

# Effective Preconditioning through Ordering Interleaved with Incomplete Factorization \*

Ingyu Lee

Padma Raghavan

Esmond G. Ng

## 1 Introduction

Consider the solution of a sparse linear system  $Ax = b$  when the matrix  $A$  is symmetric and positive definite. A typical iterative solver is obtained by using the method of Conjugate Gradients (CG) [15] preconditioned with an incomplete Cholesky (IC) factor  $\hat{L}$  [4]. The latter is an approximation to the (complete) Cholesky factor  $L$ , where  $A = LL^T$ . Consequently, the process of computing  $\hat{L}$  relies to a large extent on data structures and graph-theoretic methods used to compute  $L$  in a sparse direct solver.

During sparse Cholesky factorization, some of the elements in  $A$  that are zeroes will fill-in and become nonzeros in the factor  $L$  [9, 12]. An incomplete factor  $\hat{L}$  is obtained by performing Cholesky factorization while selectively discarding some fraction of the fill-in nonzeros as soon as they are created. The two main types of incomplete Cholesky factorization schemes involve either a symbolic *level-of-fill* approach [20], or a *drop-threshold* approach [21, 30]. For  $\hat{L}$  generated from either approach, retaining more fill typically leads to a better preconditioner, i.e., one that can significantly reduce the number of CG iterations and/or prevent failure [22]. When such preconditioners are required for the application, both level of fill and drop-threshold schemes benefit from a fill-reducing ordering scheme.

The fill incurred during Cholesky factorization depends on the sparsity structure of  $A$  and not on the numerical values of the elements. This fact is exploited in a first ordering step in direct solvers to control fill-in; ordering algorithms compute a symmetric permutation of  $A$  using graph-theoretic methods to limit the number of nonzeros in the corresponding  $L$ . Although there has been earlier work on computing orderings tailored for incomplete factorizations by D’Azevedo et al. [5, 6], typical incomplete factorization schemes tend to use orderings computed to reduce fill in the complete sparse Cholesky factor  $L$ .

Currently, level-of-fill (symbolic) and drop-threshold (numeric) incomplete factorization schemes use fill-reducing ordering schemes as follows. The ordering step is a first symbolic step (as in a direct solver) to compute a fill-reducing permutation; this permutation is directed toward reducing fill in the complete sparse Cholesky factor  $L$  and not in its incomplete form  $\hat{L}$ . In a subsequent step, strategies are incorporated to select elements to retain or discard to obtain  $\hat{L}$  as an approximation to the sparse Cholesky factor  $L$  of the permuted matrix  $A$ . Thus, in broad terms, for a given permutation  $\pi$  of  $A$  to reduce fill in  $L(\pi)$ , both symbolic and numeric forms of incomplete factorization compute  $\hat{L}(\pi)$  as an approximation to the complete Cholesky factor  $L(\pi)$  with an error matrix  $E(\pi) = L(\pi) - \hat{L}(\pi)$ .

This paper focuses on the role of fill-reducing orderings in computing effective incomplete factor preconditioners. Consider computing a permutation  $\tau$  using an ordering scheme that is specifically directed towards reducing fill in the incomplete factor  $\hat{L}(\tau)$  corresponding to a level-of-fill or drop-threshold scheme. Now there exists a complete sparse Cholesky factor  $L(\tau)$  for this permutation of  $A$

---

\*This work was funded in part by the National Science Foundation through grants NSF ACI-0102537 and NSF EIA-022191 and DMR-0205232 and in part by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract number DE-AC03-76SF00098.

with an error matrix  $E(\tau) = L(\tau) - \hat{L}(\tau)$ . We conjecture that this approach could potentially lead to a better approximation and thus a more effective preconditioner than from the traditional approach; i.e.,  $E(\tau)$  is smaller than  $E(\pi)$  where  $\tau$  and  $\pi$  are constructed to reduce fill in  $\hat{L}(\tau)$  and  $L(\pi)$  respectively.

Orderings to explicitly reduce fill in the incomplete factor  $\hat{L}$  could potentially be constructed from any of the three broad classes of fill reducing orderings, namely band/envelope or profile methods such as the Reverse Cuthill McKee [12], greedy minimum degree like heuristics [1, 11, 18, 23, 27], and divide-and-conquer nested-dissection methods [13, 17]. In addition, hybrids of the latter two classes [2, 14] are also potentially viable. However, the focus of our paper is on a natural formulation in terms of greedy, minimum degree like orderings which can be easily interleaved with the incomplete factorization process.

In this paper, we develop an interleaved minimum degree ordering and incomplete factorization scheme and we report on its performance on a large test-suite of sparse matrices. Section 2 contains notation and background. Section 3 contains a description of our main algorithm and its variants. Section 4 contains extensive empirical results which indicate that our new methods perform significantly better than the traditional schemes. Section 5 attempts to explain the improved performance of our methods and Section 6 contains some concluding remarks.

## 2 Background

In this section, we provide a brief background on incomplete Cholesky preconditioners and their application to accelerate the convergence of the method of Conjugate Gradients (CG). We start with an overview of fill-reducing ordering schemes and their role in determining the structure of the sparse Cholesky factor  $L$  and its incomplete form  $\hat{L}$ . We then continue with an overview of the two main types of incomplete factorization schemes and end with a short review on re-factorizing after applying diagonal shifts if the factorization process were to breakdown.

Henceforth, we use  $L$  to denote the sparse Cholesky factor corresponding to a specific fill-reducing permutation of the coefficient matrix  $A$ . We use  $\hat{L}$  to denote its incomplete form. For any matrix  $B$ , we use  $B_{*,i}$  to denote its  $i$ -th column and  $(B_{i,*})$  to denote its  $i$ -th row.

**Fill-reducing orderings and the sparsity structure of  $L$ .** A fill reducing ordering of  $A$  is a symmetric permutation of its rows/columns typically computed using symbolic heuristics. Such heuristics are applied either to the graph of  $A$  or to a series of elimination graphs representing the evolving structure of  $L$  as it is computed by a symbolic equivalent of Cholesky factorization. These methods are thus based on a graph model of Cholesky factorization [24, 25]; more details can be found in classic books on sparse direct methods [9, 12].

As mentioned earlier, ordering schemes consist of methods from three broad classes (band/envelope, greedy and nested-dissection) and their hybrids. Of particular relevance to this paper are the class of greedy minimum degree like orderings [1, 11, 18, 23, 27]. The minimum degree algorithm was first proposed by Tinney and Walker [29] and is the symmetric analog of the Markowitz ordering scheme [19]. The algorithm minimizes the number of operations in the rank-1 update at each step of a right-looking sparse Cholesky factorization. An efficient implementation, “multiple minimum degree” (MMD), is due to Liu [18]; Amestoy, Davis and Duff [1] have developed “approximate minimum degree” which uses an approximation to the degree to reduce ordering time. A related ordering heuristic, also proposed by Tinney and Walker [29], is the minimum deficiency algorithm, which minimizes the number of fill entries *introduced* at each step of sparse Cholesky factorization. More recently, efficient analogs of this scheme have also been developed and they tend to reduce the fill and operation count better than minimum degree at the expense of a slight penalty in ordering time [23, 27].

Fill reducing orderings can affect the quality of an incomplete factorization preconditioner. It has been generally observed that for very sparse  $\hat{L}$ , envelope type orderings, such as RCM (Reverse Cuthill Mckee), may be superior to minimum degree and its variants in accelerating the convergence of Conjugate Gradients [8]. However, for  $\hat{L}$  with more fill-in, minimum-degree performs quite well and is

generally the fill-reduction method of choice for a cache-efficient blocked implementations of incomplete Cholesky factorization [22].

**Orderings for  $\hat{L}$ .** Earlier research on developing ordering schemes to improve the quality of the preconditioner  $\hat{L}$  were based on numeric metrics. D’Azevedo et al. [5, 6] developed a greedy ordering scheme using numeric metrics related to the norm of the discarded portion of the rank-1 update matrix in a right-looking implementation of incomplete factorization. Their metric allowed explicitly selecting as the next column to be numbered, a column with the minimum discarded fill in its rank-1 update. This metric can be viewed as a numeric analog of the ‘minimum deficiency’ symbolic metric of Tinney and Walker [29]; however, it is related purely to the quality of preconditioning and not to the sparsity of  $\hat{L}$ . The ‘minimum discarded fill’ method was successful in significantly improving the quality of the preconditioner but the associated computational overheads were observed to be significant [5, 6].

**Types of Incomplete Cholesky Factorization.** The basic step in incomplete Cholesky (IC) factorization is that of computing  $\hat{L}$ , an incomplete form of the (complete) sparse factor  $L$  for a given fill-reducing permutation  $\pi$  of  $A$ . There are two broad types of methods based on either symbolic or numeric measures.

The ‘level-of-fill’ incomplete factorization is best described using the graph model of sparse Cholesky factorization [24, 26]. It uses a symbolic metric that characterizes the nonzero structure of  $L$  and thus of  $\hat{L}$ . Consider  $L$ , the Cholesky factor corresponding to a fill reducing ordering  $\pi$ . If  $L_{i,j} \neq 0$ , then there exist one or more paths from  $j$  to  $i$  in the graph of the permuted  $A$  with intermediate vertices that are numbered less than  $\min(i, j)$  [12, 24, 26]. Such a path is known as a *fill-path*. Define as  $\text{length}(p)$ , the number of intermediate vertices in  $p$ , a fill-path between vertices  $j$  and  $i$ . Consider the set  $\mathcal{F}_{i,j}$  denoting the set of fill-paths between  $j$  and  $i$ . Now the level-of-fill of the nonzero  $L_{i,j}$  is defined as  $\min_{p \in \mathcal{F}_{i,j}} \{\text{length}(p)\}$ .

In a level-of-fill  $k$  incomplete Cholesky factor  $\hat{L}$ , only those nonzeros in  $L$  that have levels of fill less than or equal to  $k$  are retained. Thus,  $\hat{L}$  corresponding to a level of fill 0 has exactly the same nonzero structure as the original  $A$  permuted according to  $\pi$ . For larger  $k$ , more fill in is retained in  $\hat{L}$  typically making it a more effective preconditioner. Henceforth, we refer to this scheme as ICF, and when necessary as ICF( $k$ ), where  $k$  is a small integer denoting the level-of-fill.

A drop-threshold incomplete Cholesky uses a numeric approach. The columns of  $\hat{L}$  are computed from left to right much as in complete sparse Cholesky factorization by accumulating updates from earlier columns, computing the square-root of the diagonal element and then scaling the sub-diagonal elements by the diagonal element. However, nonzero entries with magnitude smaller than a given threshold are discarded [21, 30]. There are several ways to define and use the threshold. For example, the threshold can be a non-negative value or a non-negative factor relative to some measure of the size of  $A$  (such as the norm of  $A$  or  $\max_{i,j} |A_{i,j}|$ ). Other restrictions can also be imposed, such as discarding from each column all but some  $m$  nonzero entries satisfying the threshold condition [28], where  $m$  is typically a small constant such as five or ten. Henceforth, we refer to such schemes as ICT, and when necessary as ICT( $\omega$ ), where  $\omega$  is the drop-threshold (a small real number). It is customary to start ICT factorization after diagonally scaling the original matrix to obtain unit diagonal values ( $D^{-1/2} \times A \times D^{1/2}$ ); this allows easier application of the threshold criterion and a suitable re-scaling can be applied when using  $\hat{L}$  to accelerate CG.

Observe that for both ICF and ICT the ordering scheme dictates the filled structure of  $L$  and thus indirectly controls the form of the incomplete factor  $\hat{L}$ .

**Diagonal Shifts on IC Failures.** In both ICF and ICT factorizations, a central problem is that of breakdowns when a diagonal element  $\hat{L}_{i,i}$  becomes close to zero or negative. The simplest approach is that of replacing the diagonal by a small positive constant and then proceeding with the incomplete factorization. However, in practice this can lead to poor quality preconditioners. Another approach for handling such failures is to consider the incomplete re-factorization of the shifted matrix  $A + \alpha I$ . Now a key question is that of determining the proper value of the shift  $\alpha$ ; ideally, it should be the smallest magnitude shift required to prevent breakdowns. If the shift is large, then it can affect the quality of preconditioning through  $\hat{L}$ . Additionally, the shift will also affect the sparsity structure (and hence

memory requirements) of  $\hat{L}$  from an ICT scheme. A general strategy is to start with a small  $\alpha$  which is then iteratively increased if a breakdown occurs [16].

### 3 An Interleaved Minimum Degree Ordering

We now develop our scheme for interleaved ordering and incomplete factorization (ICF or ICT) based on the greedy minimum degree metric of Tinney [29]. We henceforth refer to our ordering as ‘Interleaved Minimum Degree’ (IMD). The ordering  $\tau$  computed by IMD is specifically designed to reduce the number of operations in computing the incomplete factor  $\hat{L}$  whereas, in the traditional scheme, the ordering  $\pi$  is computed to reduce the number of operations to compute  $L$ .

Our method uses a purely symbolic, degree like metric to control the number of operations in IC and the fill incurred in  $\hat{L}$ . In addition, we use a simple numeric measure to break ties among columns with the same degree metric. We refer to our IMD scheme in conjunction with a level of fill ICF( $k$ ) as IMDF; likewise, we call our IMD scheme in conjunction with a drop-threshold ICT as IMDT.

Consider true sparse Cholesky factorization when the algorithm proceeds column by column from left to right along an  $N \times N$  matrix. The  $i$ -th factorization step ( $i = 1, \dots, N - 1$ ) is described by the following equations starting with  $H_1 = A$ ;  $I_p$  is the  $p \times p$  identity matrix,  $v_i$  is an  $N - i$  vector of the sub-diagonal elements of the current  $H_i$ , and  $H_{i+1}$  is the remaining  $(N - i) \times (N - i)$  submatrix.

$$\begin{aligned} H_i &= \begin{bmatrix} d_{ii} & v_i^T \\ v_i & B_{i+1} \end{bmatrix} = L_i \begin{bmatrix} 1 & 0 \\ 0 & H_{i+1} \end{bmatrix} L_i^T \\ L_i &= \begin{bmatrix} \sqrt{d_{ii}} & 0 \\ v_i/\sqrt{d_{ii}} & I_{n-i} \end{bmatrix}, \text{ and } H_{i+1} = B_{i+1} - v_i v_i^T / d_{ii}. \end{aligned}$$

In incomplete Cholesky, the remaining submatrix  $L_i$  and  $H_{i+1}$  at the  $i$  step are computed by discarding some elements of  $v_i$  and some elements from the rank-1 update by  $v_i$ . Consequently, for any  $i, 1 \leq k \leq N$ , incomplete Cholesky uses an approximation of the form  $\hat{H}_i = H_i - E_i$  where  $E_i$  is the error matrix corresponding to values dropped from  $v_i$  and  $H_{i+1}$ .

Our algorithm is but a natural extension of the original minimum degree to reduce the number of operations associated with the rank-1 update from the current column in a right-looking IC scheme. In IMD, ordering is interleaved with numeric computations. In a first step, the column to be numbered  $i$  is selected as the one with minimal degree in the current  $\hat{H}_i$ . In the second phase, this column is factored and sub-diagonal nonzeros in the column are dropped according to the level-of-fill or drop-threshold criterion. Additionally, the rank-1 update is performed with this column; if the current column is  $i$  and  $\hat{H}_{j,i} \neq 0$  is to be retained according to the level-of-fill or drop-threshold criterion, column  $i$  in  $\hat{H}$  is used to update column  $j$ . Finally, the degree metric is updated for all such neighboring columns  $j$  of  $i$ . Observe that  $j$  is the original column number in  $A$  and although this column has not yet been numbered in the IMD ordering, it will be assigned a number higher than  $i$ .

Our ordering scheme with a symbolic degree metric is fully interleaved with the incomplete factorization process. Henceforth, we use  $\hat{L}$  to denote the matrix at the  $i$ -th step from an algorithmic viewpoint starting with  $\hat{L}$  initialized to  $A$ . The first  $i - 1$  columns of this matrix are the computed columns of the final preconditioner ( $L_i$ ) while its rightmost  $n - i$  submatrix corresponds to  $\hat{H}_{i+1}$ . Figure 1 contains a schematic of our basic algorithm using this  $\hat{L}$  notation.

An additional detail involves retaining or dropping an element of the form  $\hat{L}_{p,j}$ , arising from an update to column  $j$  by the current column  $i$  in which the element  $\hat{L}_{j,i}$  and  $\hat{L}_{p,i}$  are retained. As explained above,  $j$  and  $p$  are original column numbers in  $A$ ; these columns have not yet been numbered in IMD but they will eventually be numbered higher than  $i$ . In an ICF( $k$ ) implementation, we apply the update and retain  $\hat{L}_{p,j}$  only if the level (in terms of fill-paths as discussed earlier) of this update is less than or equal to  $k$ , i.e., it arises from a fill-path of length less than or equal to  $k$ . Consider next, an ICT( $\omega$ ) scheme with a threshold condition by which  $\hat{L}_{p,j}$  is retained if  $|\hat{L}_{p,j}| \geq \omega \times \sqrt{\hat{L}_{j,j} \times \hat{L}_{p,p}}$ . Now,

```

Apply diagonal scaling, i.e.,  $A = D^{-1/2} \times A \times D^{1/2}$ 
 $N \leftarrow$  dimension of  $ofA$ 
 $\alpha \leftarrow \alpha_0$ 
while  $\hat{L}$  has not been computed do
  Initialize  $\hat{L}$  to  $A$ 
  Mark all columns ‘unnumbered’
  Set flag to ‘continue’
  while there are ‘unnumbered’ columns and flag is ‘continue’ do
    Select a column with minimum degree (from all unnumbered columns of  $\hat{L}$ ); number it  $i$ .
    Compute column  $\hat{L}_{*,i}$ .
    if  $\hat{L}_{i,i} \leq 0$  then
      Shift diagonal by  $\alpha$  and set  $\alpha \leftarrow 2 \times \alpha$ 
      Set flag to ‘breakdown’
    else
      for all remaining unnumbered columns,  $j$  do
        Retain/drop  $\hat{L}_{j,i}$  using levels of fill/threshold criterion.
      end for
      for all remaining unnumbered columns,  $j$  do
        Update  $\hat{L}_{*,j}$  if  $\hat{L}_{j,i} \neq 0$ 
        Drop/retain  $\hat{L}_{p,j} \neq 0$  using levels of fill/threshold criterion.
      end for
    end if
  end while
end while

```

Figure 1: The Interleaved Minimum Degree (IMD) Algorithm.

we apply the update and retain  $\hat{L}_{p,j}$  only if it satisfies the threshold condition at the current point in the factorization process. We believe that our approach allows us to limit the amount of intermediate storage required in a right-looking IC implementation while allowing a consistent application of the level-of-fill or drop-threshold criterion.

In all greedy schemes, tie-breaking among columns with the same value of the metric is of considerable importance [11]. We view tie-breaking as a potential opportunity to improve the quality of the preconditioner  $\hat{L}$  by using some suitable numeric measure that can be computed inexpensively as part of our IMD scheme. Henceforth, IMD refers to the formulation where ties are broken arbitrarily.

Our tie-breaking schemes are based on the magnitude of the diagonal elements of columns with the same minimal value of the degree metric. The first variant we consider is motivated by the idea of stability through pivot selection. If two or more columns have the same minimal value of the metric, then we select as the next column to be numbered the one with the largest diagonal value. We refer to this scheme as IMD(F/T)-BD (for big diagonal).

Our second tie-breaking scheme reflects a ‘greedy’ attempt to potentially reduce breakdowns during incomplete factorization; we refer to this scheme as IMD(F/T)-SD (for small diagonal). Recall that during Cholesky factorization (whether it is incomplete or not), the diagonal values can only get smaller in magnitude through subsequent rank-1 updates as  $L_{i,i} = A_{i,i} - \sum_{p < i, L_{i,p} \neq 0} \{L_{i,p}^2\}$ . Thus, to prevent potential breakdowns we break ties by selecting the column with the smallest diagonal among those with minimal degree. This scheme is also partly motivated by certain structural developments that occur in the course of minimum degree orderings [11] for  $L$ . Although our scheme is for  $\hat{L}$ , we still expect such behavior (albeit less pronounced). More specifically, there is a growth of sets of *indistinguishable* vertices (or columns) in the elimination graph. Two or more vertices *indistinguishable* vertices are fully connected to each other and in addition they have the same set of neighbors. Thus, ordering among

them does not affect sparsity. However, it does have the potential to affect the numerical process; for example, if four columns  $p, q, r, s$  are indistinguishable then the first to be numbered will result in updates to the remaining three. Now if the column with the smallest diagonal value is numbered highest among the four, it may suffer a breakdown as its diagonal value gets reduced by updates from the three columns numbered earlier. IMD-SD attempts to prevent such breakdowns by numbering the column with the smallest diagonal value first.

As the factorization proceeds, the diagonal value of the current column may cease to be a nonzero positive value. In such a case, we apply a diagonal shift of  $\alpha$  and re-apply our algorithm to the shifted matrix  $A + \alpha I$ .

In summary, our method is a column-by-column implementation of minimum degree to locally reduce the number of operations in successive rank-1 updates in an incomplete Cholesky factorization. The extra costs of implementing the ordering interleaved with the factorization correspond to the those of finding columns with minimum degree and performing the degree update. To support the latter, in our IMD with arbitrary tie breaking we use a heap data-structure. This structure will at worst add an overhead of  $O(N \log N)$  for a matrix with  $N$  columns. However, for IMD-SD and IMD-BD with numeric tie-breaking, the simple heap data-structure is no longer appropriate because it does not provide an easy mechanism to identify all columns with the same degree metric. Now, we use a combination of heaps and queues. The heap is based on the degree values; each heap-node represents a specific degree value and is in turn linked to a queue containing all columns with that value of the degree metric. Tie-breaking involves traversing the queue associated with the minimum degree. We also maintain a ‘reverse look-up index’ per column to efficiently identify the degree queue to which it belongs; this allows associated degree updates to occur in constant time.

## 4 Empirical Results

In this section, we report on the results our experiments to evaluate the performance of our IMD schemes. A key aspect concerns comparisons with ICF and ICT preceded by (i) the Minimum Discarded Fill (MDF) ordering for  $\hat{L}$  [5, 6], and (ii) the more traditional Multiple Minimum Degree (MMD) and the Reverse Cuthill McKee (RCM) orderings [10] for the complete Cholesky factor  $L$ . Our comparisons with MDF are somewhat limited because the code is no longer available [7]. We therefore developed a simple implementation of MDF in C (as suggested by the authors of MDF) to compare the quality of the preconditioner for a small number of test cases. We report on these experiments first and then consider in greater detail the performance of our IMD schemes and the traditional ICF and ICT ordered by MMD and RCM for reducing fill in the (complete) Cholesky factor. We henceforth refer to these traditional forms as MMD-ICF, MMD-ICT, RCM-ICF, and RCM-ICT. We label our ICF scheme, IMDF when arbitrary tie-breaking is used and IMDF-BD/IMDF-SD when tie-breaking is numerical (using big or small diagonal elements); similarly, we label our ICT counterparts IMDT, IMDT-BD and IMDT-SD.

For our comparisons with MDF, we consider a set of six matrices (`bccstk14`, `bcsstk16`, `s1rmt3m1`, `s2rmq4m1`, `bodyy5`, `fdm1`) representative of our larger test suite described in Table 1. MDF orderings are known to be significantly expensive to compute [5, 6]. Furthermore, our simple C implementation is not tuned for the best performance and is thus not suitable for comparing execution times. Consequently, our experiments concern the quality of preconditioners computed by MDF and our IMDF schemes. MDF is suitable only for ICF and we consider ICF with 4 levels of fill ( $k=0, 1, 2,$  and  $3$ ) for each method. Observe that even if the level of fill is the same, MDF and IMDF variants can lead to differences in the number of nonzeros in the preconditioner. We therefore report on the number of iterations of CG (required upon preconditioning through MDF, IMDF, IMDF-BD and IMDF-SD) as a function of the number of nonzeros in the incomplete factor (reported as the ‘scaled fill’,  $\frac{|\hat{L}|}{|A|}$ ). These results are shown in Figure 2; the plots clearly indicate that in most instances, our IMDF schemes produce better quality preconditioners for approximately equal amounts of fill in the incomplete factor.

We used a test suite of 34 matrices shown Table 1 for our experiments to comparatively evaluate

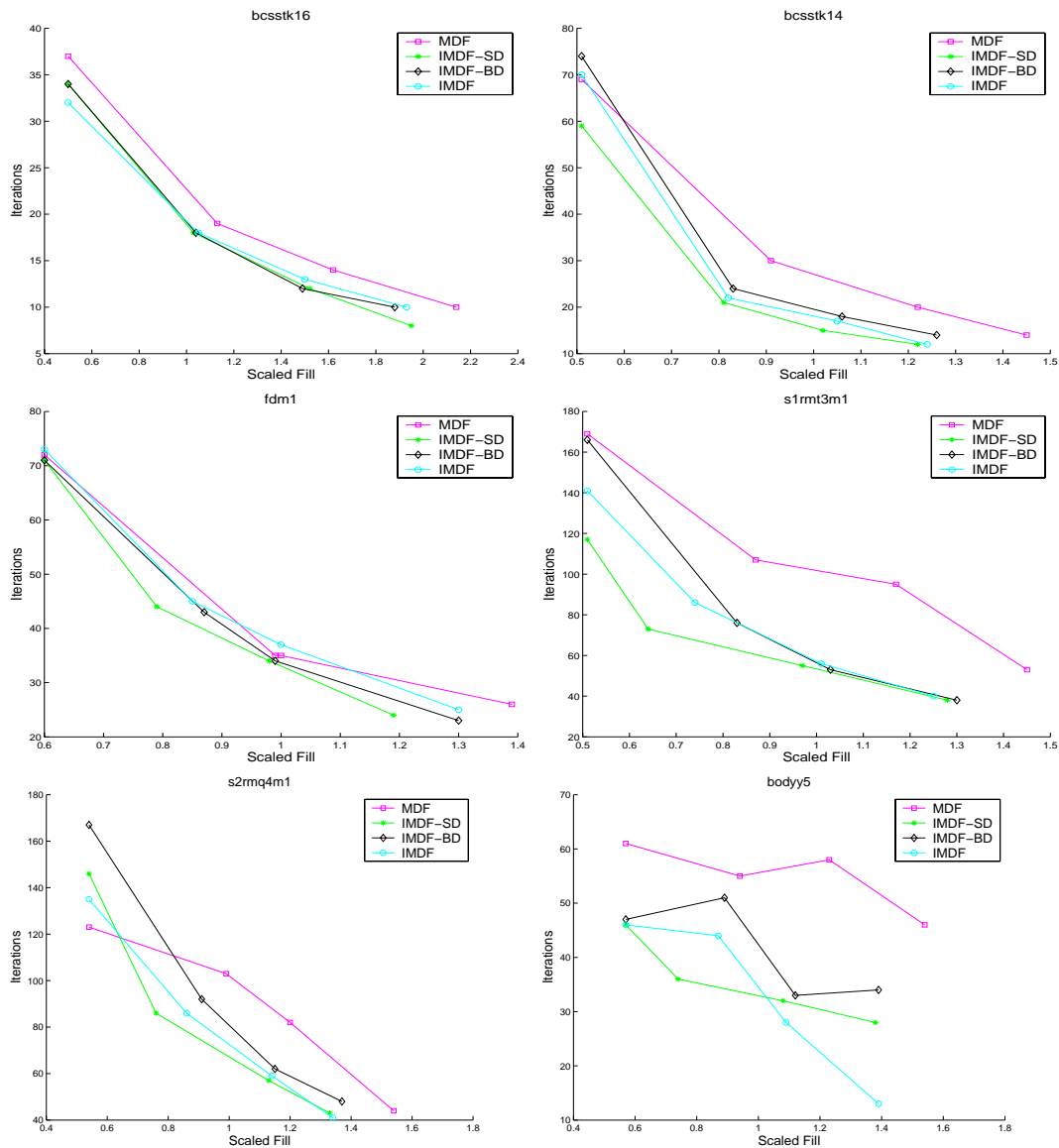


Figure 2: Quality of preconditioners using Minimum Discarded Fill (MDF), and Interleaved Minimum Degree (IMDF, IMDF-BD and IMDF-SD) schemes for levels of fill 0, 1, 2, and 3. The Y-axis indicates the number of CG iterations and the X-axis indicates the number of nonzeros in the incomplete factor relative to  $|A|$ .

Matrix	Rank	$ A $	Description
bcsstk08	1,074	7,107	TV studio
bcsstk09	1,083	9,760	Square plate clamped
bcsstk10	1,086	11,578	Buckling of a hot washer
bcsstk11	1,473	17,857	Ore car – lumped mass
bcsstk14	1,806	32,630	Roof of the Omni Coliseum
bcsstk16	4,884	147,631	US Army Corps of Engineers dam
bcsstk18	11,948	80,519	R.E. Ginna Nuclear Power Station
bcsstk19	817	3,855	Part of a suspension bridge
qa8fm	66,127	863,353	Mass matrix for a 3D acoustic
qa8fk	66,127	863,353	Finite element stiffness matrix for a 3D acoustic
s1rmt3m1	5,489	112,505	Cylindrical shell, 30x30 triangular mesh
s1rmq4m1	5,489	143,300	Cylindrical shell, 30x30 quadrilateral mesh
s2rmq4m1	5,489	143,300	Cylindrical shell, 30x30 quadrilateral mesh
s2rmt3m1	5,489	112,505	Cylindrical shell, 30x30 triangular mesh
s3dkq4m2	90,449	2,455,670	Cylindrical shell, 150x100 quadrilateral mesh
s3dkt3m2	90,449	1,921,955	Cylindrical shell, 150x100 triangular mesh
s3rmq4m1	5,489	143,300	Cylindrical shell, 30x30 quadrilateral mesh
s3rmt3m1	5,489	112,505	Cylindrical shell, 30x30 triangular mesh
s3rmt3m3	5,357	106,526	Cylindrical shell, graded mesh with 1666 triangles
bodyy4	17,546	69,548	From NASA, collected by Alex Pothen
bodyy5	18,589	73,721	From NASA, collected by Alex Pothen
bodyy6	19,366	76,787	From NASA, collected by Alex Pothen
msc00726	726	17,623	Structural engineering matrix from MSC/NASTRAN
msc01050	1,050	15,103	Structural engineering matrix from MSC/NASTRAN
msc01440	1,440	23,856	Structural engineering matrix from MSC/NASTRAN
msc04515	4,515	51,111	Structural engineering matrix from MSC/NASTRAN
msc10848	10,848	620,313	Structural engineering matrix from MSC/NASTRAN
msc23052	23,052	588,934	Structural engineering matrix from MSC/NASTRAN
1138_bus	1,138	2,596	Power system networks
5K	5,658	217,236	Electromagnetic simulations in accelerator design
26K	26,148	1,053,456	Electromagnetic simulations in accelerator design
56K	56,196	11,722,070	Electromagnetic simulations in accelerator design
fdm1	6,050	17,979	Five point stencil, Poisson equations Dirichlet boundary conditions
fdm2	32,010	95,629	Computational chemistry problem

Table 1: Characteristics of Test Matrices.

the performance of our three IMD schemes and the traditional incomplete factorization with MMD and RCM orderings. Our experiments include ICF with 4 levels of fill ( $k=0, 1, 2,$  and  $3$ ) and ICT with four threshold values ( $\omega = .1, .05, .001,$  and  $.0001$ ). In summary, we report on two types of preconditioning (ICF and ICT) each with 4 states of incompleteness and 34 problems for each method. Thus, there are 272 tests per method with a total of 1,360 tests over all five schemes.

In the first stage of our experiments we compute the incomplete factor  $\hat{L}$  after diagonal scaling of the original  $A$ . When breakdowns occur in this stage we use the diagonal shift scheme of Lin et al. [16]. We re-factor with an initial shift  $\alpha = 10^{-3}$ ; subsequent breakdowns result in refactoring with a doubling of the shift value  $\alpha$ . If  $\alpha$  becomes larger than the diagonal element in  $A$ , we conclude that the method failed to compute the incomplete Cholesky preconditioner. In the second stage of our experiments, we solve for four different right-hand side vectors using  $\hat{L}$  in the first stage as a preconditioner for CG. We use the row-sum vector as one of the right-hand-side vectors (with 3 other random vectors) and a stopping criterion of  $10^{-6}$  residual norm or maximum iterations of 1,000. A method fails to solve the linear system if the desired tolerance is not achieved within the maximum number of iterations.

In the rest of this section we report on our empirical evaluation of the five methods for ICF and ICT preconditioners. We use scatter plots of the various performance metrics over the test instances and we typically group results for ICF followed by results for ICT. Within each group (ICF or ICT), we typically provide two figures drawn to the same scale with the one to the left for our three IMD schemes and the one to the right for MMD and RCM ICF/ICT. We also provide summaries of performance metrics using simple totals and geometric means for average values.

**Quality of Preconditioners.** To evaluate the quality of preconditioners in accelerating the convergence of CG, we consider the following three metrics.

- Scaled fill: defined as the number of nonzeros in the preconditioner relative to the number of nonzeros in the original matrix, i.e.,  $\frac{|\hat{L}|}{|A|}$ .
- Iterations: the number of iterations for CG to converge using a row-sum vector of  $A$  for the right-hand-side.
- Operations: defined as the product ‘scaled fill  $\times$  iterations,’ a measure of the number of operations required to apply the preconditioner to solve for a single right-hand-side vector.

If any one of the five methods fails in either the first IC stage or in the second stage during CG iterations, we exclude that instance from figures and tables in this section.

Figure 3 shows the performance of ICF and ICT preconditioners with ‘scaled fill’ along the X-axis and ‘iterations’ along the Y-axis. A method is better if its points are tightly clustered about the bottom and left corner. In this sense, observe that IMD, IMD-SD and IMD-BD are noticeably better than the traditional MMD and RCM schemes for both ICF and ICT. IMD-SD forms the tightest cluster for both ICF and ICT with the effect being more pronounced for ICT.

We next consider the ‘operations’ metric; this is a scaled version of the actual number of operations required for a single solution (excluding preconditioner construction costs). We show this metric relative to IMD-SD for each test instance where no method fails. For each test instance, we divide the observed Operations by those for IMDF-SD (for ICF) or IMDT-SD (ICT). Figure 4 shows this ‘relative operations’ metric; a method performs better (worse) than IMD-SD if its value for that instance is smaller (greater) than 1. Observe that the plots for RCM and MMD indicate that in many instances they perform worse than any of the IMD variants. More specifically, IMD-SD is better than RCM in 60% of the ICF instances and in 80% of the ICT tests. Perhaps even more remarkable is the fact that IMD-SD is better than MMD in 90% of the ICF tests and in 95% of the ICT instances. These results indicate that using the degree metric to reduce the operation count during the rank-1 updates of the incomplete factor (as opposed to those for the complete factor) result in significantly improved preconditioners.

Summary results on the average performance of the five methods (over all successful instances) in terms of ‘scaled fill,’ ‘iterations’ and ‘operations’ are shown in Table 2. Consider the means relative to

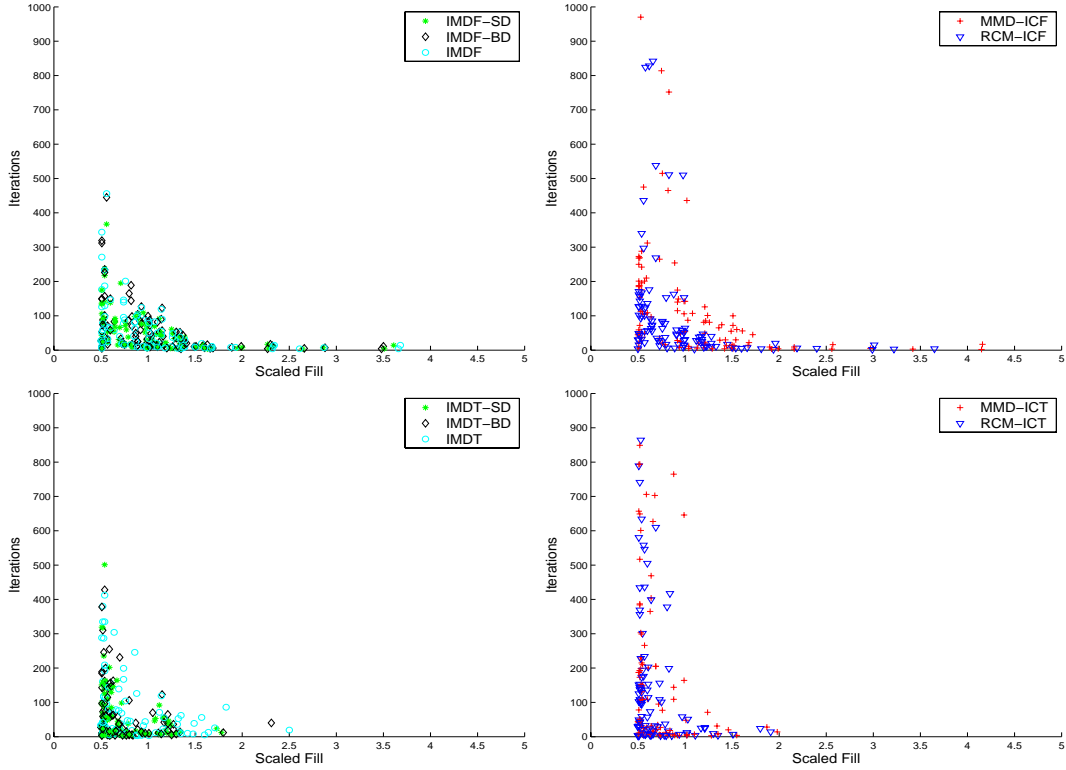


Figure 3: Scaled fill and iterations for ICF(0), ICF(1), ICF(2) and ICF(3) (top) and ICT(.1), ICT(.05), ICT(.001) and ICT(.0001) (bottom).

IMD-SD. On average, for ICF, IMDF without tie-breaking and IMDF-BD require 10% to 14% more iterations than IMDF-SD for nearly equal densities of the preconditioner. However, for ICT, the effect of the tie-breaking strategy is more pronounced; while IMDT-BD performs nearly as well as IMDT-SD, IMDT requires 33% more iterations than IMDT-SD while incurring approximately 12% more ‘Scaled Fill.’

Performance summaries (averages) in Table 2 clearly indicate significant improvements through our new IMD methods over traditional ICF and ICT using MMD or RCM orderings. RCM-ICF on average incurs 8% less scaled fill to converge with 46% more iterations than IMDF-SD while RCM-ICT (on average) requires 81% more iterations than IMDT-SD at approximately equal densities of the preconditioner. MMD-ICF requires 78% more iterations than IMDF-SD while MMD-ICT requires 107% more iterations than IMDT-SD at nearly equal densities of the preconditioner. The performance improvements from IMD are even more prominent when one considers the ‘Operations’ metric. On average, RCM-ICF requires 1.72 times the operations for IMDF-SD; the corresponding factor is 2.24 for MMD-ICF. Likewise, RCM-ICT (MMD-ICT) requires 2.1 (2.52) times the number of operations required by IMDT-SD. These results are indeed very promising and they clearly demonstrate the potential of our methods for obtaining very effective preconditioners for accelerating the convergence of CG.

**Execution Times.** We now report the observed execution times for computing the preconditioner and applying it to solve four linear systems. As indicated in Section 3, interleaving minimum degree ordering with the incomplete numeric factorizations requires more complicated data-structures and their management and thus increases the time to compute the preconditioner. As described earlier, the first stage of computations include diagonal scaling, ordering, and incomplete numeric factorization. This stage also includes re-factorizations after diagonal shifting until the preconditioner computation

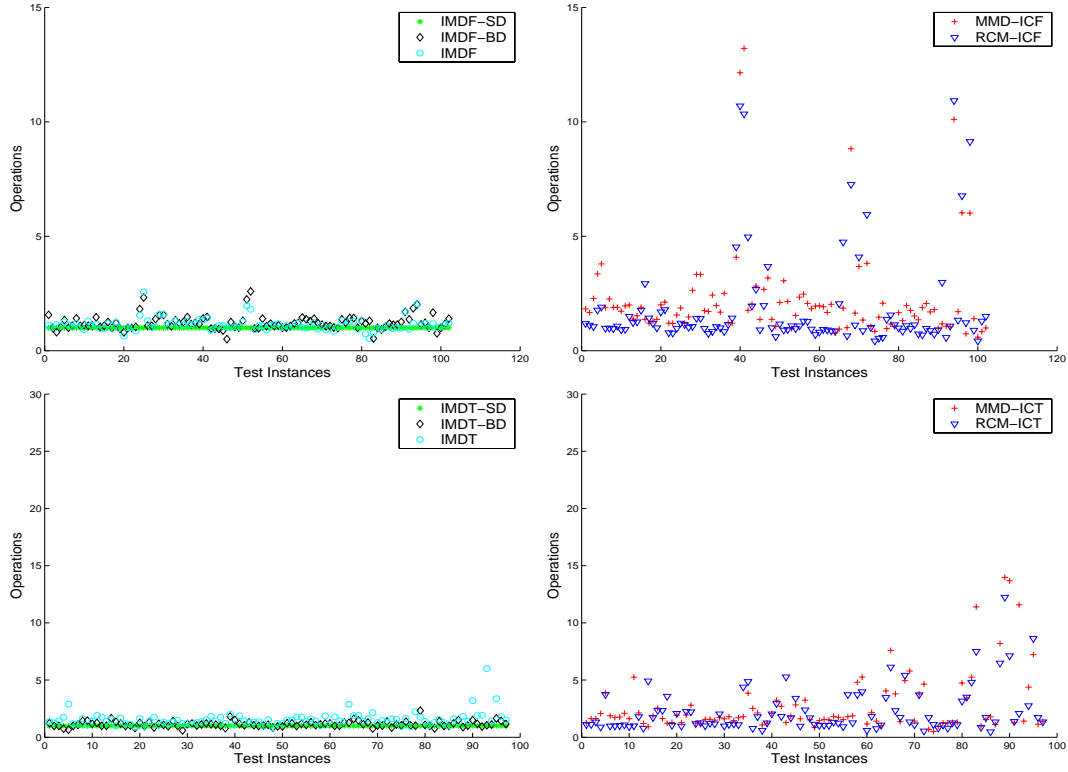


Figure 4: Operations (scaled fill  $\times$  iterations) for ICF(0), ICF(1), ICF(2) and ICF(3) (top) and ICT(.1), ICT(.05), ICT(.001) and ICT(.0001) (bottom).

Metric	Mean Performance									
	ICF (level of fill)					ICT (drop-threshold)				
	IMDF			MMD	RCM	IMDT			MMD	RCM
	SD	BD	-			SD	BD	-		
Scaled Fill	.94	.98	.96	.98	.86	.67	.68	.75	.67	.65
Iterations	28	32	31	50	41	27	29	36	56	49
Operations	26.7	31.5	29.7	49.2	35.3	18.2	20.2	26.8	37.8	31.9
	Mean Performance Relative to IMD-SD									
Scaled Fill	1.0	1.04	1.02	1.04	.92	1.0	1.01	1.12	1.01	.97
Iterations	1.0	1.14	1.11	1.78	1.46	1.0	1.07	1.33	2.07	1.81
Operations	1.0	1.18	1.16	2.24	1.72	1.0	1.08	1.26	2.52	2.1

Table 2: Average values of ‘scaled fill,’ ‘iterations’ and ‘operations.’ Relative values are computed with respect to IMD-SD; values smaller (greater) than 1 indicate that the method performed better (worse) than IMD-SD.

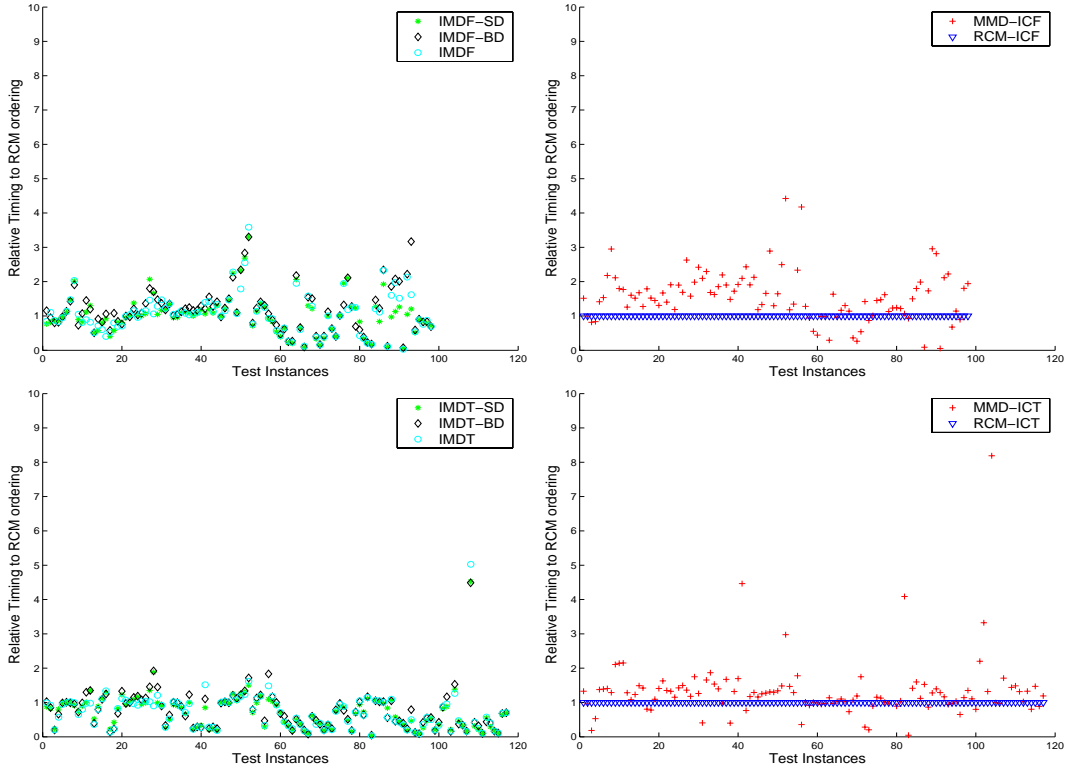


Figure 5: Total execution times for computing  $\hat{L}$  and four solutions relative RCM-ICF (top) and RCM-ICT (bottom).

succeeds or fails; the latter occurs when the maximum allowable diagonal shift  $\alpha$  is exceeded. We henceforth refer to the execution time of this stage as that for computing the preconditioner  $\hat{L}$ . We report timing results only over all successful instances. The second stage of computations concerns applying the preconditioner to solve for a given right-hand-side vector using CG. We consider a total of four different right-hand-side vectors including three selected at random and the row sum vector. Figure 5 shows total time over both stages relative to that for RCM-ICF or RCM-ICT. We chose to normalize by the latter as it tends to be the fastest scheme. The scatter plots indicate that in a majority of instances the overall performance of IMD-SD is better than that of RCM-ICF/ICT.

Summary results for the two individual stages and their totals are shown in Table 3; the top half of the table provides averages of actual execution times while the bottom half provides means relative to RCM-ICF/ICT. Our IMD variants require significantly less time than MMD-ICF/ICT to set up  $\hat{L}$ ; this improvement is a consequence of the fact that our IMD schemes require fewer re-factorizations after applying diagonal shifts. However, not surprisingly, our IMD methods are significantly slower by factors of 1.44 to 1.81 than RCM-ICF/ICT with respect to computing the preconditioner. But this higher cost of preconditioner results in significantly faster solution time for four right-hand-side vectors, RCM-ICT requires approximately 100% more time than IMDT-SD while MMD-ICT requires approximately 133% more time than IMDT-SD. When the preconditioner construction time is added to the time for solving four systems, our IMDF-SD requires 15% less time than RCM-ICF while IMDT-SD requires 46% less time than RCM-ICT. Observe that the time for solving for four systems is typically much greater than the time to compute  $\hat{L}$  for all methods. Thus, the performance benefits from our IMD schemes should be more dramatic when the cost of computing the preconditioner is amortized over a longer sequence of solutions for different right-hand-side vectors.

Metric	Mean Performance									
	ICF (level of fill)					ICT (drop-threshold)				
	IMDF			MMD	RCM	IMDT			MMD	RCM
	SD	BD	-			SD	BD	-		
$\hat{L}$	.38	.44	.28	.7	.26	.34	.34	.23	.44	.19
4 rhs	5.62	6.35	5.96	8.87	7.43	9.64	10.32	9.97	22.0	19.0
$\hat{L} + 4$ rhs	7.94	8.83	8.07	12.6	9.3	11.3	12.0	11.4	24.9	20.6
	Relative Mean Performance									
	Means Relative to RCM-ICF					Means Relative to RCM-ICT				
	ICF (level of fill)					ICT (drop-threshold)				
	IMDF			MMD	RCM	IMDT			MMD	RCM
	SD	BD	-			SD	BD	-		
$\hat{L}$	1.44	1.67	1.06	2.62	1.0	1.82	1.81	1.22	2.33	1.0
4 rhs	.75	.85	.8	1.19	1.0	.5	.54	.52	1.15	1.0
$\hat{L} + 4$ rhs	.85	0.94	.86	1.36	1.0	.54	.58	.55	1.2	1.0

Table 3: Average execution times for computing  $\hat{L}$ , four solutions, and the total time to compute  $\hat{L}$  and solving 4 systems. Relative values (with respect to RCM-ICF/ICT) less (greater) than 1 indicate that the method performed faster (slower) than RCM-ICF/ICT.

**Failures and Diagonal Shifts.** Recall that if failures occur during the computation of  $\hat{L}$ , we shift as  $A + \alpha I$  before recomputing  $\hat{L}$ . Successive shifts double  $\alpha$  and a method fails to compute  $\hat{L}$  if  $\alpha$  exceeds the maximum allowable shift value [16]. The actual shift value used before  $\hat{L}$  is computed successfully can affect the fill incurred; often  $|\hat{L}|$  from ICT is larger with a larger shift  $\alpha$ . The shift value also affects the quality of preconditioning; it is common for a larger shift results in slower convergence or breakdowns during PCG.

Our interleaved IMD orderings appear to be remarkably effective in computing  $\hat{L}$  with smaller shifts on average and a very small number of failures as shown in Table 4. For very sparse  $\hat{L}$ , such as level of fill 0 and drop threshold  $\omega = .1$ , all methods need several shifts. However, for higher fill instances, such as level of fill 3 and drop threshold  $\omega = .001$ , the number of shifts on average for RCM and MMD methods is 4.25 to 5.6 times that of IMD-SD. The need for fewer shifts for IMD methods translates to fewer breakdowns in the process of computing  $\hat{L}$ . Additionally, as the final diagonal shift is smaller, the preconditioning quality is affected less adversely translating to fewer breakdowns during PCG. The number of failures shown in Table 4 indicate that our IMD methods are remarkably robust when compared to the traditional RCM/MMD based ICT and ICF. RCM (MMD) based ICF and ICT methods have failure rates of approximately 19% (22%) over both stages for our test instances while IMD and its SD and BD variants have no failures. Thus, using the degree (structural) metric for ordering  $\hat{L}$  in IMD methods appears to indirectly improve the numerical properties of  $\hat{L}$ .

## 5 Toward Reducing Discarded Fill

As shown in the preceding section, our IMD method (and its variants based on numerical metrics for tie-breaking) result in preconditioners that are significantly more effective than traditional IC using MMD and RCM orderings. We now provide some conjectures on the underlying mechanism that could potentially explain this improved performance.

Consider the error matrix  $E$  defined as the difference matrix between the sparse complete Cholesky factor  $L$  and the incomplete Cholesky factor  $\hat{L}$  using the same ordering. For the traditional ICF/ICT with the ordering  $\pi$  computed in a first MMD or RCM step we consider the difference matrix  $E(\pi) =$

$L(\pi) - \hat{L}(\pi)$ . For an ordering  $\tau$  computed by our IMD schemes, there exists a complete sparse Cholesky factor  $L(\tau)$  corresponding to this permutation of  $A$  with the error matrix  $E(\tau) = L(\tau) - \hat{L}(\tau)$ . Such an error matrix can be used as a measure of the quality of the preconditioner [3]. We conjecture that our IMD methods on average result in smaller error matrices and thus better approximations to the corresponding true sparse Cholesky factors, i.e.,  $|E(\tau)| \leq |E(\pi)|$  where  $\tau$  and  $(\pi)$  reduce fill in  $\hat{L}(\tau)$  and  $L(\pi)$  respectively. In Figure 6, we show the 2-norm of the error matrix for three matrices over all methods. This value is consistently smaller for IMD and its SD and BD variants. IMD-SD has the smallest error matrix and it typically results in a preconditioner leading to the smallest number of PCG iterations.

Consider the assumption that the improved performance of IMD and its variants arises from the ability of such methods to compute low error incomplete factorizations. Now this attribute must be a consequence of the minimum degree metric applied to IC. We conjecture that in IMD, the minimum degree metric serves as a structural (or symbolic) measure of the *discard* matrix at each stage of the right-looking process. Note that because the degree metric is the number of sub-diagonal nonzeros satisfying the ICF/ICT criterion at the current stage, IMD automatically selects a column of degree  $m$  such that  $m^2$ , the number of operations in the corresponding rank-1 update is minimal (and no more than  $m^2$ ). This quantity  $m^2$  also represents the number of elements in the rank-1 update matrix of which some fraction will be dropped. This fraction is a structural measure of the discarded matrix at each stage.

We expect the minimal degree in IMD to be small when compared with MMD for complete sparse Cholesky factor  $L$ . In the latter case, even for near optimal fill orderings of model sparse matrices from five-point finite-difference discretizations on a two dimensional  $\sqrt{N} \times \sqrt{N}$  grids, the minimum degree will grow to  $O(\sqrt{N})$  [12]. However, for such problems, most IC schemes would limit the number of nonzeros in any column of  $\hat{L}$  to be within a small constant factor of those in  $A$ . This limited growth in minimum degree will be exactly computed and used in our IMD schemes. By using the actual degree in  $\hat{L}$ , IMD can more effectively control the sparsity of  $\hat{L}$  and limit the size of the discarded submatrix at each rank-1 update. We thus conjecture that IMD results in a more *economical* incomplete factorization process that indirectly helps reduce a structural measure of the errors incurred at each step through dropping.

To attempt to substantiate our claim, we consider ICF factorizations and compute  $\Phi$ , the size of fill-in relative to  $|A|$  and  $\Delta$  the total number of dropped elements relative to  $|A|$  for each test matrix for levels of fill 0 through 3. The number of diagonal shifts can affect both  $\Phi$  and  $\Delta$  values and we therefore report values only for instances that required no diagonal shifting for all methods. Figures 7 and 8 show  $\Phi + \Delta$  and  $\Delta$  values normalized by those for MDF-SD (set at 1 for each instance). The average values of  $\Phi + \Delta$  and  $\Delta$  are shown to the left in Figure 9 with averages of the values relative to

Method	Average Number of Diagonal Shifts									Number of Failures				
	ICF				ICT				ICF & ICT	IC		PCG		IC & PCG
	$k$				$\omega$					$k$	$\omega$	$k$	$\omega$	
	0	1	2	3	0.1	.05	.01	.001						
MMD	3.7	3.9	2.9	2.8	5.2	4.9	5.2	4.5	4.1	11	15	15	20	61 (22)
RCM	3.5	3.9	2.4	2.5	4.7	5.0	4.5	3.4	3.7	11	17	10	14	52 (19)
IMD-SD	3.0	1.6	0.8	0.5	3.1	3.2	2.3	0.8	1.9	0	0	0	0	0 (0)
IMD-BD	3.0	1.2	0.8	0.5	3.0	3.3	2.2	0.8	1.8	0	0	0	0	0 (0)
IMD	3.0	1.5	0.7	0.6	4.2	3.7	2.7	1.1	2.2	0	0	0	0	0 (0)

Table 4: Average number of diagonal shifts (left) and number of failures (right). Failure percentages are based on 272 test cases per method using the row-sum right-hand-side vector.

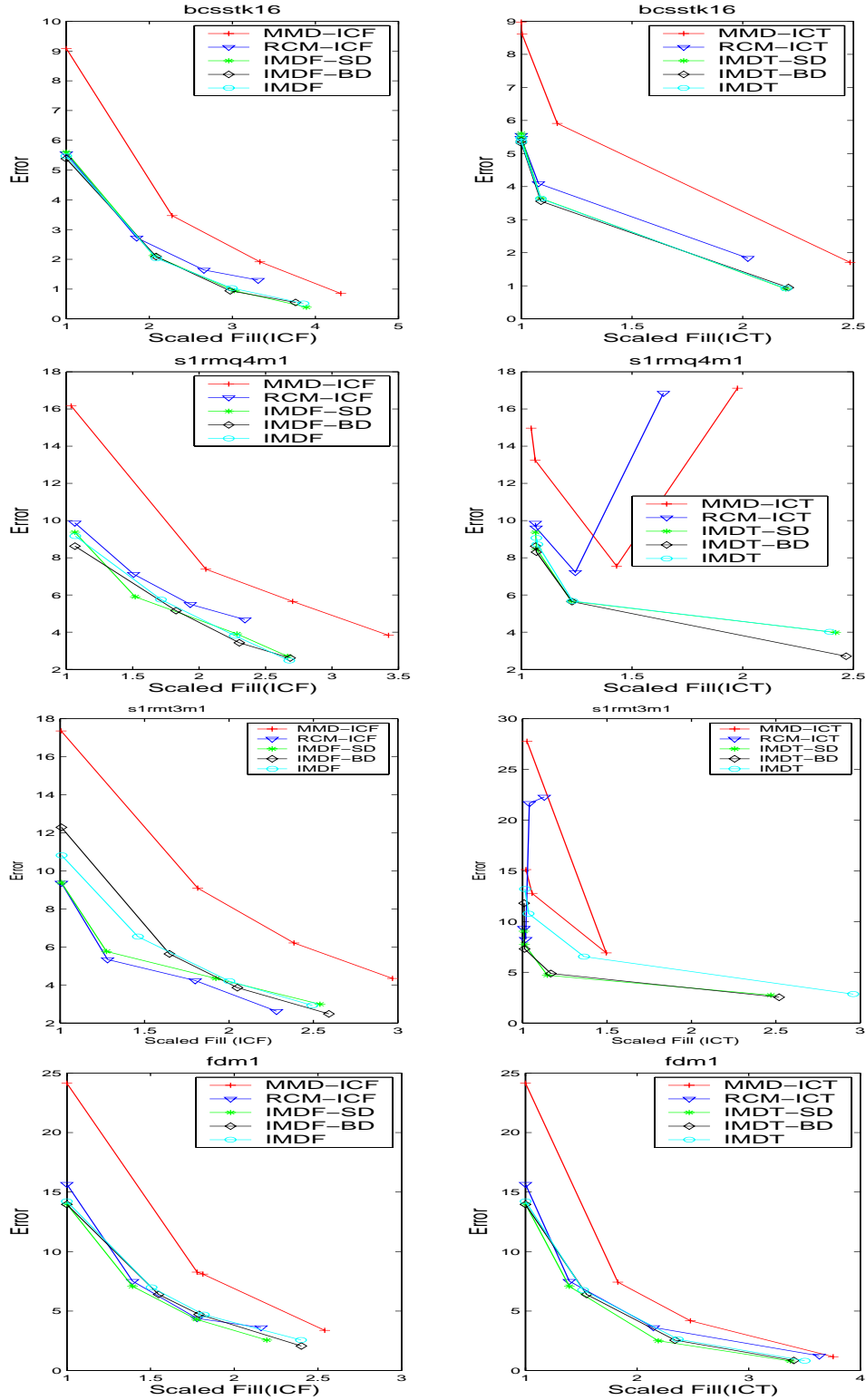


Figure 6: Sizes of the error matrix (two-norm) for test matrices bcsstk16, s1rmq4m1, s1rmt3m1 and fdm1 for ICF (left) and ICT (right) schemes.

IMDF-SD to the right. On average MMD-ICF (RCM-ICF) requires 100% (12%) more  $\Phi + \Delta$  than the IMD schemes. Considering only  $\Delta$  (as a structural measure of the discarded/error matrix), MMD-ICF on average discards 84% more elements than IMDF while RCM-ICF discards 27% more elements than IMDF.

If the degree metric in IMD does indeed serve as a structural measure to reduce the discarded fill, then the improved quality of our preconditioners would be consistent with the results reported by D’Azevedo et al. [5, 6]. The latter consider ‘minimum discarded fill’ (MDF) orderings for level-of-fill IC. MDF computes an ordering by explicitly minimizing the numeric error at each step in a right-looking incomplete Cholesky factorization. The authors report that the scheme produce better quality preconditioners than traditional ICF using MMD or RCM at the expense of significant overheads incurred for computing the MDF ordering. As reported in Section 4, in most instances, our IMDF methods produce incomplete Cholesky preconditioners of better quality than MDF. The fact that MDF and our IMD methods produce better quality preconditioners than the traditional approaches could be a consequence of reduced discarded fill. Further improvements in the quality of preconditioners produced by our IMD methods could be a consequence of improved tie-breaking and the ability of IMD and its variants to effectively limit fill in  $\hat{L}$ .

## 6 Conclusions

In this paper, we have developed a class of new IMD schemes for effective preconditioning through interleaving of greedy minimum degree ordering and incomplete factorization. Our methods perform significantly better than traditional MMD-IC both with respect to the time required for computing the preconditioner and its effectiveness in accelerating the convergence of CG, Empirical results indicate that on average RCM-IC (MMD-IC) requires 72% to 110% (124% to 152%) more operations than our IMD-SD (with small-diagonal tie-breaking) to solve a linear system using preconditioned Conjugate Gradients. This improved quality of preconditioning is achieved at the cost of increased time to compute the preconditioner; on average IMD (with no tie-breaking) requires approximately 6% (20%) more time than RCM-ICF (RCM-ICT), the fastest class of methods of computing  $\hat{L}$ ). Tie-breaking increases the time to 44% (80%) more than RCM-ICF (RCM-ICT). However, this extra cost of computing the preconditioner can be justified if it can be amortized over a sequence of solutions for different right-hand side vectors. Our experiments indicate that when total time is considered including preconditioner set-up and its application to solve four systems, IMD-SD requires the least time (on average). Thus, the overall performance of our IMD schemes should be significantly better than MMD-IC or RCM-IC when the preconditioner is computed once and used to solve a larger number of linear systems. Our IMD schemes also appear to be considerably more robust than the traditional schemes with respect to computing the preconditioner.

IMD orderings reduce the number of operations in the rank-1 update of a right-looking incomplete Cholesky factorization. We conjecture that this process results in a preconditioner  $\hat{L}$  which may be a better approximation to the complete Cholesky factor  $L$ , thus providing more effective preconditioning. We further conjecture that such better approximations are achieved by using the degree metric. A smaller degree leads to smaller upper bound in the number of elements that could be discarded at each rank-1 update, thus serving as a structural measure for limiting the error incurred at each step of the factorization. This conjecture is somewhat substantiated by our experiments which indicate that the norm of the error matrix is smaller and (on average) the number of drops incurred using IMDF is substantially smaller than for MMD-ICF and RCM-ICF.

## References

- [1] P. R. AMESTOY, T. A. DAVIS, AND I. S. DUFF, *An approximate minimum degree ordering algorithm*, SIAM J. Mat. Anal. and Appl., 17 (1996), pp. 886–905.

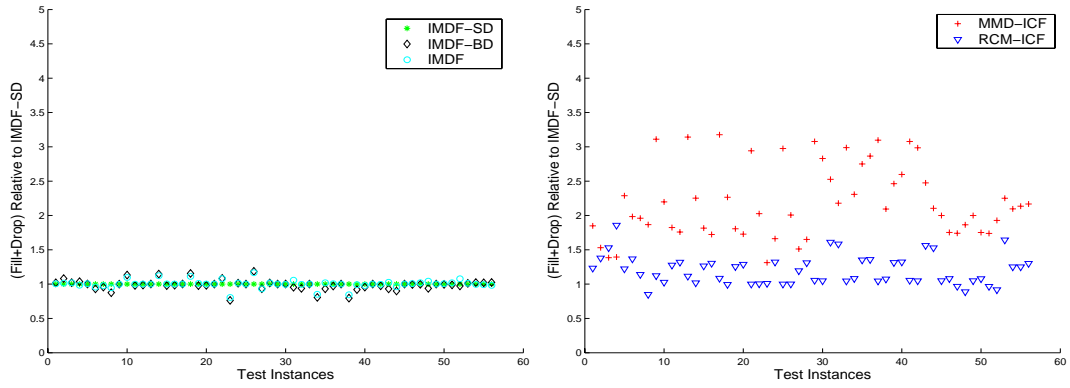


Figure 7: Filled and Dropped elements relative to  $|A| (\Phi + \Delta)$  during ICF.

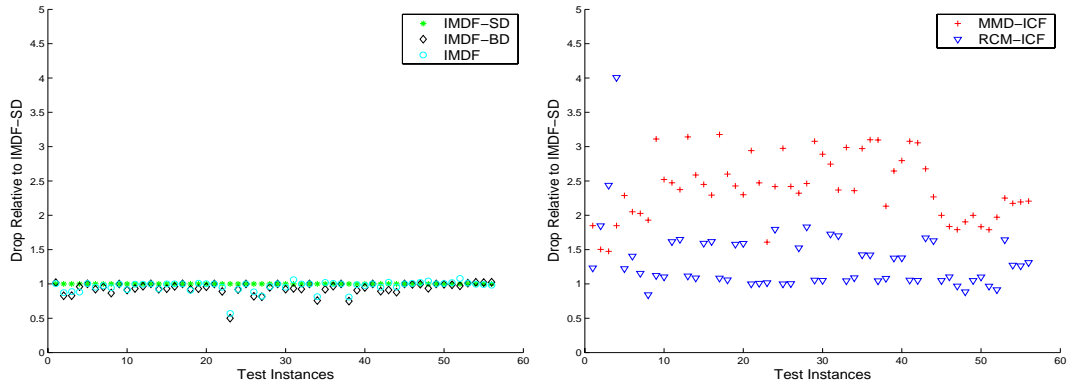


Figure 8: Dropped elements relative to  $|A| (\Delta)$  during ICF.

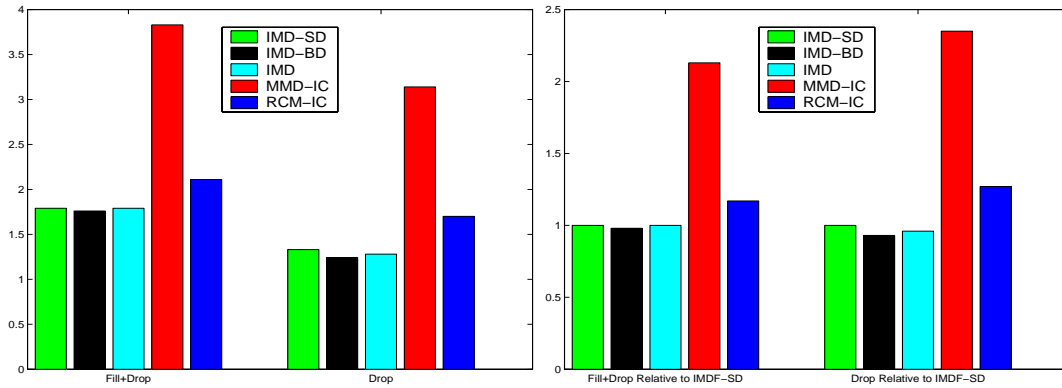


Figure 9: Summary means of  $\Phi$  (fill+drop) and  $\Delta$  (drop) are shown in the left figure; Means relative to IMDF-SD are shown in the figure to the right.

- [2] C. ASHCRAFT AND J. W. H. LIU, *Robust ordering of sparse matrices using multisection*, SIAM Journal on Matrix Analysis and Applications, 19 (1998), pp. 816–832.
- [3] O. AXELSSON AND N. MUNKSGAARD, *Analysis of incomplete factorizations with fixed storage allocation*, in Preconditioning Methods Theory and Applications, D. Evans, ed., Gordon and Breach, 1983, pp. 219–241.
- [4] P. CONCUS, G. GOLUB, AND D. O’LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, 1976, pp. 309–332.
- [5] E. D’AZEVEDO, P. FORSYTH, AND W. TANG, *Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems*, SIAM J. of Matrix Anal. Appl., 13 (1992), pp. 944–961.
- [6] ———, *Towards a cost-effective ilu preconditioner with high level fill*, BIT, 32 (1992), pp. 442–463.
- [7] ———, *Personal communication on the availability of minimum discarded fill ordering codes*, June 2004.
- [8] I. DUFF AND G. MEURANT, *The effect of ordering on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–637.
- [9] I. S. DUFF, A. M. ERISMAN, AND J. K. REID, *Direct Methods for Sparse Matrices*, Oxford University Press, Oxford, England, 1986.
- [10] J. E. CHU, A. GEORGE AND E. NG., *Sparsepak: Waterloo sparse matrix package user’s guide for sparsepa*, Tech. Rep. Technical Report CS-84-36, Department of Computer Science Cornell University, University of Waterloo, Ontario, Canada, 1984.
- [11] A. GEORGE AND J. LIU, *The evolution of the minimum degree ordering algorithm*, SIAM Review, 31 (1989), pp. 1–19.
- [12] A. GEORGE AND J. W.-H. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.
- [13] J. GEORGE, *Nested dissection of a regular finite element element mesh*, SIAM J. Numerical Analysis, 10 (1973), pp. 345–363.
- [14] B. HENDRICKSON AND E. ROTHBERG, *Improving the run time and quality of nested dissection ordering*, SIAM J. on Sci. Comput., 20 (1999), pp. 468–489.
- [15] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, National Bureau Standard J. Res., 49 (1952), pp. 409–436.
- [16] C. J. LIN AND J. MORE, *Incomplete cholesky factorizations with limited memory*, SIAM J. Sci. Comput., 21 (1999), pp. 24–45.
- [17] R. J. LIPTON, D. J. ROSE, AND R. E. TARJAN, *Generalized nested dissection*, SIAM J. Numer. Anal., 16 (1979), pp. 346–358.
- [18] J. W.-H. LIU, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 11 (1985), pp. 141–153.
- [19] H. MARKOWITZ, *The elimination form of the inverse and its application to linear programming*, Management Sci., 3 (1957), pp. 255–269.

- [20] 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.
- [21] N. MUNKSGAARD, *Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients*, ACM Trans. Math. Software, 6 (1980), pp. 206–219.
- [22] E. NG, B. PEYTON, AND P. RAGHAVAN, *A blocked incomplete cholesky preconditioner for hierarchical-memory computers*, in Iterative Methods in Scientific Computation IV, D. R. Kincaid and A. C. Elster, eds., vol. 5 of IMACS Series in Computational and Applied Mathematics, 1999, pp. 211–221.
- [23] E. G. NG AND P. RAGHAVAN, *Performance of greedy ordering heuristics for sparse Cholesky factorization*, SIAM J. Mat. Anal. and Appl., 20 (1999), pp. 902–914.
- [24] S. PARTER, *The use of linear graphs in Gaussian elimination*, SIAM Review, 3 (1961), pp. 364–369.
- [25] D. ROSE, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, R. C. Read, ed., Academic Press, 1972, pp. 183–217.
- [26] D. J. ROSE, R. E. TARJAN, AND G. S. LUEKER, *Algorithmic aspects of vertex elimination on graphs*, SIAM J. Comput., 5 (1976), pp. 266–283.
- [27] E. ROTHBERG AND S. C. EISENSTAT, *Node selection strategies for bottom-up sparse matrix ordering*, SIAM Journal on Matrix Analysis and Applications, 19 (1998), pp. 682–695.
- [28] Y. SAAD, *ILUT : A dual threshold incomplete factorization*, Num. Lin. Alg. Appl., 1 (1994), pp. 387–402.
- [29] W. F. TINNEY AND J. W. WALKER, *Direct solutions of sparse network equations by optimally ordered triangular factorization*, in Proceedings in IEEE, 1967, pp. 1801–1809.
- [30] Z. ZLATEV, *Use of iterative refinement in the solution of sparse linear systems*, SIAM J. Numer. Anal., 19 (1982), pp. 381–399.