Symmetric indefinite triangular factorization revealing the rank profile matrix

Jean-Guillaume Dumas, Clément Pernet

Symmetric indefinite triangular factorization revealing the rank profile matrix*

Jean-Guillaume Dumas† Clément Pernet†

February 26, 2018

Abstract

We present a novel recursive algorithm for reducing a symmetric matrix to a triangular factorization which reveals the rank profile matrix. That is, the algorithm computes a factorization $\mathbf{P}^T\mathbf{AP} = \mathbf{LDL}^T$ where $\mathbf{P}$ is a permutation matrix, $\mathbf{L}$ is lower triangular with a unit diagonal and $\mathbf{D}$ is symmetric block diagonal with 1×1 and 2×2 antidiagonal blocks. The novel algorithm requires $O(n^2r^{\omega-2})$ arithmetic operations. Furthermore, experimental results demonstrate that our algorithm can even be slightly more than twice as fast as the state of the art unsymmetric Gaussian elimination in most cases, that is it achieves approximately the same computational speed. By adapting the pivoting strategy developed in the unsymmetric case, we show how to recover the rank profile matrix from the permutation matrix and the support of the block-diagonal matrix. There is an obstruction in characteristic 2 for revealing the rank profile matrix which requires to relax the shape of the block diagonal by allowing the 2-dimensional blocks to have a non-zero bottom-right coefficient. This relaxed decomposition can then be transformed into a standard $\mathbf{PLDL}^T\mathbf{P}^T$ decomposition at a negligible cost.

1 Introduction

Computing a triangular factorization of a symmetric matrix is a commonly used kernel to solve symmetric linear systems, or to compute the signature of symmetric bilinear forms. Besides the fact that it is expected to save half of the arithmetic cost of a standard (non-symmetric) Gaussian elimination, it can also recover invariants, such as the signature, specific to symmetric matrices, and thus, e.g., be used to certify positive or negative definite or semidefiniteness [13, Corollary 1].

*This work is partly funded by the OpenDreamKit Horizon 2020 European Research Infrastructures project (#676541).

†Université Grenoble Alpes. Laboratoire Jean Kuntzmann, CNRS, UMR 5224. 700 avenue centrale, IMAG - CS 40700, 38058 Grenoble, cedex 9 France. {firstname.lastname}@univ-grenoble-alpes.fr

It is a fundamental computation in numerical linear algebra, and is therefore most often presented in the setting of real matrices. When the matrix is positive definite, the Cholesky factorization can be defined: $\mathbf{A} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is lower triangular for which square roots of diagonal elements have to be extracted. Alternatively, gathering the diagonal elements in a central diagonal matrix yields the LDLT factorization $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ which no longer requires square roots. Similarly as for the LU decomposition, it is only defined for matrices with generic rank profile, i.e. having their $r = \text{rank}(\mathbf{A})$ first leading principal minors non-zero. For arbitrary matrices, symmetric permutations may lead to the former situations: $\mathbf{P}\mathbf{A}\mathbf{P}^T = \mathbf{L}\mathbf{D}\mathbf{L}^T$. However, this is unfortunately not always the case. For instance there is no permutation $\mathbf{P}$ such that $\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ has a LDLT factorization with a diagonal $\mathbf{D}$. This lead to a series of generalizations where the matrix $\mathbf{D}$ was replaced first by a tridiagonal symmetric matrix by Parlett and Reid [14], improved by Aasen [1], achieving half the arithmetic cost of Gaussian elimination. Bunch and Kaufman then replaced this tridiagonal matrix by a block diagonal composed of 1 or 2-dimensional diagonal blocks.

Pivoting In numerical linear algebra, the choice of the permutation matrix is mainly driven by the need to ensure a good numerical quality of the decomposition. Bunch and Parlett [5] use a full pivoting technique, requiring a cubic number of tests. Bunch and Kaufman pivoting strategy, implemented in LAPACK, uses a partial pivoting requiring only a quadratic number of tests.

In the context of exact linear algebra, for instance when computing over a finite field, numerical stability is no longer an issue. However, the computation of echelon forms and rank profiles, central in many applications, impose further constraints on the pivoting. A characterization of the requirements for the pivoting strategy is given in [10, 11] so that a PLUQ decomposition can reveal these rank profiles and echelon forms in the non-symmetric case. In particular, it is shown that pivot selection minimizing the lexicographic order on the coordinate of the pivot, combined with row and column rotations to move the pivot to the diagonal, enable the computation of the rank profile matrix, an invariant from which all rank profile information, the row and the column echelon form can be recovered.

Recursive algorithms As in numerical linear algebra, we try to gather arithmetic operations in level 3 BLAS operations (matrix multiplication based), for it delivers the best computation throughput. Numerical software often use tiled implementations, especially when the pivoting is more constrained by the symmetry [16, 12], or in order to define communication avoiding variants [4]. In exact linear algebra sub-cubic matrix multiplication, such as Strassen's algorithm, can be extensively used with no numerical instability issues. This led to the design of recursive algorithms, which was proven successful in the unsymmetric case, including for shared memory parallel computations [9].

Contribution The contribution here is to propose a recursive algorithm producing a symmetric factorization PLDLTPT over any field, from which the rank profile matrix of the input can be recovered. This algorithm is a recursive variant of Bunch and Kaufman's algorithm [6] where the pivoting strategy has been replaced by the one developped previously by the authors in the unsymmetric case [11]. Compared to the recursive adaptation of Aasen's algorihtm in [15], our algorithm leads to a similar data partitionning but does not suffer from an arithmetic overhead compared to Aasen's algorithm. Our algorithms has time complexity $O(n^2 r^{\omega-2})$ where ω is an admissible exponent for matrix multiplication and r is the rank of the input matrix. With $\omega = 3$, the leading constant in the time complexity is $1/3$, matching that of the best alternative algorithms based on cubic time linear algebra.

In Section 2 we show that in characteristic two the rank profile matrix can not always be revealed by a symmetric factorization with antidiagonal blocks: sometimes antitriangular blocks are also required. Then we recall in Section 3 the main required level 3 linear algebra subroutines. In Section 4 we present the main recursive algorithm. An alternative iterative Crout variant is presented in Section 5 to be used as a base case in the recursion. We finally show, in Section 6, experiments of the resulting implementation over a finite field. They demonstrate the efficiency of cascading the recursive algorithm with the base case variant, especially with matrices involving a lot of pivoting. They finally confirm a speed-up by a factor of about 2 compared to the state of the art unsymmetric Gaussian elimination.

2 The symmetric rank profile matrix

2.1 The pivoting matrix

Theorem 1 recalls the definition of the rank profile matrix.

Theorem 1 ([10]). *Let $\mathbf{A} \in \mathbb{F}^{m\times n}$. There exists a unique $m \times n$ $\{0,1\}$-matrix $\mathcal{R}_A$ with r 1's in rook placement of which every leading sub-matrix has the same rank as the corresponding leading sub-matrix of $\mathbf{A}$. This matrix is called the* rank profile matrix of $\mathbf{A}$.

Lemma 1. *A symmetric matrix has a symmetric rank profile matrix.*

Proof. Otherwise, the rank of some leading submatrix of $\mathbf{A}$ and the same leading submatrix of $\mathbf{A}^T$ would be different which is absurd. □

Also, any symmetric matrix has a triangular decomposition $\mathbf{A} = \mathbf{PLDL}^T\mathbf{P}^T$ where $\mathbf{L}$ is unit lower triangular, $\mathbf{D}$ is block diagonal, formed by 1-dimensional scalar blocks or 2-dimensional blocks of the form $\left[\begin{smallmatrix} 0 & x \\ x & 0 \end{smallmatrix}\right]$ and $\mathbf{P}$ a permutation matrix.

We here further define Ψ as the support matrix of $\mathbf{D}$: namely, a block diagonal $\{0,1\}$-matrix such that $\mathbf{D} = \Psi\overline{\mathbf{D}}$, with $\overline{\mathbf{D}}$ a diagonal matrix.

Definition 1. *The pivoting matrix of a PLDLTPT decomposition is the matrix* $\Pi = \mathbf{P}\Psi\mathbf{P}^T$.

Definition 2. *A PLDLTPT reveals the rank profile matrix of a symmetric matrix* $\mathbf{A}$ *if its pivoting matrix equals the rank profile matrix of* $\mathbf{A}$.

2.2 Antitriangular blocks in characteristic two

In zero or odd characteristic, we show next that one can always find such a PLDLTPT decomposition revealing the rank profile matrix. In characteristic two, however, this is not always possible.

Lemma 2. *In characteristic 2, there is no symmetric indefinite elimination revealing the rank profile matrix of* $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$.

Proof. Let $\mathbf{J}$ be the 2×2 anti-diagonal identity matrix. This is also the rank profile matrix of $\mathbf{A}$. Now, we let $\mathbf{L} = \begin{bmatrix} 1 & 0 \\ x & 1 \end{bmatrix}$, $\overline{\mathbf{D}} = \begin{bmatrix} y & 0 \\ 0 & z \end{bmatrix}$. As the permutation matrices involved, P and Ψ, can only be either the identity matrix or $\mathbf{J}$, there are then four cases:

1. $\mathbf{A} = \mathbf{L}\cdot\overline{\mathbf{D}}\cdot\mathbf{L}^T = \begin{bmatrix} y & xy \\ xy & x^2y + z \end{bmatrix}$, but $y = 0$ and $\text{rank}(\overline{\mathbf{D}}) = \text{rank}(A) = 2$ are incompatible.
2. $\mathbf{A} = \mathbf{J}\cdot\mathbf{L}\cdot\overline{\mathbf{D}}\cdot\mathbf{L}^T\cdot\mathbf{J}^T = \begin{bmatrix} x^2y + z & xy \\ xy & y \end{bmatrix}$, but $\mathbf{J}\cdot\mathbf{I}\cdot\mathbf{J} \neq \mathbf{J} = \mathcal{R}_{\mathbf{A}}$.
3. $\mathbf{A} = \mathbf{L}\cdot\overline{\mathbf{D}}\cdot\mathbf{J}\cdot\mathbf{L}^T = \begin{bmatrix} 0 & y \\ z & xy + xz \end{bmatrix}$, but we need $y = z$ for the symmetry and then $2xy = 0 \neq 1$ in characteristic 2.
4. $\mathbf{A} = \mathbf{J}\cdot\mathbf{L}\cdot\overline{\mathbf{D}}\cdot\mathbf{J}\cdot\mathbf{L}^T\cdot\mathbf{J}^T = \begin{bmatrix} xy + xz & z \\ y & 0 \end{bmatrix}$ but the bottom right coefficient of $\mathbf{A}$ is non zero.

□

However, one can generalize the PLDLTPT decomposition to a block diagonal matrix $\mathbf{D}$ having 2-dimensional blocks of the form $\left[\begin{smallmatrix} 0 & c \\ c & d \end{smallmatrix}\right]$ (lower antitriangular). Then the support matrix Ψ of $\mathbf{D}$ is the block diagonal $\{0,1\}$ matrix such that $\mathbf{D} = \Psi\overline{\mathbf{D}}$, with $\overline{\mathbf{D}}$ an upper triangular bidiagonal matrix (or equivalently such that $\mathbf{D} = \overline{\mathbf{D}}\Psi$, with $\overline{\mathbf{D}}$ lower triangular bidiagonal).

With these generalized definitions, we show in Section 4, that there exists RPM-revealing PLDLTPT decompositions.

2.3 Antitriangular decomposition

Then, such a generalized decomposition can always be further reduced to a strict PLDLTPT decomposition by eliminating each of the antitriangular blocks. For this, the observation is that in characteristic two, a symmetric lower antitriangular 2×2 block is invariant under any symmetric triangular transformation: $\begin{bmatrix}1 & \\ x & 1\end{bmatrix}\begin{bmatrix} & c\\ c & d\end{bmatrix}\begin{bmatrix}1 & x\\ & 1\end{bmatrix} = \begin{bmatrix} & c\\ c & 2cx+d\end{bmatrix} \equiv \begin{bmatrix} & c\\ c & d\end{bmatrix} \mod 2 = \begin{bmatrix}1 & \\ & 1\end{bmatrix}\begin{bmatrix} & c\\ c & d\end{bmatrix}\begin{bmatrix}1 & \\ & 1\end{bmatrix}$. Thus for each 2×2 block in a tridiagonal decomposition, the corresponding 2×2 diagonal block in $\mathbf{L}$ can be replaced by $\mathbf{I}_2$, via a multiplication by $\left[\begin{smallmatrix}1 & \\ -x & 1\end{smallmatrix}\right]$.

Further, we have that: $\left[\begin{smallmatrix} & c\\ c & d\end{smallmatrix}\right] = \mathbf{J}\left[\begin{smallmatrix}1 & \\ c/d & 1\end{smallmatrix}\right]\left[\begin{smallmatrix}d & \\ & -c^2/d\end{smallmatrix}\right]\left[\begin{smallmatrix}1 & c/d\\ & 1\end{smallmatrix}\right]\mathbf{J}$. Now $\mathbf{J}$ commutes with the identity $\mathbf{I}_2$ matrix. Therefore we have that: $\left[\begin{smallmatrix}1 & \\ x & 1\end{smallmatrix}\right]\left[\begin{smallmatrix}1 & \\ -x & 1\end{smallmatrix}\right]\mathbf{J}\left[\begin{smallmatrix}1 & \\ c/d & 1\end{smallmatrix}\right] = \mathbf{J}\left[\begin{smallmatrix}1 & \\ x & 1\end{smallmatrix}\right]\left[\begin{smallmatrix}1 & \\ -x & 1\end{smallmatrix}\right]\left[\begin{smallmatrix}1 & \\ c/d & 1\end{smallmatrix}\right]$.

Thus, to eliminate the antitriangular blocks, create a triangular matrix $\mathbf{L}_J$ that starts as the identity and where its $i, i+1$ blocks corresponding to a $\left[\begin{smallmatrix}1 & \\ x & 1\end{smallmatrix}\right]$ block in $\mathbf{L}$ is a $\left[\begin{smallmatrix}1 & \\ c/d-x & 1\end{smallmatrix}\right]$ block (associated to an antitriangular $\left[\begin{smallmatrix} & c\\ c & d\end{smallmatrix}\right]$ block, with $d\neq 0$, in $\mathbf{D}$). Then replace the triangular matrix $\mathbf{L}$ by $\tilde{\mathbf{L}} = \mathbf{L}\cdot\mathbf{L}_J$. Also, modify the diagonal matrix $\mathbf{D}$, to $\tilde{\mathbf{D}}$ such that the $\left[\begin{smallmatrix} & c\\ c & d\end{smallmatrix}\right]$ blocks of $\mathbf{D}$ are replaced by $\left[\begin{smallmatrix}d & \\ & -c^2/d\end{smallmatrix}\right]$ blocks in $\tilde{\mathbf{D}}$. Finally, create a permutation matrix $\mathbf{P}_J$, starting from the identity matrix, where each identity block at position $i, i+1$ corresponding to an antitriangular block in $\mathbf{D}$ is replaced by $\mathbf{J}$. Then $\tilde{\mathbf{P}} = \mathbf{P}\cdot\mathbf{P}_J$.

From this we have now a symmetric PLDLTPT factorization, $A = \tilde{\mathbf{P}}\tilde{\mathbf{L}}\tilde{\mathbf{D}}\tilde{\mathbf{L}}^T\tilde{\mathbf{P}}^T$, with purely 1×1 and 2×2 antidiagonal blocks in $\tilde{\mathbf{D}}$ (but then a direct access to the rank profile matrix, $\mathbf{P}\Psi\mathbf{P}^T$, might not be possible from $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{D}}$).

In the following we present some building blocks and then algorithms computing RPM-revealing symmetric indefinite triangular factorization.

3 Building blocks

We recall here some of the standard algorithms from the BLAS3 [7] and LAPACK [2] interfaces and generalization thereof [3], which will be used to define the main block recursive symmetric eliminating algorithm.

`GEMM` $(\mathbf{C}, \mathbf{A}, \mathbf{B})$**:** general matrix multiplication. Computes $\mathbf{C} \leftarrow \mathbf{C} - \mathbf{AB}$.

`TRMM` $(\mathbf{U}, \mathbf{B})$**:** multiply a triangular and a rectangular matrix in-place. Computes $\mathbf{B} \leftarrow \mathbf{UB}$ where $\mathbf{B}$ is $m \times n$ and $\mathbf{U}$ is upper or lower triangular.

`TRMM` $(\mathbf{C}, \mathbf{U}, \mathbf{B})$**:** multiply a triangular and a rectangular matrix. Computes $\mathbf{C} \leftarrow \mathbf{C} - \mathbf{UB}$ where $\mathbf{B}$ and $\mathbf{C}$ are $m \times n$ and $\mathbf{U}$ is upper or lower triangular. This is an adaptation of the BLAS3 `TRMM` to leave the $\mathbf{B}$ operand unchanged.

`TRSM` $(\mathbf{U}, \mathbf{B})$**:** solve a triangular system with matrix right hand-side. Computes $\mathbf{B} \leftarrow \mathbf{U}^{-1}\mathbf{B}$ where $\mathbf{B}$ is $m \times n$ and $\mathbf{U}$ is upper or lower triangular.

`SYRDK` $(\mathbf{C}, \mathbf{A}, \mathbf{D})$**:** symmetric rank k update with diagonal scaling. Computes the upper or lower triangular part of the symmetric matrix $\mathbf{C} \leftarrow \mathbf{C} - \mathbf{ADA}^T$ where $\mathbf{A}$ is $n \times k$ and D is diagonal or block diagonal.

`SYRD2K` $(\mathbf{C}, \mathbf{A}, \mathbf{D}, \mathbf{B})$**:** symmetric rank $2k$ update with diagonal scaling. Computes the upper or lower triangular part of the symmetric matrix $\mathbf{C} \leftarrow \mathbf{C} - \mathbf{ADB}^T - \mathbf{BDA}^T$ where $\mathbf{A}$ and $\mathbf{B}$ are $n \times k$, $\mathbf{B}$ and $\mathbf{D}$ is diagonal or block diagonal.

In addition, we need to introduce the `TRSSYR2K` routine solving Problem 1.

Problem 1. *Let $\mathbb{F}$ be a field of characteristic different than 2. Given a symmetric matrix $\mathbf{C} \in \mathbb{F}^{n\times n}$ and a unit upper triangular matrix $\mathbf{U} \in \mathbb{F}^{n\times n}$, find an upper triangular matrix $X \in \mathbb{F}^{n\times n}$ such that $\mathbf{X}^T\mathbf{U} + \mathbf{U}^T\mathbf{X} = \mathbf{C}$.*

In characteristic 2, the diagonal of $X^T\mathbf{U}+\mathbf{U}^TX$ is always zero for any matrix X and $\mathbf{U}$, hence Problem 1 has no solution as soon as $\mathbf{C}$ has a non-zero diagonal element.

However in characteristic zero or odd, Algorithm 1 presents a recursive implementation of this routine, and is in the same time a constructive proof of the existence of such a solution. Note that it performs a division by 2 in line 2, and therefore requires that the base field has not characteristic two.

Algorithme 1 `TRSSYR2K` $(\mathbf{U}, \mathbf{C})$

Require: $\mathbf{U}$, $n \times n$ full-rank upper triangular
Require: $\mathbf{C}$, $n \times n$, symmetric
Ensure: $\mathbf{C} \leftarrow \mathbf{X}$ where $\mathbf{X}$ is $n \times n$ upper triangular, such that $\mathbf{X}^T\mathbf{U} + \mathbf{U}^T\mathbf{X} = \mathbf{C}$.

1: **if** $m = 1$ **then**
2: $\quad \mathbf{C}_{1,1} \leftarrow \frac{1}{2}\mathbf{C}_{1,1}\cdot\mathbf{U}_{1,1}^{-1}$; **return**
3: **end if**
4: Splitting $\mathbf{C} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{C}_2 \\ \mathbf{C}_2^T & \mathbf{C}_3 \end{bmatrix}$, $U = \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \\ & \mathbf{U}_3 \end{bmatrix}$ where $\mathbf{C}_1$ and $\mathbf{U}_1$ are $\lfloor \frac{n}{2} \rfloor \times \lfloor \frac{n}{2} \rfloor$.
5:
6: Find $\mathbf{X}_1$ s.t. $\mathbf{X}_1^T\mathbf{U}_1 + \mathbf{U}_1^T\mathbf{X}_1 = \mathbf{C}_1$ $\quad \{$`TRSSYR2K` $(\mathbf{U}_1, \mathbf{C}_1)\}$
7: $\mathbf{D}_2 \leftarrow \mathbf{C}_2 - \mathbf{X}_1^T\mathbf{U}_2$ $\quad \{$`TRMM` $(\mathbf{X}_1^T, \mathbf{U}_2)\}$
8: $\mathbf{X}_2 \leftarrow \mathbf{U}_1^{-T}\mathbf{D}_2$ $\quad \{$`TRSM` $(\mathbf{U}_1^T, \mathbf{D}_2)\}$
9: $\mathbf{D}_3 \leftarrow \mathbf{C}_3 - (\mathbf{X}_2^T\mathbf{U}_2 + \mathbf{U}_2^T\mathbf{X}_2)$ $\quad \{$`SYRD2K` $(\mathbf{X}_2, \mathbf{U}_2)\}$
10: Find $\mathbf{X}_3$ s.t. $\mathbf{X}_3^T\mathbf{U}_3 + \mathbf{U}_3^T\mathbf{X}_3 = \mathbf{D}_3$ $\quad \{$`TRSSYR2K` $(\mathbf{U}_3, \mathbf{D}_3)\}$

Remark 1. *Note that algorithm 1 computes the solution X in place on the symmetric storage of C: by induction $\mathbf{X}_1$ and $\mathbf{X}_3$ overwrite $\mathbf{C}_1$ and $\mathbf{C}_3$, and $\mathbf{X}_2$ overwrites $\mathbf{C}_2$ according to the specifications of the generalized `TRMM` routine.*

Lemma 3. *Algorithm `TRSSYR2K` is correct and runs in $O(n^\omega)$ arithmetic operations.*

Proof. Using the notations of Algorithm 1, let $\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 \\ & \mathbf{X}_3 \end{bmatrix}$. Then exanding $\mathbf{X}^T\mathbf{U} + \mathbf{U}^T\mathbf{X}$ gives

$$\begin{bmatrix} \mathbf{X}_1^T\mathbf{U}_1 + \mathbf{U}_1^T\mathbf{X}_1 & \mathbf{X}_1^T\mathbf{U}_2 + \mathbf{U}_1^T\mathbf{X}_2 \\ (\mathbf{X}_1^T\mathbf{U}_2 + \mathbf{U}_1^T\mathbf{X}_2)^T & \mathbf{X}_2^T\mathbf{U}_2 + \mathbf{U}_2^T\mathbf{X}_2 + \mathbf{X}_3^T\mathbf{U}_3 + \mathbf{U}_3^T\mathbf{X}_3 \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{C}_1 & \mathbf{C}_2 \\ \mathbf{C}_2^T & \mathbf{C}_3 - \mathbf{D}_3 + \mathbf{X}_3^T\mathbf{U}_3 + \mathbf{U}_3^T\mathbf{X}_3 \end{bmatrix} = \mathbf{C}.$$

which proves the correctin by induction. The arithmetic cost satisfy a recurrence of the form $T(n) = 2T(n/2) + Cn^\omega$ and is therefore $T(n) = O(n^\omega)$. □

4 A block recursive algorithm

4.1 Sketch of the recursive algorithm

The design of a block recursive algorithm is based on the generalization of the 2×2 case into a block 2×2 block algorithm. While scalars could be either 0 or invertible, the difficulty in elimination algorithms, is that a submatrix could be rank defficient but non-zero. We start here an overview of the recursive algorithm by considering that the leading principal block is either all zero or invertible. We will later give the general presentation of the algorithm where its rank could be arbitrary.

Let $\mathbf{M} \in \mathbb{F}^{(m+n)\times(m+n)}$ be the symmetric matrix to be factorized. Consider its block decomposition $\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix}$ where $\mathbf{A} \in \mathbb{F}^{m\times m}$ and $\mathbf{C} \in \mathbb{F}^{n\times n}$ are also symmetric.

If $\mathbf{A}$ is full rank, then a recursive call will produce $\mathbf{A} = \mathbf{PLDL}^T P^T$, and $\mathbf{M}$ can thus be decomposed as:

$$\mathbf{M} = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{G} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & Z \end{bmatrix} \begin{bmatrix} \mathbf{L}^T & \mathbf{G}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{P}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix},$$

where $\mathbf{G}$ is such that $\mathbf{PLDG}^T = \mathbf{B}$ and $\mathbf{Z} = \mathbf{C} - \mathbf{GDG}^T$. Thus $\mathbf{G}$ can be computed as the transpose of $\mathbf{D}^{-1}\mathbf{L}^{-1}\mathbf{P}^{-1}\mathbf{B}$ which can be obtained by a call to `TRSM`, some permutations and a diagonal scaling. Then $\mathbf{Z}$ is computed by a call to `SYRDK`. A second recursive call will then decompose $\mathbf{Z}$ and lead to the final factorization of $\mathbf{M}$.

Now if $\mathbf{A}$ is the zero matrix, one is reduced to factorize the matrix $\mathbf{N} = \left[\begin{smallmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{smallmatrix}\right]$. In order to recover the rank profile matrix, one has to first look for pivots in $\mathbf{B}$ before considering the block $\mathbf{C}$. Therefore diagonal pivoting is not an option here. Then the matrix $\mathbf{B}$, which we assume has full rank for the moment, can be decomposed in a $PLDUQ$ factorization ($\mathbf{P}$ and $\mathbf{Q}$ permutation matrices, $\mathbf{L}$ and $\mathbf{U}$ respectively unit lower and unit upper triangular, $\mathbf{D}$ is diagonal). We then need to distinguish two cases depending on whether the field characteristic is two or not.

4.1.1 Zero or odd characteristic case

If the characteristic zero or odd, $\mathbf{N}$ can thus be decomposed as:

$$\mathbf{N} = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^T \end{bmatrix} \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{G} & \mathbf{U}^T \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{D} \\ \mathbf{D} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{L}^T & \mathbf{G}^T \\ \mathbf{0} & \mathbf{U} \end{bmatrix} \begin{bmatrix} \mathbf{P}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} \end{bmatrix},$$

where $\mathbf{G}$ is such that $\mathbf{Q}^T(\mathbf{G}D\mathbf{U} + \mathbf{U}^T D\mathbf{G}^T)\mathbf{Q} = C$. To compute $\mathbf{G}$, one can first permute C to get $C' = \mathbf{Q}C\mathbf{Q}^T$ (which remains symmetric) and then use a call to `TRSSYR2K`.

4.1.2 Characteristic two case

In characteristic two, the equation $\mathbf{G}D\mathbf{U} + \mathbf{U}^T D\mathbf{G}^T = \mathbf{Q}C\mathbf{Q}^T$ in unknown $\mathbf{G}$ has in general no solution (as soon as $\mathbf{C}$ has a non-zero diagonal element).

However, one can still relax Problem 1 and allow the elimination to leave a diagonal of elements not zeroed out. Following Lemma 2, the idea is then to decompose $\mathbf{N}$ into a block tridiagonal form:

$$\mathbf{N} = \begin{bmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^T \end{bmatrix} \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{G} & \mathbf{U}^T \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{D} \\ \mathbf{D} & \Delta \end{bmatrix} \begin{bmatrix} \mathbf{L}^T & \mathbf{G}^T \\ \mathbf{0} & \mathbf{U} \end{bmatrix} \begin{bmatrix} \mathbf{P}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} \end{bmatrix},$$

where Δ is a diagonal matrix and now $\mathbf{G}$ is such that $\mathbf{Q}^T(\mathbf{G}D\mathbf{U} + \mathbf{U}^T\mathbf{D}\mathbf{G}^T + \mathbf{U}^T\Delta\mathbf{U})\mathbf{Q} = C$. Therefore Δ can be chosen such that the diagonal of $C'' = \mathbf{Q}C\mathbf{Q}^T - \mathbf{U}^T\Delta\mathbf{U} = C' - \mathbf{U}^T\Delta\mathbf{U}$ is zero. As $\mathbf{U}$ is unit upper triangular, a simple pass over its coefficients is sufficient to find such a Δ: let $\Delta_{ii} = \mathbf{C'}_{ii} - \sum_{j=1}^{i-1} \Delta_{jj}\mathbf{U}_{j,i}^2$. The algorithm is thus to permute $\mathbf{C}$ to get $\mathbf{C'}$; then compute Δ with the recursive relation above and update $\mathbf{C''} = \mathbf{C'} - \mathbf{U}^T\Delta\mathbf{U}$ with a `SYRDK`. $\mathbf{C''}$ remains symmetric but with a zero diagonal and now `TRSSYR2K` can be applied.

4.2 The actual recursive algorithm

4.2.1 First phase: recursive elimination

In the general case, the leading matrices are not full rank, and we have to consider intermediate steps. For the symmetric matrix $M \in \mathbb{F}^{(m+n)\times(m+n)}$ of Section 4.1, its leading principal block $\mathbf{A} \in \mathbb{F}^{m\times m}$ is of rank $r \leq m$. Thus its actual recursive decomposition is of the form:

$$\mathbf{A} = \mathbf{P}_1 \begin{bmatrix} \mathbf{L}_1 \\ \mathbf{M}_1 \end{bmatrix} \begin{bmatrix} \mathbf{D}_1 \end{bmatrix} \begin{bmatrix} \mathbf{L}_1^T & \mathbf{M}_1^T \end{bmatrix} \mathbf{P}_1^T,$$

where $\mathbf{L}_1 \in \mathbb{F}^{r\times r}$ is full rank unit lower triangular, $\mathbf{D}_1 \in \mathbb{F}^{r\times r}$ is block diagonal with 1 or 2-dimensional diagonal blocks, and $\mathbf{M}_1 \in \mathbb{F}^{(m-r)\times r}$. Therefore, forgetting briefly the permutations, the decomposition of M becomes:

$$\mathbf{M} = \begin{bmatrix} \mathbf{L}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{M}_1 & \mathbf{I} & \mathbf{0} \\ \mathbf{G} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{D}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Y} \\ \mathbf{0} & \mathbf{Y}^T & \mathbf{Z} \end{bmatrix} \begin{bmatrix} \mathbf{L}_1^T & \mathbf{M}_1^T & \mathbf{G}^T \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix},$$

where $\mathbf{Y}$ is such that $\mathbf{B} = \begin{bmatrix}\mathbf{L}_1\\ \mathbf{M}_1\end{bmatrix} \mathbf{D}_1\mathbf{G}^T + \begin{bmatrix}\mathbf{0}\\ \mathbf{Y}\end{bmatrix}$.

From this point on, there remains to factorize the submatrix $\left[\begin{smallmatrix}\mathbf{0} & \mathbf{Y}\\ \mathbf{Y}^T & \mathbf{Z}\end{smallmatrix}\right]$. This will be carried out by the algorithm described in the next section, working on a matrix with a zero leading principal submatrix. Supposing for now that this is possible, Algorithm 2 summarizes the whole procedure.

Algorithme 2 Recursive symmetric indefinite elimination

Require: $\mathbf{A} \in \mathbb{F}^{m\times m}$ and $\mathbf{C} \in \mathbb{F}^{n\times n}$ both symmetric, $\mathbf{B} \in \mathbb{F}^{m\times n}$.
Ensure: $\mathbf{P}$ permutation, $\mathbf{L}$ unit lower triangular, $\mathbf{D}$ block diagonal, s.t. $\left[\begin{smallmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{B}^T & \mathbf{C}\end{smallmatrix}\right] = \mathbf{PLDL}^T\mathbf{P}^T$.

1: Decompose $\mathbf{A} = \mathbf{P}_1 \begin{bmatrix}\mathbf{L}_1\\ \mathbf{M}_1\end{bmatrix} \begin{bmatrix}\mathbf{D}_1\end{bmatrix} \begin{bmatrix}\mathbf{L}_1^T & \mathbf{M}_1^T\end{bmatrix} \mathbf{P}_1^T$ {Alg. 2}
2: let $r = \text{rank}(\mathbf{B})$ s.t. $\mathbf{L}_1, \mathbf{D}_1$ and $\mathbf{U}_1$ are $r \times r$.
3: $\mathbf{B}' = \mathbf{P}_1^T\mathbf{B}$ {PERM $(\mathbf{P}_1^T, \mathbf{B})$}
4: Split $\mathbf{B}' = \begin{bmatrix}\mathbf{B}'_1\\ \mathbf{B}'_2\end{bmatrix}$ where $\mathbf{B}'_1$ is $r \times n$.
5: $X \leftarrow \mathbf{L}_1^{-1}\mathbf{B}'_1$ {TRSM $(\mathbf{L}_1, \mathbf{B}'_1)$}
6: $\mathbf{Y} \leftarrow \mathbf{B}'_2 - \mathbf{M}_1\mathbf{X}$ {GEMM $(\mathbf{B}'_2, \mathbf{M}_1, \mathbf{X})$}
7: $\mathbf{G} \leftarrow X^T\mathbf{D}_1^{-1}$ {SCAL $(X^T, \mathbf{D}_1^{-1})$}
8: $\mathbf{Z} \leftarrow C - \mathbf{G}\mathbf{D}_1\mathbf{G}^T$ {SYRDK $(C, \mathbf{G}, \mathbf{D}_1)$}
9: Decompose $\begin{bmatrix}\mathbf{0} & \mathbf{Y}\\ \mathbf{Y}^T & \mathbf{Z}\end{bmatrix} = \mathbf{P}_2\mathbf{L}_2\mathbf{D}_2\mathbf{L}_2^T\mathbf{P}_2^T$ {Alg. 3}
10: $P \leftarrow \left[\begin{smallmatrix}\mathbf{P}_1 & 0\\ 0 & \mathbf{I}_n\end{smallmatrix}\right] \cdot \left[\begin{smallmatrix}\mathbf{I}_r & 0\\ 0 & \mathbf{P}_2\end{smallmatrix}\right]$
11: $\mathbf{N}_1 \leftarrow \mathbf{P}_2^T \begin{bmatrix}\mathbf{M}_1\\ \mathbf{G}\end{bmatrix}$ {PERM $(\mathbf{P}_2^T, \begin{bmatrix}\mathbf{M}_1\\ \mathbf{G}\end{bmatrix})$}
12: $L \leftarrow \left[\begin{array}{c|c}\mathbf{L}_1 & \\ \hline \mathbf{N}_1 & \mathbf{L}_2\end{array}\right]$
13: $D \leftarrow \left[\begin{array}{c|c}\mathbf{D}_1 & \\ \hline & \mathbf{D}_2\end{array}\right]$

4.2.2 Second phase: off-diagonal pivoting

Consider $\mathbf{N} = \left[\begin{smallmatrix}\mathbf{0} & \mathbf{B}\\ \mathbf{B}^T & \mathbf{C}\end{smallmatrix}\right]$, where $\mathbf{B} \in \mathbb{F}^{m\times n}$, with $m \leq n$, has now an arbitrary rank $r \leq m$. Then its PLDUQ decomposition is of the form

$$\mathbf{B} = \mathbf{P} \begin{bmatrix}\mathbf{L}_1\\ \mathbf{M}_1\end{bmatrix} \begin{bmatrix}\mathbf{D}_1\end{bmatrix} \begin{bmatrix}\mathbf{U}_1 & \mathbf{V}_1\end{bmatrix} \mathbf{Q},$$

with $\mathbf{D}_1$ diagonal, and $\mathbf{L}_1$ and $\mathbf{U}_1$ unit square triangular matrices, all three of order r. Then consider a conformal block decomposition of $\mathbf{QCQ}^T = \left[\begin{smallmatrix}\mathbf{C}_1 & \mathbf{C}_2\\ \mathbf{C}_2^T & \mathbf{C}_3\end{smallmatrix}\right]$ where $\mathbf{C}_1$ is $r \times r$. It remains to eliminate $\mathbf{C}_1$ and $\mathbf{C}_2$ with the pivots found in

B, which leads to the following factorization:

$$\mathbf{N} = \begin{bmatrix} \mathbf{P}_1 & \\ & \mathbf{Q}_1^T \end{bmatrix} \begin{bmatrix} \mathbf{L}_1 & 0 & 0 \\ \mathbf{M}_1 & 0 & 0 \\ \mathbf{G}_1 & \mathbf{U}_1^T & 0 \\ \mathbf{G}_2 & \mathbf{V}_1^T & \mathbf{I} \end{bmatrix} \begin{bmatrix} 0 & \mathbf{D}_1 & 0 \\ \mathbf{D}_1 & 0 & 0 \\ 0 & 0 & \mathbf{Z} \end{bmatrix} \times \begin{bmatrix} \mathbf{L}_1^T & \mathbf{M}_1^T & \mathbf{G}_1^T & \mathbf{G}_2^T \\ 0 & 0 & \mathbf{U}_1 & \mathbf{V}_1 \\ 0 & 0 & 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{P}_1^T & \\ & \mathbf{Q}_1 \end{bmatrix} \tag{1}$$

where $\mathbf{G}_1$ satisfies

$$\mathbf{U}_1^T \mathbf{D}_1 \mathbf{G}_1^T + \mathbf{G}_1 \mathbf{D}_1 \mathbf{U}_1 = \mathbf{C}_1 \tag{2}$$

and $\mathbf{G}_2 = (\mathbf{C}_2 - \mathbf{V}_1^T \mathbf{D}_1 \mathbf{G}_1^T)\mathbf{U}_1^{-1}\mathbf{D}_1^{-1}$ and $\mathbf{Z} = \mathbf{C}_3 - (\mathbf{V}_1^T \mathbf{D}_1 \mathbf{G}_2^T + \mathbf{G}_2 \mathbf{D}_1 \mathbf{V}_1)$.

In order to produce a LDLT decomposition, there still remains to perform permutations to

1. compact the leading elements of the lower triangular matrix into a $2r \times 2r$ invertible leading triangular submatrix,
2. make the $\left[\begin{smallmatrix} & \mathbf{D}_1 \\ \mathbf{D}_1 & \end{smallmatrix}\right]$ matrix block diagonal with 1 or 2-dimensional diagonal blocks.

The permutation matrix

$$\mathbf{P}_c = \begin{bmatrix} \mathbf{I}_r & 0 & 0 & 0 \\ 0 & 0 & \mathbf{I}_{m-r} & 0 \\ 0 & \mathbf{I}_r & 0 & 0 \\ 0 & 0 & 0 & \mathbf{I}_{n-r} \end{bmatrix}, \tag{3}$$

corresponding to a block circular rotation, takes care of condition 1, while preserving precedence in the non-pivot rows. This is a requirement for the factorization to reveal the rank profile matrix [11]. The decomposition becomes

$$N = \begin{bmatrix} \mathbf{P}_1 & \\ & \mathbf{Q}_1^T \end{bmatrix} \mathbf{P}_c \begin{bmatrix} \mathbf{L}_1 & 0 & 0 \\ \mathbf{G}_1 & \mathbf{U}_1^T & 0 \\ \mathbf{M}_1 & 0 & 0 \\ \mathbf{G}_2 & \mathbf{V}_1^T & \mathbf{I} \end{bmatrix} \begin{bmatrix} 0 & \mathbf{D}_1 & 0 \\ \mathbf{D}_1 & 0 & 0 \\ 0 & 0 & \mathbf{Z} \end{bmatrix} \times \begin{bmatrix} \mathbf{L}_1^T & \mathbf{G}_1^T & \mathbf{M}_1^T & \mathbf{G}_2^T \\ 0 & \mathbf{U}_1 & 0 & \mathbf{V}_1 \\ 0 & 0 & 0 & \mathbf{I} \end{bmatrix} \mathbf{P}_c^T \begin{bmatrix} \mathbf{P}_1^T & \\ & \mathbf{Q}_1 \end{bmatrix} \tag{4}$$

In order to achieve Condition 2, we will transform the matrix $\left[\begin{smallmatrix} 0 & \mathbf{D}_1 \\ \mathbf{D}_1 & 0 \end{smallmatrix}\right]$ into the block diagonal matrix $\mathrm{Diag}(\left[\begin{smallmatrix} 0 & d_i \\ d_i & 0 \end{smallmatrix}\right])$ where d_i is the ith diagonal element in $\mathbf{D}_1$. To describe the process, we will focus on the matrix

$$\mathbf{N}_2 = \begin{bmatrix} \mathbf{L}_1 & 0 \\ \mathbf{G}_1 & \mathbf{U}_1^T \end{bmatrix} \begin{bmatrix} 0 & \mathbf{D}_1 \\ \mathbf{D}_1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{L}_1^T & \mathbf{G}_1^T \\ 0 & \mathbf{U}_1 \end{bmatrix} = \overline{L} \cdot \overline{\Delta} \cdot \overline{L}^T,$$

and consider a splitting in halves of the matrix $\mathbf{D}_1 = \left[\begin{smallmatrix}\mathbf{D}_{11}\\ \mathbf{D}_{12}\end{smallmatrix}\right]$ where $\mathbf{D}_{11}$ has order r_1 and $\mathbf{D}_{12}$ order r_2. This leads to the conformal decompostion

$$\begin{bmatrix}\mathbf{L}_{11} & 0 & 0 & 0\\ \mathbf{L}_{12} & \mathbf{L}_{13} & 0 & 0\\ \mathbf{G}_{11} & \mathbf{G}_{14} & \mathbf{U}_{11}^T & 0\\ \mathbf{G}_{12} & \mathbf{G}_{13} & \mathbf{U}_{12}^T & \mathbf{U}_{13}^T\end{bmatrix}\begin{bmatrix}0 & 0 & \mathbf{D}_{11} & 0\\ 0 & 0 & 0 & \mathbf{D}_{12}\\ \mathbf{D}_{11} & 0 & 0 & 0\\ 0 & \mathbf{D}_{12} & 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{L}_{11}^T & \mathbf{L}_{12}^T & \mathbf{G}_{11}^T & \mathbf{G}_{12}^T\\ 0 & \mathbf{L}_{13}^T & \mathbf{G}_{14}^T & \mathbf{G}_{13}^T\\ 0 & 0 & \mathbf{U}_{11} & \mathbf{U}_{12}\\ 0 & 0 & 0 & \mathbf{U}_{13}\end{bmatrix}$$

Then considering the permutation matrix

$$\mathbf{P}_d = \begin{bmatrix}\mathbf{I}_{r_1} & 0 & 0 & 0\\ 0 & 0 & \mathbf{I}_{r_2} & 0\\ 0 & \mathbf{I}_{r_1} & 0 & 0\\ 0 & 0 & 0 & \mathbf{I}_{r_2}\end{bmatrix},$$

one can form $\mathbf{P}_d^T\overline{\Delta}\mathbf{P}_d = \begin{bmatrix}0 & \mathbf{D}_{11} & 0 & 0\\ \mathbf{D}_{11} & 0 & 0 & 0\\ 0 & 0 & 0 & \mathbf{D}_{12}\\ 0 & 0 & \mathbf{D}_{12} & 0\end{bmatrix}$ and $\mathbf{P}_d^T\overline{L}\mathbf{P}_d = \begin{bmatrix}\mathbf{L}_{11} & 0 & 0 & 0\\ \mathbf{G}_{11} & \mathbf{U}_{11}^T & \mathbf{G}_{14} & 0\\ \mathbf{L}_{12} & 0 & \mathbf{L}_{13} & 0\\ \mathbf{G}_{12} & \mathbf{U}_{12}^T & \mathbf{G}_{13} & \mathbf{U}_{13}^T\end{bmatrix}$.
Applying this process recursively changes $\overline{\Delta}$ into the desired block diagonal form. Then the transformation of $\overline{L}$ will remain lower triangular if and only if all $\mathbf{G}_{14}$ matrices are zero: this means that $\mathbf{G}_1$ must be lower triangular in the first place.

Finding $\mathbf{G}_1$ lower triangular satifying Equation (2), is an instance of Problem 1 for which the routine `TRSSYR2K` provides a solution.

Note that the actual permutation to transform $\left[\begin{smallmatrix}\mathbf{0} & \mathbf{D}_1\\ \mathbf{D}_1 & \mathbf{0}\end{smallmatrix}\right]$ into a 2×2-blocks diagonal matrix is a permutation matrix, $\mathbf{P}_i$, resulting from the one by one interleaving of the rows of $\left[\mathbf{I}_r\ \mathbf{0}\right]$ and $\left[\mathbf{0}\ \mathbf{I}_r\right]$. If $\mathbf{e}_i = \left[0\ldots0\ 1\ 0\ldots0\right]^T$ is the i-th canonical vector, then:

$$\mathbf{P}_i = \left[\mathbf{e}_1\ \mathbf{e}_{r+1}\ \mathbf{e}_2\ \mathbf{e}_{r+2}\ \ldots\ \mathbf{e}_r\ \mathbf{e}_{2r}\right]. \tag{5}$$

Similarly the triangular factor of the factorization is thus a one by one interleaving of the rows of $\left[\mathbf{L}_1\ \mathbf{0}\right]$ and $\left[\mathbf{G}_1\ \mathbf{U}_1^T\right]$ as well as a one by one interleaving of the columns $\left[\begin{smallmatrix}\mathbf{L}_1\\ \mathbf{G}_1\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{U}_1^T\end{smallmatrix}\right]$, which overall remains triangular.

Finally, a call to Algorithm 2 produces a factorization for the remaining Z block and a final block rotation,

$$\begin{bmatrix}\mathbf{I}_{2r} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{m-r}\\ \mathbf{0} & \mathbf{I}_{n-r} & \mathbf{0}\end{bmatrix},$$

moves the intermediate zero rows and columns to the bottom right. The full algorithm is presented in details in Algorithm 3 (for zero or odd characteristic, the characteristic two case being presented afterwards in Section 4.3).

4.3 Characteristic two

The case of the characteristic two can be handled similarly, just computing the extra diagonal and updating after the `PLDUQ` decomposition, as sketched in

Algorithme 3 Rank deficient and zero leading principal symmetric elimination

Require: $\mathbf{C} \in \mathbb{F}^{n\times n}$ symmetric and $\mathbf{B} \in \mathbb{F}^{m\times n}$.

Ensure: $\mathbf{P}$ permutation, L unit lower triangular, $\mathbf{D}$ block-diagonal, s.t. $\begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{bmatrix} = \mathbf{PLDL}^T\mathbf{P}^T$.

1: Decompose $\mathbf{B} = \mathbf{P}_B \begin{bmatrix} \mathbf{L}_1 \\ \mathbf{M}_1 \end{bmatrix} [\mathbf{D}_1] [\mathbf{U}_1 \ \ \mathbf{V}_1] Q$ {PLDUQ }

2: $\mathbf{C}' \leftarrow \mathbf{QCQ}^T = \begin{bmatrix} \mathbf{C'}_1 & \mathbf{C'}_2 \\ \mathbf{C'}_2{}^T & \mathbf{C'}_3 \end{bmatrix}$ {$\mathbf{C}'_1$ first $r = rk(\mathbf{B})$ rows/columns: PERM }

3: **if** characteristic $(\mathbb{F}) = 2$ **then**

4: **for** $i = 1$ **to** r **do**

5: $\Delta_{ii} = (\mathbf{C}'_1)_{ii} - \sum_{j=1}^{i-1} \Delta_{jj} (\mathbf{U}_1)_{j,i}^2$

6: **end for**

7: $\mathbf{C}'_1 \leftarrow \mathbf{C}'_1 - \mathbf{U}_1^T \Delta \mathbf{U}_1$ {SYRDK $(\mathbf{C}'_1, \mathbf{U}_1, \Delta)$}

8: **end if**

9: $\mathbf{X}^T\mathbf{U}_1 + \mathbf{U}_1^T\mathbf{X} = \mathbf{C}'_1$ {TRSSYR2K $(\mathbf{U}_1, \mathbf{C}'_1)$}

10: $\mathbf{G}_1 \leftarrow \mathbf{X}^T\mathbf{D}_1^{-1}$ {SCAL $(\mathbf{X}^T, \mathbf{D}_1^{-1})$}

11: **if** characteristic$(\mathbb{F}) = 2$ **then** $\mathbf{X} \leftarrow \mathbf{X} + \Delta\mathbf{U}_1$ **end if** {DADD $(X, \Delta, \mathbf{U}_1)$}

12: $\mathbf{C''}_2 \leftarrow \mathbf{C'}_2 - X^T\mathbf{V}_1$ {TRMM $(X^T, \mathbf{V}_1)$}

13: $\mathbf{Y} \leftarrow \mathbf{U}_1^{-T}\mathbf{C''}_2$ {TRSM $(\mathbf{U}_1^T, \mathbf{C''}_2)$}

14: $\mathbf{Z} \leftarrow \mathbf{C'}_3 - (\mathbf{Y}^T\mathbf{V}_1 + \mathbf{V}_1^T\mathbf{Y})$ {SYRD2K $(\mathbf{Y}, \mathbf{V}_1)$}

15: **if** characteristic$(\mathbb{F}) = 2$ **then** $\mathbf{Z} \leftarrow \mathbf{Z} - (\mathbf{V}_1^T\Delta\mathbf{V}_1)$ **end if** {SYRDK $(\mathbf{Z}, \mathbf{V}_1, \Delta)$}

16: $\mathbf{G}_2 \leftarrow \mathbf{Y}^T\mathbf{D}_1^{-1}$ {SCAL $(\mathbf{Y}^T, \mathbf{D}_1^{-1})$}

17: Decompose $\mathbf{Z} = \mathbf{P}_3\mathbf{L}_3\mathbf{D}_3\mathbf{L}_3^T\mathbf{P}_3^T$ {Alg. 2}

{With $\mathbf{P}_c$ from (3), and $\mathbf{P}_i$ from (5):}

18: $P \leftarrow \begin{bmatrix} \mathbf{P}_B & 0 \\ 0 & Q^T \end{bmatrix} \begin{bmatrix} \mathbf{P}_c & 0 \\ 0 & \mathbf{I}_{n-r} \end{bmatrix} \begin{bmatrix} \mathbf{P}_i & 0 \\ 0 & \mathbf{I}_{m+n-2r} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{m+r} & 0 \\ 0 & \mathbf{P}_3 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{2r} & 0 & 0 \\ 0 & 0 & \mathbf{I}_{m-r} \\ 0 & \mathbf{I}_{n-r} & 0 \end{bmatrix}$

19: $L \leftarrow \left[\begin{array}{ccc|c} \mathbf{P}_i^T \begin{bmatrix} \mathbf{L}_1 \\ \mathbf{G}_1 & \mathbf{U}_1^T \end{bmatrix} \mathbf{P}_i & & & \\ \hline & \mathbf{G}_2 & \mathbf{V}_1^T & \mathbf{L}_3 \\ & \mathbf{M}_1 & 0 & 0 \end{array}\right]$

20: $D \leftarrow \left[\begin{array}{ccc|c} \mathbf{P}_i^T \begin{bmatrix} 0 & \mathbf{D}_1 \\ \mathbf{D}_1 & 0 \end{bmatrix} \mathbf{P}_i & & & \\ \hline & & & \mathbf{D}_3 \end{array}\right]$

Section 4.1.2. Indeed, the only issue is the division by 2 in `TRSSYR2K`, which is removed if the diagonal of $\mathbf{C'}_1$ is zero. Therefore, Algorithm 2 is unchanged, the block diagonal matrix just has lower symmetric antitriangular 2×2 blocks instead of only antidiagonal ones. The only few additional operations appear in Algorithm 3 and are the contents of the "**if** characteristic($\mathbb{F}$) = 2..." branchings.

Then the tridiagonal form with symmetric antitriangular 2×2 blocks thus obtained by Algorithm 3 can be used to either reveal the rank profile matrix (via computing Ψ, the support matrix of D, and the pivoting matrix $\mathcal{R} = P\Psi P^T$) or a PLDLTPT factorization, both at an extra linear cost, as shown in Section 2.3.

Overall, we have proven:

Theorem 2. *Algorithm 2 correctly computes a symmetric indefinite PLDLTPT factorization revealing the rank profile matrix.*

5 Base case iterative variant

The recursion of Algorithm 2 should not be performed all the way to a dimension 1 in practice. For implementations over a finite field, it would induce an unnecessary large number of modular reductions and a significant amount of data movement for the permutations. Instead, we propose in Algorithm 4 an iterative algorithm computing a PLDLTPT revealing the rank profile matrix to be used as a base case in the recursion.

This iterative algorithm has the following features:

1. it uses a pivot search minimizing the lexicographic order (following the caracterization in [11]): if the diagonal element of the current row is 0, the pivot is chosen as the first non-zero element of the row, unless the row is all zero, in which case, it is searched in the following row;
2. the pivot is permuted with cyclic shifts on the row and columns, so as to leave the precedence in the remaining rows and columns unchanged.
3. the update of the unprocessed part in the matrix is delayed following the scheme of a Crout elimination schedule [8]. It does not only improves efficiency thanks to a better data locality, but it also reduces the amount of modular reductions, over a finite field, as shown for the unsymmetric case in [9].

We denote by $\rho_{i,n}$ the cyclic shift permutation of order n moving element i to the first position: $\rho_{i,n} = (i, 0, 1, \ldots, i-1, i+1, \ldots n-1)$. Indices are 0 based, index ranges are excluding their upper bound. For instance, $\mathbf{A}_{i,0..r}$ denotes the r first elements of the $i+1$st row of $\mathbf{A}$, and $\mathbf{A}_{0..r,0..r}$ is the 0-dimensional matrix when $r = 0$.

6 Experiments

We now report on experiments of an implementation of these algorithms in the FFLAS-FFPACK library [17], dedicated to dense linear algebra over finite fields. We used the version committed under the reference e12a998 of the master

Algorithme 4 SYTRF Crout iterative base case

Require: $\mathbf{A} \in \mathbb{F}^{n \times n}$ symmetric

Ensure: $\mathbf{P}$, a permutation, $\mathbf{L}$, unit lower triangular and $\mathbf{D}$, block diagonal, such that $\mathbf{A} = \mathbf{PLDL}^T\mathbf{P}^T$

$r \leftarrow 0$; $\mathbf{D} \leftarrow \mathbf{0}$ {Denote $\mathbf{W} = \mathbf{A}$ the working matrix}

for $i = 0..n$ **do**

Here $\mathbf{W} = \begin{bmatrix} \mathbf{L} & & & \\ \mathbf{M} & 0 & 0 & \\ \mathbf{N} & 0 & \mathbf{A}_{i..n,i} & \mathbf{A}_{i..n,i+1..n} \end{bmatrix}$ with $\mathbf{L} \in \mathbb{F}^{r \times r}$

$\mathbf{v} \leftarrow \mathbf{N}_{0,0..r} \times \mathbf{D}^{-1}_{0..r,0..r}$

$\mathbf{c} \leftarrow \mathbf{A}_{i..n,i} - \mathbf{N} \times \mathbf{v}^T$

if $\mathbf{c} = 0$ **then** Loop to next iteration **end if**

Let j be the smallest index such that $x = c_j \neq 0$

if $j = 0$ **then** {Denote $\mathbf{c} = [\mathbf{0}\ x\ \mathbf{k}]^T$}

$\left[\begin{smallmatrix} \mathbf{M} \\ \mathbf{N} \end{smallmatrix}\right] \leftarrow \rho_{j,n-r} \times \left[\begin{smallmatrix} \mathbf{M} \\ \mathbf{N} \end{smallmatrix}\right]$

$\mathbf{W}_{r..n,r} \leftarrow x^{-1} \times \rho_{j,n-r} \times \left[\begin{smallmatrix} 0 \\ \mathbf{c} \end{smallmatrix}\right]$

$\mathbf{P} \leftarrow \mathbf{P} \times \rho^T_{j,n-r}$

$\mathbf{D}_{r,r} \leftarrow x$

$r \leftarrow r + 1$

else {Crout update of the row $i + j$}

$\mathbf{w} \leftarrow \mathbf{N}_{j,0..r} \times \mathbf{D}^{-1}_{0..r,0..r}$

$\mathbf{d} \leftarrow \mathbf{A}_{i..n,j+i} - \mathbf{N} \times \mathbf{w}^T (= \begin{bmatrix} \mathbf{0} & x & \mathbf{g} & y & \mathbf{h} \end{bmatrix}^T)$

Here $\mathbf{W} = \begin{bmatrix} \mathbf{L} & & & & & \\ \mathbf{M} & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & x & \mathbf{k}^T \\ \mathbf{N} & 0 & 0 & \mathbf{F} & \mathbf{g} & \mathbf{J}^T \\ & 0 & x & \mathbf{g}^T & y & \mathbf{h}^T \\ & 0 & \mathbf{k} & \mathbf{J} & \mathbf{h} & * \end{bmatrix}$

if characteristic($\mathbb{F}$) $= 2$ **then**

$y' \leftarrow 0$

$\mathbf{h}' \leftarrow \mathbf{h} - yx^{-1}\mathbf{k}$

$\mathbf{D}_{r..r+2,r..r+2} \leftarrow \begin{bmatrix} 0 & x \\ x & y \end{bmatrix}$

else

$y' \leftarrow y/2$

$\mathbf{h}' \leftarrow \mathbf{h} - y'x^{-1}\mathbf{k}$

$\mathbf{D}_{r..r+2,r..r+2} \leftarrow \begin{bmatrix} 0 & x \\ x & 0 \end{bmatrix}$

end if

Perform cyclic symmetric row and column rotations to bring W to the form $\mathbf{W} = \left[\begin{array}{ccc|ccc} \mathbf{L} & & & & & \\ \mathbf{n}' & 1 & & & & \\ & x^{-1}y' & 1 & & & \\ \hline \mathbf{M} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{N}' & x^{-1}\mathbf{g} & 0 & 0 & \mathbf{F} & \mathbf{J}^T \\ & x^{-1}\mathbf{h}' & x^{-1}\mathbf{k} & 0 & \mathbf{J} & * \end{array}\right]$

Update $\mathbf{P}$ accordingly

$r \leftarrow r + 2$

end if

end for

branch. It was compiled with gcc-5.4 and was linked with the numerical library OpenBLAS-0.2.18. Experiments are run on a single core of an an Intel Haswell i5-4690, @3.5GHz.

Computation speed are normalized as *effective Gfops*, an estimate of the number of field operations that an algorithm with classic matrix arithmetic would perform per second, divided by the computation time. For a matrix of order n and rank r, we defined this as:

$$\text{Effective Gfops} = (r^3/3 + n^2 r - r^2 n)/(10^9 \times \text{time}).$$

All experiments are over the 23-bits finite field $\mathbb{Z}/8388593\mathbb{Z}$.

Figure 1 compares the computation speed of the pure recursive algorithm, the base case algorithm and a cascade of these two, with a threshold set to its optimum value from experiments on this machine. Remark that the pure recursive variant performs rather well with generic rank profile matrices, while matrices with uniformly random rank profile matrix make this variant very slow, due to an excessive amount of pivoting. As expected, the base case Crout variant speeds up these instances for small dimensions, but then its performance stagnate on large dimensions, due to poor cache efficiency. Lastly the cascade algorithm combines the benefits of the two variants and therefore performs best in all settings. We here used a threshold $n = 128$ for the experiments with random RPM matrices, but of only $n = 48$ for generic rank profile matrices, since the recursive variant becomes competive much earlier. In most cases, the rank profile structure of given matrices is unknown a priori, making the setting of this threshold speculative. One could instead implement an introspective strategy, updating the threshold from experimenting with running instances.

n	Gen. rank prof. $r = n$		Random RPM $r = n$		Random RPM $r = n/2$	
	PLUQ	LDLT	PLUQ	LDLT	PLUQ	LDLT
100	5.81e-4	4.95e-4	6.71e-4	5.95e-4	3.79e-4	3.69e-4
200	2.29e-3	1.25e-3	3.05e-3	1.82e-3	1.81e-3	1.23e-3
500	1.99e-2	6.57e-3	3.07e-2	1.05e-2	2.04e-2	7.54e-3
1000	0.104	2.58e-2	1.15e-1	4.25e-2	6.98e-2	3.14e-2
2000	0.507	0.134	0.551	0.199	0.308	0.148
5000	4.651	1.720	4.502	2.003	2.813	1.419
10000	26.59	11.94	26.08	15.88	12.04	8.265

Table 1: Comparing computation time (s) of the symmetric (LDLT) with unsymmetric (PLUQ) triangular decompositions. Matrices with rank r, and rank profile matrix uniformly random.

Table 1 compares the computation time of the symmetric decomposition algorithm with that of the unsymmetric case (running the PLUQ algorithm of [11]). These experiments confirm a speed-up factor of about 2 between these routines, which is the expected gain in the constant in the time complexity. Note

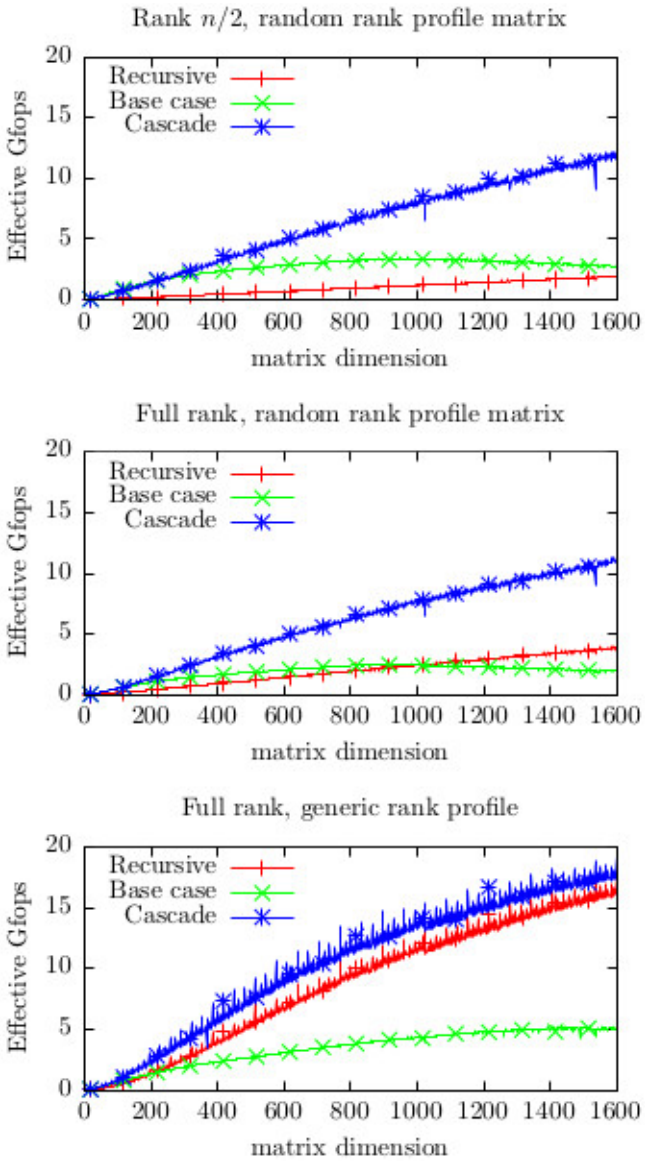

Figure 1: Computation speed of the Base Case, the pure recurisve and the cascading variant for a rank profile matrix revealing PLDLTPT decomposition. Matrices with rank half the dimension and random RPM (top), full rank with random RPM (center) or full rank with generic rank profile (bottom)

that on large instances, the PLUQ elimination performs better with random RPM instances than generic rank profiles, contrarily to the LDLT routine. This is due to the lesser amount of arithmetic operations when the RPM is random (some intermediate submatrices being rank deficient). On the other hand, these matrices generate more off-diagonal pivots, which cause more pivoting in LDLT than in PLUQ, explaining the slow down for the symmetric case.

References

[1] Jan Ole Aasen. On the reduction of a symmetric matrix to tridiagonal form. *BIT Numerical Mathematics*, 11(3):233–242, Sep 1971. doi:10.1007/BF01931804.

[2] Edward Anderson, Zhaojun Bai, Christian Bischof, L Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, et al. *LAPACK Users' guide*. SIAM, 1999. URL: http://www.netlib.org/lapack/lug/lapack_lug.html.

[3] Marc Baboulin, Dulceneia Becker, and Jack Dongarra. A Parallel Tiled Solver for Dense Symmetric Indefinite Systems on Multicore Architectures. In *IEEE 26th International Parallel & Distributed Processing Symposium (IPDPS)*, pages 14–24. IEEE, May 2012. URL: http://ieeexplore.ieee.org/document/6267820/, doi:10.1109/IPDPS.2012.12.

[4] G. Ballard, D. Becker, J. Demmel, J. Dongarra, A. Druinsky, I. Peled, O. Schwartz, S. Toledo, and I. Yamazaki. Communication-Avoiding Symmetric-Indefinite Factorization. *SIAM Journal on Matrix Analysis and Applications*, 35(4):1364–1406, January 2014. URL: http://epubs.siam.org/doi/abs/10.1137/130929060, doi:10.1137/130929060.

[5] J. R. Bunch and B. N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 8(4):639–655, December 1971. doi:10.1137/0708060.

[6] James R. Bunch and Linda Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of Computation*, 31(137):163–179, 1977. URL: http://www.jstor.org/stable/2005787, doi:10.2307/2005787.

[7] J. J. Dongarra, Jeremy Du Croz, Sven Hammarling, and I. S. Duff. A Set of Level 3 Basic Linear Algebra Subprograms. *ACM TOMS*, 16(1):1–17, March 1990. URL: http://doi.acm.org/10.1145/77626.79170, doi:10.1145/77626.79170.

[8] Jack J. Dongarra, Lain S. Duff, Danny C. Sorensen, and Henk A. Vander Vorst. *Numerical Linear Algebra for High Performance Computers*. SIAM, 1998.

[9] Jean-Guillaume Dumas, Thierry Gautier, Clément Pernet, Jean-Louis Roch, and Ziad Sultan. Recursion based parallelization of exact dense linear algebra routines for gaussian elimination. *Parallel Computing*, 57:235 – 249, 2016. doi:10.1016/j.parco.2015.10.003.

[10] Jean-Guillaume Dumas, Clément Pernet, and Ziad Sultan. Computing the rank profile matrix. In *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '15, pages 149–156, New York, NY, USA, 2015. ACM. URL: http://doi.acm.org/10.1145/2755996.2756682, doi:10.1145/2755996.2756682.

[11] Jean-Guillaume Dumas, Clément Pernet, and Ziad Sultan. Fast computation of the rank profile matrix and the generalized Bruhat decomposition. *Journal of Symbolic Computation*, 83:187–210, November–December 2017. URL: http://hal.archives-ouvertes.fr/hal-01251223, doi:10.1016/j.jsc.2016.11.011.

[12] Erik Elmroth, Fred G. Gustavson, Isak Jonsson, and Bo Kågström. Recursive blocked algorithms and hybrid data structures for dense matrix library software. *SIAM Review*, 46(1):3–45, 2004. doi:10.1137/S0036144503428693.

[13] Erich L. Kaltofen, Michael Nehring, and B. David Saunders. Quadratic-time certificates in linear algebra. In Anton Leykin, editor, *ISSAC'2011, Proceedings of the 2011ACM International Symposium on Symbolic and Algebraic Computation, San Jose, California, USA*, pages 171–176. ACM Press, New York, June 2011. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/11/KNS11.pdf.

[14] B. Parlett and J. K. Reid. On the solution of a system of linear equations whose matrix is symmetric but not definite. *BIT*, 10(3):386–397, 1970. doi:10.1007/BF01934207.

[15] Miroslav Rozložník, Gil Shklarski, and Sivan Toledo. Partitioned triangular tridiagonalization. *ACM Trans. Math. Softw.*, 37(4):38:1–38:16, February 2011. doi:10.1145/1916461.1916462.

[16] Gil Shklarski and Sivan Toledo. Blocked and recursive algorithms for triangular tridiagonalization. 2007. URL: http://www.cs.tau.ac.il/~stoledo/Bib/Pubs/ShklarskiToledo-Aasen.pdf.

[17] The FFLAS-FFPACK group. *FFLAS-FFPACK: Finite Field Linear Algebra Subroutines / Package*, 2018. v2.3.2. https://github.com/linbox-team/fflas-ffpack.