Parallelizable Preconditioned Conjugate Gradient Methods for the Cray Y-MP and the TMC CM-2

William H. Holter
I. M. Navon
Thomas C. Oppe

Supercomputer Computations Research Institute
Florida State University
Tallahassee, FL 32306-4052

June 28, 2002

Submitted to *International Journal of High Speed Computing*

Abstract

This paper presents the results of applying a number of vectorizable and parallelizable preconditioned conjugate gradient methods to the numerical solution of the diffusion equation governing the flow of groundwater in a confined two-dimensional rectangular aquifer. Four different sample problems are formulated having coefficients (transmissivities) ranging from continuous and isotropic to sharply discontinuous and anisotropic. When discretized using a second-order finite difference approximation, the resulting linear systems have symmetric and positive-definite matrices. The preconditioners applied to these problems include the Jacobi, line Jacobi, incomplete LU (ILU) and modified incomplete LU (MILU) decompositions, symmetric Gauss-Seidel, least squares polynomial, and a new alternating direction preconditioner. The suitability of each of these preconditioning methods for the Y-MP and Connection Machine 2 (CM-2) parallel computers is investigated. The iteration count and the timing are given for each method and for several problem sizes. It is shown that the ILU and MILU methods are effective on a single-processor Y-MP when vectorized using a wavefront strategy. For the CM-2, the least squares polynomial and the alternating direction preconditioners are the most effective of the methods used in the study.

1 Introduction

This paper provides a comparison of the performance characteristics of several preconditioned conjugate gradient (PCG) methods as implemented on the Cray Y-MP and the Thinking

Machines Corporation CM-2. These methods are used for solving four example problems arising from the modeling of groundwater flow via discretization of the 2-D diffusion equation. Varying properties of the groundwater flow model, such as isotropic vs. anisotropic and continuous vs. discontinuous transmissivities, give rise to large sparse systems of linear equations whose matrices can be very ill-conditioned.

For this study, traditional preconditioners that vectorize and parallelize well but have poor convergence rates, including Jacobi, line Jacobi, least squares polynomial, and symmetric Gauss-Seidel with red-black ordering are implemented on both computers. Preconditioning methods that parallelize poorly but have good convergence rates, such as ILU(k) and MILU(k) with natural ordering, are implemented on the Y-MP employing various optimization strategies. Finally, a new preconditioner resembling the Alternating Direction Implicit (ADI) method is developed in order to achieve increased parallelism while maintaining the good convergence rate enjoyed by the MILU preconditioner.

The paper is organized as follows. Section 2 contains a description of each of the four test problems in terms of their respective diffusion equation parameters and boundary conditions; Sections 3 and 4 contain a description of the PCG algorithms used in the study; Section 5 presents an evaluation of the numerical results; Section 6 presents some timing models for the methods based upon the empirical data; and, lastly, Section 7 presents some concluding remarks.

2 The Test Problems

The flow of groundwater in a confined aquifer may be described by the 2-D diffusion equation. Assuming that the principal components of transmissivity lie along the Cartesian coordinate axes, the equation may be written as:

$$s\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(a\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(b\frac{\partial u}{\partial y}\right) + f, \tag{1}$$

where s is the storage coefficient of the porous media, u is the pressure head (height of the water table) to be determined, a and b are (non-negative) transmissivities in the x and y directions, respectively, and f is a source/sink term that takes into account recharging/discharging wells. Sources and sinks are approximated by delta functions with strengths equal to their volumetric flux.

Each of the test problems is defined on the unit square whose sides are assigned appropriate boundary conditions. A steady-state solution is sought for each problem, in which case Eq. (1) simplifies to:

$$-\frac{\partial}{\partial x}\left(a\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(b\frac{\partial u}{\partial y}\right) = f. \tag{2}$$

The boundary conditions considered are no-flow (homogeneous Neumann)

$$\frac{\partial u}{\partial \eta} = 0,$$

where η is a unit vector normal to the boundary of the aquifer, and constant head (Dirichlet)

$$u = \text{constant}.$$

Discretizing on a uniform node-centered grid with distance between adjacent nodes $\Delta x = \Delta y = h$, Eq. (2) becomes:

$$-\alpha_{i-1,j}u_{i-1,j} - \alpha_{i,j}u_{i+1,j} - \beta_{i,j-1}u_{i,j-1} - \beta_{i,j}u_{i,j+1} + \sigma_{i,j}u_{i,j} = h^2 f_{i,j} = g_{i,j}, \tag{3}$$

where $i = 1, 2, \ldots, n_x$ and $j = 1, 2, \ldots, n_y$; with n_x and n_y denoting, respectively, the number of grid points in the x and y directions; and

$$\begin{aligned}
\alpha_{i,j} &= \delta\left(a_{i,j}; a_{i+1,j}\right) \\
\beta_{i,j} &= \delta\left(b_{i,j}; b_{i,j+1}\right) \\
\delta(c; d) &= \begin{cases} 2cd/(c+d), \text{the harmonic mean,} & \text{if } c+d \neq 0 \\ 0 & \text{otherwise} \end{cases} \\
\sigma_{i,j} &= \begin{cases} \alpha_{i-1,j} + \alpha_{i,j} + \beta_{i,j-1} + \beta_{i,j} & \text{if the sum} \neq 0 \\ 1 & \text{otherwise} \end{cases}
\end{aligned} \tag{4}$$

The α and β coefficients, located halfway between nodes, are given respectively by the harmonic means of the a and b coefficients at adjacent nodes in order to ensure continuity of fluid flux. The definitions given in Eq. (4) permit the inclusion of totally impervious regions in the sample problems to be considered. Also, to accommodate no-flow boundary conditions, appropriate α or β coefficients adjacent to the boundary are set equal to zero; this amounts to employing an approximation to the normal derivative that is of first order in h.

The stencil for the 5-point operator, implicit in Eq. (3), is depicted in Figure 1. Note that $\sigma_{i,j}$ and $g_{i,j}$ are collocated with $u_{i,j}$ at (x_i, y_j).

For an arbitrary rectangular uniform grid, the resulting system of equations for the pressure heads, u, may be written as

$$Au = b,$$

where A is a symmetric positive definite (SPD) matrix (or a symmetric positive *semi*-definite (SPSD) matrix if no-flow boundaries are imposed everywhere), and the vector b is a function of g and the boundary conditions. The matrix A has a pentadiagonal structure and is weakly diagonally dominant [19].

The four problems considered in this paper are described below.

2.1 EXPNA

This problem has continuous and isotropic transmissivities, a and b, and fully Dirichlet boundary conditions. The transmissivities are

$$a(x, y) = b(x, y) = 100(x + y).$$

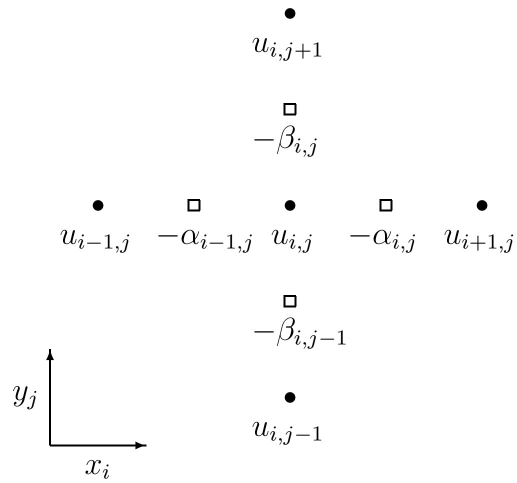

Figure 1: 5-Point Stencil

The forcing function f and the boundary conditions are chosen so that the analytic solution to Eq. (2) is

$$u(x, y) = \cos(4\pi x)\cos(4\pi y).$$

2.2 EXPNC

This problem has continuous and anisotropic transmissivities, a and b, and fully Dirichlet boundary conditions. The transmissivities are

$$\begin{aligned} a(x, y) &= 100x, \\ b(x, y) &= 100(1 - y). \end{aligned}$$

The forcing function f and the boundary conditions are chosen so that the analytic solution to Eq. (2) is the same as that given for EXPNA.

2.3 EXP10G

This problem has discontinuous and isotropic transmissivities, a and b, and fully Dirichlet boundary conditions. The values of the transmissivities range from 0 (impervious) to $100,000$ (highly permeable) and are piecewise constant in certain subdomains as given in Figure 2. In this figure, filled circles with negative g values indicate discharging wells and hollow circles with positive g values indicate recharging wells.

2.4 EXP6G

This problem has discontinuous and anisotropic transmissivities, a and b, and mixed Neumann and Dirichlet boundary conditions. The values of the transmissivities range from 0 to $100,000$ and are piecewise constant in certain subdomains as given in Figure 3. In this figure

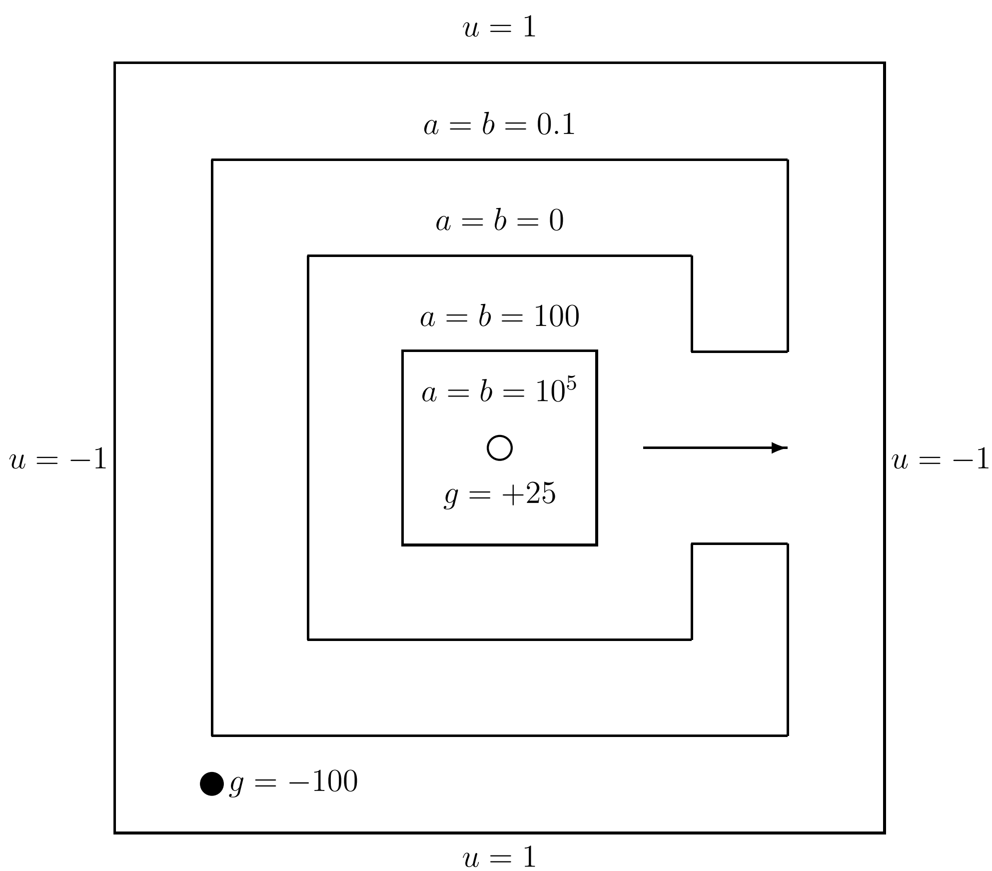

Figure 2: EXP10G

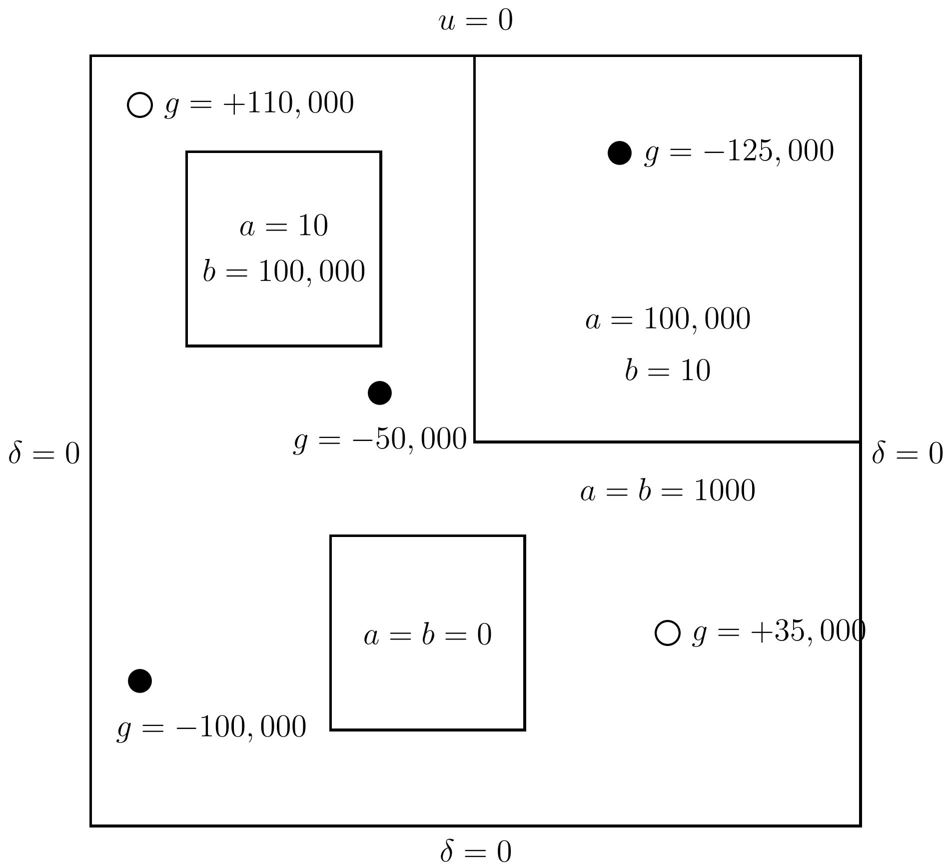

Figure 3: EXP6G

$\delta = 0$ indicates a no-flow boundary condition while $u = 0$ indicates a Dirichlet boundary condition.

3 The PCG Method

Denote the linear system to be solved as

$$Au = b$$

where A is a symmetric and positive definite matrix. Let Q be a symmetric and positive definite matrix, called the *preconditioning matrix*, such that

1. Q approximates A to a greater or lesser degree, and
2. the solution to $Q\delta = r$ given r is "easy" to compute.

The Preconditioned Conjugate Gradient (PCG) algorithm used in this study is given by:

For $n = 0, 1, 2, \ldots$ until convergence do

$$\begin{aligned}
r^{(n)} &= \begin{cases} b - Au^{(0)} & n = 0 \\ r^{(n-1)} - \alpha_{n-1} z^{(n-1)} & n \geq 1 \end{cases} \\
& \quad \text{If } \|r^{(n)}\|_2 / \|b\|_2 < \zeta, \text{ exit} \\
\delta^{(n)} &= Q^{-1} r^{(n)} \\
\gamma_n &= \langle \delta^{(n)}, r^{(n)} \rangle \\
\beta_n &= \gamma_n / \gamma_{n-1} \qquad\qquad (\beta_0 = 0) \\
p^{(n)} &= \delta^{(n)} + \beta_n p^{(n-1)} \qquad (p^{(-1)} = 0) \\
z^{(n)} &= Ap^{(n)} \\
\alpha_n &= \gamma_n / \langle p^{(n)}, z^{(n)} \rangle \\
u^{(n+1)} &= u^{(n)} + \alpha_n p^{(n)}
\end{aligned}$$

where $r^{(n)}$ are the residuals, $\delta^{(n)}$ are the pseudo-residuals, $p^{(n)}$ are the direction vectors, $u^{(n)}$ are the iterates, $\langle \cdot, \cdot \rangle$ denotes the inner product, $\| \cdot \|_2$ denotes the 2-norm, and ζ is the convergence criterion. Note that the speed with which the algorithm converges is strongly dependent on the choice of Q. In general, it is desirable that the condition number of $Q^{-1}A$ be smaller than that of A.

Many components of the PCG algorithm can be computed with optimized assembly language routines on both the Y-MP and the CM-2. For the Y-MP, the Basic Linear Algebra Subprograms (BLAS)[13] routines `sdot`, `snrm2`, and `saxpy` are used to compute inner product, 2-norm, and vector update operations, respectively. For the CM-2, the Connection Machine Scientific Software Library (CMSSL) [3] routines

1. `gbl_gen_inner_product_noadd`,
2. `gbl_gen_2_norm`, and

3. `grid_sparse_matrix_vector_mult`

are used to compute inner product, 2-norm, and matrix-vector product operations, respectively.

The matrix A is symmetrically scaled to have a unit diagonal with the transformation

$$\left(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)\left(D^{\frac{1}{2}}u\right) = \left(D^{-\frac{1}{2}}b\right),$$

where D is the main diagonal of the unscaled matrix. In general, this transformation simplifies coding and improves the condition of the matrix.

On the Y-MP, the east and north coefficients are stored by diagonals in one-dimensional vectors. This allows the matrix-vector product $z = Ap$ to be computed with long vector operations. On the CM-2, the matrix coefficients are stored as two-dimensional arrays. Note that even though the matrix A is symmetric, the west and south coefficients are stored on the CM-2 to avoid additional data motion operations.

4 Preconditioning Techniques

Let the diagonally scaled matrix A be written as

$$A = I + E + N + W + S,$$

where the E, N, W, and S matrices contain the east, north, west, and south coefficients, respectively. For the symmetric problems considered in this study, $W = E^T$ and $S = N^T$. The preconditioning methods used in the study are described below in terms of these matrices.

4.1 Jacobi

For this method, Q is the main diagonal of A. For the scaled matrix, the main diagonal is I, and thus the preconditioning step is

$$\delta = Q^{-1}r = r.$$

This operation is trivially vectorizable and parallelizable on both the Y-MP and CM-2. This method can be considered as unpreconditioned conjugate gradient on the scaled system.

4.2 Line Jacobi

For this method, $Q = T$, where T is a tridiagonal matrix formed from A. We consider two cases:

line Jacobi, x direction: In this case, $T = I + W + E$, so that T is formed from the east and west coefficients. Note that T is composed of n_y independent tridiagonal systems of length n_x.

line Jacobi, y direction: In this case, $T = I + N + S$, so that T is formed from the north and south coefficients. Note that T is composed of n_x independent tridiagonal systems of length n_y.

The line Jacobi methods in the x direction and in the y direction are implemented on both machines. In either case, the preconditioning step,

$$\delta = T^{-1} r,$$

involves solving independent tridiagonal systems of equal size. This operation vectorizes easily on the Y-MP by operating on corresponding elements of all the systems at once. A Fortran routine is used for solving independent tridiagonal systems on the Y-MP. For the CM-2, specially optimized routines in the CMSSL (version 3.0) library for factoring and solving independent tridiagonal systems are used. These routines use an algorithm that combines substructuring with cyclic reduction [3, 11]. The tridiagonal factorization and solution steps (CMSSL routines `gen_tridiag_factor` and `gen_tridiag_solve_factored`, respectively) are separated so that the factorization is done only once prior to initiation of the PCG iterations.

4.3 Symmetric Gauss-Seidel (Red-Black Ordering)

If A is permuted to have a red-black structure, then A can be written as

$$A = \begin{pmatrix} I & F_R \\ F_B & I \end{pmatrix}.$$

In this case, the symmetric Gauss-Seidel (SGS) preconditioner is given by

$$Q = \begin{pmatrix} I & 0 \\ F_B & I \end{pmatrix} \begin{pmatrix} I & F_R \\ 0 & I \end{pmatrix}.$$

The preconditioning step $\delta = Q^{-1} r$ is given by the two-step process

$$\begin{aligned} \delta_B &= r_B - F_B r_R \\ \delta_R &= r_R - F_R \delta_B \end{aligned}$$

These operations vectorize and parallelize well. On the Y-MP, constant-stride vector operations of length $\frac{1}{2}n$, where n is the system size, are used on the unpermuted matrix and vectors. On the CM-2, the optimized CMSSL matrix-vector product routine is used in conjunction with logical mask arrays to update the black points first, followed by an update of the red points.

4.4 Incomplete LU Decomposition

The ILU method is very popular on scalar computers and has been studied extensively [4, 5, 7, 8, 12, 14]. For this method, an incomplete LU decomposition of A is used for Q. Thus,

$$Q = (M + L)\, M^{-1}\, (M + U)\,,$$

where M, L, and U are diagonal, strictly lower triangular, and strictly upper triangular matrices, respectively. The elements of M are the pivots of the factorization. For the unmodified ILU method, M, L, and U are chosen so that

$$Q_{i,j} = A_{i,j} \tag{5}$$

if $i = j$, $L_{i,j} \neq 0$, or $U_{i,j} \neq 0$. For the modified ILU method (denoted MILU), M, L, and U are chosen so that Eq. (5) is satisfied if $L_{i,j} \neq 0$ or $U_{i,j} \neq 0$, and, in addition, $Q - A$ has zero row sums.

Often, L and U are chosen so that $I + L + U$ has the same nonzero element structure as A. This is denoted an ILU(0) (or MILU(0)) incomplete factorization since no fill-in of nonzero elements is allowed. If fill-in is allowed so that $I + L + U$ contains more nonzeros than A, an ILU(k) (or MILU(k)) factorization results, where $k \geq 1$ describes the level of fill-in. Typically, Q more closely approximates A with increasing values of k but also has greater storage costs.

Note that

$$Q = \left(I + \tilde{L}\right) M \left(I + \tilde{U}\right),$$

where $\tilde{L} = LM^{-1}$ and $\tilde{U} = M^{-1}U$. The solution to $Q\delta = r$ is thus effected using a three-step procedure:

1. solve $(I + \tilde{L})x = r$ for x (the forward solution),
2. solve $My = x$ for y (the diagonal solution), and
3. solve $(I + \tilde{U})\delta = y$ for δ (the backward solution).

This study investigates the use of the ILU and MILU methods for various levels of fill-in under the natural ordering of the unknowns and considers various techniques for vectorizing the forward and backward solution steps on the Y-MP. The ILU and MILU methods are not competitive on the CM-2 due to the recursiveness of these operations.

Incomplete LU factorizations with various levels of fill-in are applied with the unknowns in the natural ordering. The factorizations used are unmodified incomplete LU decomposition, ILU(k), and modified incomplete LU decomposition, MILU(k), where $k = 0, 1, 2, 3$ refers to the level of fill-in. The fill-in stencils for these factorizations are given in Figures 4–7.

Many efforts have been made to vectorize the ILU and MILU preconditioners using natural ordering [1, 2, 6, 16, 17, 18]. For the Y-MP, two approaches are tried for optimizing the forward and backward solution steps.

	N	
W	C	E
	S	

Figure 4: ILU(0) fill-in pattern

NW	N	
W	C	E
	S	SE

Figure 5: ILU(1) fill-in pattern

NWW	NW	N		
	W	C	E	
		S	SE	SEE

Figure 6: ILU(2) fill-in pattern

NWWW	NWW	NW	N			
	WW	W	C	E	EE	
			S	SE	SEE	SEEE

Figure 7: ILU(3) fill-in pattern

Line-Oriented: With this approach, described in [6], each line of nodes is updated one at a time. There is an inherent recursion between the nodes of a single line, which is a first order linear recursion for the ILU(0), ILU(1), and ILU(2) patterns and a second order linear recursion for the ILU(3) pattern. The Cray Assembly Language (CAL) routines `folrp` and `solr3` are used to implement the first and second order linear recursions, respectively.

Computational Wavefronts: In this approach, described in [1, 2], the nodes are grouped into computational wavefronts in which all nodes belonging to the same wavefront can be updated simultaneously. For the ILU(0) pattern, the computational wavefronts are the grid diagonals running from northwest to southeast. Thus

$$W_k = \{u_{i,j} | i + j - 1 = k\}, \qquad k = 1, 2, \ldots, n_x + n_y - 1$$

is the k-th wavefront, all of whose elements can be updated once the elements in W_{k-1} have been updated. The wavefronts for an (M)ILU(0) forward solve with arrows indicating dependencies are given in Figure 8. Unfortunately, the vector lengths are much shorter than the problem size, and increasing the level of fill-in results in even smaller vector lengths. For the ILU(1) pattern, the computational wavefronts are

$$W_k = \{u_{i,j} | i + 2j - 2 = k\}, \qquad k = 1, 2, \ldots, n_x + 2n_y - 2,$$

all of whose elements can be updated once the elements in W_{k-1} and W_{k-2} have been updated. The wavefronts for an (M)ILU(1) forward solve with arrows indicating dependencies are given in Figure 9.

4.5 Symmetric Alternating Direction Implicit (SADI)

Let

$$A = D + E + N + W + S,$$

where D, E, N, W, and S are the diagonal, east, north, west, and south coefficients, respectively. Let

$$H = D_H + E + W \qquad V = D_V + N + S$$

for diagonal matrices D_H and D_V chosen so that $D = D_H + D_V$ and H and V are positive semi-definite. Thus, $A = H + V$, where H is tridiagonal, and V can be permuted to be tridiagonal. Let Ω and Ω' be diagonal matrices with positive elements, and define the four-step basic iterative method:

$$\begin{aligned}
(H + \Omega)\, u^{(n+\frac{1}{4})} &= (-V + \Omega)\, u^{(n)} + b \\
(V + \Omega')\, u^{(n+\frac{1}{2})} &= (-H + \Omega')\, u^{(n+\frac{1}{4})} + b \\
(V + \Omega')\, u^{(n+\frac{3}{4})} &= (-H + \Omega')\, u^{(n+\frac{1}{2})} + b \\
(H + \Omega)\, u^{(n+1)} &= (-V + \Omega)\, u^{(n+\frac{3}{4})} + b.
\end{aligned}$$

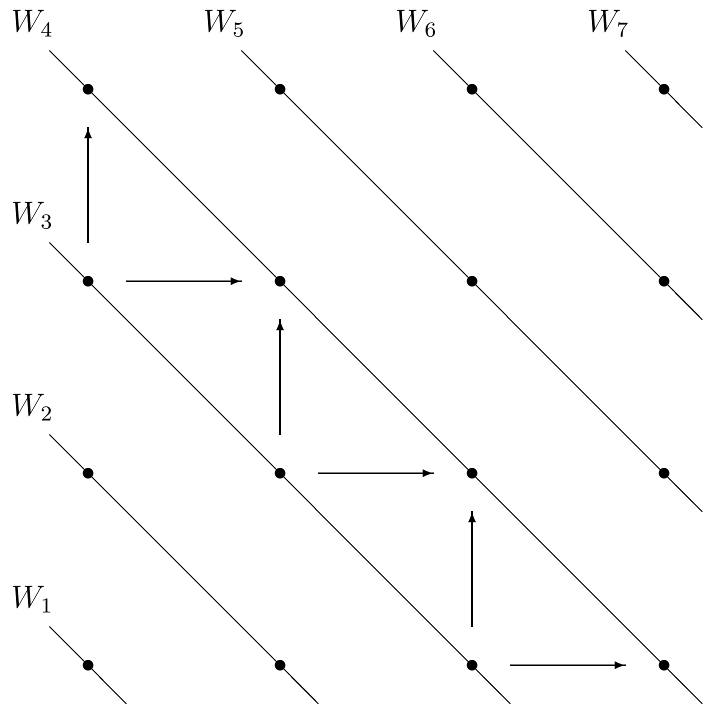

Figure 8: ILU(0) Wavefronts for 5-Point Star

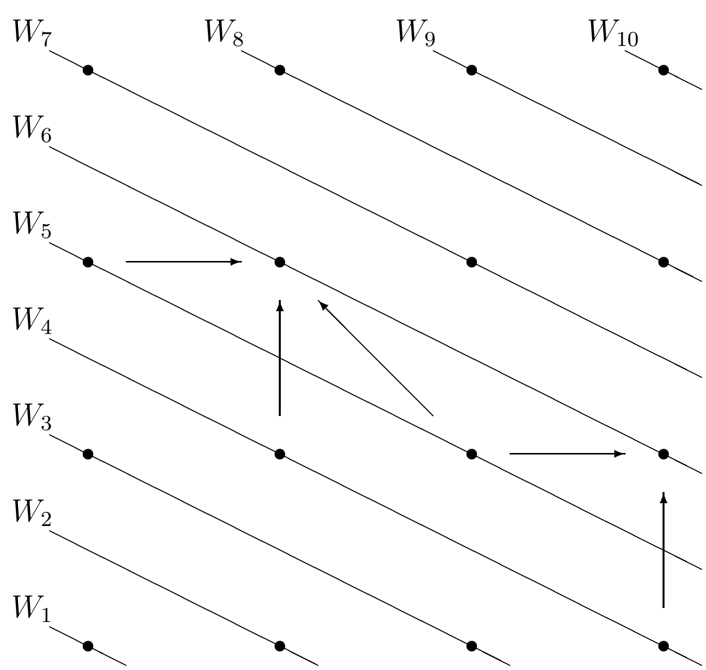

Figure 9: ILU(1) Wavefronts for 5-Point Star

This method can also be written as

$$\begin{aligned} u^{(n+\frac{1}{2})} &= G_1 u^{(n)} + k_1 \\ u^{(n+1)} &= G_2 u^{(n+\frac{1}{2})} + k_2, \end{aligned}$$

where $G_i = I - Q_i^{-1} A$, $k_i = Q_i^{-1} b$ for $i = 1, 2$, and

$$\begin{aligned} Q_1 &= (H + \Omega)(\Omega + \Omega')^{-1}(V + \Omega') \\ Q_2 &= (V + \Omega')(\Omega + \Omega')^{-1}(H + \Omega). \end{aligned}$$

The splitting matrix corresponding to this composite iterative method is given by

$$Q = Q_1 (Q_1 + Q_2 - A)^{-1} Q_2.$$

Note that if A is symmetric, $Q_2 = Q_1^T$ so $Q^T = Q$. If Q is also SPD, conjugate gradient acceleration can be applied to the basic iterative method.

We now consider the solution of $Q\delta = r$ required in the PCG algorithm. Note that

$$\begin{aligned} Q^{-1} &= Q_2^{-1}(Q_1 + Q_2 - A) Q_1^{-1} \\ &= Q_2^{-1} + Q_1^{-1} - Q_2^{-1} A Q_1^{-1}. \end{aligned}$$

Thus, an efficient way to evaluate $\delta = Q^{-1} r$ is with the two-step process:

$$\begin{aligned} v &= Q_1^{-1} r \\ \delta &= v + Q_2^{-1}(r - Av). \end{aligned}$$

This operation involves computing a matrix-vector product and solving four tridiagonal systems corresponding to $H + \Omega$ and $V + \Omega'$. $H + \Omega$ is composed of multiple independent tridiagonal subsystems of equal size corresponding to the horizontal grid lines. Similarly, $V + \Omega'$ is permutable to a tridiagonal system composed of multiple independent tridiagonal subsystems of equal size corresponding to the vertical grid lines. On the Y-MP, these tridiagonal systems are solved in vector mode by treating corresponding elements of the tridiagonal systems as vectors. On the CM-2, special CMSSL software for solving multiple independent tridiagonal systems is used. On both computers, the tridiagonal systems are factored once prior to iterating, while the tridiagonal solutions are computed for every iteration.

A crucial issue for this method is the selection of the diagonal matrices Ω and Ω'. Suppose first that $\Omega = \omega I$ and $\Omega' = \omega' I$. Suppose also that the eigenvalues of H lie in the interval $[a, b]$, and the eigenvalues of V lie in the interval $[\alpha, \beta]$. The optimal choice of the parameters ω and ω', in the sense that the spectral radius of the ADI iteration matrix is minimized [19], is given by:

Case 1: H and V have the same range. If $a = \alpha$ and $b = \beta$, the optimal parameters are given by

$$\omega = \omega' = \sqrt{ab}. \tag{6}$$

Problem	n_x	a	b	α	β
EXPNA	63	0.477313825e–3	0.999522686	0.477313825e–3	0.999522686
	127	0.115313832e–3	0.999884686	0.115313832e–3	0.999884686
	255	0.279965643e–4	0.999972003	0.279965643e–4	0.999972003
EXPNC	63	0.149084803e–3	1.96854925	0.149084803e–3	1.96854925
	127	0.366570062e–4	1.98549814	0.366570062e–4	1.98549814
	255	0.908774727e–5	1.99321588	0.908774727e–5	1.99321588
EXP10G	63	0.0	1.33206893	0.0	1.33206893
	127	0.0	1.33299770	0.0	1.33299770
	255	0.0	1.33324682	0.0	1.33324682
EXP6G	63	0.0	1.99959693	0.0	1.99894410
	127	0.0	1.99971204	0.0	1.99938719
	255	0.0	1.99976498	0.0	1.99960765

Table 1: H and V Eigenvalue Bounds

Case 2: H and V have different ranges. If $a \neq \alpha$ or $b \neq \beta$, the optimal parameters are given by

$$\omega = \frac{-p + Rq\sqrt{c}}{1 - Rs\sqrt{c}} \qquad \omega' = \frac{p + Rq\sqrt{c}}{1 + Rs\sqrt{c}} \tag{7}$$

where

$$\begin{aligned}
\theta &= \frac{2(\beta-\alpha)(b-a)}{(a+\alpha)(b+\beta)} \\
c &= \left[1 + \theta + \sqrt{\theta(2+\theta)}\right]^{-1} \\
Rs &= \frac{(\beta-\alpha)-(b-a)}{(b+\beta)-(a+\alpha)c} \\
Rq &= \frac{1}{2}\left[(b+\beta)+(b-\beta)Rs\right] \\
p &= \frac{1}{2}\left[(b-\beta)+(b+\beta)Rs\right]
\end{aligned}$$

The eigenvalue bounds of H and V for the four sample problems are given in Table 1. Using these eigenvalue bounds, optimal values for ω and ω' are computed using Eqs. (6) and (7) for problems EXPNA and EXPNC. For EXPNC, the optimal values for ω and ω' result in a preconditioning matrix Q that is not positive definite. Thus, for problems EXPNC and EXP10G, good values for ω and ω' are found by experimentation. Good values for these parameters could not be found for EXP6G. The values of ω and ω' used in the numerical experiments are given in Table 2.

Problem	n_x	ω	ω'
EXPNA	63	0.218422983e–1	0.218422983e–1
	127	0.107378086e–1	0.107378086e–1
	255	0.529110391e–2	0.529110391e–2
EXPNC	63	0.029	0.029
	127	0.023	0.023
	255	0.021	0.021
EXP10G	63	0.120	0.120
	127	0.093	0.093
	255	0.088	0.088

Table 2: ω and ω' SADI Parameters

4.6 Least Squares Polynomial

For this method, the preconditioning matrix Q is such that

$$Q^{-1} = p_n(A)$$

where p_n is an n-th degree polynomial. The polynomials $\{p_n\}$ are to be chosen so that $Q^{-1}A \approx I$, or, equivalently, that

$$I - Q^{-1}A = I - p_n(A)A$$

is small. Note that if λ is an eigenvalue of A, then $q_{n+1}(\lambda)$ is an eigenvalue of $I - Q^{-1}A$, where

$$q_{n+1}(x) = 1 - xp_n(x).$$

Since we wish these eigenvalues to be small, we can choose q_{n+1} to be small in a least squares sense over an interval that includes the eigenvalues of A. Note that

$$q_{n+1}(0) = 1.$$

Thus q_{n+1} is chosen so that

$$\int_m^M [q_{n+1}(x)]^2 \, w(x) \, dx$$

is minimized, where the interval $[m, M]$ contains the spectrum of A and w is a positive weighting function. Since A is positive definite, $m = 0$ can be chosen. For this study, the weighting function

$$w(x) = x^c (M - x)^d$$

is used where $c > -1$ and $d > -1$ are chosen to emphasize portions of the spectrum of A. Thus $c = d = 0$ results in a uniform weighting of the spectrum, while $c = d = -\frac{1}{2}$ results in a greater weighting of the extremal eigenvalues than of the interior eigenvalues. For the four test problems considered in this paper, the choice $c = d = -\frac{1}{2}$ results in a more effective preconditioner than the choice $c = d = 0$.

With this choice of weighting function, it can be shown that the $\{p_n\}$ and $\{q_{n+1}\}$ polynomials obey the recurrence relations

$$\begin{aligned} p_n(x) &= (-\alpha_n x + \beta_n + 1)\, p_{n-1}(x) - \beta_n p_{n-2}(x) + \alpha_n \\ q_{n+1}(x) &= (-\alpha_n x + \beta_n + 1)\, q_n(x) - \beta_n q_{n-1}(x), \end{aligned}$$

where

$$\begin{aligned} \alpha_n &= \frac{(2n+c+d+2)(2n+c+d+3)}{M(n+c+2)(n+c+d+2)} \\ \beta_n &= \frac{n(n+d)(2n+c+d+3)}{(n+c+2)(n+c+d+2)(2n+c+d+1)}. \end{aligned}$$

Since the unscaled matrix A is weakly diagonally dominant, it is known that $\lambda_{\max}(A) < 2$ for the diagonally scaled matrix A[19], so $M = 2$ can be chosen. Using these recurrence relations, the coefficients of $\{p_n\}$ are calculated explicitly. The preconditioning step

$$\delta = Q^{-1} r = p_n(A) r$$

is evaluated using Horner's rule and repeated matrix-vector multiplication operations.

5 Performance Analysis and Observations

The PCG methods discussed in Section 4 were applied to the four test problems described in Section 2. For all runs, the starting solution is $u^{(0)} = 0$ and the stopping criterion is

$$\frac{\|r^{(n)}\|_2}{\|b\|_2} < 10^{-6}.$$

A single processor of a Cray Y-MP with a clock cycle of 6 ns is used. The testing programs are written in single precision Fortran 77 code with the exception of calls to the BLAS routines `saxpy`, `scopy`, `sdot`, and `snrm2` and calls to the SCILIB routines `folrp` and `solr3`. Calls to `second` are used to measure CPU times. The operating system is UNICOS 7.0.2, and the Fortran compiler is CFT77 5.0.2.15. In order to minimize memory bank conflicts for some methods, the mesh sizes used for the problems are slightly different than for the CM-2: $n_x = n_y = 63$, $n_x = n_y = 127$, and $n_x = n_y = 255$.

One quadrant of a 64K CM-2 with a clock speed of 7 MHz is used for the runs. The CM-2 is equipped with Weitek double precision floating point processors. The testing programs are written in double precision Fortran 90 code with the exception of calls to CMSSL routines for performing inner products, 2-norm calculations, 5-point star matrix-vector products, and tridiagonal factorizations and solutions. The operating system is CMSS 6.1.1, the Fortran compiler is CMF 1.2 (slicewise), and the library is CMSSL 3.0. The mesh sizes used for the problems are $n_x = n_y = 64$, $n_x = n_y = 128$, and $n_x = n_y = 256$. The timing commands used to measure the CPU time are the following:

```
call cm_timer_clear (0)
call cm_timer_start (0)

code to be timed

call cm_timer_stop (0)
t_tot = cm_timer_read_elapsed (0)
t_idle = cm_timer_read_cm_idle (0)
tim = t_tot - t_idle
```

The performance results are tabulated in Tables 8 through 31 in Appendix A. In these tables, the following abbreviations are used:

Name	Meaning
ITER	number of PCG iterations required for convergence
TIMIT	corresponding CPU time in seconds, excluding factorizations
TIMFAC	factorization CPU time in seconds
$\kappa(Q^{-1}A)$	condition number of $Q^{-1}A$
JACOBI	Jacobi (Section 4.1)
LJACX	line Jacobi, x direction (Section 4.2)
LJACY	line Jacobi, y direction (Section 4.2)
SGS-RB	symmetric Gauss-Seidel using red-black ordering (Section 4.3)
ILU(k)	Incomplete LU decomposition, fill-in level k (Section 4.4)
MILU(k)	Modified Incomplete LU decomposition, fill-in level k (Section 4.4)
SADI	Symmetric Alternating Direction Implicit (Section 4.5)
LSP(k)	Least Squares Polynomial, degree k, weights $c = d = -\frac{1}{2}$ (Section 4.6)

5.1 Y-MP Results

Based on the results obtained on the Cray Y-MP, the following observations are noted:

1. The most effective methods for the problems with continuous coefficients (EXPNA and EXPNC) are SADI and the wave version of MILU(1). The most effective methods for the problems with discontinuous coefficients (EXP10G and EXP6G) are, respectively, SADI and the wave version of ILU(k) with high values of k. MILU(k) cannot be used for either problem since Q is not positive definite.

Problem	63×63	127×127	255×255
EXPNA	SADI	SADI	MILU(1) (wave)
EXPNC	SADI	MILU(1) (wave)	MILU(1) (wave)
EXP10G	SADI	SADI	SADI
EXP6G	ILU(2) (line)	ILU(1) (wave)	ILU(3) (wave)

Table 3: Best Methods on the Y-MP

2. For the ILU(k) and MILU(k) methods, the wave approach is faster than the line approach except for the smallest problem size with high values of k.

3. The red-black method SGS-RB performs better than Jacobi by halving the number of iterations and using long vector operations.

4. The least squares methods are more effective than the Jacobi method. Higher degree polynomials are more effective than lower degree polynomials up to some optimal degree.

5. For problem EXPNA, the SADI method performs very well with an iteration count close to that of MILU(3). Using good values for the iteration parameters ω and ω', the number of iterations grows according to $h^{-\frac{1}{2}}$ as does MILU(k). For EXP6G, the SADI method is not as effective since good values could not be found.

6. LJACX is faster than Jacobi for EXP6G since the number of iterations is greatly reduced. For the remaining three problems, LJACX offers little improvement over Jacobi. For all four problems, LJACY is not effective.

The best methods on the Y-MP are summarized in Table 3.

5.2 CM-2 Results

Similarly, the following observations are noted based on the results obtained on the CM-2:

1. The SADI method is the most effective method for the problems with continuous coefficients for the largest problem size. For the problems with discontinuous coefficients or for small problem sizes, the least squares polynomial method is the most effective. As noted previously, the ILU and MILU methods were not competitive on the CM-2 due to the recursiveness of the forward and backward solution steps.

2. The least squares methods are significantly faster than the Jacobi method, indicating that the inner product calculations are expensive compared to computing matrix-vector products. Higher degree polynomials are more effective than lower degree polynomials up to some optimal degree.

Problem	64×64	128×128	256×256
EXPNA	LSP(12)	LSP(11)	SADI
EXPNC	LSP(11)	LSP(12)	SADI
EXP10G	LSP(12)	LSP(11)	LSP(12)
EXP6G	LSP(12)	LSP(12)	LSP(12)

Table 4: Best Methods on the CM-2

3. The line Jacobi method LJACX is effective for problem EXP6G. LJACY is generally less effective than LJACX due to higher iteration counts or memory bank conflicts.

4. The red-black method SGS-RB performs better than the Jacobi method for small problems but worse for large problems. The number of iterations is reduced by a factor of two, but each iteration is almost twice as expensive as a Jacobi iteration.

The best methods on the CM-2 are summarized in Table 4.

6 Timing Relationships

In this section, empirical relationships derived from the data displayed in Tables 8 through 31 are presented which demonstrate, mathematically, the quantitative and qualitative behavior of the various preconditioners as a function of problem definition, problem size (number of unknowns) and the utilized computer (Y-MP and CM-2). In particular, the relationships are presented for EXPNA, for which the coefficients are continuous and isotropic, and EXP6G, for which the coefficients are sharply discontinuous and anisotropic. Not surprisingly, the relationships differ rather dramatically for the three modes of comparison. The relationships for EXPNC and EXP10G are not included since they differ in complexity somewhere between the two extremes.

Apart from the least squares polynomial preconditioner (LSP), which is treated a bit differently and discussed separately at the end of this section, approximate formulas relating number of iterations (ITER) and computation time (TIMIT) to problem size have been derived from the tabulated performance data assuming models of the form

$$\text{ITER} = c_1 h^{-\xi_1} \quad \text{and} \quad \text{TIMIT} = c_2 h^{-\xi_2}$$

where c_1, c_2, ξ_1, and ξ_2 are constants to be determined. (Recall that h is the distance between adjacent grid points and is equal to $(n_x + 1)^{-1}$ in the test problems.) Since the factors contributing to the growth of TIMIT with problem size are of primary interest, the two models are combined in the form

$$\text{TIMIT} = c(1/h)^{\xi+\eta}$$

where now ξ is the growth factor related to number of iterations and η is the corresponding factor related to computation time per iteration.

Plots of log(TIMIT) versus $\log(1/h)$ for the tabulated data indicate that linearity is approached asymptotically with increasing problem size (decreasing h). Also, the performance characteristics of the CM-2 are such that the timings for the smallest size (64×64) do not conform to our model: timings for LJACY and SADI on the 64×64 grid, for example, are nearly equal to or even greater than those obtained on the 128×128 grid. Consequently, in this study, "best" values of the model parameters are obtained by utilizing the tabulated data for only the two largest problem sizes. (In a follow-on study, it is planned to consider problem sizes greater than 256×256 so that parameter values may be obtained through application of regression techniques.) The resulting values of c, ξ, η, and $\xi + \eta$ are displayed in Table 5 for the Y-MP and the CM-2.

Ignoring LSP for the moment, the following observations are noted:

1. As one should expect, the listed values of ξ for a particular problem are machine independent, differing by less than two percent between the Y-MP and the CM-2. Similarly, the values of η for a particular machine are problem independent, differing by roughly the same percentage between EXPNA and EXP6G.

2. The values of ξ for SADI and the MILU preconditioners lie between 0.50 and 0.42; for all other preconditioners the values cluster around unity. Thus, the number of iterations for the former grows with problem size at roughly half the rate of the latter.

3. The values of ξ for EXP6G are greater than unity; for EXPNA they are less than unity. EXP6G is thus a more difficult problem to solve than is EXPNA—as one would expect since the condition number of $Q^{-1}A$ is generally two orders of magnitude greater for the former than the latter.

4. Values of η for the ILU preconditioners are the "same" as those for the corresponding MILU preconditioners. Thus, c and ξ are the important factors for determining rankings for the preconditioners within these categories.

5. The values of c are considerably larger on the CM-2 than the corresponding values on the Y-MP; however, values of the growth factor $(\xi + \eta)$ are consistently less. Hence, for each of the applicable preconditioners, there is a minimum problem size beyond which each will run faster on the CM-2 than on the Y-MP.

6. For EXPNA on the CM-2, $\xi + \eta$ is substantially smaller for SADI than for any of the applicable preconditioners, even though its value of c is substantially greater. Hence, SADI should ultimately become the "best" preconditioner for EXPNA on the CM-2.

7. The line version of ILU(2) is always faster than the line version of ILU(1) which, in turn, is always faster than the line version of ILU(0) for both EXPNA and EXP6G on the Y-MP. (Values of both c and $\xi + \eta$ for the line version of ILU(k) are monotonically decreasing as k goes from 0 to 2.)

	EXPNA				EXP6G				
Method	c	ξ	η	$\xi + \eta$	c	ξ	η	$\xi + \eta$	Notes
Y-MP									
JACOBI	.1885	0.979	1.982	2.961	1.339	1.061	1.991	3.052	
LJACX	.2622	0.956	1.969	2.925	.6286	1.127	1.976	3.103	
LJACY	.2404	0.956	1.987	2.943	.7187	1.253	1.987	3.240	
SGS-RB	.1444	0.974	1.999	2.973	1.144	1.059	1.983	3.042	
SADI	.4488	0.495	2.056	2.551	—	—	—	—	
ILU(0)	.4386	0.930	1.932	2.862	.6420	1.033	1.923	2.956	LINE
	.4236	0.930	1.838	2.768	.7363	1.033	1.799	2.832	WAVE
ILU(1)	.2985	0.929	1.921	2.850	.4337	1.023	1.930	2.953	LINE
	.5208	0.929	1.743	2.672	.7774	1.023	1.748	2.771	WAVE
ILU(2)	.2822	0.930	1.901	2.831	.4290	1.007	1.922	2.929	LINE
	.7489	0.930	1.678	2.608	1.120	1.007	1.704	2.711	WAVE
ILU(3)	.2832	0.904	1.913	2.817	.4309	0.982	1.942	2.924	LINE
	2.027	0.904	1.483	2.387	2.874	0.982	1.528	2.510	WAVE
MILU(0)	1.474	0.503	1.934	2.437	—	—	—	—	LINE
	1.377	0.503	1.848	2.351	—	—	—	—	WAVE
MILU(1)	1.850	0.427	1.929	2.356	—	—	—	—	LINE
	3.169	0.427	1.756	2.183	—	—	—	—	WAVE
MILU(2)	1.761	0.444	1.905	2.349	—	—	—	—	LINE
	4.487	0.444	1.691	2.135	—	—	—	—	WAVE
MILU(3)	1.523	0.466	1.926	2.392	—	—	—	—	LINE
	10.57	0.466	1.503	1.969	—	—	—	—	WAVE
CM-2									
JACOBI	8.213	0.979	1.235	2.214	52.35	1.081	1.240	2.321	
LJACX	89.70	0.961	0.930	1.891	219.7	1.127	0.934	2.061	
LJACY	133.2	0.961	0.930	1.891	391.8	1.267	0.923	2.190	
SGS-RB	4.698	0.974	1.393	2.367	28.96	1.078	1.401	2.479	
SADI	487.2	0.498	0.885	1.383	—	—	—	—	

Table 5: Values of c, ξ, η, and $\xi + \eta$ for problems EXPNA and EXP6G on the Y-MP and CM-2. Model: TIMIT $= c(1/h)^{\xi+\eta} = c(n_x+1)^{\xi+\eta}$ (in milliseconds)

8. Similarly, the wave version of ILU(0) is always faster than the line version of ILU(0) for both EXPNA and EXP6G on the Y-MP. Ultimately, as problem size increases, the wave version of ILU(k) becomes faster than the line version of ILU(k) for $k = 1, 2, 3$.

9. The preceding observation also applies to the wave version of MILU(k) and the line version of MILU(k) for $k = 0, 1, 2, 3$, for EXPNA on the Y-MP.

10. The wave version of ILU(3) has, by far, the smallest growth factor ($\xi + \eta = 2.510$) of any of the preconditioners for EXP6G on the Y-MP. Hence, it ultimately should become the fastest for EXP6G on that machine. For similar reasons, the wave version of MILU(3) should become the fastest for EXPNA on the same machine.

11. Values of c and $\xi + \eta$ indicate that LJACX is always faster than LJACY for EXP6G on both machines and for EXPNA on the CM-2. Further, for EXPNA on both machines and for EXP6G on the CM-2, performance of LJACX relative to JACOBI increases with increasing problem size; the reverse is true for EXP6G on the Y-MP.

12. Similar arguments hold for SGS-RB. For EXPNA on both machines, performance relative to JACOBI decreases with increasing problem size, as does that for EXP6G on the CM-2; on the other hand, SGS-RB is always faster than JACOBI for EXP6G on the Y-MP.

13. SADI starts out well for EXPNA on the Y-MP, but is ultimately overtaken and surpassed by the wave version of ILU(3) and all the MILU preconditioners.

For the least squares polynomial preconditioner, LSP, an additional parameter comes into play, namely the degree n of the polynomial. Neglecting, for the moment, explicit inclusion of the parameter h, the assumed model for the preconditioner is

$$\mathrm{TIMIT} = \frac{C_1\,(C_2 + n)}{(1+n)^{\mu}},$$

where C_1, C_2, and μ are constants dependent upon problem definition, problem size, and the utilized computer. The form of the model is prompted by an intuitive expectation that (1), the number of iterations should decrease with increasing n like

$$\frac{C_3}{(1+n)^{\mu}}$$

for some μ close to unity, the "1" being selected so that at $n = 0$, C_3 should approximate the number of JACOBI iterations; and (2), the computer time per iteration should increase linearly with n, as in $C_4 + C_5 n$. The validity of the model has been confirmed by rather exhaustive analysis of the LSP performance data recorded in Tables 8 through 31.

Employing a least squares minimization procedure, the LSP parameters C_1, C_2, and μ have been derived for problems EXPNA and EXP6G relating TIMIT to problem size and

	EXPNA			EXP6G		
Size	C_1	C_2	μ	C_1	C_2	μ
Y-MP						
63×63	0.02701	1.916	0.95	0.2480	1.722	0.95
127×127	0.2093	1.915	0.95	2.238	2.082	0.95
255×255	1.566	2.030	0.95	17.80	2.281	0.95
CM-2						
64×64	0.05383	3.125	0.95	0.4676	3.269	0.95
128×128	0.2381	2.187	0.95	2.485	2.259	0.95
256×256	1.364	1.600	0.95	15.48	1.718	0.95

Table 6: LSP parameter values of C_1, C_2, and μ for problems EXPNA and EXP6G on the Y-MP and CM-2 for three problem sizes. Model: TIMIT $= C_1(C_2+n)/(1+n)^\mu$ (in seconds) where n is the degree of the polynomial.

degree of polynomial for the Y-MP and CM-2. These values are recorded in Table 6. The degree to which the values of TIMIT produced by the model agree with those recorded in Tables 8 through 31 is illustrated in Figure 10 for EXPNA and Figure 11 for EXP6G, wherein log(TIMIT) versus n is depicted graphically for the three problem sizes. It should be noted that the "critical coefficient", r^2, measuring the goodness of fit of predicted values of TIMIT (calculated from the model) versus recorded values, lies between 0.968 and 0.994 for the two largest problem sizes for both EXPNA and EXP6G, and in all cases is greater than 0.92.

Let $T = \ln(\text{TIMIT})$. Taking the first derivative of T with respect to n, in the model, one obtains

$$T'(n) = -\left(\frac{\mu}{1+n} - \frac{1}{C_2+n}\right).$$

Setting the derivative equal to zero yields the optimum value of n for which TIMIT is a minimum, namely

$$n_{\text{opt}} = \frac{\mu C_2 - 1}{1-\mu} = 19C_2 - 20.$$

From Table 6 it may be concluded, therefore, that n_{opt} "moves to the right" on the Y-MP and "moves to the left" on the CM-2 with increasing problem size for both EXPNA and EXP6G.

Evaluating T' at $n = 1$, one may also conclude that the initial fractional rate of decrease in TIMIT with respect to increasing n is given by

$$-T'(1) = \left(\frac{\mu}{2} - \frac{1}{C_2+1}\right) = \left(47.5 - \frac{100}{C_2+1}\right)\%.$$

Thus, relative to JACOBI, the effectiveness of LSP as a preconditioner for both problems improves slightly with increasing problem size on the Y-MP, just as it deteriorates, rather

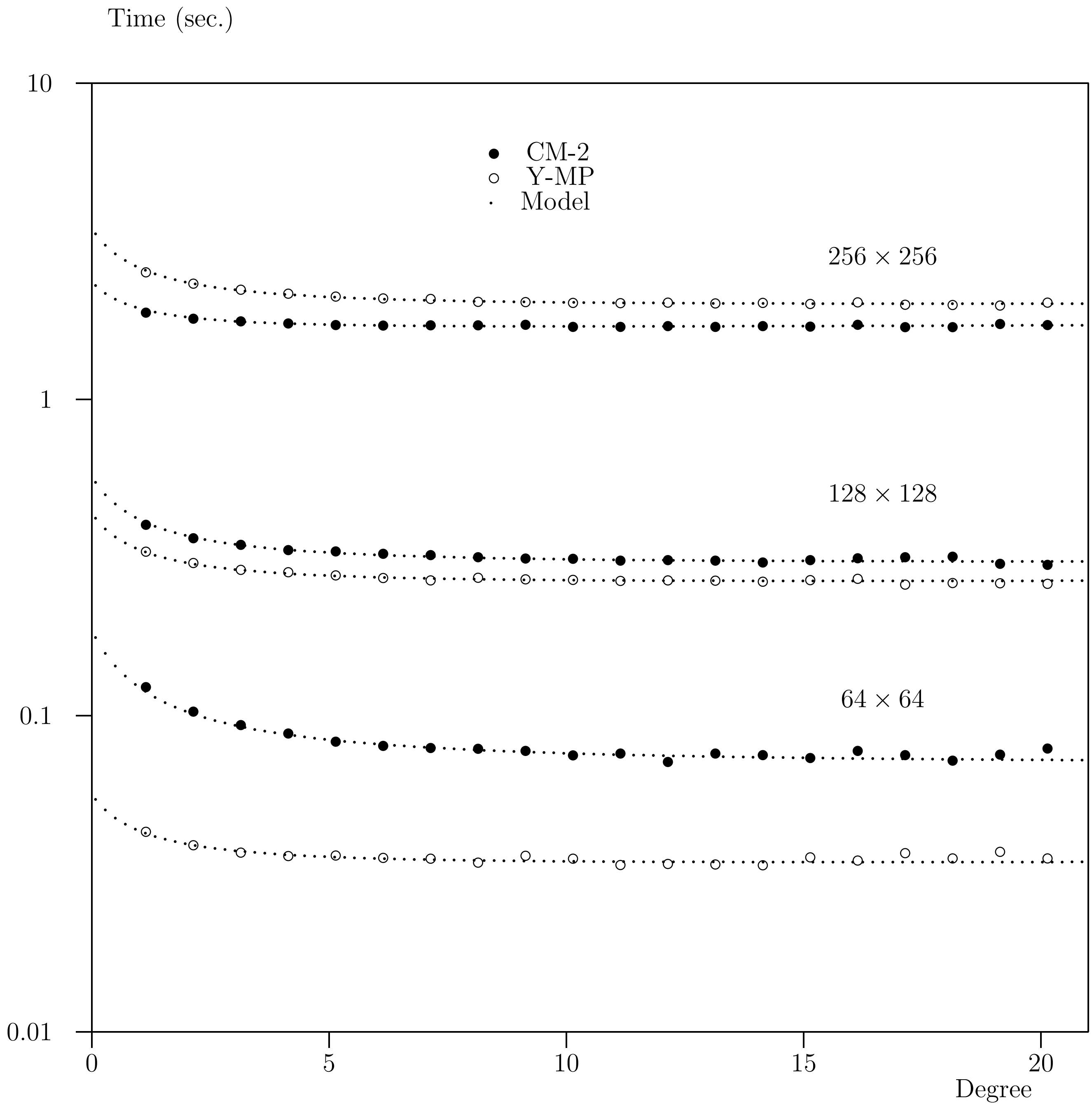

Figure 10: Time for LSP(n), EXPNA

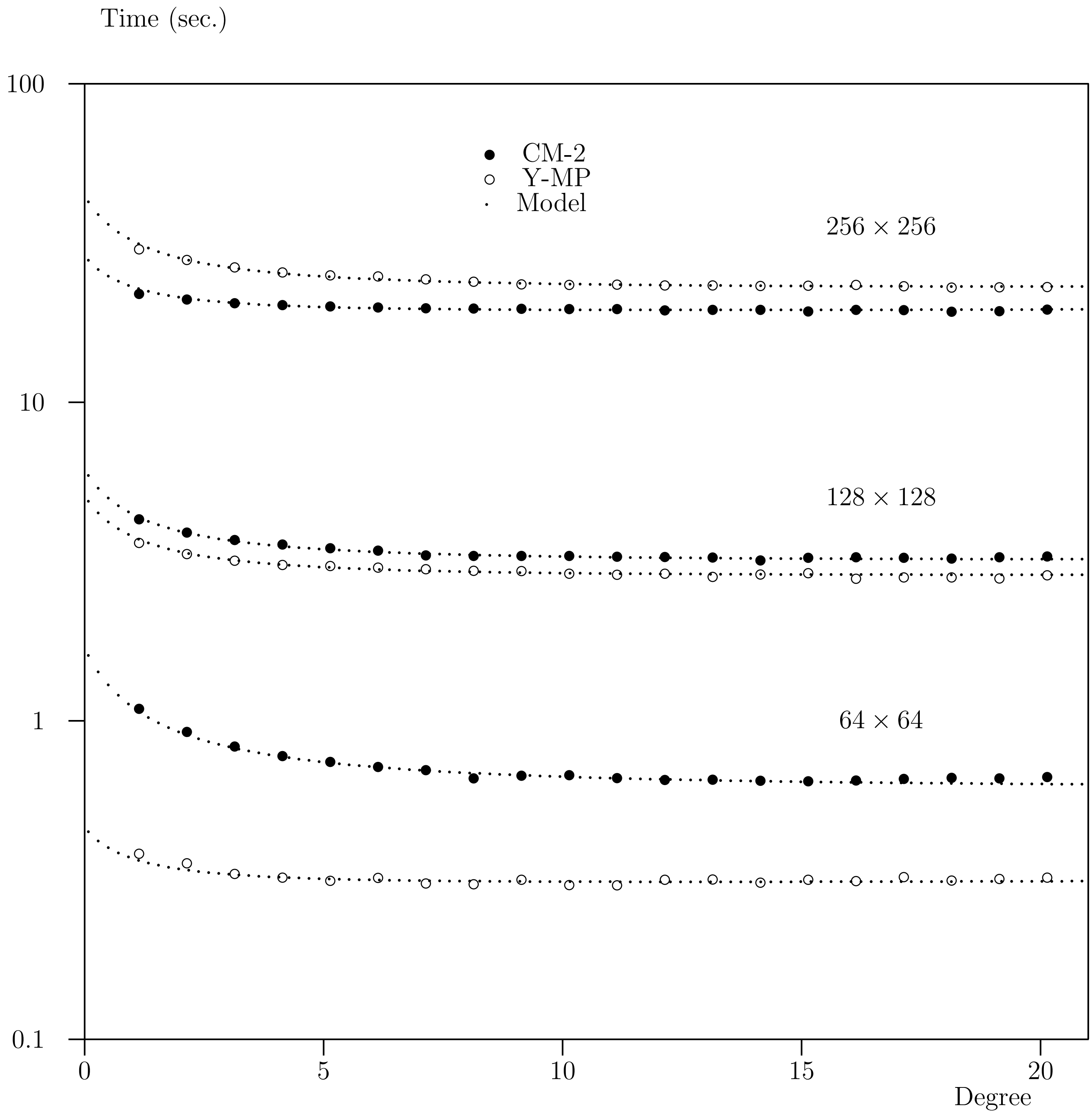

Figure 11: Time for LSP(n), EXP6G

	EXPNA				EXP6G			
Method	c	ξ	η	$\xi+\eta$	c	ξ	η	$\xi+\eta$
Y-MP								
LSP(n_{opt})	.1943	0.950	1.954	2.904	1.354	1.059	1.933	2.992
C-M2								
LSP(n_{opt})	1.315	0.939	1.593	2.532	7.590	1.067	1.587	2.654

Table 7: LSP(n_{opt}) values of c, ξ, η, and $\xi+\eta$ for problems EXPNA and EXP6G on the Y-MP and CM-2. Model: TIMIT $= c(1/h)^{\xi+\eta} = c(n_x+1)^{\xi+\eta}$ (in milliseconds)

more markedly, on the CM-2. Nevertheless, in terms of absolute effectiveness, Table 6 indicates that, for the largest problem size, LSP on the CM-2 is faster than on the Y-MP for all n, (both C_1 and C_2 for the CM-2 are less than the corresponding values on the Y-MP for both problems). The fact that LSP on the CM-2 steadily overtakes and surpasses its performance on the Y-MP is illustrated graphically in Figures 10 and 11.

Next, taking the second derivative of T with respect to n, and evaluating it at n_{opt} yields

$$T''\left(n_{\text{opt}}\right) = \frac{\mu(1-\mu)}{\left(1+n_{\text{opt}}\right)^2} = \left(\frac{4.75}{\left(1+n_{\text{opt}}\right)^2}\right)\%.$$

Inasmuch as the n_{opt} equation yields values of $10 \leq n_{\text{opt}} \leq 42$, T is thus very "flat" on either side of n_{opt} for both problems. In fact, experience indicates that the values $8 \leq n \leq 12$ produce very nearly optimum LSP performance in all cases.

Finally, evaluating TIMIT at n_{opt} leads to

$$\text{TIMIT}\left(n_{\text{opt}}\right) = \frac{C_1}{\mu}\left(\frac{C_2-1}{1-\mu}\right)^{1-\mu} = 1.220C_1\left(C_2-1\right)^{.05}$$

from which the contribution of h, ξ, and η may be determined, just as they were for the other preconditioners, using the two largest problem sizes. First note, however, that, based on Table 6 values, the term $(C_2-1)^{.05}$ for the two sizes lies in the region $(0.975, 1.012)$ for both problems for both machines. Thus, the term may be set equal to unity, for comparison purposes, and the equation becomes simply

$$\text{TIMIT}\left(n_{\text{opt}}\right) \approx 1.220C_1.$$

Let $1.220C_1 = c(1/h)^{\xi+\eta}$, where ξ and η are defined as previously (the growth component related to number of iterations and computer time per iteration, respectively). The results in Table 7 are obtained which may be appended to Table 5.

A comparison of these parameter values with those displayed in Table 5 provides some insight as to why LSP is more effective on the CM-2 than on the Y-MP (see Tables 3 and 4):

1. For EXPNA on the Y-MP, the value of $\xi+\eta$ for LSP(n_{opt}) is significantly greater than the corresponding values for SADI and all the ILU and MILU preconditioners, and its value of c is not sufficiently smaller as to make it a serious competitor.

2. A similar observation holds for EXP6G on the Y-MP. However, not only is the value of $\xi+\eta$ for LSP(n_{opt}) greater than those for the ILU preconditioners, its value of c is also greater, with the single exception of that for the wave version of ILU(3).

3. On the other hand, for both EXPNA and EXP6G on the CM-2, the value of c for LSP(n_{opt}) is smaller than those of its competitors by as much as two orders of magnitude, which tends to compensate for its larger value of $\xi+\eta$. Nevertheless, as problem size increases, LSP(n_{opt}) is ultimately overtaken by all its competitors including Jacobi (unpreconditioned conjugate gradient).

7 Concluding Remarks

This study has investigated the performance characteristics of a number of alternative preconditioners incorporated into the PCG method and applied to the solution of the two-dimensional diffusion equation on the Y-MP and the CM-2. The implementations of the resulting algorithms were optimized for both machines and subsequently utilized in solving four sample problems of varying degrees of complexity—coefficients ranging from continuous and isotropic to sharply discontinuous and anisotropic—for each of three increasingly larger problem sizes. The particular preconditioners selected for comparison were Jacobi, line Jacobi in both x and y directions (LJACX and LJACY), incomplete *LU* (ILU) and modified incomplete *LU* (MILU) decompositions, red-black Gauss-Seidel (SGS-RB), least-squares polynomial (LSP(n)), and a new symmetric alternating direction method (SADI).

The selection of the "best" preconditioner is strongly dependent upon the complexity of the problem at hand, the size of the problem, and the computer being utilized. Although the study indicates that the selection cannot be performed *a priori*, the observations given below should prove useful in the selection process for similar problems.

For continuous problems on the Y-MP, SADI and the wave versions of MILU(k) perform better than their competitors, primarily because the number of iterations required for their convergence grows like $h^{-\frac{1}{2}}$ rather than h^{-1}. For discontinuous problems, SADI is preferred if "good" values of ω and ω' can be found; otherwise, the wave versions of ILU(k) are preferable since Q for MILU(k) is not positive definite.

Neither the ILU(k) nor MILU(k) preconditioners parallelize well due to their inherent recursiveness; hence, they are not effective on the CM-2. LSP(n) (for $n \approx 10$) is most effective on that machine for "small" problems, but is eventually surpassed by SADI (if applicable) and both LJACX and LJACY as the problem size increases.

8 Acknowledgements

This research was supported in part by the Florida State University Supercomputer Computations Research Institute which is partially funded by the U.S. Department of Energy through Contract No. DE-FC05-85ER250000. Time on the Cray Y-MP was provided in part by the Florida State University Computation Center.

References

[1] C. Ashcraft. "A Moving Computation Front Approach for Vectorizing ICCG Calculations." General Motors Research Publication, GMR-5174, 1985.

[2] C. Ashcraft and R. Grimes. "On Vectorizing Incomplete Factorizations and SSOR Preconditioners." *SIAM Journal on Scientific and Statistical Computing*, Vol. 9, No. 1, January 1988, pp. 122–151.

[3] *CMSSL Release Notes, Version 3.0 Beta*, Thinking Machines Corporation, Cambridge, Massachusetts, January, 1992.

[4] T. Dupont. "A Factorization Procedure for the Solution of Elliptic Difference Equations." *SIAM Journal of Numerical Analysis*, Vol. 5, No. 4, December 1968, pp. 753–782.

[5] T. Dupont, R. P. Kendall, and H. H. Rachford. "An Approximate Factorization Procedure for Solving Self-Adjoint Elliptic Difference Equations." *SIAM Journal of Numerical Analysis*, Vol. 5, No. 3, September 1968, pp. 559–573.

[6] A. Greenbaum and G. H. Rodrigue. "The Incomplete Cholesky Conjugate Gradient Method for the STAR (5-point operator)." Research Report UCID-17574, Lawrence Livermore Laboratory, Livermore, CA., 1977.

[7] I. Gustafsson. "A Class of First Order Factorization Methods." *BIT*, Vol. 18, 1978, pp. 142–156.

[8] I. Gustafsson. *Stability and Rate of Convergence of Modified Incomplete Cholesky Factorization Methods*. Doctoral dissertation, Chalmers University of Technology and the University of Göteborg, April 1979.

[9] R. Hockney and C. Jessope. *Parallel Computers*. Adam Hilger, 1981.

[10] S. Johnsson. "Solving Tridiagonal Systems on Ensemble Architectures." *SIAM Journal on Scientific and Statistical Computing*, Vol. 8, No. 3, May 1987, pp. 354–392.

[11] C. Ho and S. Johnsson. "Optimizing Tridiagonal Solvers for Alternating Direction Methods on Boolean Cube Multiprocessors." *SIAM Journal on Scientific and Statistical Computing*, Vol. 11, No. 3, May 1990, pp. 563–592.

[12] D. S. Kershaw. "The Incomplete Cholesky–Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations." *Journal of Computational Physics*, Vol. 26, No. 1, January 1978, pp. 43–65.

[13] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. "Basic Linear Algebra Subprograms for Fortran Usage." *ACM Transactions on Mathematical Software*, Vol. 5, No. 3, September 1979, pp. 308–323.

[14] J. A. Meijerink and H. A. van der Vorst. "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix." *Mathematics of Computation*, Vol. 31, No. 137, January 1977, pp. 148–162.

[15] C. H. Tong. "Parallel Preconditioned Conjugate Gradient Methods for Elliptic Partial Differential Equations." Ph. D. Thesis, CAM Report 90-29, Department of Mathematics, University of California at Los Angeles, December 1990.

[16] H. A. van der Vorst. "A Vectorizable Variant of Some ICCG Methods." *SIAM Journal on Scientific and Statistical Computing*, Vol. 3, No. 3, September 1982, pp. 350–356.

[17] H. A. van der Vorst. "(M)ICCG for 2D Problems on Vectorcomputers." Report No. A-17, Data Processing Center, Kyoto University, Kyoto, Japan, December 1986.

[18] H. A. van der Vorst. "The Performance of FORTRAN Implementations for Preconditioned Conjugate Gradients on Vector Computers." *Parallel Computing*, Vol. 3, 1986, pp. 49–58.

[19] D. M. Young. *Iterative Solution of Large Linear Systems*. New York: Academic Press, 1971.

A Tables

In Tables 8 through 31, the following abbreviations are used:

Name	Meaning
ITER	number of PCG iterations required for convergence
TIMIT	corresponding CPU time in seconds, excluding factorizations
TIMFAC	factorization CPU time in seconds
$\kappa\left(Q^{-1}A\right)$	condition number of $Q^{-1}A$
JACOBI	Jacobi (Section 4.1)
LJACX	line Jacobi, x direction (Section 4.2)
LJACY	line Jacobi, y direction (Section 4.2)
SGS-RB	symmetric Gauss-Seidel using red-black ordering (Section 4.3)
ILU(k)	Incomplete LU decomposition, fill-in level k (Section 4.4)
MILU(k)	Modified Incomplete LU decomposition, fill-in level k (Section 4.4)
SADI	Symmetric Alternating Direction Implicit (Section 4.5)
LSP(k)	Least Squares Polynomial, degree k, weights $c = d = -\frac{1}{2}$ (Section 4.6)

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	144	0.042472		0.171640E+04	
LJACX	103	0.050855	0.000380	0.858700E+03	
LJACY	103	0.050341	0.000375	0.858700E+03	
SGS-RB	73	0.034746		0.429600E+03	
SADI	15	0.019001	0.000752	0.593808E+01	
ILU(0)	45	0.068967	0.001964	0.152530E+03	LINE
	45	0.046948	0.000739		WAVE
ILU(1)	28	0.045132	0.002184	0.575214E+02	LINE
	28	0.042488	0.001486		WAVE
ILU(2)	23	0.039449	0.002388	0.373273E+02	LINE
	23	0.048709	0.002275		WAVE
ILU(3)	17	0.037042	0.003768	0.198886E+02	LINE
	17	0.047675	0.004086		WAVE
MILU(0)	25	0.038897	0.002603	0.208639E+02	LINE
	25	0.026303	0.000826		WAVE
MILU(1)	20	0.032689	0.002561	0.110770E+02	LINE
	20	0.030692	0.001611		WAVE
MILU(2)	17	0.029528	0.002982	0.802239E+01	LINE
	17	0.036521	0.002513		WAVE
MILU(3)	14	0.030771	0.005949	0.575696E+01	LINE
	14	0.039649	0.005043		WAVE
LSP(1)	81	0.040894		0.536916E+03	
LSP(2)	56	0.037067		0.264998E+03	
LSP(3)	43	0.035161		0.158303E+03	
LSP(4)	35	0.034309		0.105387E+03	
LSP(5)	30	0.034396		0.752685E+02	
LSP(6)	26	0.033844		0.564814E+02	
LSP(7)	23	0.033650		0.439916E+02	
LSP(8)	20	0.032664		0.352410E+02	
LSP(9)	19	0.034332		0.288784E+02	
LSP(10)	17	0.033641		0.241414E+02	
LSP(11)	15	0.032126		0.204805E+02	
LSP(12)	14	0.032343		0.175974E+02	

Table 8: EXPNA, 63×63, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	278	0.327065		0.686759E+04	
LJACX	198	0.382458	0.001443	0.343430E+04	
LJACY	198	0.381542	0.001436	0.343430E+04	
SGS-RB	140	0.265556		0.171740E+04	
SADI	22	0.106589	0.002900	0.115549E+02	
ILU(0)	85	0.469813	0.007905	0.607789E+03	LINE
	85	0.288726	0.002366		WAVE
ILU(1)	52	0.302887	0.008723	0.227792E+03	LINE
	52	0.222086	0.003961		WAVE
ILU(2)	42	0.260189	0.009521	0.147135E+03	LINE
	42	0.234915	0.005867		WAVE
ILU(3)	31	0.244182	0.015156	0.774050E+02	LINE
	31	0.217154	0.010480		WAVE
MILU(0)	36	0.201577	0.010503	0.442069E+02	LINE
	36	0.123618	0.002657		WAVE
MILU(1)	29	0.170832	0.010305	0.227876E+02	LINE
	29	0.126155	0.004306		WAVE
MILU(2)	25	0.156773	0.012034	0.162546E+02	LINE
	25	0.141300	0.006666		WAVE
MILU(3)	21	0.166930	0.024581	0.115918E+02	LINE
	21	0.148774	0.012806		WAVE
LSP(1)	156	0.314617		0.214666E+04	
LSP(2)	109	0.289754		0.105894E+04	
LSP(3)	84	0.275546		0.632170E+03	
LSP(4)	69	0.270674		0.420465E+03	
LSP(5)	58	0.264401		0.299982E+03	
LSP(6)	50	0.259471		0.224893E+03	
LSP(7)	44	0.255197		0.174866E+03	
LSP(8)	40	0.259827		0.139925E+03	
LSP(9)	36	0.256977		0.114496E+03	
LSP(10)	33	0.256131		0.954386E+02	
LSP(11)	30	0.254094		0.807918E+02	
LSP(12)	28	0.255124		0.693097E+02	

Table 9: EXPNA, 127×127, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	548	2.546742		0.274724E+05	
LJACX	384	2.905259	0.005676	0.137367E+05	
LJACY	384	2.932808	0.005597	0.137367E+05	
SGS-RB	275	2.085448		0.686859E+04	
SADI	31	0.624758	0.011380	0.230215E+02	
ILU(0)	162	3.414371	0.031674	0.242893E+04	LINE
	162	1.967036	0.008406		WAVE
ILU(1)	99	2.184457	0.034838	0.908866E+03	LINE
	99	1.415249	0.013106		WAVE
ILU(2)	80	1.850989	0.038012	0.586227E+03	LINE
	80	1.432497	0.019393		WAVE
ILU(3)	58	1.720540	0.060850	0.307364E+03	LINE
	58	1.136112	0.030399		WAVE
MILU(0)	51	1.091996	0.042318	0.928515E+02	LINE
	51	0.630418	0.009544		WAVE
MILU(1)	39	0.874606	0.041434	0.465313E+02	LINE
	39	0.573123	0.014590		WAVE
MILU(2)	34	0.798797	0.049921	0.328831E+02	LINE
	34	0.620548	0.022054		WAVE
MILU(3)	29	0.875781	0.098588	0.233810E+02	LINE
	29	0.582440	0.037217		WAVE
LSP(1)	302	2.405113		0.858565E+04	
LSP(2)	211	2.213019		0.423468E+04	
LSP(3)	163	2.115581		0.252763E+04	
LSP(4)	133	2.059023		0.168087E+04	
LSP(5)	112	2.015521		0.119896E+04	
LSP(6)	97	1.988142		0.898507E+03	
LSP(7)	86	1.979721		0.698470E+03	
LSP(8)	76	1.940181		0.558643E+03	
LSP(9)	69	1.936334		0.456960E+03	
LSP(10)	63	1.925360		0.380802E+03	
LSP(11)	58	1.921363		0.322187E+03	
LSP(12)	54	1.928517		0.276159E+03	

Table 10: EXPNA, 255×255, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	166	0.048700		0.282683E+04	
LJACX	150	0.073762	0.000379	0.156490E+04	
LJACY	150	0.073355	0.000374	0.156490E+04	
SGS-RB	83	0.039637		0.707208E+03	
SADI	20	0.024959	0.000754	0.109064E+02	
ILU(0)	55	0.084052	0.001966	0.264348E+03	LINE
	55	0.057254	0.000735		WAVE
ILU(1)	34	0.054356	0.002182	0.908983E+02	LINE
	34	0.051315	0.001478		WAVE
ILU(2)	27	0.046176	0.002386	0.566950E+02	LINE
	27	0.057323	0.002269		WAVE
ILU(3)	19	0.040955	0.003768	0.301723E+02	LINE
	19	0.053077	0.004149		WAVE
MILU(0)	28	0.043384	0.002603	0.214438E+02	LINE
	28	0.029549	0.000825		WAVE
MILU(1)	21	0.034245	0.002560	0.122160E+02	LINE
	21	0.032273	0.001586		WAVE
MILU(2)	18	0.031265	0.003045	0.927800E+01	LINE
	18	0.038604	0.002543		WAVE
MILU(3)	16	0.034971	0.005973	0.721260E+01	LINE
	16	0.045132	0.005057		WAVE
LSP(1)	92	0.046451		0.883930E+03	
LSP(2)	65	0.043042		0.436142E+03	
LSP(3)	50	0.040828		0.260461E+03	
LSP(4)	41	0.039958		0.173293E+03	
LSP(5)	34	0.038641		0.123709E+03	
LSP(6)	30	0.038907		0.927763E+02	
LSP(7)	26	0.037971		0.721984E+02	
LSP(8)	23	0.037521		0.578024E+02	
LSP(9)	21	0.037581		0.473542E+02	
LSP(10)	19	0.037180		0.394927E+02	
LSP(11)	18	0.038399		0.334903E+02	
LSP(12)	16	0.036845		0.287523E+02	

Table 11: EXPNC, 63×63, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	327	0.383597		0.113251E+05	
LJACX	243	0.465141	0.001446	0.628647E+04	
LJACY	243	0.462368	0.001437	0.628647E+04	
SGS-RB	164	0.307850		0.283176E+04	
SADI	31	0.149446	0.002909	0.330615E+02	
ILU(0)	109	0.601576	0.007901	0.111337E+04	LINE
	109	0.368045	0.002368		WAVE
ILU(1)	65	0.377252	0.008721	0.385927E+03	LINE
	65	0.276979	0.003997		WAVE
ILU(2)	52	0.317764	0.009532	0.241404E+03	LINE
	52	0.290452	0.005932		WAVE
ILU(3)	36	0.281528	0.015188	0.126965E+03	LINE
	36	0.249334	0.010495		WAVE
MILU(0)	38	0.212829	0.010515	0.435445E+02	LINE
	38	0.130459	0.002653		WAVE
MILU(1)	29	0.170633	0.010308	0.248312E+02	LINE
	29	0.125644	0.004308		WAVE
MILU(2)	25	0.155461	0.012310	0.189208E+02	LINE
	25	0.142133	0.006672		WAVE
MILU(3)	22	0.174772	0.024378	0.148479E+02	LINE
	22	0.155739	0.012874		WAVE
LSP(1)	182	0.365941		0.353961E+04	
LSP(2)	127	0.335222		0.174594E+04	
LSP(3)	98	0.320001		0.104222E+04	
LSP(4)	80	0.311642		0.693142E+03	
LSP(5)	67	0.303633		0.494483E+03	
LSP(6)	59	0.303970		0.370604E+03	
LSP(7)	51	0.295230		0.288135E+03	
LSP(8)	46	0.296305		0.230495E+03	
LSP(9)	42	0.297425		0.188593E+03	
LSP(10)	38	0.294212		0.157193E+03	
LSP(11)	35	0.293790		0.133026E+03	
LSP(12)	32	0.289445		0.114052E+03	

Table 12: EXPNC, 127×127, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	639	3.027315		0.453207E+05	
LJACX	479	3.710451	0.005693	0.251979E+05	
LJACY	479	3.683182	0.005641	0.251979E+05	
SGS-RB	320	2.464935		0.113307E+05	
SADI	45	0.860395	0.011474	0.119463E+03	
ILU(0)	210	4.421512	0.031781	0.465422E+04	LINE
	210	2.598265	0.008391		WAVE
ILU(1)	128	2.830221	0.034907	0.164300E+04	LINE
	128	1.851769	0.013213		WAVE
ILU(2)	101	2.344115	0.038048	0.103663E+04	LINE
	101	1.825899	0.019411		WAVE
ILU(3)	69	2.050310	0.061099	0.542727E+03	LINE
	69	1.369302	0.030190		WAVE
MILU(0)	52	1.106436	0.042294	0.883043E+02	LINE
	52	0.640857	0.009417		WAVE
MILU(1)	39	0.874441	0.041398	0.504140E+02	LINE
	39	0.564661	0.014405		WAVE
MILU(2)	34	0.802717	0.049879	0.384921E+02	LINE
	34	0.625201	0.022126		WAVE
MILU(3)	30	0.904804	0.098571	0.303630E+02	LINE
	30	0.604751	0.037271		WAVE
LSP(1)	354	2.844981		0.141633E+05	
LSP(2)	248	2.614392		0.698559E+04	
LSP(3)	191	2.531463		0.416957E+04	
LSP(4)	156	2.434504		0.277268E+04	
LSP(5)	131	2.386826		0.197770E+04	
LSP(6)	114	2.380130		0.148202E+04	
LSP(7)	101	2.302172		0.115205E+04	
LSP(8)	90	2.318271		0.921326E+03	
LSP(9)	81	2.254195		0.753649E+03	
LSP(10)	74	2.236261		0.627953E+03	
LSP(11)	68	2.259950		0.531312E+03	
LSP(12)	63	2.267721		0.455387E+03	

Table 13: EXPNC, 255×255, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	224	0.066195		0.716080E+09	
LJACX	159	0.080075	0.000382	0.358047E+09	
LJACY	160	0.079698	0.000380	0.358037E+09	
SGS-RB	112	0.053997		0.179021E+09	
SADI	28	0.034206	0.000752	0.107402E+08	
ILU(0)	70	0.107457	0.001971	0.637390E+08	LINE
	70	0.073186	0.000744		WAVE
ILU(1)	42	0.067233	0.002191	0.225049E+08	LINE
	42	0.063564	0.001496		WAVE
ILU(2)	34	0.057819	0.002392	0.146574E+08	LINE
	34	0.071984	0.002278		WAVE
ILU(3)	24	0.051982	0.003772	0.726785E+07	LINE
	24	0.067316	0.004211		WAVE
LSP(1)	125	0.063669		0.223775E+09	
LSP(2)	87	0.057895		0.110367E+09	
LSP(3)	67	0.055011		0.658733E+08	
LSP(4)	55	0.053968		0.438018E+08	
LSP(5)	46	0.052627		0.312420E+08	
LSP(6)	40	0.052010		0.234085E+08	
LSP(7)	35	0.051365		0.181955E+08	
LSP(8)	31	0.050033		0.145415E+08	
LSP(9)	28	0.050091		0.118960E+08	
LSP(10)	26	0.050622		0.991189E+07	
LSP(11)	24	0.050504		0.838555E+07	
LSP(12)	22	0.050111		0.718978E+07	

Table 14: EXP10G, 63×63, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	458	0.540111		0.286497E+10	
LJACX	325	0.628715	0.001439	0.143252E+10	
LJACY	326	0.623105	0.001437	0.143249E+10	
SGS-RB	229	0.436954		0.716261E+09	
SADI	51	0.243623	0.002900	0.332972E+08	
ILU(0)	137	0.756294	0.007894	0.256056E+09	LINE
	137	0.465565	0.002364		WAVE
ILU(1)	84	0.485233	0.008725	0.943110E+08	LINE
	84	0.357033	0.003988		WAVE
ILU(2)	68	0.417469	0.009529	0.606882E+08	LINE
	68	0.379870	0.005991		WAVE
ILU(3)	49	0.381823	0.015482	0.314328E+08	LINE
	49	0.339193	0.010532		WAVE
LSP(1)	255	0.516748		0.895325E+09	
LSP(2)	179	0.475635		0.441582E+09	
LSP(3)	138	0.450237		0.263563E+09	
LSP(4)	113	0.439029		0.175257E+09	
LSP(5)	95	0.433594		0.124991E+09	
LSP(6)	82	0.423222		0.936656E+08	
LSP(7)	72	0.416714		0.728070E+08	
LSP(8)	65	0.416414		0.582135E+08	
LSP(9)	59	0.416343		0.476205E+08	
LSP(10)	53	0.408898		0.396666E+08	
LSP(11)	49	0.408494		0.335595E+08	
LSP(12)	45	0.403433		0.287635E+08	

Table 15: EXP10G, 127×127, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	920	4.341440		0.114474E+11	
LJACX	651	4.990820	0.005690	0.572459E+10	
LJACY	654	5.062027	0.005639	0.572435E+10	
SGS-RB	460	3.560066		0.286255E+10	
SADI	98	1.872049	0.011511	0.125950E+09	
ILU(0)	274	5.795160	0.031702	0.102808E+10	LINE
	274	3.398069	0.008391		WAVE
ILU(1)	167	3.743169	0.034977	0.381893E+09	LINE
	167	2.470868	0.013310		WAVE
ILU(2)	134	3.142412	0.038263	0.247364E+09	LINE
	134	2.459466	0.019819		WAVE
ILU(3)	98	2.942823	0.061159	0.127095E+09	LINE
	98	1.974110	0.030944		WAVE
LSP(1)	514	4.214398		0.357831E+10	
LSP(2)	361	3.899864		0.176489E+10	
LSP(3)	278	3.743033		0.105339E+10	
LSP(4)	227	3.660669		0.700457E+09	
LSP(5)	192	3.609228		0.499602E+09	
LSP(6)	166	3.539142		0.374359E+09	
LSP(7)	146	3.409709		0.290990E+09	
LSP(8)	131	3.450083		0.232694E+09	
LSP(9)	118	3.429788		0.190315E+09	
LSP(10)	108	3.384034		0.158570E+09	
LSP(11)	99	3.360585		0.134149E+09	
LSP(12)	91	3.321376		0.114961E+09	

Table 16: EXP10G, 255×255, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	1320	0.392268		0.260454E+06	
LJACX	480	0.236575	0.000378	0.371203E+05	
LJACY	993	0.488035	0.000377	0.248695E+06	
SGS-RB	659	0.316157		0.651139E+05	
ILU(0)	96	0.146199	0.001969	0.189364E+04	LINE
	96	0.100138	0.000742		WAVE
ILU(1)	62	0.098447	0.002190	0.714995E+03	LINE
	62	0.092979	0.001481		WAVE
ILU(2)	53	0.089184	0.002387	0.484406E+03	LINE
	53	0.110584	0.002288		WAVE
ILU(3)	43	0.091096	0.003771	0.253909E+03	LINE
	43	0.117128	0.004138		WAVE
LSP(1)	716	0.364189		0.813923E+05	
LSP(2)	517	0.340093		0.401434E+05	
LSP(3)	388	0.314923		0.239602E+05	
LSP(4)	317	0.306519		0.159325E+05	
LSP(5)	268	0.299674		0.113637E+05	
LSP(6)	240	0.306195		0.851518E+04	
LSP(7)	206	0.293814		0.661890E+04	
LSP(8)	185	0.292289		0.529309E+04	
LSP(9)	174	0.301962		0.432946E+04	
LSP(10)	153	0.290710		0.360712E+04	
LSP(11)	142	0.289998		0.305170E+04	
LSP(12)	137	0.301826		0.261538E+04	

Table 17: EXP6G, 63×63, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	3035	3.620642		0.102513E+07	
LJACX	1110	2.174744	0.001458	0.148357E+06	
LJACY	2484	4.836868	0.001454	0.975923E+06	
SGS-RB	1519	2.943505		0.256282E+06	
ILU(0)	197	1.087702	0.007911	0.825599E+04	LINE
	197	0.684637	0.002364		WAVE
ILU(1)	125	0.723288	0.008735	0.312277E+04	LINE
	125	0.536892	0.003997		WAVE
ILU(2)	104	0.636844	0.009526	0.209457E+04	LINE
	104	0.577452	0.005943		WAVE
ILU(3)	81	0.625329	0.015185	0.109095E+04	LINE
	81	0.558993	0.010529		WAVE
LSP(1)	1690	3.448701		0.320352E+06	
LSP(2)	1187	3.179861		0.158000E+06	
LSP(3)	917	3.033640		0.943045E+05	
LSP(4)	748	2.937570		0.627080E+05	
LSP(5)	633	2.917993		0.447262E+05	
LSP(6)	546	2.885896		0.335143E+05	
LSP(7)	482	2.853758		0.260511E+05	
LSP(8)	430	2.819671		0.208321E+05	
LSP(9)	391	2.812176		0.170394E+05	
LSP(10)	356	2.762201		0.141963E+05	
LSP(11)	322	2.740924		0.120102E+05	
LSP(12)	305	2.758649		0.102929E+05	

Table 18: EXP6G, 127×127, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$	Notes
JACOBI	6330	30.039201		0.405866E+07	
LJACX	2425	18.689889	0.005701	0.592057E+06	
LJACY	5921	45.706048	0.005693	0.385752E+07	
SGS-RB	3165	24.250905		0.101467E+07	
ILU(0)	403	8.442958	0.031727	0.354688E+05	LINE
	403	4.875713	0.008401		WAVE
ILU(1)	254	5.599851	0.034978	0.134133E+05	LINE
	254	3.664779	0.013114		WAVE
ILU(2)	209	4.848693	0.038093	0.895486E+04	LINE
	209	3.781382	0.019555		WAVE
ILU(3)	160	4.746380	0.061791	0.464959E+04	LINE
	160	3.183803	0.030933		WAVE
LSP(1)	3538	28.775477		0.126833E+07	
LSP(2)	2477	26.667278		0.625552E+06	
LSP(3)	1914	25.261012		0.373368E+06	
LSP(4)	1550	24.359400		0.248271E+06	
LSP(5)	1318	23.866203		0.177078E+06	
LSP(6)	1134	23.663889		0.132688E+06	
LSP(7)	1001	23.169630		0.103140E+06	
LSP(8)	901	22.757886		0.824774E+05	
LSP(9)	809	22.367653		0.674609E+05	
LSP(10)	738	22.244941		0.562046E+05	
LSP(11)	679	22.286974		0.475494E+05	
LSP(12)	627	22.194023		0.407511E+05	

Table 19: EXP6G, 255×255, Y-MP

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	146	0.124878		0.177048E+04
LJACX	105	0.401819	0.005381	0.885738E+03
LJACY	105	1.695205	0.014451	0.885738E+03
SGS-RB	74	0.129855		0.443119E+03
SADI	15	0.583191	0.017801	0.599783E+01
LSP(1)	82	0.117209		0.553816E+03
LSP(2)	57	0.098095		0.273317E+03
LSP(3)	44	0.088884		0.163276E+03
LSP(4)	36	0.083609		0.108674E+03
LSP(5)	30	0.078855		0.776042E+02
LSP(6)	26	0.076374		0.582659E+02
LSP(7)	23	0.075347		0.453568E+02
LSP(8)	21	0.074869		0.363565E+02
LSP(9)	19	0.073659		0.297702E+02
LSP(10)	17	0.071381		0.248895E+02
LSP(11)	16	0.072260		0.210963E+02
LSP(12)	14	0.068020		0.181285E+02

Table 20: EXPNA, 64×64, CM-2

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	280	0.387084		0.697533E+04
LJACX	199	0.878923	0.006461	0.348816E+04
LJACY	199	1.316717	0.008638	0.348816E+04
SGS-RB	141	0.464717		0.174433E+04
SADI	22	0.404919	0.013307	0.116304E+02
LSP(1)	157	0.382408		0.218033E+04
LSP(2)	110	0.346949		0.107553E+04
LSP(3)	85	0.330429		0.642095E+03
LSP(4)	69	0.317973		0.427082E+03
LSP(5)	59	0.314903		0.304712E+03
LSP(6)	51	0.309727		0.228418E+03
LSP(7)	45	0.306537		0.177625E+03
LSP(8)	40	0.302199		0.142101E+03
LSP(9)	36	0.298871		0.116295E+03
LSP(10)	33	0.298712		0.969400E+02
LSP(11)	30	0.294301		0.820714E+02
LSP(12)	28	0.295822		0.703914E+02

Table 21: EXPNA, 128×128, CM-2

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	550	1.780877		0.276874E+05
LJACX	386	3.235871	0.009806	0.138442E+05
LJACY	386	4.811940	0.015418	0.138442E+05
SGS-RB	276	2.374925		0.692235E+04
SADI	31	1.050621	0.023566	0.231072E+02
LSP(1)	303	1.793777		0.865286E+04
LSP(2)	212	1.714581		0.426783E+04
LSP(3)	164	1.683696		0.254745E+04
LSP(4)	133	1.654929		0.169404E+04
LSP(5)	112	1.638424		0.120836E+04
LSP(6)	97	1.631717		0.905524E+03
LSP(7)	86	1.635767		0.703962E+03
LSP(8)	77	1.634273		0.563014E+03
LSP(9)	70	1.640202		0.460558E+03
LSP(10)	63	1.615991		0.383770E+03
LSP(11)	58	1.616607		0.324698E+03
LSP(12)	54	1.625264		0.278328E+03

Table 22: EXPNA, 256×256, CM-2

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	168	0.143610		0.291603E+04
LJACX	153	0.583111	0.005392	0.161440E+04
LJACY	153	2.468681	0.014468	0.161440E+04
SGS-RB	84	0.147437		0.729509E+03
SADI	20	0.766265	0.017790	0.110349E+02
LSP(1)	94	0.133948		0.911805E+03
LSP(2)	66	0.113362		0.449880E+03
LSP(3)	51	0.102704		0.268661E+03
LSP(4)	41	0.094907		0.178773E+03
LSP(5)	35	0.091588		0.127602E+03
LSP(6)	30	0.087655		0.957091E+02
LSP(7)	27	0.087822		0.744718E+02
LSP(8)	24	0.085078		0.596257E+02
LSP(9)	21	0.081033		0.488367E+02
LSP(10)	20	0.083299		0.407167E+02
LSP(11)	18	0.080756		0.345217E+02
LSP(12)	17	0.081640		0.296463E+02

Table 23: EXPNC, 64×64, CM-2

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	329	0.454586		0.115028E+05
LJACX	244	1.076539	0.006454	0.638529E+04
LJACY	244	1.612855	0.008639	0.638529E+04
SGS-RB	165	0.542812		0.287620E+04
SADI	32	0.580945	0.013309	0.335722E+02
LSP(1)	183	0.445415		0.359517E+04
LSP(2)	128	0.403233		0.177334E+04
LSP(3)	99	0.384079		0.105858E+04
LSP(4)	81	0.372554		0.704019E+03
LSP(5)	68	0.362222		0.502225E+03
LSP(6)	59	0.357425		0.376423E+03
LSP(7)	52	0.353249		0.292676E+03
LSP(8)	46	0.346497		0.234112E+03
LSP(9)	42	0.347422		0.191522E+03
LSP(10)	38	0.342697		0.159635E+03
LSP(11)	35	0.341872		0.135112E+03
LSP(12)	32	0.336682		0.115846E+03

Table 24: EXPNC, 128×128, CM-2

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	641	2.075411		0.456755E+05
LJACX	481	4.030577	0.009812	0.253954E+05
LJACY	481	5.993460	0.015417	0.253954E+05
SGS-RB	321	2.759812		0.114194E+05
SADI	45	1.510728	0.023567	0.120393E+03
LSP(1)	356	2.106533		0.142741E+05
LSP(2)	249	2.012677		0.704028E+04
LSP(3)	192	1.969531		0.420221E+04
LSP(4)	157	1.951580		0.279438E+04
LSP(5)	132	1.928639		0.199318E+04
LSP(6)	114	1.914945		0.149361E+04
LSP(7)	101	1.917841		0.116108E+04
LSP(8)	90	1.906675		0.928543E+03
LSP(9)	82	1.917524		0.759540E+03
LSP(10)	75	1.919202		0.632868E+03
LSP(11)	68	1.890822		0.535467E+03
LSP(12)	63	1.891449		0.458961E+03

Table 25: EXPNC, 256×256, CM-2

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	235	0.200048		0.781338E+09
LJACX	168	0.640791	0.005389	0.390677E+09
LJACY	167	2.650017	0.014406	0.390665E+09
SGS-RB	117	0.203211		0.195336E+09
SADI	30	1.115230	0.017728	0.117126E+08
LSP(1)	131	0.185930		0.244168E+09
LSP(2)	92	0.157087		0.120425E+09
LSP(3)	71	0.142366		0.718774E+08
LSP(4)	58	0.133262		0.477937E+08
LSP(5)	49	0.127149		0.340887E+08
LSP(6)	42	0.121622		0.255439E+08
LSP(7)	37	0.118365		0.198431E+08
LSP(8)	33	0.115531		0.158763E+08
LSP(9)	30	0.114129		0.129862E+08
LSP(10)	27	0.111726		0.108197E+08
LSP(11)	25	0.110681		0.914760E+07
LSP(12)	23	0.108853		0.783556E+07

Table 26: EXP10G, 64×64, CM-2

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	469	0.646996		0.299196E+10
LJACX	331	1.458799	0.006497	0.149603E+10
LJACY	330	2.180169	0.008639	0.149600E+10
SGS-RB	234	0.767299		0.748017E+09
SADI	52	0.933184	0.013304	0.347732E+08
LSP(1)	261	0.633885		0.935020E+09
LSP(2)	183	0.575177		0.461161E+09
LSP(3)	141	0.544543		0.275248E+09
LSP(4)	115	0.526859		0.183026E+09
LSP(5)	97	0.514440		0.130543E+09
LSP(6)	84	0.507081		0.978167E+08
LSP(7)	74	0.500096		0.760345E+08
LSP(8)	66	0.494041		0.608016E+08
LSP(9)	60	0.492962		0.497196E+08
LSP(10)	55	0.492216		0.414321E+08
LSP(11)	50	0.484398		0.350396E+08
LSP(12)	47	0.489873		0.300379E+08

Table 27: EXP10G, 128×128, CM-2

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	926	2.997120		0.116977E+11
LJACX	656	5.495079	0.009849	0.584963E+10
LJACY	653	8.133537	0.015421	0.584986E+10
SGS-RB	463	3.975206		0.292510E+10
SADI	99	3.286123	0.023561	0.128718E+09
LSP(1)	516	3.051395		0.365630E+10
LSP(2)	362	2.923179		0.180349E+10
LSP(3)	280	2.867408		0.107644E+10
LSP(4)	228	2.829357		0.715781E+09
LSP(5)	193	2.814884		0.510531E+09
LSP(6)	167	2.798484		0.382547E+09
LSP(7)	147	2.783559		0.297360E+09
LSP(8)	131	2.766524		0.237780E+09
LSP(9)	119	2.773156		0.194491E+09
LSP(10)	108	2.753372		0.162039E+09
LSP(11)	100	2.768554		0.137075E+09
LSP(12)	92	2.749427		0.117477E+09

Table 28: EXP10G, 256×256, CM-2

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	1321	1.115013		0.268204E+06
LJACX	486	1.843561	0.005377	0.372601E+05
LJACY	1006	15.899689	0.014382	0.256357E+06
SGS-RB	660	1.132196		0.670514E+05
LSP(1)	738	1.038555		0.838142E+05
LSP(2)	519	0.878806		0.413379E+05
LSP(3)	400	0.792000		0.246731E+05
LSP(4)	326	0.739092		0.164065E+05
LSP(5)	277	0.707879		0.117020E+05
LSP(6)	240	0.682811		0.876864E+04
LSP(7)	213	0.666675		0.681599E+04
LSP(8)	184	0.629086		0.545060E+04
LSP(9)	173	0.641342		0.445827E+04
LSP(10)	161	0.643122		0.371444E+04
LSP(11)	147	0.629664		0.314250E+04
LSP(12)	136	0.621972		0.269320E+04

Table 29: EXP6G, 64×64, CM-2

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	3019	4.153633		0.104015E+07
LJACX	1117	4.910723	0.006466	0.148641E+06
LJACY	2493	16.413957	0.008647	0.990755E+06
SGS-RB	1513	4.932771		0.260038E+06
LSP(1)	1688	4.088706		0.325048E+06
LSP(2)	1186	3.714589		0.160316E+06
LSP(3)	917	3.523236		0.956866E+05
LSP(4)	748	3.405375		0.636271E+05
LSP(5)	630	3.316010		0.453817E+05
LSP(6)	546	3.262068		0.340056E+05
LSP(7)	471	3.149253		0.264329E+05
LSP(8)	424	3.136495		0.211374E+05
LSP(9)	387	3.138364		0.172891E+05
LSP(10)	356	3.140775		0.144044E+05
LSP(11)	327	3.117821		0.121862E+05
LSP(12)	304	3.115445		0.104439E+05

Table 30: EXP6G, 128×128, CM-2

Method	ITER	TIMIT	TIMFAC	$\kappa\,(Q^{-1}A)$
JACOBI	6360	20.573390		0.408818E+07
LJACX	2429	20.323387	0.009799	0.592625E+06
LJACY	5970	74.263136	0.015418	0.388665E+07
SGS-RB	3180	27.229709		0.102205E+07
LSP(1)	3534	20.872168		0.127756E+07
LSP(2)	2483	20.013009		0.630102E+06
LSP(3)	1908	19.487973		0.376084E+06
LSP(4)	1556	19.245263		0.250077E+06
LSP(5)	1313	19.070163		0.178366E+06
LSP(6)	1133	18.898701		0.133654E+06
LSP(7)	999	18.817947		0.103890E+06
LSP(8)	894	18.767765		0.830773E+05
LSP(9)	809	18.728504		0.679516E+05
LSP(10)	738	18.677312		0.566135E+05
LSP(11)	680	18.676897		0.478953E+05
LSP(12)	625	18.515675		0.410475E+05

Table 31: EXP6G, 256×256, CM-2