Matrix Reordering Using Multilevel Graph Coarsening for ILU Preconditioning

Daniel Osei-Kuffuor * Ruipeng Li † Yousef Saad†

January 16, 2014

Abstract

Incomplete LU factorization (ILU) techniques are a well-known class of preconditioners, often used in conjunction with Krylov accelerators for the iterative solution of linear systems of equations. However, for certain problems, ILU factorizations can yield factors that are unstable, and in some cases quite dense. Reordering techniques based on permuting the matrix prior to performing the factorization have been shown to improve the quality of the factorization, and the resulting preconditioner. In this paper, we examine the effect of reordering techniques based on multilevel graph coarsening ideas on the level-based ILU(k) and the dual threshold ILUT algorithms. We consider an aggregation-based coarsening idea that implements two main coarsening frameworks - a top-down approach, and a bottom-up approach - each utilizing one of two different strategies to select the next-level coarse graph. Numerical results are presented to support our findings.

1 Introduction

Iterative methods based on Incomplete LU (ILU) preconditioners can be quite effective for solving certain types of linear systems of equations. Early work on the use of incomplete factorizations for preconditioning [2, 18, 28, 49, 65] led to the development of the level-based ILU(k) (or incomplete Cholesky, IC(k), for the symmetric case), see, e.g., [44,58,66]. Here, the underlying concept was founded on properties of M-matrices, and this generally resulted in poorer performance of ILU(k) preconditioners for general sparse systems. More robust alternatives, such as the threshold-based ILUT factorization, were later developed for general sparse matrices [45, 56].

However, for poorly structured matrices, ILU factorizations could generate significant fill-in, making the resulting L and U factors dense and hence inefficient as preconditioners. Reordering techniques borrowed from direct solution methods, such as the reverse Cuthill-McKee (RCM) ordering [32], the minimum degree ordering [1, 33, 53, 62], and the nested dissection ordering [26, 31], have been used to help alleviate this problem. Standard ILU factorizations are also prone to instability. First, there is the occurrence of very small pivots during the factorization, which can lead to a poor approximation to the original matrix. Column (or row) pivoting techniques have been typically employed to alleviate this [6, 27, 50, 58]. Some techniques permute the columns (or rows) of the sparse matrix during the factorization, so that columns (or rows) that are likely to yield unstable pivots are moved to the end.

A different type of instability is related to the fact that the L and U factors are themselves unstable, i.e., solving linear systems with them, will lead to unstable recurrences that grow exponentially. Such cases are characterized by very large norms $\|L^{-1}\|$ and $\|U^{-1}\|$, see, e.g., [17, 29]. Several techniques, such as modified ILU methods, see for instance [19, 25, 34, 41, 47, 64], shifted ILU methods, see for instance [42, 43, 51], as

*Center for Applied Scientific Computing, L-561, Lawrence Livermore National Laboratory, Livermore, CA 94551. email: oseikuffuor1@llnl.gov. Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. DE-AC52-07NA27344.

†Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455. email: {rli,saad}@cs.umn.edu. Work supported in part by DOE under grant DE-FG 08ER25841, in part by NSF under grant NSF/DMS-1216366, and in part by the Minnesota Supercomputer Institute

well as rigorous dropping strategies for ILUT, see [7], have been proposed in the literature to address this problem.

In recent years, there has been some interest in combining ILU factorizations with multilevel methods to obtain more efficient preconditioners for general sparse linear systems. One such example is the use of ILU(k), in particular ILU(0), as a smoother for the multigrid method [63]. In [7] and [8], Bollhöfer and co-workers used the idea of column pivoting, in combination with rigorous dropping strategies, to construct a multilevel preconditioner based on the Crout version of ILUT. One of the main ideas of multilevel methods is to define an ordering of the nodes in the adjacency graph of the matrix, by splitting the nodes into coarse and fine sets. This so-called C/F splitting is typically performed to promote independence among the nodes in the fine set. This allows for an efficient formation of the Schur complement matrix for the next level of the multilevel scheme. Furthermore, the nodes in the independent set can be eliminated simultaneously, which is beneficial for parallel computing. In [57], independent set orderings are used to develop a multi-elimination ILU preconditioner with the goal of promoting parallelism. In [9], Ploeg, Botta and Wubs use a grid dependent reordering scheme to construct an effective multilevel ILU method for elliptic PDEs. Later on in [10], Botta and Wubs generalize this further and define an ordering that selects a nearly independent set of nodes that are diagonally dominant, and from which an independent fine set is extracted. This helps to promote stability in the factorization, while reducing the cost in the formation of the Schur complement matrix for the next level. In [22], Chow and Vassilevski use the strength-of-connection idea from multigrid to define the C/F splitting of the nodes. Unlike some other procedures based on attaining a diagonally dominant fine set, this approach tries to ensure that the coarse set provides a good interpolation for the original problem. Other techniques have combined multilevel graph coarsening strategies with standard threshold-based ILU factorizations to yield robust multilevel preconditioners for general sparse linear systems, see for example [59, 60].

The goal of this paper is not to propose a new type of (multilevel) ILU techniques. Instead, the focus is to combine algebraic strategies with graph algorithms as a reordering tool to improve the stability of factored or ILU-type preconditioners. Thus, the work described in this paper may be more along the lines of earlier work presented in [14, 24], where ideas from graph theory are used to develop reordering techniques for ILU and factored approximate inverse preconditioners. In [14], a minimum inverse penalty (MIP) algorithm was developed by exploiting strategies to minimize the cost of the factorization and improve the accuracy of the factorization. The authors also present a weighted nested dissection strategy, and show its effectiveness on anisotropic problems. In [24] a minimum discarded fill (MDF) algorithm is presented as a reordering technique for ILU. The results show superior performance over traditional reordering strategies such as RCM, for unstructured grid problems. However, the cost of this reordering is generally higher than that of traditional methods. For both of these papers, the underlying observation is that for problems with anisotropy and inhomogeneity, reordering strategies that exploit the algebraic properties of the coefficient matrix are generally more effective than traditional methods.

In this paper, multilevel graph coarsening strategies are exploited to define an algebraic reordering technique for ILU. This constitutes one of the main differences between the work presented in this paper and previous work. It is worth noting that in [46, 48] the authors also make use of a reordering based on a multilevel graph coarsening strategy to develop an aggregation-based algebraic multigrid technique. However, the algebraic strategy for defining the coarsening is different. In this paper, we present two different strategies for determining which nodes are good candidates for the coarse or fine sets. We apply this reordering to the matrix prior to building its ILUT factorization. A multilevel dropping strategy is adopted, to be consistent with the level structure of the reordered system, and we show that the strategies described in this paper are also feasible with the level-based ILU factorization. In more formal terms, this paper seeks to answer the following question: *Is it possible to use ideas from multilevel graph coarsening to improve the quality of the ILU factorization?* In other words, we wish to incorporate multilevel coarsening ideas within a (single level) ILU framework to derive robust and effective preconditioners?

The paper is organized as follows: In Section 2, we briefly introduce the graph coarsening idea and discuss the multilevel framework. Section 3 presents the different strategies for performing the C/F splitting for the multilevel algorithm, and the full multilevel graph coarsening algorithm is presented in Section 4. In Section 5, we describe the final reordering and discuss its adaptation to the ILU factorization. Section 6 gives a description of the model problems used in this paper. Numerical results are presented in Section 7, and we conclude in Section 8.

2 Graph Coarsening Strategies

Multilevel methods, such as multigrid [35, 54, 63] or Schur-based multilevel techniques [3, 5, 21, 60], rely on graph coarsening strategies to construct the next level matrix in the multilevel process. For multigrid, this corresponds to selecting a subset of the original or fine grid, known as the 'coarse grid', for which the solution to the residual equation $Ae = r$, for some error e and residual r, is slow to converge, see for instance [16]. For multilevel Schur-based methods, such as multilevel ILU, the coarsening strategy may correspond to selecting from the adjacency graph of the original matrix, a subset of nodes that form an independent set [60], or a subset of nodes that satisfy good diagonal dominance properties [59], or that limit growth in the inverse LU factors of the ILU factorization [7, 8].

The general idea of coarsening is as follows. Given a graph with n vertices, we would like to find a smaller graph, typically of size about $n/2$, which yields a good representation of the original graph. Suppose we have an adjacency graph $G = (V, E)$ of some matrix A. For simplicity, we assume that G is undirected. An edge, e_{ij} represents the binary relation $(i, j) : a_{ij} \neq 0$. Coarsening the graph consists of finding a 'coarse' approximation $(\hat{V}, \hat{E})$ so that $|\hat{V}| < |V|$ and such that the new graph is a faithful representation of the original graph in some sense. By recursively coarsening the original graph, we obtain a hierarchy of approximations to the original graph. There are several techniques available for coarsening a graph. The work in this paper utilizes an aggregation-based approach, which is described next.

2.1 Pairwise Aggregation

Aggregation-based techniques are a common strategy for coarsening, see for instance [21, 46, 48, 63]. The pairwise aggregation strategy seeks to simply coalesce two adjacent nodes in a graph into a single node, based on some measure of nearness. The technique is based on edge collapsing [36], which is a well known method in the multilevel graph partitioning literature. In this method, the collapsing edges are usually selected using the *maximal matching* method. A *matching* of a graph $G = (V, E)$ is a set of edges $\tilde{E}$, $\tilde{E} \subseteq E$, such that no two edges in $\tilde{E}$ have a vertex in common. A maximal matching is a matching that cannot be augmented by additional edges to obtain another matching. There are a number of ways to find a maximal matching for coarsening a graph. We use a simple greedy strategy similar to the *heavy-edge matching* approach proposed in [37]. In algebraic terms, this greedy matching algorithm simply matches a node i, with its largest (off-diagonal) neighbor $p(i)$, i.e., we have $|a_{i,p(i)}| = \max_j |a_{ij}|$. If $p(i) = p(k)$ for some other node $k \neq i$, then node i is left unmatched and considered a singleton. Once matching is done, each pair of matched nodes $(i, p(i))$ are coalesced into a new coarse node. When two nodes i, and j are coalesced into a new one, the new node is considered the parent of the two nodes i and j, and denoted by $par(i, j)$. As in standard agglomeration techniques, the parent node is represented by one of the nodes that combine to form the parent. That is, $par(i, j) = i$ or j. Singletons represent themselves as parents. Figure 1 gives an illustration of a single coarsening step. Nodes i and j are coalesced into node $par(i, j)$ which will represent both nodes in the coarse-level graph. Moreover, a singleton node will be simply mapped to itself.

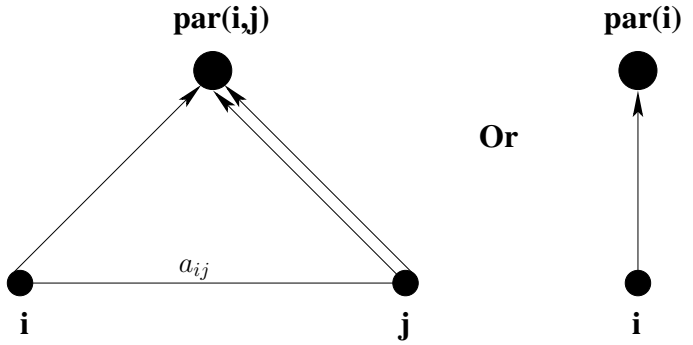


Figure 1: The coarsening process. Left: i and j are coalesced into node $par(i, j)$ and the double-arrow shows that $par(i, j)$ is represented by j; Right: singleton i is mapped into itself.

2.2 Multilevel Coarsening Scheme

The coarsening strategy can be expanded into a multilevel framework by repeating the process described above on the graph associated with the nodes in the coarse set. Let $G_\ell = (V_\ell, E_\ell)$ and define G_0 to be the original graph G and $G_1, G_2, \ldots, G_m$ be a sequence of coarse graphs such that G_ℓ is obtained by coarsening on $G_{\ell-1}$ for $1 \leq \ell \leq m$. Let $A_0 = A$ and A_ℓ be the adjacency matrix associated with graph G_ℓ. As previously mentioned, G_ℓ admits a splitting into coarse nodes, C_ℓ, and fine nodes, F_ℓ, so that the matrix A_ℓ can be reordered in one of the following ways:

$$P_\ell A_\ell P_\ell^T = \begin{bmatrix} A_{C_\ell C_\ell} & A_{C_\ell F_\ell} \\ A_{F_\ell C_\ell} & A_{F_\ell F_\ell} \end{bmatrix} \tag{1}$$

or,

$$P_\ell A_\ell P_\ell^T = \begin{bmatrix} A_{F_\ell F_\ell} & A_{F_\ell C_\ell} \\ A_{C_\ell F_\ell} & A_{C_\ell C_\ell} \end{bmatrix} \tag{2}$$

where P_ℓ is some permutation matrix. We refer to the first approach as the *Bottom-Up approach*, since we keep putting the next-level nodes (i.e. the coarse nodes) in the upper-left block. The second approach is referred to as the *Top-Down approach* in which the next-level nodes are put in the lower-right block. In both schemes, the next-level matrix is defined as $A_{\ell+1} = A_{C_\ell C_\ell}$ and its adjacency graph, $G_{\ell+1}$, is constructed from the coarse set C_ℓ, associated with the graph G_ℓ.

To construct $G_{\ell+1}$, we need to create edges, along with edge-weights, in certain ways between two coarse nodes created during the coarsening process. There are several ways of achieving this. It helps to introduce additional notation for this purpose. Referring to the left side of Figure 1, if $t = par(i, j)$ is a node in G_{l+1} we will denote the representative node j in G_l by $r(t)$ (representative child) and the other node, i, by $c(t)$ (child). This defines two mappings $r(.)$ and $c(.)$ from G_{l+1} to G_l. In the figure we have $c(t) = i$ and $r(t) = j$ for the case on the left and $r(t) = i$ and $c(t) = i$ for the case on the right.

One way to define edges at the next level is to set two coarse nodes to be adjacent in $G_{\ell+1}$ if they are adjacent in G_ℓ. With the notation just defined, this means that for any $x, t \in V_{l+1}$:

$$(t, x) \in E_{l+1} \quad \text{iff} \quad (r(t), r(x)) \in E_l.$$

This may be considered as a *one-sided* approach, since a fine node $c(t)$ in F_ℓ, does not contribute edges to the adjacency graph of its parent. The edge-weights for the new graph, $G_{\ell+1}$, using this approach, could be taken directly from the entries of the adjacency matrix of the coarse nodes in the previous graph, G_ℓ. In this case, the permutation matrices, P_ℓ, in Equations (1) and (2) are simply made up of columns of the identity matrix. Although this way of creating edges in the new coarse graph is cheap and easy to implement, it could lead to coarse graphs that are poor approximations of the original graph as the coarsening scheme progresses. Since coarsening may also be seen as a way of 'breaking' connections in the fine level graph, it is easy to see how this one-sided approach could lead to a graph of singletons, after a few coarsening steps. As a result, this approach is suitable only when a few levels of coarsening is sufficient for reordering the nodes.

An alternative to this one-sided approach is to define two coarse nodes to be adjacent in $G_{\ell+1}$ if they are parents of adjacent nodes in G_ℓ. With the notation introduced above this means that for any $x, t \in V_{l+1}$:

$$(t, x) \in E_{l+1} \quad \text{iff} \quad (r(t), r(x)) \in E_l \text{ or } (c(t), r(x)) \in E_l \text{ or } (c(t), c(x)) \in E_l.$$

This way of defining edges in the coarse graph is quite common in multilevel graph partitioning techniques, see for instance [37], and other aggregation-based methods, see for instance [48,63]. It may be considered as a *two-sided* approach, since, unlike the previous approach, the fine nodes in F_ℓ do contribute edges to the adjacency graph of their parents. Figure 2 gives an illustration of the approach. The edge-weights of the new coarse graph is typically defined as the sum of the weights of the edges in the fine graph that connects their children, or some weighted average of this sum, see for instance [46, 48, 63].

3 Preselection Techniques

The coarsening strategy described in Section 2 relies on the strength of the connections between the nodes of the graph. Each node is compared to its largest off-diagonal neighbor (or most strongly connected neighbor)

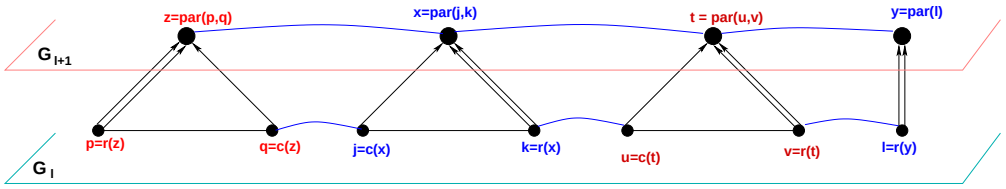


Figure 2: Two-sided approach for building the next-level coarsened graph

with respect to some defined weights. Which candidate nodes are labeled 'coarse' and which ones are labeled 'fine' during the coarsening process depends on their corresponding values in some weight vector, $\mathbf{w}$. These weights are chosen to capture some intrinsic features of the matrix, or its adjacency graph, at the current level. In this paper, we focus on diagonal dominance as a criterion for defining the weights in $\mathbf{w}$, since we wish to apply some form of incomplete factorization on the resulting reordered matrix. The goal is to obtain a reordering of the matrix such that rows (or columns) corresponding to nodes that satisfy some diagonal dominance criterion are ordered first (in the top-left corner of Equations (1) and (2)). Next, we present two different strategies of this type, and in what follows, we use $adj(i)$ to denote the adjacency (or neighbor) set of node i.

3.1 Strong Neighbor Connection Ratio

In this approach, weights are defined based on the relative size of the diagonal entry and its corresponding largest off-diagonal neighbor. We define the term γ_i as:

$$\gamma_i = \frac{|a_{ij_{max}}|}{|a_{ii}| + |a_{ij_{max}}|} \quad \text{with} \quad |a_{ij_{max}}| = \max_{j \in adj(i), j \neq i} |a_{ij}|.$$

Note that the matched pair (i, j_{max}) will be coalesced together during the coarsening process. One can think of γ_i as a measure of the degree of dependence of node i on node j_{max}. If γ_i is large (close to 1), then node i strongly depends on node j_{max}. On the other hand, if γ_i is small, then the dependence of node i on node j_{max} is only a weak one. This notion of dependence is similar to that of 'strength of connection', which is commonly used when defining coarse grid operators in algebraic multigrid, see for instance [12, 16, 21, 54, 63]. The weights, w_i, are then formally defined as:

$$w_i = \gamma_i \times (1 + |adj(i)|)$$

The quantity $(1 + |adj(i)|)$ represents the number of non-zero entries in row i of the adjacency matrix (including the diagonal entry). It is used here to scale γ_i so that nodes with fewer non-zeros have a smaller entry in $\mathbf{w}$, and hence are more likely to be ordered first in the reordered system. This serves the goal of exploiting structure in the reordering to reduce fill-in during the factorization of the reordered system.

The decision as to which node from the matched pair, (i, j_{max}), goes into the coarse or fine set, depends on the relative size of their weights, w_i and $w_{j_{max}}$, and on the direction of coarsening. For the bottom-up approach (see Equation (1)), at level ℓ of the multilevel process, the node with a smaller weight can yield a more favorable pivot for the subsequent incomplete factorization. Hence it will be a good candidate for the coarse set, C_ℓ. Conversely, for the top-down approach (see Equation (2)), the node with a larger weight will be a good candidate for the coarse set. If $w_i = w_{j_{max}}$, then either node is put in C_ℓ and the other in F_ℓ.

3.2 Relaxation

Another way to define the components of $\mathbf{w}$ for coarsening is to use relaxation sweeps in a way similar to multigrid. Relaxations are a common ingredient used in multigrid methods to damp out oscillatory errors in the residual equation. More recently, a technique known as compatible relaxation [4, 11, 13, 30, 40] has been used to select candidates for the coarse grid in algebraic multigrid (AMG). In this approach, relaxations are done on the homogeneous equation $A_\ell e_\ell = 0$, where e_ℓ is initially chosen to be in the near null space of A_ℓ. After a few relaxation sweeps, nodes that are quick to converge are put in F_ℓ. The remaining nodes (that

are slow to converge) are judged good candidates for the coarse set and put in some set U_ℓ. An independent subset of these candidates are then selected and put in C_ℓ, and the process is repeated on the remaining candidates, $U_\ell \setminus C_\ell$, until a C/F splitting of the nodes is obtained. Notice that nodes in the F_ℓ set represent rows that are 'good' in terms of diagonal dominance with respect to the associated adjacency matrix A_ℓ.

In using relaxations to form $\mathbf{w}$, we follow an approach similar to the compatible relaxation technique described above. We define γ as the vector resulting from applying a few relaxation sweeps to solve the linear system $A\gamma = 0$. Following [40], we then define the components of $\mathbf{w}$ as:

$$w_i = \frac{|\gamma_i|}{\|\gamma\|_\infty}.$$

The weight vector $\mathbf{w}$ now contains a measure of the relative convergence of all the nodes, with respect to the slowest converging node. For the top-down approach, if w_i is small (close to zero), then node i is put in F, otherwise node i is a good candidate for the coarse set. For the bottom-up approach, however, small weights represent good candidates for the coarse set.

Recall that relaxation schemes, such as Jacobi or Gauss-Seidel, are not guaranteed to converge if the matrix A is not strictly or irreducibly diagonally dominant [58]. Thus, they tend to be ineffective on numerically challenging problems, such as ones that are highly indefinite or ill-conditioned. For such problems, it is possible to transform the system matrix into one that can be handled by the relaxation scheme. In this paper, we transform the original matrix A into a new matrix L_A, that represents some graph Laplacian derived from A. Compatible relaxation schemes have been successfully used on graph Laplacian matrices as a means of defining algebraic distances between two nodes in some network graph, see for instance [20, 52]. We define the edge weights of the graph Laplacian for the matrix, A, to be based on the neighbor connection ratio as follows:

$$l_{ij} = -\frac{|a_{ij}|}{|a_{ij}| + |a_{ii}|}$$

Here, the $l_{i,j}$ represent the off-diagonal entries of the matrix L_A. This formulation represents a mapping of the edge weights onto the interval $(0, 1)$, so that entries that are small (in magnitude) in A, remain small (close to 0) in L_A, and vice versa. It is clear that if $|a_{ij}| > |a_{ii}|$, then $|l_{ij}| > 1/2$, else if $|a_{ij}| < |a_{ii}|$, then $|l_{ij}| < 1/2$. Thus the degree of dependence of node i on node j is realized in L_A as well. To complete the formation of L_A, the diagonal entries are then defined as:

$$l_{ii} = \sum_{j=1}^{n} |l_{ij}|$$

We now have a more general approach to obtain the entries in $\mathbf{w}$ via relaxations, for an arbitrary matrix. That is, instead of performing relaxations on the homogeneous equation $A\gamma = 0$, we perform them on the system $L_A\gamma = 0$. Compatible relaxation in AMG relies on the fact that the matrix, A, is non-singular, and hence the solution to the homogeneous equation converges to zero (when it converges). Notice however, that L_A is by definition, singular, and hence $L_A\gamma = 0$ does not converge to zero in general. In [20], Chen and Safro have detailed several properties about the convergence of several classical relaxation schemes applied to solve the homogeneous equation involving the graph Laplacian. Suppose the matrix L_A admits a splitting of the form $L_A = M - N$, for some matrices M and N. Then the homogeneous equation $L_A\gamma = 0$ can be re-written in the form $M\gamma = N\gamma$, yielding the iterative process:

$$\gamma^{k+1} = H\gamma^k \tag{3}$$

where $H = M^{-1}N$ is the iteration matrix. For the Jacobi relaxation method, the iteration matrix is :

$$H_{jac} = D^{-1}(E + F)$$

and for the successive over-relaxation (SOR) method, it takes for form:

$$H_{sor} = \left(\frac{D}{\omega} - E\right)^{-1} \left(\left(\frac{1}{\omega} - 1\right) D + F\right),$$

where L_A is written as $L_A = D - E - F$, in which $-F$ is the strict upper part of L_A, $-E$ its strict lower part, and D its diagonal, and where ω is the relaxation parameter for SOR. Note that $\omega = 1$ gives the Gauss-Seidel relaxation method. Assuming the graph associated with L_A is connected (and not bipartite), then H is convergent and Equation (3) converges [23]. In this case, the theory in [20] shows that $\gamma^k \to \mathbf{0}$ if the initial iterate, γ^0, is in range$(I - H)$, otherwise $\gamma^k \to \mathbf{1}$. Thus, by choosing an appropriate initial guess, the compatible relaxation scheme may be used on the graph Laplacian in a way similar to what has been described above, to define the weights in $\mathbf{w}$.

3.3 Candidate set selection

In general, all the nodes in the current graph can be considered candidates for coarsening. However, in practice, not all the nodes need to be considered. Nodes that have very small entries in $\mathbf{w}$ can be considered good enough to produce stable pivots during the ILU factorization, and hence can be immediately put in the set constituting the top-left block of the reordered system (according to the top-down or bottom-up scheme). During the preselection phase of the coarsening scheme, a node i, is selected as a candidate for coarsening if w_i is large, relative to some parameter β. Thus, changing the size of β will determine how many nodes are used for coarsening to construct the next level coarse graph.

4 The multilevel coarsening algorithm

Next we give a complete algorithm for the multilevel coarsening strategy used to reorder the system before performing the ILU factorization. The algorithm will describe the bottom-up approach, as the top-down approach follows a similar framework. The algorithm only gives a high-level description of the coarsening scheme, in order to keep it simple and more general.

Algorithm 4.1. *Multilevel Coarsening Algorithm*

1. *For $\ell = 0 : nlev$, do:*
2. *Compute $\mathbf{w}$ and form candidate set U_ℓ, for coarsening*
3. *Initialize $C_\ell = V_\ell \setminus U_\ell$*
4. *Initialize $F = \emptyset$*
5. *Obtain a C/F splitting of U_ℓ to get I_C and I_F*
6. *Set $C_\ell = C_\ell \bigcup I_C$*
7. *Set $F_\ell = F_\ell \bigcup I_F$*
8. *Build next level graph $G_{\ell+1}$*
9. *EndDo*

Algorithm 4.1 performs $nlev + 1$ levels of coarsening, starting with the original fine graph G_0. In line 2 of the algorithm, $\mathbf{w}$ is computed and a candidate set, U_ℓ, is extracted for coarsening. The nodes that do not belong to U_ℓ are assigned to the coarse set, C_ℓ (line 3), as they represent 'good' nodes that will be ordered first in the final reordering. A C/F splitting of nodes in U_ℓ is done to obtain coarse nodes, I_C, and fine nodes, I_F, which are used to update C_ℓ and F_ℓ (in lines 6 and 7). Finally, the next level graph is assembled using either the one-sided approach or the two-sided approach for defining the edge weights, discussed earlier in section 2.2.

The overall quality of the coarsening scheme is determined, in part, by the strategy used to obtain the C/F splitting in line 5. A common approach is to use a strategy that promotes independence among the nodes in the coarse set. Another approach, common in the multilevel ILU literature, focuses on promoting diagonal dominance among the nodes in the coarse set. The strategy followed in this paper uses the pairwise aggregation technique to obtain a splitting so that the nodes in the coarse set are good in terms of diagonal dominance, and are *nearly* independent. Algorithm 4.2 formally describes the technique used to obtain the C/F splitting in line 5 of Algorithm 4.1.

Algorithm 4.2. *Coarse step (C/F splitting) Algorithm*

1. *Do matching on $G_\ell = (V_\ell, E_\ell)$ to obtain pairs $(i, p(i))$*
2. *Set $mark(i) = 0$, $\forall i \in U_\ell$*

3. *For $i \in U_\ell$, do:*
4. *Set $j = p(i)$*
5. *If $mark(i) == 0$ and $mark(j) == 0$ and $i \neq j$, then*
6. *If $w_i \leq w_j$, then*
7. *Set $i \in I_C$ and $j \in I_F$*
8. *Else*
9. *Set $i \in I_F$ and $j \in I_C$*
10. *Set $mark(i) = mark(j) = 1$*
11. *EndIf*
12. *EndDo*
13. *Assign any remaining nodes to I_C or I_F*

The algorithm begins by performing a matching on the current level graph G_ℓ to obtain the pairs $(i, p(i))$, which will be used to split the candidate set, U_ℓ. This matching promotes independence among the nodes in the coarse set. Furthermore, it also dictates the structure of the resulting reordered matrix, which in turn affects the fill-in pattern of the ILU factorization applied to the matrix. Initially, all the nodes in U_ℓ are unmarked. A pair of unmarked nodes, i and j, that are matched, are compared to each other based on the relative size of their entries in $\mathbf{w}$, and assigned to the coarse set, I_C, or the fine set, I_F. These nodes are then marked, and the process is repeated until all matched nodes in U_ℓ, that are unmarked, have been marked and assigned to I_C or I_F. Finally, to complete the splitting, any remaining nodes that are not marked are assigned to the coarse or fine set. These nodes come from two different contributions. The first consists of nodes that are matched to themselves in the original graph (i.e. singletons). These nodes represent diagonal entries in the associated matrix, and hence will be good candidates for the coarse set. The other contribution comes from nodes that are matched with neighbors that have been previously marked. Note that the matching described in section 2.1 is not one-to-one. Hence a node, whose matched neighbor is involved in another matching, could be skipped by the check in line 5 of Algorithm 4.2. These nodes, which now behave like singletons, can be simply assigned to the coarse set to be dealt with at the next level of coarsening. Alternatively, they may be assigned to I_C or I_F depending on the size of their entry in $\mathbf{w}$, relative to that of their matched neighbor (albeit this neighbor is already marked). In either case, the notion of independence is compromised, as the unmarked nodes may be assigned to the same set as their matched neighbor. In the former case, independence could be recaptured at the next level as the matched pair remain neighbors at the next level. In the latter case, we obtain a C/F splitting that adheres more strictly to the requirement that 'good' nodes be assigned to the coarse set. The result is a trade-off between the diagonal dominance and the independence of the nodes in the coarse set.

5 Final reordering and ILU factorization

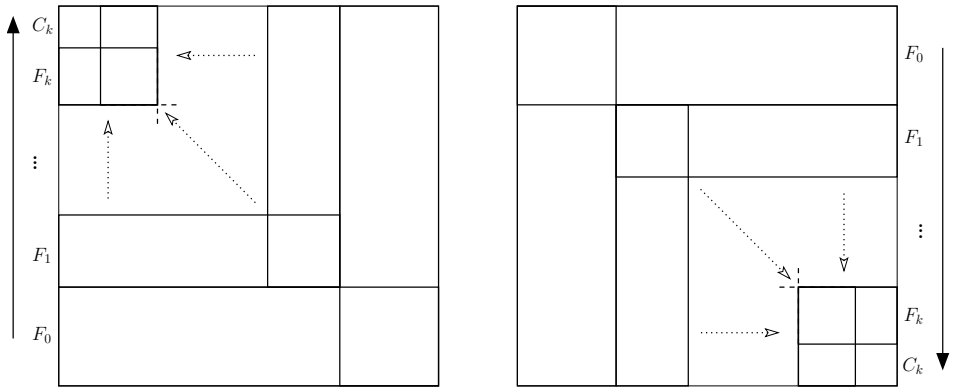
5.1 Final reordering

Using Algorithm 4.1, after each step of coarsening we obtain a C/F splitting of the current graph G_ℓ, into the sets C_ℓ and F_ℓ. For the bottom-up approach described above, after k steps of coarsening, we obtain the following sequence of sets: $C_k, F_k, \ldots, F_0$. This sequence represents the final reordering of the system matrix. Nodes in C_k will be ordered first, since they represent the 'best' nodes selected by the coarsening strategy. This will be followed by nodes in F_k, then F_{k-1} and so on, until F_0, which contains the nodes deemed to be 'worse' by the coarsening strategy. Considering each of these sets as a single level, we obtain $k + 1$ levels in the final reordered matrix.

For the top-down approach, the order of the sequence is reversed, since coarsening is done on the 'worst' nodes. Thus, nodes in F_0 represent the 'best' nodes and will be ordered first, and nodes in C_k will be ordered last. Figure 3 shows an illustration of the final reordering for the different coarsening approaches.

5.2 Adaptation to ILU factorization

Once the matrix has been reordered, an incomplete LU factorization is performed. In this paper, we consider the column-based implementation of two ILU methods: the incomplete LU factorization with level of fill



(a) Bottom-up approach (b) Top-down approach

Figure 3: Final reordering of the original matrix after k steps of coarsening using the bottom-up approach (left) and top-down approach (right).

dropping (ILU(k)); and the incomplete LU factorization with dual threshold dropping (ILUT) [58]. The ILU factorization is an approximation to the Gaussian elimination process, where small non-zero entries generated during the factorization, are dropped by some dropping criterion. During the factorization process, new non-zero entries known as 'fill-ins' may be generated at locations occupied by zero elements in the original matrix. For the ILU(k) factorization, a 'level-of-fill' parameter, k, is used to control the amount of fill-ins allowed in the L and U factors [66]. For $k = 0$, the ILU factorization drops all fill-in elements generated during the factorization process. The factors, L and U, generated as a result, will have the same pattern as the lower and upper triangular parts of A respectively. The level-of-fill concept associates each fill-in element with the 'level' at which it was formed. Initially, all non-zero entries of the matrix are assigned level zero. Then level k fill-ins are generated by products of fill-in elements at levels less than k. The ILU(k) factorization results from keeping all fill-ins that have level k or less, and dropping any fill-in whose level is higher. For the ILUT factorization, the strategy for dropping small terms is based on two parameters [56]. The first parameter is a drop tolerance, τ, which is used mainly to avoid performing an elimination if the pivot is too small. The second parameter is an integer, p, which controls the number of entries that are kept in the j-th columns of L and U. Further details of the column-based implementation of the ILUT factorization can be found in [41].

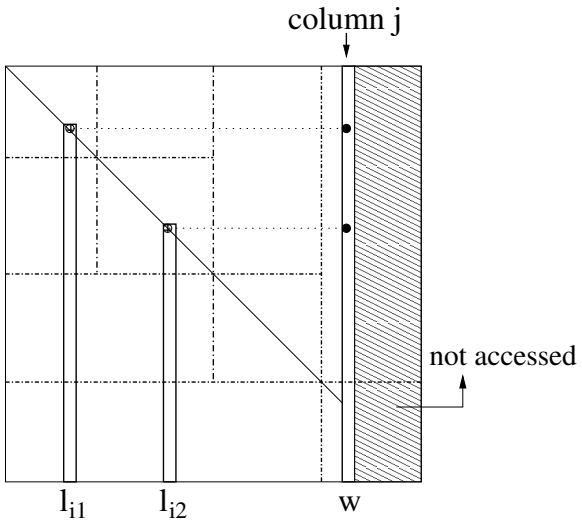


Figure 4: Column-based ILUT on reordered matrix with four levels of coarsening (bottom-up scheme).

The standard ILUT algorithm can be modified to take advantage of the reordering derived from the coarsening strategies discussed above. One simple approach is to use dropping strategies that are local to each level of the reordered matrix. In other words, if the coarsening strategy yields $\ell + 1$ levels for the reordered matrix, then the drop tolerance, τ, and the fill-level parameter, p, will each take the form of a vector of length $\ell + 1$, so that $\tau(0)$ and $p(0)$ dictate the dropping rules for the factorization of the level-zero block of columns, and so on. An illustration of the elimination process is shown in Figure 4, and a sketch of the general structure of the algorithm is given next as Algorithm 5.1. Here, we assume that **lev** is a vector of length $\ell + 1$, containing the block size (number of columns) for each level.

Algorithm 5.1. *Left-looking or JKI ordered ILUT*

0. Set $bsize = \mathbf{lev}(0)$
1. Set $\ell = 0$
2. For $j = 1 : n$, *do:*
3. $\mathbf{w} = A_{1:n,j}$
4. If $j > bsize$
5. $\ell = \ell + 1$
6. $bsize = bsize + \mathbf{lev}(\ell)$
7. EndIf
8. For $k = 1 : j - 1$ *and if* $w_k \neq 0$, *do:*
9. Apply first dropping rule to w_k *using* $\tau(\ell)$
10. If w_k *is not dropped,* $\mathbf{w}_{k+1:n} = \mathbf{w}_{k+1:n} - w_k \cdot L_{k+1:n,k}$
11. Enddo
12. For $i = j + 1, \ldots, n$, $l_{i,j} = w_i / w_j$ $(l_{j,j} = 1)$
13. Apply second dropping rule to $L_{j+1:n,j}$ *using* $\tau(\ell)$ *and* $p(\ell)$
14. For $i = 1, \ldots, j$, $u_{i,j} = w_i$
15. Apply second dropping rule to $U_{1:j-1,j}$ *using* $\tau(\ell)$ *and* $p(\ell)$
16. Enddo

In the following discussion, $l_{i,k}$ and $u_{i,k}$ represent the scalar entries at the i-th row and k-th column of the matrices L and U, respectively, $A_{1:n,j}$ denotes the j-th column of A, $\mathbf{w}_{k+1:n}$ denotes the last $n - k$ entries in the vector $\mathbf{w}$, $L_{k+1:n,k}$ denotes the last $n - k$ entries in the k-th column of L, $U_{1:j-1,j}$ denotes the first $j - 1$ entries in the j-th column of U, and so forth. The algorithm begins by setting the block size parameter, $bsize$, to the size of the first level block. This block contains the columns (or rows) associated with the nodes adjudged to be the 'best' according to the coarsening strategy. At the start of a given step j, a working column, $\mathbf{w}$, is created and set to the initial j-th column of A, $\mathbf{a}_j$, on which elimination operations will be performed. The index j is then compared to $bsize$ to determine which parameters are to be used for the elimination. The level counter ℓ, and $bsize$ are then updated accordingly (lines 5 and 6). At the beginning of the elimination process (line 9), the pivot, w_k, is dropped if it is smaller relative to some scaling parameter based on $\tau(\ell)$. Otherwise, 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. Here, $\mathbf{l}_k$ is used to denote the k-th column of L. The pivot entries, w_k, that are not dropped by the first dropping rule, will become the entries in the j-th column of U (line 14), while the entries below the diagonal of the updated vector $\mathbf{w}$, will be scaled by the diagonal entry, and become the j-th column of L (line 12). In Lines 13 and 15, the threshold, $\tau(\ell)$, is invoked again to drop small terms, then the largest $p(\ell)$ entries in the resulting i-th columns of L and U are kept, and the others are dropped.

By adopting a level-based dropping strategy, we have better control of how to drop small terms during the factorization at the different levels. Like standard ILUT, from a practical standpoint, it is often better to keep $p(\ell)$ large and vary $\tau(\ell)$ to control the dropping. One simple approach is to choose $\tau(0)$ to be relatively large for the first block, and (progressively) smaller for the subsequent blocks during the factorization. This is justified since the first block (or first set of blocks) contains nodes that are considered to have good diagonal dominance. Thus dropping during the factorization of the columns associated with these nodes could be done in a more aggressive manner. At later levels, the columns are considered to be 'poor' in terms of diagonal dominance, and a more accurate factorization may be needed to yield accurate and stable factors.

A similar approach may be used for the level-based ILU(k) method. During the factorization, the level-of-fill parameter k, can be (progressively) increased at each subsequent level from the first level. Like the

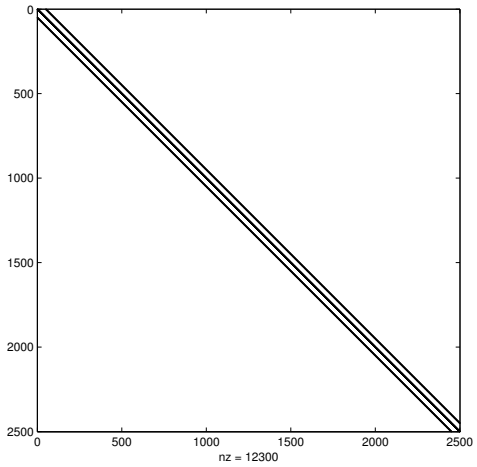


Figure 5: Nonzero pattern of the 2D Laplacian matrix.

ILUT method, this offers better control of how terms are dropped during the factorization.

Figure 5 shows the nonzero pattern of the 2D Laplacian matrix originating from the finite difference discretization of the Laplace operator on a 50×50 rectangular grid, with homogeneous Dirichlet boundary conditions. To illustrate the effect of the reordering strategy, Figure 6 shows the nonzero pattern of the same matrix reordered by the top-down version of the multilevel graph coarsening scheme. The figure depicts an example of the pattern of the reordered matrix, and the corresponding pattern of $(L+U)$ from the resulting ILU factorization, after 3, 4, and 5 levels of coarsening. For each of the reordered matrices, the number of non-zeros in $(L+U)$ is approximately 2.5 times the number of non-zeros of the original matrix.

5.3 Local reordering within the blocks

Given the level structure of the multilevel graph coarsening approach, a natural extension is the possibility of imposing some reordering within the block structure. For example, one could utilize a fill reducing technique such as RCM within each or some of the blocks in the multilevel hierarchy. This can help to further reduce fill-ins introduced during the ILU factorization, particularly when a high fill level is required for convergence. A typical scenario is when the candidate set selection phase (line 2 of Algorithm 4.1) selects only a small subset of nodes on which the coarsening is done. The resulting top- left block of the reordered matrix can be quite large and ordered in a way that is similar to that of the original matrix. Thus, reordering the nodes corresponding to this block by a fill reducing strategy like RCM can lead to a reduction in the amount of fill-ins that are discarded, and potentially yield a more accurate ILU factorization. Note that in general, at each level, the coarsening strategy produces a C/F splitting of roughly equal sizes. However, when the next coarse level graph is relatively small, then performing a local reordering is no longer necessary or beneficial. Hence it is sensible to utilize local reordering within each level when only a few levels of coarsening are performed, i.e. when $nlev$ in Algorithm 4.1 is small.

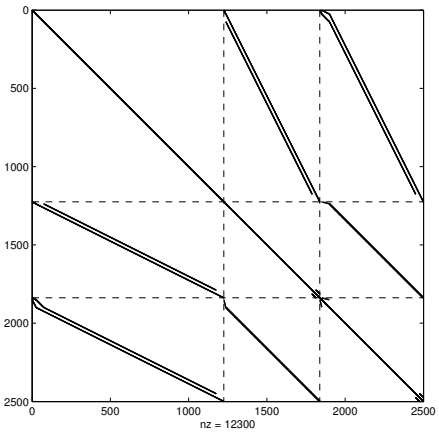
Incorporating local reordering requires only a minor modification to the multilevel graph coarsening algorithm. We insert the line

7a. Reorder F_ℓ

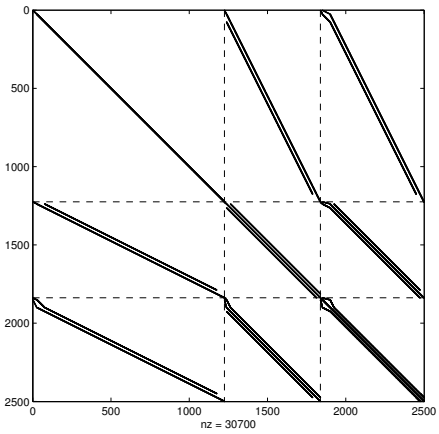
after Line 7 of Algorithm 4.1 to reorder the fine set at each level and the line:

9a. Reorder final coarse set C_{nlev}

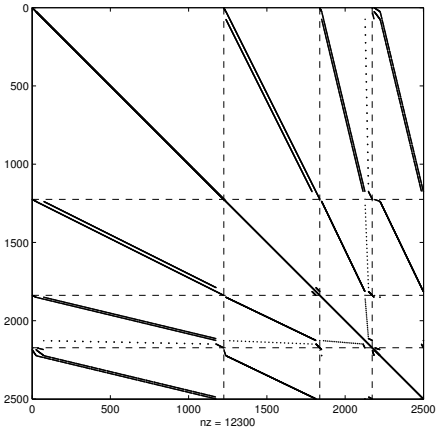
after Line 9 of Algorithm 4.1 to reorder the final coarse set. We refer to this algorithm as the Multilevel Coarsening Algorithm with Local Reordering or 'Algorithm 4.1 with reordering.'



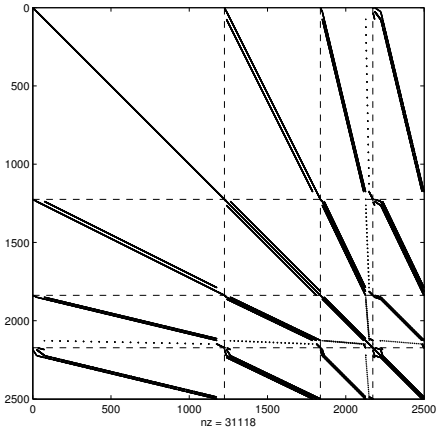
(a) Two levels of coarsening



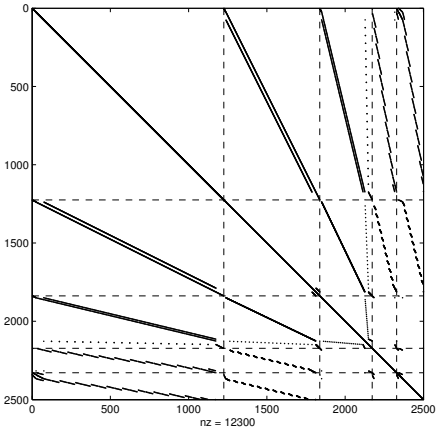
(b) Pattern of $(L+U)$



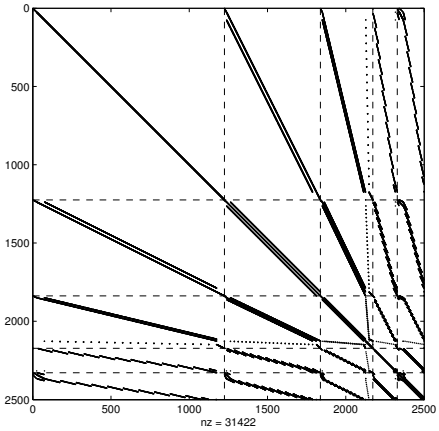
(c) Three levels of coarsening



(d) Pattern of $(L+U)$



(e) Four levels of coarsening



(f) Pattern of $(L+U)$

Figure 6: An example of the nonzero pattern of the reordered matrix and the resulting ILU factorization for the 2D Laplacian.

6 Test Problems

We present some numerical results using the new reordering strategy on various application problems. These problems are either indefinite, or exhibit numerical anisotropy that is derived from either the discretization grid or the equation coefficients. Details of the model problems are given next.

6.1 Problem I: 2D Helmholtz equation application

The Helmholtz equation is a partial differential equation of the form

$$(\Delta + k^2)u = f, \tag{4}$$

which describes the propagation of waves in media. In the above equation, f represents a harmonic source, and k represents the wave number. The numerical solution to the Helmholtz equation at high wave numbers has been the subject of extensive research. At high wave numbers, the system matrix tends to be very indefinite, causing problems for many numerical methods. Our application problem is based on the simulation of the diffraction of an acoustic wave originating from infinity through an open medium, and incident on a bounded obstacle with a circular boundary. Here, we assume the plane wave is propagating along the x-axis, and the radius of the bounded obstacle is 0.5m. For a suitable numerical solution to the problem by the finite element method, an artificial boundary condition is imposed at a distance 1.5m from the obstacle, using the Dirichlet-to-Neumann technique to satisfy the Sommerfeld radiation condition [39]. The resulting boundary value problem is discretized by the Galerkin least-squares finite-element method, using an isoparametric discretization over quadrilateral elements. The discretized problem is complex symmetric indefinite, and has size $n = 57,960$, with $nnz = 516,600$ nonzero elements.

6.2 Problem II: 2D elliptic PDE

We consider a two-dimensional elliptic partial differential equation of the form

$$-a\frac{\partial^2 u}{\partial x^2} - b\frac{\partial^2 u}{\partial y^2} - \rho u = -\left(bx^2 + ay^2 + \rho\right) e^{xy} \text{ in } \Omega, \tag{5}$$

subject to the Dirichlet boundary condition $u = e^{xy}$ on $\partial\Omega$, where $\Omega = (0,1) \times (0,1)$. The problem is discretized using a 5-point centered difference approximation on an $n_x \times n_y$ grid. The resulting coefficient matrix represents the discretization of a negative Laplacian, shifted by $\rho h^2 I$, where I is the identity matrix and the same mesh size $h = \frac{1}{n_x - 1}$ is used in both x and y directions. We take a grid of size $n_x = n_y = 258$, yielding a coefficient matrix is of size $n = 65,536$, and also $\rho = 845$, yielding a corresponding shift is $0.05I$. We set the coefficients $a = 20$ and $b = 1$ so that the resulting problem is anisotropic and symmetric indefinite.

6.3 Problem III: 3D elliptic (Helmholtz-type) equation

We consider a three-dimensional elliptic partial differential equation of the form

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} - \frac{\partial^2 u}{\partial z^2} - \rho u = -6 - \rho\left(x^2 + y^2 + z^2\right) \text{ in } \Omega, \tag{6}$$

subject to the Dirichlet boundary condition $u = x^2 + y^2 + z^2$ on $\partial\Omega$, where $\Omega = (0,1) \times (0,1) \times (0,1)$. Here, we use a 7-point centered difference approximation on an $n_x \times n_y \times n_z$ grid. We take $n_x = n_y = n_z = 66$ and $\rho = 126.75$, so that the coefficient matrix has size $n = 262,144$ and is shifted by $\rho h^2 I = 0.03I$. The resulting linear system matrix is symmetric and indefinite.

6.4 Problem IV: Stretched circle problem

The model problem is a triangular finite element discretization of a 2D isotropic diffusion problem, defined on the unit circle. The resulting triangulation is stretched by a factor of 100 in the y-direction, yielding a symmetric positive definite coefficient matrix that has strong unstructured anisotropic behavior. We consider two examples of sizes $n = 13,373$ and $n = 53,069$, corresponding to a grid spacing of size $h = 0.02$ and $h = 0.01$ respectively. See [61] for more details of the problem description.

6.5 Problem V: Unstructured triangles problem

We consider a tetrahedral finite element discretization of

$$0.01\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = f \tag{7}$$

on an unstructured mesh. The resulting coefficient matrix has size $n = 337,105$ and exhibits a non-grid-aligned anisotropic behavior.

6.6 Problem VI: Rotated anisotropy problem

Finally, we consider a 2D rotated anisotropic diffusion problem described earlier in the literature, see Equation 1 of [61]. The problem satisfies a biquadratic finite element discretization of

$$-(c^2 + \epsilon s^2)\frac{\partial^2 u}{\partial x^2} - 2(1-\epsilon)cs\frac{\partial^2 u}{\partial x \partial y} - (\epsilon c^2 + s^2)\frac{\partial^2 u}{\partial y^2} = f, \tag{8}$$

where $\epsilon = 0.001$, and c and s represent the cosine and sine of the angle of rotation, θ, respectively. For this model problem, we consider two examples, each of size $n = 261,121$, and corresponding to $\theta = \pi/16$ and $\theta = 2\pi/16$.

7 Numerical Results

The experiments were conducted sequentially on a single core of a 64-bit Linux cluster, with sixteen 2.60GHz Intel Xeon processor cores per node, and 20MB cache and 32GB memory per node. The program was compiled by the gcc compiler using -O3 optimization level. In the following experimental results, we test the effect of reordering by the multilevel graph coarsening strategy (MLGC) described above, and compare its performance against those of the natural (or original) ordering obtained from the discretization, as well as standard reordering techniques for ILU. In particular, we compare results against those of the reverse Cuthill-McKee algorithm (RCM), the multiple minimum degree algorithm (MMD), and the nested dissection algorithm (ND). We use the version of these algorithms as implemented in METIS [38].

The numerical solution for each example uses a right-preconditioned restarted GMRES [55] accelerator, with a restart dimension of 100. The maximum number of GMRES iterations is fixed at 300, and unless otherwise stated, we assume convergence when the ℓ_2-norm of the initial residual is reduced by a relative factor of 10^8. The right-hand side vector is artificially generated by assuming a solution of all ones, and we use a zero initial guess for the iterative solver.

7.1 Helmholtz equation application

In the results that follow, the MLGC strategy uses the top-down direction of coarsening, the strongest neighbor connection ratio for the preselection strategy, and the two-sided approach for computing the weights for the coarse level graph. The tolerance, β, used for the candidate set selection is set to 0.1 for each level. First, we compare the performance of the different ordering schemes on the Helmholtz problem with wavenumber $k = 2\pi$. The problem is slightly indefinite with a mesh resolution of 160 points per wavelength. Figure 7 shows the convergence profiles for standard ILUT with the natural ordering, and ILUT with RCM, ND, MMD, and the MLGC orderings. Here, the MLGC strategy does 2 steps of coarsening. The memory usage (or fill factor), for each of the preconditioners, is ≈ 2.5. This is defined as $(nnz(L+U) - n)/nnz(A)$, which is the ratio of the number of nonzero elements needed to store the L and U factors over the original number of nonzero entries of A. As shown in the figure, the MLGC ordering yields the best convergence performance. This result suggests that the performance of ILUT on the Helmholtz problem can be significantly improved by reordering the system matrix by the MLGC scheme, prior to performing the ILUT factorization. To see whether this holds true for higher values of the wavenumber, we apply the approach to solve a number of systems corresponding to increasing values of the wavenumber.

As the wavenumber increases, the Helmholtz problem becomes more indefinite, and hence more challenging to solve by standard ILUT. As shown in previous work, see for instance [39, 41, 51], standard ILUT

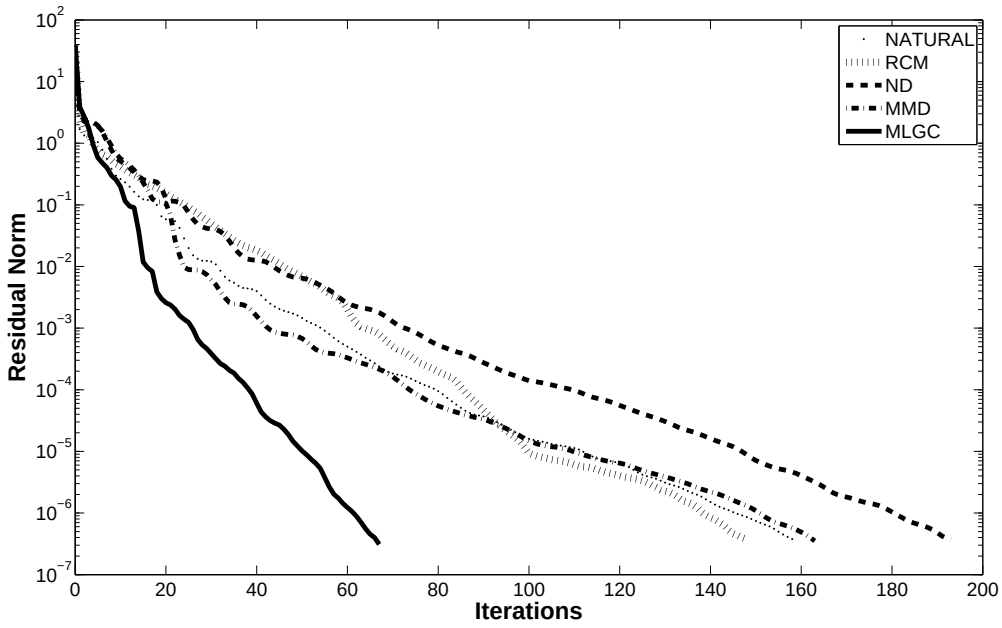


Figure 7: ILUT with different orderings on Helmholtz problem.

applied to this problem fails to converge at high wave numbers. As a result, each preconditioner, in the results that follow, is obtained by performing shifted ILUT on the (reordered) matrix of the discretized problem. The shifted ILUT method used here is based on the τ-based scheme (i.e. shifting based on the drop tolerance) described earlier in the literature [51].

Table 1 presents results for solving the linear system resulting from the Helmholtz problem, for different wave numbers k. The table shows results for the total number of iterations, and the total time taken to construct the preconditioner and solve the linear system. The memory usage (or fill factor) is kept at ≈ 3.5 for each preconditioner, to allow for a fair comparison.

The results from the table indicate that at lower values of the wavenumber, shifted ILUT with natural ordering performs better than shifted ILUT with RCM, ND, or MMD ordering. At higher wave numbers, when the system is very indefinite, the RCM ordering appears to perform better than the ND, MMD, and natural orderings. However, shifted ILUT with the MLGC ordering is by far the winner. It exhibits superior performance over the other ordering strategies for all the different values of the wavenumber. It requires fewer iterations to converge, and it converges in very little time compared to the rest. We note here that the time to compute each preconditioner (including the time to perform the reordering) is similar (under 1 second) for the different methods, and hence the total time presented in the table gives a good indication of the solution time, and correlates with the number of iterations required for convergence.

The above results do not utilize any local reordering within the blocks for the MLGC scheme. In order to see the potential benefit of local reordering within the blocks, we test the MLGC scheme on a number of different problems. The test problems considered are three anisotropic examples: Problem II, Problem IV with a mesh size of 0.01, and Problem VI with a rotation of 2π. We also include two indefinite (and isotropic) examples: Problem I with a wave number of 4π, and Problem III. Each example is first run with the MLGC strategy without any local reordering within the blocks, and the results recorded. The problem is then run again with the same parameters, only this time, we perform local reordering with RCM within each block. For the tests with Problem IV, we use the bottom-up direction of coarsening and the compatible relaxation strategy, with 5 iterations of Gauss-Siedel, to determine the weights for the preselection phase of the algorithm (see line 2 of Algorithm 4.1 with or without local reordering). For the remaining examples, we use the top-down direction of coarsening, and the strongest neighbor connection ratio strategy to define the weights. We use the two-sided approach to build the coarse level graph, and perform 2 coarsening steps

Ordering	k	$\frac{\lambda}{h}$	No. iters	Fill factor	total time
Natural	4π	80	157	3.49	3.14
	8π	40	170	3.47	3.39
	16π	20	191	3.51	3.90
	32π	10	123	3.51	2.63
RCM	4π	80	166	3.48	3.33
	8π	40	195	3.51	4.07
	16π	20	114	3.49	2.53
	32π	10	95	3.45	2.30
ND	4π	80	163	3.50	3.69
	8π	40	145	3.48	3.40
	16π	20	117	3.48	2.79
	32π	10	138	3.49	3.11
MMD	4π	80	162	3.49	3.54
	8π	40	143	3.47	3.16
	16π	20	114	3.48	2.75
	32π	10	137	3.49	3.09
MLGC	4π	80	65	3.45	1.61
	8π	40	40	3.48	1.16
	16π	20	33	3.48	1.08
	32π	10	53	3.49	1.57

Table 1: Shifted ILUT with different reorderings on the Helmholtz problem.

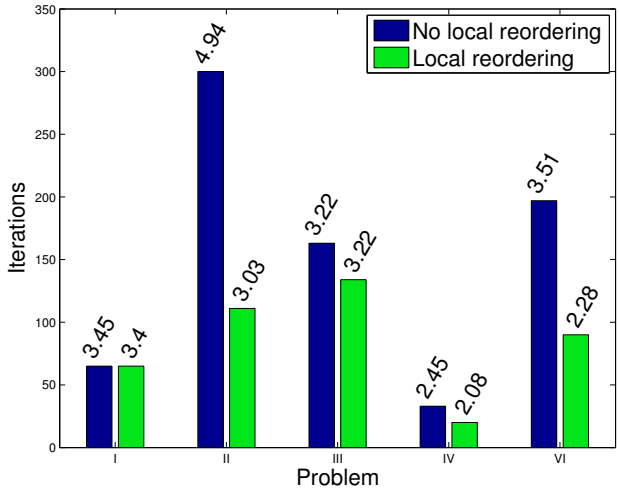


Figure 8: Effect of local reordering with RCM. The numbers on top of each bar show the fill factors.

for each example. Figure 8 gives an illustration of the results for each example, using ILUT-preconditioned GMRES.

The results indicate a general improvement in both memory and computational efficiency when local reordering is performed. For Problem II, the solution fails to converge after 300 iterations, with a relatively high fill factor, when no local reordering is done. This is significantly improved when local reordering is performed, and convergence is achieved in about 100 iterations with a much smaller fill factor. The results for Problem III only indicate a modest improvement when local reordering is performed, and the results for Problem I show no additional improvement due to local reordering. How much a problem benefits from local reordering depends on several factors such as the type of local reodering utilized, the coarsening strategy used, and the size of the local blocks. Thus, Figure 8 only illustrates the potential benefit of local reordering.

7.2 Anisotropic examples

The numerical solutions to the following problems use a restarted GMRES accelerator, coupled with the level-based ILU preconditioner, ILU(k). The right-hand-sides of the linear systems are artificially generated by assuming a solution of all ones. We allow a maximum of 300 iterations and assume convergence whenever the ℓ_2-norm of the initial residual is reduced by a relative factor of 10^{10}. The results for the MLGC strategies use 2 steps of coarsening, with local reordering by RCM. We distinguish between the preselection strategies - strongest neighbor connection ratio (MLGC-SNCR), or compatible relaxation (MLGC-CR); and we use the bottom-up direction of coarsening with the one-sided strategy to construct the coarse level graph. For the strongest neighbor connection ratio scheme, we use a tolerance of $\beta = 0.1$ for the candidate set selection at each level. For the compatible relaxation scheme, we perform 3 iterations of the Jacobi relaxation method, and we set β to 0.9 for the first level, and reduce it by a factor of 0.9 for each subsequent level. Note that this choice of β generally makes the compatible relaxation scheme more selective than the strongest neighbor connection ratio scheme. In other words, by using the bottom-up direction of coarsening for the following examples, the MLGC-CR method is more likely to yield a larger block in the top-left position of the reordered matrix ($A_{C_\ell C_\ell}$ in Equation 1), than the MLGC-SNCR method.

7.2.1 Stretched circle problem

Table 2 gives a summary of the results for the stretched circle problem, with a mesh size of 0.02, using ILU(k) with $k = (1, 2, 3)$. For all the different reordering methods, increasing the fill level parameter yields a more accurate factorization and results in better solver convergence. We observe that compared to the natural, MMD, and ND orderings, RCM performs better in terms of the iteration count to convergence, memory efficiency, and computational time. Furthermore, it appears that with the exception of the MLGC-CR method, the solution with ILU(1) was quite poor for any of the other reordering methods. The MLGC-CR method shows the best overall performance. For about the same cost in memory for constructing the preconditioner, the MLGC-CR strategy cuts the number of iterations and time to solution by over 50%, compared to RCM. The MLGC-SNCR method underperformed RCM for the solution with ILU(1). However, it shows better performance over RCM for the higher fill levels, although with a slightly larger fill factor. Note, however, that the overall time to solution for the MLGC-SNCR method is comparable to that of RCM. The results also show that the natural ordering performs better than MMD or ND, although the natural ordering generates more fill-ins, and thus, has a much larger fill factor.

Fill-reducing strategies such as RCM are known to be effective at higher fill levels, since the resulting ILU factorization becomes more accurate. The results in Table 2 indicate that even at low fill levels the MLGC-CR strategy shows good convergence. This desirable property further makes it a more attractive choice for this problem. We explore this further by considering a slightly harder problem. Figure 9 shows the convergence profiles of RCM and the MLGC-CR strategy for the stretched circle problem with a mesh size of 0.01, and using varying levels of fill. Due to the smaller mesh size and the strong anisotropy, the problem is more challenging to solve, and a relatively high fill level is needed for the iterative solver to converge. We note here that solutions with ND, MMD, and the natural orderings failed to converge in the required number of iterations, and with a reasonable fill factor. Thus we do not plot their results. For this example, the compatible relaxation scheme uses 5 Jacobi iterations. Next to each convergence profile in Figure 9, we show the memory usage (denoted ffact) for the resulting ILU preconditioner. As shown in the figure, when the MLGC strategy is used, the convergence of the iterative solver with the ILU(3) preconditioner (denoted by MLGC(3)) is superior to that of ILU(4) and ILU(5) when the RCM ordering is used. Comparing the two techniques using the same fill level for ILU shows that the MLGC strategy converges over 4 times faster for the same (after roundoff) cost in memory usage.

7.2.2 Unstructured Triangles problem

Table 3 shows results for the unstructured triangle problem. For each of the reordering strategies, a fill level parameter, $k = (1, 2)$, is used to construct the ILU preconditioner. The results indicate that the natural ordering yields the least effective preconditioner, while it generates significant fill-ins compared to the other strategies. Once again, RCM outperforms ND and MMD. However, the MLGC strategies show better convergence compared to RCM, with the MLGC-SNCR strategy performing the best. Notice,

Ordering & fill level k	Fill factor (memory usage)	Iteration count	Ordering time	Factorization time	Iteration time
Natural(1)	1.91	232	-	0.01	0.31
Natural(2)	2.68	166	-	0.03	0.23
Natural(3)	3.80	130	-	0.05	0.20
RCM(1)	1.34	178	0.01	0.01	0.21
RCM(2)	1.77	118	0.01	0.01	0.15
RCM(3)	2.29	87	0.01	0.02	0.11
ND(1)	1.62	237	0.04	0.01	0.30
ND(2)	2.12	172	0.04	0.02	0.23
ND(3)	2.59	134	0.03	0.02	0.18
MMD(1)	1.62	247	0.01	0.01	0.30
MMD(2)	2.09	175	0.01	0.02	0.22
MMD(3)	2.53	132	0.02	0.02	0.17
MLGC-SNCR(1)	1.60	192	0.01	0.01	0.24
MLGC-SNCR(2)	2.10	104	0.01	0.01	0.14
MLGC-SNCR(3)	2.59	71	0.01	0.02	0.09
MLGC-CR(1)	1.34	78	0.01	0.01	0.09
MLGC-CR(2)	1.77	50	0.01	0.01	0.06
MLGC-CR(3)	2.32	37	0.01	0.02	0.04

Table 2: Summary of results for stretched circle problem, $h = 0.02$

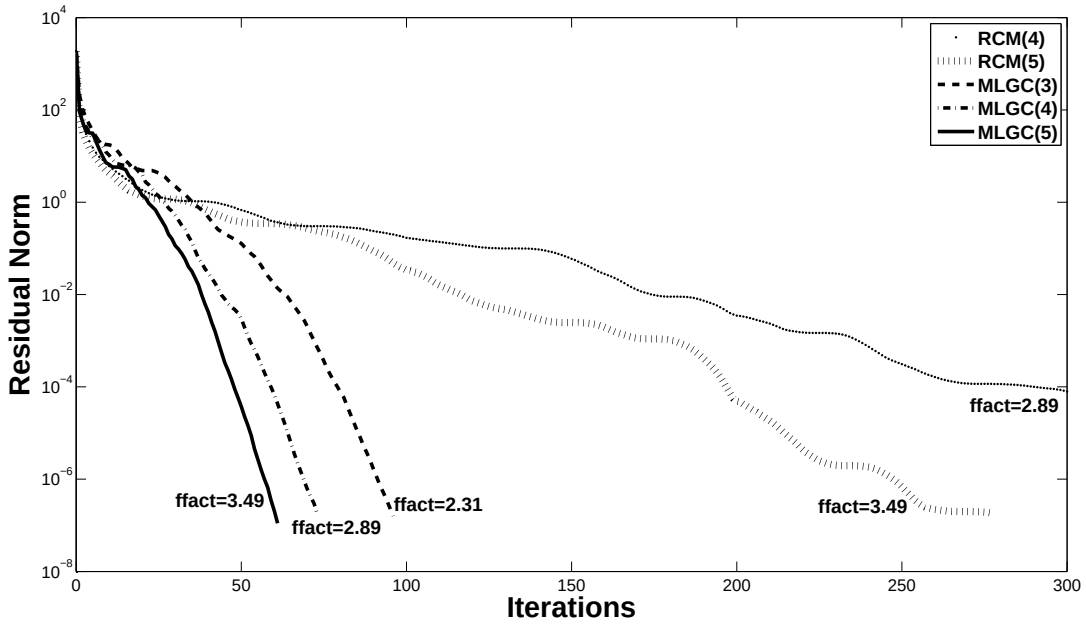


Figure 9: Convergence of MLGC and RCM for different levels of fill on the stretched circle problem, $h = 0.01$. RCM(k) (resp. MLGC(k)) denotes an ILU(k) factorization is performed on the RCM (resp. MLGC) reordered system matrix.

however, that the RCM method appears to have a slightly better overall computational time. This is to be expected considering the additional cost incurred by performing local reordering for the MLGC strategies, for a relatively modest improvement in convergence.

Ordering & fill level k	Fill factor (memory usage)	Iteration count	Ordering time	Factorization time	Iteration time
Natural(1)	4.58	80	-	3.76	6.31
Natural(2)	9.08	56	-	12.76	6.39
RCM(1)	2.06	67	0.35	1.12	3.30
RCM(2)	3.69	49	0.36	2.76	2.80
ND(1)	2.15	79	3.98	1.35	4.25
ND(2)	3.94	56	4.00	3.77	3.54
MMD(1)	2.21	77	1.35	1.57	5.29
MMD(2)	3.81	55	1.35	4.02	4.27
MLGC-SNCR(1)	1.90	62	0.45	1.11	3.57
MLGC-SNCR(2)	3.27	44	0.46	2.78	2.87
MLGC-CR(1)	2.06	65	0.98	1.12	3.17
MLGC-CR(2)	3.68	46	0.97	2.77	2.58

Table 3: Summary of results for unstructured triangle problem

Ordering & fill level k	Fill factor (memory usage)	Iteration count	Ordering time	Factorization time	Iteration time
Natural(1)	3.30	F	-	1.27	18.26
Natural(2)	7.08	7	-	5.37	0.57
RCM(1)	1.34	13	0.22	0.28	0.33
RCM(2)	1.84	10	0.20	0.45	0.29
ND(1)	1.70	43	1.95	0.85	1.50
ND(2)	2.30	28	1.95	1.36	1.03
MMD(1)	1.75	55	0.22	0.63	2.05
MMD(2)	2.08	27	0.22	0.93	0.94
MLGC-SNCR(1)	1.54	11	0.31	0.36	0.32
MLGC-SNCR(2)	2.19	8	0.31	0.67	0.28
MLGC-CR(1)	1.34	15	0.66	0.27	0.64
MLGC-CR(2)	1.84	11	0.67	0.45	0.32

Table 4: Summary of results for rotated anisotropic problem rotated by π

7.2.3 Rotated anisotropy problem

Ordering & fill level k	Fill factor (memory usage)	Iteration count	Ordering time	Factorization time	Iteration time
Natural(1)	3.30	F	-	1.25	16.43
Natural(2)	7.08	6	-	5.22	0.46
RCM(1)	1.34	31	0.22	0.28	0.87
RCM(2)	1.84	12	0.21	0.45	0.34
ND(1)	1.70	55	1.95	0.85	2.05
ND(2)	2.30	39	1.95	1.37	1.50
MMD(1)	1.75	56	0.22	0.63	2.11
MMD(2)	2.08	39	0.22	0.93	1.43
MLGC-SNCR(1)	1.64	14	0.32	0.40	0.44
MLGC-SNCR(2)	2.23	8	0.32	0.69	0.28
MLGC-CR(1)	1.34	34	0.68	0.27	0.99
MLGC-CR(2)	1.84	14	0.66	0.45	0.40

Table 5: Summary of results for rotated anisotropic problem rotated by 2π

Tables 4 and 5 give the results for the rotated anisotropy problem, with a rotation through an angle of 1π and 2π respectively. For these results, a fill level parameter, $k = (1, 2)$, is used to construct the ILU preconditioner. Once again, the results indicate that the natural ordering generates significantly more fill-ins compared to the other strategies. Furthermore, when ILU(1) is used, the natural ordering fails for both test cases. When ILU(2) is used instead, the natural ordering converges in the fewest number of iterations, but at a much higher cost than the other strategies. As in the previous example, the RCM and MLGC strategies appear to be the most efficient for this problem.

The numerical results presented above show an improvement in performance when the MLGC strategy is used, compared to the natural and other standard reordering strategies. One explanation for this comes from considering the effect of the MLGC scheme on the stability and accuracy of the ILU factorization. First, recall that the coarsening strategy compares a node to its largest neighbor (based on some weight), and assigns one to the coarse set and the other to the fine set. As a result, long recurrences during the preconditioner solve (i.e. forward solve with L and backward solve with U) involving large (neighboring) entries in L and U are potentially avoided. In other words, the norm of L^{-1} or U^{-1} is not too large, leading to more stable triangular solves. A theoretical basis of this result for the symmetric positive definite case is presented in [14]. Furthermore, the splitting of nodes into coarse or fine sets leads to a hierarchy of (block) independent sets, which can be eliminated with very little fill-ins during the factorization. Thus, few entries are dropped during the factorization, which suggests that the preconditioner obtained from the ILU factorization with this ordering should be more accurate. Moreover, since nodes that are poor in terms of diagonal dominance, (or nodes with unfavorable pivots), are relegated to be ordered last, the adverse effects of these poor nodes on the factorization is minimized. The result is that, large entries in L or U, which could lead to unstable and inaccurate factors, are avoided.

Unlike the MLGC scheme, the other reordering techniques (i.e. RCM, ND, and MMD) are based primarily on the adjacency graph of the coefficient matrix A without any consideration given to algebraic values of its entries. Thus, as indicated by the results in Figure 7 and Table 1, when the natural ordering gives an ordering that is good in terms of minimizing fill-ins, it may not be beneficial to reorder the system by an approach that is based solely on its graph.

8 Summary and Conclusion

The technique presented in this paper exploits a multilevel graph coarsening strategy to define an effective reordering for ILU-based preconditioners. The direction of coarsening for the multilevel method follows either a top-down or bottom-up approach, and two different strategies are presented for defining the splitting of nodes into coarse and fine sets. The first strategy considers the strength of connection between neighboring nodes, and the second is based on the compatible relaxation technique from algebraic multigrid. With the help of examples from the solution of the Helmholtz equation, and anisotropic problems from 2D/3D PDEs, we compared the results obtained with the proposed technique against those obtained from the natural ordering, as well as other standard reordering techniques, namely RCM, MMD and ND.

For the Helmholtz problem, the results indicate that the multilevel graph coarsening reordering strategy is superior to the natural ordering, or the ordering given by RCM, ND, or MMD. Significant gains are observed in the number of iterations required to reach convergence, and in the overall computational time, even for the highly indefinite cases. Note that at high wave numbers, reordering alone may not be enough to efficiently solve the Helmholtz problem. More robust algebraic strategies such as diagonal compensation (Shifted ILU) or modified ILU may be necessary to ensure good convergence.

The results from the anisotropic examples also indicate significant performance gains over the natural ordering. In general, comparative results show that the MLGC approach either outperformed the standard reordering strategies or gave results comparable to RCM, which in this case was the best of the standard reordering schemes.

Our results also indicate that performing some local reordering within the blocks of the MLGC scheme can be beneficial. In practice, any reordering strategy may be utilized to reorder the local blocks. In this work, we used RCM to perform local reordering for the anisotropic examples, in order to benefit from its fill-reducing property. Depending on the tolerance used in the preselection phase of the MLGC algorithm, one can control the size of the local blocks, and hence, the resulting fill factor of the ILU factorization.

We observed that when the natural ordering reduces fill-ins, it generally performs quite well compared to the standard reordering strategies. Furthermore, reorderings based solely on graph theory may be ineffective for highly indefinite problems, such as the Helmholtz problem at high wave numbers, and problems with strong (unstructured) anisotropy. In such cases, combining ideas from graph theory with algebraic information from the underlying adjacency matrix can be quite beneficial. These observations are in agreement with previous results from the literature, see e.g., [14, 15, 24]. Due to its algebraic nature, the MLGC strategy requires some parameter tuning, which makes it applicable to a wider range of problems but at the expense of added complexity. Furthermore, one can easily adapt the framework to a specific problem by defining a new C/F splitting strategy that is appropriate for that problem.

Acknowledgements

We wish to thank Kechroud, Gowda and Soulaimani for providing us with the finite-element code for acoustic wave scattering (see [39]), and J. Schroeder for providing us with the data for the anisotropic examples presented in the experiments.

References

[1] P. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM Journal on Matrix Analysis and Applications, 17 (1996), pp. 886–905.

[2] O. Axelsson, *A generalized SSOR method*, BIT, 12 (1972), pp. 443–467.

[3] O. Axelsson and P. Vassilevski, *Algebraic multilevel preconditioning methods. I*, Numer. Math., 56 (1989), pp. 157–177.

[4] R. E. Bank, *Compatible coarsening in the multigraph algorithm*, Advances in Engineering Software, 38 (2007), pp. 287–294.

[5] R. E. Bank and C. Wagner, *Multilevel ILU decomposition*, Numerische Mathematik, 82 (1999), pp. 543–576.

[6] M. Benzi, J. C. Haws, and M. Tuma, *Preconditioning highly indefinite and nonsymmetric matrices*, Tech. Report LA-UR-99-4857, CIC-19 and Los Alamos National Laboratory, Los Alamos, NM 87545, August 1999.

[7] M. Bollhöfer, *A robust ILU with pivoting based on monitoring the growth of the inverse factors*, Linear Algebra and its Applications, 338 (2001), pp. 201–213.

[8] M. Bollhöfer, M. J. Grote, and O. Schenk, *Algebraic multilevel preconditioner for the helmholtz equation in heterogeneous media*, SIAM Journal on Scientific Computing, 31 (2009), pp. 3781–3805.

[9] E. Botta, A. Ploeg, and F. Wubs, *Nested grids ILU-decomposition (NGILU)*, J. Comp. Appl. Math., 66 (1996), pp. 515–526.

[10] E. Botta and F. Wubs, *Matrix Renumbering ILU: an effective algebraic multilevel ILU*, SIAM Journal on Matrix Analysis and Applications, 20 (1999), pp. 1007–1026.

[11] A. Brandt, *Generally highly accurate algebraic coarsening*, Electronic Transactions on Numerical Analysis, 10 (2000), pp. 1–20.

[12] A. Brandt, S. F. Mc Cormick, and J. Ruge, *Algebraic multigrid (amg) for sparse matrix equations*, in Sparsity and its applications, D. J. Evans, ed., Cambridge, 1984, Cambridge Univ. Press.

[13] J. Brannick, *Compatible relaxation and coarsening in algebraic multigrid*, SIAM Journal on Scientific Computing, 32 (2010), pp. 1393–1416.

[14] R. Bridson and W.-P. Tang, *Ordering, anisotropy, and factored approximate inverses*, SIAM Journal on Scientific Computing, 21 (1999), p. 867882.

[15] ——, *A structural diagnosis of some ic orderings*, SIAM Journal on Scientific Computing, 22 (2000), p. 15271532.

[16] W. L. Briggs, V. E. Henson, and S. F. Mc Cormick, *A multigrid tutorial*, SIAM, Philadelphia, PA, 2000. Second edition.

[17] A. M. Bruaset, A. Tveito, and R. Winther, *On the stability of relaxed incomplete lu factorizations*, Mathematics of Computation, 54 (1990), pp. 701–719.

[18] N. Buleev, *A numerical method for the solution of two-dimensional and three-dimensional equations of diffusion*, Math. Sb., 51 (1960), pp. 227–238.

[19] T. Chan and H. Elman, *Fourier analysis of iterative methods for elliptic problems*, SIAM Review, 31 (1989), pp. 20–49.

[20] J. Chen and I. Safro, *Algebraic distance on graphs*, Tech. Report MCS-P1696-1009, ANL, Argonne National Laboratory, 2010.

[21] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge University Press, Cambridge, UK,, 2005.

[22] E. Chow and P. S. Vassilevski, *Multilevel block factorizations in generalized hierarchical bases*, Numerical Linear Algebra with Applications, 1 (2002), pp. 1–22.

[23] A. Dax, *The convergence of linear stationary iterative processes for solving singular unstructured systems of linear equations*, SIAM review, 34 (1990), pp. 611–635.

[24] E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, *Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), p. 944961.

[25] S. Doi and A. Hoshi, *Large numbered multicolor MILU preconditioning on SX-3/14*, Int'l J. Computer Math., 44 (1992), pp. 143–152.

[26] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, New York, 1986.

[27] I. S. Duff and J. Koster, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM Journal on Numerical Analysis, 22.

[28] 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.

[29] H. C. Elman, *A stability analysis of incomplete LU factorizations*, Mathematics of Computation, 47 (1986), pp. 191–217.

[30] R. Falgout and P. Vassilevski, *On generalizing the amg framework*, SIAM Journal on Numerical Analysis, 42 (2005), pp. 1669–1693.

[31] A. George, *Nested dissection of a regular finite element mesh*, SIAM Journal on Numerical Analysis, 10 (1973), pp. 345–363.

[32] A. George and J. Liu, *Computer solution of large sparse positive definite systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[33] ——, *Evolution of the minimum degree ordering algorithm*, SIAM review, 31 (1989), pp. 1–19.

[34] I. Gustafsson, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142–156.

[35] W. Hackbusch, *Multi-Grid Methods and Applications*, vol. 4 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 1985.

[36] Y. F. Hu and R. J. Blake, *Load balancing for unstructured mesh applications*, Nova Science Publishers, Inc., Commack, NY, USA, 2001, pp. 117–148.

[37] G. Karypis and V. Kumar, *Multilevel graph partitioning schemes*, in Proc. 24th Intern. Conf. Par. Proc., III, CRC Press, 1995, pp. 113–122.

[38] G. Karypis and V. Kumar, *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and computing Fill-Reducing Orderings of Sparse Matrices, version 4.0.1,University of Minnesota, Dept. of Comp. Sci. and Eng., Army HPC Research Center, Minneapolis*, 1998.

[39] 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.

[40] O. Livne, *Coarsening by compatible relaxation*, Numerical Linear Algebra with Applications, 11 (2004), pp. 205–227.

[41] S. Maclachlan, D. Osei-Kuffuor, and Y. Saad, *Modification and compensation strategies for threshold-based incomplete factorizations*, sisc, 34 (2012), pp. 48–75.

[42] M. Magolu Monga Made, R. Beauwens, and G. Warzee, *Preconditioning of discrete Helmholtz operators perturbed by a diagonal complex matrix*, Comm. in Numer. Meth. in Engin., 16 (2000), pp. 801–817.

[43] T. A. Manteuffel, *Shifted incomplete Cholesky factorization*, in Sparse Matrix Proceedings 1978 (Sympos. Sparse Matrix Comput., Knoxville, Tenn., 1978), SIAM, Philadelphia, Pa., 1979, pp. 41–61.

[44] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Mathematics of Computation, 31 (1977), pp. 148–162.

[45] N. Munksgaard, *Solving sparse symmetric sets of linear equations by preconditioned conjugate gradient method*, ACM Transactions on Mathematical Software, 6 (1980), pp. 206–219.

[46] A. C. Muresan and Y. Notay, *Analysis of aggregation-based multigrid*, SIAM Journal on Scientific Computing, 30 (2008), pp. 1082–1103.

[47] Y. Notay, *DRIC: a dynamic version of the RIC method*, Numer. Lin. Alg. Applic., (1994). to appear.

[48] ——, *An aggregation-based algebraic multigrid method*, Electronic Transactions on Numerical Analysis, 37 (2010), pp. 123–146.

[49] T. Oliphant, *An implicit numerical method for solving two-dimensional time-dependent diffusion problems*, Quart. Appl. Math., 19 (1961), pp. 221–229.

[50] M. Olschowka and A. Neumaier, *A new pivoting strategy for gaussian elimination*, Numerical Linear Algebra with Applications, 240 (1996), pp. 131–151.

[51] D. Osei-Kuffuor and Y. Saad, *Preconditioning Helmholtz linear systems*, Applied Numerical Mathematics, 60 (2009), pp. 420–431.

[52] D. Ron, I. Safro, and A. Brandt, *Relaxation-based coarsening and multiscale graph organization*, SIAM Multiscale Modelling and Simulations, 9 (2011), pp. 407–423.

[53] D. J. Rose, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, Graph Theory and Computing, R. C. Read, ed., Academic Press, New York, (1972), pp. 183–217.

[54] A. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, 1987, ch. 4.

[55] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM Journal on Scientific and Statistical Computing, 14 (1993), pp. 461–469.

[56] ——, *ILUT: a dual threshold incomplete ILU factorization*, Numerical Linear Algebra with Applications, 1 (1994), pp. 387–402.

[57] ——, *ILUM: a multi-elimination ILU preconditioner for general sparse matrices*, SIAM Journal on Scientific Computing, 17 (1996), pp. 830–847.

[58] ——, *Iterative Methods for Sparse Linear Systems, 2nd edition*, SIAM, Philadelpha, PA, 2003.

[59] ——, *Multilevel ILU with reorderings for diagonal dominance*, SIAM Journal on Scientific Computing, 27 (2005), pp. 1032–1057.

[60] Y. Saad and B. Suchomel, *ARMS: An algebraic recursive multilevel solver for general sparse linear systems*, Numerical Linear Algebra with Applications, 9 (2002).

[61] J. B. Schroder, *Smoothed aggregation solvers for anisotropic diffusion*, Numerical Linear Algebra with Applications, 19 (2012), pp. 296–312.

[62] W. F. Tinney and J. W. Walker, *Direct solution of sparse network equations by optimally ordered triangular factorization*, Proc. IEEE, 55 (1967), pp. 1801–1809.

[63] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, San Diego, CA, 2001.

[64] 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., vol. 1457, Lecture notes in Math., Springer Verlag, 1990.

[65] R. Varga, *Factorization and normalized iterative methods*, in Boundary problems in differential equations, R. Langer, ed., Univ. of Wisconsin Press, Madison, 1960, pp. 121–142.

[66] J. W. Watts III, *A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation pressure equation*, Society of Petroleum Engineers Journal, 21 (1981), pp. 345–353.