SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 480–498

SOLVING COMPLEX-VALUED LINEAR SYSTEMS VIA EQUIVALENT REAL FORMULATIONS*

DAVID DAY† AND MICHAEL A. HEROUX†

Abstract. Most preconditioned iterative methods apply to both real- and complex-valued linear systems. At the same time, most iterative linear solver packages available today focus exclusively on real-valued systems or deal with complex-valued systems as an afterthought. By recasting the complex problem in a real formulation, a real-valued solver can be applied to the equivalent real system.

On one hand, real formulations have been dismissed due to their unfavorable spectral properties. On the other hand, using an equivalent preconditioned real formulation can be very effective. We give theoretical and experimental evidence that an equivalent real formulation is useful in a number of practical situations. Furthermore, we show how to use the advanced features of modern solver packages to formulate equivalent real preconditioners that are computationally efficient and mathematically identical to their complex counterparts.

The effectiveness of equivalent real formulations is demonstrated by solving ill-conditioned complex-valued linear systems for a variety of large scale applications. Moreover, the circumstances under which certain equivalent real formulations are competitive is more clearly delineated.

Key words. complex linear systems, iterative methods, sparse matrices, preconditioning

AMS subject classifications. 65F10, 65F50, 65N12, 65Y05, 65Y15

PII. S1064827500372262

1. Introduction.

We describe a simple yet effective extension of a real-valued preconditioned iterative solver package to solve complex-valued linear systems such as

$$Cw = d, \tag{1.1}$$

where C is an m-by-n known complex matrix, d is a known right-hand side, and w is unknown. Although most preconditioners and iterative methods apply to either complex-valued or real linear systems [4], most preconditioned iterative solver packages treat only real-valued systems. Packages that deal with complex-valued linear systems include QMRPACK [8], PETSc [3, 2, 1], and work done at CERFACS [6]. However, even in these cases, the breadth, depth, and performance capabilities of the complex-valued solvers are not as complete as the collective set of real-valued solvers. Because of these apparent deficiencies, we are compelled to consider using the vast body of real-valued solver packages to solve complex-valued systems. This work explains when and how to leverage the existing real-valued solver packages for use with complex-valued systems.

1.1. Potential equivalent real formulations.

We begin our study of equivalent real formulations by writing (1.1) in its real and imaginary terms:

$$(A + iB)(x + iy) = b + ic. \tag{1.2}$$

*Received by the editors May 18, 2000; accepted for publication (in revised form) September 20, 2000; published electronically July 10, 2001.
http://www.siam.org/journals/sisc/23-2/37226.html
†Applied Mathematics Department, Massively Parallel Computing Research Lab, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185-1110 (dday@cs.sandia.gov, mheroux @cs.sandia.gov).

TABLE 1.1
Spectral properties of the K formulations. $\sigma(K)$ denotes the spectrum of K and $i = \sqrt{-1}$.

Matrix	Spectral properties
K_1	(i) If $\lambda \in \sigma(C)$, then $\lambda, \bar{\lambda} \in \sigma(K_1)$. (ii) If C is Hermitian (positive definite), then K_1 is symmetric (positive definite).
K_2	(i) If $\lambda \in \sigma(K_2)$, then $-\lambda, \bar{\lambda}, -\bar{\lambda} \in \sigma(K_2)$. (ii) If C is symmetric, then K_2 is symmetric.
K_3	(i) If $\lambda \in \sigma(K_3)$, then $-\lambda, \bar{\lambda}, -\bar{\lambda} \in \sigma(K_3)$. (ii) If C is symmetric, then K_3 is symmetric. (iii) $\sigma(K_3) = \sigma(K_2)$.
K_4	(i) If $\lambda \in \sigma(C)$, then $-i\lambda, i\overline{\lambda} \in \sigma(K_4)$. (ii) If C is Hermitian (positive definite), then K_4 is skew symmetric (with eigenvalues that have positive imaginary parts).

Equating the real and imaginary parts of the expanded equation, respectively, gives rise to four possible 2-by-2 block formulations, listed in (1.3)–(1.6). We call these K1 to K4, respectively.

K1 *formulation.*

$$\begin{pmatrix} A & -B \\ B & A \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}. \tag{1.3}$$

K2 *formulation.*

$$\begin{pmatrix} A & B \\ B & -A \end{pmatrix} \begin{pmatrix} x \\ -y \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}. \tag{1.4}$$

K3 *formulation.*

$$\begin{pmatrix} B & A \\ A & -B \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c \\ b \end{pmatrix}. \tag{1.5}$$

K4 *formulation.*

$$\begin{pmatrix} B & -A \\ A & B \end{pmatrix} \begin{pmatrix} x \\ -y \end{pmatrix} = \begin{pmatrix} c \\ b \end{pmatrix}. \tag{1.6}$$

For future reference, we denote the matrix associated with the K1 to K4 formulations by K_1 to K_4, respectively. For any pair of bases for R^{2m} and R^{2n}, there is a different equivalent real formulation of (1.1). K1 to K4 are representative of the many different formulations.

The convergence rate of an iterative method applied directly to an equivalent real formulation is often substantially worse than for the corresponding complex iterative method (see [7] and section 2). The slower convergence rate is due to the spectral properties of the equivalent real formulation. Table 1.1 summarizes the spectral properties of K_1 to K_4.

We digress briefly to justify certain relations in Table 1.1 that are not discussed in the following sections. The K_2 and K_3 matrices satisfy $K_2 J = -JK_2$ and $K_3 J = -JK_3$, where

$$J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}. \tag{1.7}$$

In fact, $-JK_2$ has the same eigenvalues as K_2 and is similar to K_3. Thus $\sigma(K_3) = \sigma(K_2)$ is symmetric with respect to the origin. ($\sigma(K)$ denotes the spectrum of K.)

The configuration of the eigenvalues of K_2 and K_3 is problematic for Krylov methods. If a matrix has positive eigenvalues, then augmenting that matrix in a way that reflects the spectrum through the origin degrades the convergence rate of an iterative method such as the generalized minimum residual (GMRES) method. The convergence rate for a matrix whose spectrum is symmetric with respect to the origin is the square root of the rate for the same method applied to a linear system whose spectrum lies in a half plane (see section 3.2, Example 2). Similarly, convergence rates in the K2 or K3 formulations tend to be the square root of the rates in the K1 or K4 formulations.

For the K1 formulation, if all the eigenvalues of C are on one side of the imaginary axis, then the spectrum of K_1 should not present a major problem to an iterative method such as GMRES. However, if C has eigenvalues on both sides of the imaginary axis, a property that degrades the GMRES convergence rate, then K_1 will have twice as many problematic eigenvalues. Moreover, the convex hull containing the eigenvalues of K_1 will also contain the origin. For the K4 formulation, if all the eigenvalues of C are in the upper half plane, then the eigenvalues of K_4 will be in the right half plane.

It is noteworthy that the K4 formulation of (1.1) is the same as the K1 formulation of

$$i\,\overline{C}\overline{w} = i\,\overline{d}. \tag{1.8}$$

Similarly, the K3 formulation of (1.1) is the same as the K2 formulation of (1.8). Because of these relationships, we need to consider only the K1 and K2 formulations if we are interested in the convergence properties of *unpreconditioned* Krylov methods. In fact, most of the remaining discussion is focused on the K1 and K2 formulations with K3 and K4 being special cases. However, if we want to consider the preconditioned case, especially preconditioners such as the incomplete lower/upper factorization (ILU), then K4 deserves more attention than we present in section 2.2.

If C is Hermitian, then K_1 is symmetric, and, as we will demonstrate below, the convergence rate with an equivalent real formulation is identical to the convergence rate with the original complex formulation. If C is complex symmetric, then K_2 and K_3 are also complex symmetric. Nevertheless, we still recommend K_1 for complex symmetric linear systems (see section 3.2, particularly Examples 1 and 2) because of the problem mentioned above. The best way known to the authors to precondition K_2 or K_3 is to permute back to K_1.

In view of their spectral properties, the K1 to K4 formulations have been justly criticized. Particularly, the K2 and K3 formulations appear to be unusable. However, in spite of these properties, we have found that a variation of the K1 formulation has merit. Success with the K1 formulation depends on the quality of the preconditioner to the *same* extent that success with the original complex formulation depends on the quality of the complex preconditioner. In fact, our experience shows that for the

classes of problems we are solving, in particular, eigenvalue problems for computational fluid dynamics using complex valued shift parameters, if a good preconditioner is used, then the iteration count of the K formulation (discussed below) is generally comparable (within 50%) to that of solving the original complex problem with a true complex preconditioned iterative solver and has the same robustness as a true complex solver. Given the wide availability of excellent real-valued solver packages, we view our results as noteworthy.

2. Preconditioning equivalent real systems. In trying to solve the original complex system in (1.1) via the K1 formulation in (1.3), the most interesting question is how to precondition K_1. Standard real-valued preconditioners such as Jacobi, Gauss–Seidel, or ILU applied directly to K_1 are not robust enough for our needs. Furthermore, the ordering of the unknowns is unsuitable for sparse matrix operations related to factorization, particularly ILU preconditioning.

Another preconditioner we experimented with is

$$M = \begin{pmatrix} M_A & 0 \\ 0 & M_A \end{pmatrix}, \tag{2.1}$$

where M_A is a real-valued preconditioner determined from the real part of $C = A+iB$. This approach can be viewed as a special form of block Jacobi preconditioning that may be effective if terms in A are generally larger in magnitude than those in B. However, for the problems we tested, the preconditioner in (2.1) was also not robust enough. We observed large increases in iteration count and outright failure to converge, while the solution of the complex system via a complex solver succeeded. Preconditioning by the imaginary part of C (by using the K4 formulation in (1.6)) also failed.

2.1. The K formulation. The approach that consistently gives us good results is based on preserving the sparsity pattern of C. A *block entry* matrix is a sparse matrix whose entries are all (small) dense (sub)matrices. An alternative formulation, which we call the K formulation, preserves the nonzero pattern of the block entries at the expense of doubling the size of each dense submatrix.

In the K formulation, $c_{pq} = a_{pq} + ib_{pq}$ corresponds, via the scalar K1 formulation, to the 2-by-2 block entry of the $2m$-by-$2n$ real matrix K given by

$$\begin{pmatrix} a_{pq} & -b_{pq} \\ b_{pq} & a_{pq} \end{pmatrix}. \tag{2.2}$$

For example, if

$$C = \begin{pmatrix} c_{11} & 0 & c_{13} & 0 & c_{15} \\ 0 & c_{22} & c_{23} & 0 & 0 \\ c_{31} & 0 & c_{33} & c_{34} & 0 \\ 0 & 0 & c_{43} & c_{44} & 0 \\ c_{51} & 0 & 0 & 0 & c_{55} \end{pmatrix}, \tag{2.3}$$

then

$$K = \left(\begin{array}{cc|cc|cc|cc|cc} a_{11} & -b_{11} & 0 & 0 & a_{13} & -b_{13} & 0 & 0 & a_{15} & -b_{15} \\ b_{11} & a_{11} & 0 & 0 & b_{13} & a_{13} & 0 & 0 & b_{15} & a_{15} \\ \hline 0 & 0 & a_{22} & -b_{22} & a_{23} & -b_{23} & 0 & 0 & 0 & 0 \\ 0 & 0 & b_{22} & a_{22} & b_{23} & a_{23} & 0 & 0 & 0 & 0 \\ \hline a_{31} & -b_{31} & 0 & 0 & a_{33} & -b_{33} & a_{34} & -b_{34} & 0 & 0 \\ b_{31} & a_{31} & 0 & 0 & b_{33} & a_{33} & a_{34} & -b_{34} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & a_{43} & -b_{43} & a_{44} & -b_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & b_{43} & a_{43} & a_{44} & -b_{44} & 0 & 0 \\ \hline a_{51} & -b_{51} & 0 & 0 & 0 & 0 & 0 & 0 & a_{55} & -b_{55} \\ b_{51} & a_{51} & 0 & 0 & 0 & 0 & 0 & 0 & b_{55} & a_{55} \end{array}\right). \tag{2.4}$$

In a related real formulation, also preserving the block entry structure, the K2 formulation of c_{pq} forms the corresponding 2-by-2 block entry

$$\begin{pmatrix} b_{pq} & a_{pq} \\ a_{pq} & -b_{pq} \end{pmatrix}. \tag{2.5}$$

The K3 and K4 versions are generated similarly.

2.2. Implementation within existing software packages. The properties of the K formulation defined in section 2.1 enable us to implement efficient and robust preconditioned iterative solvers for complex linear systems. We can efficiently compute and apply the exact equivalent of a complex-valued preconditioner. If the complex preconditioned linear system has nice spectral properties, then the K formulation leads to convergence that is competitive with the true complex solver.

Solvers with block entry support. There are a number of general-purpose parallel sparse iterative solver packages that do not require special matrix properties, e.g., structured grids, and thus can be applied to a wide range of problems. These packages are usually freely available and have been used in a variety of applications. Several of these packages, including Aztec [19] and PETSc [2], support block entry matrices. The matrix K in the K formulation has a natural 2-by-2 block structure that can be exploited by using block entry data structures. Using the block entry features of these packages has the following benefits.

1. Applying 2-by-2 block Jacobi scaling to K corresponds exactly to applying point Jacobi scaling to C.
2. The block sparsity pattern of K exactly matches the point sparsity pattern of C. Thus any pattern-based preconditioners such as block ILU(l) applied to K correspond exactly to ILU(l) applied to C. See section 4 for definitions of block ILU(l) and ILU(l).
3. Any drop tolerance-based complex preconditioner has a straightforward K formulation since the absolute value of a complex entry equals the scaled Frobenius norm of the corresponding block entry in K.

Other solver packages. For existing solvers that do not have block entry support, it is still possible to exploit many of the benefits of the K formulation with the following considerations.

1. It is still possible to form the K matrix, even though its underlying block structure will not be explicitly available. Many preconditioners, especially incomplete factorizations, will closely approximate the block preconditioners listed above. Ideally, a real or imaginary nonzero in C corresponds to four entries in K, two of which are zero and may fill in during an incomplete factorization.
2. If any diagonal entries of the complex matrix C have zero or very small real parts, then pointwise incomplete factorization preconditioners may fail. (Note that this is not a problem for block entry factorization if pivoting is done within the diagonal block entries.) In this situation, one might consider using the K4 equivalent of the K formulation, or one may interleave the K1 and K4 formulations at the equation level depending on the relative magnitude of the real and imaginary parts of the complex diagonal entries. In certain situations, this leads to better ILU factors. However, we have not fully explored this approach, and its overall effectiveness remains an open question to us.

As noted above, by using block entry features, we can easily and efficiently construct preconditioners for K that are equivalent to those we would form for C using a true complex preconditioner formulation. As a result, the equivalent real preconditioned matrix operation is *identical* to the true complex preconditioned operator up to a permutation. Thus solving the real equivalent form via the K formulation using a preconditioned iterative method is identical to solving the original complex system using a corresponding preconditioned complex solver, except that the two approaches use different inner product spaces. Section 3 expands upon these comments.

3. Properties of the K formulation. The K formulation of a preconditioned iterative method is comparable to the original complex formulation. Section 3.1 presents an equivalence between a preconditioned complex linear system and the corresponding K formulation. Next, the properties of K and C that influence convergence of iterative linear solvers are contrasted. The critical difference between K and C is that the eigenvalues of K are the eigenvalues of C together with their conjugates; $\sigma(K) = \sigma(C) \cup \overline{\sigma(C)}$. Section 3.2 contains a detailed survey of the influence of the spectrum of the convergence rate. The theoretical results of sections 3.1 and 3.2 are summarized in section 3.3 and illustrated using an example from computational chemistry in section 3.4.

3.1. Homomorphic properties of the K formulation. The K formulation of a coefficient matrix C results in a symmetric permutation $K = PK_1P^T$ that preserves the block entry structure of C. Note that K and K_1 are orthogonally similar and share many properties.

We use the function $f()$ to denote the matrix K that corresponds to C in the K formulation, $f(C) = K$. The *key to understanding the K formulation is that f is a homomorphism*:

$$\begin{cases} f(I) = I, \\ f(XY) = f(X)f(Y). \end{cases}$$

This observation appears not to have been made before in this context. In the K formulation of an iterative method for solving a linear system with coefficient matrix C and preconditioner M, the linear system $f(C)$ is preconditioned with preconditioner $f(M)$. The K formulation of a preconditioned iterative method inherits from the

preconditioned complex iterative method through

$$f(M^{-1}C) = f(M)^{-1}f(C)$$

all the properties that K_1 inherits from C.

We use the following framework to compare preconditioned iterative linear solvers. For clarity, a zero initial guess is assumed. A left-preconditioned Krylov solver is an algorithm that determines a sequence of approximations from the corresponding expanding Krylov subspaces

$$\mathcal{K}^k(M^{-1}C, M^{-1}d) = \text{span}(M^{-1}d, \ldots, (M^{-1}C)^{k-1}M^{-1}d). \tag{3.1}$$

Right preconditioning is similar. A preconditioned Krylov solver succeeds to the extent that the algorithm converges in a small number of iterations.

Characterizing successful preconditioners is difficult. The convergence rate of conjugate gradient methods applied to the left or right normal equations is bounded in terms of the square of the condition number

$$\text{cond}(M^{-1}C). \tag{3.2}$$

The convergence rate of GMRES is bounded in terms of the eigenvalues and condition number of the matrix of eigenvectors of $M^{-1}C$ [15]. For many problems, the maximum absolute value of the ratio of eigenvalues, the *spectral* condition number, best correlates with preconditioner effectiveness. The Krylov subspace, (3.1), the condition number of C, (3.2), and the condition number of the matrix of eigenvectors of C are all invariant under the K formulation (see below).

The K formulation preserves Krylov subspaces in the sense that

$$f((M^{-1}C)^k M^{-1}) = (f(M)^{-1}f(C))^k f(M)^{-1}$$

for left preconditioning and a similar equation applies with a right preconditioner. A prerequisite to discussing condition numbers is to relate the singular value decomposition (SVD) of C,

$$C = U\Sigma V^*,$$

to the SVD of K_1. Problem 8.6.4 in [9] is to show that if $U = U_r + iU_u$ and $V = V_r + iV_u$, where U_r,U_u,V_r,V_u are real, then K has the SVD

$$K = \begin{bmatrix} U_r & -U_u \\ U_u & U_r \end{bmatrix} \begin{bmatrix} \Sigma & \mathbf{0} \\ \mathbf{0} & \Sigma \end{bmatrix} \begin{bmatrix} V_r & -V_u \\ V_u & V_r \end{bmatrix}^T.$$

In particular, f preserves condition numbers. This implies the second property, that conditioning is preserved from the complex case:

$$\begin{aligned} \text{cond}(M^{-1}C) &= \text{cond}(f(M^{-1}C)) \\ &= \text{cond}(f(M)^{-1}f(C)). \end{aligned}$$

For clarity, we discuss eigenvalues in the unpreconditioned case. However, as above, these results to extend to the preconditioned case. As mentioned in section 3, eigenvalues of K are the eigenvalues of C together with their conjugates. Thus, for

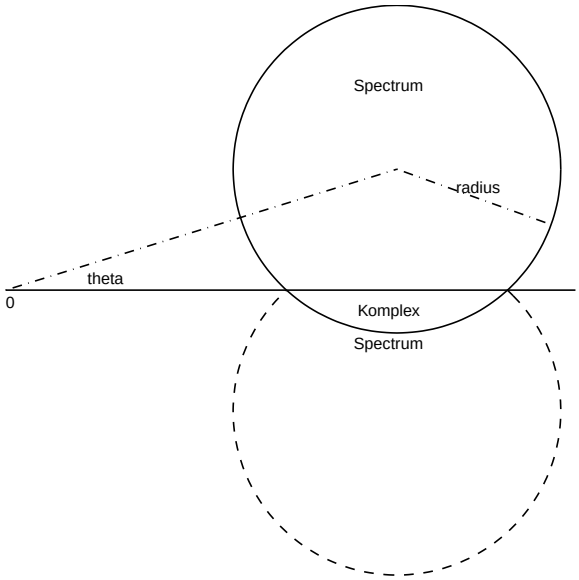

FIG. 3.1. *Asymmetric preconditioned spectrum.*

example, $M^{-1}C$ and $f(M)^{-1}f(C)$ also have the same *spectral* condition number. More precisely, Proposition 5.1 of [7] states that if C has Jordan normal form,

$$C = XJX^{-1},$$

then $K_1 = W\mathrm{diag}(J, \overline{J})W^{-1}$ for

$$W = \begin{bmatrix} X & \overline{X} \\ -iX & i\overline{X} \end{bmatrix}.$$

An observation not made explicitly in [7] is that it immediately follows that

$$\mathrm{cond}(W) = \mathrm{cond}(X).$$

As mentioned earlier, eigenvalue-based bounds on the convergence rate involve the condition number of the matrix of eigenvectors. (See also (3.3) below.)

3.2. Convergence of the K formulation. Augmenting the spectrum of C to $\sigma(K) = \sigma(C) \cup \overline{\sigma(C)}$ may or may not change the convergence rate of an iterative method. Examples 1–4 below compare the convergence rates of iterative methods in the K formulation and the original complex formulation applied to representative classes of model problems. Based on these examples, we conclude the following.

- For Hermitian linear systems, the K formulation preserves the convergence rate of the original complex formulation.
- For linear systems with an asymmetric preconditioned spectrum (see Figure 3.1), the convergence rate degrades mildly in a K formulation.

We review the prerequisite mathematical tools to keep this work self-contained.

Preliminary remarks. We selected four examples to illustrate the influence of recasting a complex linear system in an equivalent real form on the convergence rate of the linear solver. Examples 1 and 3 are paraphrased from [5]. Example 2 is classical

and reviewed here in detail. Example 4 appears to be new. Examples 1 and 2 illustrate the best and the worst properties of the K1 formulation. In Example 1 the spectral condition number is preserved, and in Example 2 the spectral condition number is squared. Fortunately, the K formulation of a successfully preconditioned system is more like Example 1 than Example 2.

Examples 3 and 4 relate the convergence rates of the K formulation of a preconditioned iterative method to the corresponding complex preconditioned iterative method if the spectrum is nearly symmetric with respect to the real axis. Successful preconditioning transforms the spectrum into a disk far from 0. Example 3 shows that the convergence rate for the complex formulation is the ratio of the radius of the disk to the distance from the center of the disk to the origin. In the K formulation the convergence rate also depends on the angle between the disk and the real axis. Example 4 shows how the convergence rate gently increases as the disk rotates away from the positive real axis.

The asymptotic convergence factor for polynomial-based methods, including preconditioned Krylov subspace methods, is κ if the nth residual norm is proportional to κ^n for some constant κ. A sharp upper bound for the asymptotic convergence factor can be determined using the complex Green's function for the convex hull of the spectrum. Following [5, pp. 90–93], an iterative method for $Cw = d$ determines $\{w_k\}$ that (hopefully) converges to w and residuals, $r_k = d - Cw_k$, such that $\{r_k\}$ converges to zero. Consider the polynomial-based iterative solution method

$$r_n = p_n(C)r_0, \quad p_n(0) = 1,$$

where each $p_n \in \Pi_n$, the space of nth degree polynomials. Next let Ω be a set containing $\sigma(C)$, and define

$$\|p_n\|_\Omega = \max_{\omega\in\Omega} |p_n(\omega)|.$$

Common choices for Ω are the convex hull of $\sigma(C)$, a disk, or an ellipse with a major axis along a ray through the origin. For clarity, we assume that C is diagonalizable: $CV = V\Lambda$. The reduction in the residual norm

$$\|r_n\|_2 = \|p_n(C)r_0\|_2 \le \mathrm{cond}(V)\|p_n\|_{\sigma(C)}\|r_0\|_2 \le \mathrm{cond}(V)\|p_n\|_\Omega\|r_0\|_2 \tag{3.3}$$

is bounded above by the spectral condition number of V and $\|p_n\|_\Omega$. A residual polynomial p_n that minimizes $\|p_n\|_\Omega$ is an *optimal polynomial* $\mathcal{P}_n(t;\Omega,0)$ and solves

$$\|\mathcal{P}_n(t;\Omega,0)\|_\Omega = \min\{\|p\|_\Omega : p \in \Pi_n, p(0) = 1\}, \quad 0 \notin \Omega.$$

The convergence of an iteration $r_n = p_n(C)r_0$ is related to the asymptotic convergence factor for the polynomial iterative method induced by $\{p_n\}$,

$$\kappa(C;p_n) = \limsup_{n\to\infty} \left(\sup_{r_0\neq 0} \frac{\|r_n\|_2}{\|r_0\|_2} \right)^{1/n}.$$

The asymptotic convergence factor for Ω is defined by

$$\kappa(\Omega) = \inf_{p_n} \sup_{\sigma(C)\subset\Omega} \kappa(C;p_n).$$

The asymptotic convergence factor corresponding to the K formulation is $\kappa(\Omega \cup \overline{\Omega})$.

Next we determine the asymptotic convergence factor for C and K in several characteristic cases. $\kappa(\Omega)$ is determined from Green's function $G(z;\Omega^c)$ for the complement Ω^c of Ω with pole at infinity

$$\kappa(\Omega) = |G(0;\Omega^c)|.$$

If Ω is connected, then G is a conformal mapping from Ω^c to the open unit disk such that $G(\infty) = 0$. In general, G has the following properties.

- $G(z;\Omega^c)$ is an analytic function on Ω^c with a single-valued modulus $|G(z;\Omega^c)| < 1$ in Ω^c.
- $G(z;\Omega^c)$ has precisely one zero at ∞.
- If $z \in \partial\Omega$, then $|G(z;\Omega^c)| = 1$.

Example 1. If C is Hermitian positive definite, then the convex hull of $\sigma(C) = \sigma(K)$ is $\Omega = [\alpha, \beta]$ for $0 < \alpha < \beta$. The first step is to derive Green's function for the complement of the interval $G(z; I^c)$, an inverse of $\phi(z) = (z + z^{-1})/2$, where I denotes the interval $[-1, 1]$. We select the square root function with a branch cut along the negative real axis and

$$\phi^{-1}(z) = \begin{cases} z + \sqrt{z^2 - 1}, & 0 \le \arg(z) \le \pi, \\ z - \sqrt{z^2 - 1}, & \pi < \arg(z) < 2\pi. \end{cases}$$

Note that the branch cut for the argument function is along the positive real axis, and for $\rho > 1$,

$$\lim_{z \to \rho, \Im(z) < 0} \arg(z^2 - 1) = 2\pi.$$

Here $\Im(z)$ denotes the imaginary part of the complex number z. In this case, the singularity in ϕ^{-1} along $[1, \infty]$ has been removed, and because ϕ^{-1} is odd, ϕ^{-1} is analytic on I^c. To show that ϕ^{-1} maps I^c to the open unit disk, note that the equation $\phi(z) = w$ is a quadratic polynomial in z and $\phi(z) = \phi(z^{-1})$.

Now for $\Omega = [\alpha, \beta]$ such that $0 < \alpha < \beta$, $G(z;\Omega^c) = \phi^{-1}(\ell(z))$ for $\ell(t) = (2t - \alpha - \beta)/(\beta - \alpha)$, and

$$\kappa([\alpha, \beta]) = |G(0)| = \frac{1}{-\ell(0) + \sqrt{\ell^2(0) - 1}} = \frac{\beta - \alpha}{\beta + 2\sqrt{\beta\alpha} + \alpha}.$$

Example 2. If iC is Hermitian positive definite, then the convex hull of $\sigma(C)$ is $[i\alpha, i\beta]$ for $0 < \alpha < \beta$ and $\kappa(i[\alpha, \beta]) = \kappa([\alpha, \beta])$. However, $\sigma(K) \subset [-i\beta, -i\alpha] \cup [i\alpha, i\beta]$, and $\kappa([-i\beta, -i\alpha] \cup [i\alpha, i\beta]) = \kappa([-\beta, -\alpha] \cup [\alpha, \beta])$. The first step is to derive Green's function for the complement of symmetric intervals $\Omega = [-1, -\eta] \cup [\eta, 1]$ for $\eta = \alpha/\beta < 1$. Green's function is a branch of the solution of

$$\psi^2 + 2\frac{2z^2 - (1 + \eta^2)}{1 - \eta^2}\psi + 1 = 0.$$

The two branches

$$\psi_\pm = \frac{2}{1 - \eta^2}\left(\pm\sqrt{(z^2 - 1)(z^2 - \eta^2)} - z^2 + \frac{1 + \eta^2}{2}\right)$$

satisfy $\psi_+\psi_- = 1$, and we seek a branch whose range is the open unit disk. We choose the branch of $\pm\sqrt{(z^2 - 1)(z^2 - \eta^2)}$ that is nearest to z^2. The branch cut for $\sqrt{\ }$ is along the negative real axis, and

$$\arg\left((z^2 - 1)(z^2 - \eta^2)\right) = \pm\pi \leftrightarrow z = x + iy \ \text{ and } \ x^2 - y^2 = \frac{1 + \eta^2}{2}.$$

This gives us Green's function

$$G(z) = \begin{cases} \frac{2}{1-\eta^2}\left(\sqrt{(z^2-1)(z^2-\eta^2)} - z^2 + \frac{1+\eta^2}{2}\right), & x^2 - y^2 \geq \frac{1+\eta^2}{2}, \\ \frac{2}{1-\eta^2}\left(-\sqrt{(z^2-1)(z^2-\eta^2)} - z^2 + \frac{1+\eta^2}{2}\right), & x^2 - y^2 < \frac{1+\eta^2}{2}. \end{cases}$$

To show that the range of G is the open unit disk, note that ψ is a quadratic polynomial and $|\psi_\pm| = 1$ if and only if $\eta^2 \leq z^2 \leq 1$. In this case, $\kappa(\Omega) = (1-\eta)/(1+\eta) = (\beta-\alpha)/(\beta+\alpha)$.

To contrast Examples 1 and 2 as in the preliminary remarks, note that β/α is the spectral condition number of C. In Example 1 the asymptotic reduction factor depends on the square root of the spectral condition number,

$$\kappa_1 \approx 1 - 2\sqrt{\frac{\alpha}{\beta}}.$$

In Example 2 the asymptotic reduction factor depends on the spectral condition number and is much larger:

$$\kappa_2 \approx 1 - 2\,\frac{\alpha}{\beta} \gg \kappa_1.$$

Examples 3 and 4 quantify the penalty for using the K formulation instead of the true complex formulation if the convex hull of the preconditioned spectrum lies inside a disk in the left half plane rotated by θ from the positive real axis. Our analysis applies in the case in which the disk intersects its conjugate. Example 3 shows that in complex arithmetic, the asymptotic convergence factor is independent of θ. Comparing (3.4) with $|w| = 1$ to (3.6) shows that, in the K formulation, the asymptotic convergence factor increases mildly with θ.

Example 3. Here $\Omega = \{z : |z - w| < r\}$ is the disk in the complex plane of radius r about a point w that is the distance $\rho = |w|$ from the origin. The asymptotic convergence factor is $\eta = r/\rho$. A conformal mapping of Ω onto the open unit disk is $G(z;\Omega) = \frac{r}{z-w}$ and

$$\kappa(\Omega) = \frac{r}{|w|} = \eta. \tag{3.4}$$

Example 4. Here Ω comes from rotating the disk $\{z : |z - \rho| < r\}$ with center $\rho > 0$ by θ and then from reflecting though the real axis as illustrated in Figure 3.1. The circles intersect: $\sin\theta < \eta = r/\rho$.

Figure 3.2 illustrates the conformal mapping from two intersecting disks to a disk. The upper left graph corresponds to Figure 3.1. In the upper right figure the two disks are transformed to a wedge by the fractional linear transformation $w(z) = (z-\mu)/(z-\nu)$ for μ and ν defined in (3.5). Next, in the lower left figure, the wedge is opened up into the imaginary axis via $v(w) = w^{\pi/\alpha}$ for α defined in (3.5). In the lower right figure, the half plane is transformed to a disk by another fractional linear transformation $u(v) = (v-1)/(v+1)$.

To sum up, the conformal mapping of the exterior of two intersecting circles to the open unit disk is given by

$$G(z) = \frac{\left(\frac{z-\mu}{z-\nu}\right)^{\pi/\alpha} - 1}{\left(\frac{z-\mu}{z-\nu}\right)^{\pi/\alpha} + 1},$$

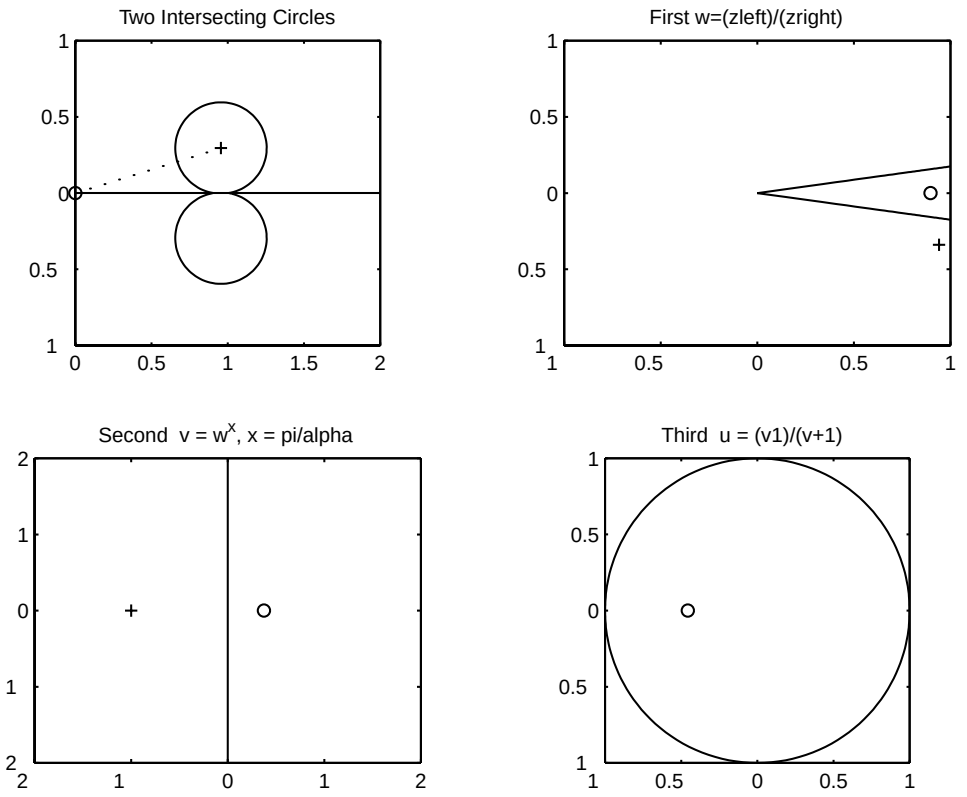

FIG. 3.2. *A conformal mapping between two intersecting disks (upper left) and a disk (lower right).*

where

$$\mu = \cos\theta - \sqrt{\eta^2 - \sin^2\theta}, \quad \nu = \cos\theta + \sqrt{\eta^2 - \sin^2\theta}, \quad \alpha = \pi - 2\tan^{-1}\left(\frac{\sin\theta}{\sqrt{\eta^2 - \sin^2\theta}}\right). \tag{3.5}$$

Also $w^{\pi/\alpha} = \exp(\frac{\pi}{\alpha}\log w)$ for $\log w = \log|w| + i\arg w$, and the branch for the argument function is placed along the negative real axis, $-i\pi < \arg w \leq i\pi$. In this case,

$$\kappa(\Omega) = (1 - (\mu/\nu)^{\pi/\alpha})/(1 + (\mu/\nu)^{\pi/\alpha}) = \eta + \frac{|\sin\theta|}{\eta\pi}(1-\eta)^2 + O(\theta^2). \tag{3.6}$$

3.3. Summary of K formulation properties. Based on the results of this section, we see that the K formulation differs from a true complex iterative solver only in the inner product space used by the iterative method. In other words, by utilizing the block entry data structures mentioned in section 2.2, we are able to provide the identical preconditioned matrix-multiply computations using the K formulation as we would for a true complex solver.

Furthermore, for complex Hermitian problems, there is no difference in asymptotic convergence rates. In fact, as is well known (e.g., see Problem 8.3.6 in [9]), if a complex matrix C is Hermitian (positive definite), the corresponding K matrix has the same eigenvalues, each with doubled multiplicity. Given the ability of the conjugate gradient

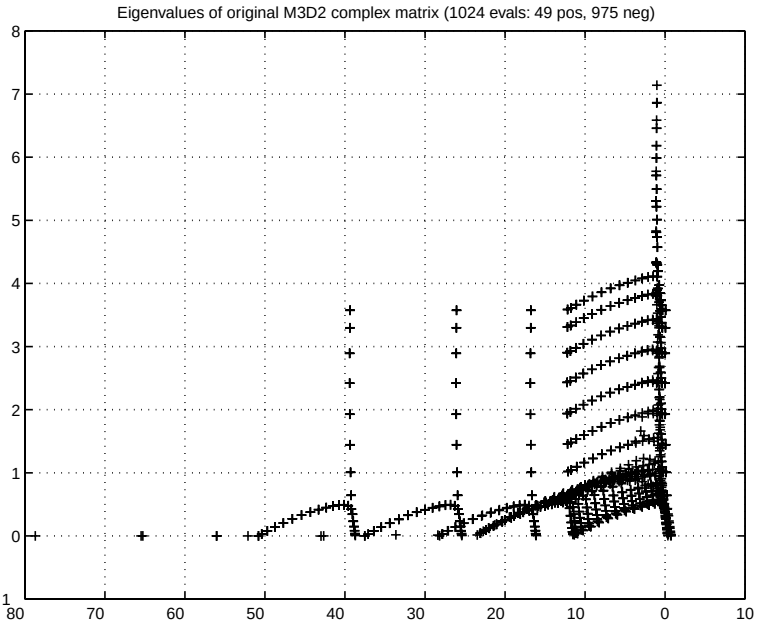

FIG. 3.3. *Eigenvalues of the original complex matrix in problem M3D2.*

method for linear systems to resolve multiple eigenvalues simultaneously, we observe in practice that the K formulation has identical convergence properties as a true complex solver for complex Hermitian problems.

For the non-Hermitian case, we saw from Examples 3 and 4 above that if the disk enclosing the spectrum of the preconditioned matrix C is not far from the point (1,0) in the complex plane, then the asymptotic convergence rate of the K formulation is close to the convergence rate of a true complex solver with the rate dictated by the size of the angle θ in Figure 3.1. As we will see in section 3.4, a high quality preconditioner tends to move the spectrum of C toward (1,0), setting up very favorable conditions for the K formulation.

3.4. Spectral case study. For problem M3D2 listed in Table 4.1, we computed the spectrum of the original and preconditioned matrix using the `eig` function of MATLAB. Figure 3.3 shows the distribution of the eigenvalues of the original matrix.[1] Figure 3.4 shows the eigenvalues of the K matrix, and, as expected, the eigenvalues of the K matrix are the eigenvalues of the complex matrix plus their reflection about the real axis.

Figure 3.5 shows the spectrum of the preconditioned matrix using `luinc(A,1e-1)` from MATLAB. `luinc` computes an incomplete LU factorization of a given sparse matrix. It provides several means to reduce the fill that would occur with an exact factorization. We chose to specify a drop tolerance. Given this drop tolerance, `luinc` will perform the LU factorization column by column as though it were doing an exact factorization, but as each column is computed, the terms in the column that are smaller in magnitude than the drop tolerance times the norm of the column are set to

[1] A note of thanks to Tom Wright and Nick Trefethen. They analyzed the pseudospectra of this matrix and determined that the eigenvalues of this matrix obtained via `eig` would be accurately computed.

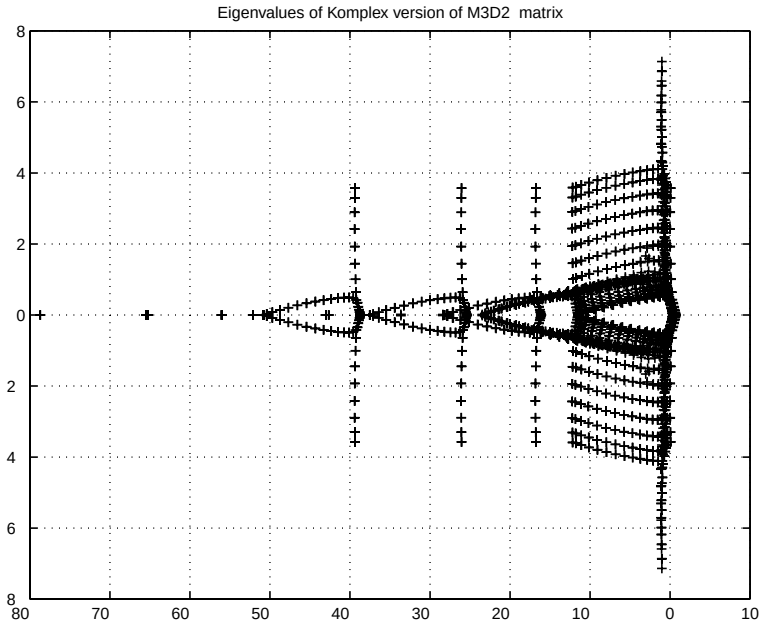

FIG. 3.4. *Eigenvalues of the K formulation matrix in problem M3D2.*

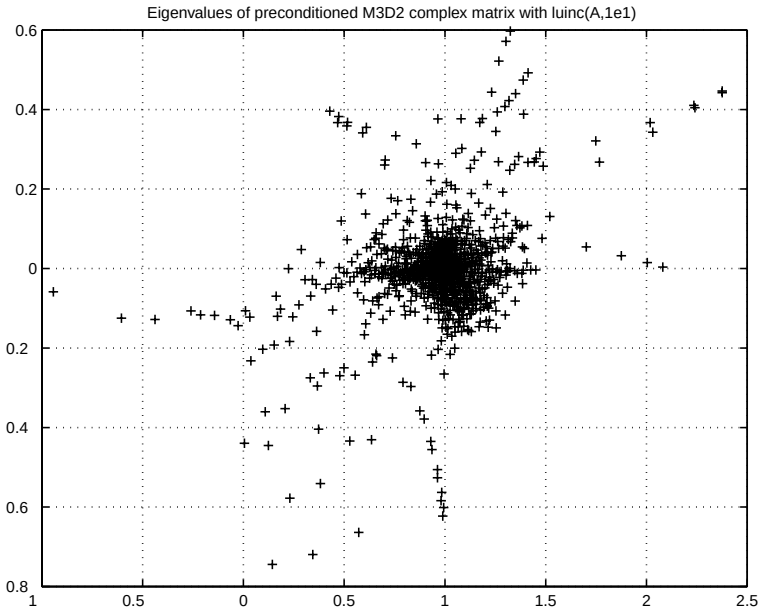

FIG. 3.5. *Eigenvalues of the complex matrix in problem M3D2, preconditioned by* `luinc(A,1e-1)`.

zero. `luinc(A,1e-1)` uses a drop tolerance of 10^{-1}. Note that the eigenvalues start to cluster around the point (1,0) in the complex plane.

Figure 3.6 shows the spectrum using `luinc(A,1e-2)`, which uses a drop tolerance of 10^{-2}. This preconditioner will typically keep more entries than `luinc(A,1e-1)` and

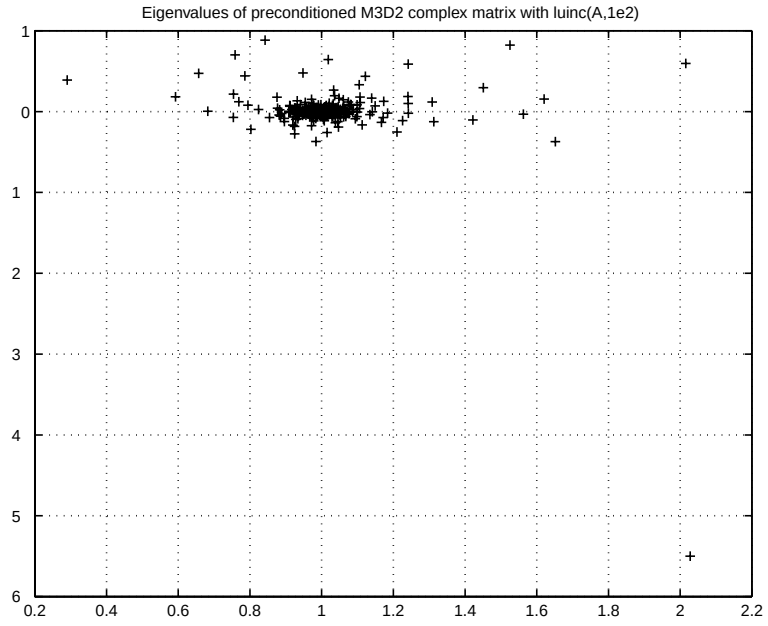

FIG. 3.6. *Eigenvalues of the complex matrix in problem M3D2, preconditioned by* `luinc(A,1e-2)`.

will be a better approximation to the exact LU factorization. With the exception of one outlyer, the eigenvalues in this case are closely clustered around (1,0).

Coupling this observation with the analysis from section 3.2, we see that high quality preconditioning, which tends to cluster the eigenvalues around the point (1,0), additionally minimizes the differences in asymptotic convergence rates between the true complex formulation and the K formulation. Thus, it simultaneously improves the convergence of both formulations and reduces the differences in convergence rates between them.

4. Computational results. We have used the K formulation to solve complex linear systems coming from two application areas (see Table 4.1). Each system comes from a real application. These problems are very ill conditioned in the sense that inexpensive preconditioners like Jacobi or block Jacobi are not sufficient for convergence. Eigenvalue problems with complex eigenvalues are important applications for complex linear solvers. Complex linear systems arise first in computing the eigenvalue using a complex shift-invert implicitly restarted Arnoldi method [11] and next in tracking the eigenvalues as a function of a parameter using Newton's method. Selecting a shift near to the eigenvalues of interest to accelerate the convergence of Arnoldi's method creates a linear system with a large spectral condition number. We have found the K formulation with ILU preconditioning to be very effective.

4.1. Overview of problems and solution methods. Our computational problems come from molecular dynamics and fluid dynamics. The first two problems listed in Table 4.1 come from given data sets where we are unfamiliar with the details of the applications. The last two problems come from efforts to understand the stability of a computed solution to a particular computational fluid dynamics problem using MPSalsa [18, 17]. The stability problems are representative of the linear

TABLE 4.1
Test problem descriptions.

Problem	Dim	# Nonzeros	Description
M3D2	1024	12480	Computational Chemistry Model I, Sherry Li, LBL/NERSC
M4D2	10000	127400	Computational Chemistry Model II, Sherry Li, LBL/NERSC
LINSTAB1	10590	276979	MPSalsa Linear Stability Analysis, Andrew Salinger, Rich Lehoucq, Cayley Transform Approach
LINSTAB2	10590	276979	MPSalsa Linear Stability Analysis, Andrew Salinger, Shift-and-Invert with shift equal to $33i$

systems solved in a complex shift-invert formulation of the eigenvalue problem.

MATLAB results were obtained using version 5.3.1 [12]. In particular, we used the built-in functions `luinc`, described in section 3.4, and `gmres`. `gmres` solves a linear system using the GMRES method [16] preconditioned by the ILU preconditioner computed by `luinc`.

The remaining results come from the Komplex Solver Package [10], an add-on module to Aztec 2.1 [19] that forms equivalent real systems from complex systems, uses Aztec to solve the real systems, and returns the complex results. For each of the problems we use overlapping domain decomposition preconditioning, specifically, additive Schwarz with one level of overlap, and we use GMRES(∞). For the M4D2 problem, the matrix has a natural block structure, and we used a block ILU preconditioner with a level fill of l (BILU(l)) on the subdomains, where we view the matrix as a sparse matrix whose entries are dense 8-by-8 matrices. The level fill parameter l refers to the pattern of the BILU factors. A level fill of $l = 0$ means that the only block entries kept are those that correspond to the pattern of the original matrix. A level fill of $l > 0$ is defined recursively to allow the BILU factors to have entries that correspond to the terms from the level fill $l - 1$ and any fill-in that comes directly from these terms. For LINSTAB1 and LINSTAB2 we replace BILU(l) with a variation of ILUT [14], a dual-threshold ILU preconditioner, as the subdomain preconditioner. The ILUT factors are computed row by row, dropping terms that fall below the threshold parameter multiplied by the norm of the row, and then keeping (at most) a prescribed number of the largest terms in the row. Details of the solvers and preconditioners can be found in [19].

TABLE 4.2
MATLAB test results using GMRES(∞) with luinc(droptol) preconditioning.

Problem	droptol	*nz(ILU)/nz(A)*	$\|r\|/\|b\|$	*C Iters*	*K Iters*
M3D2	1×10^{-3}	5.8	3×10^{-11}	12	12
	1×10^{-2}	4.5	8×10^{-11}	30	40
	1×10^{-1}	0.5	5×10^{-11}	107	181
M4D2	1×10^{-4}	13.1	5×10^{-11}	17	23
	1×10^{-3}	6.7	6×10^{-11}	72	109
LINSTAB1	1×10^{-3}	10.7	9×10^{-11}	71	93

4.2. Results. The first set of results (in Table 4.2) were obtained using a MATLAB code where we compare a true complex preconditioned iterative solver to the K formulation. For these problems, the preconditioned operators are exactly equivalent; i.e., using the notation from section 3, if M_C and M_K are the C and K preconditioners, respectively, then $M_K = f(M_C)$. Thus what we are comparing are the differences

due to having a complex inner product over n-space versus a real inner product over $2n$-space. Note that, as the quality of the preconditioner improves, the difference in iteration counts between the two approaches diminishes.

Our MATLAB results did not have any relevant solution time statistics, so we cannot precisely measure the relative costs. However, the results in Table 4.3 show that higher quality preconditioners also provide the best time to solution, up to a point where the time spent constructing and applying the higher quality preconditioner exceeds the reduction in time due to fewer iterations. These results are from the Komplex Solver Package [10], an add-on module to Aztec. We used a BILU preconditioner with 8-by-8 blocks and GMRES(∞). The best time to solution comes from BILU with a level fill of 2. This result suggests a general observation that a high quality preconditioner provides both the best time to solution and makes the difference in iteration counts between the true complex and the K formulations minimal.

TABLE 4.3
Komplex test results for M4D2 using GMRES(∞) with block ILU preconditioning, 8-by-8 blocks.

Problem	l	$\|r\|/\|b\|$	*K Iters*	Time(s)
M4D2	0	1×10^{-11}	322	354
	1	1×10^{-11}	179	182
	2	1×10^{-11}	75	95
	3	1×10^{-11}	60	125

The final set of results (in Table 4.4) comes from using the Komplex Solver Package to solve linear stability problems in computational fluid dynamics. The primary purpose of these results is to illustrate that, with a modest amount of new software development, we are able to provide a general-purpose parallel preconditioned iterative solver for complex-valued linear systems by leveraging existing real-valued solvers. Results are given for 1, 2, 4, 8, and 16 processors of an 8-node, 16-processor PC-based Beowulf [13] cluster. The second column of this table lists the ratio of nonzero entries in the ILU factors to the original matrix. In all cases we iterated until the scaled residual (the norm of the residual divided by the norm of the right-hand side) was below 10^{-13}. The increase in the ILU fill is due to the effect of overlap used by our parallel overlapping Schwarz preconditioners. Another trend we see is that the iterations increase as we add processors, an additional effect that is common with these preconditioners. However, even with these increasing serial costs, the parallel timings are quite reasonable.

TABLE 4.4
Komplex test results using GMRES(∞) with ILUT preconditioning.

Problem	*nz(ILU)/nz(A)*	$\|r\|/\|b\|$	# Proc	*K Iters*	Time(s)
LINSTAB1	2.0	1×10^{-13}	1	27	151.4
	2.2		2	38	75.6
	2.4		4	55	42.6
	2.7		8	64	22.7
	2.8		16	79	13.3
LINSTAB2	2.0	1×10^{-13}	1	49	158.7
	2.2		2	57	80.8
	2.4		4	83	45.8
	2.7		8	93	26.3
	2.8		16	112	16.7

5. Conclusions. In this paper we presented a discussion of how to solve complex-valued linear systems via equivalent real formulations. We listed approaches that failed and presented the K formulation, which works very well. Although it is clear from our results that the K formulation is not superior to a true complex solver, and clearly if you have easy access to a complex-valued solver you should use it, we do think that equivalent real formulations should receive more attention than they have in the past.

For many challenging problems, a high quality preconditioner is a requirement for convergence. Such a preconditioner has the tendency to map the spectrum around the point (1,0) in the complex plane. This, in turn, as the analysis in section 3.2 shows, minimizes the spectral difference between a true complex-valued iterative solver and the K formulation and leads to our observation that the requirement of a high quality preconditioner simultaneously provides the best solution times and diminishes the convergence differences between a true complex iterative solver and the K formulation.

Finally, for applications such as linear stability analysis, where all operators are real-valued except for the presence of complex shift value, the equivalent real formulation can be very attractive since it utilizes the native real-valued solver, and installation of the K formulation usually involves a modest amount of extra effort on the part of the application developer.

REFERENCES

[1] S. Balay, W. Gropp, L. McInnes, and B. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools for Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser Boston, Boston, 1997, pp. 163–202.

[2] S. Balay, W. Gropp, L. McInnes, and B. Smith, *PETSc* 2.0 *Users Manual*, Tech. report ANL-95/11—Revision 2.0.22, Argonne National Laboratory, Argonne, IL, 1998.

[3] S. Balay, W. Gropp, L. McInnes, and B. Smith, *PETSc home page*, http://www.mcs.anl.gov/petsc, 1998.

[4] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.

[5] B. Fischer, *Polynomial Based Iteration Methods for Symmetric Linear Systems*, 1st ed., Wiley, Chichester, UK, Teubner, Stuttgart, 1996.

[6] V. Fraysee, L. Giraud, and S. Gratton, *A Set of GMRES Routines for Real and Complex Arithmetics*, Tech. report TR/PA/97/49, CERFACS, Toulouse, France, 1997.

[7] R. W. Freund, *Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 425–448.

[8] R. W. Freund and N. Nachtigal, *QMRPACK: A package of QMR algorithms*, ACM Trans. Math. Software, 22 (1996), pp. 46–77.

[9] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[10] M. A. Heroux, *The Komplex Solver Package Reference Manual* 1.0, Sandia National Laboratories, Albuquerque, NM, 2000.

[11] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide*, SIAM, Philadelphia, 1998.

[12] MathWorks, *MATLAB home page*, http://www.mathworks.com, 2000.

[13] P. Merkey, *Beowulf home page*, http://beowulf.gsfc.nasa.gov, 1999.

[14] Y. Saad, *ILUT: A dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[15] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 1st ed., PWS Publishing Company, Boston, 1996.

[16] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[17] A. Salinger, K. Devine, G. Hennigan, H. Moffat, S. Hutchinson, and J. Shadid, *MP-Salsa: A Finite Element Computer Program for Reacting Flow Problems Part* 2—*User's Guide*, Tech. report SAND96–2331, Sandia National Laboratories, Albuquerque, NM, 1996.

[18] J. N. Shadid, H. K. Moffat, S. A. Hutchinson, G. L. Hennigan, K. D. Devine, and A. G. Salinger, *MPSalsa: A Finite Element Computer Program for Reacting Flow Problems Part* 1—*Theoretical Development*, Tech. report SAND95–2752, Sandia National Laboratories, Albuquerque, NM, 1995.

[19] R. S. Tuminaro, M. A. H. S. A. Hutchinson, and J. N. Shadid, *Official Aztec User's Guide, Version* 2.1, Sandia National Laboratories, Albuquerque, NM, 1999.