SIAM J. SCI. COMPUT.
Vol. 34, No. 1, pp. A48–A75


# MODIFICATION AND COMPENSATION STRATEGIES FOR THRESHOLD-BASED INCOMPLETE FACTORIZATIONS*

S. MACLACHLAN†, D. OSEI-KUFFUOR‡, AND YOUSEF SAAD‡

**Abstract.** Standard (single-level) incomplete factorization preconditioners are known to successfully accelerate Krylov subspace iterations for many linear systems. The classical modified incomplete LU (MILU) factorization approach improves the acceleration given by (standard) ILU approaches, by modifying the nonunit diagonal in the factorization to match the action of the system matrix on a given vector, typically the constant vector. Here, we examine the role of similar modifications within the dual-threshold ILUT algorithm. We introduce column and row variants of the modified ILUT algorithm and discuss optimal ways of modifying the columns or rows of the computed factors to improve their accuracy and stability. Modifications are considered for both the diagonal and off-diagonal entries of the factors, based on one or many vectors, chosen a priori or through an Arnoldi iteration. Numerical results are presented to support our findings.

**Key words.** incomplete factorization preconditioners, algebraic preconditioners, ILUT, modified ILU

**AMS subject classifications.** 65F08, 65N22, 65Y20

**DOI.** 10.1137/110834986

## 1. Introduction.

As physical models become ever more complex, they often result in the need to solve linear systems that are not only much larger than in the past but also intrinsically more difficult. Due to their larger sizes, these systems cannot practically be solved by direct methods, and this increases the demand for reliable forms of iterative methods that can be substituted for direct solvers. Iterative techniques based on a combination of a preconditioner and a Krylov subspace accelerator are the most common alternatives to direct methods, as they offer a good compromise between cost and robustness. Much of the recent research effort on solving sparse linear systems by iterative techniques has been devoted to the development of effective preconditioners that scale well while offering good reliability.

In this regard, multilevel methods that rely on incomplete LU (ILU) factorizations have been advocated by many authors in recent years [1, 5, 6, 7, 20, 22, 24, 25, 26, 32, 35]. While multigrid techniques [12, 38] and their algebraic counterparts (AMG) [31, 39] are known to be optimally efficient for solving some classes of discretized partial differential equations on regular meshes, they may become ineffective when faced with more general types of sparse linear systems. However, the "multilevel" or "multistage" ingredient of multigrid can be easily married with general-purpose qualities of ILU preconditioners to yield efficient, yet more general-purpose, solvers.

This paper does not aim at exploring new methods within the multilevel ILU class of techniques. It focuses instead on improving the basic component of ILU-based

*Submitted to the journal's Methods and Algorithms for Scientific Computing section May 23, 2011; accepted for publication September 23, 2011; published electronically January 31, 2012. This work was supported by NSF grants ACI-0305120 and DMS-0811022, DOE grant DE-FG-08ER25841, and by the Minnesota Supercomputing Institute and the Institute for Mathematics and Its Applications.

http://www.siam.org/journals/sisc/34-1/83498.html

†Department of Mathematics, Tufts University, 503 Boston Avenue, Medford, MA 02155 (scott.maclachlan@tufts.edu).

‡Department of Computer Science and Engineering, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455 (dosei@cs.umn.edu, saad@cs.umn.edu).

preconditioners, namely, the ILU factorization itself. Among the various options of ILU considered in the literature is the modified ILU factorization (MILU) proposed by Gustafsson [19] for the symmetric case (modified incomplete Cholesky or MIC). Note that for five-point matrices, MIC(0), where the nonzero pattern of the resulting factors is restricted to match that of the original matrix, is equivalent to the method proposed in 1968 by Dupont, Kendall, and Rachford [16]. The modification in the MIC(0) technique consists of, for every row, tracking the entries dropped in the factorization and adding their sum to the diagonal entry of that same row. This has the same effect as adding to the diagonal entry before computing the factorization, a process known as shifted incomplete LU (or shifted incomplete Cholesky) factorization [23, 27].

The result of the modified ILU process is that the product of the factors, $LU$, and the original matrix, $A$, give the exact same result when applied to a vector with constant entries. This rationale is derived from a continuity argument: the matrix, $LU$, that approximates $A$ should be exact on constants in the domain when $A$ corresponds to the discretization of an elliptic PDE. It can be observed that for matrices arising from such PDEs, MILU may be vastly superior to a standard ILU, and this improvement comes at virtually no extra cost. The method has been extensively analyzed and has given rise to a few variants; see, for example, [14, 15, 19, 29, 40]. However, most of the work on these so-called diagonal compensation techniques, to which MILU belongs, has been devoted to matrices arising from PDEs, as the analysis for the case of general sparse matrices is difficult. For the same reason, the use of these techniques for the threshold-based factorization, ILUT, has been avoided.

The focus of this paper is on combining modification strategies with threshold-based factorization, leading to new modified ILUT (MILUT) algorithms. Standard threshold-based incomplete LU (ILUT) factorization algorithms (both row-based and column-based) are reviewed in section 2. Section 3 presents three approaches that can be used to improve standard modification procedures. First, in section 3.1, we examine the question of relaxed compensation, where the modification process is tempered with the goal of improving the stability of the resulting LU factors. Here, we propose a strategy that aims to balance the accuracy of the MILUT factors with their stability. Second, section 3.2 extends this approach using complex-valued modifications, particularly in the context of indefinite operators; such approaches have been examined in many recent papers aimed at the numerical solution of the Helmholtz equation (see, for example, [3, 18, 30, 41]). Finally, section 3.3 examines the question of which vector, or vectors, should be used to guide the modification procedure, and how classical diagonal compensation strategies can be extended for matching multiple vectors. Here, a new "adaptive" modification scheme is proposed, where the modification vectors are chosen based on an Arnoldi iteration, rather than being prespecified as in typical MILU strategies; this is analogous to the family of adaptive algebraic multigrid methods that have been recently established in the literature [8, 9, 10, 11, 37]. Numerical results for these methods are given in section 4. Section 5 presents some concluding remarks.

Throughout the paper, superscript $H$ is used to denote the Hermitian transpose, as we consider matrices and vectors that may be complex valued. In the case of real arithmetic, superscript $T$ is used instead.

**2. ILU factorization with thresholds.** A common way to define a preconditioner is through an ILU factorization obtained from an approximate Gaussian elimination process. When Gaussian elimination is applied to a sparse matrix, $A$, a large number of nonzero elements in the factors, $L$ and $U$, may appear in locations

occupied by zero elements in $A$. These fill-ins often have small values and, therefore, they can be dropped to obtain a sparse approximate LU factorization, referred to as an incomplete LU (ILU) factorization. The simplest of these procedures, ILU(0) is obtained by performing the standard $LU$ factorization of $A$ and dropping all fill-in elements generated during the process. Thus, the factors, $L$ and $U$, have the same pattern as the lower and upper triangular parts of $A$ (respectively).

In the early work on ILU preconditioners, it was understood that ILU(0) could be ineffective and that more accurate factorizations could be needed. Such factorizations, denoted by ILU($k$) and IC($k$) (for incomplete Cholesky in the symmetric case), were initially derived by adopting a strategy to drop fill-ins according to their so-called levels-of-fill, first defined in the reservoir simulation literature [42]. Level-1 fill-ins, for example, are generated by products of level-zero fill-ins (at most); ILU(1), then, arises by keeping all fill-ins that have level zero or one and dropping any fill-in whose level is higher.

Because the level-of-fill concept was founded on properties of $M$-matrices, alternative techniques were soon developed for general sparse matrices. One of the first contributions along these lines is by Munksgaard [28], who defined an ILU factorization that uses a drop tolerance. Another method in the same class is ILU with threshold (ILUT) [34]. ILUT is a procedure based on a form of Gaussian elimination in which the rows of $L$ and $U$ are generated one by one. This row-wise algorithm is based on the so-called IKJ (or delayed update) Gaussian elimination process, whereby the $i$th step computes the $i$th rows of $L$ and $U$, see Algorithm 1.

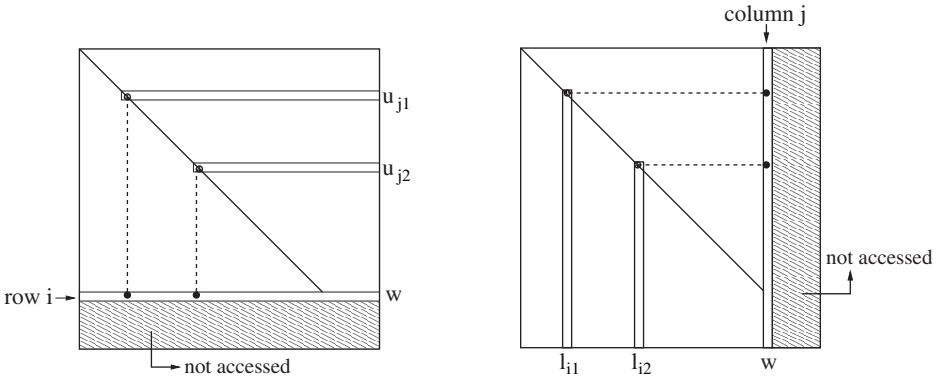
ALGORITHM 1. *IKJ-ordered Gaussian Elimination.*

0. For $i = 1 : n$, do:
1. $\quad \mathbf{w} = A_{i,1:n}$
2. $\quad$ For $k = 1 : i-1$, do:
3. $\quad\quad w_k := w_k / u_{k,k}$
4. $\quad\quad \mathbf{w}_{k+1:n} := \mathbf{w}_{k+1:n} - w_k \cdot U_{k,k+1:n}$
5. $\quad$ Enddo
6. $\quad$ For $j = 1, \ldots, i-1$, $l_{i,j} = w_j$ $(l_{i,i} = 1)$
7. $\quad$ For $j = i, \ldots, n$, $u_{i,j} = w_j$
8. Enddo

Here, and in all following discussion, $a_{i,k}$, $l_{i,k}$, and $u_{i,k}$ represent the scalar entries at the $i$th row and $k$th column of the matrices $A$, $L$, and $U$, respectively, $A_{i,1:n}$ denotes the complete $i$th row of $A$ (transposed as a column vector), while $A_{1:n,j}$ denotes the $j$th column of $A$, $\mathbf{w}_{k+1:n}$ denotes the last $n-k$ entries in the vector $\mathbf{w}$, $U_{k,k+1:n}$ denotes the last $n-k$ entries in the $k$th row of $U$ (transposed as a column vector), $L_{i,1:i-1}$ denotes the first $i-1$ entries in the $i$th row of $L$ (transposed as a column vector), and so forth. Of note in Algorithm 1 is that at the $i$th step, the $i$th row of $A$ is modified by previously computed rows of $U$, while the later rows of $A$ and $U$ are not accessed. The incomplete version of this algorithm is based on exploiting sparsity in the elimination and dropping small values according to a certain "dropping rule."

The dropping strategy in [34], which we follow in this paper, uses two parameters. The first parameter is a drop tolerance, $\tau$, which is used mainly to avoid doing an elimination if the pivot, $w_k$, is too small. The second parameter is an integer, $p$, which controls the number of entries that are kept in the $i$th rows of $L$ and $U$. Details can be found in [34, 36]. An illustration of the elimination process is shown in Figure 2.1(a), and a sketch of the general structure of the algorithm is given next as Algorithm 2.

As shown in Figure 2.1(a) and in Lines 3 and 4 of Algorithm 2, the pivot, $w_k$, is computed and compared with the dropping parameter, $\tau$, and dropped if it is smaller




(a) Row-based ILUT (b) Column-based ILUT

FIG. 2.1. *Illustration of the row-based (at left) and column-based (at right) ILUT algorithms.*

relative to some scaling parameter. Otherwise, operations of the form $\mathbf{w} = \mathbf{w} - w_k \mathbf{u}_k$ are performed to eliminate the entry associated with the pivot entry $w_k$. In lines 8 and 10, the threshold, $\tau$, is invoked again to drop small terms; then the largest $p$ entries in the resulting $i$th rows of $L$ and $U$ are kept, and the others are dropped.

ALGORITHM 2. *IKJ-ordered ILUT.*

0. For $i = 1 : n$, do:
1. $\mathbf{w} = A_{i,1:n}$
2. For $k = 1 : i-1$ and if $w_k \neq 0$, do:
3. $w_k = w_k / u_{k,k}$
4. Apply first dropping rule to $w_k$
5. If $w_k$ is not dropped, $\mathbf{w}_{k+1:n} = \mathbf{w}_{k+1:n} - w_k \cdot U_{k,k+1:n}$
6. Enddo
7. For $j = 1, \ldots, i-1$, $l_{i,j} = w_j$ $(l_{i,i} = 1)$
8. Apply second dropping rule to $L_{i,1:i-1}$
9. For $j = i, \ldots, n$, $u_{i,j} = w_j$
10. Apply second dropping rule to $U_{i,i+1:n}$
11. Enddo

The above algorithm is row-based; for column-oriented programming paradigms (when using compressed-sparse column formatting, such as within MATLAB), however, a column-based approach is more efficient. Furthermore, the triangular solves involving the $L$ and $U$ factors can be efficiently computed using a column-oriented data structure. For the column version of ILUT, at a given step $j$, the initial $j$th column of $A$, $\mathbf{a}_j$, is transformed by zeroing out entries above the diagonal element. As in the row version, operations of the form $\mathbf{w} := \mathbf{w} - w_k \mathbf{l}_k$ are performed to eliminate entries of $\mathbf{w}$ from top to bottom, until all entries strictly above the diagonal are zeroed out. In the ILU case, only a few of these eliminations are performed. An illustration is shown in Figure 2.1(b), and the complete algorithm is given in Algorithm 3. The elimination steps can be written in equation form as

$$\mathbf{a}_j - w_1 \mathbf{l}_1 - w_2 \mathbf{l}_2 \cdots - w_{j-1} \mathbf{l}_{j-1} = \hat{\mathbf{w}} + \epsilon_U + \epsilon_L, \tag{2.1}$$

where $\mathbf{l}_k$ is a column of $L$ with $k < j$ and $w_k$ is the coefficient used for the elimination of pivot $a_{kj}$. In order to avoid confusion in the notation, we introduce $\hat{\mathbf{w}}$ as the version of the transformed $\mathbf{w}$ with zero entries in positions $1, \ldots, j-1$. Furthermore,

for the sake of simplifying notation, we write (2.1) for dense columns, so the scalars $w_k$ are understood to be zero except for a few. The column $\epsilon_U$ contains the terms, $w_k$, which were dropped by the first dropping rule. The column $\epsilon_L$ contains the entries dropped by the post-dropping on $\mathbf{w}$, which is achieved by the second dropping rule. The resulting column, $\hat{\mathbf{w}}$, which now has zeros above position $j$, is divided by its diagonal entry and becomes the $j$th column of $L$, while the scalars $w_k$, representing the eliminated pivot $a_{kj}$, will constitute the $j$th column of $U$. Dropping in lines 3, 7, and 9 of Algorithm 3 may be handled in the same way as the row variant described in Algorithm 2.

ALGORITHM 3. *Left-looking or JKI ordered ILUT.*

0. For $j = 1 : n$, do:
1. $\mathbf{w} = A_{1:n,j}$
2. For $k = 1 : j-1$ and if $w_k \neq 0$, do:
3. Apply first dropping rule to $w_k$
4. If $w_k$ is not dropped, $\mathbf{w}_{k+1:n} = \mathbf{w}_{k+1:n} - w_k \cdot L_{k+1:n,k}$
5. Enddo
6. For $i = j+1, \ldots, n$, $l_{i,j} = w_i / w_j$ $(l_{j,j} = 1)$
7. Apply second dropping rule to $L_{j+1:n,j}$
8. For $i = 1, \ldots, j$, $u_{i,j} = w_i$
9. Apply second dropping rule to $U_{1:j-1,j}$
10. Enddo

**3. Modifying ILUT.** Two important considerations must be addressed when constructing an incomplete factorization (or, indeed, any other) preconditioner. Of primary importance from a theoretical point of view is the accuracy of the preconditioner. This is typically expressed in terms of the spectral equivalence of the preconditioner, $B$, to the system matrix, $A$, expressed by conditions such as

$$\alpha \mathbf{x}^T B \mathbf{x} \leq \mathbf{x}^T A \mathbf{x} \leq \beta \mathbf{x}^T B \mathbf{x} \quad \forall \mathbf{x}, \tag{3.1}$$

when $A$ and $B$ are both symmetric and positive definite. Here, the performance of the Krylov accelerator may be bounded in terms of the spectral equivalence bound, $\frac{\beta}{\alpha}$, and this bound may be sharp if the spectrum of $B^{-1}A$ is roughly evenly distributed between $\alpha$ and $\beta$. If, however, the spectrum is significantly clustered, this bound may be insufficient, since only an upper bound is guaranteed by the theory. On the other hand, of significant importance from a practical (or computational) point of view is the stability of the preconditioner. For incomplete factorization preconditioners, this was first observed by Elman [17], who defined the term and showed that disastrous situations can arise wherein the norm of $U^{-1}L^{-1}$ can be huge, even though $A$ is (relatively) well-behaved. Later, Bollhöfer [4] defined rigorous dropping strategies with a goal of specifically making the inverse factors, $L^{-1}$ and $U^{-1}$, not too large.

The argument behind these robust ILUs is based on writing the matrix, $A$, as a sum of the preconditioner plus an error term, $A = B + E$, and then considering the resulting preconditioned matrix, $B^{-1}A = I + B^{-1}E$. When an iterative method is applied to this preconditioned system, what matters is how well $I + B^{-1}E$ approximates the identity matrix. In exact arithmetic, all that matters is the spectral equivalence condition in (3.1); however, for practical computation, the boundedness in norm of $B^{-1}A$ (or, equivalently, of $B^{-1}E$) is also important. In many ways, the stability of a preconditioner is measured directly by $\|B^{-1}E\|$ in the $L^1$ or $L^\infty$ norm. For many practical cases, this norm can be much larger than the spectral radius of the preconditioned system, and this mismatch leads to significant differences between the

theoretical and actual performance of the preconditioned Krylov iteration. Indeed, ILUT is quite prone to this type of instability for indefinite problems; it is possible to construct examples where $\|B^{-1}\|_1$ is arbitrarily large for simple matrices, $A$.

Here, we aim to look at the role of modification of the triangular factors within threshold-based incomplete factorizations from the point of view of both stability and accuracy. In particular, we consider three questions:

1. Can we extend the classical modified ILU idea, based on diagonal compensation techniques, to safeguard the stability of the modified ILUT factorization?

2. For indefinite problems, can we use the ideas of complex-valued (and purely imaginary) perturbations to improve stability of the modified ILUT preconditioners?

3. Can we improve the modified ILUT schemes by adaptively choosing an appropriate vector (or set of vectors) to be used to satisfy the matching constraint? Furthermore, what modification condition(s) should be imposed when building the ILUT factorization?

Choosing a vector or set of vectors to match may be motivated primarily by accuracy considerations, which constrain the spectral properties of the preconditioners. Choosing the conditions to impose on the factorization, on the other hand, amounts to selecting coefficients so that $LU$ matches $A$ in some optimal way. In all cases, this is done primarily from the perspective of maintaining stability; a modification strategy that leads to small diagonal or large off-diagonal coefficients is expected to harm stability, while one that improves the relative conditioning of coefficients within a row or column will improve stability.

**3.1. Optimal spreading for modified ILUT.** In this subsection, we discuss an extension to the classical modified ILU idea by considering an optimal way of distributing the compensation term among the nonunit diagonal as well as the $L$ part of the factorization. Here and in section 3.2, we focus on the column version of ILUT given as Algorithm 3, while section 3.3 follows the row version of ILUT given as Algorithm 2.

Recall that from (2.1), $\hat{\mathbf{w}}$ is the vector that results from eliminating nonzeros in positions 1 through $j-1$ in the $j$th column of $A$, $\mathbf{a}_j$. In the modified version of ILUT, this column undergoes another modification before it is stored as a column of $L$. Specifically, we write the modification as the addition of a column $\mathbf{s}$, giving

$$\mathbf{a}_j - w_1\mathbf{l}_1 - w_2\mathbf{l}_2 \cdots - w_{j-1}\mathbf{l}_{j-1} - (\hat{\mathbf{w}} + \mathbf{s}) = \epsilon_U + \epsilon_L - \mathbf{s}. \tag{3.2}$$

We then obtain the diagonal entry of $U$ by setting $u_{j,j} = \hat{w}_j + s_j$ and the $j$th column of $L$ as $\mathbf{l}_j = (\hat{\mathbf{w}} + \mathbf{s})/w_j$.

Before discussing the choices of $\mathbf{s}$, we note an important practical consequence of the effect of *post-dropping* in the $U$ part of $\mathbf{w}$. By this, we mean dropping further entries among $w_1, \ldots, w_{j-1}$ after elimination step $j$ is complete (i.e., after line 5 of Algorithm 3). If, at this stage, any entry, $w_k$ for $1 \le k \le j-1$, is dropped (i.e., assigned a value of 0), then the resulting correction to (2.1) becomes more complicated, as it is not just $w_k$ but the column $w_k\mathbf{l}_k$ that must be accounted for in (3.2). This means $\epsilon = \epsilon_U + \epsilon_L$ will be modified by the addition of $w_k\mathbf{l}_k$, which can lead to significant additional fill-in elements. This suggests that it is reasonable to avoid post-dropping in $U$ when we base our analysis on (3.2) and, so, we adopt this strategy here. An analogous strategy is possible for the lower-triangular factor in the row-based algorithm; however, since we do not make use of (3.2) in section 3.3, we do not investigate this possibility for the row-based scheme.

Following (3.2), we consider the modified vector

$$\hat{\mathbf{w}} + \mathbf{s} = \begin{pmatrix} \mathbf{0} \\ \eta \\ \mathbf{l} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \sigma \\ \mathbf{z} \end{pmatrix}, \tag{3.3}$$

where, as discussed above, we choose $\mathbf{s}$ to affect only the lower-triangular part of the factorization. In (3.3), $\eta$ represents the diagonal entry of the column after elimination (but before scaling), $\sigma$ is a perturbation to this diagonal entry, the column vector $\mathbf{l}$ represents the strict lower part of the corresponding (unmodified) column in $L$, and $\mathbf{z}$ is the perturbation to $\mathbf{l}$.

In the classical modified ILU factorization, we simply take $\mathbf{z}$ to be the zero vector and ask that the inner product of the error vector, $\epsilon - \mathbf{s}$, and a given vector, $\mathbf{t}$, be zero. In the most common case, $\mathbf{t} = \mathbf{1} \equiv (1, 1, \ldots, 1)^T$, giving $\mathbf{1}^T(\epsilon - \sigma \mathbf{e}_j) = 0$ (where $\mathbf{e}_j$ is the $j$th column of the identity), or $\sigma = \mathbf{1}^T \epsilon$, i.e., the sum of the dropped entries. We refer to this approach as modified ILU with exact diagonal compensation, since the exact sum of the dropped entries is added back onto the diagonal. We shall denote this optimal (exact) compensation, which satisfies the matching constraint with respect to the vector $\mathbf{t}$, as $\sigma^*$. The more general scenario, where $\mathbf{t}$ is an arbitrary vector with no zero elements, leads to

$$\mathbf{t}^H(\epsilon - \sigma \mathbf{e}_j) = 0 \quad \rightarrow \quad \sigma^* = \frac{\mathbf{t}^H \epsilon}{\mathbf{t}^H \mathbf{e}_j}. \tag{3.4}$$

**3.1.1. Relaxed compensation.** As mentioned earlier, ILUT behaves quite differently from the usual ILU($k$) strategies and, in particular, exact diagonal compensation may affect the factors in a negative manner when the modified diagonal term, $\eta + \sigma$, is closer to zero than $\eta$, decreasing the diagonal dominance of the row or column under consideration.

To motivate our proposed strategy, we begin by equation (3.2). Consider the general situation where the "compensation column," $\mathbf{s}$, is any column which has at most the same sparsity pattern as $\hat{\mathbf{w}}$. Note that $\epsilon$ (the vector of dropped entries) and $\hat{\mathbf{w}}$ are structurally orthogonal and, therefore, so are $\mathbf{s}$ and $\epsilon$. Thus, $\|\epsilon - \mathbf{s}\|_2^2 = \|\epsilon\|_2^2 + \|\mathbf{s}\|_2^2$, which suggests that, from the point of view of accuracy, we should keep $\|\mathbf{s}\|_2$ small. Regarding (3.3), however, we would like to add a portion of the dropped entries to $\hat{\mathbf{w}}$ with the goal of making the scaled column of $L$ as "stable" as possible. This means that, from the point of view of stability, we want to modify the diagonal entry of $\hat{\mathbf{w}} + \mathbf{s}$ (i.e., the diagonal entry of $U$) and, possibly, the lower part as well so that (cf. (3.3))

(A) $|\eta + \sigma|$ is not small,

(B) $\|\mathbf{l} + \mathbf{z}\|_2$ is as small as possible.

Considering (A) alone, we must balance the contrasting requirements of accuracy and stability. Although it is desirable to make the modified diagonal entry $|\eta + \sigma|$ large relative to other entries, one should note that choosing $\sigma$ to be arbitrarily large would result in a factorization that poorly approximates the original matrix, $A$. Thus, it is necessary to control the size of $\sigma$. However, simply applying the exact compensation by choosing $\sigma = \sigma^*$ as prescribed in (3.4) can adversely affect stability by taking the diagonal term closer to zero when $\eta$ and $\sigma$ have opposite signs. Hence, considering (A), one option is to add a fraction, $\alpha > 0$, of $\sigma^*$ so that $\sigma = \alpha \sigma^*$.

Considering (B), we aim to choose $\mathbf{z}$, a sparse column with the same sparsity pattern as $\mathbf{l}$, to simultaneously minimize $\|\mathbf{z}\|_2$ (the additional discarded fill-in) and $\|\mathbf{l} + \mathbf{z}\|_2$, relative to $\eta + \sigma$ (the resulting column of $L$). However, it is difficult to solve

this optimization problem, particularly where both unknowns ($\sigma$ and $\mathbf{z}$) are considered at the same time. Thus, we formulate an optimization strategy that handles conditions (A) and (B) above with $\mathbf{z}$ as the only unknown; subsequently, we exploit the solution of this optimization problem as a guide for choosing $\sigma$.

So, fixing $\sigma$, we pose the optimization problem of

$$\min_{\mathbf{z}} \quad \frac{\|\mathbf{l}+\mathbf{z}\|_2^2}{|\eta+\sigma|^2} \quad \text{subject to the constraint} \quad |\sigma|^2+\|\mathbf{z}\|_2^2 \le \gamma^2,$$

where the constraint, $|\sigma|^2+\|\mathbf{z}\|_2^2 \le \gamma^2$ (for $\gamma > 0$), is used to safeguard the accuracy of the factorization as an approximation to the original matrix, $A$. We introduce the penalty term, $\mu \ge 0$, and solve instead the penalized problem,

$$\min_{\mathbf{z}} \left( \frac{1}{|\eta+\sigma|^2}(\mathbf{l}^H\mathbf{l}+2\mathbf{z}^H\mathbf{l}+\mathbf{z}^H\mathbf{z}) + \mu\left(|\sigma|^2+\mathbf{z}^H\mathbf{z}-\gamma^2\right)\right),$$

whose minimum is reached when

$$\mathbf{z} = \frac{-1}{(1+\mu|\eta+\sigma|^2)}\,\mathbf{l}. \tag{3.5}$$

From the KKT conditions, complementary slackness implies that $\mu(|\sigma|^2+\mathbf{z}^H\mathbf{z}-\gamma^2)=0$. Recall that $\mu \ge 0$ and notice that if $\mu = 0$, then the constraint is inactive. Hence, for $\mu > 0$, we have

$$\mathbf{z}^H\mathbf{z} = \gamma^2 - |\sigma|^2,$$

which gives (by substitution of $\mathbf{z}$ from (3.5))

$$|\eta+\sigma|^4\mu^2 + 2|\eta+\sigma|^2\mu + \left(1-\frac{\mathbf{l}^H\mathbf{l}}{\gamma^2-|\sigma|^2}\right) = 0.$$

We can then solve for $\mu$ as

$$\mu = \frac{1}{|\eta+\sigma|^2}\left(-1+\sqrt{\frac{\mathbf{l}^H\mathbf{l}}{\gamma^2-|\sigma|^2}}\right). \tag{3.6}$$

To obtain a valid solution for $\mu$, two conditions need to be satisfied. First, we need $\mu > 0$, implying that

$$\frac{\mathbf{l}^H\mathbf{l}}{\gamma^2-|\sigma|^2} > 1 \quad \Rightarrow \quad |\sigma|^2 > \gamma^2 - \mathbf{l}^H\mathbf{l}.$$

Notice that since $|\sigma|^2 > 0$ and $\gamma^2$ may be less than $\mathbf{l}^H\mathbf{l}$, we can express the above inequality, without loss of generality, as

$$|\sigma| > \sqrt{\max(\gamma^2 - \mathbf{l}^H\mathbf{l}, 0)},$$

where $\max(a,b)$ simply returns the maximum of $a$ and $b$. While this gives a lower bound for $\sigma$, we also need to ensure that $\sigma$ is not too large, so that $\mu$ remains real-valued. We further require that

$$\frac{\mathbf{l}^H\mathbf{l}}{\gamma^2-|\sigma|^2} > 0 \quad \Rightarrow \quad |\sigma| < \gamma.$$

Putting these two inequalities together, we get that

$$\sqrt{\max(\gamma^2 - \mathbf{l}^H\mathbf{l}, 0)} < |\sigma| < \gamma.$$

In real arithmetic, this reduces to

$$-\gamma < \sigma < -\sqrt{\max(\gamma^2 - \mathbf{l}^T\mathbf{l}, 0)} \quad \text{or} \quad \sqrt{\max(\gamma^2 - \mathbf{l}^T\mathbf{l}, 0)} < \sigma < \gamma. \tag{3.7}$$

**3.1.2. Choice of $\boldsymbol{\sigma}$.** The parameter $\gamma$ controls the size of the modifications, $\|\mathbf{s}\|_2^2 = |\sigma|^2 + \|\mathbf{z}\|_2^2$, and hence must be carefully chosen. Choosing $\gamma = |\sigma^*|$, the exact diagonal compensation factor from (3.4) naturally lends itself to the modified ILUT scheme, since it guarantees that $\sigma$ will be a fraction of the weighted sum of the dropped terms. For the special case of $\sigma^* = 0$ (when there are no dropped terms or, more generally, when $\mathbf{t}^H\epsilon = 0$), the choice of $\gamma > 0$ can be based on a default value, such as the drop tolerance, $\tau$, used in ILUT. When $\gamma > 0$, the sign of $\sigma$ is chosen to match that of the exact diagonal compensation factor. Thus, in real arithmetic, if $\sigma^* < 0$, then $\sigma$ is chosen to satisfy $-\gamma < \sigma < -\sqrt{\max(\gamma^2 - \mathbf{l}^T\mathbf{l}, 0)}$; otherwise, $\sqrt{\max(\gamma^2 - \mathbf{l}^T\mathbf{l}, 0)} < \sigma < \gamma$.

It remains to pick the size of $\sigma$ within the interval $(\sqrt{\max(\gamma^2 - \mathbf{l}^H\mathbf{l}, 0)}, \gamma)$, given that the optimization problem provides no further guidance. We propose a criterion based on choosing $\sigma$ within its allowable interval in order to address the stability of the resulting factors. Thus, our choice is guided by the size and sign of the resulting perturbations. For simplicity, we assume real arithmetic in our analysis and provide a generalization to complex arithmetic afterward.

First note that when $\mathbf{l} = \mathbf{0}$, the optimization problem clearly attains a minimum with $\mathbf{z} = \mathbf{0}$. Furthermore, extending the inequalities in (3.7) gives $|\sigma| = \gamma$. Thus, we must consider the choice of $\sigma$ only when $\mathbf{l} \neq \mathbf{0}$. In this case, we have a nontrivial range of possible values of $\sigma$ and must choose whether $|\sigma|$ should be closer to $\gamma$ or $\sqrt{\max(\gamma^2 - \mathbf{l}^T\mathbf{l}, 0)}$. To make this decision, we return to criterion (A), which states that $|\eta + \sigma|$ should not be "small." Thus, if $\eta$ and $\sigma$ are of different signs, it is sensible to select $|\sigma|$ to be close to the lower bound $\sqrt{\max(\gamma^2 - \mathbf{l}^T\mathbf{l}, 0)}$, so that $|\eta + \sigma|$ is not made smaller than necessary. Assuming this choice of $\sigma$ is small enough, then from (3.6) we see that $\mu$ is not too large (since $\gamma^2 - \sigma^2$ is larger) and, hence, $\mathbf{z}$ is a significant modification on $\mathbf{l}$. In other words, adding $\sigma$ to the diagonal improves the matching condition, whereas $\mathbf{z}$ serves to stabilize the column (balancing out the negative effect of $\sigma$). However, if $\eta$ and $\sigma$ are of the same sign, then it makes sense to choose $|\sigma|$ to be close to the upper bound, $\gamma$, to benefit from both improving the matching condition as well as stabilizing the column. Notice that this choice of $|\sigma|$ increases the magnitude of $\mu$ and hence decreases the norm of $\mathbf{z}$.

The choice for $\sigma$ is then formally defined as

$$\sigma = \begin{cases} 0 & \text{if} \quad \mathbf{l}^T\mathbf{l} = 0, \\ sgn(\sigma^*) \times (\sqrt{\max(\gamma^2 - \mathbf{l}^T\mathbf{l}, 0)} + \beta \times \rho) & \text{if} \quad sgn(\eta) = sgn(\sigma^*), \\ sgn(\sigma^*) \times (\sqrt{\max(\gamma^2 - \mathbf{l}^T\mathbf{l}, 0)} + \varepsilon \times \rho) & \text{otherwise}, \end{cases} \tag{3.8}$$

where $sgn(.)$ is the signum function, $\rho = \gamma - \sqrt{\max(\gamma^2 - \mathbf{l}^T\mathbf{l}, 0)}$ is the width of the interval for choosing $\sigma$, $\varepsilon \in (0, 1)$ is small, and $\beta \in (0, 1)$ is not small. For general sparse matrices, we propose setting $\varepsilon$ to be equal to the drop tolerance, $\tau$, of the ILUT factorization and $\beta$ to be equal to the "diagonal dominance ratio" of the column, given by $\beta = |a_{j,j}|/\|\mathbf{a}_j\|_1$. If the $j$th column of the original matrix, $A$,

is diagonally dominant, then $\beta$ is closer to one, making $\sigma$ large. As a result, $\mu$ is also large, which makes the modification with $\mathbf{z}$ small. Since the column is already diagonally dominant, the modification with $\mathbf{z}$ need not be significant. On the other hand, if the $j$th column of $A$ is not diagonally dominant, then $\beta$ is small, and the resulting modification with $\mathbf{z}$ is not too small.

For matrices with complex coefficients, (3.8) takes the more general form

$$\sigma = \begin{cases} 0 & \text{if } \mathbf{l}^H\mathbf{l} = 0, \\ sgn(\sigma^*) \times (\sqrt{\max(\gamma^2 - \mathbf{l}^H\mathbf{l}, 0)} + \beta \times \rho) & \text{if } \begin{array}{l} sgn(\mathrm{Re}(\eta)) = sgn(\mathrm{Re}(\sigma^*)) \text{ and} \\ sgn(\mathrm{Im}(\eta)) = sgn(\mathrm{Im}(\sigma^*)), \end{array} \\ sgn(\sigma^*) \times (\sqrt{\max(\gamma^2 - \mathbf{l}^H\mathbf{l}, 0)} + \varepsilon \times \rho) & \text{otherwise.} \end{cases}$$

The main difference between this form and (3.8) is that the sign comparison between $\eta$ and $\sigma^*$ takes into account both the real and imaginary parts of the complex term. Also, note that since the $sgn(\sigma^*) = \sigma^*/|\sigma^*|$ is complex, $\sigma$ will take on a complex value as well.

**3.2. Complex modification.** While the above modification strategy offers an improvement on unmodified ILUT, even for some very poorly conditioned systems (see section 4.1), we still observe poor performance when the original system matrix is very indefinite. This is not altogether surprising, as indefinite linear systems generally require a more accurate factorization or complicated preconditioning approach, such as those in [3, 18, 41]. However, simply improving the accuracy of the factorization (by decreasing the drop tolerance) can yield factors that are even more unstable than the original system matrix, making the factorization ineffective as a preconditioner. Diagonal compensation techniques have been proposed to handle this issue [3, 23, 30]. However, real-valued perturbations to the diagonal have been found to be not very effective, and most of the successful methods have relied on the use of complex (or imaginary) perturbations instead.

Previous work, such as in [18, 23, 30, 41], has shown that purely imaginary shifts of symmetric and indefinite problems have the effect of clustering eigenvalues on a circle in the right half-plane, with an accumulation point near one. This results in a more stable factorization, which leads to a more effective ILU-based preconditioner. The approach in these papers is based on shifting the diagonal entries by an appropriate perturbation prior to performing the ILU factorization (shifted ILU). This is somewhat different from the classical modified ILU approach, which is based on matching the effect of the preconditioning matrix and the original matrix on some vector. Nonetheless, ideas from the former approach can be incorporated into the modified ILU scheme.

If a uniform perturbation $\sigma$ is used to modify the diagonal in the modified ILUT scheme and is known prior to the factorization, then the resulting factorization has the same effect as the shifted ILU scheme. Thus, to motivate the use of a complex $\sigma$, we consider the following simplified analysis. Let $A$ be a Hermitian matrix,

$$B = A + \sigma I_n,$$

$\sigma = \nu + i\theta$ for some real numbers $\nu, \theta \geq 0$. Suppose that we use the exact LU factorization of $B$ as the preconditioner, $M = LU(B)$. Then, the eigenvalues of the preconditioned matrix $M^{-1}A$ satisfy

$$\mu_j = \frac{\lambda_j}{(\lambda_j + \nu) + i\theta} = \frac{\lambda_j(\lambda_j + \nu)}{(\lambda_j + \nu)^2 + \theta^2} - i\frac{\lambda_j\theta}{(\lambda_j + \nu)^2 + \theta^2}, \tag{3.9}$$

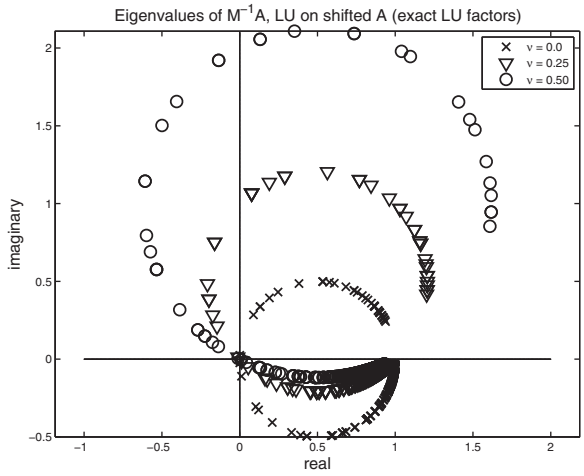



FIG. 3.1. *Spectrum of the preconditioned matrix $M^{-1}A$, where $M = LU(B)$ for $B = A + \sigma I_n$ and $\sigma = \nu + 0.25i$.*

where the $\lambda_j$ are the eigenvalues of $A$, and the $\mu_j$ are the eigenvalues of the preconditioned matrix, $M^{-1}A$. From the above equation and the identity $\lambda_j = \frac{1}{2}(\lambda_j + \nu + i\theta + \lambda_j - \nu - i\theta)$, we obtain

$$\left|\mu_j - \frac{1}{2}\right| = \frac{1}{2}\frac{|(\lambda_j - \nu) - i\theta)|}{|(\lambda_j + \nu) + i\theta)|}. \tag{3.10}$$

We observe from (3.10) that for $\nu = 0$, all eigenvalues $\mu_j$ of the preconditioned system lie on the circle centered at $1/2 + 0i$ with radius $1/2$. This, however, is not true for $\nu > 0$, where a (real) positive $\lambda_j$ will be mapped to a complex number within a distance of $1/2$ from the center of this circle, whereas a negative $\lambda_j$ will be mapped to a complex number beyond a distance of $1/2$ from the center. Figure 3.1 shows the spectrum of the preconditioned matrix, $M^{-1}A$, using the exact LU factors of $B = A + \sigma I_n$, where $A$ is obtained from shifting the matrix for the finite-difference discretization of the Laplace operator, $-\Delta$, on a $25 \times 25$ grid. The discretization assumes a scaling based on the mesh size parameter, $h$, so that the resulting matrix initially has 4 on the diagonal and four off-diagonal entries of $-1$ each. The matrix is then shifted by adding a negative shift of $-1.0$ to the diagonal, to make it indefinite. Note that (because of the scaling by $h^2$) this shift is quite severe and makes it a challenge to iterative methods to solve a linear system involving this matrix. The resulting matrix has size $n = 625$, with 49 negative eigenvalues (smallest is $\lambda_1 = -0.9708$ and largest is $\lambda_{625} = 6.9708$). The figure shows plots of the preconditioned spectrum with $\sigma = \nu + 0.25i$, for different values of $\nu$. We observe a better clustering of the eigenvalues for smaller values of $\nu$. Note, however, that the resulting preconditioner based on $A + \sigma I_n$ will be both indefinite and non-Hermitian; hence, MINRES is no longer usable as the Krylov solver, and GMRES or BiCGStab must be used instead.

**3.2.1. Imaginary perturbations for modified ILUT.** While the shifted ILU (and also shifted multigrid) approaches have been shown to improve on unshifted preconditioners for indefinite, Helmholtz-type problems, modified approaches based on similar reasoning offer better control of the perturbation and so more accurate preconditioners for these problems. In what follows, we present an approach for

using imaginary perturbations similar to the compensation strategy described above to improve the quality of the modified ILUT factorization. To do so, the above optimization problem is solved replacing $\sigma$ by $i\sigma$ (where $\sigma \in \mathbb{R}$). In other words, we rewrite the problem as

$$\min_{\mathbf{z}} \left( \frac{1}{|\eta + i\sigma|^2} (\mathbf{l}^H \mathbf{l} + 2\mathbf{z}^H \mathbf{l} + \mathbf{z}^H \mathbf{z}) + \mu( \, |\sigma|^2 + \mathbf{z}^H \mathbf{z} - \gamma^2) \right).$$

Following the same approach as before yields

$$\mathbf{z} = \frac{-1}{(1 + \mu|\eta + i\sigma|^2)} \, \mathbf{l}.$$

Note that the penalty term, $\mu$, remains essentially the same as in (3.6), giving the optimal ranges for $\sigma$ as

$$-\gamma < \sigma < -\sqrt{\max(\gamma^2 - \mathbf{l}^H \mathbf{l}, 0)} \quad \text{or} \quad \sqrt{\max(\gamma^2 - \mathbf{l}^H \mathbf{l}, 0)} < \sigma < \gamma.$$

In this approach, the sign of $\sigma$ is chosen to match the sign of the imaginary part of the diagonal term, $\eta$ (if $\eta$ is complex; if $\eta$ is real, the sign can be chosen arbitrarily). This improves $|\eta + i\sigma|$, making the factorization more diagonally dominant. We then need to decide whether to take $\sigma$ to be close to its lower or upper bound within this interval. As before, choosing $|\sigma|$ to be close to the lower bound leads to smaller values of $\mu$ and, thus, a larger correction in $\mathbf{z}$. Choosing a larger $|\sigma|$ (close to the upper bound), on the other hand, gives a smaller correction in $\mathbf{z}$. This latter option is more appealing, since the increase in the diagonal entries from the imaginary perturbation is offset by the more accurate (less perturbed) modification of the rest of $\hat{\mathbf{w}}$. This gives

$$\sigma = s \times \left( \sqrt{\max(\gamma^2 - \mathbf{l}^H \mathbf{l}, 0)} + \beta \times \rho \right),$$

where $\rho$ is the size of the interval as before, $\beta$ is close to 1 (e.g., $\beta = 1 - \tau$, where $\tau$ is the ILUT drop tolerance), and the sign $s$ is defined as

$$s = \begin{cases} 1 & \text{if} \quad \text{Im}(\eta) > 0, \\ -1 & \text{otherwise.} \end{cases}$$

**3.2.2. The modified ILUT algorithm with relaxed compensation.** Algorithm 4 formally describes the column version of the modified ILUT scheme discussed above.

ALGORITHM 4. *Left-looking or JKI ordered MILUT.*

0. Select a vector, $\mathbf{t}$, for matching
1. For $j = 1 : n$, do:
2. $\quad \mathbf{w} = A_{1:n,j}$
3. $\quad$ Initialize $\gamma = 0$
4. $\quad$ For $k = 1 : j - 1$ and if $w_k \neq 0$, do:
5. $\quad\quad$ Apply first dropping rule to $w_k$
6. $\quad\quad$ If $w_k$ is not dropped, $\mathbf{w}_{k+1:n} = \mathbf{w}_{k+1:n} - w_k \cdot L_{k+1:n,k}$
7. $\quad\quad$ Else, $\gamma = \gamma + w_k \cdot t_j$
8. $\quad$ Enddo
9. $\quad$ Apply second dropping rule to $\mathbf{w}_{j+1:n}$

10. For $k = j+1, \ldots, n$, if $\mathbf{w}_k$ is dropped, update $\gamma = \gamma + w_k \cdot t_j$
11. Do relaxed compensation on $\mathbf{w}_j$ and $\mathbf{w}_{j+1:n}$ with parameter $\gamma$
12. For $i = j+1, \ldots, n$, $l_{i,j} = w_i/w_j$ $(l_{j,j} = 1)$
13. For $i = 1, \ldots, j$, $u_{i,j} = w_i$
14. Enddo

The above algorithm extends Algorithm 3 by tracking the dropped terms to satisfy the matching condition. At line 9 of the above algorithm, we prune the $L$ part of $\mathbf{w}$ by applying the second dropping rule prior to performing the modification. This is necessary because we need to scale the resulting (modified) column of $L$ by the (modified) diagonal. Furthermore, notice that we avoid post-dropping in $U$. The only dropping in $U$ is handled by the first dropping rule (at line 5 of the algorithm).

Note that the updates to the size parameter, $\gamma$, accumulate terms of the form $w_k t_j$ and not, as is usually done in a row-based calculation, $w_k t_k$. Although we calculate the ILU factorization in a columnwise manner, the matching condition is always a row-wise condition, that $(A\mathbf{t})_i = (LU\mathbf{t})_i$. Accumulating $\gamma$, however, is a columnwise calculation. Thus, we use row-wise values of $w_k$, the dropped fill-in entries, but fix $t_j$ based on the column that we are considering.

**3.3. Row-based modification.** Finally, we turn our attention to the choice of $\mathbf{t}$ and a generalization where multiple vectors are chosen. Because of the natural disconnect between row-based matching criteria and column-based factorizations, this subsection follows a row-based ILUT approach, as given in Algorithm 2.

**3.3.1. Choice of vectors.** The original modified incomplete Cholesky algorithm [16, 19] may be seen as a modification of the standard incomplete Cholesky factorization by adjusting the diagonal coefficient to match the constant vector. For the discretizations of elliptic PDEs considered there, the constant vector is a reasonable choice from an accuracy point of view as it yields small Rayleigh quotients for these matrices and, as such, can easily lead to large spectral equivalence ratios, $\frac{\beta}{\alpha}$, if $\mathbf{1}^T B\mathbf{1}$ is much larger than $\mathbf{1}^T A\mathbf{1}$. If $A$ is a symmetric and positive-definite M-matrix, then the theory of modified incomplete factorizations may be extended based on M-matrix properties and matching (or nearly matching) a given positive vector (such as the eigenvector of the M-matrix, $A$, corresponding to its smallest eigenvalue) [2].

The attraction of modifying the preconditioner $B$ to match the action of $A$ on a space associated with the eigenvectors corresponding to its small eigenvalues is a natural one from the perspective of accuracy. Consider the spectral equivalence bound,

$$\alpha \mathbf{x}^T B\mathbf{x} \leq \mathbf{x}^T A\mathbf{x} \leq \beta \mathbf{x}^T B\mathbf{x},$$

for a unit-length vector $\mathbf{x}$ that yields a (relatively) small Rayleigh quotient $\frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T\mathbf{x}} = \mathbf{x}^T A\mathbf{x} \approx \lambda_{\min}(A)$. If the action of $B$ on $\mathbf{x}$ is far from that of $A$, it may lead to either a very small lower equivalence bound $\alpha$ (if $\mathbf{x}^T B\mathbf{x} \gg \mathbf{x}^T A\mathbf{x}$) or a large upper equivalence bound $\beta$. Because $\alpha$ and $\beta$ are relative quantities (the extrema of the generalized Rayleigh quotient $\frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T B\mathbf{x}}$), they are strongly influenced by inaccuracies when the numerator is either very small or very large. In particular, a fixed (absolute) error in $\mathbf{x}^T(B-A)\mathbf{x}$ will most strongly affect the bound for small values of the numerator; thus, the classical modifications applied for M-matrices make intuitive sense from the accuracy viewpoint.

Motivated by accuracy, then, the question of which vector (or set of vectors) to choose for the modification is answered by considering the spectral properties of $A$. In the case where $A$ comes from the discretization of a differential equation, these vectors may be chosen based on the known properties of the differential operator.

For second-order scalar elliptic differential equations, for example, the constant and linear functions are in the null-space of the dominant differential operator and so are good candidates for vectors to be matched. For the equations of linear elasticity, the so-called rigid body modes of translation and rotation in two or three dimensions would make good candidates for modification (just as they make good candidates on which to base the prolongation operator within smoothed-aggregation type multigrid methods [39]).

If the origins of the matrix $A$ are not as well known, due to either an unknown discretization process or incomplete knowledge of the original problem, then choosing such vectors a priori may be difficult or impossible. Instead, an estimate of the extreme eigenvectors of $A$ may be made to use in constraining the preconditioner (similar to the use of prototypical error vectors in adaptive multigrid algorithms [9, 10, 11]). Using a limited number of steps of the Arnoldi algorithm, for example, can give good approximations to the extremal eigenvalues of $A$, along with the associated Ritz vectors. Analogously to the classical modified incomplete factorization approach, an estimate of the smallest eigenvalue of a symmetric and positive-definite matrix, $A$, can be made and the preconditioner modified to match the associated approximate eigenvector. More generally, however, for a fixed (and small) Krylov subspace, estimates of the eigenvalues of $A$ closest to zero may be obtained, along with the corresponding approximate eigenvectors; these (approximate) eigenvectors can be used in the modification.

**3.3.2. Choice of coefficients.** To apply modification to Algorithm 2 after each row of $L$ and $U$ is computed (including all dropping steps), the computed coefficients must be modified to match the chosen vector or vectors. When only a single vector, $\mathbf{x}$, is chosen to be matched, an obvious choice is to modify the nonunit diagonal (of $U$) so that $LU\mathbf{x} = A\mathbf{x}$ as in the column-based approach discussed above, so long as this does not make the diagonal entry unduly small. As above, modifying more coefficients may, however, lead to better stability. When several vectors have been chosen to be matched, however, only a poor overall matching is possible when modifying only a single coefficient. In this case, modifying several coefficients allows a better (and possibly perfect) match of a set of vectors and may also be used to improve stability.

The choice of which coefficients to modify, however, may greatly affect both the accuracy and stability of the factorization. If the chosen set forces some large coefficients to become small, or small off-diagonal coefficients in $L$ and $U$ to become large, then the accuracy of the overall preconditioner may suffer. If, in particular, the nonunit diagonal entry in row $i$ of $U$ is forced to become quite small in comparison to the off-diagonals in row $i$, then stability of the computation will be negatively affected, as discussed above. A simpler strategy than that proposed above for choosing which coefficients to modify is to choose a subset of the largest coefficients of the $L$ and $U$ factors. If the misfit for each vector $\mathbf{x}$ to be matched is relatively small ($((A-LU)\mathbf{x})_i$ is small compared to $x_j$ for $j$ such that $l_{i,j}$ and $u_{i,j}$ will be modified), then small modifications (relative to the size of $l_{i,j}$ and $u_{i,j}$) will be sufficient to match the vectors. Thus, the stability of the factorization will not be negatively affected by the modifications, and the accuracy is not expected to suffer significantly either. Alternately, modification could be used on indices where entries in $L$ and $U$ are expected to be large based on the pattern of the matrix $A$. For example, modification of $l_{i,j}$ or $u_{i,j}$ if $a_{i,j} \neq 0$ or if $|j-i|$ is not too large.

Let $n_v$ be the number of (linearly independent) vectors to be matched and $n_c$ be the number of coefficients to be modified. Given $\mathbf{l}_{i,\star}$ and $\mathbf{u}_{i,\star}$, the $i$th rows of $L$

and $U$ computed within ILUT, we look to modify these rows so that the resulting factorization satisfies $(\tilde{L}\tilde{U}\mathbf{x}^{(k)})_i = (A\mathbf{x}^{(k)})_i$ for $k = 1, \ldots, n_v$. Writing $\tilde{l}_{i,j} = l_{i,j} + \delta_{i,j}$ and $\tilde{u}_{i,j} = u_{i,j} + \delta_{i,j}$, the matching condition for vector $\mathbf{x}^{(k)}$ can be rewritten as

$$\sum_{j<i}(l_{i,j} + \delta_{i,j})(U\mathbf{x}^{(k)})_j + \sum_{j\geq i}(u_{i,j} + \delta_{i,j})x_j^{(k)} = (A\mathbf{x}^{(k)})_i$$

$$\text{or } \sum_{j<i}(U\mathbf{x}^{(k)})_j\delta_{i,j} + \sum_{j\geq i}x_j^{(k)}\delta_{i,j} = ((A - LU)\mathbf{x}^{(k)})_i.$$

So, the updates to the coefficients in row $i$ of $L$ and $U$ can be found by solving the linear system $X\delta = \mathbf{r}$, where each row in the system matrix $X$ corresponds to a given vector $\mathbf{x}^{(k)}$ with the first $i-1$ columns taking values of $(U\mathbf{x}^{(k)})_j$ and the last $n - i + 1$ columns taking values of $x_j^{(k)}$, and each entry in the residual $\mathbf{r}$ given by the misfit $r_k = ((A - LU)\mathbf{x}^{(k)})_i$. As only a subset of the entries in row $i$ of $L$ and $U$ will be considered for modification, $X$ can be restricted from $n$ columns to $n_c$ columns, corresponding only to those weights selected for modification. As the rank of $X$ is at most $n_v$, it is possible for this system to be overdetermined, have a unique solution, or be underdetermined. In the overdetermined case, a least-squares solution is considered, minimizing $\|X\delta - \mathbf{r}\|$. In the underdetermined case, the solution which results in the smallest changes, $\delta$, to the coefficients in $L$ and $U$ is chosen.

**3.3.3. The generalized MILUT algorithm.** Algorithm 5 gives the overall algorithm for computing the modified ILUT factorization of a given matrix $A$ using the scheme proposed in this section.

ALGORITHM 5. *IKJ-ordered MILUT algorithm.*

0. Select $n_v$ vectors for matching, $\mathbf{x}^{(k)}$, $k = 1, \ldots, n_v$
1. For $i = 1, \ldots, n$, do
2. $\quad \mathbf{w} = A_{i,1:n}$
3. $\quad$ For $k = 1, \ldots, i-1$ and if $w_k \neq 0$,
4. $\quad\quad w_k = w_k/u_{kk}$
5. $\quad\quad$ Apply a dropping rule to $w_k$
6. $\quad\quad$ If $w_k$ is not dropped, $\mathbf{w}_{k+1:n} = \mathbf{w}_{k+1:n} - w_k \cdot U_{k,k+1:n}$
7. $\quad$ Enddo
8. $\quad$ For $k = i+1, \ldots, n$, apply a dropping rule to $w_k$
9. $\quad$ For $j = 1, \ldots, i-1$, $l_{i,j} = w_j$
10. $\quad$ Apply a dropping rule to $\mathbf{l}_{i,\star}$
11. $\quad$ For $j = i, \ldots, n$, $u_{i,j} = w_j$
12. $\quad$ Apply a dropping rule to $\mathbf{u}_{i,\star}$
13. $\quad$ For $k = 1, \ldots, n_v$, compute $r_k = ((A - LU)\mathbf{x}^{(k)})_i$
14. $\quad$ For $k = 1, \ldots, n_v$, do
15. $\quad\quad$ For $m_j < i$, $X_{k,m_j} = (U\mathbf{x}^{(k)})_{m_j}$
16. $\quad\quad$ For $m_j \geq i$, $X_{k,m_j} = x_{m_j}^{(k)}$
17. $\quad$ Enddo
18. $\quad$ Solve $X\delta = \mathbf{r}$
19. $\quad$ For $m_j < i$, $l_{i,m_j} = l_{i,m_j} + \delta_{i,j}$
20. $\quad$ For $m_j \geq i$, $u_{i,m_j} = u_{i,m_j} + \delta_{i,j}$
21. Enddo

The added computational cost for the modification algorithm need not be significant. Considering only the costs within the factorization, added steps include the computation of the residuals $r_k$, computation of the entries in $X$, solving the

system for $\delta$, and updating the weights. Computing the residuals and matrix $X$ may be accelerated by computing the products $A\mathbf{x}^{(k)}$ in line 0 when the vectors $\mathbf{x}^{(k)}$ are chosen and by computing (and storing) the partial vectors $U\mathbf{x}^{(k)}$ as each row in $U$ is finalized. Note that no extra storage is needed to store the partial products $U\mathbf{x}^{(k)}$ as the entries $x_j^{(k)}$ for $j < i$ are not needed at stage $i$ of the algorithm and so may be overwritten. Thus, matrix $X$ may be assembled at no cost (other than $n_c \times n_v$ memory accesses). To compute the residuals, the partial matrix-vector product $\sum_{j<i} l_{i,j}(U\mathbf{x}^{(k)})_j + \sum_{j\geq i} u_{i,j}x_j^{(k)}$ must be evaluated. However, in the ILUT algorithm, dropping in lines 10 and 12 is typically done so that each row of $L$ and $U$ is allowed no more than a fixed factor times the average number of nonzeros in each row of $A$. Thus, the total cost of all these computations is no more than that factor times the total number of nonzeros in each row of $A$. Solving the system at line 18 may be efficiently done using either $LQ$ or $QR$ factorization of $X$.

**4. Numerical results.** In what follows, we present some numerical results of the modified ILUT schemes described in the previous section. These results are obtained from applications governed by the standard two-dimensional (2D) finite-difference Laplacian, some shifted and scaled problems, and the Helmholtz equation.

**4.1. Results for MILUT with relaxed compensation.** We present some numerical results for the relaxed compensation strategy for MILUT, obtained from the solution to the minimization problem described in section 3.1.1. The tests shown here are run with the column version of the MILUT algorithm, programmed in C, on a dual-core AMD Opteron 1 GHz machine with 4 GB of RAM.

In the first set of examples, we test the relaxed compensation strategy on systems that are symmetric and positive definite, but poorly conditioned. The base problem is a 2D finite-difference discretization of the Laplace operator $-\Delta$ on a uniform grid using centered differences. As before, the discretization assumes a scaling based on the mesh size parameter $h$ so that the resulting matrix initially has 4 on the diagonal, and four off-diagonal entries of $-1$. This is then shifted by a small negative term $\varrho$ to make it indefinite. We then construct the normal matrix $A = V^TV$ from the resulting indefinite matrix $V$. Note that although $A$ is now symmetric and positive definite, solving a linear system involving $A$ can still be quite challenging, due to its large condition number. Furthermore, shifting $V$ to make it indefinite can result in a cluster of eigenvalues close to zero for the normal matrix $A$, which makes it a more challenging problem for solution by Krylov methods. The right-hand side vectors $\mathbf{b}$ are constructed by choosing the entries of the solution vector $\mathbf{x}$ from the uniform distribution on $[0, 1]$ and computing $\mathbf{b} = A\mathbf{x}$. Since these vectors are chosen randomly, the results displayed in the tables below are obtained from averaging several runs of the same problem for each test case. The condition numbers of the resulting matrices range from $3 \times 10^8$ to $10^{11}$ but do not correlate strongly with problem size or shift, due to the formation of the normal equations.

In what follows, we compare the performance of the relaxed compensation scheme proposed in this paper against that of shifted ILU(0) on solving the system $A\mathbf{x} = \mathbf{b}$. For the shifted ILU(0) preconditioner, the diagonal of matrix $A$ is perturbed by a small shift before performing the incomplete factorization. This shift serves to improve the quality of the ILU(0) factorization and makes it a more effective preconditioner [13, 27, 36]. Finding the optimal shift is not trivial; hence, for our tests, we run the same problem for several different diagonal shifts within the interval $[0, 1]$, using an increment of 0.1, and select the shift yielding the best result for shifted ILU(0).

TABLE 4.1

*Results for shifted ILU(0) on preconditioning the system $A\mathbf{x} = \mathbf{b}$, where $A = V^T V$ and $V$ is a shifted Laplace matrix with the shift $\varrho = -0.05, -0.1, -0.5$.*

| $\varrho$ | grid | nnz | $c_F$ | #iters | time | $\|\|(LU)^{-1}\mathbf{1}\|\|_2$ | $\|\|A - LU\|\|_F$ |
|---|---|---|---|---|---|---|---|
| $-0.05$ | $100 \times 100$ | 128004 | 1.0 | 99 | $0.92s$ | $4.77e+03$ | 137.94 |
| | $150 \times 150$ | 289504 | 1.0 | 107 | $2.23s$ | $7.55e+03$ | 208.48 |
| | $200 \times 200$ | 516004 | 1.0 | 113 | $3.95s$ | $1.03e+04$ | 279.00 |
| | $300 \times 300$ | 1164004 | 1.0 | 97 | $7.60s$ | $1.59e+04$ | 420.00 |
| $-0.1$ | $100 \times 100$ | 128004 | 1.0 | 109 | $0.99s$ | $1.08e+03$ | 128.65 |
| | $150 \times 150$ | 289504 | 1.0 | 98 | $2.02s$ | $1.65e+03$ | 194.16 |
| | $200 \times 200$ | 516004 | 1.0 | 127 | $4.36s$ | $2.22e+03$ | 259.66 |
| | $300 \times 300$ | 1164004 | 1.0 | 109 | $8.18s$ | $3.37e+03$ | 390.68 |
| $-0.5$ | $100 \times 100$ | 128004 | 1.0 | 169 | $1.44s$ | $4.48e+02$ | 142.44 |
| | $150 \times 150$ | 289504 | 1.0 | 156 | $3.20s$ | $6.77e+02$ | 215.17 |
| | $200 \times 200$ | 516004 | 1.0 | 150 | $5.44s$ | $9.06e+02$ | 287.89 |
| | $300 \times 300$ | 1164004 | 1.0 | 144 | $10.47s$ | $1.36e+03$ | 433.34 |

TABLE 4.2

*Results for MILUT with relaxed compensation on preconditioning the system $A\mathbf{x} = \mathbf{b}$, where $A = V^T V$ and $V$ is a shifted Laplace matrix with the shift $\varrho = -0.05, -0.1, -0.5$.*

| $\varrho$ | grid | nnz | $c_F$ | #iters | time | $\|\|(LU)^{-1}\mathbf{1}\|\|_2$ | $\|\|A - LU\|\|_F$ |
|---|---|---|---|---|---|---|---|
| $-0.05$ | $100 \times 100$ | 128004 | 1.82 | 59 | $0.74s$ | $1.67e+03$ | 25.49 |
| | $150 \times 150$ | 289504 | 1.83 | 65 | $2.05s$ | $2.52e+03$ | 38.15 |
| | $200 \times 200$ | 516004 | 1.83 | 77 | $3.93s$ | $3.78e+03$ | 50.88 |
| | $300 \times 300$ | 1164004 | 1.84 | 69 | $7.46s$ | $5.09e+03$ | 76.42 |
| $-0.1$ | $100 \times 100$ | 128004 | 1.82 | 62 | $0.79s$ | $1.52e+03$ | 25.91 |
| | $150 \times 150$ | 289504 | 1.83 | 60 | $1.68s$ | $2.28e+03$ | 38.11 |
| | $200 \times 200$ | 516004 | 1.83 | 67 | $3.25s$ | $3.03e+03$ | 51.79 |
| | $300 \times 300$ | 1164004 | 1.84 | 63 | $6.78s$ | $4.54e+03$ | 77.82 |
| $-0.5$ | $100 \times 100$ | 128004 | 1.61 | 83 | $1.05s$ | $4.24e+02$ | 25.47 |
| | $150 \times 150$ | 289504 | 1.61 | 83 | $2.30s$ | $6.10e+02$ | 38.08 |
| | $200 \times 200$ | 516004 | 1.61 | 81 | $3.61s$ | $7.96e+02$ | 50.73 |
| | $300 \times 300$ | 1164004 | 1.61 | 82 | $8.09s$ | $1.17e+03$ | 76.15 |

Table 4.1 details the numerical results for this shifted ILU(0) strategy. Here and in all the tests in this subsection, we use restarted GMRES with a restart dimension of 100 and an initial guess $\mathbf{x}^{(0)} = \mathbf{0}$. The maximum number of outer GMRES iterations for these tests is fixed at 500. We assume convergence of the iteration when the $\ell_2$-norm of the residual is reduced by a relative factor of $10^7$. We introduce the quantity $||(LU)^{-1}\mathbf{1}||_2$ which represents a lower bound estimate of the conditioning of the inverse $LU$ factors as a measure of the stability of the factorization. We also introduce the quantity $||A - LU||_F$ as a measure of the accuracy of $LU$ as an approximation to $A$. To measure the cost of the storage and computation of matrix-vector products with the preconditioner, we define the fill-factor as $c_F = (nnz_L + nnz_U - n)/nnz$, where $n$ is the dimension of the problem, $nnz$ is the number of nonzeros of the original matrix $A$, and $nnz_L$ and $nnz_U$ are the number of nonzeros in the $L$ and $U$ matrices, respectively. Since the unit diagonal of $L$ is not stored, it is compensated for by subtracting $n$ in the equation. We also report the total time needed for the computation of the factors and solution of the linear system. For this set of examples, we fix the matching vector, $\mathbf{t}$, used to constrain the MILUT method, to be the vector of all ones.

Table 4.2 shows results for the MILUT method with relaxed compensation on this example. The results from these tables indicate superior performance for the modified ILUT method over the shifted ILU(0) method. We observe that the modified ILUT

TABLE 4.3

*Results for standard ILUT (although with a modified dropping rule) on preconditioning the system $A\mathbf{x} = \mathbf{b}$, where $A = V^T V$ and $V$ is a shifted Laplace matrix with the shift $\varrho = -0.05, -0.1, -0.5$.*

| $\varrho$ | grid | nnz | $c_F$ | #iters | time | $\|\|(LU)^{-1}\mathbf{1}\|\|_2$ | $\|\|A - LU\|\|_F$ |
|---|---|---|---|---|---|---|---|
| $-0.05$ | $100 \times 100$ | 128004 | 1.75 | 183 | $2.22s$ | $2.26e+02$ | 38.15 |
| | $150 \times 150$ | 289504 | 1.76 | 186 | $5.03s$ | $3.38e+02$ | 57.47 |
| | $200 \times 200$ | 516004 | 1.76 | 212 | $9.94s$ | $4.50e+02$ | 76.78 |
| | $300 \times 300$ | 1164004 | 1.76 | 198 | $21.70s$ | $6.72e+02$ | 115.40 |
| $-0.1$ | $100 \times 100$ | 128004 | 1.75 | 157 | $1.87s$ | $2.17e+02$ | 39.66 |
| | $150 \times 150$ | 289504 | 1.76 | 149 | $3.96s$ | $3.23e+02$ | 59.88 |
| | $200 \times 200$ | 516004 | 1.76 | 186 | $8.71s$ | $4.29e+02$ | 80.09 |
| | $300 \times 300$ | 1164004 | 1.76 | 177 | $20.32s$ | $6.41e+02$ | 120.52 |
| $-0.5$ | $100 \times 100$ | 128004 | 1.60 | 130 | $1.86s$ | $1.26e+02$ | 62.91 |
| | $150 \times 150$ | 289504 | 1.60 | 154 | $3.88s$ | $1.86e+02$ | 94.90 |
| | $200 \times 200$ | 516004 | 1.61 | 141 | $6.23s$ | $2.46e+02$ | 127.04 |
| | $300 \times 300$ | 1164004 | 1.61 | 138 | $15.30s$ | $3.66e+02$ | 190.94 |

method yields factors that are often more stable and always more accurate and that the iteration based on MILUT converges in fewer iterations for all the test cases.

For these structured-grid problems, the memory efficiency of ILU(0) is difficult to beat, particularly since threshold-based ILU strategies generally require some fill-in (beyond the zero level-of-fill) in order to get a good approximation by the preconditioner. Nonetheless, the resulting added computational cost in the modified ILUT factorization (due to fill-in) is made up for by the good iteration counts. As shown in Tables 4.1 and 4.2, the iteration count for the MILUT method is on average about 1.5 times better than that of the shifted ILU(0) method. This savings in iterations is multiplied because we are using GMRES. Since GMRES converges faster with the MILUT preconditioner, fewer Arnoldi vectors need to be stored and orthogonalized, resulting in even more savings than just by the simple ratio of iterations.

The standard ILUT, using the same dropping criteria as the MILUT method with relaxed compensation, yields poor and unstable factors for these problems. This leads to convergence failure of the restarted GMRES iterations. However, adjusting the dropping rule so that it is based on comparing the size of the fill-in entry or column update ($w_k \cdot L_{j,k}$, $j = k+1, \ldots, n$, in line 3 of Algorithm 3) to some scaling parameter related to the drop tolerance yields factors that were "good enough" to handle the problem. Notice that this dropping rule differs from the approach discussed in section 2 (see Algorithms 2 and 3), where the comparison is directly between the pivot element, $w_k$, and the scaling parameter. Results for this approach are shown in Table 4.3. For a fair comparison with the results in Table 4.2, we aim to attain similar fill-factors for both the standard ILUT method and the MILUT method. Comparing the results in Tables 4.2 and 4.3, we see that the MILUT method shows better performance than the standard ILUT method, particularly in terms of the iteration counts, computational times, and accuracy. Nonetheless, we observe that the standard ILUT method shows better stability measures than the MILUT method. This may be attributed to the new dropping rule imposed to help the convergence of the standard ILUT method.

One interesting observation is that as more fill-in is allowed in the factorization, the performance of the standard ILUT method deteriorates. We further observe that small decrements in the drop tolerance $\tau$ often result in sudden (significant) jumps in the fill-factor $c_F$ (sometimes by an order of magnitude). From the standard ILUT algorithm (Algorithm 2 or 3), we see that the method can be prone to generating

dense columns during the factorization. For instance, if the dropping rule is such that very few or no fill-in is dropped during the elimination of a particular column, then the $L$ and $U$ parts of the column may become dense. This is undesirable because the dense column can very easily be propagated across the remaining columns yet to be eliminated, making them dense as well (leading to large fill-factors). Further, suppose that during the elimination of a subsequent column, a pivot entry is judged small enough and hence discarded by the dropping rule. Then the resulting error in the column (and hence in the factorization) is a factor of the dense column, which can be significant. As such, the resulting $L$ and $U$ factors could be of poor quality, even though they may be dense (fill-factor may be large).

MILUT with relaxed compensation, on the other hand, is less susceptible to these problems. Adding the compensation $\sigma$ to the diagonal promotes dropping in $L$, making it less likely to be dense. Furthermore, recall from section 3.1.1 that the choice of the modification $\mathbf{z}$ is based on stability considerations. However, it turns out that it may also have some implications on the accuracy of the factorization, particularly when $\mathbf{z}$ is small. The error in the column, induced by dropping the pivot entry, may be reduced by adding the column $\mathbf{z}$ to $\mathbf{l}$, leading to a more accurate factor for the $L$ part of the factorization.

**4.2. Results for imaginary compensation.** In what follows, we present some numerical results on an acoustic wave diffraction problem, governed by the Helmholtz equation. We compare the performance of standard ILUT with that of MILUT, where the modification is by an imaginary term as discussed earlier in section 3.2.

The physical problem models a plane wave propagating along the $x$-axis and an incident on a bounded obstacle in the form of a disk of radius 0.5 meters. The computational domain is discretized by the Galerkin least-squares finite-element method, using an isoparametric discretization over quadrilateral elements, on a $161 \times 361$ grid. An artificial boundary condition is imposed at a distance 1.5 meters from the obstacle, using the Dirichlet-to-Neumann technique, to satisfy the Sommerfeld radiation condition [21]. The resulting system has size $n = 57960$, with 516,600 nonzero entries and is complex, symmetric (but not Hermitian), and indefinite. We use restarted GMRES with a restart dimension of 100 and an initial guess $\mathbf{x}^{(0)} = \mathbf{0}$. The maximum number of GMRES iterations is fixed at 500, and we assume convergence when the $\ell_2$-norm of the residual is reduced by a relative factor of $10^7$. The right-hand side is artificially created by assuming the entries of the solution vector are chosen from the uniform distribution on [0, 1]. We solve the system for increasing values of the wavenumber $k$; because we consider a fixed grid, this makes the overall mesh resolution (measured in number of points per wavelength, ppw) decreasing as we increase $k$ (from the highest of 160 ppw for $k = 2\pi$ to a low of 10 ppw for $k = 32\pi$), leading to more indefinite and challenging problems. For each value of $k$, we run the problem several times, and the results shown are averaged over the runs.

The fill level parameter $p$ for the factorization is fixed at 1000 for both standard ILUT and MILUT, and the drop tolerance $\tau$ is adjusted so that the fill-factor is similar for both methods. As before, the dropping rule for standard ILUT is adjusted so that it is based on the size of the fill-in (update) entry rather than on the pivot entry in order to yield "good" factors.

From Table 4.4, we observe a remarkable performance for the MILUT scheme, compared to standard ILUT. For high values of the wavenumber, the standard ILUT factorization can become unstable. For this set of examples, ILUT requires less fill-in in order to produce stable factors. However, this implies that the preconditioner

TABLE 4.4
*Results for ILUT and MILUT with imaginary modification on the Helmholtz problem with different values of the wavenumber $k$.*

| | $k$ | $c_F$ | #iters | total time | $\|\|(LU)^{-1}\mathbf{1}\|\|_2$ | $\|\|(A-LU)\mathbf{1}\|\|_2$ |
|---|---|---|---|---|---|---|
| ILUT | $2\pi$ | 2.67 | 135 | $23.54s$ | $1.92e+03$ | 10.38 |
| | $4\pi$ | 2.68 | 156 | $31.93s$ | $2.06e+03$ | 10.53 |
| | $8\pi$ | 2.74 | 198 | $32.37s$ | $2.94e+03$ | 11.25 |
| | $16\pi$ | 2.42 | $> 500$ | $75.90s$ | $1.88e+04$ | 21.30 |
| | $32\pi$ | * | * | * | * | * |
| MILUT | $2\pi$ | 2.61 | 114 | $21.15s$ | $2.38e+03$ | 18.82 |
| | $4\pi$ | 2.65 | 121 | $22.56s$ | $2.61e+03$ | 18.90 |
| | $8\pi$ | 2.64 | 134 | $20.40s$ | $3.32e+03$ | 22.05 |
| | $16\pi$ | 2.48 | 206 | $31.68s$ | $2.44e+03$ | 37.26 |
| | $32\pi$ | 3.86 | 182 | $34.01s$ | $3.03e+02$ | 36.02 |

is poorly approximated, which makes it difficult for the solver to converge. This is evident for the example with wavenumber $k = 16\pi$, where the solver fails to converge within the required number of iterations with ILUT as preconditioner. If more fill-in is allowed, the resulting factors become unstable, with values of $||(LU)^{-1}\mathbf{1}||_2$ on the order of $10^{100}$ or larger. This renders the factors useless as preconditioners for any iterative method. Comparing the results for ILUT with those of MILUT, we see that MILUT is more efficient and robust on the Helmholtz problem. By using imaginary perturbations to modify the diagonal, the resulting factors from the MILUT scheme are quite stable and yield good results for the problem, even at high wavenumbers.

For the test problem with wavenumber $k = 32\pi$, ILUT shows no signs of convergence. Even when fill-in is reduced so that the fill-factor $c_F$ is $\approx 1.0$, the value of $||(LU)^{-1}\mathbf{1}||_2$ for the resulting $LU$ factors is of the order of $10^{144}$. MILUT, however, produces stable factors and was successful in solving the problem. These results agree quite well with the results in [30], where imaginary perturbations were used to construct a shifted ILUT preconditioner for the Helmholtz problem. Numerical results showed that the resulting preconditioner was effective on the Helmholtz problem at relatively high wavenumbers and low mesh resolutions. It is worth noting that the approach used in [30] differs from the optimal strategy discussed in this paper. The focus of [30] is on improving the quality of the preconditioner by improving the diagonal dominance of the rows of the matrix prior to performing the ILUT factorization. The compensation strategy that defines the imaginary shift relies on two heuristics, both aimed at improving diagonal dominance entirely through diagonal compensation. Here, we use a different approach to define the imaginary shift, by solving an optimization problem with stability and accuracy constraints, during the ILUT factorization. The resulting compensation is active not only on the diagonal entry but also on the entries in the $L$-part of the corresponding column.

**4.3. Results for the row-based MILUT scheme.** In what follows, we present some numerical results for the row-based MILUT scheme. In these results, we have implemented Algorithm 5 as a modification of the ILUT routine from SPARSKIT [33], a Fortran-77 toolkit for working with sparse matrices. The implementation includes an Arnoldi algorithm for computing the matching vectors, based on a given size of the Krylov subspace. Both the Arnoldi algorithm and the MILUT algorithm are tightly coupled to the LAPACK and BLAS packages. All results are computed on a dual-processor 3.0 GHz Xeon machine with 2 GB of RAM. For all results in this subsection with ILUT and MILUT, the maximum fill-in is limited by allowing up to 20 nonzero elements in each row of $L$ and $U$ (not counting the unit diagonal).

TABLE 4.5
*Performance of ILUT on 2D finite-difference Laplacian with drop tolerance of* 0.01.

| Grid | $n$ | $nnz$ | $c_F$ | $t_{\text{setup}}$ | $t_{\text{solve}}$ | # iters. |
|---|---|---|---|---|---|---|
| $65 \times 65$ | 3969 | 19593 | 2.88 | 0.005 | 0.03 | 17 |
| $129 \times 129$ | 16129 | 80137 | 2.94 | 0.02 | 0.18 | 29 |
| $257 \times 257$ | 65025 | 324105 | 2.97 | 0.07 | 1.33 | 47 |
| $513 \times 513$ | 261121 | 1303561 | 2.99 | 0.30 | 12.15 | 74 |

TABLE 4.6
*Performance of ILU($k$) on 2D finite-difference Laplacian with levels of fill $k = 3$, and $k = 4$.*

| Grid | $n$ | $nnz$ | $k$ | $c_F$ | $t_{\text{setup}}$ | $t_{\text{solve}}$ | # iters. |
|---|---|---|---|---|---|---|---|
| $65 \times 65$ | 3969 | 19593 | 3 | 2.54 | 0.004 | 0.03 | 18 |
| $129 \times 129$ | 16129 | 80137 | 3 | 2.57 | 0.02 | 0.18 | 31 |
| $257 \times 257$ | 65025 | 324105 | 3 | 2.59 | 0.07 | 1.28 | 46 |
| $513 \times 513$ | 261121 | 1303561 | 3 | 2.59 | 0.29 | 12.49 | 78 |
| $65 \times 65$ | 3969 | 19593 | 4 | 3.30 | 0.006 | 0.03 | 15 |
| $129 \times 129$ | 16129 | 80137 | 4 | 3.35 | 0.03 | 0.17 | 25 |
| $257 \times 257$ | 65025 | 324105 | 4 | 3.37 | 0.11 | 1.05 | 37 |
| $513 \times 513$ | 261121 | 1303561 | 4 | 3.39 | 0.43 | 8.98 | 61 |

**4.3.1. 2D Laplacian.** First, we consider the (unshifted) 2D finite-difference Laplacian with Dirichlet boundary conditions on a uniform grid of the unit square. As the dropping strategy within the incomplete factorization preconditioners considered here is not symmetric, we consider the performance of these strategies as preconditioners for GMRES. The system matrix is created in the standard way; then a random vector $\mathbf{x}$ (with each entry $x_i$ independently chosen from a uniform distribution on $[0, 1]$) is chosen as the solution, and the right-hand side $\mathbf{b} = A\mathbf{x}$ is computed. We choose a random solution as for this problem we expect the matching vector chosen in the modification process to have significant structure (e.g., the constant vector) and do not want this structure to impact the performance of the Krylov space method.

As a baseline for comparison, we consider the performance of the standard ILUT algorithm [34] as a preconditioner for GMRES. Fixing the drop tolerance as 0.01 yields the results reported in Table 4.5. Setup and solve times, $t_{\text{setup}}$ and $t_{\text{solve}}$ (resp.), are also reported (rounded to the nearest hundredth of a second unless less than 0.01), as are the number of iterations needed to reduce the $\ell_2$-norm of the residual by a relative factor of $10^7$.

Performance can also be compared to a level-of-fill-based strategy, ILU($k$) [36, section 10.3.3]. Because the control over the complexity of the resulting preconditioner is less precise (as only integer levels of fill may be chosen), it is not possible to closely match the preconditioner complexities of Table 4.5. Thus, in Table 4.6, we give results for levels of fill ($k$) of three and four, with slightly smaller and slightly larger overall complexities. Here, as expected, performance is slightly worse than that of ILUT when the preconditioner complexity is smaller and somewhat better when the preconditioner complexity is larger. If memory requirements do not pose a constraint, we see that ILU(4) outperforms both ILUT and ILU(3) in terms of iteration counts and solution time on all grids.

We begin testing the MILUT algorithm in the setting of classical modified ILU, i.e., with the diagonal entries of the $U$ factor modified so that $LU\mathbf{1} = A\mathbf{1}$, where $\mathbf{1}$ is the vector of all ones. Table 4.7 shows the results of these tests, first for the same drop tolerance $\tau = 0.01$ as used for ILUT in Table 4.5 and then with the drop tolerance adjusted so that the preconditioner complexities $c_F$ nearly match those of ILUT.

TABLE 4.7
*Performance of MILUT based on the constant vector on 2D finite-difference Laplacian.*

| Grid | $n$ | $nnz$ | $\tau$ | $c_F$ | $t_{\text{setup}}$ | $t_{\text{solve}}$ | # iters. |
|---|---|---|---|---|---|---|---|
| $65 \times 65$ | 3969 | 19593 | 0.010 | 3.15 | 0.01 | 0.03 | 16 |
| $129 \times 129$ | 16129 | 80137 | 0.010 | 3.35 | 0.06 | 0.17 | 26 |
| $257 \times 257$ | 65025 | 324105 | 0.010 | 3.47 | 0.22 | 1.25 | 43 |
| $513 \times 513$ | 261121 | 1303561 | 0.010 | 3.54 | 0.92 | 12.39 | 75 |
| $65 \times 65$ | 3969 | 19593 | 0.016 | 2.78 | 0.01 | 0.03 | 14 |
| $129 \times 129$ | 16129 | 80137 | 0.016 | 2.89 | 0.05 | 0.13 | 19 |
| $257 \times 257$ | 65025 | 324105 | 0.017 | 2.92 | 0.20 | 0.72 | 26 |
| $513 \times 513$ | 261121 | 1303561 | 0.017 | 2.96 | 0.78 | 4.62 | 38 |

The results in Table 4.7 are somewhat surprising. As expected, we see some improvement in the performance of MILUT over that of ILUT for the same fixed drop tolerance. In part, this is expected because of the well-known theoretical analysis of modified incomplete factorizations, but it is also to be expected because the preconditioner complexities are somewhat larger than those for ILUT. What is surprising is that when the drop tolerance is raised (so that fewer nonzero entries are kept in the preconditioner), the performance of the modified preconditioners uniformly improves. While unexpected, this is not impossible, as the modification of the diagonal entries in early rows of the matrix (which, of course, depends on the drop tolerance) has a significant effect on the construction of the preconditioner for the later rows of $A$. Thus, extra dropping may result in a better spectral equivalence between the preconditioner and $A$, even though they are further apart in an elementwise sense.

If a good vector for use in the modification process is not known beforehand, then it is possible to expose a good candidate through an Arnoldi process. On the left of Figure 4.1, the number of iterations required to reduce the residual by a relative factor of $10^7$ using the resulting modified ILUT-preconditioned GMRES algorithm based on the vector corresponding to the smallest Ritz value is shown. Here, the drop tolerance was fixed at 0.01. For both the $128^2$ and $256^2$ grids, we see significant variation in the number of iterations required for convergence, until the size of the Arnoldi space approaches the number of cells in one direction of the mesh. The Arnoldi iterations were started with a vector whose entries were chosen using a pseudo-random number generator with a uniform distribution on $[0, 1]$ (and the same vector was used as the starting vector for Arnoldi for all tests on a given grid). On the right of Figure 4.1, we see that even though there is significant variation in the performance of the resulting preconditioners, the Rayleigh quotient of the selected vectors is monotonically decreasing toward the smallest eigenvalue of the matrix (as expected).

As seen in Figure 4.1 for the finite-difference Poisson matrix, the number of steps of the Arnoldi process needed to guarantee good solver performance is, unfortunately, proportional to (the square root of) the matrix dimension. Thus, for fixed size of the Arnoldi subspace, we expect the performance of preconditioners defined in this way to degrade as problem size increases. In Table 4.8, 30 steps of the (unpreconditioned) Arnoldi iteration on $A$ are performed with the vector corresponding to the smallest Ritz value used in the modification. Results are given for both a fixed drop tolerance $\tau$ and with drop tolerances adjusted so that the preconditioner complexities are close to those of Table 4.5. In both cases, we see preconditioner performance that degrades with problem size (as expected). For fixed $\tau = 0.01$, performance is similar to that of unmodified ILUT or ILU($k$), in terms of both number of iterations and total time to solution, and for variable $\tau$ performance is closer to that achieved by MILUT based on the constant vector. However, in comparison to the results in Table 4.7, we see that the

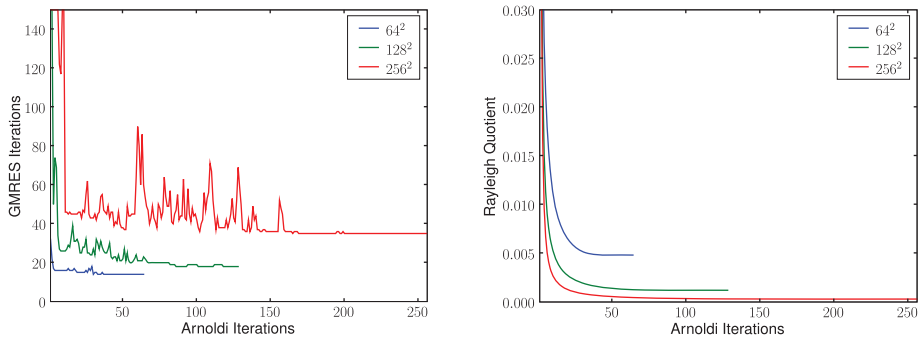



FIG. 4.1. *Number of iterations of the resulting modified ILUT-preconditioned GMRES required to reduce residual by a relative factor of* $10^7$ *for increasing dimension of Arnoldi process (left) and Rayleigh quotient of the vector associated with the smallest Ritz value used in this modification (right).*

TABLE 4.8
*Performance of MILUT based on the vector associated with the smallest Ritz value after* 30 *steps of the Arnoldi iteration on* 2*D finite-difference Laplacian.*

| Grid | $n$ | $nnz$ | $\tau$ | $c_F$ | $t_{\text{setup}}$ | $t_{\text{solve}}$ | # iters. |
|---|---|---|---|---|---|---|---|
| $65 \times 65$ | 3969 | 19593 | 0.010 | 3.14 | 0.04 | 0.03 | 14 |
| $129 \times 129$ | 16129 | 80137 | 0.010 | 3.34 | 0.16 | 0.18 | 27 |
| $257 \times 257$ | 65025 | 324105 | 0.010 | 3.46 | 0.86 | 1.25 | 43 |
| $513 \times 513$ | 261121 | 1303561 | 0.010 | 3.53 | 4.30 | 15.38 | 78 |
| $65 \times 65$ | 3969 | 19593 | 0.016 | 2.81 | 0.04 | 0.03 | 15 |
| $129 \times 129$ | 16129 | 80137 | 0.017 | 2.85 | 0.15 | 0.15 | 24 |
| $257 \times 257$ | 65025 | 324105 | 0.017 | 2.92 | 0.83 | 0.89 | 33 |
| $513 \times 513$ | 261121 | 1303561 | 0.017 | 2.96 | 4.20 | 7.14 | 53 |

setup times alone for computing the Arnoldi vectors and the MILUT factorization are comparable to the minimum total times required for ILUT with modification based on the constant vector. Thus, even though improved preconditioner performance may be realized, it is difficult to justify the added expense of computing the large Krylov subspaces necessary to gain better spectral accuracy starting from a random initial guess.

The potential to achieve better performance, however, suggests that the MILUT approach coupled with Arnoldi iteration may be viable for determining a better preconditioner than arises from classical MILU approaches. For the Poisson problem, it is known that the constant vector is "close to" the eigenvector belonging to the smallest eigenvalue of $A$ (where closeness is measured in terms of the Rayleigh quotient). This is, in fact, often the case, where some properties of the discrete operator are known a priori. Figure 4.2 shows how the number of iterations required of GMRES to reduce the residual by a relative factor of $10^7$ varies with the number of Arnoldi steps and how the Rayleigh quotient of the vector associated with the smallest Ritz value changes when starting from the vector of all ones. Again, we see that while the Rayleigh quotient of this vector steadily decreases, there is some significant variation in the GMRES iteration count, although it is not as significant as in the case of a random starting vector for the Arnoldi iteration (shown in Figure 4.1).

**4.3.2. 2D Helmholtz.** Here, we consider the same Helmholtz matrices as were considered in section 4.2; however, because we use only the row-based modification strategy without imaginary compensation, we consider only the two smallest

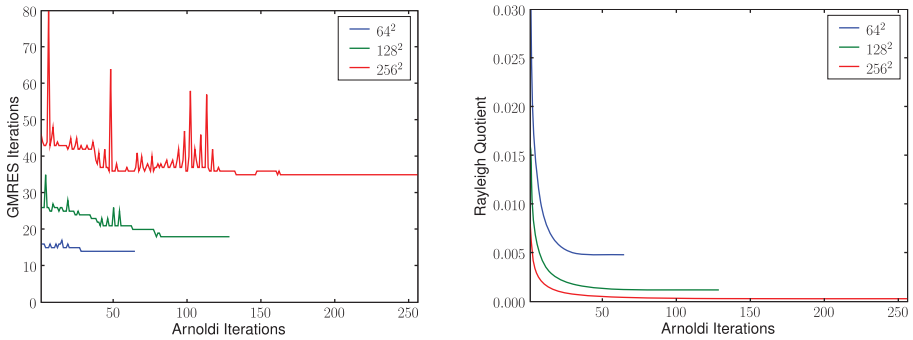



FIG. 4.2. *Number of iterations of the resulting MILUT preconditioned GMRES required to reduce residual by a relative factor of* $10^7$ *for increasing dimension of Arnoldi process started using the constant vector (left) and Rayleigh quotient of the vector associated with the smallest Ritz value used in this modification (right).*

TABLE 4.9

*GMRES iteration counts and preconditioner complexities for row-based ILUT with specified dropping tolerance* $\tau$ *(ILUT(*$\tau$*)) applied to the Helmholtz problem with wavenumber* $k = 2\pi$.

| | ILUT(0.001) | ILUT(0.002) | ILUT(0.003) | ILUT(0.004) | ILUT(0.005) |
|---|---|---|---|---|---|
| Iters. | 178 | 194 | 213 | 227 | 260 |
| $c_F$ | 4.38 | 4.08 | 3.78 | 3.36 | 3.10 |

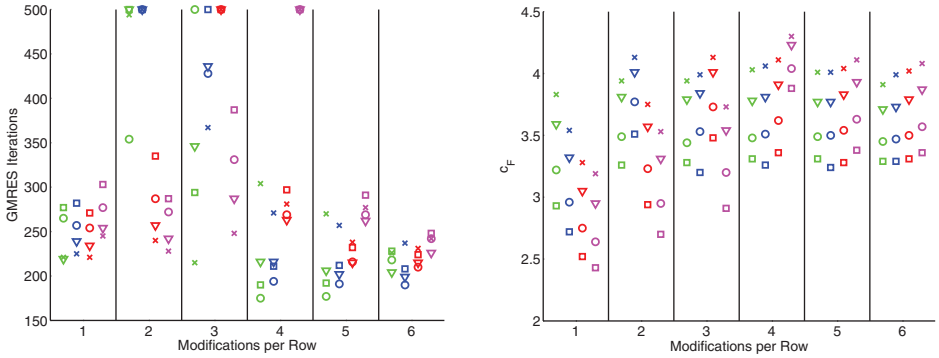
TABLE 4.10

*GMRES iteration counts and preconditioner complexities for row-based level-of-fill ILU (ILU(k)) applied to the Helmholtz problem with wavenumber* $k = 2\pi$.

| | ILU(3) | ILU(4) | ILU(5) | ILU(6) | ILU(7) |
|---|---|---|---|---|---|
| Iters. | 281 | 241 | 215 | 191 | 173 |
| $c_F$ | 2.33 | 2.77 | 3.21 | 3.65 | 4.09 |

wavenumber problems, $k = 2\pi$ and $k = 4\pi$. The factorization and compensation strategies used here are very different from those discussed in section 3.2; thus, these represent two challenging test problems without those approaches, and the results we report here are generally worse than those obtained using imaginary compensation or even using the column-based factorization and dropping strategy used in section 4.2. Thus, for reference, Tables 4.9 and 4.10 provide preconditioned GMRES iteration counts for solution (reducing the residual by a relative factor of $10^7$) and preconditioner complexity $c_F$ for the Helmholtz problem with wavenumber $k = 2\pi$. In both cases, we see the expected behavior; as more fill-in is allowed within the ILU factors, the precondioners improve (when measured purely in terms of the GMRES iteration counts).

Figure 4.3 presents the results for a range of tests of the MILUT strategy applied to the problem with wavenumber $k = 2\pi$, where the vectors are selected by taking those with Ritz values closest to zero (in modulus) after applying 40 Arnoldi steps to a random initial guess. The plot of GMRES iteration counts shows several interesting results. First, as expected, a smaller dropping tolerance $\tau$ often (but not always) leads to a lower iteration count and always leads to a larger preconditioner complexity (although not by uniform increments). For a fixed number of modifications per row, using more vectors (looking from left-to-right within each group of results) is




(a) GMRES iteration counts (b) Preconditioner complexities

FIG. 4.3. *GMRES iteration counts and preconditioner complexities for row-based MILUT applied to the Helmholtz problem with wavenumber* $k = 2\pi$. *In both plots, the data are organized by the number of coefficients modified in each row, from* 1 *(the nonunit diagonal) through* 6. *For each group, the four columns show results for using one through four vectors to guide the modification from left to right, also highlighted by color (green for one vector, blue for two, red for three, and magenta for four). Within each column, results for four different dropping tolerances are shown,* $\times$ *for* $\tau = 0.005$, $\triangledown$ *for* $\tau = 0.006$, $\bigcirc$ *for* $\tau = 0.008$, *and* $\square$ *for* $\tau = 0.01$. *Failure to converge is denoted by* 500 *iterations.*

sometimes useful, but not always. Indeed, there seems to be no large-scale trend in the effect of using more vectors; when four modifications per row are used, matching four vectors (so that the modification is uniquely determined) is never successful, but when using only two modifications per row, using four vectors is always best, and three vectors is always better than using only one or two. In many but not all cases, iteration counts improve when modifying more coefficients with a fixed number of vectors and dropping tolerance; however, the effects are not uniform. When matching only a single vector, the best results are achieved when modifying four, five, or six coefficients per row, depending on the dropping tolerance used. The best preconditioner from this set of tests is achieved with a single vector, modifying four coefficients with $\tau = 0.008$, taking 175 iterations. This preconditioner has a complexity $c_F = 3.48$. Achieving comparable iteration counts using either standard ILUT or ILU($k$) requires preconditioner complexities of over 4, showing some notable improvement using this modification strategy.

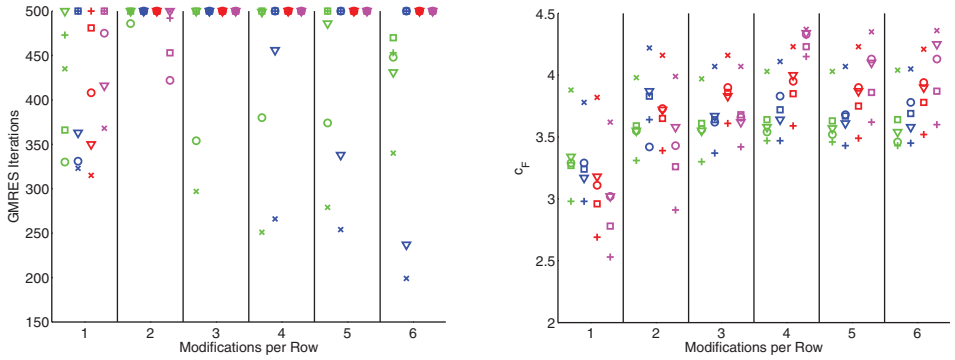
The Helmholtz problem with wavenumber $k = 4\pi$ is significantly more difficult for this family of preconditioners. Results for classical ILUT and ILU($k$) approaches are shown in Tables 4.11 and 4.12. We note that for drop tolerances $\tau > 0.05$ and levels of fill $k < 5$, we see uniformly poor convergence without reaching the relative residual tolerance within 500 GMRES iterations. Plots for the MILUT strategy are shown in Figure 4.4, where we now vary both the drop tolerance and the size of the Arnoldi space. Here, we see that many of the attempted sets of parameters lead to failure of GMRES to converge within 500 iterations. In one case, with $\tau = 0.008$ and 80 Arnoldi steps, the problem using two vectors to modify two coefficients leads to a failure in solving for the modification in one row of the factorization due to the linear system for the modification being of less than full rank; in practice, this could be addressed by recognizing this upon failure of the LAPACK routine dgels and retrying with a smaller but full rank system, but we do not follow this approach here as such failures are, in our experience, very rare.

TABLE 4.11
*GMRES iteration counts and preconditioner complexities for row-based ILUT with specified dropping tolerance $\tau$ (ILUT($\tau$)) applied to the Helmholtz problem with wavenumber $k = 4\pi$.*

| | ILUT(0.001) | ILUT(0.002) | ILUT(0.003) | ILUT(0.004) | ILUT(0.005) |
|---|---|---|---|---|---|
| Iters. | 365 | 388 | 424 | 475 | $> 500$ |
| $c_F$ | 4.39 | 4.15 | 3.86 | 3.44 | 3.18 |

TABLE 4.12
*GMRES iteration counts and preconditioner complexities for row-based level-of-fill ILU (ILU(k)) applied to the Helmholtz problem with wavenumber $k = 4\pi$.*

| | ILU(4) | ILU(5) | ILU(6) | ILU(7) | ILU(8) |
|---|---|---|---|---|---|
| Iters. | $> 500$ | 486 | 430 | 385 | 350 |
| $c_F$ | 2.77 | 3.21 | 3.65 | 4.09 | 4.53 |




(a) GMRES iteration counts (b) Preconditioner complexities

FIG. 4.4. *GMRES iteration counts and preconditioner complexities for row-based MILUT applied to the Helmholtz problem with wavenumber $k = 4\pi$. In both plots, the data are organized by the number of coefficients modified in each row, from 1 (the nonunit diagonal) through 6. For each group, the four columns show results for using one through four vectors to guide the modification from left to right, also highlighted by color (green for one vector, blue for two, red for three, and magenta for four). Within each column, results for different dropping tolerances and sizes of the Arnoldi subspace are shown, $\times$ for $\tau = 0.005$ with 120 Arnoldi steps, $\triangledown$ for $\tau = 0.008$ with 120 Arnoldi steps, $\bigcirc$ for $\tau = 0.008$ with 80 Arnoldi steps, $\square$ for $\tau = 0.008$ with 40 Arnoldi steps, and $+$ for $\tau = 0.01$ with 40 Arnoldi steps. Failure to converge is denoted by 500 iterations.*

From Figure 4.4, it appears that using only a single modification per row is the most robust strategy, although it does not lead to the most efficient preconditioner by a substantial margin. When modifying only the nonunit diagonal, the best results were achieved using three vectors to overdetermine the modification with $\tau = 0.005$ and using 120 Arnoldi steps. Here, we also see the important differences between using 40, 80, or 120 Arnoldi steps to determine the modification vectors. Only when using a single vector to make the diagonal modification is using 40 steps more successful than using fewer; with more vectors, using more steps of Arnoldi to determine the vectors used to guide the modification seems to be the best practice. The best results overall from these sets of parameters occur when using two vectors to guide the modification of six coefficients per row. With $\tau = 0.005$, the system can be solved in 199 iterations with a preconditioner complexity of 4.05. This is nearly half as many iterations as the unmodified ILUT or ILU($k$) approaches required with similar preconditioner complexities. This demonstrates the potential usefulness of the generalizations of the

standard single-vector, single-coefficient modifications of ILU that have been explored before in the literature, although this introduces the somewhat daunting challenge of finding the correct combination of parameters needed for a given problem.

**5. Conclusions.** This paper describes three procedures for defining modifications of ILU factorizations with threshold-based dropping. The first procedure, based on the column version of ILUT, extends the standard diagonal compensation idea for MILUT by spreading the compensation term over the nonunit diagonal, as well as the nonzero entries in the $L$ part of the column, in an optimal way. The technique is further extended to exploit the use of imaginary shifts for the compensation term. Numerical results show that these MILUT methods are robust for both ill-conditioned and indefinite problems. By adding a compensation term to the $L$ part of the column as well as the nonunit diagonal, it appears that the MILUT method is less susceptible to the problems that cause instabilities and inaccuracies in the standard ILUT factorization. The third procedure uses an arbitrary set of "matching" vectors for which the product $LU$ is constrained to give the same product as the matrix, $A$. By modifying multiple coefficients in each row of the ILUT factors based on a least-squares fitting problem, substantially improved performance is realized.

**Acknowledgments.** We would like to thank Kechroud, Gowda, and Soulaimani for providing us with the finite-element code for acoustic wave scattering, governed by the Helmholtz equation; see [21].

REFERENCES

[1] R. E. Bank and C. Wagner, *Multilevel ILU decomposition*, Numer. Math., 82 (1999), pp. 543–576.
[2] R. Beauwens, *Modified incomplete factorization strategies*, in Preconditioned Conjugate Gradient Methods, Lecture Notes in Math. 1457, Springer, Berlin, 1990, pp. 1–16.
[3] M. Bollhöfer, M. J. Grote, and O. Schenk, *Algebraic multilevel preconditioner for the Helmholtz equation in heterogeneous media*, SIAM J. Sci. Comput., 31 (2009), pp. 3781–3805.
[4] M. Bollhöfer, *A robust ILU with pivoting based on monitoring the growth of the inverse factors*, Linear Algebra Appl., 338 (2001), pp. 201–213.
[5] E. Botta, A. Ploeg, and F. Wubs, *A fast linear-system solver for large unstructured problems on a shared-memory computer*, in Proceedings of the Conference on Algebraic Multilevel Methods with Applications, O. Axelsson and B. Polman, eds., 1996, pp. 105–116.
[6] E. Botta, A. Ploeg, and F. Wubs, *Nested grids ILU-decomposition (NGILU)*, J. Comput. Appl. Math., 66 (1996), pp. 515–526.
[7] E. Botta and F. Wubs, *Matrix renumbering ILU: An effective algebraic multilevel ILU*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 1007–1026.
[8] A. Brandt, J. Brannick, K. Kahl, and I. Livshits, *Bootstrap AMG*, SIAM J. Sci. Comput., 33 (2011), pp. 612–632.
[9] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation ($\alpha$SA)*, SIAM J. Sci. Comput., 25 (2004), pp. 1896–1920.
[10] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation ($\alpha$SA) multigrid*, SIAM Rev., 47 (2005), pp. 317–346.
[11] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive algebraic multigrid*, SIAM J. Sci. Comput., 27 (2006), pp. 1261–1286.
[12] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.
[13] T. F. Chan and H. A. van der Vorst, *Approximate and Incomplete Factorizations*, ICASE/LARC Interdisciplinary Ser. Sci. Engrg. 56, 1994, pp. 167–202.
[14] T. Chan and H. Elman, *Fourier analysis of iterative methods for elliptic problems*, SIAM Rev., 31 (1989), pp. 20–49.
[15] S. Doi and A. Hoshi, *Large numbered multicolor MILU preconditioning on SX-3/14*, Internat. J. Comput. Math., 44 (1992), pp. 143–152.

[16] T. Dupont, R. Kendall, and H. Rachford, *An approximate factorization procedure for solving self-adjoint elliptic difference equations*, SIAM J. Numer. Anal., 5 (1968), pp. 559–573.
[17] H. C. Elman, *A stability analysis of incomplete LU factorizations*, Math. Comput., 47 (1986), pp. 191–217.
[18] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik, *A novel multigrid based preconditioner for heterogeneous Helmholtz problems*, SIAM J. Sci. Comput., 27 (2006), pp. 1471–1492.
[19] I. Gustafsson, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142–156.
[20] P. Henon and Y. Saad, *A parallel multistage ILU factorization based on a hierarchical graph decomposition*, SIAM J. Sci. Comput., 28 (2006), pp. 2266–2293.
[21] R. Kechroud, A. Soulaimani, Y. Saad, and S. Gowda, *Preconditioning techniques for the solution of the Helmholtz equation by the finite element method*, Math. Comput. Simul., 65 (2004), pp. 303–321.
[22] Z. Li, Y. Saad, and M. Sosonkina, *pARMS: A parallel version of the algebraic recursive multilevel solver*, Numer. Linear Algebra Appl., 10 (2003), pp. 485–509.
[23] M. Magolu Monga Made, R. Beauwens, and G. Warzee, *Preconditioning of discrete Helmholtz operators perturbed by a diagonal complex matrix*, Comm. Numer. Methods Engrg., 16 (2000), pp. 801–817.
[24] M. Magolu Monga Made and H. van der Vorst, *A generalized domain decomposition paradigm for parallel incomplete LU factorization preconditionings*, Future Generation Comput. Syst., 17 (2001), pp. 925–932.
[25] M. Magolu Monga Made and H. van der Vorst, *Parallel incomplete factorizations with pseudo-overlapped subdomains*, Parallel Comput., 27 (2001), pp. 989–1008.
[26] M. Magolu Monga Made and H. van der Vorst, *Spectral analysis of parallel incomplete factorizations with implicit pseudo-overlap*, Numer. Linear Algebra Appl., 9 (2002), pp. 45–64.
[27] T. A. Manteuffel, *Shifted incomplete Cholesky factorization*, in Sparse Matrix Proceedings 1978, SIAM, Philadelphia, 1979, pp. 41–61.
[28] N. Munksgaard, *Solving sparse symmetric sets of linear equations by preconditioned conjugate gradient method*, ACM Trans. Math. Software, 6 (1980), pp. 206–219.
[29] Y. Notay, *DRIC: A dynamic version of the RIC method*, Numer. Linear Algebra Appl., 1 (1994), pp. 511–532.
[30] D. Osei-Kuffuor and Y. Saad, *Preconditioning Helmholtz linear systems*, Appl. Numer. Math., 60 (2009), pp. 420–431.
[31] J. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. McCormick, ed., Front. Appl. Math. 3 SIAM, Philadelphia, 1987.
[32] Y. Saad and B. Suchomel, *ARMS: An algebraic recursive multilevel solver for general sparse linear systems*, Numer. Linear Algebra Appl., 9 (2002).
[33] Y. Saad, *SPARSKIT: A Basic Tool Kit for Sparse Matrix Computations*, Technical report RIACS-90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffett Field, CA, 1990.
[34] Y. Saad, *ILUT: A dual threshold incomplete ILU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[35] Y. Saad, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (1996), pp. 830–847.
[36] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[37] H. De Sterck, T. A. Manteuffel, S. F. McCormick, Q. Nguyen, and J. Ruge, *Multilevel adaptive aggregation for Markov chains, with application to web ranking*, SIAM J. Sci. Comput., 30 (2008), pp. 2235–2262.
[38] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, New York, 2001.
[39] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.
[40] H. van der Vorst, *The convergence behaviour of preconditioned CG and CGS in the presence of rounding errors*, in Preconditioned Conjugate Gradient Methods, O. Axelsson and L. Kolotilina, eds., Lecture Notes in Math. 1457, Springer-Verlag, Berlin, 1990.
[41] M. B. van Gijzen, Y. A. Erlangga, and C. Vuik, *Spectral analysis of the discrete Helmholtz operator preconditioned with a shifted Laplacian*, SIAM J. Sci. Comput., 29 (2007), pp. 1942–1958.
[42] J. W. Watts III, *A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation pressure equation*, Soc. Petroleum Engineers J., 21 (1981), pp. 345–353.