Incomplete factorization

Karl Meerbergen

KU Leuven

October 16, 2013

Literature

- James Demmel, Applied numerical Linear Algebra, SIAM, 1997.

  Gives a good overview of preconditioning techniques.

- Anne Greenbaum, Iterative methods for solving linear systems, SIAM, 1997.

  Gives an excellent overview of Krylov methods. The book also briefly discusses preconditioning techniques.

- Youcef Saad, Iterative methods for sparse linear systemss, SIAM, 2003.

  Gives an excellent overview of Krylov methods. The book also discusses incomplete factorization techniques.

LU-factorization

- Linear algebra:

$$A = LU$$

  with L unit lower triangular and U upper triangular
- Numerical linear algebra:
  - Pivoting for stability.
  - Fill-in when matrices are sparse.
  - High computational cost $O(n^3)$ for a full matrix.
- Solving linear system $Ax = b$ with backtransformation:
  - $Ly = b$
  - $Ux = y$

LU-factorization

- Algorithm:

  for $k = 1, \dots, n$ **do**
    for $i = k + 1, \dots, n$ **do**
      $a_{ik} = a_{ik} / a_{kk}$
      for $j = k + 1, \dots, n$ **do**
        $a_{ij} = a_{ij} - a_{ik} a_{kj}$
      end for
    end for
  end for

- There are six (6) different forms to write this algorithm: the six permutations of 3 loops.
- This algorithm is the *kij* form
- Different forms are used in different circumstances

Right-looking LU-factorization

- *kji* form is column oriented and right-looking.
- Algorithm:

for $k = 1, \ldots, n$ **do**
  for $j = k+1, \ldots, n$ **do**
    $a_{jk} = a_{jk}/a_{kk}$
    for $i = k+1, \ldots, n$ **do**
      $a_{ij} = a_{ij} - a_{ik}a_{kj}$
    end for
  end for
end for

Other forms

- The six forms:

type	orientation	updating
kji	column	right-looking
jki	column	left-looking
jik	column	left-looking
kij	row	right-looking
ikj	row	
ijk	row	

- Naming convention from the FORTRAN world where matrices are column oriented.

Sparse matrix storage formats

- Two formats are in use (and a few variations):
  - Coordinate format
  - Compressed sparse column/row format
- Illustrate for the following matrix

$$\begin{bmatrix} -3.5 & 1.4 & & 8 & -2 \\ & -3 & & & \\ & & 3 & 2 & \\ & & & 1 & -5 \\ & & & & -5 \end{bmatrix}.$$

Sparse matrix storage formats

$$\begin{bmatrix} -3.5 & 1.4 & & 8 & -2 \\ & -3 & & & \\ & & 3 & 2 & \\ & & & 1 & -5 \\ & & & & -5 \end{bmatrix}.$$

- Coordinate format
  - `rows`: the row number of the non-zero
  - `columns`: the column number of the non-zero
  - `values`: the value of the non-zero

A possible COO representation for the example is

`rows`:	1	2	3	3	1	1	4	1	4	5
`columns`:	1	2	3	4	2	4	4	5	5	5
`values`:	-3.5	-3	3	2	1.4	8	1	-2	-5	-5

The non-zeroes can be stored in any order: this is one of the great advantages of the format.

Sparse matrix storage formats

$$\begin{bmatrix} -3.5 & 1.4 & & 8 & -2 \\ & -3 & & & \\ & & 3 & 2 & \\ & & & 1 & -5 \\ & & & & -5 \end{bmatrix}.$$

- Compressed sparse column (row) format Sorting the coordinate format by column we obtain the following data:

`rows r`:	*1*	*1*	*2*	*3*	*3*	*4*	*1*	*1*	*4*	*5*
`columns c`:	*1*	*2*	*2*	*3*	*4*	*4*	*4*	*5*	*5*	*5*
`values v`:	*-3.5*	*1.4*	*-3*	*3*	*2*	*1*	*8*	*-2*	*-5*	*-5*

Replace the columns array by an array that points
to the beginning of each column in the array of row numbers and values:

`columns c`: *1* *2* *4* *5* *8* *11*

Sparse matrix storage formats

- j-th element of `column` is a pointer to the beginning and end of a column
- Fast access of a column is thus possible
- It simplifies and speeds up matrix operations such as matrix vector product.
- For matrix assembly, the coordinate format is much more convenient since elements can be added in any order.
- For matrix assembly, the compressed format requires the use of insertion in an array, which is expensive.

LU factorization of a sparse matrix

- $A = LU$: $L + U$ is usually denser than A.

$$
\begin{bmatrix}
\times & & \times & \times & & \times \\
& \times & & \times & & \\
\times & & \times & & \times & \\
\times & \times & & \times & & \times \\
& & \times & & \times & \times \\
\times & & & \times & \times & \times
\end{bmatrix}
\quad A = LU \Longrightarrow \quad
\begin{bmatrix}
\times & & \times & \times & & \times \\
& \times & & \times & & \\
\times & & \times & \circ & \times & \circ \\
\times & \times & \circ & \times & \circ & \times \\
& & \times & \circ & \times & \times \\
\times & & \circ & \times & \times & \times
\end{bmatrix}
$$

- Factorization: $A = (L + D)D^{-1}(D + U)$ with L strictly lower diagonal and U strictly upper triangular
- Factorization:

for $k = 1, \ldots, n$ **do**
    for all (i, j) for which $a_{i,k}a_{k,j} \neq 0$ **do**
        $f_{i,j} = -a_{i,k}a_{k,j}/a_{k,k}$
        $a_{ij} = a_{i,j} + f_{i,j}$
    end for
end for

How does it work?

- Factorization: $A = (L+D)D^{-1}(D+U)$
- i,j element:

$$
\begin{aligned}
a_{i,j} &= \sum_{k=1}^{\min(i,j)} l_{i,k} d_k^{-1} u_{k,j} \\
l_{i,j} &= a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} d_k^{-1} u_{k,j} \quad (u_{j,j} d_j^{-1} = 1) \\
u_{i,j} &= a_{i,j} - \sum_{k=1}^{i-1} l_{i,k} d_k^{-1} u_{k,j} \quad (l_{i,i} d_i^{-1} = 1)
\end{aligned}
$$

or when A stores L, D^{-1} and U:

$$a_{i,j} = a_{i,j} - \sum_{k=1}^{\min(i,j)} a_{i,k} a_{k,j} / a_{k,k}$$

How does it work?

- Algorithm:

  for $k = 1, \ldots, n$ **do**
      for all (i, j) for which $a_{i,k}a_{k,j} \neq 0$ **do**
          $f_{i,j} = -a_{i,k}a_{k,j}/a_{k,k}$
          $a_{ij} = a_{i,j} + f_{i,j}$
      end for
  end for

- Factorize row k and column k and update all other elements.

$$\left[\begin{array}{cc|c|c} \multicolumn{2}{l|}{\mathit{Done}} & a_{1,k} & \mathit{Done} \\ & & \vdots & \\ \hline a_{k,1} & \cdots & a_{k,k} & \\ \hline \multicolumn{2}{l|}{\mathit{Done}} & & \mathit{Update} \end{array}\right]$$

Variations

- Some prefer the following decomposition:

$$A = (L + D)D^{-1}(U + D)$$

  with L strictly lower triangular and U strictly upper triangular
- Classical factorization in numerical analysis text books:

$$A = (L + I)(U + D)$$

- LDU factorizaton:

$$A = (L + I)D(U + I)$$

- Cholesky factorization:

$$A = (L + D)(L + D)^T$$

  where A is symmetric positive definite.

Demo of fill-in

Incomplete factorization

- Let S be the sparsity pattern of A: $S = \{(i,j) : a_{i,j} \neq 0\}$.
- Algorithm:

  for $k = 1, \ldots, n$ **do**

      for all $(i,j) \in S$ for which $a_{i,k} a_{k,j} \neq 0$ **do**

          $f_{i,j} = -a_{i,k} a_{k,j} / a_{k,k}$

          $a_{ij} = a_{i,j} + f_{i,j}$

      end for

  end for

M-matrix

- LU-factorization exists for a positive definite matrix (= Cholesky factorization, no pivoting required)
- Incomplete LU factorization may fail, even when the matrix is non-singular
- Sufficient condition for success:

Definition ($A \in \mathbb{C}^{n \times n}$ is an M-matrix)

1. $a_{ii} > 0$, for $i = 1, \ldots, n$,
2. $a_{ij} \leq 0$, $i, j = 1, \ldots, n$, $i \neq j$, and
3. A is nonsingular and all elements of A^{-1} are non-negative.

- When A is a matrix with non-positive off-diagonal entries, this is equivalent with any of the following statements:
  1. The eigenvalues of A have positive real parts.
  2. $A + A^T$ is positive definite.
  3. All principal minors of A are M-matrices.

Example

- Example:
- Laplacian:

$$L = \left[\begin{array}{ccc|ccc} 4 & -1 & & -1 & & \\ -1 & 4 & -1 & & -1 & \\ & -1 & 4 & & & -1 \\ \hline -1 & & & 4 & -1 & \\ & -1 & & -1 & 4 & -1 \\ & & -1 & & -1 & 4 \end{array}\right]$$

- Eigenvalues are 1.5858, 3.0000, 3.5858, 4.4142, 5.0000, and 6.4142
- Inverse

$$L^{-1} = \frac{1}{2415}\begin{bmatrix} 712 & 225 & 68 & 208 & 120 & 47 \\ 225 & 780 & 225 & 120 & 255 & 120 \\ 68 & 225 & 712 & 47 & 120 & 208 \\ 208 & 120 & 47 & 712 & 225 & 68 \\ 120 & 255 & 120 & 225 & 780 & 225 \\ 47 & 120 & 208 & 68 & 225 & 712 \end{bmatrix}$$

Example

- Matrix

$$A = \begin{bmatrix} 3 & & -1 & -1 & & -1 \\ & 2 & & -1 & & \\ -1 & & 3 & & -1 & \\ -1 & -1 & & 2 & & -1 \\ & & -1 & & 3 & -1 \\ -1 & & & -1 & -1 & 4 \end{bmatrix}$$

  has eigenvalues 0.2246, 1.5249, 2.5493, 3.2019, 4.1355, 5.3638 and has non-positive off-diagonal elements, so it is an M matrix.

- (Full) Cholesky factorization:

$$\widetilde{L} = L+D = \begin{bmatrix} 1.7321 & & & & & \\ 0 & 1.4142 & & & & \\ -0.5774 & 0 & 1.6330 & & & \\ -0.5774 & -0.7071 & -0.2041 & 1.0607 & & \\ 0 & 0 & -0.6124 & -0.1179 & 1.6159 & \\ -0.5774 & 0 & -0.2041 & -1.2964 & -0.7908 & 1.1485 \end{bmatrix}$$

Example

- Matrix

$$A = \begin{bmatrix} 3 & & -1 & -1 & & -1 \\ & 2 & & -1 & & \\ -1 & & 3 & & -1 & \\ -1 & -1 & & 2 & & -1 \\ & & -1 & & 3 & -1 \\ -1 & & & -1 & -1 & 4 \end{bmatrix}$$

  has eigenvalues 0.2246, 1.5249, 2.5493, 3.2019, 4.1355, 5.3638 and has non-positive off-diagonal elements, so it is an M matrix.
- incomplete Cholesky factorization:

$$\widetilde{L} = \begin{bmatrix} 1.7321 & & & & & \\ 0 & 1.4142 & & & & \\ -0.5774 & 0 & 1.6330 & & & \\ -0.5774 & -0.7071 & 0 & 1.0801 & & \\ 0 & 0 & -0.6124 & 0 & 1.6202 & \\ -0.5774 & 0 & 0 & -1.2344 & -0.6172 & 1.3274 \end{bmatrix}$$

Example

- incomplete Cholesky factorization:

$$\widetilde{L} = \begin{bmatrix} 1.7321 & & & & & \\ 0 & 1.4142 & & & & \\ -0.5774 & 0 & 1.6330 & & & \\ -0.5774 & -0.7071 & 0 & 1.0801 & & \\ 0 & 0 & -0.6124 & 0 & 1.6202 & \\ -0.5774 & 0 & 0 & -1.2344 & -0.6172 & 1.3274 \end{bmatrix}$$

with

$$R = A - \widetilde{L}\widetilde{L}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.3333 & 0 & -0.3333 \\ 0 & 0 & -0.3333 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.3333 & 0 & 0 & 0 \end{bmatrix}.$$

The eigenvalues of $\widetilde{L}^{-1}A\widetilde{L}^{-T}$ are 1.0000, 0.5305, 1.3536, 1.0000, 1.0000, 1.0000. The spectrum is pretty well clustered around one.

ILU

Theorem

1. *For any M-matrix A and for any sparsity pattern S that contains the main diagonal elements, there exist unique L and U (L lower triangular with ones on the main diagonal and U upper triangular) where $L+U$ have the sparsity pattern S, so that*

$$A-(L+D)D^{-1}(U+D)=R$$

where $r_{ij}=0$ for all $(i,j)\in S$.

2. *For any symmetric M matrix and (symmetric) sparsity pattern S containing the main diagonal, there exists a unique lower triangular matrix L so that*

$$A-(L+D)D^{-1}(U+D)=R$$

where $r_{ij}=0$ for all $(i,j)\in S$.

Modified ILU (MILU)

- Modify D so that $Ae = (L + D)D^{-1}(D + U)e$
- Motivation: preconditioner has the same 'mass' as the original matrix.
- Mathematical explanation: for second order self-adjoint elliptic equations, the ILU preconditioner produces $\mathrm{cond}((LU)^{-1}A) = O(h^{-2})$ while MILU produces $\mathrm{cond}((LU)^{-1}A) = O(h^{-1})$
- Exact factorization of element $a_{i,j}$:

$$\sum_{j=1}^{n} a_{i,j} = \sum_{j=1}^{n} \sum_{k=1}^{\min(i,j)} l_{i,k} d_k^{-1} u_{k,j}$$

- With dropping strategy:

$$\sum_{j=1,(i,j)\in S}^{n} \sum_{k=1}^{\min(i,j)} l_{i,k} d_k^{-1} u_{k,j} + \sum_{j=1,(i,j)\notin S}^{n} \sum_{k=1}^{\min(i,j)} l_{i,k} d_k^{-1} u_{k,j}$$

- Accumulate lost elements in D (the elements that belong to $R = A - L \cdot U$)

Modified ILU (MILU)

for $k = 1, \ldots, n$ **do**
    for (i, j) for which $a_{i,k}a_{k,j} \neq 0$ **do**
        Compute $f_{i,j} = -a_{i,k}a_{k,j}/a_{k,k}$.
        if $(i, j) \in S$ **then**
            $a_{i,j} = a_{i,j} + f_{i,j}$
        else
            $a_{i,i} = a_{i,i} + f_{i,j}$
        end if
    end for
end for

Other alternatives than MILU

- Relaxed ILU (RILU)

for $k = 1, \ldots, n$ **do**
  for (i,j) for which $a_{i,k}a_{k,j} \neq 0$ **do**
    Compute $f_{i,j} = -a_{i,k}a_{k,j}/a_{k,k}$.
    if $(i,j) \in S$ **then**
      $a_{i,j} = a_{i,j} + f_{i,j}$
    else
      $a_{i,i} = a_{i,i} + \alpha f_{i,j}$ with $0 < \alpha < 1$
    end if
  end for
end for

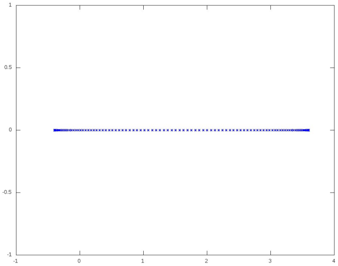
- Helmholtz equation:

$$-\nabla^2 u - k^2 u = f \quad \Rightarrow \quad (A - k^2 I)\mathbf{u} = \mathbf{f}$$

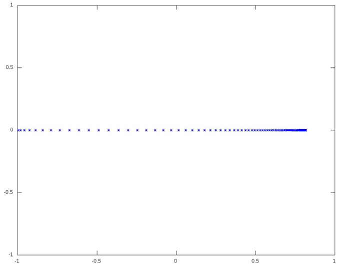
$A - k^2 I$ is indefinite and usually not an M matrix. Instead, build the preconditioner for $A + k^2 I$

Helmholtz equation

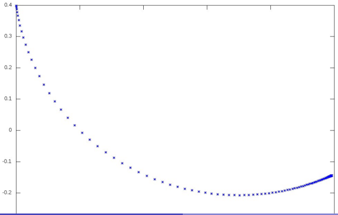
- Spectrum of $A - k^2 I$



- Spectrum of $(A + k^2 I)^{-1}(A - k^2 I)$



- Spectrum of $(A - k^2 I + i2k^2 I)^{-1}(A - k^2 I)$



Additional fill-in

- (M)ILU use the sparsity pattern of A for S
- The number of nonzero terms in the residual $R = A - (L + D)D^{-1}(U + D)$ can be reduced by using a larger set S.
- ILU does not work well for matrices that are not M-matrices.
- Various ideas:
  - Level of fill
  - Threasholding small absolute values
  - Pivoting
  - Other orderings

ILU with level of fill

- Fill-in = zero elements of A that become nonzero during factorization.
- There is an intuitive feeling that fill-in resulting from other fill-in elements is less important than fill-in coming from non-zero elements of A.

Definition

Level of fill

$$\text{Levfill}(a_{i,j}) = \min(\text{Levfill}(a_{i,j}), \max(\text{Levfill}(a_{i,k}), \text{Levfill}(a_{k,j})) + 1)$$

- Example

$$\begin{bmatrix} 0 & & 0 & 0 & & 0 \\ & 0 & & 0 & & \\ 0 & & 0 & \boxed{1} & 0 & 1 \\ 0 & 0 & \boxed{1} & 0 & 2 & 0 \\ & & 0 & \boxed{2} & 0 & 0 \\ 0 & & 1 & 0 & 0 & 0 \end{bmatrix}$$

ILU with level of fill

- Compute LevFill:

  $\mathrm{LevFill}(a_{i,j}) = 0$ for all $(i,j) \in S$ and $\mathrm{LevFill}(a_{i,j}) = \infty$ for all $(i,j) \notin S$.
  for $k = 1, \ldots, n$ **do**
      for all (i,j) **do**
          $\mathrm{Levfill}(a_{i,j}) = \min(\mathrm{Levfill}(a_{i,j}), \max(\mathrm{Levfill}(a_{i,k}), \mathrm{Levfill}(a_{k,j})) + 1)$
      end for
  end for

Example

- Matrix

$$A = \begin{bmatrix} 3 & & -1 & -1 & & -1 \\ & 2 & & -1 & & \\ -1 & & 3 & & -1 & \\ -1 & -1 & & 2 & & -1 \\ & & -1 & & 3 & -1 \\ -1 & & & -1 & -1 & 4 \end{bmatrix}$$

- ILU(1):

$$\widetilde{L} = \begin{bmatrix} 1.7321 & & & & & \\ 0 & 1.4142 & & & & \\ -0.5774 & 0 & 1.6330 & & & \\ -0.5774 & -0.7071 & -\mathit{0.2041} & 1.0607 & & \\ 0 & 0 & -0.6124 & 0 & 1.6202 & \\ -0.5774 & 0 & -\mathit{0.2041} & -1.2964 & -0.6944 & 1.2093 \end{bmatrix}$$

The eigenvalues of $(\widetilde{L}\widetilde{L}^T)^{-1}A$ are 1.0000, 0.8323, 1.0000, 1.0782, 1.0000, and 1.0000.

Example

- Note that

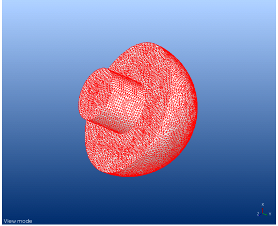
$$A - \widetilde{L}\widetilde{L}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.1250 & 0 \\ 0 & 0 & 0 & -0.1250 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- For ILU(0), we had

$$R = A - \widetilde{L}\widetilde{L}^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.3333 & 0 & -0.3333 \\ 0 & 0 & -0.3333 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.3333 & 0 & 0 & 0 \end{bmatrix}.$$

Example

- Model for acoustic radiation to infinity



  - $72,976$ unkowns
  - Finite elements
  - Infinite elements for acoustic radiation
  - Load consists of ‘rotating modes’.
  - $Ax = f \quad A(\omega) = K + i\omega C - \omega^2 M$

- Timings for different preconditioners

ILU(0)		ILU(1)	
P_1	P_2	P_1	P_2
72.03	50.01	89.03	88.28

with $P_1 =$ ILU for A and $P_2 =$ ILU for $A((1-i)\omega)$.

- ILU(1) requires 1.8 times the memory of ILU(0).

Threasholding and pivoting

- ILU(p) works well for M-matrices. Many problems are not M-matrices
- Throw away small elements in the matrix and use a direct solver: this does not work well when all elements are of the same order
- Throw away small elements in the factorization (ILUT)

  for $k = 1, \ldots, n$ **do**
      Compute the elements in row k and column k.
      Throw away all elements smaller than $\tau \|a_{k,:}\|$.
      Keep the largest p elements.
  end for

- ILUTP: ILUT with pivoting
  - Pivoting can help significantly in improving the quality of the preconditioner. As for direct methods, it is advantageous to pivot elements with large modulus to the main diagonal.

Multilevel techniques

- *Also see domain decomposition*
- ILU is hard to parallelize: the algorithm is sequential in principle.
- This parallelization of ILU is a problem similar to the parallelization of sparse direct methods. This is discussed in another lecture: see the choice of renumbering and the resulting elimination tree.
- For ILU, we can use an approach similar to domain decomposition:

$$\begin{bmatrix} A_1 & & & A_{1,4} \\ & A_2 & & A_{2,4} \\ & & A_3 & A_{3,4} \\ A_{4,1} & A_{4,2} & A_{4,3} & A_4 \end{bmatrix} \approx \begin{bmatrix} I & & & \\ & I & & \\ & & I & \\ \widetilde{A}_{4,1} & \widetilde{A}_{4,2} & \widetilde{A}_{4,3} & I \end{bmatrix} \cdot \begin{bmatrix} A_1 & & & A_{1,4} \\ & A_2 & & A_{2,4} \\ & & A_3 & A_{3,4} \\ & & & S \end{bmatrix}$$

- We have that $A_{4,i} A_i^{-1} \approx \widetilde{A}_{4,i}$ and

$$S \approx A_4 - \sum \widetilde{A}_{4,i} A_i^{-1} A_{i,4}$$

Multilevel techniques

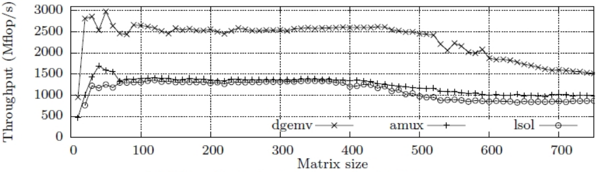
- In AMRS and IMF, a multilevel approach is adopted, where S is repeatedly treated in the same way.

$$\left[\begin{array}{c|cc} D_1 & \multicolumn{2}{c}{U_1} \\ \hline L_1 & \multicolumn{2}{c}{\underbrace{\begin{matrix} D_2 & U_2 \\ L_2 & S_2 \end{matrix}}_{S_1}} \end{array}\right] \approx \left[\begin{array}{c|cc} D_1 & \multicolumn{2}{c}{U_1} \\ \hline \widetilde{L}_1 & \multicolumn{2}{c}{\underbrace{\begin{matrix} \widetilde{D}_2 & \widetilde{U}_2 \\ \widetilde{L}_2 & \widetilde{S}_2 \end{matrix}}_{\widetilde{S}_1}} \end{array}\right]$$

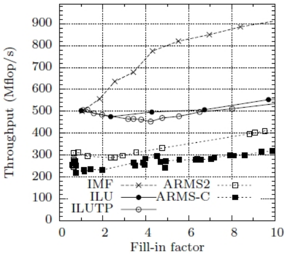
  with $\widetilde{L}_1 \approx L_1 D_1^{-1}$, etc.
- ARMS
  - D, L, U, S are sparse matrices. $\widetilde{L}_i$ is a sparse approximation of $A_{i,i+1}D_i^{-1}$.
- IMF
  - D_i is a block diagonal matrix
  - L_i is stored as a factored matrix $A_{i,i+1}D_i^{-1}$ where S_i^{-1} is stored explicitly
  - The only approximation is in the Schur complement.

Multilevel techniques

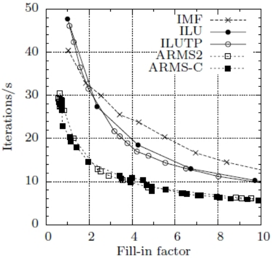
- Sparse arithmetic is less efficient than dense arithmetic.
- ARMS uses sparse matric operations
- In IMF, D_i is stored as a dense block diagonal matrix



(a) Throughput of computational primitives.



(b) Number of operations of a BiCgSTAB iteration per second.



(c) Number of BiCgSTAB iterations per second.