Electronic Transactions on Numerical Analysis.
Volume 8, pp. 88-114, 1999.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

NUMERICAL EXPERIMENTS WITH PARALLEL ORDERINGS FOR ILU PRECONDITIONERS*

MICHELE BENZI†, WAYNE JOUBERT†, AND GABRIEL MATEESCU‡

Abstract. Incomplete factorization preconditioners such as ILU, ILUT and MILU are well-known robust general-purpose techniques for solving linear systems on serial computers. However, they are difficult to parallelize efficiently. Various techniques have been used to parallelize these preconditioners, such as multicolor orderings and subdomain preconditioning. These techniques may degrade the performance and robustness of ILU preconditionings. The purpose of this paper is to perform numerical experiments to compare these techniques in order to assess what are the most effective ways to use ILU preconditioning for practical problems on serial and parallel computers.

Key words. Krylov subspace methods, preconditioning, incomplete factorizations, sparse matrix orderings, additive Schwarz methods, parallel computing.

AMS subject classifications. 65F10, 65F15.

1. Introduction.

1.1. Motivation and focus.

Krylov subspace methods [21] are customarily employed for solving linear systems arising from modeling large-scale scientific problems. The convergence of these iterative methods can be improved by *preconditioning* the linear system $Ax = b$. The preconditioned system $M^{-1}Ax = M^{-1}b$ (for left preconditioning) can be solved faster than the original system if the *preconditioner* M is an efficient and good approximation of A; efficient, in the sense that the cost of solving $Mu = v$ is much smaller than the cost of solving $Au = v$, and good in the sense that the convergence rate for the preconditioned iteration is significantly faster than for the unpreconditioned one.

Incomplete factorization techniques [24],[17],[20] provide a good preconditioning strategy for solving linear systems with Krylov subspace methods. Usually, however, simply applying this strategy to the full naturally ordered linear system leads to a method with little parallelism. Incomplete factorization is also useful as an approximate subdomain solver in domain decomposition-based preconditioners, such as Additive Schwarz Method (ASM) preconditioning [23].

In this paper we study the effect of the following algorithm parameters on the convergence of preconditioned Krylov subspace methods:

- Symmetric reorderings of the matrix: this applies to incomplete factorization (ILU) preconditioners;
- Subdomain overlap: this applies to Additive Schwarz preconditioners.

Symmetric permutations of the linear system have been first used in direct factorization solution methods for reducing the operation count and memory requirements. For example, the Minimum Degree ordering is effective for direct solvers in that it tends to reduce the number of nonzeros of the L and U factors [12]. Other reorderings can have a beneficial effect on incomplete factorizations employed as preconditioners, e.g., by providing a parallel preconditioner [21]; the storage for the preconditioner is typically controlled by the incomplete

*Received October 26, 1998. Accepted February 26, 1999. Communicated by D. Szyld. This work was supported in part by the Department of Energy through grant W-7405-ENG-36 with Los Alamos National Laboratory. This research was performed in part using the resources located at the Advanced Computing Laboratory of Los Alamos National Laboratory.

†Scientific Computing Group, MS B256, Los Alamos National Laboratory, Los Alamos, NM 87545, USA. E-mail {benzi,wdj}@lanl.gov.

‡VTLS, Inc., 1701 Kraft Dr., Blacksburg, VA 24060, USA. E-mail mateescug@vtls.com.

factorization scheme. However, parallel orderings may degrade the convergence rate, and allowing fill may diminish the parallelism of the solver.

In this paper, we consider structurally symmetric matrices arising from partial differential equations (PDEs) discretized on structured grids using finite differences. We focus on symmetric permutations as they represent similarity transformations and preserve the spectrum of the linear system matrix. Furthermore, symmetric permutations preserve the structural symmetry of a matrix and the set of diagonal elements.

Additive Schwarz methods [23] derive a preconditioner by decomposing the problem domain into a number of overlapping subdomains, (approximately) solving each subdomain, and summing the contributions of the subdomain solves. Variants of ASM are obtained by varying the amount of overlap and the subdomain solvers. When the subdomains are solved approximately using an incomplete factorization, the resulting preconditioner can be thought of as a "parallel ILU" strategy. Like block Jacobi, ASM has good parallelism and locality, but these advantages could be offset by a high iteration count of the underlying Krylov subspace method.

1.2. Related work. A lexicographic ordering of the grid points in a regular two- or three-dimensional grid is obtained by scanning the grid nodes and assigning numbers to the nodes in the order in which they are seen in the scanning. A widely used ordering is the *Natural Order* (NO), which is the order induced by labeling the grid nodes from the bottom up, one horizontal line at a time and (for three-dimensional grids) scanning consecutive vertical planes. Several reorderings have been considered in the literature as alternatives to the natural order. Among these are *Minimum Degree* (MD), *Multiple Minimum Degree* (MMD), *Reverse Cuthill-McKee* (RCM), *Nested Dissection* (ND), and *Multicoloring* (MCL). For a description of these reorderings see [6],[11],[12],[16],[21]. MMD, MD, and ND reorderings attempt to minimize the fill in the factors, while RCM reduces the bandwidth of the matrix.

The *degree of parallelism* (DOP) of a preconditioning algorithm is the number of processors that can work simultaneously on constructing or applying the preconditioner. MCL provides large grain parallelism for no-fill incomplete LU factorization [21]. With multicoloring, the linear system is partitioned in subsets of equations that do not have any internal dependencies between unknowns. For direct solvers, fill-reducing (ND, MD, MMD) and bandwidth-reducing (RCM) reorderings are superior to NO [12]. The usefulness of these reorderings for incomplete factorization preconditioners is not well established.

A simple incomplete factorization of a matrix A is $A = \bar{L}\bar{U} + R$, where the triangular matrices $\bar{L}$ and $\bar{U}$ have the same nonzero structure as the lower and upper triangular parts of A, respectively, and R is the *residual* matrix. This strategy is known as no-fill ILU, and is denoted by ILU(0). For symmetric positive definite (SPD) problems, $\bar{U} = \bar{L}^T$ and the factorization is called no-fill *incomplete Cholesky*, denoted by IC(0). One can attempt to improve the effectiveness of an incomplete LU factorization by allowing fill-in in the triangular factors $\bar{L}$, $\bar{U}$. The ILU(1) method allows nonzeros entries for the elements with level of fill at most one (see [21], pp. 278–281); the corresponding factorization for SPD problems is denoted by IC(1). A more sophisticated preconditioner is a dual-dropping ILU preconditioner [20], denoted by ILUT(τ, p), where τ is the dropping threshold and p is the maximum number of nonzeros of fill allowed in a row above those present in the original matrix.

The effect of reorderings on the performance of ILU-type preconditioners has been studied by Duff and Meurant [7], Benzi *et al.* [2], and Saad [19], among others. Duff and Meurant have studied the impact of reorderings on incomplete factorization preconditioning for SPD problems. The sources of inaccuracy in incomplete factorizations and the effect of reorderings on accuracy and stability have been analyzed by Chow and Saad [4] and Benzi *et al.* [2].

Let $\hat{A} = \bar{L}\bar{U}$. The residual matrix $R = A - \hat{A}$ measures the accuracy of the incomplete factorization. Let $\|\cdot\|_F$ denote the Frobenius norm of a matrix. Chow and Saad [4] and Benzi *et al.* [2] have shown that for nonsymmetric problems $\|R\|_F$ may be an insufficient characterization of the convergence of the preconditioned iterative methods. This is in contrast with symmetric positive definite problems where R provides a good measure of convergence. These authors have shown that the stability of the triangular solves is gauged by the Frobenius norm of the *deviation from identity* matrix, $E = I - A\hat{A}^{-1}$. For ILU(0), $\|R\|_F$ can be small, while $\|E\|_F$ may be very large, i.e., $\hat{A}$ can be very ill-conditioned. Note that $E = R\hat{A}^{-1}$.

Benzi *et al.* [2] have shown that RCM and MD reorderings can be beneficial for problems that are highly nonsymmetric and far from diagonally dominant. Specifically, MD has the effect of stabilizing the ILU(0) triangular factors, while RCM improves the accuracy of incomplete factorizations with fill.

Saad [19] has shown that improving the accuracy of the preconditioner, by replacing ILU(0) with dual dropping ILU preconditioning ILUT(τ, p), greatly improves the performance of red-black (RB) ordering, so that by increasing p the ILUT preconditioner induced by RB will eventually outperform the one induced by NO, as measured by iterations or floating point operations. RB ordering is the simplest variant of MCL, in which the grid points are partitioned in two independent subsets (see Subsection 2.1 and [21], page 366).

1.3. Contributions of the paper. Our main contribution is to show that parallel orderings can perform well even in a sequential environment, producing solvers requiring a lower wall-clock time (and in some cases reduced storage needs) than NO, especially for ILUT preconditioning of two-dimensional problems. We also show that for problems which are highly nonsymmetric and far from diagonally dominant these orderings are still better than NO, but they are outperformed by RCM. For such problems we also observe that parallel orderings can have a stabilizing effect on the ILU(0) preconditioner.

We propose and investigate a new MCL ordering, which allows for parallel ILU(1) and IC(1) preconditioners. We perform numerical experiments with multicoloring for nonsymmetric problems and incomplete factorizations with fill-in, extending the work done by Poole and Ortega [18] and by Jones and Plassmann [15] who studied the effect of MCL reorderings on IC(0) preconditioners for SPD problems. Our experiments suggest that for nonsymmetric problems, the loss in convergence rate caused by switching from NO to multicolorings for ILU(0) and ILU(1) preconditioners is compensated by the large DOP of the multicolorings.

We extend the study [7] in two ways: first, we show that RB applied to symmetric ILUT can outperform NO; second, we look at RB applied to nonsymmetric problems. We further the study of Benzi *et al.* [2] on orderings for nonsymmetric problems by considering MCL and ND orderings.

We further the work of Saad by considering the performance of RB on larger problems (Saad considers problems with up to $25^3 = 15,625$ unknowns, while we consider up to $160,000$ unknowns) and comparing RB with RCM and ND, in addition to NO. Finally, we assess the scalability of one-level overlapping ASM preconditioning for the set of model problems defined below.

1.4. Model problems. Although in practical experience it is often desirable to solve problems with complex physics on possibly unstructured grids, a minimal requirement of an effective parallel scheme is that it work well on simple, structured problems. For this reason, we will focus on a small set of model problems on structured grids which are "tunable" in terms of problem size and difficulty. The numerical experiments throughout the paper use the following three model PDE problems, all of them with homogeneous Dirichlet boundary conditions. Below we denote by Δ the Laplace operator in two or three dimensions.

Problem 1. The Poisson's equation:

$$-\Delta u = g. \tag{1.1}$$

Problem 2. Convection-diffusion equation with convection in the xy-plane:

$$-\varepsilon\Delta u + \frac{\partial}{\partial x}e^{xy}u + \frac{\partial}{\partial y}e^{-xy}u = g. \tag{1.2}$$

Problem 3. Convection-diffusion equation with convection in the z-direction:

$$-\varepsilon\Delta u + \frac{\partial}{\partial z}u = g. \tag{1.3}$$

The domain of the unknown function u is either the unit square or the unit cube, respectively for two-dimensional (2D) and three-dimensional (3D) problems. The discretization scheme employs a regular $n_x \times n_y \times n_z$ grid ($n_z = 1$ for 2D), five- and seven-point stencils for 2D and 3D, respectively, and centered finite differences. Let N be the number of unknowns of the linear system, $N = n_x\, n_y\, n_z$. The right-hand side of the linear system $Ax = b$ arising from discretization is artificial, i.e., b is obtained by computing $b = A\hat{e}$, where $\hat{e} = (1, 1, \ldots, 1)^T$.

For Problems 2 and 3, we vary the weight of the convection term by changing the parameter ε. The set of values we choose for ε are $\{1/100, 1/500, 1/1000\}$, where a smaller value of ε gives a less diagonally dominant coefficient matrix A for a fixed problem size [2]. More precisely, it can be easily verified that Problem 2 gives rise to a diagonally dominant M-matrix if and only if $\varepsilon/h > e/2$, where h denotes the mesh size. The corresponding condition for Problem 3 is $\varepsilon/h > 1/2$. Note that for a fixed h, the value of ε controls also the deviation from symmetry of the coefficient matrix, in the sense that the Frobenius norm of the symmetric part of A decreases as ε decreases, while the norm of the skew-symmetric part remains constant. Thus, for a fixed h, the discrete problem becomes more difficult as ε is made smaller; see also [2]. As is well-known, the discretization becomes unstable and nonphysical solutions may result when the conditions on ε/h are violated, particularly when boundary layers are present in the solution. In this case, the computed solutions are still useful in that they allow to determine the existence and location of such boundary layers; see, e.g., [9]. In the remainder of the paper, we will not address the issue of whether the discrete solution is a good approximation to the continuous solution.

We have employed three Krylov subspace accelerators as follows: Conjugate Gradient (CG) [14] for Problem 1, Bi-CGSTAB [25] for Problem 2, and GMRES(30) [22] for Problem 3. For a description of all these Krylov subspace methods, see [21]. The stopping criterion is $\|r_k\| \leq \varepsilon_R\|r_0\| + \varepsilon_A$, where $\|\cdot\|$ is the Euclidean norm, $r_k = b - Ax_k$, and $\varepsilon_R = \varepsilon_A = 10^{-5}$. The overall conclusions of this study are not dependent on the particular choice of the right-hand side and of the stopping criterion.

1.5. Computing environment. We have used code from the *PETSc* [1] library and our own code; the experiments have been performed on an SGI Origin 2000 machine with 32 nodes.

The experiments in Section 4 are based on the PETSc toolkit which provides message passing-based interprocess communication, with Single-Program Multiple-Data (SPMD) as the parallel programming model. PETSc is based on the industry-standard MPI message passing application programming interface. The toolkit has been installed on the Origin 2000 machine we have used in the debugging mode (-g compiler option). The timing data are obtained using the PETSc function *PetscGetTime()* which gives the wall-clock time. The toolkit uses left-preconditioning and the residual used in the stopping test corresponds to the preconditioned problem.

The code for the experiments in Sections 3 and 5 has been compiled using the compiler options `-pfa` (automatic parallelization of do-loops) `-n32` (new 32-bit objects) `-O0` (turn off optimization). The timing data are obtained using the Fortran intrinsic function *dtime()* from which the user CPU time is extracted. The code uses right preconditioning. The runs in Section 3 use one processor, while those in Section 5 use eight processors.

2. Background.

2.1. Multicoloring orderings. Given a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, the *graph coloring problem* is to construct a partition $(C_1, C_2, \ldots, C_c)$ of the set V such that all vertices in the same part C_i form an independent set, i.e., vertices in the same subset are not connected by any edge. The minimum number of colors necessary for a graph, $\chi = c_{min}$, is the *chromatic number* of the graph. The relevance of the chromatic number from a computational point of view is that all unknowns in the same subset can be solved in parallel. Thus, the number of inherent sequential steps is greater or equal to the chromatic number of the graph.

With each matrix $A \in R^{N\times N}$ we can associate a graph G_A such that $V = \{1, 2, \ldots, N\}$ and $(i, j) \in E \leftrightarrow a_{ij} \neq 0$. For arbitrary A, finding the chromatic number of G_A is NP-hard, but in practice a suboptimal coloring suffices. For the 5-point stencil discretization, it is easy to find a 2-coloring of the graph, commonly called *red-black* coloring. For red-black coloring, the degree of parallelism in applying an IC(0) or ILU(0) preconditioner is $N/2$ (N is the number of unknowns), which justifies considering this ordering strategy. The problem with this approach is that a no-fill preconditioner obtained with RB may result in poor convergence.

For SPD matrices arising from finite difference discretizations of two-dimensional elliptic problems, Duff and Meurant [7] have shown, by way of experiments, that RB reordering has a negative effect on the convergence of conjugate gradient preconditioned with IC(0). Poole and Ortega [18] have observed similar convergence behavior for multicolorings.

2.2. Scalability. Let $T_p(n, K)$ be the time complexity of executing an iterative linear solver on a parallel computer with p processors, where K is a parameter equal to the number of subproblems (for example, the number of subdomains in the case of ASM) into which the solver divides the original problem of size N, and n is the subproblem size, $N = n\,K$.

Following Gustafson [13], an algorithm is *scalable* if the time complexity stays constant when the subproblem size is kept constant while the number of processors and the number of subproblems both increase p times, i.e., $T_p(n, p) = O(T_1(n, 1))$. The scalability of an iterative linear solver can be decomposed into two types of scalability (see, for example, Cai *et al.* [3]): (i) *algorithmic* or *numerical scalability*, i.e., the number of iterations is independent of the problem size; (ii) *parallel scalability*, i.e., the parallel efficiency $T_1(n,p)/p\,T_p(n,p)$ remains constant when p grows.

In the case of linear solvers for problems arising from elliptic PDEs, it is likely that the two conditions for scalability cannot be simultaneously achieved; see, for example, Worley [26]. Hence, in practice a method is considered scalable if the number of iterations depends only weakly on the number of subproblems and the parallel efficiency decreases only slightly as p increases.

3. Performance of serial implementations. In this section we are concerned with the serial performance of reorderings for ILU preconditioners. We look at the effect of RB, RCM, and ND on several ILU preconditioners.

If fill is allowed, the advantage of RB providing a parallel preconditioner is greatly diminished; however, by allowing fill the accuracy of the RB-induced preconditioner may improve so much that it outperforms NO, thereby making RB an attractive option for uniprocessor computers. Indeed, Saad has observed [19] that, for a given problem and dropping tolerance τ, there is an amount of fill p_0, such that if $p > p_0$ then the preconditioner ILUT(τ, p) induced by RB outperforms the corresponding preconditioner induced by NO. On the other hand, it has been observed by Benzi *et al.* [2] that, for highly nonsymmetric problems, RCM outperforms NO when fill in the preconditioner is allowed. We further these studies by performing numerical experiments on NO, RB, *and* RCM, thereby comparing RB to RCM, and considering larger problems (more than 100,000 unknowns).

For the drop tolerance-based solvers we use global scaling, i.e., we divide all the matrix elements by the element of largest magnitude in the matrix. Thus, all entries in the scaled matrix are in the interval $[-1, 1]$. This scaling has no effect on the spectral properties of A, but it helps in the choice of the (absolute) drop tolerance, which is a number $0 < Tol < 1$. As an alternative, we tried the relative drop tolerance approach suggested in [20], where fill-ins at step i are dropped whenever they are smaller in absolute value than Tol times the 2-norm of the ith row of A. For the problems considered in this paper, the two drop strategies gave similar results. For 2D problems, we let $n_x = n_y$, and for 3D problems, we let $n_x = n_y = n_z$.

The number of iterations (I), the solver times (T) in seconds, and the memory requirements of the preconditioners are shown in Tables 3.1–3.6. Here $Fill$ denotes the amount of fill-in (in addition to the nonzero entries in the matrix) in the fill-controlled preconditioners. We denote as SILUT a modification of the ILUT algorithm which gives rise to a symmetric preconditioner whenever the original matrix is symmetric. Symmetry is exploited by storing only the upper triangular part of A. Throughout the paper we use as unit for storage measurement $1k = 1024$. The solver time reported is the sum of the time to construct the preconditioner and the time taken by the iterations of the preconditioned Krylov subspace method. The bold fonts indicate the smallest iteration count, time, and fill for each preconditioner.

3.1. Symmetric positive definite problems. We consider symmetric ILUT preconditioning and incomplete Cholesky preconditioners with no fill and with level of fill one, denoted respectively by IC(0) and IC(1). We examine the effect of RB and ND orderings on the convergence of preconditioned CG. As already mentioned, allowing fill in the preconditioner largely limits the parallelism provided by RB ordering. Except for the IC(0) preconditioner, for which the degree of parallelism of applying the preconditioner is $N/2$, the other preconditioners have modest parallelism. The results are reported in Tables 3.1–3.2.

We may draw several conclusions from these results. First, nested dissection ordering is not competitive with other orderings for IC(0), IC(1) and SILUT, thus we will exclude this ordering from the subsequent discussion. Second, our numerical results for IC(0) indicate the same performance for NO and RCM; this is in accordance with theoretical results [27] which show that the ILU(0) preconditioner induced by RCM is just a permutation of that induced by NO. Third, while NO and RCM are the best orderings for IC(0), the best ordering for IC(1) is RB, followed by RCM and NO. This is true for both 2D and 3D problems.

The best reordering for SILUT from an iteration count standpoint is RCM, followed by RB and NO. In the 2D case, notice that RB, even though requiring slightly more iterations than RCM, leads to a preconditioner that has a fill of roughly 2/3 of that of the RCM preconditioner. This has an effect on the solver time: for small problem sizes ($N = 32^2, 64^2$, not shown) the best reordering is RCM, but for the larger problems RB is the best reordering, in terms of CPU time. Here, the higher iteration count of RB as compared to RCM is out-

TABLE 3.1
Iterations (I), time (T), and fill versus problem size (N), for Problem 1, 2D domain, with different reorderings and preconditioners

$N{=}n_x^2$	Ord	Preconditioner										
		IC(0)		IC(1)			SILUT(.01,5)			SILUT(.001,10)		
		I	T (sec)	I	T (sec)	Fill (k)	I	T (sec)	Fill (k)	I	T (sec)	Fill (k)
128^2	NO	**57**	**3.70**	39	3.30	**15.8**	32	2.74	**31.4**	14	2.65	154
	RB	94	5.95	**31**	**2.68**	31.5	25	**2.31**	39.3	15	**1.78**	**78.2**
	RCM	**57**	3.79	39	3.32	**15.8**	**24**	2.33	46.3	**12**	2.03	121
	ND	110	6.89	59	4.91	28.5	45	3.87	40.3	25	2.87	79.6
256^2	NO	**109**	27.9	67	21.5	**63.5**	54	18.1	**126**	24	16.3	628
	RB	183	45.4	**52**	**17.1**	127	41	**14.2**	158	27	**11.4**	**316**
	RCM	**109**	**27.8**	67	21.5	**63.5**	**39**	14.8	188	**21**	12.6	499
	ND	175	44.6	108	35.4	114	86	29.7	163	41	18.6	323
300^2	NO	**126**	**43.8**	78	33.8	**87.3**	63	28.7	**174**	26	24.0	865
	RB	214	72.6	**60**	**26.8**	174	47	**22.5**	217	30	**17.4**	**435**
	RCM	**126**	44.1	78	33.5	**87.3**	**45**	23.3	259	**22**	17.9	688
	ND	199	69.5	116	51.5	150	98	47.2	244	47	29.4	467
400^2	NO	**166**	105	102	78.2	**155**	82	66.2	**310**	32	50.1	1544
	RB	278	179	**78**	**62.3**	310	61	**51.1**	388	36	**36.9**	**775**
	RCM	**166**	**104**	102	77.6	**155**	**59**	53.1	463	**28**	39.0	1230
	ND	259	163	149	118	268	112	96.8	432	64	69.0	826

TABLE 3.2
Iterations (I), time (T), and fill versus problem size (N), for Problem 1, 3D domain, with different reorderings and preconditioners

$N = n_x^3$	Ord	Preconditioner										
		IC(0)		IC(1)			SILUT(.01,5)			SILUT(.001,10)		
		I	T (sec)	I	T (sec)	Fill (k)	I	T (sec)	Fill (k)	I	T (sec)	Fill (k)
16^3	NO	**14**	**0.35**	10	**0.44**	**10.5**	10	0.39	13.8	8	0.55	38.1
	RB	16	0.40	**9**	0.49	15.8	15	0.43	**9.8**	8	**0.39**	**19.5**
	RCM	**14**	**0.35**	10	0.46	**10.5**	**9**	**0.36**	13.8	**6**	0.47	38.7
	ND	21	0.48	13	0.56	14.2	14	0.46	12.1	11	0.52	25.9
32^3	NO	**23**	**4.03**	17	4.96	**90.1**	16	4.31	119	14	6.28	313
	RB	31	6.16	**15**	**4.88**	135	27	6.59	**79.3**	14	**4.79**	**158**
	RCM	**23**	4.13	17	4.92	**90.1**	**15**	**4.07**	119	**11**	5.38	317
	ND	38	6.01	23	6.54	122	27	6.23	97.9	18	6.01	209
40^3	NO	**28**	9.41	20	10.9	**178**	19	9.64	236	16	13.5	615
	RB	38	12.7	**17**	**9.21**	267	31	13.1	**155**	17	**9.37**	**310**
	RCM	**28**	**9.36**	20	10.9	**178**	**18**	**9.38**	236	**13**	11.8	621
	ND	45	14.3	27	15.1	241	33	14.8	191	22	13.7	409
50^3	NO	**34**	22.5	24	25.2	**351**	**22**	22.8	466	18	29.7	1206
	RB	47	29.5	**20**	**19.4**	527	37	29.3	**303**	19	**19.1**	**607**
	RCM	**34**	**22.3**	24	25.0	**351**	**22**	**21.9**	466	**16**	26.5	1216
	ND	55	34.5	33	34.6	477	39	34.0	376	27	32.6	804

weighed by the lower cost of applying the RB preconditioner, so that overall RB becomes the best reordering for large problems with a moderately high level of fill such as 5 or 10.

In the 3D case the best ordering for SILUT depends on the level of fill allowed: for SILUT(.005, 5) the fastest solution is obtained with RCM, whereas RB is the best ordering for SILUT(.001, 10). This appears to be due to the fact that RB does a better job at preserving sparsity in the incomplete factors, while the convergence rate is only marginally worse than with RCM.

3.2. Convection-diffusion problems. As we move from SPD problems to problems which are nonsymmetric and lack diagonal dominance, the relative merits of RB and RCM observed in the previous subsection change. In this subsection, we consider two- and three-dimensional instances of Problem 2, by setting $\varepsilon = 1/100$ and $\varepsilon = 1/500$, and compare the performance of the RB, RCM, NO, and ND reorderings.

TABLE 3.3
Iterations (I), time (T), and fill versus problem size (N), for Problem 2, 2D domain, $\varepsilon = 1/100$, with different reorderings and preconditioners

$N{=}n_x^2$	Ord	Preconditioner										
		ILU(0)		ILU(1)			ILUT(.005,5)			ILUT(.001,10)		
		I	T (sec)	I	T (sec)	Fill (k)	I	T (sec)	Fill (k)	I	T (sec)	Fill (k)
128^2	NO	**32**	3.41	18	2.34	**31.5**	16	2.55	80.6	8	2.07	177
	RB	106	11.0	**14**	2.11	63.0	11	1.84	71.8	8	1.74	116
	RCM	**32**	**3.33**	**14**	**1.85**	**31.5**	**9**	**1.47**	**62.7**	**4**	**1.10**	**115**
	ND	90	9.39	38	4.99	57.0	21	3.18	73.0	11	2.41	128
256^2	NO	89	40.0	52	28.2	**127**	41	26.8	324	22	21.1	771
	RB	204	94.6	**34**	**21.3**	254	30	19.6	**286**	18	15.1	**528**
	RCM	**87**	**39.4**	42	22.8	**127**	**26**	**17.2**	309	**13**	**12.1**	592
	ND	182	83.6	83	48.4	228	55	35.0	296	28	22.8	532
300^2	NO	**106**	**64.7**	62	46.1	**174**	51	45.7	475	26	33.8	1091
	RB	239	143	**43**	**35.1**	349	38	32.4	**393**	22	24.6	**733**
	RCM	**106**	65.0	54	40.3	**174**	**34**	**30.3**	430	**17**	**20.6**	842
	ND	201	125	104	83.3	300	66	58.0	427	35	38.5	759
400^2	NO	159	**173**	90	117	**310**	66	106	924	37	82.1	1988
	RB	313	336	**66**	**97.4**	621	56	**84.0**	**700**	32	61.3	**1313**
	RCM	**158**	175	83	108	**310**	**54**	84.5	776	**26**	**55.1**	1555
	ND	277	310	148	210	536	96	152	766	50	96.5	1357

TABLE 3.4
Iterations (I), time (T), and fill versus problem size (N), for Problem 2, 3D domain, $\varepsilon = 1/100$, with different reorderings and preconditioners

$N{=}n_x^3$	Ord	Preconditioner										
		ILU(0)		ILU(1)			ILUT(.005,5)			ILUT(.001,10)		
		I	T (sec)	I	T (sec)	Fill (k)	I	T (sec)	Fill (k)	I	T (sec)	Fill (k)
16^3	NO	**10**	**0.42**	5	**0.35**	**21.1**	4	0.42	34.1	3	0.66	71.5
	RB	19	0.77	**4**	**0.35**	31.6	6	0.45	**19.8**	4	0.53	**39.1**
	RCM	**10**	0.43	5	**0.35**	**21.1**	**3**	**0.33**	32.1	**2**	**0.45**	61.3
	ND	15	0.62	7	0.50	28.4	6	0.55	24.0	4	0.72	45.0
32^3	NO	**8**	**3.16**	8	4.63	**180**	6	4.10	193	6	5.94	401
	RB	36	13.1	7	4.77	270	6	3.95	**156**	5	4.63	**308**
	RCM	**8**	3.17	**5**	**3.30**	**180**	**5**	**3.56**	205	**3**	**3.80**	410
	ND	28	10.6	12	7.31	244	9	5.96	178	7	7.37	348
40^3	NO	**10**	**7.81**	9	10.1	**356**	8	9.48	332	7	11.9	730
	RB	48	34.3	8	10.3	534	8	9.51	**299**	6	9.91	**601**
	RCM	**10**	7.85	**4**	**5.65**	**356**	**4**	**5.83**	370	**3**	**6.75**	659
	ND	33	25.1	15	17.3	483	12	14.1	329	9	16.1	659
50^3	NO	**15**	**23.6**	12	33.4	**703**	11	24.0	587	10	29.2	1332
	RB	57	83.5	10	29.1	1054	11	23.8	**557**	8	23.5	1114
	RCM	**15**	23.7	**6**	**15.2**	**703**	**6**	**14.9**	648	**5**	**16.7**	**1103**
	ND	40	62.5	19	43.3	954	15	34.2	602	11	35.1	1246

The effects of RB, RCM, and ND on the iteration count and execution time are illustrated in Tables 3.3–3.6. Notice that, as for the incomplete Cholesky factorization, ND gives poor performance under all aspects (iterations, time, storage), in all test cases except for the 3D domain, $\varepsilon = 1/500$, with ILU(0).

First, consider the mildly nonsymmetric problems, $\varepsilon = 1/100$ (Tables 3.3 and 3.4). In the two-dimensional case (Table 3.3), RCM and NO are equivalent (up to round-off) for ILU(0) and are the best orderings. However, for ILU(1) for larger problems, RB is the best ordering from a time and iteration count standpoint. The second best ordering is RCM, followed by NO. Notice that RCM and NO induce the same size of the ILU(1) preconditioner, which has about half the size of the fill-in induced by RB. For ILUT, the time, iteration, and storage cost of RB and RCM are close, with RCM being somewhat better for $p = 10$. For larger problems RCM is generally the least expensive in terms of time and iterations, while RB induces the smallest fill. Notice that for large fill the relative storage saving obtained with RB as compared to RCM is less than the corresponding saving for the SPD case.

In the three-dimensional case (Table 3.4), RCM is the best ordering (or nearly so) for all preconditioners, often by a wide margin. RB, which is much worse than NO with ILU(0), gives a performance that is close to that of NO with ILU(1) and ILUT(.005, 5) and is somewhat better than NO with ILUT(.001, 10).

Next, consider the moderately nonsymmetric problems, $\varepsilon = 1/500$ (see Tables 3.5 and 3.6), where a † indicates failure to converge in 5000 iterations.

TABLE 3.5
Iterations (I), time (T), and fill versus problem size (N), for Problem 2, 2D domain, $\varepsilon = 1/500$, with different reorderings and preconditioners

$N{=}n_x^2$	Ord	Preconditioner										
		ILU(0)		ILU(1)			ILUT(.005,5)			ILUT(.001,10)		
		I	T (sec)	I	T (sec)	Fill (k)	I	T (sec)	Fill (k)	I	T (sec)	Fill (k)
128^2	NO	**44**	**4.5**	10	1.4	**31**	9	1.8	99	5	1.8	206
	RB	106	11.2	**5**	**1.0**	63	**4**	1.0	77	4	1.3	130
	RCM	46	4.7	10	1.4	**31**	**4**	**0.9**	**68**	**2**	**0.8**	**110**
	ND	82	8.6	39	5.3	57	12	2.2	75	6	1.7	130
256^2	NO	**6**	**3.0**	12	7.3	**127**	12	8.1	214	9	8.4	482
	RB	244	114.	8	6.0	254	8	5.8	199	5	5.0	353
	RCM	**6**	3.2	**5**	**3.6**	**127**	**5**	**3.6**	**97**	**3**	**3.0**	**222**
	ND	180	85.1	79	46.7	228	22	14.7	255	10	9.4	457
300^2	NO	**20**	**12.6**	15	11.9	**174**	16	14.5	353	10	12.8	753
	RB	276	162.	11	10.4	349	10	9.6	310	7	8.7	553
	RCM	**20**	12.9	**5**	**5.0**	**174**	**5**	**5.1**	**183**	**4**	**5.0**	**343**
	ND	196	121.	93	72.1	300	25	22.4	378	12	15.1	660
400^2	NO	**48**	**52.8**	28	39.4	**310**	27	43.2	738	14	31.4	1495
	RB	332	355.	18	29.3	621	18	29.3	689	11	23.0	1126
	RCM	**48**	53.9	**12**	**18.1**	**310**	**11**	**17.6**	**462**	**5**	**11.4**	**781**
	ND	244	275.	120	170.	536	44	71.2	728	19	39.5	1271

For two-dimensional domains (Table 3.5), NO and RCM are still the best orderings for ILU(0), while RB is much worse than these two. For ILU(1), with the only exception of the case $N = 128^2$, the best ordering is RCM; the performance of RB is about midway between RCM and NO. RCM outperforms RB by all criteria: iterations, time, and preconditioner size. This differs from the mildly nonsymmetric case, for which RB for ILU(1) typically wins in terms of CPU time. The results for ILUT are qualitatively similar to those for ILU(1), with RCM being the clear winner, and RB being better than NO. Notice that the size of the ILUT(.001,10) preconditioner induced by RB is always larger than that of the RCM-induced preconditioner. This behavior is different from that observed for Problem 1 and the $\varepsilon = 1/100$ instance of Problem 2 and indicates that the advantage of RB leading to a smaller ILUT(τ, p)

preconditioner size than RCM for p large enough is problem-dependent.

TABLE 3.6
Iterations (I), time (T), and fill versus problem size (N), for Problem 2, 3D domain, $\varepsilon = 1/500$, with different reorderings and preconditioners

$N{=}n_x^3$	Ord	ILU(0) I	ILU(0) T (sec)	ILU(1) I	ILU(1) T (sec)	ILU(1) Fill (k)	ILUT(.005,5) I	ILUT(.005,5) T (sec)	ILUT(.005,5) Fill (k)	ILUT(.001,10) I	ILUT(.001,10) T (sec)	ILUT(.001,10) Fill (k)
16^3	NO	†		63	3.05	**21.1**	8	0.75	38.9	4	1.05	78.1
	RB	**56**	**2.22**	17	1.01	31.6	19	1.19	**19.9**	7	0.93	**39.8**
	RCM	4792	190	**15**	**0.82**	**21.1**	**4**	**0.40**	30.5	**3**	**0.62**	66.4
	ND	64	2.58	1640	80.7	28.4	26	1.58	24.1	6	0.96	45.2
32^3	NO	†		17	8.73	**180**	7	6.09	313	5	10.4	630
	RB	**38**	**14.0**	12	7.17	270	13	7.85	**159**	8	8.42	**319**
	RCM	†		**9**	**5.98**	**180**	**4**	**3.96**	255	**3**	**4.83**	533
	ND	46	20.4	31	16.4	244	13	8.90	194	6	9.17	366
40^3	NO	85	62.9	15	15.7	**356**	6	11.0	613	5	17.6	1231
	RB	46	38.2	11	13.2	534	13	15.2	**312**	6	15.8	**624**
	RCM	68	58.3	**9**	**10.3**	**356**	**3**	**5.87**	505	**3**	**9.67**	1059
	ND	**39**	**32.0**	23	25.2	483	11	15.8	379	7	19.2	718
50^3	NO	34	**52.3**	16	33.4	**703**	7	23.7	1196	5	33.9	2404
	RB	51	75.5	**8**	20.8	1054	11	26.5	**609**	6	26.4	**1218**
	RCM	**34**	52.9	9	**20.6**	**703**	**4**	**13.6**	977	**3**	**18.9**	2069
	ND	39	78.4	21	46.9	954	11	31.5	744	8	40.0	1409

The results for the 3D domain (see Table 3.6) are similar to those for 2D, with two exceptions. First, for ILU(0), RB and ND are better than NO and RCM for $N = 16^3, 32^3, 40^3$; note that ILU(0) with the natural ordering (as well as RCM) is unstable for $N = 16^3, 32^3$. Second, for ILUT, the minimum size of the preconditioner is induced by RB; this is similar to the behavior observed in Tables 3.2 and 3.4. A salient characteristic of the ILUT preconditioners for the 3D case is that the iteration counts are almost insensitive to the problem size, which suggests that the incomplete factorization is very close to a complete one.

3.3. Summary of results. The experiments above suggest that RB and RCM reorderings can be superior to NO: they may lead to a reduction in the number of iterations and, for threshold-based ILU, to a smaller amount of fill-in. For SPD problems and for mildly nonsymmetric problems, RB and RCM are the best reorderings for the set of preconditioners considered. The winner is either RB or RCM, depending on the type of preconditioner and on the problem size.

On the other hand, RCM is the best reordering for highly nonsymmetric problems, for almost all preconditioners and problem sizes covered in this section; for the few cases where RCM is not the best choice, its performance is very close to the best choice. Therefore, the robustness of RCM makes it the best choice for problems with strong convection. We mention that similar conclusions hold when more adequate discretization schemes are used; see the experiments with locally refined meshes in [2]. While it was already known that RB can outperform NO provided enough fill is allowed (Saad [19]), here we have found that RCM is even better.

We observe that, for the preconditioners with fill, the convergence is faster for the convection-diffusion problems than for the Poisson's equation corresponding to the same problem size, preconditioner, and order.

We should make a comment on the methodology of these experiments. It should be pointed out that these experiments are not exhaustive, and ideally one would like to say that for a given problem, for any fixed (τ, p), one method is better than another, which would imply that the given method for its best (τ, p) is better than the other method for *its* best (τ, p), i.e., the choice of the best ordering is not an artifact of the parameterization of the knobs (τ, p). However, such a test would require a very large number of experiments with different (τ, p) values which would not be practical. Furthermore, we feel the results we have given do give a general sense of the comparative performance of the methods.

In this Section we have not been concerned with the issue of parallelism, and the experiments above were meant to assess the effect of different orderings on the quality of ILU preconditionings in a sequential setting. The remainder of the paper is devoted to an evaluation of different strategies for parallelizing ILU-type preconditioners.

4. Additive Schwarz preconditioning. In this section we examine the effect of the problem size and number of subdomains (blocks the matrix is split into) on the convergence of ASM. The subdomain size, denoted by n, is chosen such that the subdomain boundaries do not cross the z-axis. The experiments in this section use Problems 1 and 3. The number of subdomains is always a divisor (or a multiple) of n_0, the number of grid points along one dimension. Note that this is not always a power of 2.

The ASM preconditioner divides the matrix in overlapping blocks (corresponding to subdomains of the PDE problem), each of which is approximately solved with IC(0) in the SPD case, and ILU(0) in the nonsymmetric case. We call these variants ASM.IC0 and ASM.ILU0, respectively.

The amount of overlap is denoted by δ. Since Additive Schwarz preconditioners can be improved by employing subdomain overlapping, we consider three levels of overlaps: $\delta = 0, 1, 4$; a 0-overlap gives an approximate (since subdomain solves are ILU(0)) Block Jacobi preconditioner. We have employed the ASM preconditioner provided by the PETSc library. Conceptually, the preconditioner is formed by summing the (approximate) inverse of each block, where for each block a restriction operator extracts the coefficients corresponding to that block. A limited interpolation is employed, in which the off-block values for each block are ignored (this is the PETSc's PC_ASM_RESTRICT preconditioner). The number of processors is $\min\{16, K\}$; more processors caused a performance degradation for the problem sizes considered here.

We perform two kinds of scalability experiments. In the first kind, we fix the problem size N and increase the number of subdomains K. In the second, we fix the subdomain size, n, and increase K. We monitor the iteration counts and the execution times. The grid nodes are numbered row-wise in each xy-plane, so each matrix block corresponds to a subdomain in an xy-plane.

4.1. Scalability for constant problem size. For the first kind of scalability experiments, we measure for Problems 1 and 3 the number of iterations and solution times versus the number of subdomains K.

The results for Problem 1 solved with CG preconditioned with ASM are shown in Figures 4.1 through 4.4. The best wall-clock time is reached for about 2^4 subdomains, for both 2D and 3D problems. From a running-time standpoint, $\delta = 0$ is the best choice, even though the iteration count is smaller (for K large enough) when $\delta = 4$. The improvement in convergence brought by the overlap is not large enough to compensate for the higher cost of iterations as compared to zero-overlap. On the other hand, an overlap $\delta > 0$ prevents the iteration count from increasing significantly with K, and this effect is more pronounced in the 2D case. Notice, however, that the number of iterations does not decrease monotonically with increasing overlap: for example, increasing δ from 1 to 4 for 3D, $N = 40^3, K = 2^5$, increases the number of iterations by 10%. Moreover, for $K = 2$ a zero-overlap gives the smallest number of iterations.

ETNA
Kent State University
etna@mcs.kent.edu

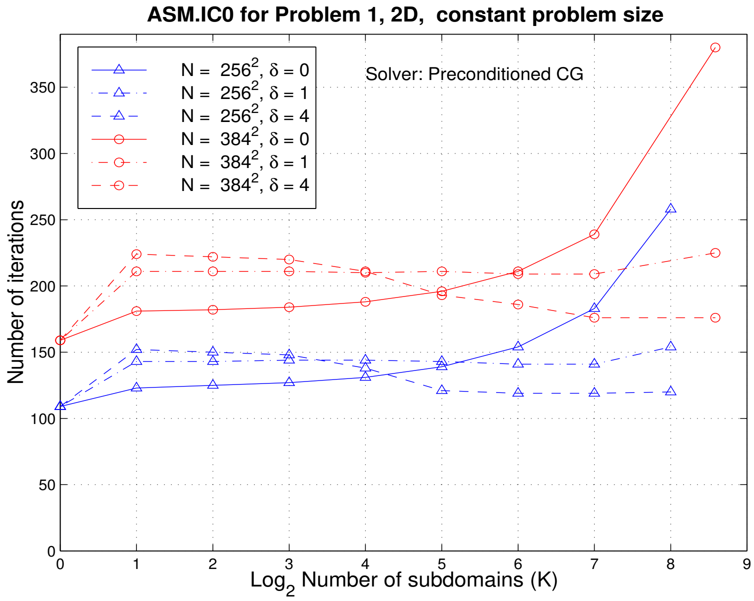

FIG. 4.1. *Iterations versus number of subdomains for Problem 1, 2D domain.*

The experimental results for Problem 3 (3D only) solved with GMRES preconditioned with ASM are shown in Figures 4.5 and 4.6 for $\varepsilon = 1/100$, and Figures 4.7 and 4.8 for $\varepsilon = 1/500$. Notice that for Problem 3, unlike Problem 1, a zero-overlap leads to a number of iterations which grows significantly with the number of subdomains. To keep the iteration count low for $K > 2^4$ a nonzero overlap is necessary; an overlap $\delta = 1$ seems to be more efficient than $\delta = 4$, since for the latter the reduction in the iteration count is offset by the

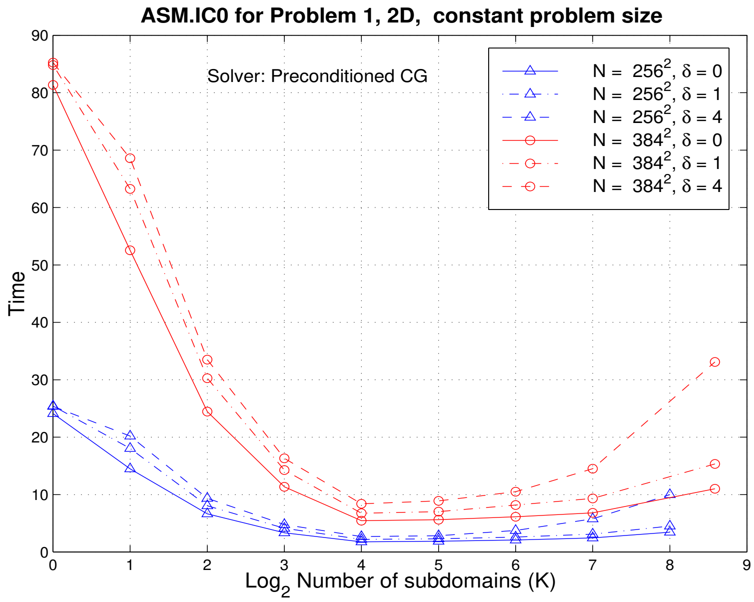

FIG. 4.2. *Time versus number of subdomains for Problem 1, 2D domain.*

ETNA
Kent State University
etna@mcs.kent.edu

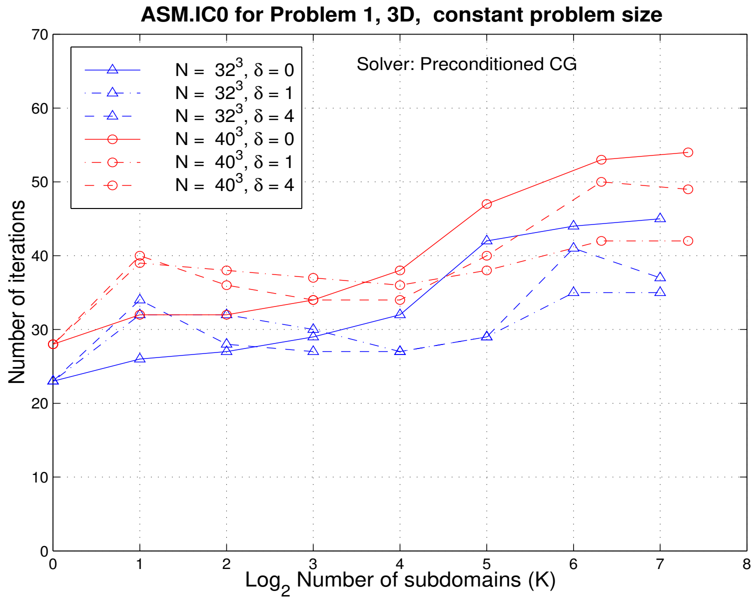

FIG. 4.3. *Iterations versus number of subdomains for Problem 1, 3D domain.*

higher cost per iteration. The rise in iteration count and communication cost with the number of subdomains concur to cause a "valley" shape of the execution time. Thus the execution time can be reduced by increasing K up to a maximum (about 16 for the cases under test), beyond which the time increases with K.

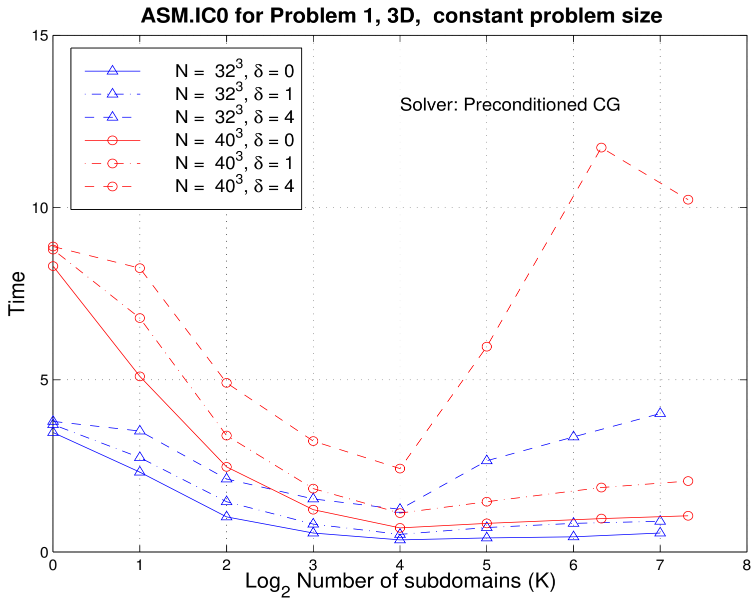

FIG. 4.4. *Time versus number of subdomains for Problem 1, 3D domain.*

ETNA
Kent State University
etna@mcs.kent.edu

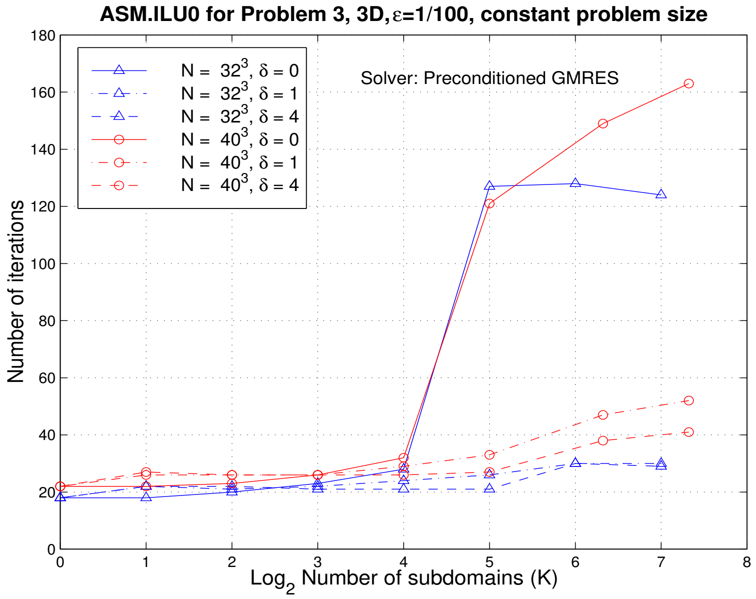

FIG. 4.5. *Iterations versus number of subdomains for Problem 3, 3D, $\varepsilon = .01$.*

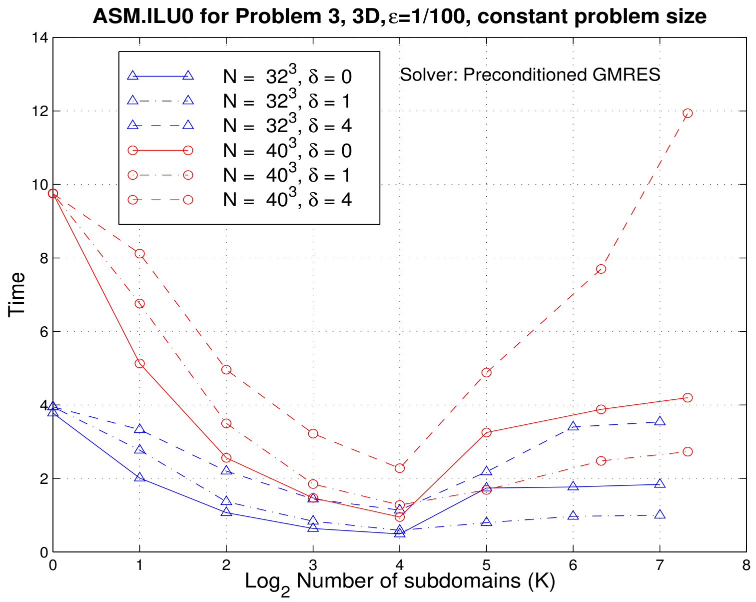

FIG. 4.6. *Time versus number of subdomains for Problem 3, 3D, $\varepsilon = .01$.*

4.2. Scalability for increasing problem size. For the second kind of scalability experiments, the number of iterations and execution times for solving Problem 1 with the CG method are shown in Figures 4.9 through 4.12. The similar plots for Problem 3 (3D only) solved with GMRES are shown in Figures 4.13 and 4.14 for $\varepsilon = 1/100$, and in Figures 4.15 and 4.16 for $\varepsilon = 1/500$. Notice that we have used the same scale for all plots to facilitate comparisons.

The algorithmic scalability for Problem 1 is satisfactory. The number of iterations in-

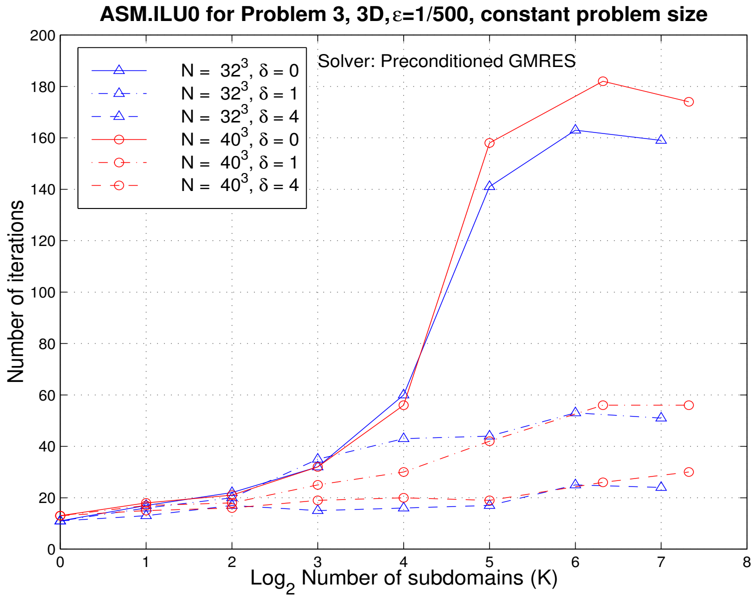

FIG. 4.7. *Iterations versus number of subdomains for Problem 3, 3D,* $\varepsilon = .002$*.*

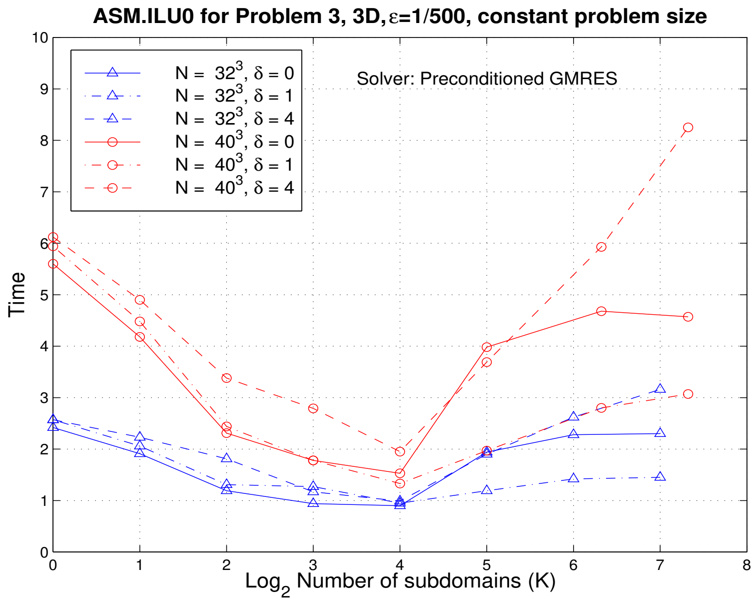

FIG. 4.8. *Time versus number of subdomains for Problem 3, 3D,* $\varepsilon = .002$*.*

creases slowly with K for all the overlaps considered. Notice that the best execution times are obtained with no overlap; an overlap $\delta = 1$ leads to a slight increase in time, while for $\delta = 4$ the cost per iteration is so high that it outweighs the reduction in the number of iterations.

On the other hand, for Problem 3, the number of iterations increases significantly with K for the overlaps $\delta = 0, 1$. With $\delta = 4$, the number of iterations is kept low, but the algorithm's computation and communication requirements increase so much that the execu-

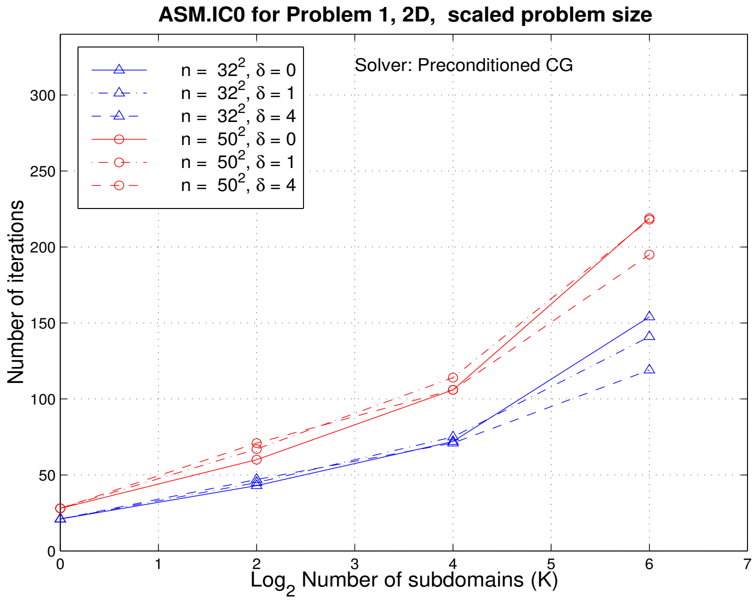

FIG. 4.9. *Iterations versus number of subdomains for Problem 1, 2D domain.*

tion time exceeds that for $\delta = 0$. One reason for the modest scalability is that the subdomain boundaries cut the convection streamlines [5]; thus, we should expect poor performance since the preconditioner fails to model important physical behavior.

The above results suggest that the Additive Schwarz preconditioners have limited scalability with respect to problem size, at least when ILU(0) is used as an approximate subdomain solver.

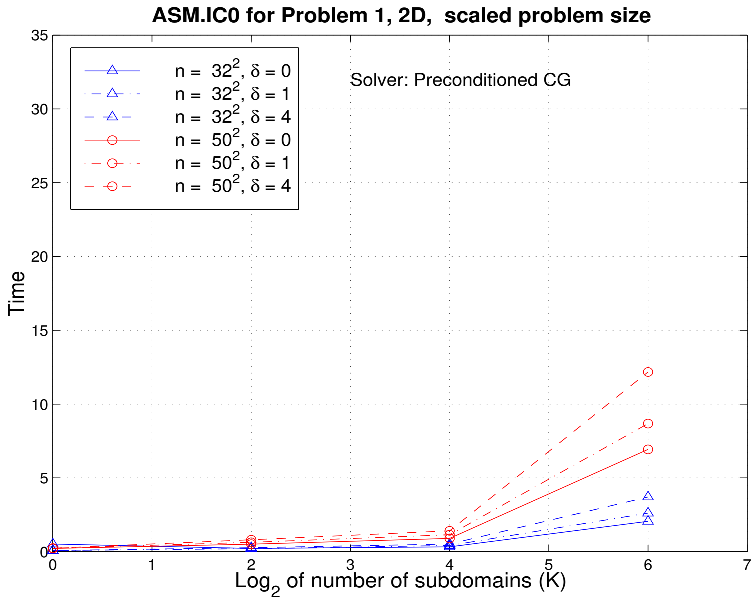

FIG. 4.10. *Time versus number of subdomains for Problem 1, 2D domain.*

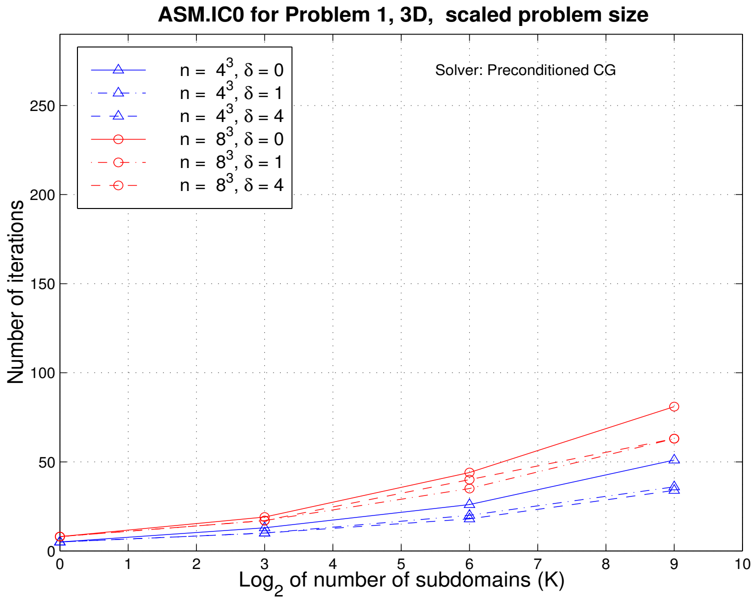

FIG. 4.11. *Iterations versus number of subdomains for Problem 1, 3D domain.*

4.3. Comparison with Red-Black ordering. Both ASM and no-fill incomplete factorization methods with RB ordering lead to highly parallel algorithms. A natural question is which one should be preferred. For Poisson's equation in 2D, a direct comparison between the two methods suggests that PETSc's ASM on 16 processors is faster than IC(0) with RB, even assuming a perfect speed-up for the latter. This shows that the degradation in the rate of convergence induced by the RB ordering in the symmetric case is a serious drawback. For

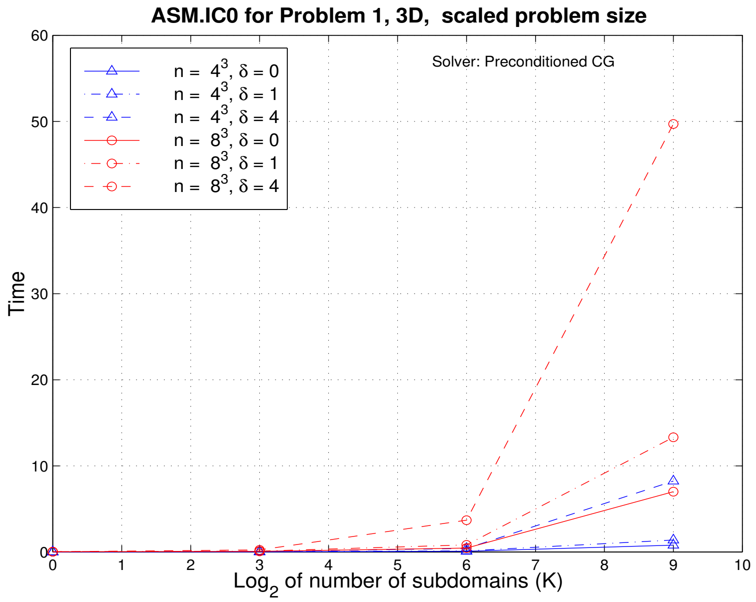

FIG. 4.12. *Time versus number of subdomains for Problem 1, 3D domain.*

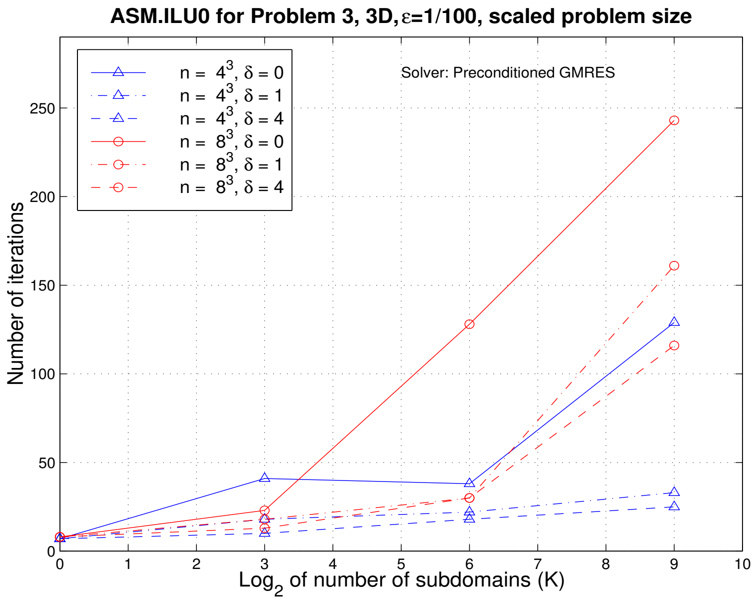

FIG. 4.13. *Iterations versus number of subdomains for Problem 3, 3D,* $\varepsilon = .01$.

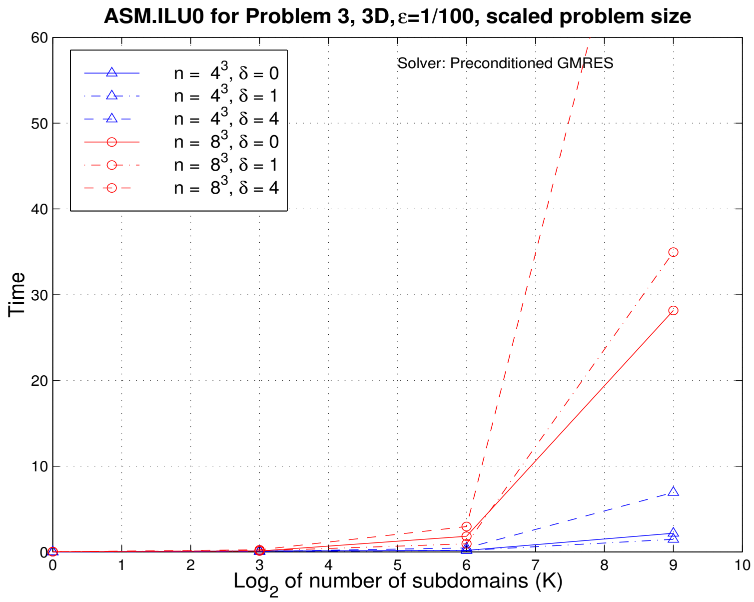

FIG. 4.14. *Time versus number of subdomains for Problem 3, 3D,* $\varepsilon = .01$.

the Poisson equation in 3D, the degradation in convergence rate induced by RB is less severe, and the two methods give comparable performance. This remains true for problems that are only mildly nonsymmetric.

For highly nonsymmetric problems with strong convection, as we saw, ASM has poor convergence properties, whereas ILU(0) with RB ordering appears to be fairly robust. In this case, ILU(0) with RB ordering appears to be better, although the performance of ASM can be improved by taking the physics into account when decomposing the domain and by adding a

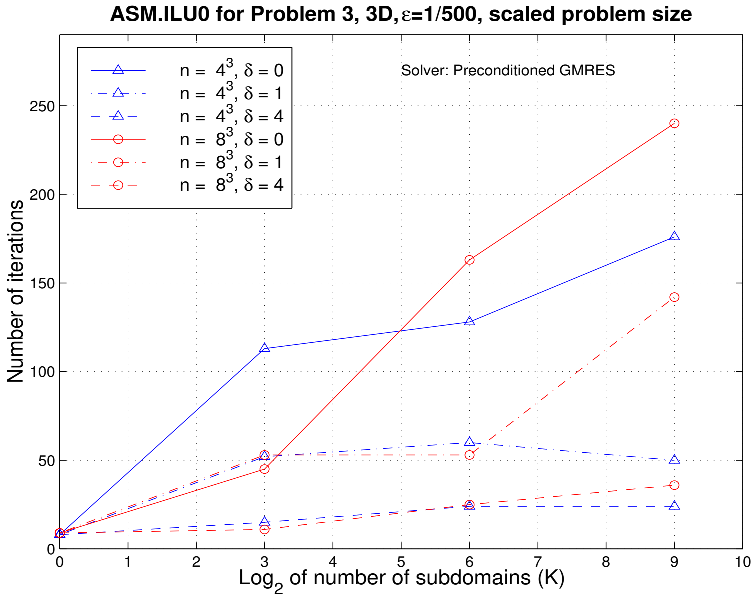

FIG. 4.15. *Iterations versus number of subdomains for Problem 3, 3D,* $\varepsilon = .002$.

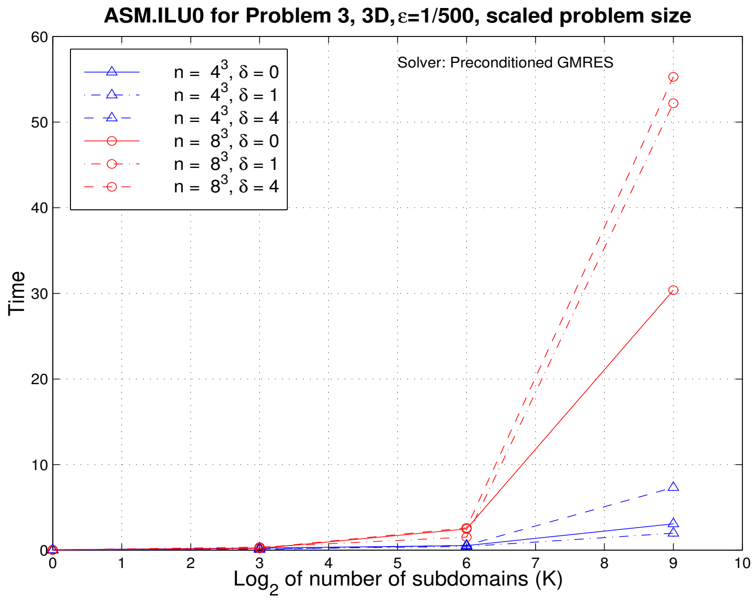

FIG. 4.16. *Time versus number of subdomains for Problem 3, 3D,* $\varepsilon = .002$.

coarse grid correction [23].

5. Multicolorings. In this section we consider multicoloring reorderings for which ILU(0) and ILU(1) preconditioners can be easily parallelized. Assume a regular grid whose nodes are evenly colored, i.e., each of the N_c colors is employed to color N/c nodes. Then ILU(0) preconditioners derived by multicoloring have a DOP of N/N_c. The upside of using multicoloring-based ILU(0) preconditioners is the parallelism. The downside is that multicol-

ETNA
Kent State University
etna@mcs.kent.edu

oring orderings may lead to slower convergence than the NO-based ILU(0) preconditioners. As we will see, however, this need not always be the case.

It is not trivial to improve the accuracy of MCL-based ILU preconditioners by allowing fill, since this may compromise the DOP. We propose a new MCL strategy, called PAR, which gives an ILU(1) preconditioner with $O(N)$ degree of parallelism. We perform numerical experiments highlighting the performance of the PAR ordering and of other commonly used multicolorings.

Poole and Ortega [18] have observed that multicoloring orderings may provide a compromise between the parallelism of RB and typically better convergence of NO. Their experiments for symmetric positive definite problems suggest that multicoloring leads to an increase in the number of iterations by 30 to 100% relative to NO.

Notice that RCM, or for that matter any ordering, can be viewed as a special case of multicoloring [18]. To see this for RCM, consider an $n_x \times n_y$ two-dimensional grid and a second order PDE in which the differential operator does not contain mixed derivatives. Then the RCM ordering can be obtained by assigning one color to each "diagonal" of the grid; therefore, RCM is an $n_x + n_y - 1$ multicoloring. However, this variant of MCL requires $O(\sqrt{N})$ colors, which limits the DOP to $O(\sqrt{N})$. Therefore, RCM is a form of MCL which does not have a scalable DOP. Similarly, for any ordering, the induced set of independent sets or "wavefronts" can be thought of as a set of colors; however, the number of colors grows as the problem size grows for many orderings of interest.

5.1. Parallel reorderings for ILU(0) and ILU(1). We consider three multicolor orderings: RB and two other orderings, denoted by N_c^2COL and PAR, which are described below. We compare the performance of these orderings with the baseline natural order (NO). Preconditioning is done with ILU(0) and ILU(1); ILU(1) is employed only for the PAR ordering which provides parallelism for ILU(1).

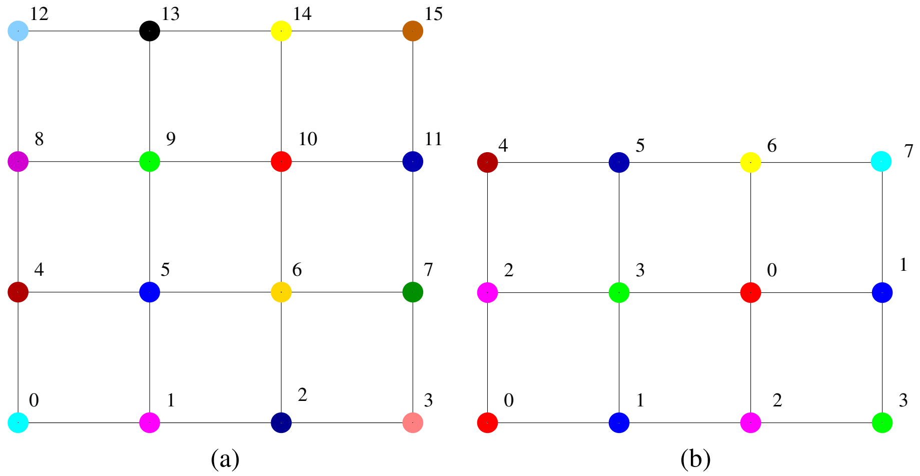

FIG. 5.1. 16-*COL pattern (a) and PAR pattern (b) for a two-dimensional grid.*

The N_c^2COL ordering is based on coloring a square block of $N_c \times N_c$ grid points with N_c^2 colors assigned in natural order. By repeating this coloring "tile" so as to cover the entire grid, we obtain the N_c^2COL ordering. Figure 5.1 (a) shows the coloring tile for $N_c = 4$. The resulting matrix structure is called a *16-colored matrix*; see [18]. The N_c^2COL ordering is generalized to 3D domains as follows. If c is the color of the grid point (i, j, k), then

$N_c^2 - 1 - c$ is the color of the grid point $(i, j, k+1)$, where $0 \leq c \leq N_c - 1$, $1 \leq i \leq n_x$, $1 \leq j \leq n_y$, $1 \leq k < n_z$.

The PAR ordering is based on an 8-coloring scheme obtained by using the coloring pattern shown in Figure 5.1 (b) to cover the entire grid. This pattern induces an ordering for which the permuted matrix $\hat{A} = PAP^T$ has the following properties:

- $\hat{A}$ has the diagonal blocks diagonal;
- The ILU(1) factors of $\hat{A}$ still have the diagonal blocks diagonal. In other words, no edge between nodes of the same color is introduced in the factorization.

The matrix pattern obtained by applying the PAR ordering to the matrix induced by a 5-point stencil discretization on an 8×8 grid is shown in Figure 5.2; the pattern of the ILU(1) factorization for this order is shown in Figure 5.3. Notice that the ILU(1) preconditioner has 8 diagonal blocks, four blocks of dimension 12, and four of dimension 4, each of them having all the off-diagonal entries zero.

In general, for a rectangular grid with N nodes, the degree of parallelism of the PAR preconditioner is $O(N/12)$; this arises from using colors 5 to 8 once for each 12-node subgrid. The PAR coloring can also be generalized to 3D domains, leading to a 16-color pattern, as shown in Figure 5.4.

The PAR coloring can be easily generalized to an arbitrary mesh topology, in which case at most d^2 colors suffice, where d is the degree of the adjacency graph G. Indeed, the PAR coloring can be defined by the property that any two vertices in G connected by a path of length at most two have different colors. Thus, a PAR-type coloring for ILU(1) is any coloring of the graph of A^2. Incidentally this shows that, in general, the PAR ordering for ILU(1) is not unique, since there exists more than one coloring of the graph of $G(A^2)$. More generally, any coloring of the graph of A^{2k} yields a PAR-type coloring for ILU(k).

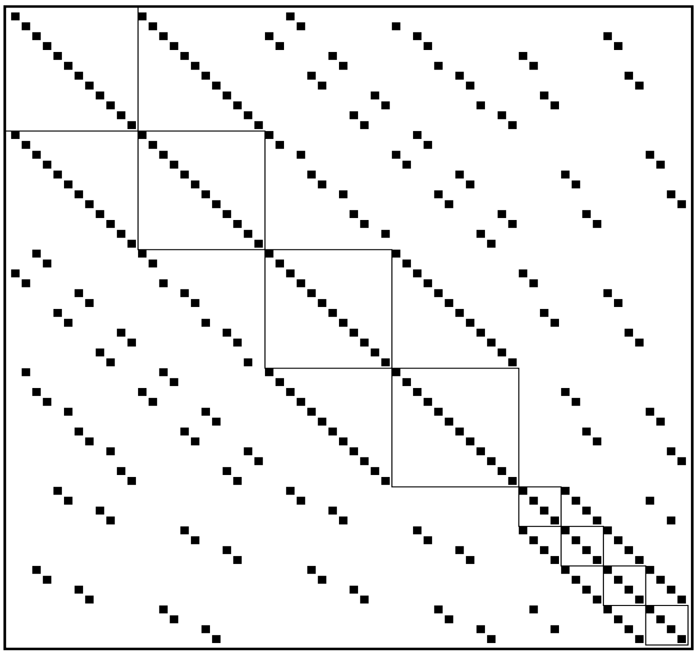

FIG. 5.2. *Matrix Pattern induced by the PAR ordering,* 8×8 *grid.*

ETNA
Kent State University
etna@mcs.kent.edu

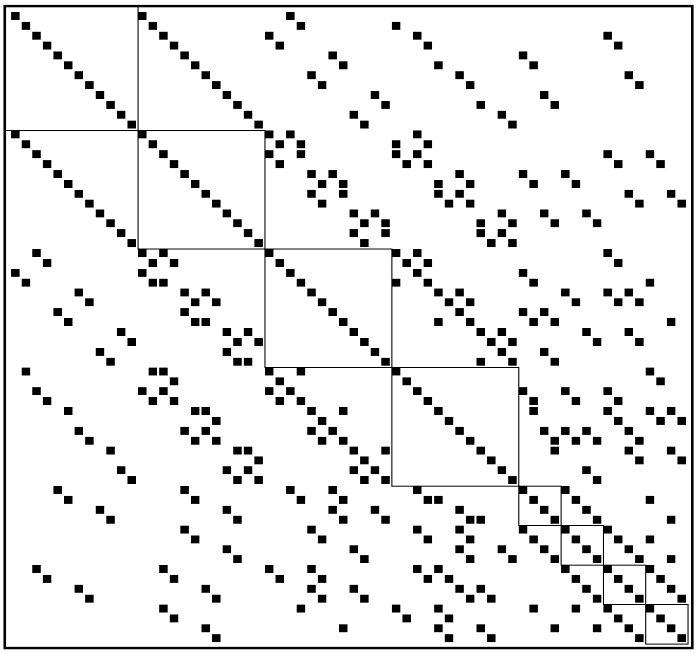

FIG. 5.3. *ILU(1) preconditioner pattern for the PAR ordering,* 8×8 *grid.*

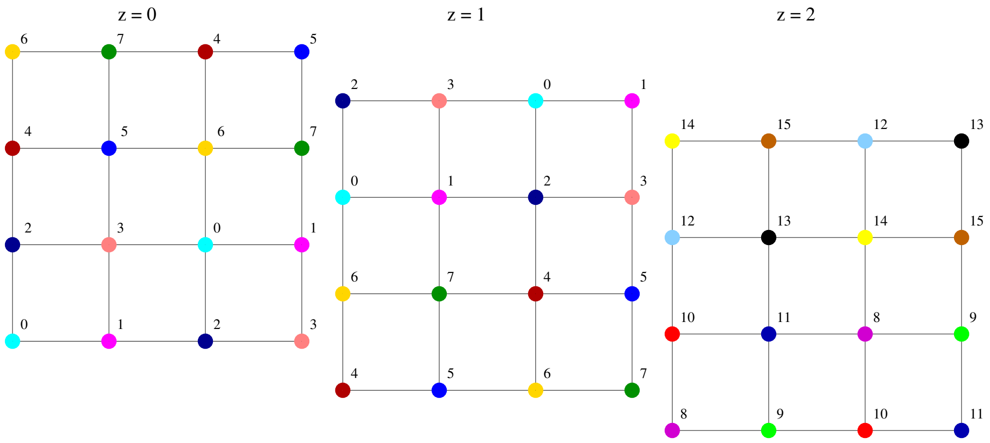

FIG. 5.4. *PAR pattern for a three-dimensional grid.*

In the following two subsections we give numerical results for Problems 1 and 2. We abbreviate the reordering-preconditioner combination by the name of the reordering method separated by a dot from the name of the preconditioner; for instance, PAR.ILU1 denotes the parallel coloring combined with ILU(1) preconditioning. For Problem 1, the linear solver is CG, while for Problem 2 it is Bi-CGSTAB; we have also tested Transpose-Free QMR (see [10]) and obtained results very close to those for Bi-CGSTAB. Bold fonts are used to

ETNA
Kent State University
etna@mcs.kent.edu

indicate the best iteration counts and times for all parallel reorderings, i.e., excluding NO, the performance of which is given for the purpose of comparing the best parallel reordering with NO. A † indicates failure to converge in 5000 iterations.

5.2. Experiments for Poisson's equation. Tables 5.1 and 5.2 show the number of iterations and the execution times for Problem 1, for 2D and 3D domain, respectively.

TABLE 5.1
Iterations (I) and Times (T) for PAR, RB, and N_c^2COL order, Problem 1, 2D

Ord.prec	$N = 32^2$		$N = 64^2$		$N = 128^2$		$N = 256^2$		$N = 300^2$	
	I	T	I	T	I	T	I	T	I	T
NO.IC0	21	0.13	34	0.61	57	3.72	109	28.07	126	44.90
PAR.IC1	**19**	0.15	**31**	0.77	**52**	4.43	**100**	32.95	**116**	52.78
PAR.IC0	26	0.17	42	0.73	72	4.77	139	37.26	161	57.80
RB.IC0	25	0.14	48	0.82	94	5.88	183	49.23	214	74.90
4COL.IC0	26	0.15	47	0.80	80	5.20	156	40.66	182	63.82
16COL.IC0	24	0.16	40	0.71	67	4.47	129	35.92	151	57.31
64COL.IC0	23	**0.13**	37	**0.66**	62	**4.19**	120	**32.02**	139	**50.05**

TABLE 5.2
Iterations (I) and Times (T) for PAR, RB, and N_c^2COL order, Problem 1, 3D

Ord.prec	$N = 12^3$		$N = 16^3$		$N = 32^3$		$N = 40^3$	
	I	T	I	T	I	T	I	T
NO.IC0	11	0.13	14	0.34	23	4.06	28	9.47
PAR.IC1	**10**	0.20	**12**	0.53	**20**	6.36	**24**	13.50
PAR.IC0	13	0.17	16	0.37	27	4.75	30	**10.22**
RB.IC0	12	0.14	16	0.39	31	5.03	38	11.88
4COL.IC0	13	0.14	17	0.46	30	5.06	37	11.96
16COL.IC0	12	**0.13**	15	0.38	28	4.83	34	11.10
64COL.IC0	13	0.15	15	**0.36**	27	**4.64**	32	10.40

As expected, MCLs give more iterations than NO. However, the larger the number of colors, the smaller the convergence penalty for MCL, so that 64COL is the MCL with the smallest execution time or nearly so. Of course, using more colors decreases the DOP of the preconditioner.

The results for the PAR ordering are interesting. For the 2D case, PAR.IC1 provides good rate of convergence and serial timings close to those obtained with NO. In the 3D case, with the exception of the smallest problem $N = 12^3$, PAR.IC0 is attractive in view of its good convergence rate and DOP.

While PAR.IC1 has the best convergence rate, RB has the highest DOP. On platforms with a small numbers of processors, PAR is expected to be competitive. On the other hand, when the parallel performance is at a premium, modest over-coloring, e.g., using four colors, gives a good convergence-parallelism tradeoff. Excessive multicoloring, e.g., 64COL does not seem to be competitive, because of the limited DOP.

5.3. Experiments for convection-diffusion equation. Tables 5.3 and 5.4 show the number of iterations and timing for Problem 3, for 2D and 3D domain, respectively, and increasing convection and problem size. We recall that for fixed ε, increasing the problem size decreases the relative size of the skew-symmetric part of the matrix, since convection is a first-order term. Also, the matrix tends to become more diagonally dominant.

For a sequential implementation of the ILU(0) preconditioner, NO may be the best ordering for problems with small convection, e.g., $\varepsilon = 1/100$. For problems with moder-

ETNA
Kent State University
etna@mcs.kent.edu

TABLE 5.3
Iterations (I) and Times (T) for PAR, RB, and N_c^2COL order, Problem 2, 2D

ε^{-1}	Ord.Prec	Number of unknowns							
		$N = 64^2$		$N = 128^2$		$N = 256^2$		$N = 300^2$	
		I	T	I	T	I	T	I	T
100	NO.ILU0	8	0.23	32	3.30	89	39.10	106	63.45
	PAR.ILU1	20	0.69	45	5.80	**96**	55.24	**113**	88.03
	PAR.ILU0	30	0.80	63	6.50	129	58.94	155	96.78
	RB.ILU0	59	1.52	106	10.71	204	92.14	239	141.5
	4COL.ILU0	39	1.03	79	8.23	159	74.12	198	122.5
	16COL.ILU0	23	0.62	53	5.58	120	57.96	142	89.49
	64COL.ILU0	**15**	**0.41**	**41**	**4.35**	106	**49.24**	127	**78.76**
500	NO.ILU0	34	0.88	44	4.58	6	3.01	20	12.42
	PAR.ILU1	**27**	0.92	42	5.48	91	52.45	104	81.27
	PAR.ILU0	37	1.02	49	5.07	125	57.16	152	94.53
	RB.ILU0	57	1.50	106	10.87	244	109.6	276	163.2
	4COL.ILU0	63	1.65	73	7.64	169	78.44	199	121.3
	16COL.ILU0	31	0.84	**34**	**3.69**	99	47.68	116	73.19
	64COL.ILU0	29	**0.80**	**34**	3.75	**55**	**26.00**	**64**	**40.03**
1000	NO.ILU0	†	†	†	†	†	†	†	†
	PAR.ILU1	282	8.81	48	6.18	86	49.44	101	80.68
	PAR.ILU0	70	1.88	57	5.89	102	46.70	127	79.91
	RB.ILU0	60	**1.59**	108	10.99	213	95.45	254	155.9
	4COL.ILU0	123	3.21	96	10.03	150	69.96	180	113.9
	16COL.ILU0	64	1.70	**50**	**5.31**	**75**	**36.48**	**97**	**60.98**
	64COL.ILU0	**59**	1.60	78	8.15	251	117.6	115	72.23

TABLE 5.4
Iterations (I) and Times (T) for PAR, RB, and N_c^2COL order, Problem 2, 3D.

ε^{-1}	Ord.Prec	Number of unknowns					
		$N = 16^3$		$N = 32^3$		$N = 40^3$	
		I	T	I	T	I	T
100	NO.ILU0	10	0.43	8	3.80	10	8.04
	PAR.ILU1	**9**	0.58	12	7.27	15	16.84
	PAR.ILU0	13	0.53	18	6.89	24	18.02
	RB.ILU0	19	0.77	36	13.37	48	34.91
	4COL.ILU0	18	0.74	21	8.55	26	19.52
	16COL.ILU0	11	0.46	14	5.64	18	14.05
	64COL.ILU0	**9**	**0.39**	**10**	**4.00**	**13**	**10.06**
500	NO.ILU0	†		†		85	63.05
	PAR.ILU1	95	4.73	44	23.14	35	47.19
	PAR.ILU0	**56**	2.23	47	17.54	43	31.88
	RB.ILU0	**56**	**2.21**	**38**	**14.05**	46	32.95
	4COL.ILU0	1379	55.93	138	59.26	123	91.13
	16COL.ILU0	98	3.94	57	21.84	45	33.55
	64COL.ILU0	†	197.3	41	15.60	**33**	**24.70**
1000	NO.ILU0	†		†		†	
	PAR.ILU1	4030	194.8	†		528	505.1
	PAR.ILU0	157	6.08	97	35.91	96	73.37
	RB.ILU0	**74**	**2.90**	**66**	**24.31**	**55**	**40.01**
	4COL.ILU0	†	196.1	†		†	
	16COL.ILU0	379	14.91	†		187	138.3
	64COL.ILU0	1300	51.27	†		†	

ately strong convection ($\varepsilon = 1/500$), NO is typically not competitive (the exception being $N = 256^2$). For very strong convection ($\varepsilon = 1/1000$), NO consistently fails.

On the other hand, for the test cases considered, there is always a multicoloring that provides good convergence. Moreover, multicoloring typically outperforms NO for $\varepsilon = 1/500, 1/1000$, the exceptions being $\varepsilon = 1/500$, 2D domain, $N = 256^2, 300^2$. The instability of the ILU(0) preconditioners induced by NO for strongly nonsymmetric problems has been explained by Elman [8] (see also Benzi *et al.* [2]) by the ill-conditioning of the back and forward solves. In these cases, MMD induces an ILU(0) preconditioner that leads to stable solves; see [2]. Here we find that parallel orderings often have a stabilizing effect on ILU(0) for convection-dominated problems (see the 3D case, $\varepsilon = 1/1000$).

Therefore, we conclude that ILU(0) with a parallel ordering is attractive for convection-dominated problems, particularly in a parallel environment where MMD would lead to a preconditioner which does not parallelize well. However, selecting a good multicoloring method is not straightforward. Out of the 24 test cases considered here, 64COL is the best choice in 12, RB in 6, 16COL and PAR.ILU1 in 4 (there are two ties). The iteration count for PAR.ILU1, 64COL, and 16COL decreases when the problem size increases (except for 2D, $N \geq 256^2$) for $\varepsilon = 1/500, 1/1000$. RB has a similar behavior for $\varepsilon = 1/1000$. This is in agreement with the fact that as N increases the problem becomes closer to being diagonally dominant and symmetric, since we increase the size of the problem by reducing the grid step.

In a parallel implementation the best choice may be either RB or 64COL, depending on the number of available processors. Specifically, RB may be a better choice if sufficiently many processors are available, due to the larger DOP. In contrast, 16COL and 64COL (which result in faster convergence for $\varepsilon = 1/100, 1/500$ but have less inherent parallelism) are attractive when a small number of processors is available.

6. Conclusions. Our numerical experiments show that preconditioners derived from finite difference matrices reordered with multicoloring strategies can outperform those derived for the naturally ordered system in a significant number of cases. RB is effective for SPD and nonsymmetric problems close to being diagonally dominant, provided that we allow some fill in the incomplete factorization. Even though multicolorings may lead to more Krylov subspace iterations, they have at least three advantages:

- for mildly nonsymmetric problems, the storage required by the preconditioner is typically smaller for RB than for NO with the same ILUT preconditioning method;
- some multicolorings, such as the PAR method, can induce ILU(1) preconditioners that can be parallelized, unlike NO and RCM;
- for highly nonsymmetric problems, MCLs tend to stabilize the incomplete triangular factors.

The first advantage is important when memory is at a premium. For large problems, the time to apply the preconditioner may become a bottleneck for the iterative method. The size of the preconditioner affects the time through the operation count for preconditioning and, for large preconditioners, through memory hierarchy effects which lead to expensive LOAD operations for out-of-cache data. Because RB gives smaller ILUT preconditioners, it is a viable alternative to NO and RCM.

The effect of red-black ordering on symmetric dual dropping ILU preconditionings is positive; similarly for the effect on ILUT for moderately nonsymmetric problems.

RB appears to be fairly robust, although less than RCM. RCM also has the desirable property that its performance improves nearly monotonically as τ is reduced and p is increased in ILUT(τ, p). Unfortunately, RCM has low DOP and does not lead to a scalable algorithm.

The numerical results for ASM with IC(0) or ILU(0) subdomain solver indicate good scalability of the method for Poisson's equation, but poor scalability for convection-dominated problems. Using a significant overlap, such as four cells, succeeds in reducing the growth in the iteration count with the number of subdomains, but leads to a large cost per iteration which offsets the gain of reduced number of iterations. A multilevel approach such as a coarse grid correction (two-level strategy) appears to be necessary in order to keep the number of iterations bounded when the degree of parallelism (number of subdomains) increases and to keep the operation cost per iteration $O(N)$.

Acknowledgments. We would like to thank two anonymous referees for their careful reading of the manuscript. Their comments and suggestions helped us improve the exposition of this article. This work was performed while the third author was a PhD student at Virginia Polytechnic and State University and a Graduate Research Assistant at Los Alamos National Laboratory.

REFERENCES

[1] S. BALAY, W. GROPP, L. C. MCINNES, AND B. SMITH, *Petsc 2.0 User's Manual*, Report ANL 95/11, Argonne National Laboratory, Argonne, IL, 1997.
[2] M. BENZI, D. B. SZYLD, AND A. VAN DUIN, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM J. Sci. Comput., 20 (1999), to appear.
[3] X.–C. CAI, W. D. GROPP, D. E. KEYES, R. G. MELVIN, AND D. P. YOUNG, *Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput., 19 (1998), pp. 246–265.
[4] E. CHOW AND Y. SAAD, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math., 86 (1997), pp. 387–414.
[5] M. A. DELONG, *Two Examples of the Impact of Partitioning with Chaco and Metis on the Convergence of Additive Schwarz-Preconditioned FGMRES*, Los Alamos National Laboratory Technical Report LA-UR-97-4181, Los Alamos, NM, 1997.
[6] I. S. DUFF, M. ERISMAN, AND J. K. REID, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, UK, 1986.
[7] I. S. DUFF AND G. A. MEURANT, *The effect of ordering on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–657.
[8] H. C. ELMAN, *A stability analysis of incomplete LU factorizations*, Math. Comp., 47 (1986), pp. 191–217.
[9] H. C. ELMAN, *Relaxed and stabilized incomplete factorizations for non-self-adjoint linear systems*, BIT, 29 (1989), pp. 890–915.
[10] R. W. FREUND, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.
[11] A. GEORGE, *Computer Implementation of the Finite Element Method*, Report STAN-CS-208, Stanford University, Department of Computer Science, 1971.
[12] A. GEORGE AND J. W. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[13] J. L. GUSTAFSON, *Reevaluating Amdahl's law*, Comm. ACM, 31 (1988), pp. 532–533.
[14] M. R. HESTENES AND E. L. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.
[15] M. T. JONES AND P. E. PLASSMANN, *Scalable iterative solution of sparse linear systems*, Parallel Comput., 20 (1994), pp. 753–773.
[16] J. W. LIU, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 11 (1985), pp. 141–153.
[17] 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 (1977), pp. 148–162.
[18] E. L. POOLE AND J. M. ORTEGA, *Multicolor ICCG methods for vector computers*, SIAM J. Numer. Anal., 24 (1987), pp. 1394–1418.
[19] Y. SAAD, *Highly parallel preconditioners for general sparse matrices*, in Recent Advances in Iterative Methods, IMA Volumes in Mathematics and its Applications, G. Golub, R. M. Luskin, and A. Greenbaum, eds., vol. 60, NY, 1994, Springer-Verlag, pp. 165–199.
[20] ———, *ILUT: A dual-threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[21] ———, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, MA, 1996.
[22] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[23] B. SMITH, P. BJØRSTAD, AND W. GROPP, *Domain Decomposition: Parallel Multilevel Methods for Elliptic PDEs*, Cambridge University Press, New York, 1996.

[24] R. S. VARGA, *Factorizations and normalized iterative methods*, in Boundary Problems in Differential Equations, R. E. Langer, ed., Univ. of Wisconsin Press, Madison, WI, 1960, pp. 121–142.
[25] H. A. VAN DER VORST, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.
[26] P. H. WORLEY, *Limits on parallelism in the numerical solution of linear PDEs*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 1–35.
[27] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.