ROW PROJECTION METHODS FOR LARGE NONSYMMETRIC LINEAR SYSTEMS

R. BRAMLEY AND A. SAMEH
CENTER FOR SUPERCOMPUTING RESEARCH AND DEVELOPMENT
UNVERSITY OF ILLINOIS - URBANA *

Abstract. Three conjugate gradient accelerated row projection (RP) methods for nonsymmetric linear systems are presented and their properties described. One method is based on Kaczmarz's method and has an iteration matrix that is the product of orthogonal projectors; another is based on Cimmino's method and has an iteration matrix that is the sum of orthogonal projectors. A new RP method which requires fewer matrix-vector operations, explicitly reduces the problem size, is error reducing in the 2-norm, and consistently produces better solutions than other RP algorithms is also introduced. Using comparisons with the method of conjugate gradient applied to the normal equations, the properties of RP methods are explained.

A row partitioning approach is described which yields parallel implementations suitable for a wide range of computer architectures, requires only a few vectors of extra storage, and allows computing the necessary projections with small errors. Numerical testing verifies the robustness of this approach and shows that the resulting algorithms are competitive with other nonsymmetric solvers in speed and efficiency.

Key Words. iterative methods, nonsymmetric linear systems, Kaczmarz, Cimmino

1. Introduction. Over the last few years much research effort has been invested in developing iterative solvers for nonsymmetric linear systems $Ax = b$, where A is large, sparse, and nonsingular. These solvers can be grouped roughly into the four categories of *matrix splitting*, *CG-like, residual polynomial*, and *symmetrization* methods.

Matrix splitting methods and their acceleration via CG include the earliest iterative techniques for solving linear systems, and are based on splitting the coefficient matrix as $A = M - N$. This category includes the Jacobi, Gauss-Seidel, and successive overrelaxation methods, and convergence is assured if the spectral radius $\rho(M^{-1}N)$ is less than 1. This condition can often be shown to hold if, e.g., A is irreducible and diagonally dominant. Hageman and Young [18] describe many of these splitting techniques and provide other conditions on A that imply convergence.

The second category, CG-like methods, was motivated by the success of the conjugate gradient (CG) method for symmetric positive definite systems. Generalized conjugate residuals (GCR), Orthomin, generalized minimum residual (GMRES), Axelsson's method, and Orthodir are included in this category (see Saad [31] for a summary of these methods). Like CG, these solvers generate a Krylov subspace K using only matrix-vector products with A and then enforce some minimization or orthogonality property on K; they differ primarily in how the basis of K is formed and which inner product is used to define orthogonality or minimality. Since computation and storage grow with the iteration number for these methods, they are most often used in a trun-

* Work supported by grants NSF-MIP-8410110,DOE DE-FG02-85ER25001, AT&T-AFFL-67-SAMEH, NSF-CCR-8717942,AFOSR-85-0211, NSF CCR-900000N(NCSA), and Digital Equipment Corp.

cated or restarted form. In 1982 H. Elman [14] proved that restarted versions of many of these methods converge provided that the symmetric part $(A+A^T)/2$ of A is positive definite. In spite of this restriction, the restarted GMRES(k) algorithm in particular is currently one of the more popular nonsymmetric solvers.

The second category generates residuals $r_k = b - Ax_k$ that satisfy

$$r_k = p_k(A)r_0 \tag{1}$$

where r_0 is the initial (restarted) residual and $p_k(\lambda)$ is a residual polynomial, i.e., $p_k(0) = 1$. A third category, residual polynomial methods, directly uses this idea by finding a convex region in the complex plane containing the eigenvalues of A. Iterates are then formed that satisfy (1) for some class of polynomials. Chebyshev polynomials [3, 27, 28] use ellipses for the convex region and Chebyshev polynomials to define $p_k(\lambda)$. Least squares polynomial methods [32] use an approximation of the convex hull $cvx\ (\sigma(A))$ of the extremal eigenvalues of A for the enclosing region and polynomials $p_k(\lambda)$ that minimize a weighted sum of squares on the boundary of $cvx\ (\sigma(A))$. Because of the restriction $p_k(0) = 1$, residual polynomial methods fail when the origin is in $cvx\ (\sigma(A))$.

All three categories above are thus restricted in applicability, requiring, e.g., $A+A^T$ to be positive definite or the spectrum of A to lie on one side of the imaginary axis. Symmetrization methods, the fourth category, implicitly or explicitly create a related symmetric positive definite system and then use one of the powerful iterative methods applicable to such systems. The most popular such approach is to use CG on the normal equations $A^TAx = A^Tb$. Actually forming A^TA can cause a loss of information and entail a large preprocessing cost as well as increased storage. Even if this is not done, a more serious problem is that the condition number $\kappa(A^TA)$ is the square of $\kappa(A)$. This can lead to outright failure of the solver.

In general, most nonsymmetric solvers either require storage and computation that grows excessively with the iteration number, special spectral properties of A to assure convergence, or a symmetrization process with potentially disastrous effect on the coefficient matrix. One group of methods which avoids these difficulties is that of accelerated row projection (RP) algorithms. Partition $A \in \Re^{N\times N}$ into m block rows:

$$A^T = [A_1, A_2, \ldots, A_m], \tag{2}$$

and partition the vector b conformally. A *row projection* (RP) method is any algorithm which requires the computation of the orthogonal projections $P_ix = A_i(A_i^TA_i)^{-1}A_i^Tx$ of a vector x onto range(A_i), $i = 1, 2, \ldots, m$. Note that the nonsingularity of A implies that A_i has full rank and so $(A_i^TA_i)^{-1}$ exists.

This paper presents three such methods and describes their properties. The first (KACZ) has an iteration matrix formed from the product of orthogonal projectors. A new method (V-RP) is derived from KACZ to reduce the number of matrix-vector operations needed and to reduce the problem size explicitly. The third RP method (CIMM) uses the sum of orthogonal projectors. Conjugate gradient (CG) acceleration is used on all three, and for KACZ this is shown to allow a reduction in the amount of work needed per iteration while for V-RP an error-reducing algorithm results. Most importantly, the underlying relationship between RP methods and CG applied to the normal equations is shown. This provides an explanation for the behaviour of RP methods, a basis for comparing them, and a guide for their effective use.

Possibly the most important implementation issue for RP methods is that of choosing the row partitioning which defines the projectors. An approach is shown for three dimensional elliptic partial differential equations which yields algorithms with large scale parallelism, requires only a few extra vectors of storage, and allows the computation of the necessary projections with small errors. Numerical testing shows that the algorithms have superior robustness and can be competitive with other nonsymmetric solvers in speed and efficiency.

2. Three Row Projection Methods.

2.1. Row Projection Method KACZ.

The simplest RP method can be derived geometrically. Let $H_i = \{x : A_i^T x = b_i\}$ be the affine set of solutions to the i^{th} block row of equations. The solution x^* to $Ax = b$ is the unique intersection point of those affine sets, and the method of successive projections gives the iteration

$$x_{k+1} = Q_u x_k + b_u = (I - P_m)(I - P_{m-1}) \cdots (I - P_1)x_k + b_u, \tag{3}$$

where

$$\begin{aligned} b_u \;=\; & \hat{b}_m + (I - P_m)\hat{b}_{m-1} + (I - P_m)(I - P_{m-1})\hat{b}_{m-2} + \cdots \\ & +(I - P_m) \cdots (I - P_2)\hat{b}_1, \end{aligned}$$

and $\hat{b}_i = A_i(A_i^T A_i)^{-1} b_i$.

In 1939 S. Kaczmarz [20] proposed iteration (3) and proved convergence for the $m = N$ case where each block row A_i^T consists of a single row of A. Since then many authors [10, 11, 19, 29, 30, 37] have examined the convergence of related iterative methods. The theoretical robustness of (3) is remarkable and the iteration converges even when A is singular or rectangular. However, as with any linear stationary process the rate of convergence is determined by the spectral radius of Q_u and can be arbitrarily slow. For this reason Elfving [12] and Björck and Elfving [4] proposed symmetrizing Q_u by following a forward sweep through the rows with a backward sweep, and introduced an iteration parameter to get

$$x_{k+1} = Q(\omega)x_k + T(\omega)b, \tag{4}$$

$$Q(\omega) = (I - \omega P_1)(I - \omega P_2) \cdots (I - \omega P_m)^2 \cdots (I - \omega P_2)(I - \omega P_1), \tag{5}$$

with $T(\omega)$ defined by (8). When A is nonsingular and $0 < \omega < 2$, the spectrum $\sigma(I - Q(\omega))$ lies in the interval $(0, 1]$ and so the CG method can be applied to solve

$$(I - Q(\omega))x = T(\omega)b. \tag{6}$$

Also, (4) is equivalent to applying block SSOR to

$$
\left\{ \begin{array}{rcl} AA^T y &=& b \\ x &=& A^T y \end{array} \right. \tag{7}
$$

where the blocking is that induced by the row partitioning of A [12]. This gives a simple expression for $T(\omega)$:

$$
T(\omega) = A^T (D + \omega L)^{-T} D (D + \omega L)^{-1}, \tag{8}
$$

where $AA^T = L + D + L^T$ is the usual splitting into block lower triangular, diagonal, and upper triangular parts. This also shows that solving (6) using the CG algorithm can be placed in the category of symmetrization methods.

Although Björck and Elfving tested an accelerated RP algorithm, their work concentrated on the single row ($m = N$) case applied to weighted least squares problems. Previous work dealing with the use of block forms is given by C. Kamath and A. Sameh [21, 22]. Using sample problems drawn from two dimensional nonself-adjoint elliptic partial differential equations, they numerically examined the issues of suitable row partitionings and methods for the numerical solution of the induced projections, primarily for the $m = 2$ or 3 case. By comparisons with CG-like methods and preconditioned CG applied to the normal equations they showed that the RP algorithm solved selected problems for which most of the other methods failed.

The first implementation issue is the choice of ω in (6). Normally the 'optimal' ω is defined as the $\omega_{\min}$ that minimizes the spectral radius of $Q(\omega)$. In [22] $\omega_{\min} = 1$ is proven for the case where A is partitioned into two block rows, i.e., $m = 2$. This is no longer true for $m > 3$, as can be seen by considering

$$
A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} A_1^T \\ A_2^T \\ A_3^T \end{bmatrix} .
$$

For $Q(\omega)$ defined in (4) it can be proven that

$$
\begin{array}{rcl} \rho(Q(1)) &=& (7 + \sqrt{17})/16 = 0.69519 \pm 10^{-5}, \\ \rho(Q(0.9)) &\leq& 0.68611 \pm 10^{-5} < \rho(Q(1)), \end{array}
$$

and so $\omega_{\min} \neq 1$.

Throughout this paper, however, $\omega = 1$ will be used for three reasons. Because CG acceleration is to be applied the entire distribution of the spectrum must be considered, not simply the spectral radius. When $\omega = 1$, many of the eigenvalues of the system matrix in (6) are exactly 1:

FACT 2.1. *At least rank(A_1) of the eigenvalues of $Q(1)$ are zero.*

Proof. $(I - P_1)x = 0$ for $x \in \text{range}(P_1) = \text{range}(A_1)$. From the definition of $Q(\omega)$, $\text{null}(I - P_1) \subseteq \text{null}(Q(1))$. $\square$

Since CG acceleration requires in exact arithmetic a number of iterations equal to the number of distinct eigenvalues, this suggests that numerically fewer iterations are needed as compared to when $\omega \neq 1$. A second reason is that numerical experience shows $\rho(Q(\omega))$ is not sensitive to changes in ω. This matches classical results for SSOR iterations, which are not as sensitive to ω as the corresponding SOR methods. The small improvement that does result from choosing $\omega_{\min}$ is more than offset by the introduction of extra nonzero eigenvalues. The third reason for using $\omega = 1$ is given by:

FACT 2.2. *When $\omega = 1$ and $x_0 = 0$, $A_1^T x_k = b_1$ holds in exact arithmetic on every iteration of (4).*

Proof. Using the definition $P_i = A_i(A_i^T A_i)^{-1} A_i^T$, (8) can be expanded to show that the i^{th} block column of $T(1)$ is given by

$$\prod_{j=1}^{i-1}(I - P_j)\left[I + \prod_{j=i}^{m}(I - P_j)\prod_{j=m-1}^{i+1}(I - P_j)\right](A_i^T)^{\dagger} \tag{9}$$

The first product above should be interpreted as I when $i = 1$ and the third product should be interpreted as I when $i = m-1$ and 0 when $i = m$, so that the first summand in forming $T(1)b$ is $[I + (I - P_1)\cdots(I - P_m)\cdots(I - P_2)](A_1^T)^{\dagger}b_1$. Since $A_1^T(I - P_1) = 0$, $A_1^T x_1 = A_1^T(A_1^T)^{\dagger}b_1 = b_1$. The succeding iterates x_k are obtained by adding elements in $\text{range}(Q(1)) \subseteq \text{range}(I - P_1) = \text{null}(A_1^T)$ to x_1. □

Later we show that this continues to hold when CG acceleration is used. This feature means that when $\omega = 1$ equations deemed more important than the others can be put into the first block and kept satisfied to machine precision.

The most important reason for choosing $\omega = 1$ is an *a posteriori* one; the resulting algorithm works well. For the remainder of the paper $Q = Q(1)$ and $T = T(1)$ will be used. The resulting system

$$\begin{aligned} (I - Q)x &= \tilde{b} \\ Q &= (I - P_1)(I - P_2)\cdots(I - P_m)\cdots(I - P_2)(I - P_1) \\ \tilde{b} &= Tb \end{aligned} \tag{10}$$

will be called the KACZ system.

2.2. Row Projection Method V-RP. A new RP iteration with computational and theoretical advantages over (10) is now introduced. Let $A_i = Q_i U_i$ be the QR decomposition of A_i for $i = 1, 2, \ldots, m$. Note that $P_i = Q_i Q_i^T$ in this case. Let $AA^T = L + D + L^T$ be the decomposition of AA^T into the strictly lower block triangular, block diagonal, and strictly upper block triangular parts induced by the block row partitioning. As shown in [18, p.31], the block SSOR iteration for $AA^T y = b$ can be written as

$$\begin{aligned} y_{k+1} &= [I - \omega(2-\omega)(D + \omega L)^{-T} D (D + \omega L)^{-1} AA^T] y_k \\ &+ \omega(2-\omega)(D + \omega L)^{-T} D (D + \omega L)^{-1} b \end{aligned}$$

For $\omega = 1$ this can be viewed as an iteration to solve the system

$$[(D + L)^{-T} D (D + L)^{-1} AA^T] y = (D + L)^{-T} D (D + L)^{-1} b. \tag{11}$$

The KACZ system is obtained by replacing $A^T y$ with x and premultiplying by A^T, or equivalently applying a similarity transformation with $W = A^T$ to the coefficient matrix in (11) to yield

$$[A^T(D+L)^{-T}D(D+L)^{-1}A]x = A^T(D+L)^{-T}D(D+L)^{-1}b. \tag{12}$$

In [4] applying a similarity transformation using $W = D^{-1/2}(D+L)^T$ was proposed; when CG acceleration is used this has the desirable property of being an error reducing process in the two-norm, instead of reducing the error in an elliptic norm. Unfortunately, computing $D^{-1/2}$ is only practical in the $m = N$ case where each A_i^T is a single row of A.

However, the triangular factors U_i can be found in the block case either from an orthogonal decomposition of A_i or a Cholesky factorization of $A_i^T A_i$. Let

$$Z = diag(U_1, U_2, \ldots, U_m) \in \Re^{N\times N}$$

and consider applying a similarity transformation using $W = Z^{-T}(D+L)^T$ to (11). The resulting system is

$$Z(D+L)^{-1}AA^T(D+L)^{-T}Z^T z = Z(D+L)^{-1}b \tag{13}$$

with $z = Z^{-T}(D+L)^T y$ and

$$x = A^T y = A^T(D+\omega L)^{-T}Z^T z. \tag{14}$$

The motivation for proposing this new system follows. Suppose for convenience that each A_i^T contains the same number N/m of rows of A; in this case the KACZ system has at least $1/m$ of its eigenvalues equal to 1. When CG is applied to $I - Q$, implicitly a system only $(m-1)/m$ the size of the original system is being solved. Can this reduction in problem size be made explicit? Using (13) and some algebraic manipulation, the coefficient matrix can be written as $V = S_V^T S_V$, where

$$\begin{aligned} S_V &= A^T(D+L)^{-T}Z^T \\ &= [Q_1, (I-P_1)Q_2, \ldots, (I-P_1)\cdots(I-P_{m-1})Q_m]. \end{aligned} \tag{15}$$

Since Q_1 has orthonormal columns, the (1,1) block entry of $S_V^T S_V$ is $Q_1^T Q_1 = I$, and the (1,i) block is $Q_1^T(I-P_1)\left(\prod_{j=2}^{i-1}(I-P_j)\right)Q_i$. Since $Q_1^T(I-P_1) = Q_1^T - Q_1^T Q_1 Q_1^T = 0$, and $S_V^T S_V$ is symmetric, the first row and column are zero except for the (1,1) block entry. For the case $m = 3$, e.g., this implies $I - V$ has the form

$$\begin{bmatrix} I & 0 & 0 \\ 0 & {Q_2}^T(I-P_1)Q_2 & {Q_2}^T(I-P_1)(I-P_2)Q_3 \\ 0 & {Q_3}^T(I-P_2)(I-P_1)Q_2 & {Q_3}^T(I-P_2)(I-P_1)(I-P_2)Q_3 \end{bmatrix} \tag{16}$$

Clearly the first block of unknowns requires no iterations to find, and so the CG algorithm only need be applied to a reduced system. An important point is that this reduction has no detrimental effect on the spectrum of the iteration matrix. This is because (10) and (13) have matrices with identical eigenvalues since they are similar to the system matrix in (11). The resulting system

$$
\begin{aligned}
(I-V)z &= b_V \\
V &= I - Z(D+L)^{-1}AA^T(D+L)^{-T}Z^T \\
x &= A^T(D+L)^{-T}Z^Tz \\
b_V &= Z(D+L)^{-1}b
\end{aligned}
\tag{17}
$$

will be called the the V-RP system.

Similar to KACZ, V-RP keeps the first block of equations exactly satisfied when used as a linear stationary iteration, but without requiring $x_0 = 0$.

THEOREM 2.1. *For any starting vector z_0, the iteration $z_{k+1} = Vz_k + Z(D+L)^{-1}b$ yields iterates x_k that satisfy $A_1^T x_k = b_1$, where x_k satisfies (14).*

Proof. Let the vector $z_k = (z_k^{(1)}, \ldots, z_k^{(m)})$ be partitioned conformally with the row partitioning of A. Referring to the system matrix shown in (16), note that $z_k^{(1)}$ is simply the first block of the modified right hand side $Z(D+L)^{-1}b$. $(D+L)$ is block lower triangular with its (1,1) block equal to $A_1^TA_1$, so the first block of $Z(D+L)^{-1}b$ is $z_k^{(1)} = U_1(A_1^TA_1)^{-1}b_1 = U_1(U_1^TU_1)^{-1}b_1 = U_1^{-T}b_1$. From (15), (14), $A_1 = Q_1U_1$, and $Q_1^T(I-P_1) = 0$,

$$
\begin{aligned}
A_1^Tx_k &= U_1^TQ_1^T\left[Q_1, (I-P_1)Q_2, \ldots, (I-P_1)\cdots(I-P_{m-1})Q_m\right]z_k \\
&= U_1^Tz_k^{(1)} = U_1^TU_1^{-T}b_1 = b_1.
\end{aligned}
$$

□

2.3. Row Projection Method CIMM. The third RP method can be derived as a preconditioner for the CG algorithm. Premultiply the system $Ax = b$ by $\tilde{A} = [A_1(A_1^TA_1)^{-1}, A_2(A_2^TA_2)^{-1}, \ldots, A_m(A_m^TA_m)^{-1}]$ to obtain

$$
(P_1 + P_2 + \cdots + P_m)x = \tilde{A}b. \tag{18}
$$

This system can also be derived as a block Jacobi method applied to the system (7); see [11]. For nonsingular A this system is symmetric positive definite and the CG algorithm can be applied. The advantage of this approach is that the projections can be computed in parallel and then added. In 1939 G. Cimmino [9] first proposed an iteration related to (18), and since then it has been examined by several others [1, 11, 12, 15, 25, 26, 38]. Later we will show how the individual projections can, for a wide class of problems, be computed with parallelism. In this case, the Cimmino method allows computations to proceed with two levels of parallelism, making it especially suitable for hierarchical memory machines such as Cedar [24].

2.4. Connection Between RP Systems and the Normal Equations. Although KACZ and V-RP can be derived as a block SSOR, and CIMM as a block Jacobi, method for (7), a more instructive comparison can be made with CGNE, conjugate gradients applied to the normal equations $A^TAx = A^Tb$. All four methods

Method	W^T
CGNE	$[A_1, \ A_2, \ \cdots \ , A_m]$
CIMM	$[Q_1, \ Q_2, \ \cdots \ , Q_m]$
KACZ	$\left[P_1, \ (I-P_1)P_2, \ (I-P_1)(I-P_2)P_3, \ \cdots \ , \prod_{i=1}^{m-1}(I-P_i)P_m\right]$
V-RP	$\left[Q_1, \ (I-P_1)Q_2, \ (I-P_1)(I-P_2)Q_3, \ \cdots \ , \prod_{i=1}^{m-1}(I-P_i)Q_m\right]$

TABLE 1

Comparison of System Matrices for Four Methods

consist of CG applied to a system with coefficient matrix W^TW, where W^T is shown for the four methods in Table 1. Intuitively an ill conditioned matrix A has some linear combination of rows approximately equal to the zero vector. For a row partitioned matrix, near linear dependence may occur *within* the blocks, that is, some linear combination of the rows within a particular block is approximately zero, or *across* the blocks, that is, the linear combination must draw from more than one block row A_i^T. Now let $A_i = Q_iU_i$ be the orthogonal decomposition of A_i, and note that both Q_i and $P_i = Q_iQ_i^T$ have the perfect condition number of 1. Examining the matrices W shows that CGNE works on A^TA, and the ability to form a near linear dependence from both within and across blocks enters into the system matrix. CIMM replaces each A_i with Q_i, which has orthonormal columns. Hence CIMM removes the ability to form linear dependence within the blocks, but has no effect on that between the blocks. V-RP also replaces each A_i with the perfectly conditioned Q_i and then goes a step further by recursively orthogonalizing between blocks in the following way: The first block column of W^T is orthogonal to the others since $Q_1(I - P_1) = 0$. If the orthogonalizing $I - P_1$ factor is removed from the other block columns of W^T, then the second block column is orthogonal to the remaining ones since $Q_2(I - P_2) = 0$. This process continues until the last block is reached. The recursive nature of this partial orthogonalization can be seen by rewriting W^T for V-RP as

$$W^T = [Q_1, \quad (I-P_1)[Q_2, \quad (I-P_2)[Q_3, \quad (I-P_3)\cdots[Q_{m-1}, \quad (I-P_{m-1})Q_m]\cdots]]].$$

KACZ has the same effect as V-RP, since $P_i^T(I - P_i) = 0$ in the same way that $Q_i(I - P_i) = 0$.

Several implications follow from this heuristic argument. First, the system matrix spectrum for KACZ and V-RP should be better than that of CIMM which turn should be better than that of CGNE, where ‘better’ means having fewer small eigenvalues and many more eigenvalues near the maximal one. Section 4 will show that for $m = 2$ the first comparison is true, and the second comparison holds if condition numbers are used to measure ‘better’. Furthermore, by computing the spectra for small grid sizes we have found these comparisons are valid for the problems in Section 6, where $m = 9$.

A second implication of this argument is that RP methods will require fewer iterations for matrices A where the near linear dependence comes primarily from within a block row rather than between block rows. A third implication of the heuristic argument for the relative performance of RP methods is that the number of block rows should be kept as small as possible. The reason is that the partial orthogonalization between blocks in the W of Table 1 becomes less effective as more block rows appear. In terms of the heuristic, progressively more orthogonalizing factors $I - P_i$ must be stripped away before the orthogonalization effect between block row i and the succeeding block rows is felt. Keeping the number of block rows small can also be seen to be important because, e.g., for the $m = N$ case all of the ability to form a near linear dependency between rows of the system matrix occurs between the block rows, where the outer CG acceleration method must deal with it.

3. CG Acceleration. Although the CG algorithm can be applied directly to the row projection systems, special properties allow a reduction in the amount of work required by KACZ and provide another advantage of V-RP. CG acceleration for RP methods was proposed in [4] and tested in [7, 21, 22]. The reason that a reduction in work is possible and $A_1^T x_k = b_1$ on every iteration of CG applied to KACZ follows from:

THEOREM 3.1. *Suppose that the CG algorithm is applied to the KACZ system, let* $r_k = \tilde{b} - (I - Q)x_k$ *be the residual and* d_k *be the search direction. If* $x_0 = \tilde{b}$ *is chosen as the starting vector, then* r_k, $d_k \in$ *range*$(I - P_1)$ *for all* k.

Proof.

$$\begin{aligned} r_0 &= \tilde{b} - (I - Q)\tilde{b} = Q\tilde{b} \\ &= (I - P_1)(I - P_2)\cdots(I - P_m)\cdots(I - P_2)(I - P_1)\tilde{b} \in \text{range}(I - P_1) \end{aligned}$$

Since $d_0 = r_0$, the same is true for d_0. Suppose the theorem holds for $k - 1$. Then $d_{k-1} = (I - P_1)d_{k-1}$ and so

$$\begin{aligned} w_k &\equiv (I - Q)d_{k-1} \\ &= (I - P_1)d_{k-1} - (I - P_1)(I - P_2)\cdots(I - P_m)\cdots(I - P_2)(I - P_1)d_{k-1} \\ &\in \text{range}(I - P_1) \end{aligned}$$

Since r_k is a linear combination of r_{k-1} and w_k, $r_k \in \text{range}(I - P_1)$. Since d_k is a linear combination of d_{k-1} and r_k, $d_k \in \text{range}(I - P_1)$. The result follows by induction. $\square$

This reduces the requisite number of projections from $2m - 1$ to $2m - 2$ because the first multiplication by $(I - P_1)$ when forming $(I - Q)d_k$ can be omitted. For V-RP, the next section will show that the algorithm can be implemented so that effectively only $2m - 2$ projections are required on each iteration, for any starting vector. Using $x_0 = \tilde{b}$ for KACZ also keeps the first block of equations satisfied in exact arithmetic:

COROLLARY 3.2. *If* $x_0 = \tilde{b}$ *is chosen in the CG algorithm applied to the KACZ system, then* $A_1^T x_k = b_1$ *for all* $k \geq 0$.

Proof. The proof of Fact 2.2 shows that $\tilde{b} \in (A_1^T)^{\dagger} b_1 + \text{range}(I - P_1)$. Since $A_1^T(I - P_1) = A_1^T(I - A_1 A_1^{\dagger}) = 0$, $A_1^T x_0 = A_1^T (A_1^T)^{\dagger} b_1 = b_1$ because A_1 has full column rank. For $k > 0$, $d_{k-1} \in \text{range}(I - P_1)$, so

$$A_1^T x_k = A_1^T(x_{k-1} + \alpha_k d_{k-1}) = A_1^T x_{k-1} = \cdots = A_1^T x_0 = b_1.$$

$\square$

V-RP has already been shown to allow an explicit reduction of the problem size. Its second major advantage is that when accelerated by CG, the resulting algorithm is error-minimizing in the two-norm:

THEOREM 3.3. *Suppose CG acceleration is applied to the V-RP system and let x^* be the solution to $Ax = b$. Then the k^{th} iteration minimizes $\| x - x^* \|$ over all $x \in S_V^T * \text{span}[x_0 - x^*, \ldots, x_{k-1} - x^*]$, where S_V is defined by (15).*

Proof. Recall that $x_k = A^T(D+L)^{-T}Z^T z_k = S_V z_k$, where $V = I - S_V^T S_V$. The CG algorithm minimizes the quadratic form ([16])

$$\begin{aligned} z^T(I-V)z - 2z^T b_V &= z^T S_V^T S_V z - 2z^T S_V^T A^{-1} b \\ &= x^T x - 2x^T x^* = \| x - x^* \|^2 - \| x^* \|^2 . \end{aligned}$$

Since $\| x^* \|^2$ is a constant, this is equivalent to minimizing $\| x - x^* \|^2$. The subspace over which this minimization occurs is

$$\begin{aligned} \text{span}[r_0, \ldots, r_{k-1}] &= \text{span}[S_V^T A^{-1} b - S_V^T S_V z_0, \ldots, S_V^T A^{-1} b - S_V^T S_V z_{k-1}] \\ &= S_V^T * \text{span}[x^* - x_0, \ldots, x^* - x_{k-1}] \end{aligned}$$

□

In contrast CG acceleration in the KACZ system minimizes $\| S_V^T(x_k - x^*) \|$ over the subspace $S_V S_V^T * \text{span}[x_0 - x^*, \ldots, x_{k-1} - x^*]$, a result established in [4].

In summary, CG acceleration for KACZ allows one projection per iteration to be omitted and one block of equations to be kept exactly satisfied, provided that $x_0 = \tilde{b}$ is used. CG acceleration for V-RP minimizes the two-norm rather than the A-norm of the error.

4. The $m = 2$ Case. When $m = 2$ is chosen the matrix A is partitioned into two block rows, and a complete eigenanalysis of RP methods is possible using the concept of the angles θ_k between the two subspaces $L_i = \text{range}(A_i)$, $i = 1, 2$. The definition presented here follows that of A. Björck and G. Golub in [5], but for convenience L_1 and L_2 are assumed to have the same dimension. The smallest angle $\theta_1 \in [0, \pi/2]$ between L_1 and L_2 is defined by

$$\begin{aligned} &\cos\theta_1 = \max_{u \in L_1} \max_{v \in L_2} u^T v \\ &\text{subject to } \| u \| = \| v \| = 1. \end{aligned}$$

Let u_1 and v_1 be the attainment vectors; then for $k = 2, 3, \ldots, N/2$ the remaining angles between the two subspaces are defined as

$$\begin{aligned} &\cos\theta_k = \max_{u \in L_1} \max_{v \in L_2} u^T v \\ &\text{subject to} \left\{ \begin{array}{rcl} \| u \| &=& \| v \| = 1 \\ u_i^T u &=& v_i^T v = 0, \quad i = 1, 2, \ldots, k-1. \end{array} \right. \end{aligned}$$

Furthermore, when u_i and v_j are defined as above, $u_i^T v_j = 0$, $i \neq j$ also holds. From this one can obtain the CS decomposition [16, 36], which is stated below in terms of the projectors P_i.

THEOREM 4.1 (CS DECOMPOSITION). *Let $P_i \in \Re^{N\times N}$ be the orthogonal projectors onto the subspaces L_i, for $i = 1, 2$. Then there are orthogonal matrices U_1 and U_2 such that*

$$P_1 = U_1 \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} U_1^T, \quad P_2 = U_2 \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} U_2^T, \quad U_1^T U_2 = \begin{bmatrix} C & -S \\ S & C \end{bmatrix},$$

$$\begin{aligned} C &= diag(c_1, c_2, \ldots, c_{N/2}) \\ S &= diag(s_1, s_2, \ldots, s_{N/2}) \\ I &= C^2 + S^2 \\ 1 &\geq c_1 \geq c_2 \geq \cdots \geq c_{N/2} \geq 0. \end{aligned} \tag{19}$$

In the theorem $c_k = \cos\theta_k$ and $s_k = \sin\theta_k$, where the angles θ_k are defined above. Now consider the nonsymmetric RP iteration matrix $Q_u = (I - \omega P_1)(I - \omega P_2)$. Letting $\alpha = 1 - \omega$, and substituting in the expressions for P_1 and P_2 from the CS decomposition gives

$$Q_u = U_1 \begin{bmatrix} \alpha^2 C & -\alpha S \\ \alpha S & C \end{bmatrix} U_2^T$$

Applying a similiarity transformation using the matrix U_2^T gives

$$U_2^T Q_u U_2 = \begin{bmatrix} \alpha^2 C^2 + \alpha S^2 & (1-\alpha)CS \\ \alpha(1-\alpha)CS & C^2 + \alpha S^2 \end{bmatrix} \tag{20}$$

Since each block in the above 2×2 matrix is diagonal, $U_2^T Q_u U_2$ has the same eigenvalues as its scalar 2×2 principal submatrices. These eigenvalues are given by

$$(1-\alpha)^2 c_i^2 + 2\alpha \pm |1-\alpha| c_i \sqrt{(1-\alpha)^2 c_i^2 + 4\alpha} \tag{21}$$

for $i = 1, 2, \ldots, N/2$. For a given α the modulus of this expression is a maximum when c_i is largest, i.e., when $c_i = c_1$. The spectral radius of Q_u can then be found by taking $c_i = c_1$ and the positive sign in (21). Doing so and minimizing with respect to α gives $\omega_{\min} = 1 - \alpha_{\min} = 2/(1 + s_1)$ and a spectral radius of $(1 - s_1)/(1 + s_1)$. The same result was derived in [12] using classical SOR theory. The benefit of the CS decomposition is that the full spectrum of Q_u is given and not simply the eigenvalue of largest modulus. In particular when $\omega = 1$ the eigenvalues of the nonsymmetric RP iteration matrix Q_u become $\left\{c_1^2, c_2^2, \ldots, c_{N/2}^2, 0\right\}$ with 0 repeated $N/2$ times. Now applying the CS decomposition to the symmetrized RP iteration matrix $Q(\omega)$ gives

$$\begin{aligned} Q(\omega) &= (I - \omega P_1)(I - \omega P_2)(I - \omega P_1) \\ &= U_1 \begin{bmatrix} \alpha^2(\alpha C^2 + S^2) & \alpha(\alpha - 1)CS \\ \alpha(\alpha-1)CS & \alpha S^2 + C^2 \end{bmatrix} U_1^T, \end{aligned} \tag{22}$$

where again $\alpha = 1 - \omega$. When $\omega = 1$, the eigenvalues of the symmetrized matrix are identical to those of the unsymmetrized matrix Q_u. One objection to the symmetrization process is that $Q(\omega)$ requires three projections while Q_u only needs two. The previous section, however, showed that when $\omega = 1$ KACZ can be implemented with only two projections per iteration.

When $m = 2$ the value $\omega = 1$ minimizes the spectral radius of $Q(\omega)$ as was shown in [22]. This result can be obtained from the representation (22) in the same way as (20) was obtained.

The CS decomposition also allows the construction of an example showing that no direct relationship need exist between the singular value distribution of A and its related RP matrices. Define

$$A = \begin{bmatrix} A_1^T \\ A_2^T \end{bmatrix} = \begin{bmatrix} 0 & D \\ S & C \end{bmatrix} \tag{23}$$

where each block is $N/2 \times N/2$, C and S satisfy (19), and $D = diag(d_1, d_2, \ldots, d_{N/2})$. The projectors become $P_1 = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$ and $P_2 = \begin{bmatrix} C \\ S \end{bmatrix} \begin{bmatrix} C & S \end{bmatrix}$, and the eigenvalues of $I - Q(1)$ are $\{s_1^2, s_2^2, \ldots, s_{N/2}^2, 1\}$ while those of A are $\left[c_i \pm \sqrt{c_i^2 + 4 s_i d_i}\right]/2$. A is nonsingular iff each s_i and d_i is nonzero. If the s_i's are chosen to be close to 1 while the d_i's are chosen to be close to 0, $I - Q(1)$ has eigenvalues clustered near 1 while A has singular values close to both 0 and 1. Hence A is badly conditioned while $I - Q(1)$ is very well-conditioned. Conversely if the d_i's are chosen to be large while the s_i's are close to 0 in such a way that $d_i s_i$ is near 1, then A is well conditioned while $I - Q(1)$ is badly conditioned. Hence the conditioning of A and its induced RP system matrix may differ greatly. Although this example is contrived, Section 7 presents two PDE's with induced matrices A that have identical singular values but drastically differing spectra for the corresponding RP matrices.

In Section 2.4 a heuristic argument was given implying that the eigenvalue distribution for the KACZ and V-RP systems is better for CG acceleration than that of CIMM, which in turn is better than that of CGNE, CG applied to $A^T A x = A^T b$. For $m = 2$, the eigenvalues of KACZ and V-RP are $\{s_1^2, s_2^2, \ldots, s_{N/2}^2, 1\}$ while those of CIMM are easily seen to be $\{1 - c_1, \ldots, 1 - c_{N/2}, 1 + c_{N/2}, \ldots, 1 + c_1\}$, verifying that KACZ and V-RP have better spectral distribution than CIMM. The next theorem shows that in terms of condition numbers the heuristic argument is valid when $m = 2$.

THEOREM 4.2. *When* $m = 2$, $\kappa(A^T A) \geq \kappa(P_1 + P_2) \geq \kappa(I - (I - P_1)(I - P_2)(I - P_1))$. *Furthermore, let* $c_1 = \cos\theta_1$ *be the canonical cosine corresponding to the smallest angle between range(*A_1*) and range(*A_2*). Then* $\kappa(P_1 + P_2) = (1 + c_1)^2 \kappa(I - (I - P_1)(I - P_2)(I - P_1))$.

Proof. Without loss of generality suppose that range(A_1) and range(A_2) have dimension $N/2$. Let $U_1 = [G_1, G_2]$ and $U_2 = [H_1, H_2]$ be the matrices defined in Theorem 4.1 so that $P_1 = G_1 G_1^T$, $P_2 = H_1 H_1^T$, and $G_1^T H_1 = C$. Set $X = G_1^T A_1$ and $Y = H_1^T A_2$ so that $A_1 = P_1 A_1 = G_1 G_1^T A_1 = G_1 X$ and $A_2 = H_1 Y$. It is easily verified that the eigenvalues of $P_1 + P_2$ are $1 \pm c_i$, corresponding to the eigenvectors $g_i \pm h_i$ where $G_1 = [g_1, g_2, \ldots, g_{N/2}]$ and $H_1 = [h_1, h_2, \ldots, h_{N/2}]$. Furthermore, $G_1 g_1 = H_1 h_1 = e_1$ and $G_1 h_1 = H_1 g_1 = c_1 e_1$, where e_1 is the first unit vector. Then $A^T A = A_1 A_1^T + A_2 A_2^T = G_1 X X^T G_1^T + H_1 Y Y^T H_1^T$, so $(g_1 + h_1)^T (A^T A)(g_1 + h_1) = (1 + c_1)^2 e_1^T (A^T A) e_1$ and

$(g_1 - h_1)^T(A^TA)(g_1 - h_1) = (1 - c_1)^2 e_1^T(A^TA)e_1$. The minimax characterization of eigenvalues gives

$$\begin{aligned}\lambda_{\max}(A^TA) &\geq (1+c_1)^2 e_1^T(A^TA)e_1/(g_1+h_1)^T(g_1+h_1)\\ &= (1+c_1)^2 e_1^T(A^TA)e_1/2(1+c_1) = (1+c_1)e_1^T(A^TA)e_1/2\end{aligned}$$

and $\lambda_{\min}(A^TA) \leq (1-c_1)e_1^T(A^TA)e_1/2$. Hence $\kappa(A^TA) \geq \dfrac{1+c_1}{1-c_1} = \kappa(P_1+P_2)$, proving the first inequality. For the second, from the CS Decomposition the eigenvalues of $(I-(I-P_1)(I-P_2)(I-P_1))$ are $\{s_1^2, s_2^2, \ldots, s_{N/2}^2, 1\}$ where $s_i^2 = 1 - c_i^2$ are the canonical sines and 1 is of multiplicity $N/2$. Then

$$\frac{\kappa(P_1+P_2)}{\kappa(I-(I-P_1)(I-P_2)(I-P_1))} = \frac{1+c_1}{1-c_1} \div \frac{1}{s_1^2} = (1+c_1)^2.$$

$\square$

Note that $(1+c_1)^2$ is a measure of the lack of orthogonality between range(A_1) and range(A_2), and so measures the partial orthogonalization effect described in Section 2.4.

5. Implementation for 3D Elliptic PDE's. The primary implementation issue for RP methods is how to partition the rows of A. Since any row partitioning gives a convergent algorithm, the criteria for choosing one can be based on computational considerations. This section describes the set of test problems, outlines the row partitioning criteria, and presents the row partitioning used in the testing.

5.1. Test Problems. Test problems are obtained from the seven point centered difference operator [35] for elliptic partial differential equations

$$au_{xx} + bu_{yy} + cu_{zz} + du_x + eu_y + fu_z + gu = F \tag{24}$$

where $a - g$ are functions of (x, y, z) and the domain is the unit cube $[0,1] \times [0,1] \times [0,1]$. Dirichlet boundary conditions are imposed in a manner described later. When discretized using n grid points in each direction the resulting system is of order $N = n^3$.

This class of test problems is chosen for four reasons. First, it includes important applications such as computational fluid dynamics and structural mechanics. Second, by refining the mesh the problem size grows rapidly with n, allowing test problems that can be scaled up to realistic sizes. Results are shown for problems of order 216000, corresponding to a $60 \times 60 \times 60$ mesh. Third, by selecting the coefficient functions of the partial differential equation, matrices A can be created with or without properties such as diagonal dominance, definite symmetric part, eigenvalues on both sides of the imaginary axis, or extreme ill conditioning. Fourth, the problems provide a specific example on how to select m, partition the rows into blocks, and analyze the subproblems that define the projectors.

The collection of test problems and their predetermined solutions follows. These are generalized versions of two-dimensional problems from a variety of sources

Problem:	1	2	3	4	5	6
$\mathrm{Re}(\lambda) > 0$	Yes	Yes	No	Yes	Yes	Yes
$A + A^T$ Definite	Yes	No	No	No	No	No
Estimated $\kappa(A)$	9	57	40000	4786	36	77

TABLE 2
Properties of A for $N = 1728$

[13, 22, 23, 34]. Each test problem has boundary conditions chosen to give a predetermined solution to the partial differential equation, so that the norm of the error vector as well as that of the residual vector can be checked.

Problem 1: $\triangle u + 1000 u_x = F$
Problem 2: $\triangle u + 1000 \exp(xyz)(u_x + u_y - u_z) = F$
Problem 3: $\triangle u + 100 x u_x - y u_y + z u_z + 100(x + y + z)u/xyz = F$
Problem 4: $\triangle u - 10^5 x^2 (u_x + u_y + u_z) = F$
Problem 5: $\triangle u - 1000(1 + x^2)u_x + 100(u_y + u_z) = F$
Problem 6: $\triangle u - 1000\,[(1 - 2x)u_x + (1 - 2y)u_y + (1 - 2z)u_z] = F$

Problem 1 has the preassigned solution $u = xyz(1 - x)(1 - y)(1 - z)$, problem 2 has $u = x + y + z$, and problems 3-6 have $u = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$. These test problems exhibit properties that cause difficulties for nonsymmetric iterative solvers, as summarized in Table 2 for a $12 \times 12 \times 12$ grid. 'Re $(\lambda) > 0$' indicates whether or not the eigenvalues lie in the right half plane of the complex plane; if this does not hold then residual polynomial methods will fail. '$A + A^T$ definite' indicates whether or not the symmetric part of A is definite; if the answer is yes then the sufficient conditions for GMRES(k) to converge hold. $\kappa(A)$ is an estimate of the condition number obtained by applying the LINPACK routine DPOCO to $A^T A$.

5.2. Row Partitioning Goals. The first criterion for a row partitioning is that the projections $P_i x = A_i(A_i^T A_i)^{-1} A_i^T x$ must be efficiently computable. One way to achieve this is through parallelism: if A_i^T is the direct sum of blocks C_j for $j \in S_i$, then P_i is block diagonal [22]. The computation of $P_i x$ can then be done by assigning each block of P_i to a different processor on a multiprocessor machine, or by vectorizing the computations across the blocks on a vector machine. The second criterion is storage efficiency. The additional storage should be $\mathcal{O}(N)$, that is, a few extra vectors, the number of which must not grow with increasing problem size N. The third criterion is that some bound on the condition number of the subproblems induced by the projections should be provided. The need for this is clear when $m = 1$ and A is partitioned into a single block of rows; in this case KACZ simply solves the normal equations $A^T A x = A^T b$, an approach that can fail if $\kappa(A)$ is large. More generally when $m > 1$, computing $y = P_i x$ requires solving a system of the form $A_i^T A_i v = w$. Hence the accuracy with which P_i can be computed depends on $\kappa(A_i^T A_i)$ and potentially large errors might occur in the projections if that condition number is large. A practical and cheap numerical way of bounding $\kappa(A_i^T A_i)$ for a given problem should be available. The fourth criterion is that the number m of projectors should be kept as small as possible, and should not

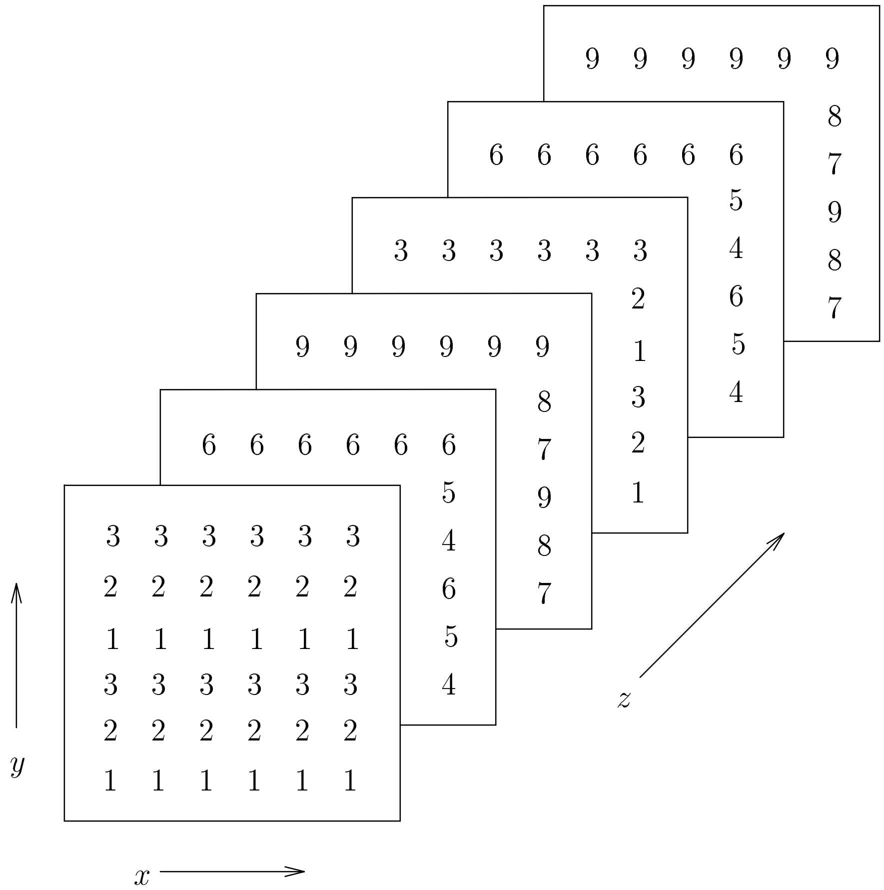

FIG. 1. *Row Partitioning for* $m = 9$

depend on N. One reason was presented in Fact 2.1: when the block rows have equal numbers of rows of A, $1/m$ of the eigenvalues of the system matrix are exactly 1. For V-RP the possible reduction in problem size also decreases with increasing m.

In summary, the row partitioning should allow parallelism in the computations, require at most $\mathcal{O}(N)$ storage, give well conditioned subproblems, and have m a small constant. One of the more useful results of this paper is that all four goals can be achieved simultaneously for important classes of problems.

5.3. Row Partitioning of Test Problems. Row partitionings schemes ranging from $m = 4$ up to $m = 27$ have been tested on the test problems. For brevity only one, a $m = 9$ partitioning generalized from a scheme in [22], is presented. Consider an $n \times n \times n$ mesh, and for convenience assume that n is a multiple of 3 and that lines in the mesh are numbered within each plane in the natural order. Place equations corresponding to nodes on lines $1, 4, 7, \ldots, n-2$ on the planes numbered $1, 4, 7, \ldots, n-2$ into the first block row A_1^T. Then assign those on lines $2, 5, 8, \ldots, n-1$ on the planes numbered $1, 4, 7, \ldots, n-2$ into the second block row A_2^T. In general, A_i^T consists of nodes on lines $l, l+3, l+6, \ldots$ on planes $p, p+3, p+6, \ldots$ where $l = (i+2) \bmod 3 + 1$ and $p = \lfloor \frac{i-1}{3} \rfloor + 1$. Figure 1 shows this scheme for a $6 \times 6 \times 6$ grid, with each node marked by the number i of its assigned block row A_i^T.

Because each line of nodes assigned to A_i^T is separated from the others by at least two other lines or planes, their seven point difference stars do not intersect and so A_i^T consists of $n^2/9$ separate subblocks of n coupled equations each. Each projector P_i can then be computed with a parallelism of $n^2/9$. What of the other row partitioning criteria? Let the subblocks of A_i^T be denoted C_j^T, for $j \in S_i$ where S_i is an index set of cardinality $n^2/9$. Dropping the subscripts temporarily, a typical C_j^T has the form

$$C^T = [0, D_1, 0, D_2, T, D_3, 0, D_4, 0] \tag{25}$$

with D_i a diagonal and T a tridiagonal matrix. The matrix C^TC is positive definite because when A is nonsingular each subblock is necessarily of full rank n. More importantly, C^TC is pentdiagonal and so its Cholesky factor R consists of 3 diagonals, and the Cholesky factors for all of the subblocks for all of the block rows A_i^T can be stored using only 3 additional vectors.

Surprisingly, the third requirement that each subproblem be well-conditioned also is generally satisfied by the $m = 9$ partitioning. The details can be found in [6], but this can be seen intuitively because C^TC consists of the normal equations of T in (25), with the squares of the diagonal blocks D_i added to the main diagonal of T^TT, making it strongly diagonally dominant. This suggests that good bounds for the extremal eigenvalues of C^TC can be obtained from Gerschgorin estimates. However, for some problems the Gerschgorin Disk Theorem provides an estimate $\lambda_{min}(C^TC) \leq 0$. In these cases a better bound on the smallest eigenvalue comes from an application of the minimax characterization of singular values:

THEOREM 5.1. *Suppose that* $C^T = [D_1\ 0\ D_2\ T\ D_3\ 0\ D_4]$ *where each* $D_i \in \Re^{n\times n}$ *is diagonal, and let* $\delta_i = \min_k |D_i|_{kk}$. *Then*

$$\sigma_{min}^2(C) = \lambda_{min}(C^TC) \geq \delta_1^2 + \delta_2^2 + \delta_3^2 + \delta_4^2.$$

Proof. Let $z \in \mathcal{R}^n$, $\| z \| = 1$. Then

$$\begin{aligned} \| Cz \|^2 &= \| D_1z \|^2 + \| D_2z \|^2 + \| Tz \|^2 + \| D_3z \|^2 + \| D_4z \|^2 \\ &\geq \| D_1z \|^2 + \| D_2z \|^2 + \| D_3z \|^2 + \| D_4z \|^2 \geq \delta_1^2 + \delta_2^2 + \delta_3^2 + \delta_4^2. \end{aligned} \tag{26}$$

Minimizing $\| Cz \|$ over all such z gives the stated result. □

When C corresponds to a line of nodes at the top or bottom of a plane, or is within the first or last plane of the grid, one or more of the δ_i's is zero but the theorem is still valid. Table 3 shows results for the test problems on a $48 \times 48 \times 48$ grid with the Gerschgorin estimate of the smallest singular value replaced by that of Theorem 5.1 when it is larger. The actual condition numbers are calculated using routines from Eispack. Clearly the subproblems are well-conditioned, so the row partitioning with $m = 9$ satisfies all of the criteria for a suitable row partitioning.

The testing results show that the approach taken here for partitioning A is extremely effective. However, it is of limited use if applicable only to seven point difference operator matrices. The reasoning for selecting the partitioning is based on a natural

Problem	1	2	3	4	5	6
Estimated	7.94	1.92	74.9	1608	12.9	2.19
Actual	4.93	1.87	61.2	518	8.56	2.13

TABLE 3
$\max_j \kappa(C_j)$ *for* $n = 48$

decoupling induced by the computational molecule used. This decoupling still occurs if, e.g., the domain is irregular or a 27-point difference operator is applied. Extensions to Neumann or periodic boundary conditions are also possible, again by considering the decoupling available on the mesh. Although more complicated, such a decoupling occurs with finite element methods applied to partial differential equations since usually the elements are chosen to have common support with only a few other elements. What determines the storage requirements of $3N$ for all of the Cholesky factors is that the longest line in the computational star contains 3 nodes. More generally, if the longest line in the computational star has l nodes, the Cholesky factors require only lN storage. The scheme used here for row partitioning can thus be extended to other, less simple, problems.

5.4. Other Approaches. Recently, M. Arioli, I. Duff, J. Noailles, and D. Ruiz [2] experimented with a block Cimmino method for more general sparse systems. Their approach is to reorder the matrix into a block tridiagonal form, and then to use a row partitioning into two block rows. In this case the goal is not to have equal sized blocks for load balancing, but rather to have few nonzero columns in the blocks of A_1^T shared with the nonzero columns in the blocks of A_2^T. This allows the use of a novel block diagonal right preconditioner which damps out the overlap between the shared nonzero columns and thereby increases the angles between the range of A_1 and A_2. This reduces the number of CG iterations (c.f. Section 2.4) but at the cost of making the subproblems more ill-conditioned. Arioli *et. al.* then handle this by using a direct method applied to an augmented system for solving the subproblems.

5.5. Implementation Details. The crucial operations in the RP algorithms are the preprocessing stage, computation of the modified right hand side vectors, and forming the matrix-vector products needed for the acceleration schemes. V-RP also requires a post-processing stage to recover the solution x from the auxillary unknown z. For convenience an $n \times n \times n$ grid is used, with n a multiple of 3.

Preprocessing for RP methods consists of computing the necessary Cholesky factors R_j. As section 5.3 showed, each C_j is well-conditioned, so the procedure is to explicitly form the normal equations and then perform an $\mathrm{LDL^T}$ factorization. Recalling that S_i is the set of indices j for which C_j is a subblock of A_i, this stage consists of $n^2/9$ parallel tasks:

Cholesky Factorization:

For $i = 1, 2, \ldots, 9$ and $j \in S_i$ *(in parallel)*

Form $R_j = (C_j)^T(C_j)$ *(vectorized)*
Perform $\mathrm{LDL^T}$ factorization on R_j, overwriting R_j with the result. *(sequential)*
End For

R_j is stored as three vectors, and the diagonal is stored as $(D_j)^{-1}$ so that the backsolves can be performed using multiplication rather than division. The $\mathrm{LDL^T}$ factorization is an inherently sequential process, unsuitable for vector machines. For those computers the order of the loops is reversed, that is, the operations are vectorized *across* the blocks C_j rather than parallelized between the blocks.

To compute the KACZ modified right hand side $\tilde{b}$, let $U_i = \mathrm{diag}(R_{j_1}, R_{j_2}, \ldots, R_{j_{n^2/9}})$, for $j_k \in S_i$. By overwriting $\tilde{b}$ can be found using a temporary vector $w^T = (w_1^T, w_2^T, \ldots, w_m^T)$ of length N and $4m-2$ triangular solves with U_i or U_i^T, $2m-2$ multiplications by U_i or U_i^T, $3m$ multiplications by A_i and $m^2 - m$ by A_i^T. The predominant expense is the triangular solves. Furthermore, each of the matrix-vector operations is implemented with $n^2/9$ parallelism.

Computation of $\tilde{b}$ from b for RP-m:

$w = b$
For $i = 1, 2, \ldots, m$
    $w_i = (U_i^T U_i)^{-1} w_i$
    $\tilde{b} = A_i w_i$
    For $j = i+1, i+2, \ldots, m$
        $w_j = w_j - A_j^T \tilde{b}$
    End For
End For
For $i = 1, 2, \ldots, m-1$
    $w_i = U_i^T U_i w_i$
End For
For $i = m, m-1, \ldots, 1$
    $\tilde{b} = A_i w_i$
    For $j = 1, 2, \ldots, i-1$
        $w_j = w_j - A_j^T \tilde{b}$
    End For
    If $i > 1$, $w_{i-1} = (U_{i-1}^T U_{i-1})^{-1} w_{i-1}$
End For
$\tilde{b} = A_1 w_1$
For $i = 2, 3, \ldots, m$
    $\tilde{b} = \tilde{b} + A_i w_i$
End For

The V-RP modified right hand side b_V can be computed at a slightly lower cost, requiring $3m-2$ triangular backsolves by U_i^T or U_i.

Computation of b_V from b for V-RP:

For $i = 1, 2, \ldots, m$
$\quad (b_V)_i = U_i^{-T} b_i$
End For
$w = A_m U_m^{-1} (b_V)_m$
For $i = m-1, m-2, \ldots, 2$
$\quad (b_V)_i = (b_V)_i - U_i^{-T} A_i^T w$
$\quad w = w + A_i U_i^{-1} (b_V)_i$
End For
$(b_V)_1 = (b_V)_1 - U_1^{-T} A_1^T w$

For KACZ the matrix-vector products needed by CG cost $4m-2$ triangular solves with U_i or U_i^T and $4m-2$ multiplications by A_i or A_i^T. For CIMM, this number is $2m$. V-RP can be implemented with $4m-5$ of each operation, if the following algorithm is used:

Computation of $v = (I - V)d$ for V-RP:

$w = A_m U_m^{-1} d_m$
For $i = m-1, \ldots, 3, 2$
$\quad w = w + A_i U_i^{-1} (d_i - U_i^{-T} A_i^T w)$
End For
$w = w - A_1 (U_1^T U_1)^{-1} A_1^T w$
For $i = 2, \ldots, m-1$
$\quad v_i = U_i^{-T} A_i^T w$
$\quad w = w - A_i U_i^{-1} v_i$
End For
$v_m = U_m^{-T} A_m^T w$

Note that this only uses the block components numbered from 2 to m of the vectors d and v, reflecting the explicit reduction in the problem size allowed by V-RP. V-RP must also retrieve x from the auxillary vector z, and this requires $2m-1$ triangular solves and $2m-2$ multiplications by A_i or A_i^T.

6. Testing Results. Tests were run on three machines to examine different aspects of the algorithms. First, problems of order $N = 13824$ were run on one processor of the University of Illinois' National Center for Supercomputing Application's Cray X-MP in order to test the robustness and vectorization of the methods. Then problems of order $N = 216000$ were tested on one processor of the NCSA Cray-2 to verify that the results are valid for larger problem sizes. Finally the RP methods were tested on an eight-processor Alliant FX/8 at the Center for Supercomputing Research and Development to demonstrate that the algorithms can be implemented efficiently on a shared memory multiprocessor. This section describes the other nonsymmetric solvers to which the RP methods are compared, the stopping tests used, and briefly summarizes the results comparing the methods. Fuller details of the results, including tables showing numbers of iterations, times, and residual and error norms are provided in [6].

6.1. Description of Other Methods Tested. The RP algorithms are compared to two other basic methods, GMRES(10) (see [31] for references to this and the other Krylov subspace methods of this section) and CGNE, CG applied to $A^T Ax = A^T b$. GMRES(10) is from the PCGPAK library and has been optimized for the Cray-X/MP by Scientific Computing Associates [33], with portions written in Cray Assembly Language. The fixed value of $k = 10$ is chosen because larger values do not improve the robustness of GMRES(k) until k becomes a significant fraction of the problem size, and because iterative algorithms with $\mathcal{O}(N)$ additional storage are of primary interest in this paper. Whenever GMRES is refered to without an argument, GMRES(10) is understood. The reasons for selecting GMRES instead of another Krylov subspace method are the popularity of GMRES, and that Orthomin and GCR give similar results.

GMRES is also implemented with ILU and MILU preconditioners [17]. Because only GMRES is implemented with these preconditioners, the combination of GMRES with ILU preconditioning is abbreviated ILU, and similarly for MILU. PCGPAK allows the user to pass a shift parameter η to the preconditioning routine that helps guard against failure of the preconditioner by factoring $A + \eta I$ instead of A. This parameter is set to 0 in the experiments because in practice several tries at the incomplete factorization may be necessary to find a workable value for the parameter.

CGNE is also implemented with a preconditioner, found by performing an ILU* factorization on A to get $A \approx LU$. Then CG is applied to the normal equations of the left-preconditioned system $(LU)^{-1}Ax = (LU)^{-1}b$. When combined with this preconditioner CGNE is denoted as ILCG. This is not the same as forming the normal equations $A^T A$ and then performing an incomplete Cholesky factorization of $A^T A$; earlier work [7] with such a preconditioner for CGNE applied to two dimensional problems showed that it also suffered robustness problems.

6.2. Stopping Tests. The primary stopping test used is

$$\| Ax_k - b \|^2 \leq 10^{-9}. \tag{27}$$

For KACZ, the algorithm checks the *pseudo-residual* $\tilde{b} - (I - Q)x_k$. When it is less than ϵ_ρ, the *true residual* $b - Ax_k$ is checked. If the true residual is small enough, the algorithm stops. Otherwise the CG tolerance is adjusted by the assignment $\epsilon_\rho \leftarrow (0.7\epsilon_\rho / \| Ax_k - b \|^2)\epsilon$. Initially the tolerance for the pseudo-residual is equal to that on the true residual, i.e., $\epsilon_\rho = 10^{-9}$. Thus if the pseudo-residual's norm is 10 times smaller than the true residual's norm, the tolerance is decreased by 0.07, where the additional factor 0.7 is present to prevent the adjustment being made too often. Once this is done, the CG iteration resumes. The same procedure is used for CIMM and V-RP. However, since V-RP works on an auxillary vector z_k, the corresponding x_k must be retrieved each time the true residual is computed. The preconditioned PCGPAK routines use

Method	$N = 13824$						$N = 216000$					
	1	2	3	4	5	6	1	2	3	4	5	6
KACZ												
V-RP			MX						MX			
CIMM									MX			
GMRES			RS	RS					RS	TM		
ILU		RS	RS	RS		RS		UP	UP	UP		
MILU	RS	RS	RS		RS	RS	RS	UP	RS		RS	UP
CGNE			MX	MX					MX	MX		
ILCG		UP	MX	MX		MX		UP	TM	UP		TM

TABLE 4
Failures Among Methods

the residual of the preconditioned system as the stopping test, so the same tolerance adjustment procedure is used with them. When resumed this way, the preconditioner is not recomputed.

Additional stopping tests, corresponding to failure conditions, are also imposed. A maximum number of iterations is set to 4001, except for CGNE which is allowed 8001 iterations. These limits are generous and when a method fails because the maximum allowed iterations are reached, iterating further has little effect.

For the test problems there is no guarantee that the preconditioners will exist or provide a better system. The preconditioner is called *unstable* if a zero pivot occurs during the factorization, or if the iterations experience overflow because of the use of a preconditioner.

The PCGPAK routines have one more stopping criterion. If the residual does not change by a difference of at least 10^{-6} over the course of 10 successive iterations, the residual is said to *stagnate*. Omitting or relaxing this test simply changes the reason for failure from stagnated residual to maximum iterations reached.

Finally, for the problems with $N = 216000$ a maximum CPU time limit of 20 minutes is set for all of the methods. This is done for budgetary reasons. These error conditions are summarized and given the following codes in the tables of results:

- MX: Maximum iterations reached
- UP: Unstable preconditioner
- RS: Residual stagnates
- TM: Maximum allowed CPU time reached

6.3. Comparison of RP Methods with Other Methods.

6.3.1. Robustness. Robustness results are shown in Table 4. KACZ is the only algorithm tested that succeeds in all cases, while CIMM and V-RP follow with only 1 and 2 failures, respectively. All the other methods fail at least in four cases. Furthermore, KACZ is the only method that solves problem 3 for $N = 216000$.

Problems with large off-diagonal elements were the source of most of the failures of the preconditioned methods. A possible explanation is that ILU and ILCG ignore fill-in positions when forming the incomplete factors and for matrices with large entries far from the central dense band of A, those positions can be large. MILU accounts for the fill-in positions by moving their contributions to the main diagonal, but it failed in 10 cases. For the test problems MILU still provides incomplete factors that are not good approximations to the factors of A, which is revealed by noting that the initial residual of the preconditioned system often is many orders of magnitude larger than that of the unpreconditioned system. Unpreconditioned GMRES fails on problems 3 and 4, for which A has an indefinite symmetric part, but it succeeds on problems 2, 5, and 6 which also have indefinite symmetric parts. This confirms that positive definiteness of $A + A^T$ is sufficient but not necessary for convergence. Unpreconditioned CGNE fails on problems 3 and 4, which have a poor distribution of eigenvalues for the conjugate gradient algorithm, viz., they have many small eigenvalues.

Can the robustness of the GMRES(k)-based methods be improved by increasing the number k of vectors stored? To answer this these algorithms were run again on the test problems of size $N = 13824$ for which they failed. Values of k up to 40 give no improvement in the robustness. Furthermore, since for some problems the GMRES(k)-based methods report a stalled residual after only a few iterations, those problems were also tested again by letting them run for as long a time as consumed by the successful RP methods. This resulted in no significant change in the error or residual norms, even when several hundred additional iterations were performed.

Why are RP methods more robust than GMRES algorithms? Each iteration of GMRES(k) minimizes the components of the residual contributed by the eigenvectors of A restricted to a Krylov subspace. If the residual is orthogonal to that subspace, the method stalls. Even when the residual is not orthogonal to the Krylov subspace, the minimization is over a subspace of dimension k, which becomes negligible for large N. However, every projection of KACZ and V-RP minimizes the residual on a subspace of dimension $N/9$, a fixed fraction of the problem size. Although each iteration requires more work than one iteration of GMRES(k), a more robust method results.

It should be noted that the test problems were chosen partly to be difficult for Krylov subspace methods, and may not represent a fair mix of problems normally encountered. It should also be noted that some of the problems with robustness could possibly be obviated by using modified upwind differencing. However, the results show that such a change is unnecessary if RP methods are used. Furthermore, if a nonlinear PDE is being solved and a problem similar to the test problems is generated in the nonlinear iteration, it may not be easy or practical to change the differencing scheme adaptively.

6.3.2. Accuracy. The final residual and error norms are shown in Figures 2 and 3, respectively, where for each problem the three points plotted are the smallest norm from the RP methods, the smallest from the GMRES-based methods, and the smallest from CGNE or ILCG. No point is plotted for the GMRES methods for problem 3 or for CGNE or ILCG for problem 4 because the final norms produced are $\mathcal{O}(1)$ or larger. CGNE produces the smallest residual norm on 8 of the 12 cases, while KACZ does so

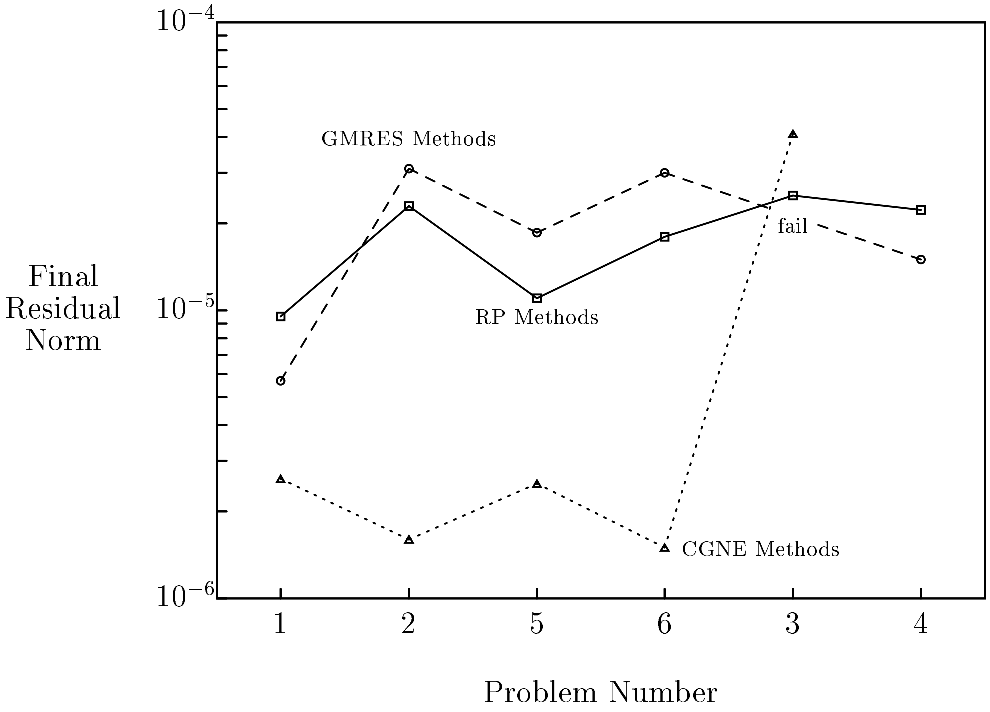

FIG. 2. *Best Residual Norms for* $N = 13824$

in 2 cases and CIMM and MILU in one each. V-RP and CGNE produce the smallest error norm in four cases, GMRES in one case, and ILU in three cases. Although V-RP fails on problem 3 for both problem sizes by using the full number of iterations allowed, it produces a smaller error norm than all of the other methods, even though its residual norm is larger than that of the successful solvers. An explanation of this behaviour lies in Theorem 3.3, which states that V-RP minimizes the error, while KACZ minimizes the error in an elliptic norm.

6.3.3. Speed and Efficiency. The robustness property is expected, since RP methods were designed to achieve it. The pleasing result is that in spite of the greater amount of work done per iteration, RP methods are the fastest of the solvers tested on 8 of the 12 experiments with the $N = 13824$ problems run on the Cray X-MP and the $N = 216000$ problems run on the Cray-2. In the other four experiments an RP method is still competitive with the fastest solver. Figure 4 shows the execution times for problems of size $N = 13824$, where for each problem the three points plotted are the fastest of the RP methods, the fastest of the GMRES-based methods, and the fastest of CGNE or ILCG. Since no Krylov method solved problem 3 and CGNE and ILCG failed on both problems 3 and 4, those data points are omitted.

The speed of RP methods is because of their parallelism. Roughly 85% of the computation time of KACZ is consumed in computing the matrix-vector products needed for the CG acceleration, and this has a natural parallelism or vector length of $n^2/9$, where $N = n^3$. GMRES computes the matrix-vector product with a vector length of only 7.

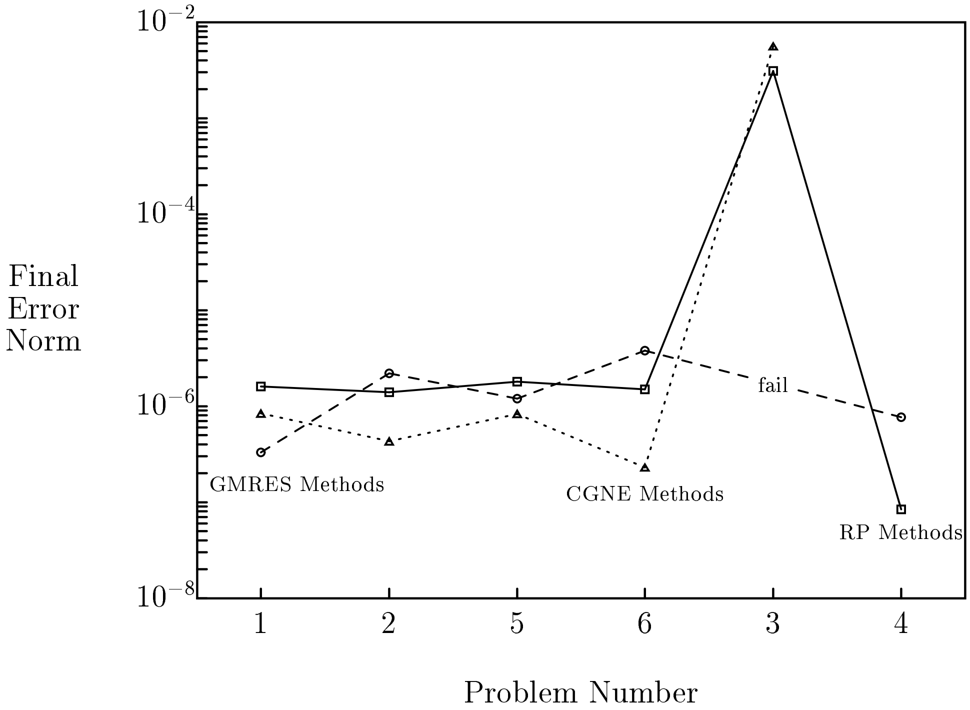

FIG. 3. *Best Error Norms for* $N = 13824$

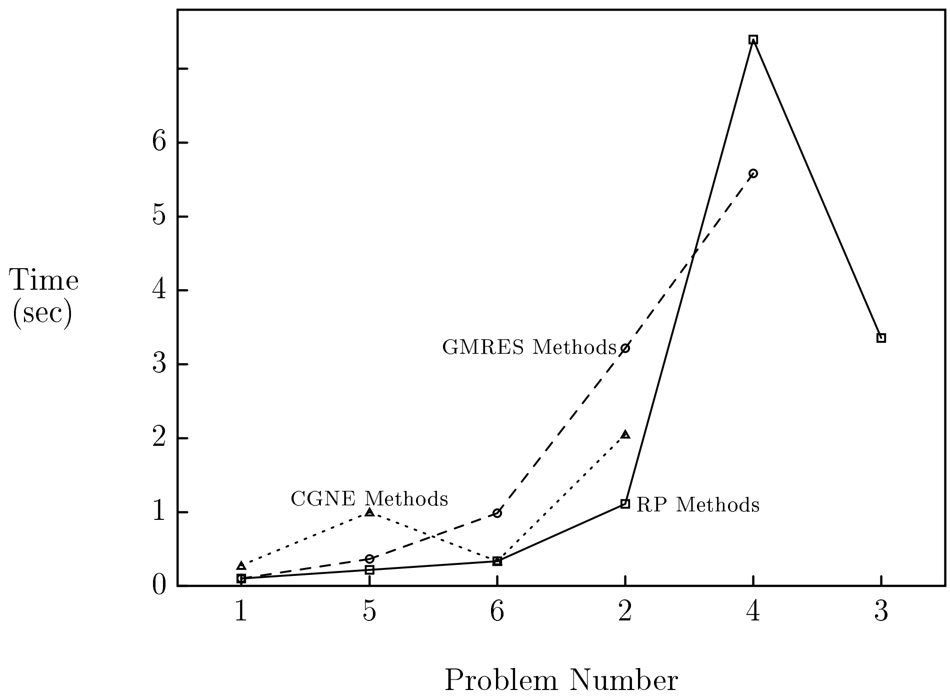

FIG. 4. *Fastest Times for* $N = 13824$

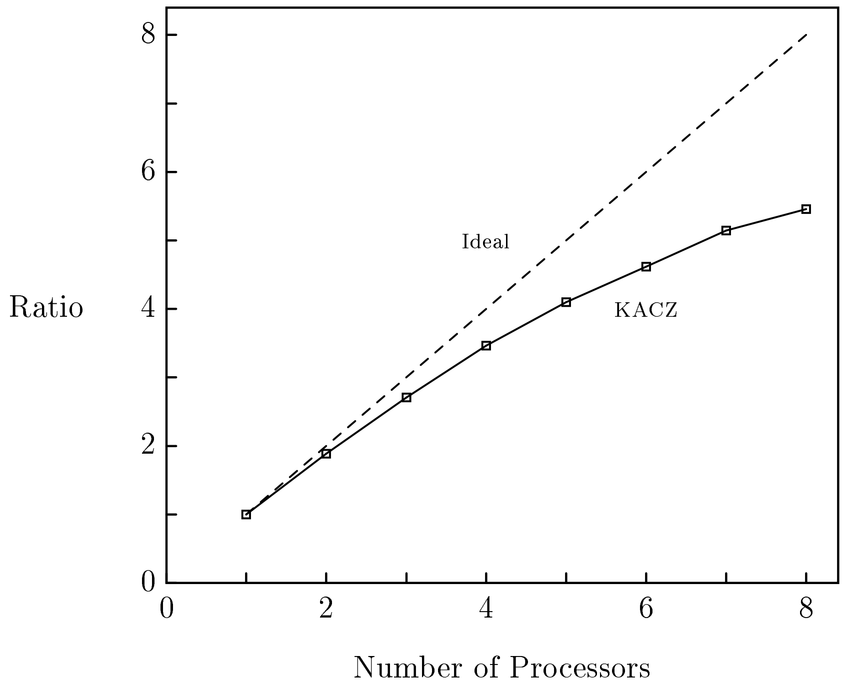

FIG. 5. *Ratio of Times for p Processors on Alliant FX/8*

Even though more parallelism is available for GMRES since each row can be handled separately, the parallel tasks are small. This is partly because of the data structure that PCGPAK uses to store A, and GMRES can run faster if the matrix-vector product is performed by diagonals rather than rows. However, GMRES benefits by having crucial parts written in Cray Assembly Language, while the RP methods are written exclusively in Fortran. For CGNE, relatively poor performance results primarily from the large number of iterations it requires.

6.3.4. Multiprocessor Results. The preceding results were obtained on single processors of Crays using vectorization across the $n^2/9$ block rows C_j^T that comprise a block row A_i^T. The algorithms can also be implemented on a shared memory multiprocessor by treating the $n^2/9$ subproblems defined by a projector as separate tasks to be assigned to different processors, with vectorizaton available in each subtask from the matrix-vector multiplications. The RP methods were implemented this way on an eight processor Alliant FX/8. Figure 5 shows the ratio of execution time on 1 processor of the Alliant FX/8 to that on p processors for KACZ, averaged over all 6 problems. For comparison, a dashed line with slope 1 is included showing the ratio corresponding to ideal parallelism. On 8 processors KACZ runs between 5 and 6 times faster, and the efficiency is 63% - 75%.

7. Discussion and Summary. Section 2.4 presented an explanation for the properties of RP methods in terms of the ill-conditioning of a matrix arising from within

Method	1	2	3	4	5	6
KACZ	8	109	666	715	20	33
V-RP	9	113	4001	813	20	33
CIMM	17	343	572	2000	51	92
CGNE	90	682	8001	8001	325	114

TABLE 5
Number of Iterations Required for $N = 13824$

Problem	KACZ	V-RP	CIMM	GMRES	CGNE
1	8	9	17	269	90
$1'$	35	36	73	269	90

TABLE 6
Number of Iterations for Problems 1 and 1'

and across blocks rows A_i^T. One implication of that argument is proven for the two block row case by Theorem 4.2, which says that KACZ should require fewer CG iterations than CIMM, which in turn should require fewer than CGNE. Table 5 shows that this relationship also holds for the $m = 9$ case tested here; the only exception is that CIMM required fewer iterations than KACZ on problem 3 with $N = 13824$. Generally CIMM requires 2-3 times as many iterations as KACZ, while CGNE requires so many iterations on some problems that it uses the maximum number allowed.

Another implication of the heuristic explanation of RP properties is that fewer iterations are needed for a given RP method when rows that make small angles with each other are placed in the same block row instead of in different block rows. This is confirmed by considering:
Problem $1'$: $\triangle u + 1000u_y = F$,
which is the same as problem 1 but with the first derivative term u_y instead of u_x. The resulting matrix A has the same spectrum as that of problem 1, but the number of iterations required increases for the RP methods while remaining constant for GMRES and CGNE, as shown in Table 6 for $N = 13824$. For both problem 1 and $1'$ it can be shown that each row a_i of A makes a small angle with at most two other rows. For problem 1 these other two rows are located in the same block row as a_i, while for $1'$ they are located in different block rows. For problem 1 the RP methods remove this near linear dependence by implicitly replacing A_i with Q_i, an orthornormal basis matrix for range(A_i). For problem $1'$, the near linear dependence is across block rows and must be handled by the outer CG iteration.

The usefulness of this explanation of RP properties is that it indicates how one should partition the matrix and number the nodes for a given problem. For example, the grid for CFD problems should be arranged so that the lines of nodes that form the block rows A_i^T are aligned in the direction of predominant flow; this occurs in problem 1 but not in problem $1'$. Furthermore, RP methods like other iterative solvers work well if the system is diagonally dominant. If there are large off-diagonal elements in the diagonal closest to the main diagonal, RP algorithms can still work well. Although every worker in the field of RP methods has recognized the importance of having the angles between the block rows large, this seems to be the first guideline on how to achieve it for practical problems without having to actually compute the angles involved.

Although the testing results of this paper are for the structured sparse systems arising from seven point centered differences, Arioli *et. al.* [2] have recently applied block Cimmino to more general sparse systems and introduced an innovative preconditioner with good results. In [2] they report on this approach, and include tests on some of the problems from this paper.

In summary CG accelerated row projection methods can have a robustness unmatched by other nonsymmetric solvers, and successfully solve large systems with indefinite real parts and eigenvalues arbitrarily distributed in the complex plane. The row partitioning scheme described here allows large scale parallelism suitable for both vector and multiprocessor machines, and yields algorithms competitive in speed with other solvers. The new method V-RP is error-minimizing and gives better solutions than the other methods, as well as explicitly reducing the problem size. Finally, the relationship with conjugate gradients applied to the normal equations is shown and an explanation for the behaviour of RP algorithms is provided. This explanation is verified both theoretically and numerically.

REFERENCES

[1] R. Ansorge, Connections between the Cimmino-methods and the Kaczmarz-methods for the solution of singular and regular systems of equations, *Computing*, 367-375, 33(1984).

[2] M. Arioli, I. Duff, J. Noailles, and D. Ruiz, A block projection method for general sparse matrices, submitted to *SIAM J. Sci. Stat. Comp.*, February 1990.

[3] S. Ashby, Chebycode: A Fortran implementation of Manteuffel's adaptive Chebyshev algorithm, Report No. UIUCDCS-R-85-1203, Department of Computer Science, University of Illinois - Urbana, May 1985.

[4] A. Björck, T. Elfving, Accelerated projection methods for computing pseudo-inverse solutions of systems of linear equations, *BIT*, 145-163, 19(1979).

[5] A. Björck, G. Golub, Numerical methods for computing angles between linear subspaces, *Math. Comp.* 579-594, 27 (1973).

[6] R. Bramley, Row projection methods for linear systems, *CSRD Tech. Rept. 881*, Center for Supercomputing Research and Development, Univ. Illinois - Urbana, (1989).

[7] R. Bramley, A. Sameh, A robust parallel solver for block tridiagonal systems, *CSRD Tech. Rept. 806*, Center for Supercomputing Research and Development, Univ. Illinois - Urbana, (1988).

[8] R. Bramley, A. Sameh, Row projection methods for large nonsymmetric linear systems, *CSRD Tech. Rept. 957 (revised)*, Center for Supercomputing Research and Development, Univ. Illinois - Urbana, (1990).

[9] G. Cimmino, Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari, *Ric. Sci. Progr. tecn. econom. naz.*, 326-333, 9(1939).

[10] P. Eggermont, G. Herman, A. Lent, Iterative algorithms for large partitioned linear systems, with applications to image reconstruction, *Lin. Alg. Appl.*, 37-67, 40(1881).

[11] T. Elfving, Group-iterative methods for consistent and inconsistent linear equations, *Rept. LITH-MAT-R-1977-11*, Dept. Math. Linköping Univ., (1977).

[12] T. Elfving, Block iterative methods for consistent and inconsistent linear equations, *Numer. Math.*, 1-12, 35(1980).

[13] H. Elman, G. Golub, Iterative methods for cyclically reduced non-self-adjoint linear systems, *UMIACS-TR-88-87, CS-TR-2145*, Dept. of Comp. Sci., University of Maryland, (1988).

[14] H. Elman, Iterative methods for large, sparse, nonsymmetric systems of linear equations, Dept. of Comp. Sci., *Yale Univ. Res. Rept. 229*, (1982).

[15] P. Gilbert, Iterative methods for the three-dimensional reconstruction of an object from projections, *J. Theor. Biol.*, 105-117, 36(1972).

[16] G. Golub, C. Van Loan, *Matrix Computations*, John Hopkins University Press, Baltimore (1983).

[17] I. Gustafsson, Modified incomplete Cholesky methods, *Preconditioning Methods: Theory and Applications*, ed. D. Evans, 265-293, Gordon and Breach, New York, (1983).

[18] L. Hageman, D. Young, *Applied Iterative Methods*, Academic Press, New York, (1981).

[19] G. Herman, A. Lent, P. Lutz, Relaxation methods for image reconstruction, *Comm. ACM*, 152-158, 21(2), (1978).

[20] S. Kaczmarz, Angenäherte Auflösung von Systemen linearer Gleichungen, *Bull. intern. Acad. polonaise Sci. lettres (Cracouie); Class sci. math. natur.: Seira A. Sci. Math.*, 355-357 (1939).

[21] C. Kamath, Solution of nonsymmetric systems of equations on a multiprocessor, *CSRD Tech. Rept. 591*, Center for Supercomputing Research and Development, Univ. Illinois - Urbana, (1986).

[22] C. Kamath, A. Sameh, A projection method for solving nonsymmetric linear systems on multiprocessors, *Parallel Computing*, 291-312, 9(1988/1989).

[23] D. Kincaid, D. Young, Adapting iterative algorithms developed for symmetric systems to nonsymmetric systems, *Elliptic Problem Solvers*, 353-359, ed. M. Schultz, Academic

Press, New York, (1981).

[24] D. Kuck, E. Davidson, D. Lawrie, A. Sameh, Parallel supercomputing today and the Cedar approach, *Science*, 967-974, 231(1986).

[25] A. Kydes, R. Tewarson, An iterative method for solving partitioned linear equations, *Computing* 357-363, 15(1975).

[26] A. Lakshminarayanan, A. Lent, Methods of least squares and SIRT in reconstruction, *J. Theor. Biol.*, 267-295, 76(1979).

[27] T. Manteuffel, The Tchebychev iteration for nonsymmetric linear systems, Numer. Math., 307-327, 28(1977).

[28] T. Manteuffel, Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration, *Numer. Math.*, 183-208, 31(1978).

[29] S. Nelson, M. Neumann, Generalizations of the projection method with applications to SOR theory for Hermitian positive semidefinite linear systems, *Numer. Math.* 123-141, 51 (1987).

[30] W. Peters, Lösung linearer Gleichungssysteme durch Projektion auf Schnitträume von Hyperebenen und Berechnung einer verallgemeinerten Inversen, *Beit. Numer. Math.*, 129-146, 5(1976).

[31] Y. Saad, M. Schultz, Conjugate gradient-like algorithms for solving nonsymmetric linear systems, *Math. Comp*, 417-424, 44(1985).

[32] P. Saylor, Leapfrog variants of iterative methods for linear algebraic equations, *J. Comp. Appl. Math.*, 169-193, 24(1988).

[33] Scientific Computing Associates, Inc., *PCGPAK User's Guide*, New Haven, CT (1987).

[34] A. Sherman, An empirical investigation of methods for nonsymmetric linear systems, *Elliptic Problem Solvers*, 429-434, ed. M. Schultz, Academic Press, New York, (1981).

[35] G. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Clarendon Press, Oxford (1978).

[36] G. Stewart, On the perturbation of pseudo-inverses, projections, and linear least squares problems, *SIAM Rev.*, 634-662, 19(1977).

[37] K. Tanabe, A projection method for solving a singular system of linear equations, *Numer. Math.*, 203-214 17(1971).

[38] T. Whitney, R. Meany, Two algorithms related to the method of steepest descent, *SIAM J. Numer. Anal.*,109-118, 4(1967).

[39] J. Wilkinson, Modern error analysis, *SIAM Rev.*, 548-568, 13(1971).