Block-ADI Preconditioners for Solving Sparse Nonsymmetric Linear Systems of Equations *

Sangback Ma and Youcef Saad †

Abstract. There is currently a regain of interest in the Alternating Direction Implicit (ADI) algorithms as preconditioners for iterative methods for solving large sparse linear systems, because of their suitability for parallel computing. However, the classical ADI iteration is not directly applicable to Finite Element (FE) matrices, tends to converge too slowly for 3-D problems, and the selection of adequate acceleration parameters, remains a difficult task. In this paper we propose a Block-ADI approach, which overcomes some of these problems. In particular we derive a simple inexpensive heuristic for selecting the acceleration parameters. The new approach can be viewed as a combination of the classical ADI method and a domain decomposition approach.

Key words: Preconditioning; ADI; Block ADI ; Preconditioned GMRES; Domain Decomposition.
AMS (MOS) Subject Classification: 65F10.

1 Introduction

Iterative solutions of large sparse linear systems by Krylov subspace methods, require the use of preconditioning techniques in order to converge in a reasonable number of iterations. When implemented on parallel computers, standard preconditioners based on incomplete factorization, such as ILU [14] or the more elaborate versions such as ILUT[17, 24], using a wavefront or level-scheduling approach realize a reasonable speed-up on vector computers or parallel computers with a small number of processors [1, 2, 5]. However, the lengths of the wavefronts are not uniform, and this contributes to poor load balancing, and puts a limit to the maximum achievable speed-up due to Amdahl's law. Alternatively, parallelism can also be obtained through *multicoloring*. For example, if the matrix has property A, as is the case for the standard 5-point matrices obtained from centered Finite Difference (FD) discretizations of elliptic Partial Differential Equations (PDE's), there is a partition of the grid-points in two disjoint subsets such that the unknowns of any one subset are only related to unknowns of the other subset. This enables to produce a reordered matrix having a block-tridiagonal matrix, where the diagonal blocks are diagonal matrices and there are several different ways of exploiting this structure. For example, the unknowns associated with one of the subsets can be easily eliminated, and the resulting reduced system is often well-conditioned. This 'two-coloring' often referred to as a red-black or checkerboard ordering, can be generalized to arbitrary sparse matrices by using multi-coloring and the generalizations of the standard ILU techniques based on such ideas can be easily derived [19]. Polynomial preconditioners are quite appealing because

*This research was supported by NIST grant 60NANB2D11272 and by The Minnesota Supercomputer Institute.
†University of Minnesota, Computer Science Department, 4-192 EE/CSci Building, 200 Union Street S.E., Minneapolis, MN 55455

of the requirement that only matrix – vector product operations need to be optimized, a task that is reasonably straightforward. They can also always be combined with other preconditioners to improve their overall performance and this may constitute their best possible use, at least in the non-Hermitian case.

Another standard approach that has been employed in the past is Domain Decomposition. Most domain decomposition preconditioners rely on an efficient solution technique for the Schur complement. If we wish to achieve high speed-ups we must increase the number of subdomains, and as a result the Schur complement will become bigger and more complex. This eventually becomes difficult to program and, in addition, will yield diminishing returns.

Going back to standard preconditioners, we recall that historically SSOR was used first as a preconditioner to the conjugate gradient methods [3, 4] well before incomplete factorization techniques became popular. Similarly, standard relaxation or block relaxation techniques have often provided easy-to-implement and yet reasonably efficient preconditioners. However, insufficient work has been done on the parallel implementations of these techniques. In [18], SOR and SSOR and multicoloring techniques were combined and compared with 'good' ILUT implementations. The main result in [18] is that these types of techniques can be quite efficient, and far superior to the standard preconditioners on some problems, provided a multi-step approach is used, namely provided that $k > 1$ steps of SOR or SSOR are taken at each preconditioning operation instead of just one as is usually done, where k is some parameter.

In this paper we investigate an ADI type iteration viewed from a *Domain Decomposition* angle. Our goal is to obtain a method whose computational structure is amenable to parallel computing as is the ADI iteration. In addition we would like the method to be applicable to Finite Element applications, possibly for 3-D problems, just as in a domain decomposition approach.

2 Classical ADI Methods

The ADI method was introduced by Peaceman and Rachford [15] in 1955, to solve the discretized boundary value problems for elliptic and parabolic PDEs. The finite difference discretization of the model elliptic problem

$$\begin{aligned} -\triangle u &= f, \ \ \Omega = [0,1] \times [0,1] \\ u &= 0 \ \ on \ \ \delta\Omega \end{aligned} \tag{1}$$

with 5-point centered finite difference discretization, with $n+2$ mesh-points in the x – direction and $m+2$ points in the y direction, leads to the solution of a linear system of equations of the form

$$Au = b \tag{2}$$

where A is a matrix of dimension $N = n \times m$. Without loss of generality and for the sake of simplicity, we will assume for the remainder of this paper that $m = n$, so that $N = n^2$.

Writing the discretization in x and y direction into matrices H and V respectively, leads to a linear system of equations

$$(H + V)u = b \tag{3}$$

where both H and V are sparse and possess a special structure. In particular, with suitable reordering, H and V are *tridiagonal*.

Starting with some initial guess u_0, the Alternative Direction Implicit procedure for solving (3) generates a sequence of approximations $u_i, i = 1, 2, \ldots$ given by the following algorithm

ALGORITHM **2.1. Peaceman Rachford(PR) ADI**

$$(H + \rho_i I)u_{i+1/2} = -(V - \rho_i I)u_i + b \tag{4}$$

$$(V + \rho_i I)u_{i+1} = -(H - \rho_i I)u_{i+1/2} + b, \tag{5}$$

where the ρ_i's are positive parameters. The ADI method, applied to positive definite systems, was extensively studied in the 1950s and 1960s, see, e.g, the books of Varga [21] and Wachspress [22]. In this case H and V have real eigenvalues and the following is a summary of the main ADI results in this situation.

1. Any stationary iteration($\rho_i = c > 0$, for all i) is convergent if H and V are symmetric positive definite.

2. For the model problem the asymptotic rate of convergence of ADI with optimal fixed ρ is equal to that of SSOR with optimal ω, so the spectral radius is $\omega_b - 1$. Since each ADI iteration takes more work than an SSOR iteration, a sequence of parameters ρ_i is often used in order to compete with SSOR ([21]).

3. The rate of convergence of ADI can be appreciably increased by the application of sequence of parameters, ρ_i, that are used in cyclic order. The theory of the convergence and the selection of good parameters when more than one one cycle is used, has not been fully developed. For those cases when $HV = VH$, a satisfactory theory does exist [6]. For the model problem with a single optimal ρ the time complexity is $O(n^3)$. In [15] a procedure is described to reduce the time complexity to $O(n^2 \log n)$ with a sequence of ρ parameters.

For 3-D problems Douglas[7] proposed a variant of the classical ADI, which has better convergence behavior than the classical ADI. Let $A = H + V + W$, where H, V , and W contains the x-, y-, z-, directional derivatives, respectively.

ALGORITHM **2.2. Douglas Rachford(DR) ADI**

$$(H + \rho_i I)u_{i+1/3} = -(H + 2V + 2W - \rho_i I)u_i + 2b \quad (6)$$

$$(V + \rho_i I)u_{i+2/3} = -(H + V + 2W - \rho_i I)u_i - Hu_{i+1/3} + 2b \quad (7)$$

$$(W + \rho_i I)u_{i+1} = -(H + V + W - \rho_i I)u_i - Hu_{i+1/3} - Vu_{i+2/3} + 2b \quad (8)$$

The convergence behaviors of this algorithm are different from those of the two-dimensional case. For example for fixed $r > 0, \rho_i = r, \forall i$, three dimensional mesh regions can be exhibited for which this variant of ADI fails. Also in practice the convergence is very slow, and as a result it has rarely been used. There are alternative formulations which sweep through planes as opposed to lines of the domain.

More recently, the focus has turned into the implementation and efficiency issues of ADI methods on parallel computers. The degree of parallelism of the algorithm is of the order of the grid points, but the best complexity that can be achieved for each step is $O(\log n)$ using cyclic reduction in both directions to solve the tridiagonal systems [12, 10]. Other strategies exist which combine divide-and-conquer tridiagonal solvers and cyclic reduction.

3 Block versions of ADI

Since H and V depend on the finite difference discretizations of original PDEs, the classical ADI is not defined for FE matrices. For example, the piecewise linear shape functions on triangles give rise to 7-point matrices, for which there is no natural splitting of A into the sum of two matrices H and V that are both tridiagonal, or defined discretizations of one-dimensional operators. The question then arises as to how to generalize the classical ADI for Finite Elements applications. There are several options available. In this paper we will simply use a technique which is based on recasting the Peaceman Rachford ADI in the framework of *Domain Decomposition* methods.

3.1 The classical ADI and Domain Decomposition

In Algorithm 2.1 H is the discretization matrix of x-directional derivatives. In terms of domain decomposition the domain is decomposed into *horizontal* lines. Then H is obtained by applying the original PDE on the subdomains, while imposing the Neumann boundary conditions on the *vertical* sides. Similarly for V. After H and V are found we could write A as

$$A = H + (A - H) = V + (A - V).$$

These two splittings of A are used in each of two stages of the iteration (4). A parameter ρ_i was added to the diagonals of H and V as an acceleration parameter. In other words ADI can be viewed as an extreme case of domain decomposition in the plane, where the subdomain consists of nonoverlapping horizontal rectangles consisting of one line each. We can also view ADI as a means of using a domain decomposition strategy to reduce two-dimensional domains into 1-dimensional subdomains. By alternating between the x and y directions we can achieve the overlapping between the domains that is desirable in domain decomposition. As we noted earlier in domain decomposition the convergence deteriorates if the number of subdomains increases and there is no overlap between the subdomains. By the *alternation* we hope to achieve the equivalent effect of overlapping subdomains.

3.2 A Block-ADI Algorithm

We have seen in the previous discussions that the two stages of the classical ADI are characterized by the way in which the matrix A is split in two additive components. It is natural to think of considering the subdomains of horizontal/vertical *stripes* consisting of a few, say k, lines, instead of just one line. The same procedure as in the classical ADI can then be defined. Let us call ADI(k) this variant of ADI, and let $H^{(k)}$ and $V^{(k)}$ denote the matrices obtained by applying the original PDE on this decomposition of the domains. In essence, for each of the two domain partitionings, these matrices are obtained from the original matrix by neglecting the interactions between grid points across interfaces, or rather replacing them with Neuman boundary conditions. Then A is split as

$$A = H^{(k)} + (A - H^{(k)}) = V^{(k)} + (A - V^{(k)})$$

from which we can define our block ADI procedure, denoted by ADI(k).

ALGORITHM **3.1. ADI(k)**

$$(H^{(k)} + \rho_i I)u_{i+1/2} = -(A - H^{(k)} - \rho_i I)u_i + b \quad (9)$$

$$(V^{(k)} + \rho_i I)u_{i+1} = -(A - V^{(k)} - \rho_i I)u_{i+1/2} + b \quad (10)$$

- Note that ADI(k) is defined for FE matrices as well as for FD matrices.
- In a parallel environment the simplest form of parallelism is obtained by processing the domains independently, which will naturally achieve a speed-up of n/k (in case $m = n$). Further speed-ups can also be achieved. Although the matrices $H^{(k)}$ and $V^{(k)}$ are not tridiagonal for $k > 1$, they are banded with a bandwidth of $2k+1$ and parallel banded solvers [11] can be used for each subdomain independently to yield higher speed-ups.
- If we assume an exact LU factorization is used to solve (9), we expect that the cost will be of the order k times that of the classical ADI. However, since $H^{(k)}$ and $V^{(k)}$ will be closer to A, or fewer interfaces are neglected in $H^{(k)}$ and $V^{(k)}$, the ADI(k) is likely to converge faster. As a result there is a tradeoff between the cost due to the number of ADI iterations and the cost of solving each of the equations (9) and (10). An optimal value of k should achieve a balance between these two costs.

- For 3-D problems we expect the 3-D equivalent of Block-ADI to perform better than DR ADI for large k. The main reason why the DR ADI converges so slowly is that H, V, and W are each too poor approximations to A. In terms of the splitting of A the spectral radius associated with the ADI iteration will be smaller as each of H, V or W gets closer to A. The matrices $H^{(k)}, V^{(k)}$, and $W^{(k)}$ are likely to be better approximations to A, for larger k. As in the two-D case the potential for achievable speed-up is reduced but for 3-D problems we still might have enough parallelism compared with two-D problems. However, this remains to be verified by numerical tests.

4 The acceleration parameters

As we noted earlier ADI can be efficient when we cycle with a decreasing sequence of parameters $\{\rho_i, i = 1, ..., l\}$. In fact for the model problem there is a complete theory on how to select the parameters optimally. This makes the algorithm converge within discretization errors, in a number of steps of the order of $O(n^{1/l})$ with a fixed l [15]. If $HV = VH$, and H and V are symmetric, positive definite, there exists satisfactory theory based on common eigenvectors [6]. If V and H commute but are not symmetric with possibly complex eigenvalues, G. Starke derives an algorithm to determine the optimal ρ values for $l = 1$ and $l = 2$, and N. Ellner and E. Wachspress propose an algorithm for the case when the imaginary parts of eigenvalues are small relative to the real part. In all of the previous cases, these algorithms or formulas require some a-priori knowledge about the spectra of H and V. However, in practice the condition that H and V commute is much too restrictive. Actually, it dictates that the underlying PDE be separable in which case faster techniques may exist. In this paper we are interested in the more general situation where the PDE is not separable. This implies that there no longer exists a common set of eigenvectors, and this makes the analysis difficult. If H and V are symmetric, then we still could derive an *upper bound* for the spectral radius [23].* If H and V are not symmetric, which is true in the presence of *convection* terms in the underlying PDE, then the above upper bounds no longer hold, since $\|A\|_2$ is not equal to the spectral radius of A.

As a result, for the general case, we must turn to heuristics to find optimal or nearly optimal iteration parameters ρ_i. For convenience we define ADI(k, l) to be the ADI(k) iteration using l parameters.

4.1 Multigrid Motivation

We now go back to the simple case where $l = 1, HV = VH$, and H and V are symmetric, positive definite. Also assume that we know a, b such that

$$a \leq \lambda(H), \lambda(V) \geq b$$

where $\lambda(A)$ denotes the spectrum of the matrix A. Then the optimal ρ is given by $\sqrt{ab}$ [23]. For model problem $a \approx h^2, b \approx 2$, hence $\rho \approx \sqrt{2}h$. In other words the optimal ρ is linearly proportional to h. In general this would be the case when the *diffusion* term dominates in the discretized matrix(If h becomes very small, the discretized matrix is more dominated by diffusion terms).

Based on this observation we first find the optimal value of ρ in a much coarser grid, then we use the above relationship to predict the optimal ρ for the current grid. The following algorithm attempts to find an optimal set of l values, for a combination ADI-GMRES, starting from a coarser grid with t initial values. Note that these parameters are likely to be optimal only for the combination ADI-GMRES and not for ADI considered as a separate algorithm. The difference between the two can be quite important.

ALGORITHM **4.1.** *MG(n, t, k, l)*

1. *Given $N = n^2$ large enough, choose a* coarser *Grid with $h_m = 4h$, and $N_m = N/16$.*

*Actually, for l greater than 1 we need a modified form of the classical ADI formulation.

2. *Choose $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_t \geq 0$, such that ρ_1 is large enough and ρ_t is small enough.(If the matrix is properly normalized, $\rho_1 \approx 0.1, \rho_t \approx 10, t \approx 6$ for model problem.)*

3. *Run $GMRES(m) - ADI(k)$ t times, each with ρ_i as its single parameter, until convergence.*

4. *Find ρ_j with the lowest iteration number.*

5. *Spread l values of ρ around $\frac{\rho_j}{4}$ (which is the estimated optimal ρ for the original grid)*

In step 1 we even could start from $h_m = 8h$, depending on the size of N. In step 3 rather than terminating when the residual norm is reduced by a given ϵ, we could alternately terminate with a fixed number of iterations and in step 4, find ρ_j with the smallest residual norm.

Regarding the cost of the procedure, it is clearly impossible to predict it exactly. An estimate may be obtained only by making the assumption that the number of iterations required to reduce the residual norm by a factor of ϵ is proportional to $\sqrt{N} = n$. Then the single run of GMRES(m)-ADI(k) in step 3 would be roughly $\frac{1}{16 \cdot 4}$ of the cost for the original size using the same ρ. Under this assumption, the cost of MG(n, t, k, l) would be $5/64$ of the cost for the full system, when $t = 5$. For $h_m = 8h$ the cost becomes basically negligible.

The main feature of the algorithm is that by utilizing a multigrid motivated heuristic we avoid the otherwise sensitive problem of finding the optimal ρ, while by choosing a sequence of ρ parameters we expect to exploit the full power of ADI. Note that the heuristic is based on the assumption on a simple growth rule for the optimal parameters when h varies. We do not know whether this rule is valid for the general problems which we are considering. The heuristics can be still be used in these cases and seems to perform quite well, according to our numerical tests.

5 Numerical Experiments

We consider three test problems based on elliptic PDE's on square grids. We discretized the problems using centered finite difference discretizations in all cases. The mesh sizes vary from test to test and are reported independently in this section.

- **Problem 1** Poisson Equation on a Square

$$\begin{aligned} -\triangle u &= f, \ \ \Omega = [0,1] \times [0,1] \\ u &= 0 \ \ on \ \ \delta\Omega \\ f &= x(1-x) + y(1-y) \end{aligned}$$

- **Problem 2** Elman's problem [9]

$$\begin{aligned} -(bu_x)_x - (cu_y)_y + (du)_x + du_x + (eu)_y + eu_y + fu &= g \qquad (11) \\ \Omega = [0,1] \times [0,1] & \\ u = 0 \ \ on \ \ \delta\Omega & \end{aligned}$$

where $b = \exp(-xy), \ c = \exp(xy), \ d = \beta(x+y),$
$e = \gamma(x+y), \ f = \frac{1}{(1+xy)},$
and g is such that exact solution $u = x\exp(xy)\sin(\pi x)\sin(\pi y)$

- **Problem 3** Discontinuous coefficients problem described by Eq. (11) where b and c are given by

$$\begin{aligned} b &= 100,\ 0 < x < 0.5 \\ &= 0.01,\ 0.5 < x < 1 \\ c &= 100,\ 0 < y < 0.5 \\ &= 0.01,\ 0.5 < y < 1 \end{aligned}$$

First we note that all of the test problems are in two-D geometries. By a suitable reordering $H^{(k)}$ and $V^{(k)}$ can be made into matrices of bandwidth $2k+1$. Thus, using the horizontal natural ordering for $V^{(k)}$ and the vertical natural ordering for $H^{(k)}$, as is illustrated in Figure 1 for the case $m = n = 4$, both matrices will be pentadiagonal when $k = 2$. A direct band solver is used for (9). Ignoring the initial factorization costs, the cost of solving (9) is roughly k times that of ADI(1), the classical ADI. In our experiments we used $k = 2, 4, 8$. For these values of k we expect the cost of solving (9) to remain reasonable. However, this may not be true for large problems from three-dimensional domains, since the bandwidth of (9) may become too large for banded solvers to be economical and we may have to consider *iterative* methods. As for the MG(n, t, k, l) we set t=5. We chose l to be 6, but depending on the problems the optimal l might vary.

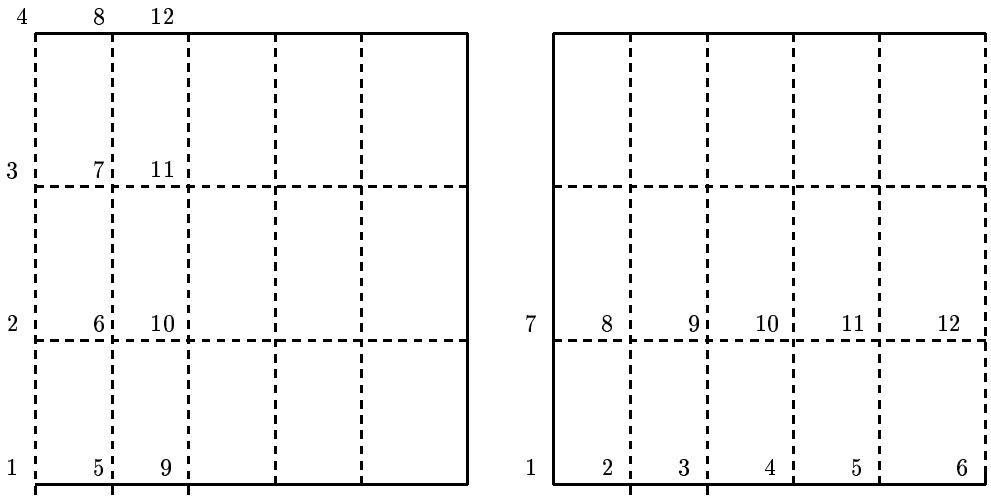

Figure 1: Illustration of $H^{(4)}$ in the vertical natural and horizontal natural orderings

The flexible GMRES routine allowing variable preconditioner at each iteration was used with m=10, $\epsilon = 10^{-6}$ for the *outer* iteration, where m is the dimension of Krylov subspace associated with the GMRES method. As for the finite element discretizations we used piecewise linear functions on triangles. In two-D problems they give rise to sparse matrices with 7 diagonals. Experiments were done on a Cray-2 and a Sun-4. The Timings reported were obtained on the Cray-2.

Table 1 contains the iteration number and CPU time on a Cray-2 of GMRES(10)-ADI(1,6) with a fixed parameter ρ taking various values versus those of GMRES(10)-ILU(0) and GMRES(10)-ILU(8). For the GMRES-ILU(0) and GMRES-ILUT(8) we used level-scheduling and jagged diagonal data structures for efficient vectorization on Cray-2[16]. For problem 3 with $\rho = 0.1$ and $\rho = 0.2$ the convergence was not reached within a reasonable amount of CPU time, so they are listed as 'N.A.'. For problem 2 with FDM, ADI(1,6) preconditioning with $\rho = 0.1, 0.2, 0.5$ takes less time than ILU(0) or ILUT(8) preconditioning. For problem 3 with FDM, ILU(0) or ILUT(8) preconditioning works better. Note that the MG(n, t, k, l) heuristic was not adopted. Also with GMRES-ILUT the parallelism on a one processor Cray-2 is limited to vectorization, the length of vector register, while for ADI(1,6) preconditioning the

mimimum parallelism is n=128.(If we use cyclic reduction the parallelism could much exceed it.) So on massively parallel machines ADI preconditioning has a bigger potential for parallelism.

Table 2-4 compares the iteration number with the ρ parameters obtained by the MG$(n,5,k,6)$ heuristic versus the iteration numbers with the optimal ρ, l=1, obtained empirically. It shows that MG$(n,5,k,6)$ is always superior to the optimal ρ, l=1.

Table 5-9 lists for various but fixed ρ the iteration number of GMRES-ADI(k,6) for k=1, 2, 4. It is intended to show the sensitivity to ρ and the effectiveness of increasing k. The sensitivity of iteration numbers to ρ can be easily seen. We also see that the iteration number is decreasing as k increases. But considering the cost of ADI(k,6) to be roughly k times that of ADI(1,6) we need a drop in iteration number in the same fraction, to be competitive in total costs. However, the table shows that is rarely the case. In a quite few cases we even notice stagnation. One possible explanation is that the iteration number for ADI(1,6) is already quite small.

FDM Problem 2 with $\gamma = 50$, β=1								
N = 128^2	ρ=0.1	ρ=0.2	ρ=0.5	ρ=1	ρ=2	ρ=3	ILU(0)	ILUT(8)
Iter #	15	12	13	36	78	108	93	14
CPU	3.29	2.61	2.73	7.55	15.97	22.83	4.28	4.26
FDM Prob 3 with $\gamma = 50, \beta = 1$								
Iter #	N.A.	N.A.	63	36	49	43	191	42
CPU	N.A.	N.A.	13.41	10.28	10.25	9.17	7.46	6.45

Table 1: Iteration and CPU time of GMRES(10)-ADI(1,6) with constant ρ vs. GMRES(10)-ILU on Cray-2. N is the dimension of the matrix.

Problem 1		
	Heuristic	Optimal ρ
N=64^2		
k	l=6	l = 1
1	7	8
2	6	7
4	4	5
N=128^2		
1	8	12
2	7	9
4	5	7

Table 2: ADI(k,6) preconditioning with MG$(n,5,k,6)$ heuristic vs ADI(k,6) preconditioning with a theoretically determined optimal ρ

6 Conclusion

We have proposed a block version of the ADI algorithm based on a domain decomposition viewpoint. In addition, we have derived a simple yet effective way of getting optimal acceleration parameters for the ADI-preconditioned GMRES iteration, based on a multigrid approach. The numerical experiments reported indicate that the approach holds some promise for the parallel solution of Elliptic PDE's on arbitrary domains. Two attractive features of the method are its ease of implementation, and its generality. Although a full comparison with the best domain decomposition approaches has yet to be made,

FDM Problem 2 $\gamma = 50, \beta = 1$		
	Heuristic	Optimal ρ
N=64^2		
k	l=6	l = 1
1	9	9
2	3	4
4	3	3
N=128^2		
1	9	12
2	5	6
4	3	5

Table 3: ADI(k,6) preconditioning with MG(n,5,k,6) heuristic vs ADI(k,6) preconditioning with a theoretically determined optimal ρ

FDM Problem 3 $\gamma = 50, \beta = 1$		
N=64^2		
k	Heuristic l=6	Optimal ρ l = 1
1	18	23
2	12	16
4	10	11
N=128^2		
1	32	35
2	17	22
4	13	15

Table 4: ADI(k,6) preconditioning with MG(n,5,k,6) heuristic vs ADI(k,6) preconditioning with a theoretically determined optimal ρ

the method itself can be viewed as a domain decomposition approach and for this reason its performance may be comparable. However, a distinct feature from a traditional domain decomposition approach is the use of acceleration parameters.

Acknowledgements. The authors would like to acknowledge the support of the Minnesota Supercomputer Institute which provided the computer facilities and an excellent research environment to conduct this research.

References

[1] E. C. Anderson, " Parallel implementation of preconditioned conjugate gradient methods for solving sparse systems of linear equations", Technical Report 805, CSRD, University of Illinois, Urbana, IL, 1988. MS Thesis.

Problem 1 with N=128^2										
	ρ									
k	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
1	13	14	12	14	20	24	29	29	32	35
2	10	9	9	9	10	12	12	14	18	19
4	7	7	8	9	10	11	12	13	17	19

Table 5: Iteration of GMRES(10)-ADI(k,6) for various values of ρ and k

FDM Problem 2 with $N = 64^2, \gamma = 50$, β=1										
	ρ									
k	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
1	25	15	14	13	12	11	10	10	9	9
2	8	5	4	4	4	4	4	5	5	5
4	5	3	3	3	4	4	4	5	5	5

Table 6: Iteration of GMRES(10)-ADI(k,6) for various values of ρ and k

[2] E. C. Anderson and Y. Saad, " Solving sparse triangular systems on parallel computers", *International Journal of High Speed Computing*, 1:73–96, 1989.

[3] O. Axelsson, " Conjugate Gradient Type-methods for Unsymmetric and Inconsistent Systems of Linear Equations", Technical Report 74-10, CERN, Geneva, 1974.

[4] O. Axelsson, " A Survey of Preconditioned Iterative Methods for Linear Systems of Algebraic Equations", *BIT*, 25:166–187, 1985.

[5] D. Baxter, J. Saltz, M. H. Schultz, S. C. Eisenstat, and K. Crowley, " An experimental study of methods for parallel preconditioned Krylov methods", Technical Report 629, Computer Science, Yale University, New Haven, CT, 1988.

[6] G. Birkhoff, R. Varga, S. R., and D. Young, "Alternating Direction Implicit Methods", in Advances in Computers, pp.189-273, Academic Press, New York, 1962

[7] J. Douglas, "Alternating Direction Methods for Three Space Variables", *Numerische Mathematik* Vol. 4, 1962, pp. 41-63.

[8] N. Ellner and E. Wachspress, "Alternating Direction Implicit Iteration for systems with Complex Spectra", *SIAM J. Numerical Analysis*, Vol. 28, No. 3, pp. 859-870, 1991

[9] H. Elman, "Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations", Ph. D Thesis, Yale University, 1982

[10] D. Gannon and J. van Rosendale, " On the impact of communication complexity in the design of parallel algorithms", *IEEE Trans. Comp.*, C-33(12):1180–1194, 1984.

[11] S. L. Johnsson, " Solving narrow banded systems on ensemble architectures", *ACM, TOMS*, 11(3), 1985.

[12] S. L. Johnsson, Y. Saad, and M. H. Schultz. " The alternating direction algorithm on multiprocessors", *SIAM J. Sci. Statist. Comp*, 8:686–700, 1987.

[13] W. J. Layton and P. J. Rabier. "Peaceman Rachford procedure and domain decomposition for finite element problems", Technical report, University of Pittsburgh, Pittsburgh, PA, (1991).

[14] J. A. Meijerink and H. A. van der Vorst, " An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix", *Math. Comp.*, 31(137):148–162, 1977.

[15] D. Peaceman and H. Rachford, "The Numerical Solution of Elliptic and Parabolic Differential Equations", *Journal of SIAM.*, Vol. 3, pp. 28-41, 1955

[16] Y. Saad, "Krylov Subspace Methods on Supercomputers", *SIAM J. Sci. Stat*, Vol. 10, No. 6, pp. 1200-12332, Nov, 1989

FDM Problem 2 with $N = 128^2, \gamma = 50$, β=1										
	ρ									
k	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
1	19	15	13	12	12	12	12	12	12	13
2	8	6	6	7	8	9	10	11	14	17
4	5	5	6	7	8	9	10	11	13	16

Table 7: Iteration of GMRES(10)-ADI(k,6) for various values of ρ and k

FDM Problem 3 with $N = 64^2, \gamma = 50$, β=1									
	ρ								
k	2	3	4	5	6	7	8	9	10
1	25	26	25	25	24	23	23	23	25
2	18	18	17	16	16	17	19	20	22
4	11	11	12	14	16	18	19	20	22

Table 8: Iteration of GMRES(10)-ADI(k,6) for various values of ρ and k

[17] Y. Saad, "ILUT: A Dual Threshold Incomplete LU Factorization", UMSI 92/38, 1992

[18] Y. Saad, "Highly Parallel Preconditioners for General Sparse Matrices", Technical Report –, University of Minnesota, Army High Performance Computing Research Center, Minneapolis, Minnesota, 1992.

[19] Y. Saad, " ILUM: A Parallel Multi-elimination ILU Preconditioner for General Sparse Matrices", Technical Report –, University of Minnesota, Army High Performance Computing Research Center, Minneapolis, Minnesota, 1992. In preparation.

[20] G. Starke, "Optimal Alternating Direction Implicit Parameters for Nonsymmetric Systems of Linear Equations", *SIAM J. Numerical Analysis*, Vol. 28, No. 5, pp. 1431-1445, 1991

[21] R. Varga, Matrix Iterative Analysis, Prentice-Hall, New York, 1962

[22] E. Wachspress, Iterative Solution of Elliptic Systems, Prentice-Hall, New York, 1966

[23] D. Young, Iterative Solution of Large Linear Systems, Academic Press, New York, 1971.

[24] Z. Zlatev, " Use of iterative refinement in the solution of sparse linear systems", *SIAM J. Numer. Anal.*, 19:381–399, 1982.

FDM Problem 3 with $N = 128^2, \gamma = 50$, β=1									
	ρ								
k	1	1.5	2	2.5	3	3.5	4	4.5	5
1	49	48	48	45	43	38	35	35	36
2	26	24	23	22	23	24	26	27	29
4	16	15	16	18	19	22	27	30	33

Table 9: Iteration of GMRES(10)-ADI(k,6) for various values of ρ and k