid problems. On parallel processors, where the cost of substructuring effectively doubles the cost arithmetic costs of the tridiagonal solves in one direction [18], the saving in the arithmetic costs is even higher (approximately $17n^2$ flops for the $ADG(\rho, 1)$ compared to $31n^2$ flops for ADI on a $n \times n$ grid on multiple processors).

At this stage, we point out that ADI smoothers are not best for the solution of the model problem (2). For the constant coefficient problem, full Gauss–Seidel is probably the most efficient smoother due to its low arithmetic costs per iteration [4]. From our own experiments on RISC based architectures, we have observed that a single $ADG(\rho, 1)$ iteration has arithmetic costs which are approximately between two and three times those for a single full Gauss–Seidel iteration. For variable coefficient problems however, ADI methods are often used since their higher arithmetic costs are offset by better smoothing properties.

4.2 Numerical Results

In this section, we describe the performance of the $ADG(\rho, k)$ method discussed in Section 4.1. The test problems are linear systems arising from the discretization of elliptic partial differential equations.

4.2.1 Variable Coefficient Problems on a Square Grid

The experiments in this section relate to the solution of a non self-adjoint, non separable equation (16) on the unit square.

$$-(e^{-xy}u_x)_x - (e^{xy}u_y)_y + [(0.5 - x)u_x + (0.5 - y)u_y] + \frac{u}{1 + x + y} = g(x, y) \quad (16)$$

N	procs	$ADG(\sqrt{8}, 1)$	ADI
1024	2	13.46	28.22
	4	6.67	11.43
	8	3.39	6.12
512	2	2.43	3.63
	4	1.64	3.63
	8	0.95	1.49

Table 2: Timing for the solution of a general 2D elliptic PDE using $ADG(\rho, 1)$, IBM SP2

The right hand side $g(x, y)$ was chosen such that

$$u(x, y) = x \sin(\pi x) \sin(\pi y).$$

The equation solved on the unit cube was given by

$$-(e^{-xyz}u_x)_x - (e^{xyz}u_y)_y - (e^{-xyz}u_z)_z + \frac{u}{1+x+y+z} = g(x, y, z) \quad (17)$$

where $g(x, y, z)$ was chosen such that

$$u(x, y, z) = x \sin(\pi x) \sin(\pi y) \sin(\pi z).$$

The error was measured as the two norm of the difference of the computed and the exact solution. In this case, we have found that the application of $ADG(\rho, k)$ competitive with other smoothers such as Gauss–Seidel with red–black ordering. For (16) we have found that the use of a single $ADG(\rho, 1)$ sweep as effective as the use of four Gauss–Seidel sweeps as a smoother. In this case, we find that the total runtime for the use of $ADG(\rho, k)$ as a smoother is less than the use of, say, Gauss–Seidel as a smoother.

Table 2 contains timings from the execution of the code on the IBM SP2 with thin nodes. Table 3 contains timing for the solution of (17) of the problem on the IBM SP2.

5 Conclusions

We have attempted to show that the $ADG(\rho, k)$ method is an effective approximation to the ADI method – especially in the context of a multigrid smoother. The lower computational cost of $ADG(\rho, k)$ coupled with its parallelizability make it an effective smoother for elliptic problems.

Due to the lower memory requirements of the $ADG(\rho, k)$ method and the effective use of data in the cache this approach offers a significant speedup in the case of RISC processors. Since the method is trivially parallelizable it can also be used as an effective smoother for multigrid problems on parallel machines. Since

N	procs	$ADG(\sqrt{8}, 1)$	ADI
64	1	3.93	7.88
	2	2.40	4.43
	4	1.32	2.68
	8	0.72	1.64
	16	0.50	1.27
32	1	0.49	0.95
	2	0.39	0.60
	4	0.24	0.42
	8	0.15	0.33
	16	0.28	0.43

Table 3: Timing for the solution of a general 3D elliptic PDE $ADG(\rho, 1)$, IBM SP2

the parallel overhead of the scheme can be further reduced by the aggregation of the Gauss–Seidel steps as outlined in [14], the method is effective even on parallel machines with high latency.

The combination of the approximation with redundant computation allows us to complete an iteration with a single communication event between neighboring processors, as opposed to $O(\log p)$ communication events on p processors.

In the case of problems with variable coefficients, the $ADG(\rho, k)$ approximation allows us to compute the solution without storing the factors of the matrix. This allows for much larger problems to be solved without resorting to the use of virtual memory.

The ideas of this paper can also be applied to alternating direction line SOR methods with little or no modification. Alternating direction line SOR methods are often used as smoothers for multigrid methods since their smoothing properties are slightly better than the corresponding pointwise smoothers [5] (a smoothing rate of 0.2 for line SOR vs a smoothing rate of 0.25 for point SOR).

References

- [1] R. E. Bank and C. C. Douglas. Sharp estimates for multigrid rates of convergence with general smoothing and acceleration. *SIAM Journal of Numerical Analysis*, 22:617–633, 1985.
- [2] T. F. Chan, K. R. Jackson, and B. Zhu. Alternating direction incomplete factorizations. Technical Report YALEU/DCS/tr208, Department of Computer Science, Yale University, August 1981.

- [3] C. C. Douglas. *MPI multigrid codes for 2D and 3D elliptic problems*. MGNET at <http://na.cs.yale.edu/mgnet/www/Codes/douglas>, 1995.
- [4] C. C. Douglas. Private communication, August 1996.
- [5] I. S. Duff, R. G. Grimes, and J. G. Lewis. User's guide for the Harwell–Boeing sparse matrix collection (Release I). Technical Report TR/PA/92/86, CERFACS, Toulouse, 1992.
- [6] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. The MIT Press, 1994.
- [7] W. Hackbusch. On the multi–grid method applied to difference equations. *Computing*, 20:291–306, 1978.
- [8] W. Hackbusch. *Multi–Grid Methods and Applications*. Springer–Verlag, Berlin, 1985.
- [9] M. Hatzopoulos. Parallel linear system solvers for tridiagonal systems. In D. J. Evans, editor, *Parallel Processing Systems*. Cambridge University Press, Cambridge, 1982.
- [10] D. Heller, D. Stevenson, and J. Traub. Accelerated iterative methods for the solution of tridiagonal linear systems on parallel computers. *J. ACM*, 23:636–654, 1976.
- [11] D. C. Jespersen. Multigrid methods. In Gene H. Golub, editor, *Studies in Numerical Analysis*, volume 24, pages 270–318. The Mathematical Association of America, Rhode Island, 1984.
- [12] L. Johnsson, Y. Saad, and M. Schultz. Alternating direction methods on mutiprocessors. Technical Report YALE/DCS/tr381, Department of Computer Science, Yale University, 1985.
- [13] J. Douglas Jr. and H.H. Rachford Jr. On the numerical solution of heat conduction problems in two or three space variables. *Trans. Amer. Math. Soc.*, 82:421–439, 1956.
- [14] Sachit Malhotra. *Topics in Multigrid Methods*. PhD thesis, Yale University, 1996.
- [15] G. I. Marchuk. Splitting and alternating direction methods. In P. G. Ciarlet and J.-L. Lions, editors, *Handbook of Numerical Analysis*, volume 1, pages 197–462. North–Holland, Amsterdam, 1990.
- [16] D. W. Paceman and H. H. Rachford Jr. The numerical solution of parabolic and elliptic differential equations. *J. of Soc. Indust. Appl. Math*, 3:28–41, 1955.
- [17] D. J. Rose. An algorithm for solving a special class of tridiagonal systems of linear equations. *Comm. ACM*, 12:234–236, 1969.
- [18] F. Saied. *Numerical Techniques for the Solution of the Time–dependent Schrödinger Equation and their Parallel Implementation*. PhD thesis, Yale University, 1990. Available as YALEU/DCS/tr811, Department of Computer Science, Yale University.
- [19] F. Saied, C. T. Ho, S. L. Johnsson, and M. H. Schultz. Solving Schrödinger's equation on the Intel iPSC by the Alternating Direction Method. In M. T. Heath, editor, *Hypercube Multiprocessors*. SIAM, Philadelphia, 1987.

- [20] T. L. Tysinger and D. A. Caughey. Alternating direction implicit methods for the Navier Stokes equations. *AIAA J.*, 30:2158–2161, 1992.
- [21] H. A. van der Vorst. Large tridiagonal and block tridiagonal linear systems on vector and parallel computers. *Parallel Computing*, 5:45–54, 1987.
- [22] R. S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 1962.