Computer Physics Communications 53 (1989) 237–252
North-Holland, Amsterdam

PARALLEL ELLIPTIC PRECONDITIONERS: FOURIER ANALYSIS AND PERFORMANCE ON THE CONNECTION MACHINE *

Tony F. CHAN, C.-C. Jay KUO

Department of Mathematics, University of California, Los Angeles, CA 90024, USA

and

Charles TONG

Department of Computer Science, University of California, Los Angeles, CA 90024, USA

Received 24 August 1988

We study the performance of several widely used preconditioners for 2D elliptic partial differential equations (SSOR, ILU, MILU and polynomial preconditioners) with the natural and red–black orderings implemented on the Connection Machine (CM). Their performance is primarily influenced by two factors: the rate of convergence and the ease of parallelization. The convergence rate is analyzed by Fourier analysis and confirmed with experimental results. Although the naturally ordered SSOR and MILU preconditioners have convergence rates one order faster than the other preconditioners, the experiments show that the red–black ordered SSOR, ILU, MILU, polynomial preconditioners takes less execution time than their naturally ordered counterparts. This is partially due to the fact that the red–black ordering provides more parallelism than the natural ordering.

1. Introduction

This paper concerns the analysis and implementation of parallel iterative methods for solving elliptic partial differential equations (PDEs). Specifically, we are interested in the preconditioned conjugate gradient method (PCG), which has been successfully used in many areas of scientific computing. We demonstrate that parallel machines can change the relative cost of algorithms, and study the tradeoffs one must make when implementing them on a massively parallel computer such as the Connection Machine.

The fundamental tradeoff is one between the rate of convergence and the ease of parallelization, particularly concerning the implementation of preconditioning, which is an important component of the PCG algorithm. Many preconditioners that have been derived before the advent of parallel computing are unfortunately very much sequential in nature and hence cannot be implemented efficiently on a massively parallel computer. A principal obstacle to parallelization is the sequential manner in which many preconditioners use in traversing the computational grid [21,22]. The data dependence implicitly prescribed by the method fundamentally limits the amount of parallelism available. For the past several years, there has been a lot of research in search of efficient parallel preconditioners [2,4]. One method is to reorder the sequence of operations in the construction of a sequential preconditioner in order to reduce the data dependency and maximize the number of operations that can be performed in parallel. An example of such a parallel ordering is the classical red–black ordering for five point discretizations, which requires only two iterations to cover the

* This work was supported in part by the Department of Energy under contract DE-FG03-87ER25037, the National Science Foundation under contracts NSF-DMS87-14612 and BBS 87 14206, the Army Research Office under contract DAAL03-88-K-0085 and by the Research Institute for Advanced Computer Science, NASA Ames.

whole grid. Another method is to invent new preconditioners that are by design parallel in nature. An example is the class of polynomial preconditioners that we shall discuss later. However, both of the above methods generate preconditioners whose convergence behaviors are in general different from those of their sequential counterparts. In many cases, these new methods in fact converge considerably slower. Therefore, a fundamental issue is whether the gain in the amount of parallelism can overcome the loss in the convergence rate.

One important factor in this tradeoff is the convergence rate of iterative algorithms. The convergence rates for sequential preconditioners have been studied extensively and are well understood [6,8]. However, the convergence rates for the same preconditioners with the parallel orderings have only recently been studied [18]. While they have been observed empirically to converge much slower than their sequential counterparts [2,4], there are few proofs of their convergence rates. One goal of this paper is therefore to present a framework for analyzing the convergence behavior of elliptic preconditioners using Fourier analysis, which has been very successful in providing insights into their performance [8,18]. For the sequential natural ordering, a von Neumann type Fourier analysis is used and most of the classical convergence results can be recovered. For the parallel red–black ordering, a two-color fourier analysis is used which provides rigorous convergence rates for the model Poisson problem.

Obviously, the tradeoff will also depend on the particular computer used in the implementation and the number of processors available. In general, the more processors there are, the more important it is to have an inherently parallel algorithm. The computer used in our numerical experiments, a Connection Machine with 16 K processors, is a massively parallel computer and therefore favors algorithms with massive parallelism built in. As we shall see, this is essentially confirmed by our numerical experiments.

The outline of the paper is as follows. In section 2, we give a brief survey of preconditioners for elliptic PDEs, and show how to construct the SSOR, ILU, MILU and polynomial preconditioners for the model Poisson problem for both the natural and red–black orderings. Section 3 introduces two Fourier analytical tools which are then used to analyze the spectra of the preconditioned Laplacian operators with naturally and red–black ordered preconditioners, respectively. In particular, the SSOR preconditioned Laplacian is used as example to demonstrate the analysis. Finally, the details of implementing the PCG algorithm on the Connection Machine and the results of our experiments are presented in section 4.

2. Brief survey of elliptic preconditioners

Consider the numerical solution of the following self-adjoint elliptic PDEs

$$\frac{\partial}{\partial x}\left(p(x, y)\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(q(x, y)\frac{\partial u}{\partial y}\right) + \sigma(x, y)u=f(x, y), \tag{2.1}$$

where $p(x, y)>0$ and $q(x, y)>0$ and $\sigma(x, y) \leq 0$ on a closed domain Ω with Dirichlet, Neumann or mixed boundary conditions. A special example of (2.1) is the Poisson equation, where $p(x, y)=q(x, y)=1$ and $\sigma(x, y)=0$, which arises in many engineering and physical applications. We can discretize eqs. (2.1) with a finite-different or finite-element scheme and obtain a system of N linear difference equations

$$Au_{\mathrm{d}}=f_{\mathrm{d}}, \tag{2.2}$$

where A is a sparse $N \times N$ matrix, and u_{d} and f_{d} are discrete approximations of u and f, respectively. Due to the self-adjoint property of the differential operator in (2.1), the discretized coefficient matrix A is symmetric and positive definite (SPD).

There are numerous iterative algorithms used to solve the discretized linear system (2.2). Basic iterative methods [14], such as the Jacobi and Gauss–Seidel methods, can be obtained by performing relaxation on each grid value. The convergence rates of basic iterative methods are slow and take at least $\mathcal{O}(N)$ iterations to converge. Thus, in practice, basic iterative methods are usually combined with efficient accelerating procedures to im-

prove their convergence rates. One kind of acceleration is the SOR (Successive OverRelaxation) method which accelerates the basic Gauss–Seidel relaxation and improves the convergence rate to $\mathcal{O}(\sqrt{N})$ iterations to converge. In addition to the SOR acceleration, there are two other kinds of acceleration procedures which have been commonly used, namely, the Chebyshev Semi-Iterative (CSI) and the Conjugate Gradient (CG) procedures [13,14]. Since accelerating parameters are determined in the process of the CG procedure but have to be estimated in advance for the CSI procedure, the CG acceleration is often preferred.

Roughly speaking, by applying the CSI or CG procedures to a basic iterative method whose corresponding coefficient matrix is SPD and has a condition number κ, the resulting method requires $\mathcal{O}(\sqrt{\kappa})$ iterations to converge [6]. For example, since the basic Jacobi iteration has a condition number $\mathcal{O}(N)$, the number of iteration required for the CSI- or CG-accelerated Jacobi is $\mathcal{O}(\sqrt{N})$, which is the same order as the SOR method. The advantage of the CSI or CG procedure over the SOR procedure is that they are widely applicable while the SOR acceleration is primarily used to accelerate the consistently ordered matrix [14,25,26] and, consequently, faster convergence rates may be achieved by using the CSI or CG acceleration. To see this, let us consider the SSOR (Symmetric SOR) iteration with the natural ordering, which has a condition number $\mathcal{O}(\sqrt{N})$ (see section 3.1), to be accelerated by the CSI or CG procedure. The resulting method requires only $\mathcal{O}(N^{1/4})$ iterations to converge.

Since the convergence rate is controlled essentially by the condition number of the coefficient matrix A, it is desirable (and an active research topic) to seek a system which is closely related to (2.2) and whose coefficient matrix has a smaller condition number. One such technique is called *preconditioning* [6,13] and the resulting system is the *preconditioned* system. In mathematical terms, we transform (2.2) into

$$Bv = g, \tag{2.3}$$

where the solution v can be easily converted to the desired solution u_d of the original system (2.2) and the condition number of B is smaller than that of A. Once common way to construct (2.3) is to determine a preconditioning matrix M, or *preconditioner*, so that M can be inverted efficiently and $M \approx A$ so that the condition number $\kappa(M^{-1}A)$ is small. Thus, by multiplying (2.2) with M^{-1} and relating the result with (2.3), we have

$$B = M^{-1}A, \quad v = u_d, \quad g = M^{-1}f_d. \tag{2.4}$$

The SSOR [5], ILU (Incomplete LU) [20], MILU (Modified ILU) [11] and polynomial preconditioners all belong to this type [6]. Recently, another preconditioning technique known as the *hierarchical basis* or the *multilevel* preconditioning [2,27] is under study. Performance and analysis of preconditioners with hierarchical basis will be reported elsewhere. In this paper, we concentrate on the SSOR, ILU, MILU and polynomial preconditioners and describe their forms for the rest of this section.

2.1. Model Poisson problem and orderings

In the following discussion, a simple example is chosen to illustrate the construction of various preconditioners. The example is the discrete Poisson equation on the square $\Omega = [0,1]^2$ with appropriate boundary conditions,

$$\frac{1}{h^2}\left(u_{j+1,k} + u_{j-1,k} + u_{j,k+1} + u_{j,k-1} - 4u_{j,k}\right) = f_{j,k}, \tag{2.5}$$

where h is the grid spacing and $u_{j,k}$ is used to approximate the value of $u(jh, kh)$. Note that the total number N of variables is related to h by $N \approx h^{-2}$. It is convenient to rewrite the difference equation (2.5) in terms of shift operators

$$\begin{aligned} &A_{j,k}u_{j,k} = -\frac{h^2}{4}f_{j,k}, \\ &A_{j,k} = 1 - \tfrac{1}{4}\left(E_x + E_x^{-1} + E_y + E_y^{-1}\right). \end{aligned} \tag{2.6}$$

where E_x and E_y are shift operators along the x- and y-directions,

$$E_x u_{j,k} = u_{j+1,k}, \quad E_x^{-1}u_{j,k} = u_{j-1,k},$$
$$E_y u_{j,k} = u_{j,k+1}, \quad E_y^{-1}u_{j,k} = u_{j,k-1}.$$

In (2.6) $A_{j,k}$ is the local operator at the grid point (jh, kh) and the collection of local operators $A_{j,k}$ at all grid points gives the global operator denoted by A_{op}. The major difference between the global operator A_{op} and the coefficient matrix A is that a 1D ordering has to be specified for describing A but not for A_{op}.

The ordering of grid points in general plays an important role in determining the form of the coefficient matrix and hence that of preconditioners. Two commonly used orderings, i.e. the natural and red–black orderings, will be discussed and compared in this paper. In the natural ordering, grid points are ordered in the rowwise (or columnwise) fashion. By the red–black ordering, grid points are partitioned into red and black groups, where a grid point (j, k) is red if $j+k$ is even and black if $j+k$ is odd, and then red points are ordered before black points and the natural ordering is used within each group. In the context of parallel computation, we are interested in maximizing the number of operations at grid points among which there is no data dependence. Consequently, these operations can be performed in parallel and the ordering for these grid points does not affect the final result. For example, in the solution if (2.5), it is appropriate to consider the following natural and red–black orderings:

natural ordering (parallel version)

$(j, k) < (j', k')$ if $j+k < j'+k'$,

red–black ordering (parallel version)

$(j, k) < (j', k')$ if

(j, k) red and (j', k') black,

where the order between grid points is denoted by the inequality sign. These two orderings for the grid points on a uniform 6×6 square grid are illustrated in fig. 1. Note that the same ordering number is assigned to grid points (j, k) with the same $j+k$ in the natural ordering and grid points of the same color in the red–black ordering. This implies that operations at these grid points can be performed in parallel. The red–black ordering is more attractive than the natural ordering in parallel computation, as far as the computation time

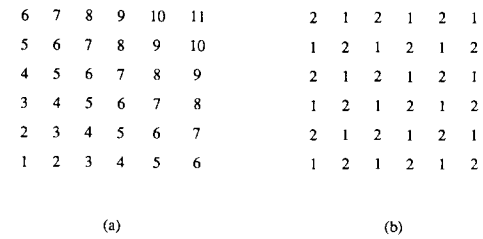


Fig. 1. Parallel (a) natural and (b) red/black orderings.

per iteration is concerned, since it takes two steps to sweep all grid points while the natural ordering takes $\mathcal{O}(\sqrt{N})$ steps. Nevertheless, the convergence rate of some iterative algorithms may be slowed down by changing from the natural ordering to the red–black ordering as analyzed in section 3.

2.2. SSOR, ILU, MILU and polynomial preconditioners

2.2.1. SSOR preconditioner

Generally speaking, given an ordering and its corresponding coefficient matrix A, we can partition A into

$$A = D - L - U, \tag{2.7}$$

where D, L and U are, respectively, the diagonal and strictly upper and lower triangular matrices. The SSOR preconditioner is then defined to be [5]

$$M_{\text{S}} = (D - \omega L) D^{-1} (D - \omega U), \tag{2.8}$$

where ω is the relaxation parameter.

For the model Poisson problem with the *natural* ordering, the partitioning (2.7) leads to the following local operators

$$D_{j,k} = 1, \quad L_{j,k} = \tfrac{1}{4}\left(E_x^{-1} + E_y^{-1}\right),$$
$$U_{j,k} = \tfrac{1}{4}\left(E_x + E_y\right).$$

Hence, by using operator algebra, the local operator for the naturally ordered SSOR preconditioner can be computed as

$$\begin{aligned}(M_{\text{S}})_{j,k} &= (1 - \omega L_{j,k})(1 - \omega U_{j,k}) \\ &= 1 - \frac{\omega}{4}\left(E_x + E_y + E_x^{-1} + E_y^{-1}\right) \\ &\quad + \frac{\omega^2}{16}\left(2 + E_x^{-1}E_y + E_x E_y^{-1}\right),\end{aligned} \tag{2.9}$$

which corresponds to a 7-point stencil. The local operator for the Laplacian preconditioned by the SSOR preconditioner (2.9) can also be computed as

$$\left(M_S^{-1}A\right)_{j,k} = \left[1 - \frac{\omega}{4}\left(E_x + E_y + E_x^{-1} + E_y^{-1}\right) + \frac{\omega^2}{16}\left(2 + E_x^{-1}E_y + E_xE_y^{-1}\right)\right]^{-1} \times \left[1 - \tfrac{1}{4}\left(E_x + E_y + E_x^{-1} + E_y^{-1}\right)\right]. \tag{2.10}$$

For the model Poisson problem with the *red–black* ordering, the partition (2.7) gives the local operators $D_{j,k} = 1$ and

$$L_{j,k} = \begin{cases} 0, & (j,\ k) \text{ red}, \\ \tfrac{1}{4}\left(E_x + E_x^{-1} + E_y + E_y^{-1}\right), & (j,\ k) \text{ black}, \end{cases}$$

$$U_{j,k} = \begin{cases} \tfrac{1}{4}\left(E_x + E_x^{-1} + E_y + E_y^{-1}\right), & (j,\ k) \text{ red}, \\ 0, & (j,\ k) \text{ black}. \end{cases}$$

Therefore, from (2.8), we obtain the red–black ordered SSOR preconditioner as

$$\left(M_S\right)_{j,k} = \begin{cases} 1 - \dfrac{\omega}{4}\left(E_x + E_x^{-1} + E_y + E_y^{-1}\right), \\ \quad (j,\ k) \text{ red}, \\ 1 - \dfrac{\omega}{4}\left(E_x + E_x^{-1} + E_y + E_y^{-1}\right) \\ \quad + \dfrac{\omega^2}{16}\left(E_x + E_x^{-1} + E_y + E_y^{-1}\right)^2, \\ \quad (j,\ k) \text{ black}. \end{cases} \tag{2.11}$$

The stencil representations of local operators $D_{j,k} - \omega L_{j,k}$, $D_{j,k} - \omega U_{j,k}$ and $M_{j,k}$ with the natural and red–black orderings are depicted in fig. 2. Note that local operators $L_{j,k}$, $U_{j,k}$ and $(M_S)_{j,k}$ are homogeneous at all grid points when the natural ordering is used, but take different forms at red and black points when the red–black ordering is used. In section 3, we will show how to analyze the spectra of these operators with two different Fourier analytical tools.

2.2.2. Incomplete factorization

The ILU and MILU factorizations, originally defined in refs. [20,11] respectively, are to construct $M = LU$ such that L and U have the same sparsity pattern as A and $M \approx A$. Specifically, it is required for both the ILU and MILU factorizations that the off-diagonal nonzero elements of A should have the same values as the corresponding elements of M. The major difference between them is that the ILU factorization requires that the diagonal elements of A and M be also the same while the MILU factorization requires that the row sum of M differ from the row sum of A by a small quantity ch^2, where c is a constant independent of h.

The factorizing local operators $L_{j,k}$ and $U_{j,k}$ generally have different coefficients associated with different grid points due to the boundary effects. However, these coefficients usually reach their asymptotic constant values for the region sufficiently far away from boundaries. In the following, we ignore the boundary effect and investigate the asymptotic preconditioners.

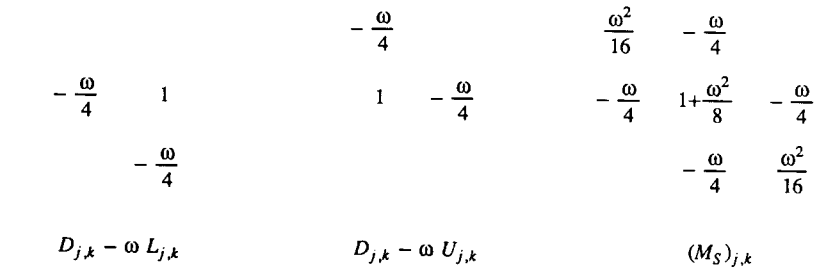
2.2.2.1. ILU preconditioner. For the model Poisson problem with the natural ordering, consider the local operators $L_{j,k}$ and $U_{j,k}$ [8]

$$L_{j,k} = \tfrac{1}{4}\left(a - E_x^{-x} - E_y^{-1}\right), \qquad U_{j,k} = 1 - \frac{1}{a}E_x - \frac{1}{a}E_y, \tag{2.12}$$

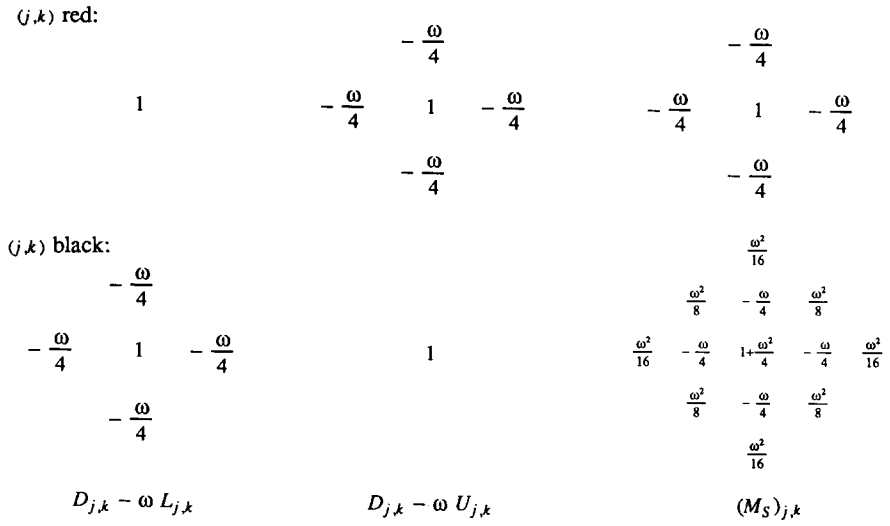
where a is a constant to be determined. Since the operator $L_{j,k}$ (or $U_{j,k}$) has nonzero coefficients for the terms 1, E_x^{-1} and E_y^{-1} (1, E_x and E_y), the sparse pattern of L (or U) is the same as that of the original matrix A for the lower (or upper) triangular part. The ILU preconditioner $(M_I)_{j,k}$ is defined as the product of $L_{j,k}$ and $U_{j,k}$:

$$\left(M_I\right)_{j,k} = \tfrac{1}{4}\left[a + \frac{2}{a} - \left(E_x + E_y + E_x^{-1} + E_y^{-1}\right) + \frac{1}{a}\left(E_xE_y^{-1} + E_x^{-1}E_y\right)\right]. \tag{2.13}$$

By comparing (2.6) and (2.13), we see that the operator $(M_I)_{j,k}$ has the same coefficients as $A_{j,k}$ for terms corresponding to E_x, E_x^{-1}, E_y and E_y^{-1}, which constitute the nonzero off-diagonal terms



(a) natural ordering



(b) red/black ordering

Fig. 2. Stencil representations of local operators for the SSOR preconditioner.

for the sparse matrix A. The ILU factorization requires that $(M_1)_{j,k}$ and $A_{j,k}$ have the same coefficient for the diagonal term as well. This condition imposes that (see fig. 3).

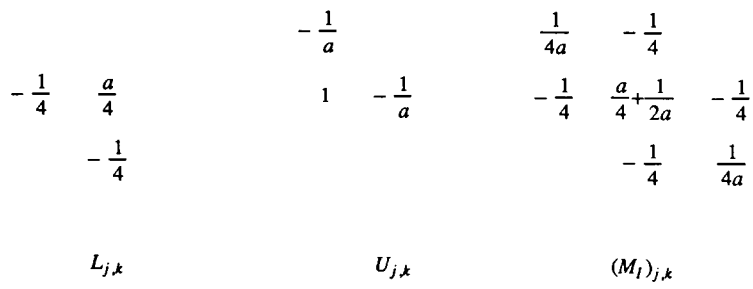
$$a + \frac{2}{a} = 4.$$

The choice of a is $2 + \sqrt{2}$, since this value corresponds to the asymptotic values observed in ILU factorization of the model Poisson problem with Dirichlet boundary conditions.

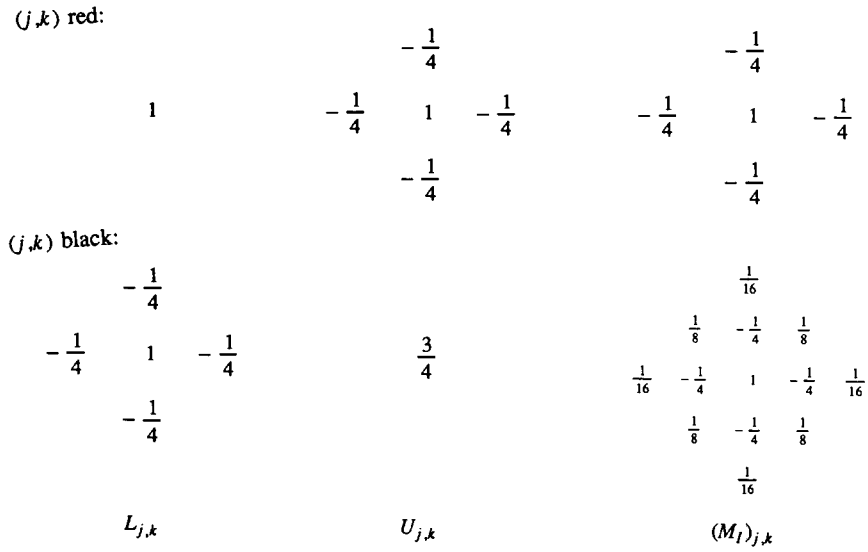
In the red–black ordering context, one can similarly show that the ILU factorization leads to the local operators $L_{j,k}$ and $U_{j,k}$ [18],

$$L_{j,k} = \begin{cases} 1, & (j,\, k) \text{ red}, \\ 1 - \frac{1}{4}\left(E_x + E_x^{-1} + E_y + E_y^{-1}\right), & \\ & (j,\, k) \text{ black}, \end{cases}$$

$$U_{j,k} = \begin{cases} 1 - \frac{1}{4}\left(E_x + E_x^{-1} + E_y + E_y^{-1}\right), & \\ & (j,\, k) \text{ red}, \\ \frac{3}{4}, & (j,\, k) \text{ black}. \end{cases}$$



(a) natural ordering



(b) red/black ordering

Fig. 3. Stencil representations of local operators for the ILU preconditioner.

The asymptotical red–black ILU preconditioner is then defined to be their product (see fig. 3).

2.2.2.2. MILU preconditioner. The MILU preconditioner has the same sparsity pattern as the ILU preconditioner so that (2.12) and (2.13) also apply. That is, for the model Poisson problem with the natural ordering, we can also represent the MILU preconditioner in form

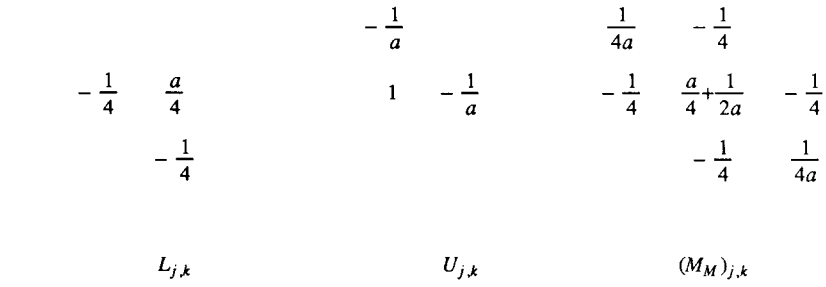
$$(M_{\mathrm{M}})_{j,k} = \frac{1}{4}\left[a + \frac{2}{a} - \left(E_x + E_y + E_x^{-1} + E_y^{-1}\right) + \frac{1}{a}\left(E_x E_y^{-1} + E_y E_x^{-1}\right)\right].$$

Their difference primarily relies on how the constant a is determined. According to the definition of the MILU factorization, the row sum of $(M_{\mathrm{M}})_{j,k}$ and that of $A_{j,k}$, which equals to zero, should differ by a small quantity δ. This leads to the following condition on a,

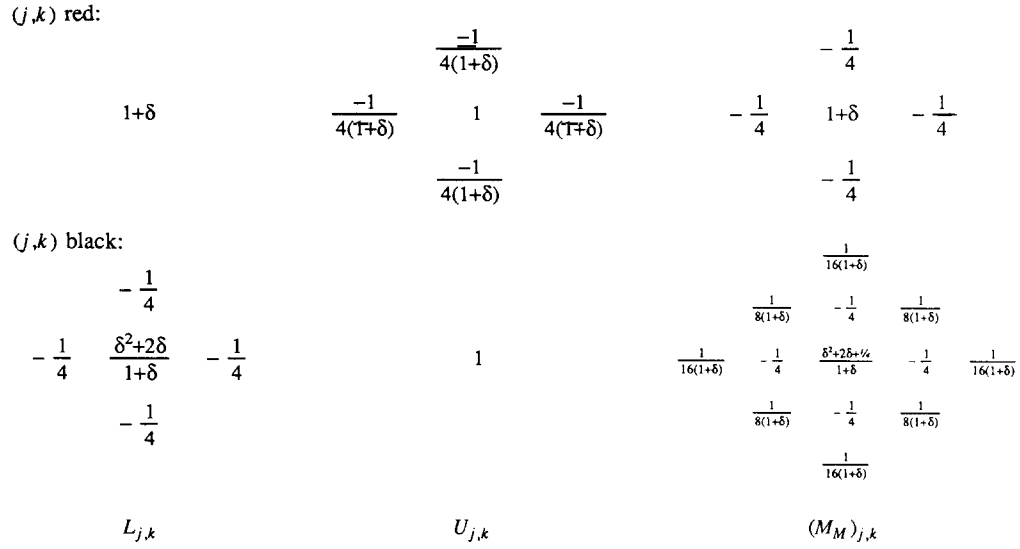
$$\frac{1}{4}\left(a + \frac{4}{a} - 4\right) = \delta, \tag{2.14}$$

where $\delta = 4^{-1}ch^2$ and $c > 0$, which gives

$$a = 2 + \tfrac{1}{2}ch^2 + \tfrac{1}{2}\sqrt{8ch^2 + c^2h^4}\,.$$



(a) natural ordering



(b) red/black ordering

Fig. 4. Stencil representations of local operators for the MILU preconditioner.

Similarly, for the red–black ordering, one can find that the MILU factorization leads to the local operators $L_{j,k}$ and $U_{j,k}$

$$L_{j,k} = \begin{cases} 1+\delta, & (j,\,k)\ \text{red}, \\ 1+\delta-\dfrac{1}{1+\delta}-\frac{1}{4}\left(E_x+E_x^{-1}+E_y+E_y^{-1}\right), & \\ & (j,\,k)\ \text{black}, \end{cases}$$

$$U_{j,k} = \begin{cases} 1-\dfrac{1}{4(1+\delta)}\left(E_x+E_x^{-1}+E_y+E_y^{-1}\right), & \\ & (j,\,k)\ \text{red}, \\ 1, & (j,\,k)\ \text{black}, \end{cases}$$

where $\delta = \tilde{c}h^2$. Thus, their product defines the asymptotical red–black MILU preconditioner. The stencil forms of local operators $L_{j,k}$, $U_{j,k}$ and $(M_{\mathrm{M}})_{j,k}$ are depicted in fig. 4.

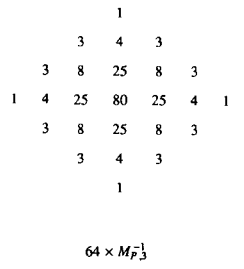


Fig. 5. Stencil representation of the local operator for the polynomial preconditioner.

2.2.3. Polynomial preconditioner

The polynomial preconditioner is based on the idea that the inverse of the Laplacian $A = I - B$ can be approximated by truncating its Taylor series expansion,

$$A^{-1} = (I - B)^{-1} \approx I + B + B^2 + \cdots + B^m = M_{P,m}^{-1}, \qquad (2.15)$$

where $B = (E_x + E_x^{-1} + E_y + E_y^{-1})/4$. Therefore, we can use $M_{P,m}^{-1}$ as the polynomial preconditioner. The stencil form of a typical case, $M_{P,3}^{-1}$, is shown in fig. 5. More generally, we may consider the polynomial preconditioner with weighting coefficients,

$$M_{P,m}^{-1} = \sum_{l=0}^{m} a_l B^l, \qquad (2.16)$$

so that coefficients a_l, $0 \le l \le m$, can be chosen to minimize the condition number of the polynomial preconditioned system $M_{P,m}^{-1}A$ for fixed m. The simplest case (2.15), where $a_l = 1$ for $0 \le l \le m$, is tested in this paper. Note that since the $(n+1)$th iteration u^{n+1} only depends on the nth iteration u^n for the polynomial preconditioned system, the computed value u^{n+1} will remain the same for any ordering. Therefore, with the polynomial preconditioner, preconditioning operations at all grid points can be performed in parallel. In actual implementation, the product $M_{P,m}^{-1}v$ can be computed by repeated application of B to v. Note also that in addition to the simple sitting $A = I - B$ considered above, one may use a general splitting scheme $A = E - F$, where E is easily invertible, so that B can be replaced by $E^{-1}F$ in (2.16).

3. Analysis of convergence rates

To understand the convergence rate of an iterative algorithm, one approach known as the matrix iterative analysis [14,26] is to use tools of numerical linear algebra. Another approach is to apply the algorithm to a simple model problem which consists of a constant-coefficient PDE on a regular domain with appropriate boundary conditions so that we are able to understand its behavior precisely by studying its effect on Fourier modes. The convergence behavior of this iterative algorithm for a more general class of problems, for which the model problem is typical, can be therefore estimated. The advantage of the matrix approach is its general applicability. It can be applied to PDEs of irregular geometries and spatially varying coefficients and discretized by nonuniform grids, as long as the corresponding iteration matrices satisfy some desired properties. In contrast, the Fourier approach can be rigorously applied to a small class of problems only. However, the Fourier approach has certain advantages. First, the matrix approach is in general much more complicated than the Fourier approach. Second, for simple problems, results obtained from the matrix approach are not as informative or as sharp as those obtained from the Fourier approach. For example, such results include the eigenvalue distribution of the preconditioned operator, the estimated of the relaxation parameter for the SOR method and the smoothing rate for multigrid methods. Third, for complicated problems, the predicted convergence behavior of iterative algorithms by using Fourier analysis is often consistent to what we observe in numerical experiments. Thus, the simplicity of the Fourier approach does not restrict its practical applicability.

Fourier or modified Fourier analysis has been used successfully to analyze numerical methods for elliptic PDEs for years. The model problem for 2nd-order self-adjoint elliptic PDEs is the Poisson equation on a square with Dirichlet boundary conditions. For the model Poisson problem, the SOR iteration was analyzed with Fourier-like basis functions by Frankel [12] and Young [25]. Brandt used Fourier analysis (or local mode analysis) to study the error smoothing property for

multigrid methods [7]. Stüben and Trottenberg performed a two-grid analysis to analyze both the error smoothing and the coarse-grid correction with Fourier basis functions [23]. Fourier analysis has also been applied to the analysis of the 5-point or 9-point SOR iteration with the natural or multicolor ordering [3,16–19], preconditioners for elliptic problems with the natural or red–black ordering [8,18], and problems arising from the domain decomposition context [9]. It is worthwhile to mention that Brandt used Fourier analysis to analyze the behavior of high frequency Fourier components which are not sensitive to the change of boundary conditions. However, by comparing our analysis and experiments, we observe an interesting fact, namely, Fourier analysis also gives a good estimate of the behavior of low frequency Fourier components which are supposed to be sensitive to different types of boundary conditions.

3.1. Natural ordering

3.1.1. Fourier analysis

Consider a 2D periodic sequence $u_{j,k}$ defined on a uniform square grid of spacing h with period $M \times M$, where $M = h^{-1}$. That is, $u_{j,k} = u_{j+M,k}$ and $u_{j,k} = u_{j,k+M}$ for arbitrary integer j and k. The sequence $u_{j,k}$ can be expanded as

$$u_{j,k} = \sum_{\xi=0}^{M-1} \sum_{\eta=0}^{M-1} \hat{u}_{\xi,\eta}\, e^{i2\pi(\xi j+\eta k)h}, \tag{3.1}$$

where the Fourier coefficient $\hat{u}_{\xi,\eta}$ gives the frequency domain representation of sequence $u_{j,k}$. It is easy to check that the complex sinusoid $e^{i2\pi(\xi j+\eta k)h}$ is the eigenvector for operators formed by the linear combination of shift operators with constant coefficients, and its eigenvalue can also be found easily. For example, from (2.6), we have

$$A_{j,k}\, e^{i2\pi(\xi j+\eta k)h} = \hat{A}(\xi, \eta)\, e^{i2\pi(\xi j+\eta k)h},$$
$$\hat{A}(\xi, \eta) = 1 - \tfrac{1}{2}[\cos(\xi 2\pi h) + \cos(\eta 2\pi h)]. \tag{3.2}$$

Similarly, the spectra of the SSOR, ILU and MILU preconditioned Laplacian derived in the previous section can be easily obtained by examining them in the frequency domain. From the matrix view point, to change from the space domain to the frequency domain is equivalent to a similarity transformation by which a sparse matrix is transformed into a diagonal matrix [8].

It is worthwhile to point out that to apply the above framework to the model Poisson problem, we have to adopt some simplifications. First, Dirichlet boundary conditions of the model Poisson problem has to be replaced by periodic boundary conditions. Second, preconditioners with spatially varying coefficients are replaced by asymptotical constant-coefficient preconditioners. For more detailed discussion on these two issues, we refer to ref. [8].

3.1.2. Condition number of the SSOR preconditioned Laplacian

According to the above framework, the eigenvalues of the SSOR preconditioner M_S for the eigenmode $e^{i2\pi(\xi j+\eta k)h}$ can be determined as

$$(M_S)_{j,k}\, e^{i2\pi(\xi j+\eta k)h} = \hat{M}_S(\xi, \eta)\, e^{i2\pi(\xi j+\eta k)h},$$
$$\hat{M}_S(\xi, \eta) = 1 - \frac{\omega}{2}[\cos(\xi 2\pi h) + \cos(\eta 2\pi h)] + \frac{\omega^2}{8}\{1 + \cos[(\xi - \eta)2\pi h]\}. \tag{3.3}$$

Following (3.2) and (3.3), we find that the spectrum of the SSOR preconditioned Laplacian $M_S^{-1}A$ can be simply computed as

$$\lambda_{\xi,\eta} = \hat{M}_S^{-1}(\xi, \eta)\hat{A}(\xi, \eta) = \{1 - \tfrac{1}{2}[\cos(\xi 2\pi h) + \cos(\eta 2\pi h)]\} \Big/ \Big\{1 - \frac{\omega}{2}[\cos(\xi 2\pi h) + \cos(\eta^2 \pi h)] + \frac{\omega^2}{8}\{1 + \cos[(\xi - \eta)2\pi h]\}\Big\}.$$

The condition number of the SSOR preconditioned Laplacian is defined to be the ratio of its largest and smallest positive eigenvalues. Although the relaxation parameter ω might be chosen to minimize it, a direct calculation of its optimal value is difficult. On the other hand, the optimal parameter which minimizes the condition number with respect to a particular set of wavenumbers,

i.e. $\xi + \eta = M$ and $1 \le \xi \le M-1$ is more tractable and can be found to be $2/[1 + 2\sin(\pi h)]$. It can be argued that this value is truly the optimal one with respect to all wavenumbers as well and, with this optimal value the condition number of the SSOR preconditioned Laplacian is $\mathcal{O}(h^{-1})$ [8].

3.2. Red–black ordering

Due to the red–black ordering, the resulting system of iteration equations is not spatially homogeneous but is *periodic* with respect to grid points. Consequently, the Fourier modes are not eigenfunctions for the multicolor system, and therefore a straightforward Fourier analysis does not apply. By exploiting the periodic property, we reformulate the conventional Fourier analysis as a two-color Fourier analysis.

3.2.1. Two-color Fourier analysis

Consider a 2D sequence $u_{j,k}$ defined on a uniform square grid of spacing h with zero boundary values, i.e. $u_{j,k} = 0$ if $j, k = 0$ or M where $M = h^{-1}$ is even. We can expand it with Fourier series as

$$u_{j,k} = \sum_{\xi=1}^{M-1} \sum_{\eta=1}^{M-1} \hat{u}_{\xi,\eta} \sin(\xi\pi jh) \sin(\eta\pi kh). \quad (3.4)$$

The function $u_{j,k}$ at the red and black points defines two sequences: the red sequence $u_{\mathrm{r}j,k}$ and the black sequence $u_{\mathrm{b}j,k}$. They can be expanded in Fourier series, respectively, as

$$u_{\mathrm{r}j,k} = \sum_{(\xi,\eta)\in K_{\mathrm{r}}} \hat{u}_{\mathrm{r},\xi,\eta} \sin(\xi\pi jh) \sin(\eta\pi kh),$$

$j + k$ even, (3.5a)

$$u_{\mathrm{b}j,k} = \sum_{(\xi,\eta)\in K_{\mathrm{b}}} \hat{u}_{\mathrm{b},\xi,\eta} \sin(\xi\pi jh) \sin(\eta\pi kh),$$

$j + k$ odd, (3.5b)

where

$$K_{\mathrm{b}} = \{(\xi, \eta) \in I^2 : \xi + \eta \le M-1,\ \xi, \eta \ge 1 \quad \text{or} \quad \eta = M - \xi,\ 1 \le \xi \le M/2 - 1\},$$

and $K_{\mathrm{r}} = K_{\mathrm{b}} \cup \{(M/2, M/2)\}$.

It is straightforward to check that the Fourier coefficients $\hat{u}_{\xi,\eta}$, $\hat{u}_{M-\xi,M-\eta}$ in (3.4) and $\hat{u}_{\mathrm{r},\xi,\eta}$, $\hat{u}_{\mathrm{b},\xi,\eta}$ in (3.5) are related via

$$\begin{pmatrix} \hat{u}_{\mathrm{r},\xi,\eta} \\ \hat{u}_{\mathrm{b},\xi,\eta} \end{pmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} \hat{u}_{\xi,\eta} \\ \hat{u}_{M-\xi,M-\eta} \end{pmatrix}, \quad (\xi, \eta) \in K_{\mathrm{b}}, \quad (3.6a)$$

$$\hat{u}_{\mathrm{r},\xi,\eta} = \hat{u}_{\xi,\eta}, \quad (\xi, \eta) = (M/2, M/2). \quad (3.6b)$$

We can interpret (3.6) as follows. Through the red–black decomposition (3.5), the component $(M - \xi, M - \eta)$ in the high frequency region is folded into the component (ξ, η) in the low frequency region so that there exist two computational waves in the low frequency region. Note also that K_{r} and K_{b} differs only by a single element $(M/2, M/2)$ and, therefore, at the frequency $(M/2, M/2)$ we have only a scalar $\hat{u}_{\mathrm{r},M/2,M/2}$, which is considered as the degenerate case.

Now, let us apply the two-color Fourier analysis to the model Poisson problem with the red–black ordering. Without loss of generality, we concentrate on the case where $u_{j,k}$ is zero on boundaries, since a nonzero $u_{j,k}$ on the boundary can always be moved to the right-hand side and treated as part of the forcing function. In addition, since the forcing term $f_{j,k}$ with $j, k = 0$ or M does not appear, it can be viewed as zero. Consequently, the red–black Fourier series expansion (3.5) for both $u_{j,k}$ and $f_{j,k}$ are well defined.

By substituting (3.5) into (2.6) and relating the Fourier coefficients of red and black waves, we can transform (2.6) from the space domain into the red–black Fourier domain. It is a block diagonal matrix equation, in which the equation for a nondegenerate frequency (ξ, η) can be written as

$$\hat{A}(\xi, \eta) \begin{pmatrix} \hat{u}_{\mathrm{r},\xi,\eta} \\ \hat{u}_{\mathrm{b},\xi,\eta} \end{pmatrix} = -\frac{h^2}{4} \begin{pmatrix} \hat{f}_{\mathrm{r},\xi,\eta} \\ \hat{f}_{\mathrm{b},\xi,\eta} \end{pmatrix}, \quad \hat{A}(\xi, \eta) = \begin{bmatrix} 1 & -\alpha_{\xi,\eta} \\ -\alpha_{\xi,\eta} & 1 \end{bmatrix}, \quad (3.7)$$

where $\alpha_{\xi,\eta} = (\cos(\xi\pi h) + \cos(\eta\pi h))/2$ is the Fourier transform of the space domain operator $(E_x + E_x^{-1} + E_y + E_y^{-1})/4$. Since $(\xi, \eta) \in K_{\mathrm{b}}$, $0 \le \alpha_{\xi,\eta} <$

1. With minor modification, we can also treat the degenerate frequency $(M/2, M/2)$.

3.2.2. Condition number of the SSOR preconditioned Laplacian

By using the two-color Fourier analysis, we can transform (2.11) to the frequency domain

$$\begin{aligned}\hat{M}_S(\xi,\eta) &= \left[\hat{D}(\xi,\eta)-\omega\hat{L}(\xi,\eta)\right]\hat{D}^{-1}(\xi,\eta)\\ &\quad\times\left[\hat{D}(\xi,\eta)-\omega\hat{U}(\xi,\eta)\right]\\ &= \begin{bmatrix}1 & 0\\ -\omega\alpha_{\xi,\eta} & 1\end{bmatrix}\begin{bmatrix}1 & -\omega\alpha_{\xi,\eta}\\ 0 & 1\end{bmatrix}\\ &= \begin{bmatrix}1 & -\omega\alpha_{\xi,\eta}\\ -\omega\alpha_{\xi,\eta} & 1+\omega^2\alpha_{\xi,\eta}^2\end{bmatrix},\end{aligned}\tag{3.8}$$

where $\hat{D}(\xi,\eta)$ is a 2×2 identity matrix. From (3.7) and (3.8), we find that the SSOR preconditioned operator $M_S^{-1}A$ has the spectral representation

$$\hat{M}_S^{-1}(\xi,\eta)\hat{A}(\xi,\eta) = \begin{bmatrix}1-\omega\alpha_{\xi,\eta}^2+\omega^2\alpha_{\xi,\eta}^2 & -\alpha_{\xi,\eta}+\omega\alpha_{\xi,\eta}-\omega^2\alpha_{\xi,\eta}^3\\ -\alpha_{\xi,\eta}+\omega\alpha_{\xi,\eta} & 1-\omega\alpha_{\xi,\eta}^2\end{bmatrix},$$

which has two eigenvalues

$$\begin{aligned}\lambda_{\xi,\eta,\pm} &= 1-\tfrac{1}{2}\alpha_{\xi,\eta}^2\omega(2-\omega)\pm\tfrac{1}{2}\alpha_{\xi,\eta}\\ &\quad\times\left[\alpha_{\xi,\eta}^2\omega^2(2-\omega)^2-4\omega(2-\omega)+4\right]^{1/2}.\end{aligned}$$

The condition number $\kappa(M_S^{-1}A)$ is determined by the ratio of $\max|\lambda_{\xi,\eta,+}|$ and $\min|\lambda_{\xi,\eta,-}|$, which can be minimized by choosing an appropriate relaxation parameter ω. It turns out that the optimal relaxation parameter is 1, and with this parameter the condition number of the SSOR preconditioned Laplacian is [18]

$$\begin{aligned}\kappa\left(M_S^{-1}A\right) &= \left[1-\cos^2(\pi h)\right]^{-1}\\ &\approx \pi^{-2}h^{-2}=\mathcal{O}(h^{-2}).\end{aligned}$$

3.3. Summary of convergence rates

In the previous section, the analysis of the SSOR preconditioner with the natural and red–black orderings is used as example to demonstrate the fact that Fourier analysis provides a simple and effective tool to obtain the eigenvalue information as well as the condition number of the preconditioned Laplacian. Similarly, Fourier analysis can also be applied to ILU, MILU and polynomial preconditioners [8,10,18]. We list the condition number of the preconditioned system with various preconditioners in table 1, where h is the grid spacing. Note that the total number N of grid points is related to h via $N\approx h^{-2}$.

Table 1
Comparison of condition numbers

Preconditioner	Natural	R/B
none(Laplacian)	$\mathcal{O}(h^{-2})$	$\mathcal{O}(h^{-2})$
SSOR	$\mathcal{O}(h^{-1})$	$\mathcal{O}(h^{-2})$
ILU	$\mathcal{O}(h^{-2})$	$\mathcal{O}(h^{-2})$
MILU	$\mathcal{O}(h^{-1})$	$\mathcal{O}(h^{-2})$
polynomial	$\mathcal{O}(h^{-2})$	$\mathcal{O}(h^{-2})$

For details of deriving condition numbers and their corresponding constants for the SSOR, ILU and MILU preconditioners with the natural and red–black orderings, readers are referred to refs. [8,18], respectively. The polynomial preconditioner has been studied in ref. [10]. Although the red–black ordered preconditioners do not improve the order of the condition number with respect to the original Laplacian, they do improve the constant by a certain factor [18].

The convergence rates of stationary and preconditioned iterative algorithms are compared in table 2, where convergence rates are expressed in terms of numbers of iteration required for the reduction of initial residual by a constant factor.

The first three entries above indicate the convergence rates of stationary iterative algorithms

Table 2
Comparison of convergence rates

Iterative algorithm	Convergence rate
basic(Jacobi, GS)	$\mathcal{O}(h^{-2})$
SOR(natural)	$\mathcal{O}(h^{-1})$
SOR(R/B)	$\mathcal{O}(h^{-1})$
CG(basic)	$\mathcal{O}(h^{-1})$
PCG(MILU; natural)	$\mathcal{O}(h^{-1/2})$

with or without acceleration. Note that the SOR acceleration speeds up the basic Jacobi and Gauss–Seidel iterations and gives the same order of convergence rate for both the natural and red–black orderings [1,16]. In the last two entries, we consider the CG acceleration. The basic CG refers to the application of CG to the Laplacian without preconditioning. It is well known that the convergence rate of a preconditioned iterative method depends on the condition number as well as the distribution of the eigenvalues of the preconditioned system [6]. In our context, the convergence rate is in the order of the square root of the condition number so that the quantities listed in table 1 can be converted into quantities of convergence rates straightforwardly. In table 2, only two of them (the basic CG and the PCG with that naturally ordered MILU preconditioner) are listed for comparison.

4. Implementation and performance on the Connection Machine

In the previous section, we analyzed the rate of convergence of different iterative algorithms. However, unlike the case of sequential machines, this analysis alone is not sufficient to predict the exact performance of each algorithm on parallel machine, since the ease of parallelization has to be taken into account as well. The performance of parallel preconditioners on the Connection Machine will be compared below.

4.1. The Connection Machine

The Connection Machine (CM) [15,24] is a massively parallel architecture consisting of 65536 single bit processors. The CM system consists of two parts – a front end machine and a hypercube of 64 K processors. The front end computer, currently supported by the Symbolics 3600 or the VAX 8800 machines, provides instruction sequencing, program development, networking and low speed I/O. The CM program contains two types of statements – those operating on single data items and those operating on a whole data set at once. The single-data-item instruction is executed in the front end, whereas the large-data-set instruction is executed in the CM hypercube to increase parallelism.

The CM hypercube consists of 4096 (2^{12}) chips, each with 16 processors and a router, to form a 12 dimensional hypercube. The router is a communication processor that allows any on-chip processor to communicate with other processors in the system. In addition to the router network, there is a separate communication facility called the NEWS grid. That is, each processor is wired to its four nearest neighbors in a 2D regular grid. Machine instructions allow message to be sent, a bit at a time, to the processor to the north, south, east or west on this grid. Communication on this NEWS grid is very fast compared to the router communication and is encouraged for short distance communication between processors. Moreover, on the CM-2, there are 8 Kbytes of random access memory associated with each physical processor; and also the Weitek floating point processors are incorporated into the system, with one Weitek chip shared between two 16-processor CM chips.

An important feature of the CM system is its support for virtual processors. A run-time configuration command *cold-boot* may be used to specify how each physical processor simulates a small rectangular array of virtual processors – except, of course, for the case that such a processor appears to be correspondingly slower and has only a fraction of the memory of a physical processor.

The CM supports two software environments: Common LISP and C. A standard extension for each language is provided so that it supplies the complete interface to the hypercube. The extensions are *LISP and C*, respectively. The basic idea for the extension is that scalar variables reside on the front end, while vector of parallel variables are distributed over the cube.

The CM used in our experiment is a 16 K-node CM-2 without the Weitek floating point accelerator running at a clock frequency of about 6.47 MHz. The front end computer used is the Symbolics 3600 and the language used for program development is *LISP.

4.2. Implementation and results

On iteration of the PCG (Preconditioned Conjugate Gradient) method can be written as follows [13]:

Solve $Mz_k = r_k$ for z_k
$\beta_{k+1} = (z_k, r_k)/(z_{k-1}, r_{k-1})$,
$p_{k+1} = z_k + \beta_{k=1} p_k$,
$\alpha_{k+1} = (z_k, r_k)/(p_{k+1}, Ap_{k+1})$,
$x_{k+1} = x_k + \alpha_{k+1} p_{k+1}$,
$r_{k+1} = r_k - \alpha_{k+1} Ap_{k+1}$.

From above, we see that it requires two inner products (or three, if the calculation of the residue to check convergence is included), three multiply-and-add operations in the form of $ax + b$ where x ad b are vectors and a is a scalar, one matrix–vector product calculation, plus the operations required for preconditioning. The multiply-and-add operation as well as the matrix–vector product can be performed in parallel on all the unknowns (in this case, the unknowns are grid points on a 2D grid). On the Connection Machine these unknowns are mapped onto processors in a one-to-one manner (2D grid mapping onto 2D grid), so that these two operations can be completed in constant time (independent of the number of unknowns). The inner product operation, however, cannot be done in constant time. The hypercube configuration of the CM allows multiplication of two elements in parallel and accumulation of the partial sums in the form of a binary tree. The execution time of this operation is thus $\mathcal{O}(\log N)$ where N is the total number of grid points. In summary, without considering the preconditioning, the execution time of each iteration is $\mathcal{O}(\log N)$.

For preconditioning, one needs to solve a system of the form $Mz = r$ where M is a $N \times N$ matrix and z and r are vectors of length N. Suppose the preconditioner can be factorized into the product of a low and upper triangular matrices L and U. The solution to the system $Mz = r$ consists of a forward solver (L^{-1}) and a backward solver (U^{-1}). For the natural ordering case, due to data dependency, the forward solver cannot be performed in parallel on all unknowns, but instead the unknowns can only be operated on sequentially ion a diagonal fashion, starting from lower-left corner (which corresponds to unknown #1) and proceeding diagonally to the upper-right corner (unknown #N). Thus, the forward solver can be at best solved in $\mathcal{O}(N^{1/2})$ time with maximum parallelism $\mathcal{O}(N^{1/2})$. The backward solver behaves similarly except in an opposite direction, i.e. starting from the unknown #N at the upper-right corner to the unknown #1 at the lower-left corner.

The limited parallelism offered by the natural ordering results in low utilization of massively parallel machines such as the CM. To achieve a higher degree of parallelism, we need a different ordering scheme for the preconditioning, for example, the red–black ordering. With the red–black ordering, unknowns at red points do not depend on unknowns on other red points and the same is true for black unknowns, so that the preconditioning procedure can be performed simultaneously at all red points and then all black points and takes only constant time. To increase the processor utilization on the CM, it might be desirable to map one red and one black unknown onto one processor. However, to simplify the program development, this mapping was not yet implemented in our codes.

In spite of its attractive feature of offering higher degree of parallelism and, thus, lower execution time, the red–black ordering has a drawback, namely, it affects the rates of convergence in a negative way as analyzed in the previous section and indicated in table 1. To seek a suitable method for a particular type of parallel machine, we have to consider the tradeoff between low execution time and fast convergence rate. As for the CM, the red–black ordering seems to be superior for the problem that we have tested, which will be demonstrated by the results of our experiments.

We solve the 2D Poisson equation with Dirichlet boundary condition on the Connection Machine using PCG with various preconditioners:

1. no preconditioning (CG),
2. ILU with the natural ordering,
3. MILU with the natural ordering,
4. SSOR with the natural ordering,
5. ILU with the red–black ordering,

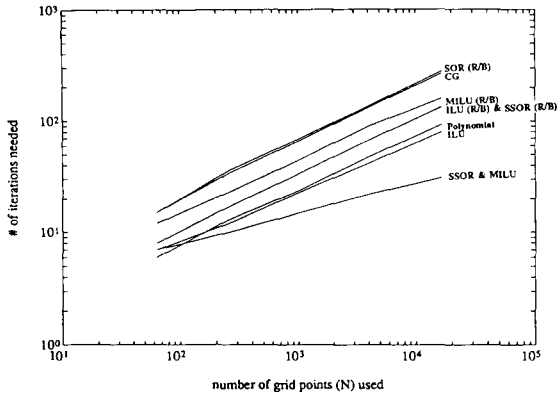


Fig. 6. Convergence rates for SOR (R/B) and PCG methods.

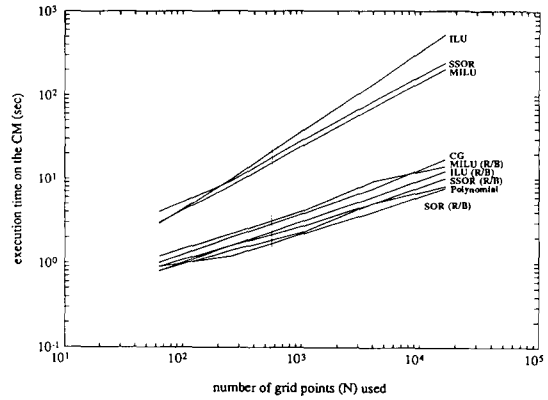


Fig. 7. CM execution time for SOR(R/B) and PCG methods.

6. MILU with the red–black ordering,
7. SSOR with the red–black ordering,
8. Polynomial preconditioner using 4 terms ($M_{P,3}^{-1}$)

For comparison, the SOR method with the red–black ordering is also tested for the same model problem. The convergence rates are plotted as function of N, the number of grid points, in fig. 6. Their convergence rates expressed in $\mathcal{O}(N^{\beta})$, can be obtained by examining the slope of the corresponding plot. The results are summarized in table 3. We see from table 3 that the orders of convergence rate obtained from our experiments data confirm those obtained from analysis. As far as the convergence rate is concerned, it is clear that the PCG method with naturally ordered SSOR and MILU preconditioners ($\beta = 0.25$) behaves better than other tested methods ($\beta = 0.5$). Although most solution methods plotted in fig. 6 have the same order of convergence rates ($\beta = 0.5$), they have different proportional constants. The experiment shows that the red–black ordered SOR method has the largest constant while the naturally ordered ILU has the smallest one.

Table 3
Comparison of convergence rates

Preconditioner	Analysis	CM results
basic CG	$\mathcal{O}(N^{0.50})$	$\mathcal{O}(N^{0.49})$
ILU(natural)	$\mathcal{O}(N^{0.50})$	$\mathcal{O}(N^{0.45})$
MILU(natural)	$\mathcal{O}(N^{0.25})$	$\mathcal{O}(N^{0.27})$
SSOR(natural)	$\mathcal{O}(N^{0.25})$	$\mathcal{O}(N^{0.27})$
ILU(R/B)	$\mathcal{O}(N^{0.50})$	$\mathcal{O}(N^{0.49})$
MILU(R/B)	$\mathcal{O}(N^{0.50})$	$\mathcal{O}(N^{0.50})$
SSOR(R/B)	$\mathcal{O}(N^{0.50})$	$\mathcal{O}(N^{0.49})$
polynomial	$\mathcal{O}(N^{0.50})$	$\mathcal{O}(N^{0.50})$

The CM execution time for different preconditioners as function of the number of grid points is plotted in fig. 7. Based on this figure, we find that although the convergence rates of MILU and SSOR with the natural ordering are much better than those of other preconditioners, their execution time are far inferior than those of preconditioners with the red–black ordering. Thus, the fast convergence rate does not guarantee low execution time. The reason is that naturally-ordered preconditioners have to be performed more sequentially so that much more execution time is needed. In our experiments, a preconditioner with the red–black ordering always takes less execution time than its counterpart with the natural ordering. An interesting observation is that the red–black SOR method has the slowest convergence rate but the shortest execution time. This is partly due to the fact that costly inner-product operations in the CG algorithm can be avoided in the SOR method for the model Poisson problem. This suggests that the local relaxation method [16] should be also attractive for the CM for PDEs with spatially varying coefficients. Note that the ease of of parallelization alone does not guarantee low execution time since it is observed that even though the Jacobi method offers the highest degree of parallelism, it is much slower than many other methods for its slow convergence rate.

We conclude from our CM experiments that to search for a better preconditioner on parallel machines, tradeoffs have to be carefully considered between the fast convergence rate and the ease of parallelization of the preconditioner [22]. A preconditioner showing promises of better performance in both the convergence rate and the ease of parallelization is preconditioning with the hierarchical basis [2,27]. Its analysis and performance in the CM machine is under our current study.

Acknowledgement

The authors would like to thank ISI (Information Sciences Institute), University of Southern California, for providing the CM computing facility for the experiments reported in section 4.

References

[1] L.M. Adams and H.F. Jordan, SIAM J. Sci. Stat. 7 (2) (1986).
[2] L.M. Adams and E.G. Ong, in: Parallel Computations and Their Impact on Mechanics, ed. A.K. Noor (American Society of Mechanical Engineers, New York, NY, 1987) p. 171.
[3] L.M. Adams, R.J. LeVeque and D.M. Young, SIAM J. Num. Anal. (to appear).
[4] C.C. Ashcraft and R.G. Grimes, SIAM J. Sci. Stat. Comput. (1988) 122.
[5] O. Axelsson, BIT 13 (1972) 443.
[6] O. Axelsson, BIT 25 (1985) 166.
[7] A. Brandt, Math. Comput. 31 (1977) 333.
[8] T.F. Chan and H.C. Elman, CAM Report 87-04, Department of Mathematics, UCLA (February 1988).
[9] T.F. Chan and D.C. Resasco, in: First Intern. Symp. on Domain Decomposition Methods for Partial Differential Equations (SIAM, Philadelphia, PA, 1988) p. 217.
[10] J. Donato, Term paper, Department of Mathematics, UCLA (1988).
[11] T. Dupont, R.P. Kendall and H.H. Rachford, Jr., SIAM J. Num. Anal. 5 (1968) 559.
[12] S.P. Frankel, Math. Tables Aids Comput. 4 (1950) 65.
[13] G.H. Golub and C. Van Loan, Matrix Computation (John Hopkins Univ. Press, Baltimore, MD, 1983).
[14] L.A. Hageman and D.M. Young, Applied Iterative Methods (Academic Press, New York, NY, 1981).
[15] W.D. Hillis, The Connection Machine (MIT Press, Cambridge, MA, 1985).
[16] C.-C.J. Kuo, B.C. Levy and B.R. Musicus, SIAM J. Sci. Stat. Comput. 8 (1987) 550.
[17] C.-C.J. Kuo and B.C. Levy, SIAM J. Num. Anal. (to appear).
[18] C.-C.J. Kuo and T.F. Chan, CAM Report, Department of Mathematics, UCLA (1988).
[19] R.J. LeVeque and L.N. Trefethen, Numerical Analysis Report 86-6, Department of Mathematics, MIT, Cambridge, MA (September 1986).
[20] J.A. Meijerink and H.A. van der Vorst, Math. Comput. 31 (1977) 148.
[21] J.M. Ortega and R.G. Voigt, SIAM Rev. 27 (1985) 149.
[22] Y. Saad and M.H. Schultz, Research Report YALEU/DCS/RR-425 (October 1985).
[23] K. Stüben and U. Trottenberg, in: Multigrid Methods, eds. W. Hackbusch and U. Trottenberg (Springer-Verlag, New York, NY, 1982) p. 1.
[24] L.W. Tucker and G.G. Robertson, Computer 21 (1988) 26.
[25] D.M. Young, Doctoral Thesis, Harvard University (1950).
[26] D.M. Young, Iterative Solution of Large Linear Systems (Academic Press, New York, NY, 1971).
[27] H. Yserentant, Num. Math. 49 (1986) 379.