

Parallel Multigrid with ADI-like Smoothers in Two Dimensions

Craig C. Douglas¹, Sachit Malhotra², and Martin H. Schultz³

¹ Department of Mathematics, University of Kentucky, 715 Patterson Office Tower, Lexington, KY 40506-0027, USA

² One Century Tower, 265 Church Street, New Haven, CT 06510-7010, USA

³ Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA

Abstract. Alternating direction iterative (ADI) methods do not usually work well on parallel computers due to having to do parallel rather than serial tridiagonal solves in all but one dimension. An ADI-like iteration is developed and analyzed which does not require parallel tridiagonal solves in any direction, has at least as good of a convergence rate as ADI, and has almost no communication when imbedded as a smoother inside of a multigrid solver. Numerical experiments on a network of workstations and a parallel computer are included.

1 Introduction

The alternating direction iterative (ADI) method is a smoother where information moves quickly in each of the dimensions of a boundary value problem. For problems on rectangular, regular meshes, the tridiagonal solves in each direction provide robustness in excess of line SOR when used as a smoother in a multigrid solver.

On parallel computers, tridiagonal solves across computer memory boundaries slow classical ADI down to the point where ADI is not acceptable as a smoother for parallel multigrid. Usually, a data transpose is performed in order to speed up the tridiagonal solves or a cyclic reduction-like step is used.

In this chapter, a transpose free ADI-like method is analyzed and tested. The tridiagonal matrices to be inverted for elliptic problems are strongly diagonally dominant. Gauss-Seidel converges to the solution of the original system quickly. Hence only a very small number of Gauss-Seidel sweeps are required to obtain an accurate approximation.

The method runs fast all geometric directions, not just one. When only a small number of iterations are used, as is typical in multigrid solvers, using a small overlap of subdomains allows only one communication of data between processors. This makes the ADI-like method a good candidate as a smoother in parallel multigrid solvers.

In §2, classical ADI is briefly reviewed. In §3, some classical Gauss-Seidel theory is reviewed. In §4, the new ADG method is defined and analyzed. In §4.1, convergence of ADG is proven in a multigrid iteration. Numerical experiments are provided on a network of workstations and a parallel computer.

2 Classical ADI

In this section we review the classical ADI algorithm which will motivate the main results of this chapter. 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

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

where $\gamma(x, y)$ is a prescribed function on the boundary Γ of R [10]. 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

$$Au = k, \quad \text{where } A = H + V + \Sigma. \quad (2)$$

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* [10]. In particular, they are reducible to the direct sum of irreducible Stieltjes matrices [10]. For the rest of the discussion we shall ignore the matrix Σ , which can be incorporated into the discussion by defining the modified matrices [4]

$$H' = H + \Sigma/2 \quad \text{and} \quad V' = V + \Sigma/2.$$

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 [10, 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 [10]. 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 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

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

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

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

Proof: Note that (8) is a restatement of a well known result [7] and that (10) is derived from repeated applications of the matrix in (8). \square

Corollary 2. *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 $\text{ADG}(\rho, k)$ 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 $\text{ADG}(\rho, k)$ 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 $\text{ADG}(\rho, k)$ 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 3. *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, $u_{j,m}^{\text{ADI}}$ be the iterate produced by a single iteration of ADI ((5) and (6)) and $u_{j,m}^{\text{ADG}}$ be the iterate produced by a single application of the $\text{ADG}(\rho, k)$ method. If*

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

$$\Delta_{j,m} \equiv u_{j,m}^{\text{ADG}} - u_{j,m}^{\text{ADI}}$$

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).

Lemma 3 can be proved by considering the effect of each half step of $\text{ADG}(\rho, k)$ on the eigenvectors of the matrix A (see [8]). Note that (13) measures the difference in the iterates produced by the $\text{ADG}(\rho, k)$ and ADI methods.

4.1 $\text{ADG}(\rho, k)$ 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 $\text{ADG}(\rho, k)$ as a preconditioner for general elliptic problems. Although the analysis in this section is for the model problem (2) numerical experience suggests that the $\text{ADG}(\rho, k)$ method works well in practice as a preconditioner for non constant coefficient problems.

Theorem 4. *For the model problem defined by (2), the two grid method [1], using s smoothing steps of $\text{ADG}(\rho, k)$ 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.

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

s	ADI ($\equiv \text{ADG}(\sqrt{8}, \infty)$)	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

The proof of Theorem 4 can be found in [8]. The theorem can be extended to more than 2 levels easily using the technique in [1].

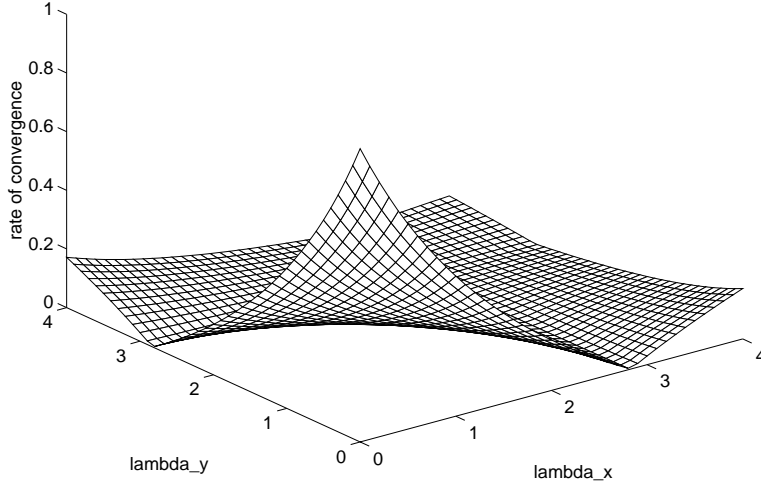


Fig. 1. ADI in two dimensions: effect on different $v_{j,m}$

For $\text{ADG}(\rho, k)$, using Lemma 3 and (11), we have

$$T_{\text{ADG}(\rho, k)} = (I - M_\rho^{lk})T_\rho - M_\rho^{lk}(H + \rho I)^{-1}(V - \rho I)$$

where M_ρ^{lk} 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 is for the two grid method for Poisson's equation, we have found that one Gauss-Seidel sweep (i.e. an application of $\text{ADG}(\rho, 1)$) 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 [9], 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 [3]. 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.

Poisson's Equation on a Square Domain The experiments in this section relate to the solution of (1) on the unit square. The problem was discretized using centered finite differences on a uniform grid. A nested iteration V cycle [1] was used for the solution of the problem. The code was based on [2] with the appropriate communication and computation routines added for the solution of the resulting tridiagonal systems. MPI [6] was used as the communication library. The codes were executed on an IBM SP2 at the IBM T.J. Watson Research Center at Yorktown Heights, NY. The same code was recompiled and executed on a network of SPARC10 (SunOS 5.3) workstations connected by an ethernet at the Computer Science Department at Yale University.

The right hand side was chosen such that the solution to the problem was given by

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

and the error was measured as the two norm of the difference of the computed solution and the exact solution.

Table 2 contains timings from the execution of the code on the IBM SP2 with thin nodes. Table 3 contains timing from the execution of the code on a network of SPARC10 workstations connected by the ethernet.

Table 2. Timing for the solution of Poisson's Equation using $ADG(\rho, 1)$, IBM SP2

N	procs	$ADG(\sqrt{8}, 1)$	ADI
1024	1	37.59	74.49
	2	17.63	33.92
	4	8.72	14.61
	6	5.50	9.43
	8	4.22	7.36
	10	3.41	5.92
512	1	7.58	11.56
	2	3.22	4.60
	4	2.11	3.07
	6	1.43	2.20
	8	1.21	1.89
	10	1.03	1.68
256	1	1.24	1.54
	2	0.90	0.11
	4	0.66	0.93
	6	0.55	0.82
	8	0.49	0.75
	10	0.51	0.79

N	p	$ADG(\sqrt{8}, k)$	ADI
2048	2	75.78	158.59
	4	38.89	76.65
	6	24.41	46.41
	8	17.98	34.50
	10	14.17	27.48
	12	11.94	22.31
	14	10.31	19.14
	16	9.26	16.13
	18	8.28	15.35
	20	7.43	13.76
	22	6.89	13.48
	24	6.52	12.30

Variable Coefficient Problems on a Square Grid The experiments in this section relate to the solution of a non self-adjoint, non separable equation (14) on the unit square.

$$\begin{aligned}
 & -(e^{-xy}u_x)_x - (e^{xy}u_y)_y + [(0.5 - x)u_x + (0.5 - y)u_y] + \\
 & u/(1 + x + y) = g(x, y)
 \end{aligned} \tag{14}$$

Table 3. Timing for the solution of Poisson's Equation using $ADG(\rho, 1)$, Sparcstation10, ethernet

N	procs	$ADG(\sqrt{8}, 1)$	ADI
1024	1	129.92	244.47
	2	67.89	144.96
	4	40.19	88.09
	8	31.76	52.09
512	1	23.55	37.21
	2	15.91	21.75
	4	10.30	15.17
	8	6.91	12.53

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

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

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 (14) 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 as a smoother is less than the use of, say, Gauss-Seidel as a smoother.

Table 4 contains timings from the execution of the code on the IBM SP2 with thin nodes.

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 the parallel overhead of the scheme can be further reduced by the aggregation of the Gauss-Seidel steps, the method is effective even on parallel machines with high latency.

Table 4. Timing for the solution of a general elliptic PDE using $ADG(\rho, 1)$, IBM SP2

N	procs	$ADG(\sqrt{8}, 1)$	ADI
1024	2	13.46	28.22
	4	6.67	11.43
	6	4.37	7.29
	8	3.39	6.12
	10	2.60	4.80
512	2	2.43	3.63
	4	1.64	3.63
	6	1.13	1.74
	8	0.95	1.49
	10	0.78	1.33

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 chapter 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. Bank, R. E., Douglas, C. C.: Sharp estimates for multigrid rates of convergence with general smoothing and acceleration. *SIAM J. Numer. Anal.*, **22** (1985) 617–633
2. Douglas, C. C.: MPI multigrid codes for 2D and 3D elliptic problems. *MGNet* (1995). Available electronically at <http://casper.cs.yale.edu/mgnet/www/Codes/douglas>
3. Douglas, C. C.: Private communication (August 1996)

4. Douglas, J. and Rachford, H. R.: On the numerical solution of heat conduction problems in two or three space variables. *Trans. Amer. Math. Soc.*, **82** (1956) 421–439
5. Duff, I. S., Grimes, R. G., Lewis, J. G.: User's guide for the Harwell–Boeing sparse matrix collection (Release I). Tech. Rep. TR/PA/92/86, CERFACS, Toulouse (1992)
6. Gropp, W., Lusk, E., Skjellum, A.: *Using MPI: Portable Parallel Programming with the Message Passing Interface*. The MIT Press (1994)
7. Hackbush, W.: On the multi-grid method applied to difference equations, *Computing*, **20** (1978) 291–306
8. Malhotra, S.: *Topics in Mutigrid Methods*. PhD thesis, Yale University, New Haven, CT, USA (1996). Available electronically through MGNet, <http://casper.cs.yale.edu/mgnet/www/mgnet-papers.html>
9. Saied, F.: *Numerical Techniques for the Solution of the Time-dependent Schrödinger Equation and their Parallel Implementation*. PhD thesis, Yale University, New Haven, CT, USA (1990). Available as YALEU/DCS/tr811, Department of Computer Science, Yale University
10. Varga, R. S.: *Matrix Iterative Analysis*. Prentice-Hall, Inc, Englewood Cliffs, NJ, USA (1962)