A comparison of preconditioners for complex-valued matrices

Daniel Osei-Kuffuor* Yousef Saad*

December 23, 2007

Abstract

A number of applications in quantum and plasma physics produce linear systems of equations that can be very challenging to iterative methods. Often these systems are in the complex field and they tend to be highly indefinite, which makes it difficult to extract an effective preconditioner. This paper primarily presents a comparative study of four preconditioners for complex-valued matrices. We compare the memory efficiency and performance of the ARMS, ILUT, ILUC, and ILUK preconditioners on a specific application area stemming from the diffraction of an acoustic wave incident on a bounded obstacle (governed by the Helmholtz Wave Equation), and on other general problems from quantum and plasma physics. The paper also addresses briefly some of the approaches of formulating a complex system into an equivalent real form, to make them feasible with real preconditioners and iterative solvers.

1 Introduction

When solving a linear system with a complex coefficient matrix, one is faced with two possible broad choices. First, it is possible to treat the problem as complex and to extend any of the techniques developed for real matrices to deal with the complex case. The second is to convert the complex system into a real one whose size is double that of the original system. Here again there are several possible choices for performing this conversion and some choices are better than others. The main question that is addressed is: does the conversion to real systems make it intrinsically more difficult to obtain an effective preconditioner than for the original complex system? This has been examined by Freund [14]. Freund showed that aside from the increase in size, converting a complex matrix into an equivalent real form, creates a coefficient matrix whose spectra are reflected about the real and/ or imaginary axes of the complex plane, thus making such a system unfavorable for solution by Krylov methods. Later, extensive research on this issue has been conducted, see, e.g. [9, 24, 4, 1]. Heroux and Day [9] studied various ways of reformulating a complex matrix into real form, with emphasis on efficient block (diagonal) preconditioning techniques. They showed that an effective/robust preconditioner is capable of minimizing the spectral difference between a true complex iterative solver, and a real formulation.

*Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E, Minneapolis, MN 55455. Email: {dosei, saad}@cs.umn.edu. This work was supported by NSF under grant ACI-0305120, by DOE under grant DE-FG02-03ER25585, and by the Minnesota Supercomputer Institute.

In this paper, we present a comparative study of the application of various complex-valued preconditioners to the solution of problems arising from applications in quantum and plasma physics. The preconditioners considered here are the Incomplete LU factorization with level of fill (ILUK) [27], the Incomplete LU Factorization with threshold (ILUT) [27], the Crout version of ILUT (ILUC) [21, 20, 6], and the Algebraic Recursive Multilevel Solver [30, 22, 32], with non-symmetric permutations (ARMS-ddPQ).

We are interested in the comparison of these preconditioners based on their memory efficiency and convergence properties, and we show that for the most difficult problems, ARMS-ddPQ generally performs better.

For the specific application problem governed by the Helmholtz equation, the task is to solve a complex-valued linear system resulting from the diffraction of an acoustic wave, at high wave numbers [18, 13, 2, 19, 31]. We determine an appropriate range for the wave number, for which the problem is hardest, and subsequently perform the analysis and comparison of the various preconditioners.

For the general case, we conduct tests using matrices derived from various applications from the Harwell-Boeing sparse matrix collection [10], and the University of Florida's sparse matrix collection [8]. We select a few interesting problems from these collections for use as additional test cases for our analysis.

The rest of the paper is organized as follows. In section 2, we describe our application areas - specific and general - and discuss any modifications or assumptions used in our study. We then present the four preconditioners involved in our study in section 3, and introduce the parameters used in our experiments. We also give a brief discussion of some of the approaches for converting a complex system into an equivalent real form, with respect to the acoustic diffraction application area. In section 4, we give a comparative analysis of results for the different preconditioners on the application problems, and also discuss and show the results of some equivalent real formulations of the complex system using a real iterative solver. We draw conclusions on our observations in Section 5.

2 Introduction of Application Areas

2.1 Acoustic Wave Diffraction Application

The Helmholtz equation is a partial differential equation which governs the propagation of an electromagnetic wave through a medium. Mathematically defined as

$$(\Delta + k^2)\Phi = f, \tag{1}$$

where f represents a harmonic source, and k represents the wave number, the Helmholtz equation possesses rich mathematical and physical properties. When the wave number k is zero the above equation reduces to the basic Poisson equation. When $k^2 < 0$, the above equation represents the spatial part of the traditional wave (diffusion) equation.

The wave number is directly related to the wavelength λ, by the relation $k = \frac{2\pi}{\lambda}$, and to the speed of the wave ω, by $k = \frac{\omega}{c}$, where c is the speed of light in the vacuum.

We use the finite element code developed in [17, 18] to deal with problems of acoustic diffraction. The code simulates the diffraction of a wave originating from infinity through an

open medium, incident on a bounded obstacle with boundary $\Gamma = \partial\Omega$, of circular shape. The corresponding boundary value problem (BVP) characterized by the Helmholtz equation is as follows:

$$\begin{cases} \Delta u + k^2 u &= f, \text{ in } \Omega \in \mathbb{R}^2 \\ u &= \alpha, \text{ on } \Gamma \\ \lim\limits_{r\to\infty} \sqrt{r}\left(\dfrac{\partial u}{\partial n} - iku\right) &= 0 \end{cases} \tag{2}$$

where α is determined by the use of Bessel functions. The last equation in the above BVP is referred to as the Sommerfeld radiation condition and it models the non-reflecting condition at the boundary, thereby guaranteeing a unique solution to the above problem.

This BVP is however not suitable for solution via the finite element method, primarily because of the condition that the incident wave originates from infinity. The problem is therefore reformulated to introduce an artificial boundary, by enforcing the so-called Dirichlet-to-Neumann (DtN) technique. Thus, the radiation condition at infinity is replaced by a boundary condition on the artificial boundary, Γ_{art}. The resulting problem becomes:

$$\begin{cases} \Delta u + k^2 u &= f, \text{ in } \Omega \in \mathbb{R}^2 \\ u &= \alpha, \text{ on } \Gamma \\ \frac{\partial u}{\partial n} &= -Mu \text{ on } \Gamma_{art} \end{cases} \tag{3}$$

where M denotes the DtN operator. Discretization of this resulting BVP was done via variational formulation, using the Galerkin Least-Squares discretization scheme. Details of the reformulation of the BVP and the finite element discretization of the result may be found in the paper by Kechroud at al. [17] and references therein.

2.1.1 Methods and Assumptions

We note that as with the physical problem, the properties of the operator in (1) will be highly dependent on the value of k. At high values of k, we are dealing with waves of very short wavelengths. This result, known in the literature as the "very short wave problem", has been recognized as one of the most challenging problems for modern numerical techniques [2, 33].

To compare the performance of preconditioners on difficult problems in this area therefore, it is reasonable to choose values of k that are high. For most applications governed by the Helmholtz equation, whose discretization is based on the finite difference scheme, the discretization is controlled by the relation $kh = \frac{2\pi h}{\lambda} = \alpha$, where α is some small conveniently chosen constant, and h is the mesh size. In other words, the number of grid points per wavelength is fixed to be constant. Hence a large value of k (or a small value of λ) corresponds to a small mesh size. Although this relation fosters a more accurate solution to the "very short wave" problem, reducing h for a correspondingly high k invariably results in a larger system to solve and thus a more difficult problem. Since we are interested in the performance of the solvers, we do not attempt to enforce the above requirement on the mesh-size. We use instead a finite element discretization on a fixed mesh size which results in a fixed size linear system, and we vary the wave number k. As before, the formulation of a difficult problem based on this approach also depends on k, and can be deduced by looking at the spectrum of the operator. From (1), we note that the definiteness (or indefiniteness) of the operator is controlled by k^2

(and hence k). For small values of k, the spectrum of the operator lies on the left (negative definite). For very large values of k, the spectrum is shifted to the right (positive definite). And for some intermediate k, the spectrum lies both on the left and right of zero (indefiniteness). The hardest case, from the point of view of indefiniteness, would be to pick k such that $k^2 \approx \lambda^*$, where λ^* is the median of the spectrum of the operator. Like the earlier approach, this choice of k results in a problem whose solution is numerically very challenging.

From the finite element discretization, we obtain the stiffness matrix $[K]$, and mass matrix $[M]$, related in a simplified form as follows:

$$([K] - k^2[M])\{u\} = \{f\} = \{0\}.$$

To obtain values of k, for which the system will be hard to solve, we compute the generalized eigenvalues of the pair $[K], [M]$, and choose k's so that k^2 is close to λ^*.

From the description of our problem, and the finite element discretization, we note that the stiffness and mass matrices are both real-valued. Complexity in our system was thus introduced by the effect of the DtN condition in the discretization at the artificial boundary [18]. The wave number k, is defined as a complex number, and implemented as a multiple of π. We typically choose k such that k is purely real. Choosing k with a non-zero imaginary part may be understood as implementing spatial effects on the wave -(attenuations)- into our model. The resulting discretized linear system to be solved has a coefficient matrix A that is sparse, symmetric and complex. It is however neither Hermitian nor diagonally dominant. In our experiments, we consider values of k for which the system is fairly difficult and most difficult to solve, and present the results on the performance of the various preconditioners.

2.2 Other General Application Areas

We also considered linear systems originating from a number of applications obtained from the Harwell-Boeing Sparse Matrix Collection (HBSMC) from the Matrix Market[1] [10], and the University of Florida Sparse Matrix Collection (UFSMC) maintained by Tim Davis [8]. We make our choices based on problem difficulty as well as interesting structural characteristics. The matrices considered here, with their corresponding application areas, include:

- **ted_AB.cua** - FEM coupled linear thermoelasticity equation on a bar. (D. Bindel, UC Berkeley. (UFSMC)).
- **dwg961b.cua** - From a generalized eigenvalue problem in Dispersive Waveguide Structures. (S. Gedney, U.D. Navsariwala, U of Kentucky. (HBSMC)).
- **young4c.cua** - From modeling acoustic scattering phenomena (also governed by the Helmholtz Equation). (David Young, Boeing Computer Services. (UFSMC, HBSMC)).
- **MHD set** - Matrices from the MHD set of magnetohydrodynamics applications from plasma physics. (Booten et. al. U. of Utrecht, Netherlands. (HBSMC)).

Matrices in the Matrix Market format were converted to Harwell-boeing format, for compatibility with our code, with the BeBOP Sparse Matrix Converter utility [15].

[1] http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/

3 Preconditioners and Solution Method

Our comparison focuses on iterative methods for solving the linear systems discussed above. Specifically, we employ preconditioned Krylov subspace methods, with the accelerator GMRES [29, 27], coupled with one of four different ILU (Incomplete LU Factorization) - based preconditioners. Next, the preconditioners implemented in our iterative approach will be briefly described.

3.1 Incomplete LU Factorization with Level of Fill (ILU(k))

ILU(k) is the incomplete LU factorization with level of fill see, e.g., [27], where k indicates fill-level. Level of fill here refers to the order of the fill-ins generated while performing the incomplete LU factorization.

In ILU(k), all fill-in elements with level of fill not exceeding k are kept, and the rest discarded. The idea behind this is that the level of fill of a particular fill-in element gives an indication of the size of that element. The higher the level of fill, the greater the number of fill-ins allowed, and the more exact the approximation. Thus an ILU(0) preconditioner drops all fill-ins generated during the factorization, whereas ILU(1) and ILU(2) drop all fill-ins beyond the first and second levels respectively.

3.2 Incomplete LU Factorization with Threshold Dropping (ILUT)

ILUT is another ILU-based preconditioner that implements dropping strategies based on some threshold parameters [26, 27]. A common approach is the dual truncation technique - ILUT(τ,p) - in which dropping during the factorization is based on two parameters: the drop tolerance (droptol $= \tau$), and the fill level (lfil $= p$).

The preconditioner is constructed based on the following rules:

- During the elimination for a particular row i, an entry $a_{i,k}$ is dropped if $|a_{i,k}| < \tau_i$, where τ_i, the relative tolerance, is the product of τ and the original norm of the i-th row.
- After a row has been updated, apply the previous dropping rule once again to discard all updated elements that are less than the relative tolerance. Then keep only the p largest elements in the L and U parts of the row respectively.

The drop tolerance, τ, serves to reduce computational cost, while the parameter lfil, p, reduces memory usage, by controlling the number of entries kept per row [27].

It is possible for the ILUT approach to fail for a number of reasons. The most obvious of these is when the original matrix A, contains a zero pivot (i.e. a zero diagonal entry). A remedy to this problem is to implement column pivoting. This leads us to a column pivoting variant of the ILUT called ILUTP [27]. ILUTP will usually result in a more robust preconditioner than ILUT, but this often comes with an additional cost in memory usage as ILUTP will tend to generate more fill-in. In both cases of the ILUT preconditioner construction, the rule of thumb is to take a large lfil value, and use the droptol to control the amount of fill-in. This generally yields good results without compromising memory efficiency.

3.3 Crout Incomplete LU Factorization (ILUC)

The Crout ILU [21], is similar to the other ILU-based approaches, with the main difference being the variant of the implemented factorization. Most ILU-based approaches such as ILU(k) and ILUT, make use of the ikj variant of the Gaussian Elimination algorithm (also called left-looking column LU factorization, for column versions of the algorithm, or upward looking for row versions).

As the name suggests, the Crout ILU approach on the other hand implements the Crout variant of the Gaussian elimination. This variant of the Gaussian Elimination essentially applies the ikj variant to compute the U part of the factorization, and a transposed version computes the L part. Thus the k^{th} step computes the $U(k, k:n)$ and the $L(k:n, k)$ parts of the factorization [21, 16, 12].

Like the ILUT approach, ILUC also implements the dual truncation technique in a similar way as described by the dropping rules above. Here too, pivoting may be implemented to obtain the more robust ILUCP version [20, 6].

3.4 Algebraic Recursive Multilevel Solver (ARMS)

3.4.1 Standard ARMS

ARMS [30] is a multilevel ILU preconditioner (a combination of ILU-based techniques with multilevel techniques), which implements a recursive solution to the construction of the preconditioner. The general mode of operation in the construction of a multilevel preconditioner is to first separate the points in the matrix into two subsets corresponding to the "coarse" set and the "fine" set. A block factorization of the matrix is then obtained from this partitioning and the process is continued recursively on the Schur complement associated with the coarse set. A brief sketch of the algorithm is now given.

In what follows, lev is the current level of the factorization: $0 \leq lev \leq last_{lev}$. A key step in ARMS is to find an independent set ordering P which permutes a matrix into the form

$$PAP^T = \begin{pmatrix} B & F \\ E & C \end{pmatrix},$$

where B is a block diagonal matrix corresponding to an independent set. Then, at level lev of the recursion, the ARMS procedure reorders the matrix in the above form and then factors it as follows,

$$P_{lev} A_{lev} P_{lev}^T = \begin{pmatrix} B_{lev} & F_{lev} \\ E_{lev} & C_{lev} \end{pmatrix} \approx \begin{pmatrix} L_{lev} & 0 \\ E_{lev} U_{lev}^{-1} & I \end{pmatrix} \begin{pmatrix} U_{lev} & L_{lev}^{-1} F_{lev} \\ 0 & A_{lev+1} \end{pmatrix}, \tag{4}$$

where P_{lev} and P_{lev}^T are the permutation and its transpose, respectively.

The idea then is to consider the matrix A_{lev+1} as the matrix of the next level and process it in turn by using the above strategy recursively.

ARMS(A_{lev}) factorization

1. If $lev = last\_lev$ then
2. Compute $A_{lev} \approx L_{lev}U_{lev}$
3. Else:
4 Find an independent set permutation P_{lev}
5. Apply permutation $A_{lev} := P_{lev}^T A_{lev} P$
6. Compute block factorization
7. Call ARMS(A_{lev+1})
8. EndIf

The $last_{lev}$ variable in the above pseudocode is determined explicitly by the ARMS parameter $nlev$, which is the maximum number of levels for the recursion, or implicitly by the size of the Schur complement constructed. The size of the Schur complement, A_{lev+1}, is compared to the ARMS parameter $bsize$, which determines the minimum size of the Schur block. Hence if the size of A_{lev+1} is less than $bsize$, then $lev+1 = last_{lev}$. Note that if $nlev = 0$ or $bsize \geq |A|$, then $last_{lev} = 0$ and the ARMS preconditioning operation is similar to an ILUT or ILUTP operation, depending on which approximation technique is used in step 2. In addition, to these two ARMS parameters, ARMS also includes the usual ILU parameters, droptol and lfil, for dropping during the factorization at the intermediate levels, as well as the last level. Another key feature that adds to the robustness of this approach is that in step 2 of the factorization, different approximations may be used to solve the last system. In addition, Step 4 includes a simple heuristic to improve the quality of the resulting factors in the process: all rows that are judged poor from the point of view of diagonal dominance, are not considered as part of the independent set [30].

For very poorly structured problems, a further improvement can be achieved by considering nonsymmetric permutations instead of the simple strategies based on independent sets. This leads to ARMS with the diagonally-dominant PQ ordering discussed next.

3.4.2 ARMS-ddPQ

To improve robustness and performance on systems with poorly structured matrices, many preconditioners have been developed which make use of some form of permutation in their construction, see for instance [5, 11, 25].

ARMS-ddPQ[2] is an extension of the standard ARMS that relies on nonsymmetric permutations to produce a better quality factorization. This is a two-sided approach where permutations P and Q are applied to the rows and columns respectively, to transform a matrix A into

$$PAQ^T = \begin{pmatrix} B & F \\ E & C \end{pmatrix}. \tag{5}$$

Unlike in standard ARMS, there is no particular structure and no assumptions for the B block. The permutation pair P,Q is selected such that the B block has the most diagonally dominant rows after the nonsymmetric permutation, and a few nonzero elements, so as to reduce fill-in.

[2] P and Q refer to the permutations used, and "dd" indicates that the ideal permutation would yield a diagonally dominant B block. [28]

At level lev of the above algorithm, the coefficient matrix is reordered as in (5), and the following block factorization is approximately computed (instead of (4)), [28]

$$P_{lev}A_{lev}Q_{lev}^T = \begin{pmatrix} B_{lev} & F_{lev} \\ E_{lev} & C_{lev} \end{pmatrix} \approx \begin{pmatrix} L_{lev} & 0 \\ E_{lev}U_{lev}^{-1} & I \end{pmatrix} \begin{pmatrix} U_{lev} & L_{lev}^{-1}F_{lev} \\ 0 & A_{lev+1} \end{pmatrix}. \tag{6}$$

The permutations P and Q are obtained in 3 steps:

1. Preselection Step: Filter out poor rows and sort the selected rows. This ensures that the selected rows best satisfy the diagonal dominance criterion.

2. Matching Step: Scan the candidate entries in the order given by the preselection, and accept them into a matching set M, or reject them. Here M is a set of n_M pairs (p_i, q_i), where $n_M \leq n$, the size of the matrix, and $1 \leq p_i, q_i \leq n$, for $i = 1, ..., n_M$ and $p_i \neq p_j$ for $i \neq j$, and $q_i \neq q_j$ for $i \neq j$. See [28, 23] for a few approaches to construct the matching set.

3. Matching Set Completion: The case where $n_M = n$ in the Matching step corresponds to a full permutation pair (P, Q). However, for $n_M \leq n$, the partial matching set M can be easily completed to a complete matching set (full pair of permutations (P, Q)) by a greedy approach.

Additional details on all the preconditioners described above can be found in a number of references, see, e.g., [27, 21, 20, 6, 30, 22, 32, 28, 23].

3.5 Equivalent Real Formulation

The main reasons for wanting to use real instead of complex arithmetic are (*i*) more convenient coding and (*ii*) availability of software codes for solving real-valued systems. A secondary consideration is whether or not there could be any intrinsic gain in an approach based on real-valued formulations versus complex formulations.

Consider the complex-valued linear system

$$Ax = b, \tag{7}$$

where A is an m x n known complex matrix, b is a m x 1 known right-hand side, and x is the n x 1 unknown solution vector. Decomposing (7) into its real and imaginary parts, we obtain the following system:

$$(A_r + iA_c)(x_r + ix_c) = b_r + ib_c, \tag{8}$$

where the subscripts r and c identify the real and imaginary parts respectively. As mentioned earlier, there are a number of ways of converting (7) into real form, the choice of which is dependent on the type of problem. Considering the Helmholtz equation application described above, we note that the order of magnitude of the imaginary parts is small compared to that of the real parts of the entries in the complex matrix. Hence we have a choice of one, out of the two possible formulations shown below, and denoted as the $K1$ and $K2$ (see [9]), or the A_* and A_{**} (see [14]) formulations:

Matrix	Spectral/ Symmetry Properties
K1	(a) If $\lambda_k \in \Lambda(A)$, then λ and $\bar{\lambda} \in \Lambda(K1)$. (b) If A is Hermitian, $\Rightarrow$ K1 is symmetric. (c) If A is symmetric, $\Rightarrow$ K1 is symmetric in structure only. (d) If A is positive definite, $\Rightarrow$ K1 is positive definite.
K2	(a) If $\lambda \in \Lambda(K2)$, and $\lambda \in \mathbb{R}$, then $-\lambda \in \Lambda(K2)$. (b) If $\lambda \in \Lambda(K2)$, and $\lambda \in \mathbb{C}$, then $-\lambda$, $\bar{\lambda}$ and $-\bar{\lambda} \in \Lambda(K2)$. (c) If A is Hermitian, $\Rightarrow$ K2 is symmetric in structure only. (d) If A is symmetric, $\Rightarrow$ K2 is symmetric.

Table 1: Summary of the spectral and symmetry properties of the real formulations. We use $\Lambda(.)$ to denote the spectrum of a matrix. [9, 14]

K1 formulation:

$$\begin{pmatrix} A_r & -A_c \\ A_c & A_r \end{pmatrix} \begin{pmatrix} x_r \\ x_c \end{pmatrix} = \begin{pmatrix} b_r \\ b_c \end{pmatrix}. \tag{9}$$

K2 formulation:

$$\begin{pmatrix} A_r & A_c \\ A_c & -A_r \end{pmatrix} \begin{pmatrix} x_r \\ -x_c \end{pmatrix} = \begin{pmatrix} b_r \\ b_c \end{pmatrix}. \tag{10}$$

The choice of which formulation to use depends on the spectral properties of the complex matrix, and hence of the resulting equivalent real formulation. Table 1 shows the spectral and symmetry properties of the $K1$ and $K2$ formulations, with respect to the original complex matrix.

Since our complex matrix is symmetric (but not Hermitian), the fact that the $K2$ formulation preserves symmetry makes it an attractive choice. However, from the nature and spectral properties of this formulation, and from Proposition 5.1(b) of [14], it is known that

$$\Lambda(K2) = \{\lambda \in \mathbb{C} \mid \lambda^2 \in \Lambda(\bar{A}A)\}.$$

Since $\bar{A}A$ is Hermitian, we have that $\Lambda(\bar{A}A) \geq 0$, and hence $\Lambda(K2) \subset \mathbb{R}$. Table 1 indicates that the spectrum of $K2$ is symmetric about the real (and imaginary) axes. This makes the problem difficult to solve by an iterative method like GMRES.

Opting for the $K1$ formulation does not help much. Unlike the case of $K2$, the spectrum of the $K1$ formulation is not symmetric about the imaginary axis. However, our choice of wave number k shifts the spectrum of the operator A such that the Hermitian part $(A + A^H)/2$ of A is indefinite. This implies that the spectrum for the $K1$ formulation also contains the origin [14], and hence it too will cause problems.

The authors of [9] propose an alternate approach that generally performs better than the $K1$ or $K2$ formulation, by preserving the sparsity structure of the original complex matrix A. In this approach, each entry in A is converted into a 2 x 2 block entry via the $K1$ (or $K2$) formulation. The resulting matrix (and system of equations) is twice the size of the original complex matrix, and is simply a permutation of the $K1$ (or $K2$) matrix. We shall refer to these as the $PK1$ and $PK2$ formulations respectively. An earlier discussion and implementation of this approach may be found in section (5) of [7].

Preconditioner	k	No. iters	Setup Time (s)	Iter. Time (s)
ILU(2)	10π	87	0.02	0.59
	60π	1500	-	-
	100π	1500	-	-
ILUC	10π	11	0.11	0.07
	60π	1500	-	-
	100π	3	0.1	0.02
ILUT	10π	11	0.1	0.07
	60π	1500	-	-
	100π	4	0.09	0.03
ILUTP	10π	10	0.08	0.07
	60π	1069	0.07	11.7
	100π	3	0.1	0.01
ARMS-ddPQ	10π	11	0.29	0.11
	60π	160	0.26	2.08
	100π	4	0.3	0.04

Table 2: Comparison of preconditioners on application problem with different wave numbers. Critical wave number $k^* = 60\pi$

4 Numerical Results and Discussion

4.1 Acoustic Wave Diffraction Application

We compare the above mentioned preconditioners to solve the acoustic wave diffraction problem discribed in section 2.1, with respect to the following physical setting. We model a plane wave propagating along the x-axis, and incident on a bounded obstacle in the form of a disk of radius 0.5m. We implement a second order Bayliss-Turkel boundary condition [3] at the artificial boundary, at a distance 0.5m from the obstacle. That is, at an outer radius of 1.0m. We discretize the system using an isoparametric discretization over quad elements. Although the analytic solution to this problem is known, we do not consider it here since we are more interested in comparing the performance of the preconditioners on the hardest problems.

In all cases, we allow a maximum of 1500 iterations and set the tolerance for stopping to $1.0\text{x}10^{-8}$. In table 2, we compare the performance of the preconditioners on a problem of size 7200. We allowed a fill factor of approximately 4.0. That is, the ratio of the number of non-zero entries in the preconditioned matrix, divided by that of the original matrix is approximately 4.0. Here, we also include the performance of the ILUTP preconditioner. For the ILUK preconditioner, we typically choose the level of fill parameter = 2. It must be noted that for hard problems such as this, more fill-in might be required in order to get a good result. ILU(0) was not considered here since it did not converge for wave numbers $\geq 7\pi$. The fill factors (or memory cost) for the ILUK preconditioner are typically smaller, due to the dropping technique imposed in this approach. The higher the value of the fill-level parameter, the bigger the fill factor, and usually, the better the convergence. The ILUK approach is ideal for easier problems, since the preconditioner construction is cheap. It may also be used as a means of detecting harder problems and hence as a criteria for selecting a more robust preconditioner. As observed from the table, we see that the performance of ILU(2) is limited, primarily due to the value of the fill-level parameter we chose. ILUC and ILUT both work well for k values below and above the critical value $k^* = 60\pi$, for which the problem is hardest. ILUTP works quite well, and

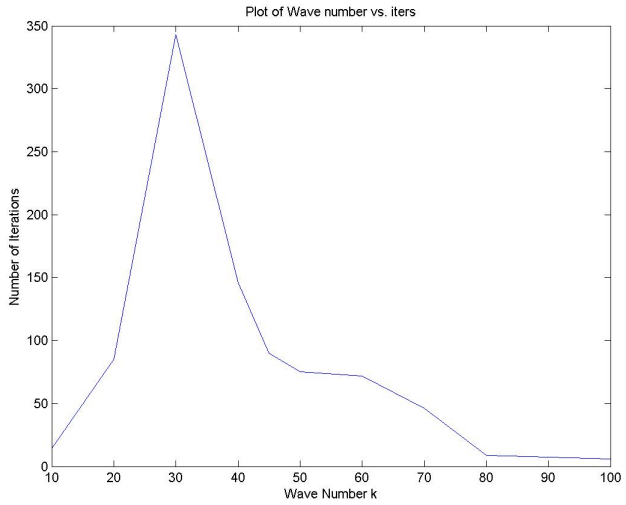


Figure 1: Convergence profile of ARMS-ddPQ over variable wave number k. (System size $= 1800$, and $k^* \approx 30\pi$)

manages to get some results for critical k. However, clearly ARMS-ddPQ outperforms the rest by converging after only 160 iterations, for the hardest case.

In order to better understand the effect of the wave number on the problem difficulty, we present a separate analysis on the performance of ARMS-ddPQ on a separate problem, again varying the wave number, and observing the number of iterations required for convergence. In figure 1, we depict the convergence profile of the ARMS-ddPQ preconditioner, over different values of k. To better understand this profile, we recall our earlier discussion on our choice of the critical wave number k^*. We observe a very sharp increase in the number of iterations required for convergence, for k between 10 and 30, and a corresponding decrease for k between 30 and 100. Although the wave number axis is coarse, it is still possible to deduce from the plot that the number of iterations increases as we shift the spectrum of the operator to the right. That is, as we increase k, the problem seems harder to solve. After shifting the spectrum past the critical k value, we observe a decreasing iteration count, corresponding to the problem becoming easier. This agrees very well with the earlier discussion that increasing k shifts the spectrum of the operator from negative definite (easy), to indefinite (hard), and then to positive definite (easy).

4.2 General Applications

In the tests which follow, we keep the fill factor for the various preconditioners to be ≤ 3.0. We chose the level of fill parameter for the ILUK approach to be the largest level of fill such that the resulting fill factor is ≤ 3.0. In table 3, we see that compared to ILUC, ILUT and ILUK exhibit inferior performance for all the test cases. In terms of memory efficiency, again ILUC performs better than ILUT and ILUK. However, comparing ARMS-ddPQ and ILUC, we see a comparable performance for the first two cases, with ARMS-ddPQ outperforming ILUC in the last case. In terms of memory efficiency, we observe a better memory efficiency in the

Matrix	Preconditioner	No. of iters	Iter. Time (s)	Setup Time (s)	Fill Factor
ted_AB.cua	ILU(1)	1000	-	0.84	2.0
	ILUT	67	1.83	12.3	2.0
	ILUC	8	0.15	0.22	1.0
	ARMS-ddPQ	8	0.15	0.34	1.65
dwg961b.cua	ILU(2)	96	0.11	0.0	2.0
	ILUT	93	0.1	0.0	2.0
	ILUC	10	0.01	0.01	2.0
	ARMS-ddPQ	7	0.02	0.01	3.0
young4c.cua	ILU(3)	102	0.08	0.0	2.48
	ILUT	34	0.02	0.0	2.89
	ILUC	31	0.02	0.0	2.69
	ARMS-ddPQ	15	0.0	0.01	2.2

Table 3: Comparison of Preconditioners on matrices from various applications from Matrix Market and U of Florida Sparse Matrix Collections.

first two cases, for ILUC. This is plausible since each intermediate level (elimination) operation can introduce some fill-in in the factorization at that level. However, we see a better memory efficiency for ARMS-ddPQ, compared to ILUC, in the last case.

Testing the preconditioners on the matrix mhd1280a.cua from the MHD set, ILUK, ILUT and ILUC all failed to construct a preconditioner to solve the system. This is because the matrix contains some zero diagonal entries (figure 2). However, ARMS-ddPQ, as well as variants of the ILUT and ILUC preconditioners that exploit pivoting, are able to construct an efficient preconditioner from this matrix by implementing some form of row or column pivoting to take care of the zero diagonals.

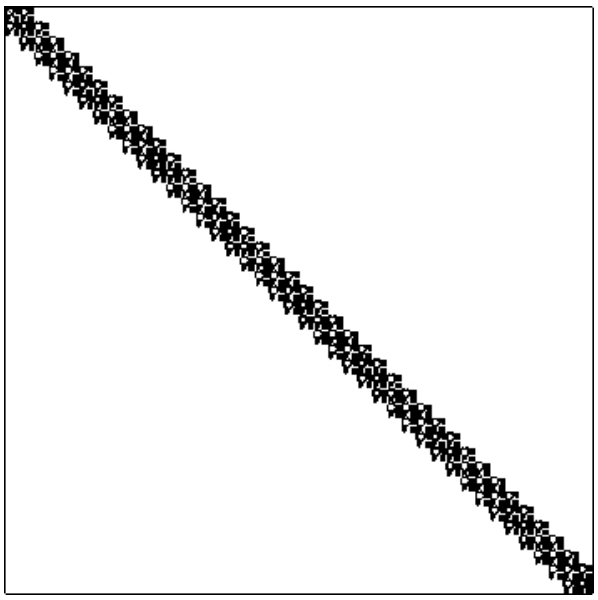


Figure 2: Structure of mhd1280a.cua matrix. Careful observation shows sparse regions along the main diagonal.

ARMS-ddPQ	No. of iters	Iter. Time (s)	Setup Time (s)	Fill Factor
Complex Matrix	8	0.01	0.02	2.1
$K1$	21	0.07	0.08	3.0
$K2$	22	0.07	0.07	3.0
$PK1$	10	0.02	0.03	3.0
$PK2$	10	0.02	0.02	3.0

Table 4: Comparison of ARMS-ddPQ on a complex system and its real formulations, from an application in acoustic wave diffraction governed by the Helmholtz Equation, with wave number $k = 30\pi$.

ILUTP	No. of iters	Iter. Time (s)	Setup Time (s)	Fill Factor
Complex Matrix	5	0.01	0.01	1.74
$K1$	45	0.10	0.07	3.0
$K2$	46	0.11	0.07	3.0
$PK1$	10	0.02	0.02	2.33
$PK2$	10	0.02	0.01	2.33

Table 5: Comparison of ILUTP on a complex system and its real formulations, from an application in acoustic wave diffraction governed by the Helmholtz Equation, with wave number $k = 30\pi$.

4.3 Equivalent Real Formulation Results

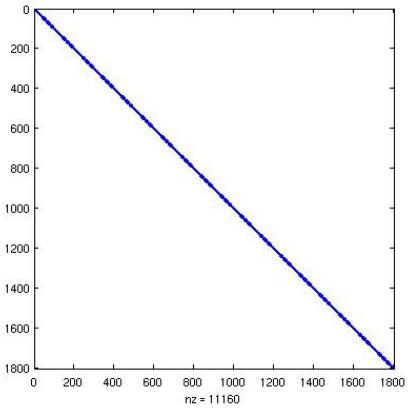
We compare the performance of the ARMS-ddPQ and ILUTP preconditioners to solve a small complex system, with the performance of the real versions of the preconditioners on the equivalent real formulations of the complex system. The test case is from the acoustic diffraction application with wave number $k = 30\pi$. The system size is 1800 (i.e 3600 for each real case). We allow a maximum fill factor of 3.0 for all test cases.

Table 4 shows comparative results for solving the complex system and the real formulations with the ARMS-ddPQ preconditioner. In Table 5, we present a similar comparison, this time using the ILUTP approach as the preconditioner for both complex and real cases.

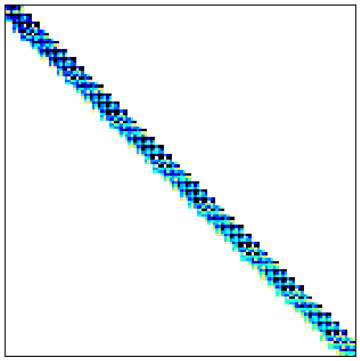
We observe from these tables that the PK formulations have better performance compared with the K formulations. This agrees well with the findings in [9]. Moreover, the results indicate that preconditioning the complex system yields better performance than any of the real formulations - a result which concurs with those of [14] and [9]. We see also that the results with the ARMS-ddPQ preconditioner are much closer to each other, compared with those of the ILUTP preconditioner. These results suggest that the ARMS-ddPQ preconditioner reduces the convergence differences between the complex GMRES solver and its real counterpart, applied to a real formulation of the complex system - a typical feature of a robust preconditioner [9].

4.4 Effect of Nonsymmetric Permutations

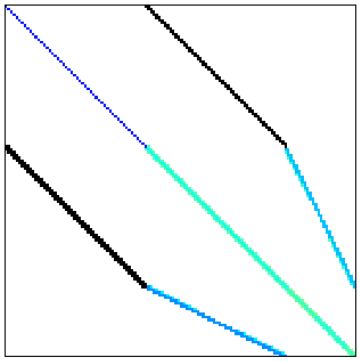
In our final test, we show the effect of the nonsymmetric permutation technique by comparing the convergence performance of ARMS-ddPQ with that of standard ARMS, on problems with different structures. The matrices considered in this test were obtained from applications in acoustic wave diffraction with a fairly high wave number to increase problem difficulty, mag-



(a)



(b)



(c)

Figure 3: Structures of matrices from: (a) An application in acoustic wave diffraction; (b) An application in MHD (from the Matrix Market); (c) An application in linear thermoelasticity [8]

netohydrodynamics, and linear thermoelasticity. Figure 3 shows the structure of the matrices considered. We ensured an identical fill factor for both preconditioners, in each case. Figure 4 shows the convergence profiles of both preconditioners in each of the three cases.

In all three cases, we see that the performance of ARMS-ddPQ is better than that of standard ARMS - twice as fast in the first case. For easier problems however, it is very possible that the standard ARMS preconditioner will perform just as well as (or sometimes even better) than ARMS-ddPQ. This is because if the matrix already possesses some nice structural properties, performing nonsymmetric permutations could destroy some of these properties, thus making the problem harder to solve. Nonetheless, for hard problems, ARMS-ddPQ consistently performs better.

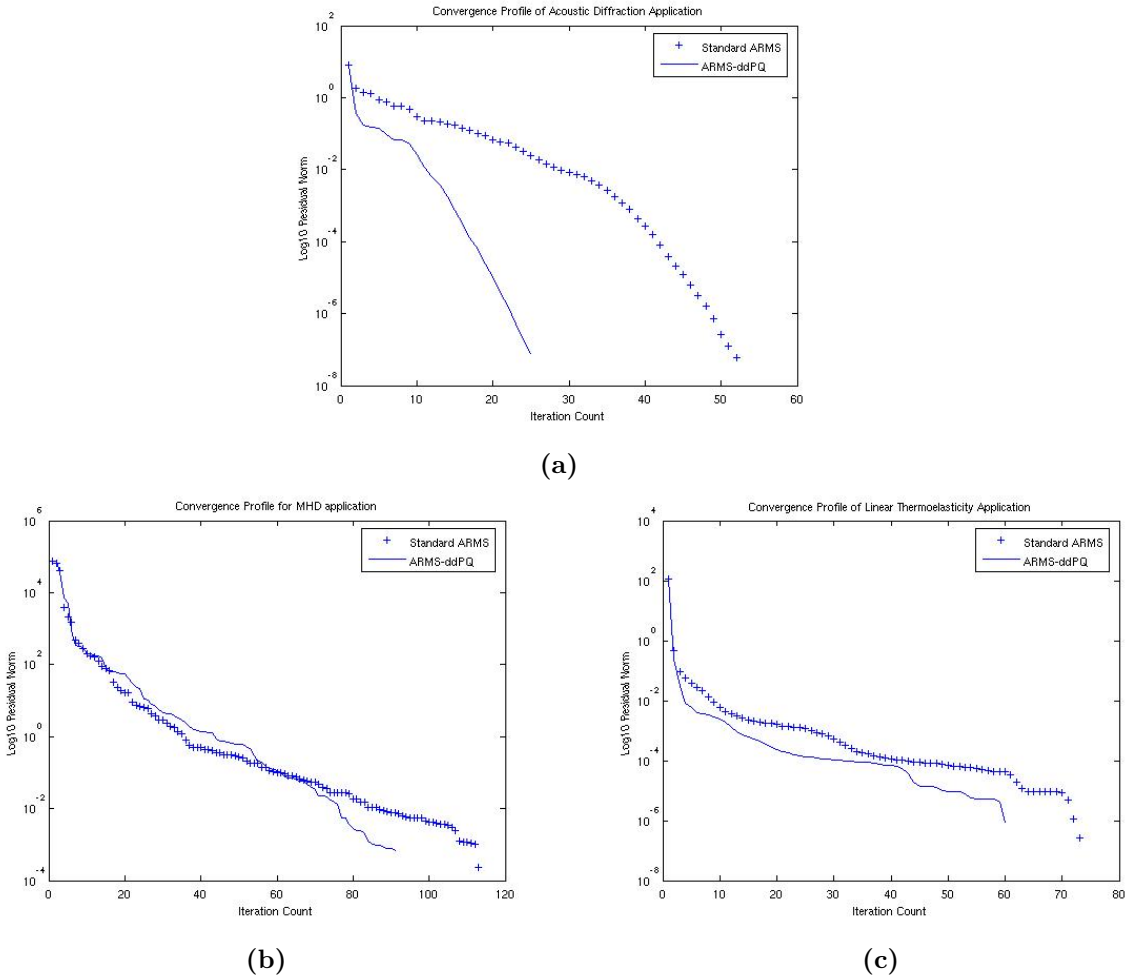


Figure 4: Convergence Histories - Iteration count vs. Residual norm. (a) - Application in acoustic wave diffraction; (b) - Application in MHD (from the Matrix Market); (c) - Application in linear thermoelasticity [8]

5 Conclusion

The above discussion confirms that for the majority of difficult problems, it is preferable to use a complex preconditioner instead of reformulating the problem into a real one for use with a real preconditioner. Among the complex preconditioners available, the family of ILU-based factorizations provide effective and relatively robust techniques.

In this paper, we have considered highly indefinite problems and observed that for such problems, standard ILU techniques encounter serious difficulties, unless some kind of nonsymmetric pivoting is performed in the incomplete factorization. Two techniques of this type are the ILUT factorization with pivoting (ILUTP) and ARMS with ddPQ ordering. We observed that ARMS-ddPQ performed better than ILUTP when taking into account both storage and computational cost. This is to be attributed to the multilevel nature ARMS-ddQP.

Like standard ARMS, ARMS-ddPQ also exploits a number of techniques to further enhance

performance. It attempts to reduce fill-in and preserve sparsity by dropping small terms, and includes quite a few parameters that can be adjusted to obtain a good preconditioner for the situation at hand.

References

[1] Adams, M. F. Algebraic multigrid methods for direct frequency response analysis in solid mechanics, 2006.

[2] Bao, G., Wei, G. W., and Zhao, S. Numerical solution of the Helmholtz equation with high wavenumbers. *Internat. J. Numer. Methods Engrg. 59*, 3 (2004), 389–408.

[3] Bayliss, A., Gunzburger, M., and Turkel, E. Boundary conditions for the numerical solution of elliptic equations in exterior regions. *SIAM Journal on Applied Mathematics 42*, 2 (1982), 430–451.

[4] Benzi, M., and Bertaccini, D. Real-valued iterative algorithms for complex symmetric linear systems, 2006.

[5] Benzi, M., Haws, J. C., and Tuma, M. Preconditioning highly indefinite and nonsymmetric matrices. *SIAM Journal on Scientific Computing 22*, 4 (2001), 1333–1353.

[6] Bollhfer, M. A robust ILU based on monitoring the growth of the inverse factors.

[7] Christodoulou, K. N., and Scriven, L. E. Finding leading modes of a viscous free surface flow: An asymmetric generalized eigenproblem. *J. Scient. Comput. 3* (1988), 355–406.

[8] Davis, T. University of Florida sparse matrix collection. NA Digest, vol. 92, no. 42, October 16, 1994, NA Digest, vol. 96, no. 28, July 23, 1996, and NA Digest, vol. 97, no. 23, June 7, 1997.

[9] Day, D., and Heroux, M. A. Solving complex-valued linear systems via equivalent real formulations. *SIAM J. Sci. Comput. 23*, 2 (2001), 480–498 (electronic). Copper Mountain Conference (2000).

[10] Duff, I. S., Grimes, R. G., and Lewis, J. G. Users' guide for the Harwell-Boeing sparse matrix collection (Release I), 1992.

[11] Duff, I. S., and Koster, J. On algorithms for permuting large entries to the diagonal of a sparse matrix. *SIAM Journal on Matrix Analysis and Applications 22*, 4 (2001), 973–996.

[12] Eisenstat, S. C., Schultz, M. H., and Sherman, A. H. Algorithms and data structures for sparse symmetric gaussian elimination. *SIAM Journal on Scientific and Statistical Computing 2*, 2 (1981), 225–237.

[13] Erlangga, Y. A., Vuik, C., and Oosterlee, C. W. On a class of preconditioners for solving the Helmholtz equation. *Appl. Numer. Math. 50*, 3-4 (2004), 409–425.

[14] Freund, R. W. Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices. *SIAM J. Sci. Stat. Comput. 13*, 1 (1992), 425–448.

[15] Hoemmen, M., Vuduc, R., and Nishtala, R. Bebop sparse matrix converter. http://bebop.cs.berkeley.edu/smc/.

[16] Jones, M. T., and Plassmann, P. E. An improved incomplete Cholesky factorization. *ACM Transactions on Mathematical Software 21*, 1 (1995), 5–17.

[17] Kechroud, R., Soulaimani, A., and Saad, Y. Preconditionning techniques for the solution of the Helmholtz equations by the finite element method. In *Proc. 2003 Workshop in wave phenomena in physics and engineering: New models, algorithms and applications* (2003), V. K. et al., Ed., ICCSA 2003, LCNS 2668, Springer-Verlag, pp. 847–858.

[18] Kechroud, R., Soulaimani, A., Saad, Y., and Gowda, S. Preconditioning techniques for the solution of the helmholtz equation by the finite element method. *Math. Comput. Simul. 65*, 4-5 (2004), 303–321.

[19] Lahaye, D., Vandewalle, S., and Hameyer, K. Finite element solution of the helmholtz equation with high wave number part ii: The h-p version of the FEM. *SIAM J. Numer. Anal. 34*, 1 (1997), 315–358.

[20] Li, N., and Saad, Y. Crout versions of the ILU factorization with pivoting for sparse symmetric matrices. Tech. Rep. umsi-2004-044, msi, uofmad, 2004. Appeared in ETNA, vol. 20 (2006), pp. 75-85.

[21] Li, N., Saad, Y., and Chow, E. Crout versions of ILU for general sparse matrices. Tech. Rep. umsi-2002-021, msi, uofmad, 2002.

[22] Li, Z., Saad, Y., and Sosonkina, M. pARMS: a parallel version of the algebraic recursive multilevel solver. *nlaa 10* (2003), 485–509.

[23] MacLachlan, S., and Saad, Y. Greedy coarsening strategies for non-symmetric problems. Tech. Rep. umsi-2006-xxx, msi, uofmad, 2006.

[24] Munankarmy, A., and Heroux, M. A comparison of two equivalent real formulations for complex-valued linear systems. *American Journal of Undergraduate Research 1*, 3 (2002).

[25] Olschowka, M., and Neumaier, A. A new pivoting strategy for Gaussian elimination. *Linear Algebra and its Applications 240*, 1–3 (1996), 131–151.

[26] Saad, Y. ILUT: a dual threshold incomplete *LU* factorization. *Numerical linear algebra with applications 1*, 4 (1994), 387–402.

[27] Saad, Y. *Iterative Methods for Sparse Linear Systems, 2nd edition*. SIAM, Philadelpha, PA, 2003.

[28] Saad, Y. Multilevel ILU with reorderings for diagonal dominance. *SIAM J. Sci. Comput. 27*, 3 (2005), 1032–1057.

[29] Saad, Y., and Schultz, M. H. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput. 7*, 3 (1986), 856–869.

[30] Saad, Y., and Suchomel, B. ARMS: An algebraic recursive multilevel solver for general sparse linear systems. *nlaa 9* (2002).

[31] Shen, J., and Wang, L.-L. Spectral approximation of the Helmholtz equation with high wave numbers. *SIAM J. Numer. Anal 43*, 2 (2005), 623–644.

[32] Sosonkina, M., Saad, Y., and Cai, X. Using the parallel algebraic recursive multilevel solver in modern physical applications. *Future Generation Computer Systems 20* (2004), 489–500.

[33] Zienkiewicz, O. Achievements and some unsolved problems of the finite element method. *Int. J. Numer. Mthds. Eng. 47* (2000), 9–28.