Incomplete factorization preconditioners and their updates with applications - I[1,2]

Daniele Bertaccini, Fabio Durastante

Moscow – August 24, 2016

[1] Università di Roma Tor Vergata

[2] Università degli Studi dell'Insubria
Department of Science and High Technology

Notes of the course: *"Incomplete factorization preconditioners and their updates with applications"*.
Lesson 1. Sparse Matrices, M-Matrices, Preconditioning, Incomplete LU

An "A^{-1}" in a formula almost always means "solve a linear system" and almost never means "compute A^{-1}."

Golub–Van Loan

This set of lectures is dedicated to the treatment of a class of algorithms for the calculation of an *incomplete factorization* of a large square matrix which in many cases will also be **sparse**[3] or with some **decaying properties**. But let us proceed step by step. The presentation of these topics is derived from the one that will be in Bertaccini and Durastante [2017].

[3] A first attempt of definition:

"The matrix may be sparse, either with the non-zero elements concentrated on a narrow band centered on the diagonal or alternatively they may be distributed in a less systematic manner. We shall refer to a matrix as dense if the percentage of zero elements or its distribution is such as to make it uneconomic to take advantage of their presence."

[Wilkinson and Reinsch, 1971]

In all the following considerations our model problem will be the solution of a **system of linear algebraic equation** of the form

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{R}^{N\times N}, \quad \mathbf{x}, \mathbf{b} \in \mathbb{R}^N, \mathrm{nnz}(A) \in O(N)\; N \to +\infty. \qquad (1)$$

Moreover, we are going to recast part o equation (1) in the following definition

Definition 1: Sparse Matrix

Given a matrix $A \in \mathbb{R}^{N\times N}$ we say that A is **sparse** if and only if the number of element of A that is different from zero, i.e., $\mathrm{nnz}(A)$, is a Big-Oh of N for $N \to +\infty$, i.e., $\mathrm{nnz}(A) \in O(N)$ for $N \to +\infty$.

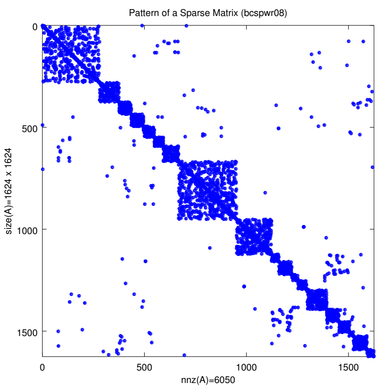


Figure 1: This is the so-called pattern of a generic sparse matrix A, that is a picture in which the element $a_{i,j}$ of the matrix are represented by a dot at coordinate (i, j) if and only if $a_{i,j} \neq 0$.
Both in MATLAB and OCTAVE this kind of picture can be obtained by using the command `spy(A)`.

Therefore we are going to deal with the solution of systems of algebraic linear equations, *possibly with sequences of them*. We are interested in solving this problems by means of **Krylov Subspace Methods**, a popular class of iterative solvers whose core operation, from the computational point of view, is represented by the matrix-vector product. This methods try to build a solution $\mathbf{x} = A^{-1}\mathbf{b}$ in an adaptive way in a particular subspace of $\mathbb{R}^N$. We are going to think the general information on this methods as a prerequisite or, if not, as a black-box in which we are going to put inside our **preconditioners**, that will be the focus of our lessons.

So we are looking for a non singular sparse transformation, represented by a matrix M, for preprocessing the system in such a way to have better convergence properties for the iterative methods.

We could perform a *left preconditioning* scheme:

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}, \qquad (2)$$

a *right preconditioning* scheme:

$$\begin{cases} AM^{-1}\mathbf{u} = \mathbf{b}, \\ \mathbf{x} = M^{-1}\mathbf{u}. \end{cases} \tag{3}$$

or a *split preconditioning* scheme, for which we seek the matrix M in a factorized form $M = M_1 M_2$:

$$\begin{cases} M_1^{-1}AM_2^{-1}\mathbf{u} = M_1^{-1}\mathbf{b}, \\ \mathbf{x} = M_2^{-1}\mathbf{u}. \end{cases} \tag{4}$$

However, what kind of transformation are we looking for? We want a transformation that is sparse, namely that is represented by a sparse matrix, that is algorithmically easy to achieve to use for *matrix-vector* product and that give us better convergence, in term of speed and numerical stability. The first problem in this search is that our requirements are mutually contradictory, so let's start looking at them singularly in search of some kind of relaxation or compromise.

Clearly we have to account the time for computing our transformation and the time needed for the application to the matrix A of our preconditioner M^{-1}. Observe that the application is not intended as computing the matrix-matrix product, too expensive to be taken into account, but is intended as computing the effect of the application of that product on a generic column vector. Summarizing the above, we can express the time required to calculate the solution T_{slv} of the linear system $A\mathbf{x} = \mathbf{b}$ with a preconditioned iterative method as:

$$T_{\text{slv}} = T_{\text{setup}} + N_{\text{it}} \times T_{\text{it}}, \tag{5}$$

where T_{setup} is the time for computing our transformation, N_{it} is the number of iteration of the iterative solver needed to obtain the solution within the correct tolerance and T_{it} is the time needed for each of the iteration.

Now we can express the first issue of our demands, we have to find a balance between the request of having M similar to A, i.e. $M^{-1}A \approx I$, and the increasing of the T_{setup} and the T_{it}.

At last we have to discuss in more detail the meaning of the increasing the convergence rate of the iterative system, namely the decrease of the number of iterations needed to obtain the desired tolerance.

To have an accurate bound for the residual, i.e. an estimate for the number of iteration, we have to introduce the concept of (*strict*) **cluster of eigenvalues**.

Definition 2: [Tyrtyshnikov, 1997] – Cluster (strict)

A sequence of matrices $\{A_n\}_{n\geq 0}$, $A_n \in \mathbb{C}^{n\times n}$, has a **strict cluster** of eigenvalues in $p \in \mathbb{C}$ if, $\forall \varepsilon > 0$, if the number of eigenvalues of A_n **not in** $D(p,\varepsilon) = \{z \in \mathbb{C} \mid |z - p| < \varepsilon\}$ is limited by a constant r that does not depend on n. Eigenvalues not in the *strict cluster* are called **outlier** eigenvalues.

Krylov-subspace iterative methods are constructed as projection methods onto an m-dimensional sub-spaces $\mathcal{K}$ of $\mathbb{R}^n$ and orthogonal to another m-dimensional subspace of $\mathbb{R}^n$ called $\mathcal{L}$. The approximate solution $\tilde{\mathbf{x}}$ of $A\mathbf{x} = \mathbf{b}$ is generated imposing:

$$\tilde{\mathbf{x}} \in \mathbf{x}^{(0)} + \mathcal{K}, \quad \text{such that } \mathbf{b} - A\tilde{\mathbf{x}} \perp \mathcal{L},$$

where $\mathbf{x}^{(0)}$ is our starting guess for the solution. In this way the approximate solution is defined as:

$$\begin{cases} \tilde{\mathbf{x}} = \mathbf{x}^{(0)} + \delta, & \delta \in \mathcal{K}, \\ < \mathbf{r}^{(0)} - A\delta, \mathbf{w} >= 0, & \forall \mathbf{w} \in \mathcal{L}. \end{cases},$$

where $\mathbf{r}^{(0)}$ is the first residual, $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$, and the orthogonality condition is imposed on the new residual $\mathbf{r}_{\text{new}} = \mathbf{r}^{(0)} - A\delta$. Then at each step of the projection methods another couple of sub-spaces is generated, using the data of the precedent steps, and a new δ is generated.

To give the construction of the methods of our interest we define the **Krylov subspace** of order m generated by the matrix $A \in \mathbb{R}^{n\times n}$ and the residual vector $\mathbf{r}^{(0)}$ as $\mathcal{K}_m(A, \mathbf{r}^{(0)})$:

$$\text{Span}\left\{\mathbf{r}^{(0)}, A\mathbf{r}^{(0)}, A^2\mathbf{r}^{(0)}, \ldots, A^{m-1}\mathbf{r}^{(0)}\right\}.$$

That is taken as the $\mathcal{K} = \mathcal{K}_m(A, \mathbf{r}^{(0)})$ space of the projection method, while the different choices of the subspace $\mathcal{L}$, and the way of preconditioning the system, make the different method.

Building the approximation of $\tilde{\mathbf{x}}$ with $\mathcal{K}_m$ we have that at each step the built approximation is of the form:

$$A^{-1}\mathbf{b} \approx \mathbf{x}^{(m)} = \mathbf{x}^{(0)} + q_{m-1}(A)\mathbf{r}^{(0)}$$

with $q_{m-1}(x) \in \mathbb{R}_{m-1}[x]$ polynomials with real coefficients and degree less or equal to $m-1$.

As a general point of view we could says that **reducing the number of iteration is closely related to altering the spectral property of the sparse matrix** $A \in \mathbb{R}^{n\times n}$ of the system.

For a complete discussion on the subject see [Saad, 2003, Bertaccini et al., 2013, Bertaccini and Durastante, 2017, Olshanskii and Tyrtshnikov, 2014].

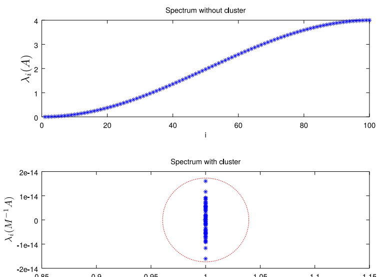


Figure 2: Plots of the spectrum of the Discrete 1D Laplacian and of its preconditioned version.

If we can certify the presence of a (strict) cluster in the spectrum of the matrices of our linear systems we can have that the convergence of our underlying Krylov Method will be improved (excepting some particular situation regarding highly non normal matrices and some other particular classes).

The Incomplete LU Factorization

One of the most popular technique for solving linear systems in the dense case is represented by the Gauss-elimination algorithm. Implementation of this algorithm gives rise to the LU factorization algorithm. The central idea is obtaining a factorization of the matrix A of the system as the product $A = LU$, where L and U are a lower triangular matrix and an upper triangular matrix, respectively. Then the solution of the system is obtained by backward solving the two triangular systems.

In this section we are going to illustrate some changes of this strategy to obtain sparse factors for sparse matrices. As a matter of fact computing an exact factorization of the sparse matrix A will lead us to have factors with an high level of fill in, see for example the pattern of the complete LU factorization in figure (3). This is clearly a situation we want to avoid due to the general remarks made earlier about preconditioners effectiveness: augmenting the number of nonzeros elements makes the calculation of matrix-vector products more expensive.

What we want, introducing the strategy devised in [Meijerink and van der Vorst, 1977], is to generate some sparse approximation of the LU factors of A, trading accuracy for sparsity as:

$$A = \tilde{L}\tilde{U} - R \tag{6}$$

where we put the unwanted fill-in in the residual matrix R for discarding. But how do we chose what are the elements to discard? Namely how do we chose the element to put in the R matrix? We can have two fundamental kind of choices, factorizing in respect to a fixed pattern P, i.e. we have that the (i, j) entry of the LU product is non zero if and only if the (i, j) entry is on the preassigned pattern P, or factorizing in respect to some other topological condition. While on the other hand there is the choice of factorizing with a drop tolerance ε_{TOL} that discards all the elements that are below the tolerance.

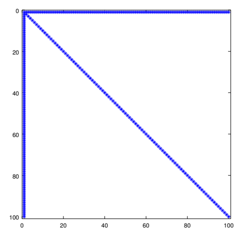
To show that this strategy is consistent we need to introduce the class of the M-matrices[4].

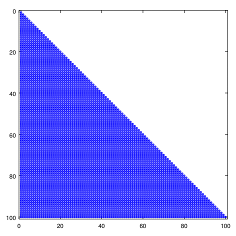
Definition 6: M-matrix 1

A matrix A is an M-matrix if it can be written as:

$$A = sI - B, \quad s > 0, \quad B \geq 0$$

with $s > \rho(B)$.


(a) A matrix


(b) L matrix

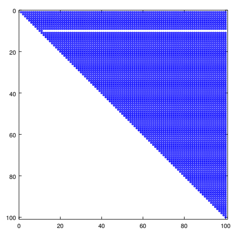

(c) U matrix

Figure 3: Full LU Factorization for the arrow matrix, with first row and column $[4, -1, \ldots, -1]$ and main diagonal $[4, 2, \ldots, 2]$.

[4] There exists many equivalent definitions of M-matrices (see [Berman and Plemmons, 1979]), other the two we have introduced here we recall also:

Definition 3: M-matrix 3

A matrix A is an M-matrix if it has all positive diagonal elements ($a_{i,i} > 0$) and non-positive off–diagonal elements ($a_{i,j} \leq 0$), and there exists a diagonal matrix D such that AD is strictly diagonally dominant, i.e.:

$$a_{i,i}d_i > \sum_{j \neq i} |a_{i,j}| d_j, \quad i = 1, \ldots, n$$

Definition 4: M-matrix 4

A matrix A is an M-matrix if all its principal submatrices are inverse positive.

Definition 5: M-matrix 5

A matrix A is an M-matrix if it has a regular splitting, i.e. $A = M - N$ with $M^{-1} \geq 0$ and $N \geq 0$ such that $\rho(M^{-1}N) < 1$.

Definition 7: *M*-matrix 2

A matrix $A \in \mathbb{R}^{n\times n}$ is called a non-singular M-matrix if:

1. $a_{i,i} > 0 \; \forall\, i = 1,2,\ldots,n$;
2. $a_{i,j} \leq 0 \; \forall\, i \neq j$, $i,j = 1,2,\ldots,n$;
3. $\det(A) \neq 0$;
4. A is inverse-positive, i.e. $A^{-1} \geq 0$.

To proceed we need now to prove that the standard algorithm for computing the LU factorization admits the possibility of inserting a dropping procedure of the extra-diagonal elements along its stages. Therefore, following [Fan, 1960], that both the algorithm of Gaussian Elimination and the dropping of extra-diagonal elements preserves M-matrices.

Theorem 1

Gaussian elimination preserves the M-matrix property.

Proof

We start observing that definition (4) implies that all principal submatrices of an M-matrix are themselves M-matrices. Therefore we can restrict ourselves, without loss of generality, to prove that if $L^{(1)}$ is an elementary Gauss transformation on A, then $A^{(1)} = L^{(1)}A$ is still an M-matrix. Now, we have that:

$$L^{(1)} = \begin{pmatrix} 1 & & & \\ -\frac{a_{2,1}}{a_{1,1}} & 1 & \mathbf{O} & \\ -\frac{a_{3,2}}{a_{1,1}} & & \ddots & \\ \vdots & \mathbf{O} & & \ddots \\ -\frac{a_{n,1}}{a_{1,1}} & & & 1 \end{pmatrix} \geq 0.$$

Moreover, $A^{(1)}$ is invertible as a product of invertible matrices. The elements of $A^{(1)}$ are given by:

$$\begin{cases} a^{(1)}_{i,j} = a_{i,j} - \frac{a_{i,1}}{a_{1,1}} a_{1,j} & i > 1 \\ a^{(1)}_{i,i} = a_{1,1} & i = 1 \end{cases}$$

and, by the sign properties of the M-matrix A, we have that:

$$a^{(1)}_{i,i} > 0, \text{ and } a^{(1)}_{i,j} \leq 0 \text{ for } i \neq j.$$

Let us now consider $\left(A^{(1)}\right)^{-1}$, to ensure that A is an M-matrix we need to show that $\left(A^{(1)}\right)^{-1} \geq 0$. Since the first column of $A^{(1)}$

All the construction of the Gauss-Elimination process relies on a matrix description of the zeroing process. If we suppose having $\mathbf{v} \in \mathbb{R}^n$ with $v_k \neq 0$ we can build the vector

$$\mathbf{t}^T = [\underbrace{0,\ldots,0}_{k}, t_{k+1},\ldots,t_n],$$

$$t_i = \frac{v_i}{v_k},\; i = k+1,\ldots,n,$$

and define the **elementary Gauss transformation** as the unit lower triangular matrix given by

$$M_k = I_n - \mathbf{t}\mathbf{e}_k^T.$$

Therefore given a matrix $C \in \mathbb{R}^{n\times r}$ applying a Gauss transformations amounts to the *outer product update*:

$$M_k C = (I_n - \mathbf{t}\mathbf{e}_k^T)C = C - \mathbf{t}\mathbf{C}_{k,:},$$

that can be computed in a row by row fashion. Now if we assume that $A \in \mathbb{R}^{n\times n}$ we can build $\{M_k\}_{k=1}^{n-1}$ Gauss elementary transformation such that

$$A^{(k-1)} = M_{k-1} \cdot \ldots \cdot M_1 A$$

is upper triangular in the columns that goes from 1 through $k-1$. This is possible only if the element $a^{(k-1)}_{k,k}$ is nonzero, i.e., if the so called pivots are different from zero $\{a^{(k)}_{k,k} \neq 0\}_{k=1}^{n-1}$. Then the U of the LU factorization is obtained by putting $U = A^{(n-1)}$ and $L = M_1^{-1} \cdot \ldots \cdot M_{n-1}^{-1}$ where the inverse of the elementary Gauss transformation is obtained easily as

$$M_k^{-1} = I_n + \mathbf{t}\mathbf{e}_k^T.$$

This construction is feasible if and only if the pivots are non null, that is equivalent to having that the determinant of the leading principal submatrix of A are non null, i.e., $\det A(1:k,1:k) \neq 0 \; \forall k = 1,\ldots,n-1$.

Detailed information on the implementation and the possibility of inserting a permutation (pivoting) strategy inside this algorithm can be found in [Golub and Van Loan, 1996, Chapter 3].

has only one nonzero element $a_{1,1}^{(1)} = a_{1,1}$ we have that $\left(A^{(1)}\right)^{-1}$ has only one nonzero element in the first column, which is equal to $1/a_{1,1}$ and thus:

$$\left(A^{(1)}\right)^{-1} \mathbf{e}_1 = \frac{1}{a_{1,1}} \mathbf{e}_1 > 0.$$

On the other hand we also have:

$$\left(A^{(1)}\right)^{-1} \mathbf{e}_j = A^{-1} \left(L^{(1)}\right)^{-1} \mathbf{e}_j = A^{-1} \mathbf{e}_j \geq 0, \;\; j > 1$$

and therefore we conclude: $\left(A^{(1)}\right)^{-1} \geq 0$, and so we have proved that $A^{(1)}$ is an M-matrix in the sense of definition 7.

Lemma (1) implies that in the LU decomposition of A, if A is an M-matrix, then so is U. Now we need to prove that also dropping preserves M-matrices.

Theorem 2

Dropping off-diagonal elements preserves the M-matrix property.

Proof

Using definition (4) we have that dropping $a_{i,j}$ elements for $i \neq j$ does not alter the property of M-matrix.

The two precedent lemmas (1,2) imply that an incomplete LU factorization, $A = \tilde{L}\tilde{U} - R$, if A is an M-matrix, then so is $\tilde{U}$.

Theorem 3

If A is an M-matrix the inverse of the elementary Gauss trasnformation $L^{(i)}$ is an M-matrix.

Proof

Without loss of generality we can restrict ourself to the $L^{(1)}$ matrix, it differs from the identity matrix just for the element:

$$l_{i,1}^{(1)} = -\frac{a_{i,1}}{a_{1,1}} \geq 0,$$

by the property of M matrix of A, hence $L^{(1)} \geq 0$. The inverse of $L^{(1)}$ is the $L^{(1)}$ matrix with off-diagonal element changed in sign, so $(L^{-1})_{i,1}^{(1)} \leq 0$, so the matrix fulfils the definition of M-matrix 7.

Theorem 4

If A isn an M-matrix, then so is the factor L in the LU factorization: $A = LU$.

Proof

Formalizing the Gauss elimination process as:

$$L^{(n)}L^{(n-1)} \cdot \ldots \cdot L^{(1)}A = U,$$

is straightforward writing:

$$L = \left(L^{(1)}\right)^{-1}\left(L^{(2)}\right)^{-1} \cdot \ldots \cdot \left(L^{(n)}\right)^{-1} \Rightarrow l_{i,j} = \left(L^{(j)}\right)^{-1}_{i,j}$$

and in this way L satisfies the definition of M-matrix (7).

Putting all this results together we can formalize the incomplete LU factorization existence as:

Theorem 5: Incomplete *LU*

Let A be an M-matrix and P a given nonzero pattern diagonal including, then the $a_{k,k}$ in the Gauss factorization algorithm (1) are $a_{k,k} \neq 0 \; \forall \, k$, and it produces an incomplete factorization $A = \tilde{L}\tilde{U} - R$ in which both $\tilde{L}$ and $\tilde{U}$ are nonsingular M-matrices; this is also a regular splitting.

And so we have obtained our algorithm[5] for incomplete LU factorization:

Algorithm 1: General Incomplete *LU* Factorization.

```
Input: Matrix $A = (a_{i,j})_{i,j=1,...,n}$, sparsity pattern $P$.
Output: Matrix $A$ factorized in incomplete $LU$ form.
for $i = 2, 3, \ldots, n$ do
    for $k = 1, \ldots, i-1$ do
        if $(i,k) \in P$ then
            $a_{i,k} \leftarrow a_{i,k}/a_{k,k}$;
            for $j = k+1, \ldots, n$ do
                if $(i,j) \in P$ then
                    $a_{i,j} \leftarrow a_{i,j} - a_{i,k} \cdot a_{k,j}$;
```

[5] There exists many variations of this strategy, namely, for each possible algorithm for computing LU factorization the incomplete formulation have been established. The first simple idea can be choosing P as the sparsity pattern of the matrix A, in this way we have defined the ILU(0) algorithm. For example, a more general version would be choosing a pattern P imposing only a certain level of fill-in of the matrix. At the the beginning of the factorization we can discern two kind of *level of fill*:

$$\text{lev}_{i,j} = \begin{cases} 0 & a_{i,j} \neq 0, \text{ or i = j} \\ \infty & \text{otherwise} \end{cases} . \quad (7)$$

Now we have to propagate the levels by following the pattern of access of the elimination algorithm, namely:

$$\text{lev}_{i,j} = \min\{\text{lev}_{i,j}, \text{lev}_{i,k} + \text{lev}_{k,j} + 1\}.$$

When we access the $a_{i,j}$ element modifying it as $a_{i,j} = a_{i,j} - a_{i,k} \cdot a_{k,j}$. Then the algorithm ILU(p) is obtained simply as the algorithm $ILU(P_p)$ where P_p is the sparsity pattern given by:

$$P_p = \{(i,j) \mid \text{lev}_{i,j} \leq p\}.$$

Now we are going to concern ourselves with the existence of an *incomplete LU factorization* via the using of a dropping strategy. References to this algorithm can be found in [Saad, 2003, 1994, Axelsson and A., 2001]. The principal idea of the algorithm is replacing an element with zero if it satisfies a set of dropping criteria related to the value of the element.

We can apply a dropping-rule row-wise by applying the same rule to all te elements of the row, as in algorithm (2), in which the vector **w** is a working-vector for the elements on the row.

Algorithm 2: Algorithm ILUT, row-wise.

```
Input: A sparse matrix $A \in \mathbb{R}^{n\times n}$.
Output: $L, U$ calculated with dropping.
1  for $i = 1, \dots, n$ do
2      $\mathbf{w} \leftarrow \mathbf{a}_{i,:}$;
3      for $k = 1, \dots, i-1$ and $w_k \neq 0$ do
4          $w_k \leftarrow w_k / a_{k,k}$;
5          Apply a dropping rule (1) to $w_k$;
6          if $w_k \neq 0$ then
7              $\mathbf{w} \leftarrow \mathbf{w} - w_k \cdot \mathbf{u}_{k,:}$;
8      Apply a dropping rule (2) to row $\mathbf{w}$;
9      for $j = 1, \dots, i-1$ do
10         $l_{i,j} \leftarrow w_j$;
11     for $j = i, \dots, n$ do
12         $u_{i,j} \leftarrow w_j$;
13     $\mathbf{w} \leftarrow 0$;
```

Depending on the choices of the dropping strategies in the algorithm (2) points (1) and (2) we can construct various incomplete factorization algorithms, and also the ILU(0) algorithm can be reinterpreted in this way.

Definition 8: ILUT(p,τ)

We define the ILUT(p,τ) algorithm chosing the following dropping rule:

(1) We define the relative tollerance $\tau_i = \tau \cdot \mathbf{a}_{i,:}$, then if $\omega_k < \tau_i$ $\Rightarrow \omega_k = 0$.

(2) Apply the dropping-rule to the whole vector **w**, then keep only the p largest modulus element in the L part of the row and the p largest modulus element in the U part of the row.

The diagonal elements are *never* dropped.

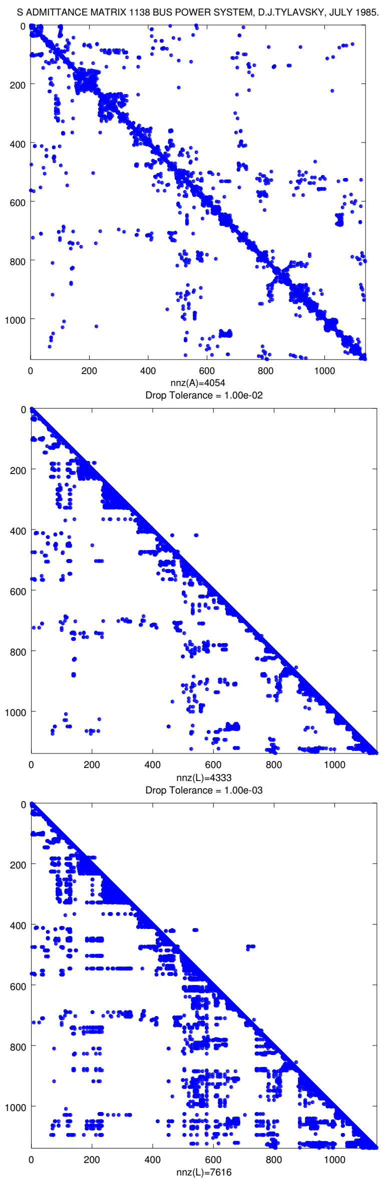


Figure 4: L factors at various drop-tolerance for the matrix 1138 Bus.

The parameter p in the algorithm controls the fill-in of the matrix, i.e. the memory usage. The idea is similar to the level of fill-in of the ILU(p) algorithm, however it relies upon the value of the elements and not upon their position.

Now we want to to prove the existence of this factorization, in a similar way to what we have done for the ILU(P), to achieve this we have to do some preliminary work. Firstly we have to define a class of matrices, similarly to what we have done with the M-matrices:

Definition 9: $\hat{M}$-matrix

A matrix $H \in \mathbb{R}^{n\times n}$ is an $\hat{M}$-matrix if it satisfies the following condition:

1. $h_{i,i} > 0 \, \forall \, 1 \leq i < n$ and $h_{n,n} \geq 0$;
2. $h_{i,j} \leq 0 \, \forall \, i,j = 1,2,\ldots,n$ and $i \neq j$;
3. $\sum_{j=i+1}^{n} h_{i,j} < 0, \forall \, 1 \leq i < n$.

Definition 10: Row Sum

Given an $\hat{M}$-matrix $H \in \mathbb{R}^{n\times n}$ we define the **row sum** of the i-th row as:

$$\operatorname{rs}(\mathbf{h}_{i,:}) = <\mathbf{h}_{i,;}, \mathbf{e}> = \sum_{j=1}^{n} h_{i,j}.$$

Definition 11: Dominance

Given an $\hat{M}$-matrix $H \in \mathbb{R}^{n\times n}$ we says that the **row $\mathbf{h}_{i,:}$** is **diagonally dominant** if $\operatorname{rs}(\mathbf{h}_{i,:}) \geq 0$. We say that an **$\hat{M}$-matrix** $H \in \mathbb{R}^{n\times n}$ is **diagonally dominant** if all its rows are diagonally dominant.

To prove the existence of the factorization we are in need to given a slight different dropping rule, selecting some particular elements that are not to be dropped in any case, namely:

Definition 12: Drop strategy II

$\forall \, i < n$ we define:

$$a_{i,j_i} = \max_{j=i+1,\ldots,n} |a_{i,j}|,$$

The elements generated in position (i, j_i) during the ILUT procedure (2) are not subject to the dropping rule (8).

We need also to establish the following notation relative to the algorithm (2):

Remark 1. *The row vector $\mathbf{w}$ resulting from line 4 of the algorithm (2) will be denoted in the following as $\mathbf{u}_{i,:}^{(k+1)}$[6]. In this way the algorithm at the generic step $k = 1,\ldots,i-1$ become:*

$$l_{i,k} = u_{i,k}^{(k)} / u_{k,k}, \tag{8}$$

if $|l_{i,k}|$ meets the dropping rule $l_{i,k} = 0$, else:

$$u_{i,j}^{(k+1)} = u_{i,j}^{(k)} - l_{i,k} \cdot u_{k,j} - r_{i,j}^{k}, \quad j = k+1,\ldots,n \tag{9}$$

[6] Note that the elements $u_{i,j}^{(k+1)} = 0$ for $j \leq k$

where, following the new notation, we have set $\boldsymbol{u}^1_{i,:} = \boldsymbol{a}_{i,:}$ *and where* $r^{(k)}_{i,j}$ *is the accounting of the dropping strategy, i.e.*

$$\begin{array}{lll} r^{(k)}_{i,j} = 0 & \Rightarrow & \textit{No dropping,} \\ r^{(k)}_{i,j} = u^{(k)}_{i,j} - l_{i,k} \cdot u_{k,j} & \Rightarrow & u^{(k+1)}_{i,j} \textit{ dropped.} \end{array}$$

in this way the i-th row of U at the i-th step of the Gaussian elimination is:

$$\boldsymbol{u}_{i,:} = \boldsymbol{u}^{(i)}_{i-1,:} \tag{10}$$

and this satisfies the relation:

$$\boldsymbol{a}_{i,:} = \sum_{k=1}^{i} l_{k,j} \cdot u^{(k)}_{i,:} + \boldsymbol{r}_{i,:}.$$

where $\boldsymbol{r}_{i,:} = (r_{i,j})_{j \leq k}$.

Theorem 6: $ILUT(p,\tau)$ **existence**

Given a matrix $A \in \mathbb{R}^{n \times n}$ such that A is a diagonally dominant $\hat{M}$-matrix (11), then the rows $\mathbf{u}^{(k)}_{i,:}$, $k = 0, 1, 2, \ldots, i$ defined by:

$$\begin{aligned} u^{(k+1)}_{i,j} &= u^{(k)}_{i,j} - l_{i,k} \cdot u_{k,j} - r^k_{i,j}, \quad j = k+1, \ldots, n; \\ \mathbf{u}^{(0)}_{i,:} &= \mathbf{0}, \\ \mathbf{u}^{(1)}_{i,:} &= \mathbf{a}_{i,:}, \end{aligned}$$

satisfy the following relation for $k = 1, \ldots, l$:

$$u^{(k)}_{i,j} \leq 0 \ \ j \neq i, \tag{11}$$

$$\mathrm{rs}(\mathbf{u}^{(k)}_{i,:}) \geq \mathrm{rs}(\mathbf{u}^{(k-1)}_{i,:}) \geq 0, \tag{12}$$

$$u^{(k)}_{i,i} > 0 \text{ when } i < n \text{ and } \mathbf{u}^k_{n,n} \geq 0. \tag{13}$$

Proof

We will prove the risult by induction over k. The result is trivially true for $k = 0$. To prove relation (11) we start from:

$$u^{(k+1)}_{i,j} = u^{(k)}_{i,j} - \underbrace{l_{i,k}}_{\leq 0} \cdot \underbrace{u_{k,j}}_{\leq 0} - \underbrace{r^k_{i,j}}_{\substack{=0, \text{ or} \\ =u^{(k)}_{i,j} - l_{i,k} \cdot u_{k,j}}} ,$$

Ando so we have $u^{(k+1)}_{i,j} \leq u^{(k)}_{i,j} \leq 0$ or, being replaced by 0, we have $u^{(k+1)}_{i,j} \leq 0$. So (11) is proved.

Now we are going to prove (12) supposing that it holds true for k. By the precedent argument we have that $\mathbf{r}^{(k)}_{i,j} = 0$ except when the j-th element in the row is dropped, in which case $u^{(k+1)}_{i,j} = 0$, and $\mathbf{r}^{(k)}_{i,j} = u^{(k)}_{i,j} - l_{i,k} \cdot u_{k,j} \leq 0$. Therefore we have thath $\mathbf{r}^{(k)}_{i,j} \leq 0$ always. Moreover, when an element in position (i,j) is not dropped, then:

$$\mathbf{u}_{i,j}^{(k+1)} := u_{i,j}^{(k)} - l_{i,k} \cdot u_{k,j} \leq u_{i,j}^{(k)},$$

and in particular by the dropping rule (12) we have that for $i < n$, we will always have for $j = j_i$: $u_{i,j_i}^{k+1} \leq u_{i,j_i}^{(k)}$. Now we can consider the row sums:

$$\begin{aligned}\mathrm{rs}(\mathbf{u}_{i,:}^{(k)}) &= \mathrm{rs}(\mathbf{u}_{i,:}^{(k)}) - l_{i,k} \cdot \mathrm{rs}(\mathbf{u}_{k,:}) - \mathrm{rs}(\mathbf{r}_{i,:}^{(k)}) \\ &\geq \mathrm{rs}(\mathbf{u}_{i,:}^{(k)}) - l_{i,k} \cdot \mathrm{rs}(\mathbf{u}_{k,:}) \\ &\geq \mathrm{rs}(\mathbf{u}_{i,:}^{(k)}),\end{aligned}$$

and this establishes (12).

It remains to prove the last relation (13), now by the previous one (12) we have that for $i < n$:

$$\begin{aligned}u_{i,i}^{(k+1)} &\geq \sum_{j=k+1}^{n} \left(-u_{i,j}^{(k+1)}\right) = \sum_{j=k+1}^{n} \left|u_{i,j}^{(k+1)}\right| \\ &\geq \left|u_{i,j_i}^{(k+1)}\right| \geq \left|u_{i,j_i}^{(k)}\right| \geq \ldots \\ &\geq \left|u_{i,j_i^{(1)}}\right| = \left|a_{i,j_i}\right|\end{aligned}$$

and by the definition of the dropping rule (12) and the property of A being an $\hat{M}$-matrices we have the proof.

In the next lesson we will use what we have built here to build another preconditioner that has the possibility of being computed and executed on high performance architectures, in this we will follow the approach in [Bertaccini and Filippone, 2016]. Moreover, for a survey of other preconditioning technique and some other informations on the one presented here see also [Benzi and Tuma, 1999].

References

O. Axelsson and Barker V. A. *Finite element solution of boundary value problems: theory and computation*. Classics in Applied Mathematics. Society for Industrial Mathematics, illustrated edition edition, 2001. ISBN 9780898714999,0898714990.

M. Benzi and M. Tuma. A comparative study of sparse approximate inverse preconditioners. *Appl. Numer. Math.*, 30 (2):305–340, 1999.

A. Berman and R. J. Plemmons. Nonnegative matrices. *The Mathematical Sciences, Classics in Applied Mathematics,*, 9, 1979.

D. Bertaccini and F. Durastante. *Iterative methods and preconditioning for large and sparse linear systems with applications*. Chapman & Hall, 2017. In Preparation.

D. Bertaccini and S. Filippone. Sparse approximate inverse preconditioners on high performance GPU platforms. *Computers & Mathematics with Applications*, 71(3):693–711, 2016. ISSN 0898-1221. `http://dx.doi.org/10.1016/j.camwa.2015.12.008`. URL `http://www.sciencedirect.com/science/article/pii/S0898122115005763`.

D. Bertaccini, C. Di Fiore, and P. Zellini. *Complessità e iterazione numerica. Percorsi, matrici e algoritmi veloci nel calcolo numerico*. Programma di mat. fisica elettronica. Bollati Boringhieri, 2013. ISBN 9788833958644.

Ky Fan. Note on m-matrices. *The Quarterly Journal of Mathematics*, 11(1):43–49, 1960.

G. H. Golub and C. F. Van Loan. *matrix computations, 3rd*. Johns Hopkins Univ Press, 1996.

J Meijerink and Henk A van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. comp.*, 31(137):148–162, 1977.

M. A Olshanskii and E. Tyrtshnikov. *Iterative methods for linear systems: theory and applications*. SIAM, 2014.

Y. Saad. Ilut: A dual threshold incomplete lu factorization. *Numerical linear algebra with applications*, 1(4):387–402, 1994. URL `http://www-users.cs.umn.edu/ saad/PDF/umsi-92-38.pdf`.

Y. Saad. *Iterative Methods for Sparse Linear Systems: Second Edition*. Society for Industrial and Applied Mathematics, 2003. ISBN 9780898718003.

E Tyrtyshnikov. *A Brief Introduction to Numerical Analysis*. Birkhauser, 1997.

J. H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation Vol. II - Linear Algebra*. Springer-verlag, 1971. URL `http://books.google.it/books?id=toZbQwAACAAJ`.