Generalized fraction-free LU factorization for singular systems with kernel extraction

David Dureisseix

Generalized fraction-free LU factorization for singular systems with kernel extraction

David Dureisseix

Université de Lyon, INSA-Lyon, LaMCoS CNRS UMR5259, F-69621 Villeurbanne CEDEX, France

Abstract

Linear systems are usually solved with Gaussian elimination. Especially when multiple right hand sides are involved, an efficient procedure is to provide a factorization of the left hand side. When exact computations are required in an integral domain, complete fraction-free factorization and forward-backward substitutions are useful. This article deals with the case where the left hand side may be singular. In such a case, kernels are required to test a solvability condition and to derive the general form of the solutions. The complete fraction-free algorithms are therefore extended to deal with singular systems and to provide the kernels with exact computations on the same integral domain where the initial data take their entries.

This is a preprint of an article published in its final form in *Linear Algebra and its Applications* 436(1):27-40, 2012, DOI:10.1016/j.laa.2011.06.013,

It can be accessed at http://www.sciencedirect.com/science/article/pii/S0024379511004617

Keywords: exact factorization; linear system; symbolic computations; integral domain; CFFLU; Gaussian elimination

MSC codes: 65F05; 15A23; 68W30

1 Introduction

Linear systems with entries in an integral domain $\mathcal{Z}$ arise from several applications such as symbolic and exact polynomial computations [5, 14], applications to cryptography [24], computational geometry [25, 27, 18], signal processing, etc.

Though other techniques are available, see [32] and associated references, we consider herein a direct solution technique. We denote such a problem with $Ax = b$, for which the right hand side b also takes its values in $\mathcal{Z}$. If the left hand side is a square matrix, $A \in \mathcal{Z}^{n\times n}$, and is regular, the Cramer's expression of the solution allows to conclude that it lies in the field of the quotient space $\mathcal{Q}$, and that the solution of the scaled problem $Ax = (\det A)b$, where the determinant of the left hand side is $(\det A) \in \mathcal{Z}$, also lies in $\mathcal{Z}^n$.

An efficient direct solving procedure, especially with multiple right hand sides, is to provide a factorization of the matrix A, relying on the Gaussian elimination technique. If the factorization could be performed without computing fraction, all the intermediate terms will also lie in $\mathcal{Z}$ and the solution of the scaled problem is exact; the initial solution is then $(\det A)^{-1}x \in \mathcal{Q}^n$. Such a factorization is named as fraction-free [3, 4, 30, 21, 34, 8, 22] and recently, a complete fraction-free algorithm has been proposed [30, 33] for factorization, forward and backward substitutions.

Section 2 briefly recalls the original complete fraction-free algorithms, Section 3 deals with the case of singular matrices, providing a suited regularization, and the kernel computations, as well as the treatment of the surjection case with a rectangular matrix. Finally, Section 4 proposes two test examples.

2 Complete fraction-free $LD^{-1}U$ factorization of a full rank matrix

The algorithms provided in [33] are partially coded in in `fflas` library [10] (BLAS for matrices over finite fields), and in the `SymPy` library (Python library for symbolic mathematics) [29]. They factorize the regular matrix PA, where $A \in \mathcal{Z}^{n\times m}$ ($m \geq n$) and P is a n-by-n permutation matrix, into $PA = LD^{-1}U$, with a n-by-n lower triangular matrix L, a n-by-n diagonal matrix D, a n-by-m upper trapezoidal matrix U, all having their entries in $\mathcal{Z}$. The matrices L and U are of the form:

$$L = \begin{bmatrix} p_1 & & & & \\ L_{21} & p_2 & & & \\ \vdots & \vdots & \ddots & & \\ L_{n-1,1} & L_{n-1,2} & \dots & p_{n-1} & \\ L_{n,1} & L_{n,2} & \dots & L_{n,n-1} & 1 \end{bmatrix} \tag{1}$$

and

$$U = \begin{bmatrix} p_1 & U_{12} & \dots & U_{1,n-1} & U_{1,n} & \dots & U_{1,m} \\ & p_2 & \dots & U_{2,n-1} & U_{2,n} & \dots & U_{2,m} \\ & & \ddots & \vdots & \vdots & & \vdots \\ & & & p_{n-1} & U_{n-1,n} & \dots & U_{n-1,m} \\ & & & & p_n & \dots & U_{n,m} \end{bmatrix} \tag{2}$$

The p_i are the pivots, and

$$D = \operatorname{diag}(p_1, p_1p_2, \dots, p_{n-2}p_{n-1}, p_{n-1}) \tag{3}$$

The pseudo-codes for dense matrices are recalled in A.

If $n = m$, $p_n = \det A$ and p_i is the principal minor of the top left i-by-i sub-block of A (therefore the positive definiteness can be tested with $\forall i,\ p_i > 0$).

Concerning the forward substitution, since all the divisions are exact in $\mathcal{Z}$ [30], one has the property:

$$\forall b \in \mathcal{Z}^n, \quad DL^{-1}b \in \mathcal{Z}^n \tag{4}$$

therefore DL^{-1} gets all its entries in $\mathcal{Z}$.

For the backward substitution, only the first n-by-n sub-block $\hat{U}$ of U is used. The property related to the backward substitution is:

$$\forall y = DL^{-1}b,\ b \in \mathcal{Z}^n, \quad \hat{U}^{-1}(\det \hat{A})y \in \mathcal{Z}^n \tag{5}$$

where $\hat{A}$ is the first n-by-n sub-block of A, and $\det \hat{A} = p_n = \hat{U}_{n,n}$.

3 Case of a singular matrix

The case of a full rank surjective matrix A is trivial and has been considered in [10, 20]. We are concerned herein with the case of a singular square operator ($m = n$), and the case with more unknowns than equations ($m > n$) with a possible rank deficiency.

3.1 Standard LU factorization of a square singular matrix

When $\det A = 0$, a solvability condition should be satisfied for the existence of a solution for the system $Ax = b$. If the kernel of the adjoint of A (or the transpose A^T of A according to a scalar product on $\mathcal{Z}^n$) is generated by the column vectors stored in matrix S, this solvability condition is

$$S^T b = 0 \tag{6}$$

Once satisfied, this condition allows to define the set of solutions, up to a vector in the kernel of A:

$$x = A^\dagger b + Ru \tag{7}$$

R stores independent column vectors generating the kernel of A, and u is the column vector of the coordinates of a solution in the kernel of A. $A^\dagger$ is an arbitrary generalized inverse of A [23, 12, 15] (any generalized inverse generates the same family of solutions).

If $S^T b \neq 0$, there is no solution.

For sake of simplicity, we assume that a standard LU factorization of A [9], with a unitary lower triangular matrix L, can be performed without pivoting. Null diagonal entries of U correspond to null pivots; they are as numerous as the size of the kernel, which will be denoted with r [19]. The LU factorization can easily be modified to return the factorization of a regularized matrix $\bar{A}$ and to allow the extraction of the kernel vector sets R and S. The corresponding generalized inverse is then $A^\dagger = \bar{A}^{-1}$. This is described in the following developments.

Consider first an arbitrary splitting of the unknowns in two sets, denoted with subscripts 1 and 2. The matrix A can be split accordingly in several blocks:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{8}$$

where A_{11} and A_{22} are diagonal square blocks. If A_{11} is regular (which is assumed here), this system can be condensed onto the second set of unknowns:

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22}^\star \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2^\star \end{bmatrix} \tag{9}$$

with the Schur complement $A_{22}^\star = A_{22} - A_{21}A_{11}^{-1}A_{12}$ and the condensed right hand side $b_2^\star = b_2 - A_{21}A_{11}^{-1}b_1$. The initial system reads:

$$\underbrace{\begin{bmatrix} I_1 & 0 \\ A_{21}A_{11}^{-1} & I_2 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22}^\star \end{bmatrix}}_{A} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \underbrace{\begin{bmatrix} I_1 & 0 \\ A_{21}A_{11}^{-1} & I_2 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2^\star \end{bmatrix}}_{b} \tag{10}$$

where I_k is the identity matrix (with a size consistent to its use).

The previous condensation operation can also be performed recursively, line per line of the left upper block A_{11} of the system (8) to give:

$$\begin{bmatrix} U_{11} & U_{12} \\ 0 & A_{22}^\star \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} c_1 \\ b_2^\star \end{bmatrix} \tag{11}$$

and the initial system reads:

$$\underbrace{\underbrace{\begin{bmatrix} L_{11} & 0 \\ L_{21} & I_2 \end{bmatrix}}_{L^{(1)}} \underbrace{\begin{bmatrix} U_{11} & U_{12} \\ 0 & A_{22}^\star \end{bmatrix}}_{U^{(1)}}}_{A} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \underbrace{\begin{bmatrix} L_{11} & 0 \\ L_{21} & I_2 \end{bmatrix} \begin{bmatrix} c_1 \\ b_2^\star \end{bmatrix}}_{b} \tag{12}$$

for which the *same* Schur complement has been computed recursively (this can be easily proved using the sub-block factorization $A_{11} = L_{11}U_{11}$, $A_{12} = L_{11}U_{12}$, $A_{21} = L_{21}U_{11}$ and the uniqueness of the factorization without pivoting).

Assume now that the second diagonal block, of size $r \times r$, corresponds to all the null pivots (this would be obtained with a total pivoting strategy): $A_{22}^\star = 0$. In this case, the factorization is completed and $L = L^{(1)}$, $U = U^{(1)}$. Existence of a solution is ensured if and only if $b_2^\star = 0$, which is the aforementioned solvability condition. This condition reads: $b_2^\star = B^T L^{-1} b = 0$, where $B \in \mathcal{Z}^{n \times r}$ is a Boolean matrix containing non null terms only on its second block, which is an identity block of size $r \times r$: $B^T = \begin{bmatrix} 0 & I_2 \end{bmatrix}$. By identification with (6), the kernel of A^T is therefore

$$S = L^{-T} B \tag{13}$$

One way to resume the solution of the system is to 'clamp' the undetermined unknowns (the last ones) to an arbitrary fixed value, for instance identity, and to backsolve the remaining regular left upper block. Null pivots in the partial factorization are therefore replaced with identity, and dedicated partially condensed right hand sides (as many as the kernel size) are settled:

$$\underbrace{\begin{bmatrix} U_{11} & U_{12} \\ 0 & I_2 \end{bmatrix}}_{\bar{U}} R = B = \begin{bmatrix} 0 \\ I_2 \end{bmatrix} \tag{14}$$

$\bar{U}$ is then invertible and the solution of this system is a generating set of independent vectors of the kernel of A:

$$R = \bar{U}^{-1}B \tag{15}$$

This is the factorization of a regularized version of A:

$$\bar{A} = L\bar{U} = L\left(U + \begin{bmatrix} 0 & 0 \\ 0 & I_2 \end{bmatrix}\right) = LU + \begin{bmatrix} L_{11} & 0 \\ L_{12} & I_2 \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & I_2 \end{bmatrix} = A + \begin{bmatrix} 0 & 0 \\ 0 & I_2 \end{bmatrix} \tag{16}$$

A and $\bar{A}$ only differ with the kernel sub-space.

3.2 Completely fraction-free factorization of a singular square matrix

The goal of this section is to extend the previous factorization, regularization and kernel extraction to fraction-free computations. The proposed factorization requires a total pivoting strategy [26, 2] (though only the first non null pivot is searched for, not the pivot with maximal amplitude). In this case, the null pivots are postponed to the last degree-of-freedom positions, and a null block is built at the stage where no pivoting is possible anymore. Assuming at a first step that pivoting was not necessary for sake of simplicity, the system to solve is identical to (11) with a null block $A_{22}^{(1)} = 0$, and a regular block U_{11}:

$$\begin{bmatrix} U_{11} & U_{12} \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} c_1 \\ b_2^{(1)} \end{bmatrix} \tag{17}$$

and the initial system reads:

$$\underbrace{\underbrace{\begin{bmatrix} L_{11} & 0 \\ L_{21} & I_2 \end{bmatrix}}_{L}\underbrace{\begin{bmatrix} D_{11} & 0 \\ 0 & I_2 \end{bmatrix}^{-1}}_{D^{-1}}\underbrace{\begin{bmatrix} U_{11} & U_{12} \\ 0 & 0 \end{bmatrix}}_{U}}_{A}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \underbrace{LD^{-1}\begin{bmatrix} c_1 \\ b_2^{(1)} \end{bmatrix}}_{b} \tag{18}$$

The blocks U_{11}, U_{12}, L_{11}, L_{21} and D_{11} are given from expressions in (1), (3) and (2). The solvability condition is still $b_2^{(1)} = 0$.

To provide a fraction-free factorization and kernel extraction, two approaches can be derived: (i) using a fraction-free regularization $\bar{A}$ of A that allows to fulfill the standard fraction-free factorization, (ii) deal directly with the equivalent rectangular system (once the solvability condition is satisfied):

$$\begin{bmatrix} U_{11} & U_{12} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = c_1 \tag{19}$$

Choosing one solution or the other depends on implementation issues. The first solution is described in the following, while the second one is discussed in Section 3.3.

As previously, to resume the factorization of a regularized matrix $\bar{A} \in \mathcal{Z}^{n\times n}$, the replacement of the null block $A_{22}^{(1)}$ should be performed in a suited way: Referring to the recursion formula on line 20 in algorithm 1, the null pivots should be replaced with the last non null pivot found in the previous factorization of the first block, i.e. p_{n-r} (therefore, for $k \geq n-r+1$, $A_{i,k} = 0$, and the new block $A_{22}^{(1)}$ is not modified by the factorization). Finally, one gets the modified blocks:

$$\bar{L}_{22} = \begin{bmatrix} p_{n-r}I_{r-1} & 0 \\ 0 & I_r \end{bmatrix}, \quad \bar{D}_{22} = \begin{bmatrix} p_{n-r}^2 I_{r-1} & 0 \\ 0 & p_{n-r} \end{bmatrix} \quad \text{and} \quad \bar{U}_{22} = p_{n-r}I_r \tag{20}$$

the modified factorization:

$$\bar{L} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & \bar{L}_{22} \end{bmatrix}, \quad \bar{D} = \begin{bmatrix} D_{11} & 0 \\ 0 & \bar{D}_{22} \end{bmatrix} \quad \text{and} \quad \bar{U} = \begin{bmatrix} U_{11} & U_{12} \\ 0 & \bar{U}_{22} \end{bmatrix} \tag{21}$$

and one recovers (16):

$$\det \bar{A} = p_{n-r}, \quad \bar{A} = \bar{L}\bar{D}^{-1}\bar{U} = A + \begin{bmatrix} 0 & 0 \\ 0 & \bar{L}_{22}\bar{D}_{22}^{-1}\bar{U}_{22} \end{bmatrix} = A + \begin{bmatrix} 0 & 0 \\ 0 & I_2 \end{bmatrix} \tag{22}$$

with the regularized problem:

$$\bar{U}x = \bar{D}\bar{L}^{-1}b \tag{23}$$

The fraction-free forward and backward substitution algorithms are preserved as previously; their usage on the factorization of a regularized matrix ensures that the solutions are still taking their entries in $\mathcal{Z}$.

For kernel construction, one needs some matrices B and C in $\mathcal{Z}^{n\times r}$ with a regular second block of size $r \times r$:

$$B = \begin{bmatrix} 0 \\ B_{22} \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} 0 \\ C_{22} \end{bmatrix} \tag{24}$$

Then, $S = \bar{L}^{-T}\bar{D}B$ and $R = \bar{U}^{-1}C$.

Suitable choices for B_{22} and C_{22} can be made to preserve fraction-free computations. In a first step, one can choose $B_{22} = I_2$, Indeed, since $\bar{D}\bar{L}^{-1} \in \mathcal{Z}^{n\times n}$, see (4), its transpose has also its entries in $\mathcal{Z}$ which allows to conclude that $S \in \mathcal{Z}^{n\times r}$. In a second step, we propose to select $C_{22} = (\det \bar{A})\bar{D}_{22}$. Indeed, $\bar{L}\bar{D}^{-1}\begin{bmatrix}0 & \bar{D}_{22}\end{bmatrix}^T \in \mathcal{Z}^{n\times r}$ and since $\bar{L}\bar{D}^{-1}\bar{U}R = (\det \bar{A})\bar{L}\bar{D}^{-1}\begin{bmatrix}0 & \bar{D}_{22}\end{bmatrix}^T$, the solving fraction-free algorithm will provide $R \in \mathcal{Z}^{n\times r}$ by design (5).

If pivoting was necessary, the previous proofs are still valid since the pivoting does not alter the intrinsic properties. The factorization of A reads $A = P\bar{L}\bar{D}^{-1}UQ$ and the factorization of the regularized matrix is $\bar{A} = P\bar{L}\bar{D}^{-1}\bar{U}Q$. P and Q are permutation matrices. They are orthogonal matrices and the reverse permutation operation is P^T and Q^T. This is used for notation purpose only since the permutation matrices are not explicitly stored in the pseudo-codes of this article in A and B, conforming to the BLAS implementation standards [6]. The associated kernels are:

$$S = P\bar{L}^{-T}\bar{D}B \qquad \text{and} \qquad R = Q^T\bar{U}^{-1}(\det \bar{A})C \tag{25}$$

In practice, these expressions are used as follows. For the kernel S, once the multiple r right hand sides are stored in B, a call to the forward substitution routine that allows to solve for a transpose left hand side (algorithm 5) is performed. The result is then permuted with P to produce the kernel S. Concerning the kernel R, the multiple r right hand sides stored in C are used. This time, the call to the backward substitution routine (algorithme 3) is performed, and a reverse permutation with Q on the result allows to obtain R.

3.3 Case of a rectangular matrix

The extension of the previous algorithm to matrix $A \in \mathcal{Z}^{n\times m}$ with $m > n$ is easy to derive for the standard case [16].

For the fraction-free case, we will first consider the case where $\operatorname{rank} A = n$. In this case, the factorization up to the stage (19) leads to a regular block U_{11}. For an original rectangular system, it reads:

$$A = \begin{bmatrix}A_{11} & A_{12}\end{bmatrix} = LD^{-1}\begin{bmatrix}U_{11} & U_{12}\end{bmatrix} \quad \text{and} \quad b = LD^{-1}c_1 \tag{26}$$

For any case, the problem is:

$$U_{11}x_1 = c_1 - U_{12}x_2 \tag{27}$$

With a regular block U_{11}, the kernel R is such that $U_{11}R$ is proportional to $U_{12} = DL^{-1}A_{12}$. With the same argument as before (5), to get $R \in \mathcal{Z}^{n\times(m-n)}$, it is proposed to choose $R = U_{11}^{-1}(\det A_{11})U_{12}$. Indeed, in such a case, $LD^{-1}U_{11}R = (\det A_{11})A_{12}$. With $\operatorname{rank} A = n$, no solvability condition apply and S does not exist.

With a rank-deficient matrix A ($\operatorname{rank} A = n - r$) and a total pivoting strategy, using the same regularization as before, the stage (19) reads:

$$\begin{bmatrix}\bar{U}_{11} & U_{12}\end{bmatrix}\begin{bmatrix}x_1' \\ x_2'\end{bmatrix} = \bar{D}\bar{L}^{-1}P^Tb \quad \text{with} \quad \begin{bmatrix}x_1' \\ x_2'\end{bmatrix} = Q\begin{bmatrix}x_1 \\ x_2\end{bmatrix} \tag{28}$$

or

$$\bar{U}_{11}x_1' = \bar{D}\bar{L}^{-1}P^Tb - U_{12}x_2' \tag{29}$$

This problem has the same structure as (23), therefore, the kernel of A^T has the same expression:

$$S = P\bar{L}^{-T}\bar{D}B \quad \in \mathcal{Z}^{n\times r} \tag{30}$$

and the kernel of A is:

$$R = Q^T\begin{bmatrix}\bar{U}_{11}^{-1} & 0 \\ 0 & I\end{bmatrix} d \begin{bmatrix}C & U_{12} \\ 0 & I\end{bmatrix} \quad \in \mathcal{Z}^{m\times(r+m-n)} \tag{31}$$

where d is the $(n-r, n-r)$ entry of $\bar{U}_{11}$. Once the matrix A has been factorized, and the permutations built, this kernel is obtained by: (i) calling the backward substitution routine with the multiple $(r+m-n)$ right hand sides $\begin{bmatrix} C & U_{12} \end{bmatrix}$, (ii) the completion of the results with the $(m-n)$ lines $d \begin{bmatrix} 0 & I \end{bmatrix}$, and (iii) the reverse permutation of the result.

3.4 Implementation issues

Since the regularization procedure is quite easy to code, and does not require memory management for intermediate storage, this is the preferred solution we chose for dealing with the singular case. The treatment of the rectangular case also embeds this regularization procedure when $\operatorname{rank} A < n$. The pseudo-codes in B describe this implementation.

The main drawback of the fraction-free algorithms is the large growing rate of the terms during the factorization and the substitutions (though this is reduced when compared to a division-free algorithm since exact divisions are taken into account here). For instance, for integer computations, $\mathcal{Z} = \mathbb{Z}$, the growth in the dynamic range of the integers that must be represented is recalled in [33]: if the length of $a \in \mathbb{Z}$, $a \neq 0$, is defined as $\lambda(a) = \lfloor \log |a| + 1 \rfloor$ where $\lfloor \cdot \rfloor$ is the rounding down to the nearest integer, and if the lengths of the elements of A are bounded by ℓ, then the elements of L and U have their lengths bounded by $n(\ell + \log n)$.

A classical storage strategy for the integers is to use a double precision declaration and to certify that no cancellation error will occur. With IEEE standard for binary floating-point arithmetic (ANSI-IEEE 754-1985) on 32-bit architectures, to avoid cancellation error, integers stored in REAL*8 should be less than $2^{53} - 1$, i.e. their length should be less than 16. For integer storage the value is limited to $2^{31} - 1$, i.e. their length should be less than 10.

For the case of large-size systems, alternative solutions consist in using arbitrary adaptive precision algebra libraries [17, 13], modular methods such as a residue number system (RNS) [31], or high-order lifting techniques [28].

Though additional gain in dynamic range can be reached by using greatest common divisor searches per completed line of the matrix, this may not improve the worst case.

4 Test cases

For illustration and eventual checking purposes, we propose two examples for $\mathcal{Z} = \mathbb{Z}$. The first one arise from computational geometry: To match incompatible 3D finite element meshes, the core problem for mesh intersection of linear elements is to intersect a line (defined by a segment A_1, A_2) with a plane (defined by a triangle M_1, M_2, M_3). Though using barycentric coordinates may not be the most efficient approach [25], these coordinates are required for the associated information transfer problem from one mesh to another [11]. Therefore, the problem is to find the point $M = \sum_{i=1}^{3} \lambda_i M_i = \sum_{j=1}^{2} \mu_j A_j$ with the partition of unity $\sum_{i=1}^{3} \lambda_i = \sum_{j=1}^{2} \mu_j = 1$. The problem therefore reduces to find $x = \begin{bmatrix} \lambda_2 & \lambda_3 & \mu_2 \end{bmatrix}^T$ such that $Ax = b$ with $A = \begin{bmatrix} \underline{M_1 M_2} & \underline{M_1 M_3} & -\underline{A_1 A_2} \end{bmatrix}$ and $b = [\underline{M_1 A_1}]$. For robustness purposes, a snap rounding of nodal coordinates is used; this leads to integer coordinates. Checking singularities is mandatory due to singular positions that occur for instance on planar boundaries of the triangulated domain where segments and triangles are parallel. This is for instance the case when (see Figure 1)

$$M_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \; M_2 = \begin{bmatrix} 16 \\ 8 \\ 0 \end{bmatrix}, \; M_3 = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix}, \; A_1 = \begin{bmatrix} 12 \\ 6 \\ -1 \end{bmatrix}, \; A_2 = \begin{bmatrix} -4 \\ -2 \\ 3 \end{bmatrix}$$

which leads to:

$$A = \begin{bmatrix} 16 & 0 & 16 \\ 8 & 0 & 8 \\ 0 & 4 & -4 \end{bmatrix} \qquad \text{and} \qquad b = \begin{bmatrix} 12 \\ 6 \\ -1 \end{bmatrix}$$

The in-place factorization of A gives the matrix (the two last rows have been permuted: P is stored as [1 3 3] and Q is stored as [1 2 3] in the pseudo-code provided[1], conforming to the BLAS standard [6])

$$\begin{bmatrix} 16 & 0 & 16 \\ 0 & 64 & -64 \\ 8 & 0 & 64 \end{bmatrix}$$

[1] some possible duplicates in P or Q arise from the choice of using an incremental permutation strategy for efficiency purposes, see associated pseudo-codes.

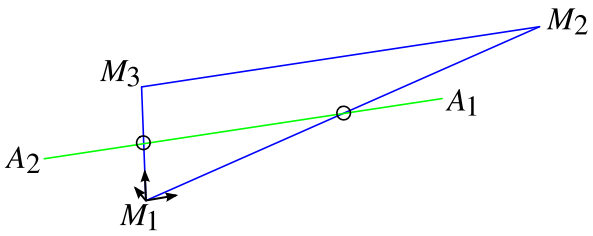

Figure 1: Test case of a 3D segment-triangle intersection

for which the fraction-free factorization of the regularized matrix $\bar{A}$ can be extracted:

$$L = \begin{bmatrix} 16 & 0 & 0 \\ 0 & 64 & 0 \\ 8 & 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 16 & 0 & 0 \\ 0 & 1024 & 0 \\ 8 & 0 & 64 \end{bmatrix}, \quad \bar{U} = \begin{bmatrix} 16 & 0 & 16 \\ 0 & 64 & -64 \\ 0 & 0 & 64 \end{bmatrix}$$

The determinant of $\bar{A}$ is $d = \bar{U}_{3,3} = 64$, while the factorization of the singular matrix A would have given:

$$U = \begin{bmatrix} 16 & 0 & 16 \\ 0 & 64 & -64 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad LD^{-1}U = \begin{bmatrix} 16 & 0 & 16 \\ 0 & 4 & -4 \\ 8 & 0 & 8 \end{bmatrix}$$

$LD^{-1}U$ is indeed the permutation of A, and the permutation of $\bar{A}$ (with $ln_p = [3]$) is

$$LD^{-1}\bar{U} = \begin{bmatrix} 16 & 0 & 16 \\ 0 & 4 & -4 \\ 8 & 0 & 9 \end{bmatrix}$$

Backward permutations of the kernels give their expressions in the original basis: $R = \begin{bmatrix} -64 & 64 & 64 \end{bmatrix}^T$ and $S = \begin{bmatrix} -32 & 64 & 0 \end{bmatrix}^T$. One can check that $A\,R = A^T\,S = 0$ and that the solvability condition is satisfied: $S^T b = 0$. With the particular choice of pseudo-inverse of A being $A^\dagger = \bar{A}^{-1}$, the classical fraction-free forward substitution gives (in the permuted basis): $y = (LD^{-1})^{-1}P^{-1}b = \begin{bmatrix} 12 & -16 & 0 \end{bmatrix}^T$ and the backward substitution gives (in the original basis): $x = Q^{-1}\bar{U}^{-1}y = \begin{bmatrix} 48 & -16 & 0 \end{bmatrix}^T$.

One can finally check that any combination of the form (7) is a solution, for instance: $Ax - d\,b = 0$. In this case, the solution of the intersection problem is $\lambda_1 = 1/2$, $\lambda_2 = (48 - 64u)/d$, $\lambda_3 = (-16 + 64u)/d$, $\mu_1 = (d - 64u)/d$, $\mu_2 = 64u/d$, with an arbitrary u; this is indeed the whole line (A_1, A_2). Extremal solutions (for which at least one of the barycentric coordinate is in $\{0, 1\}$ and all are in $[0, 1]$) can therefore easily be checked. They are the intersections with the triangle edges: ($\lambda_1 = \lambda_3 = 1/2$, $\lambda_2 = 0$, $\mu_1 = 1/4$, $\mu_2 = 3/4$) and ($\lambda_1 = \lambda_2 = 1/2$, $\lambda_3 = 0$, $\mu_1 = 3/4$, $\mu_2 = 1/4$).

The next example is extracted from [7], with a singular matrix obtained by replacing two columns by linear combination of others. It is designed to illustrate the growth in the dynamical range of the entries in $\mathcal{Z}$. Consider the following problem:

$$A = \begin{bmatrix} 68 & 25 & 11 & 26 & 55 \\ 66 & -36 & -32 & -51 & 17 \\ 134 & -11 & -21 & -25 & 72 \\ -5 & 85 & 58 & -22 & -25 \\ -73 & 60 & 47 & -48 & -80 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 5 \\ 10 \\ 15 \\ 30 \\ 25 \end{bmatrix}$$

The in-place factorization of $\bar{A}$ gives the matrix (with the permutation of rows corresponding to [1 2 4 4 5] and no column permutation, but with final pivots $ln_p = [4\ 5]$):

$$\begin{bmatrix} 68 & 25 & 11 & 26 & 55 \\ 66 & -4098 & -2902 & -5184 & -2474 \\ -5 & 5905 & 11006 & 532491 & 300715 \\ 134 & -4098 & 0 & 11006 & 0 \\ -73 & 5905 & 11006 & 0 & 11006 \end{bmatrix}$$

where

$$L = \begin{bmatrix} 68 & 0 & 0 & 0 & 0 \\ 66 & -4098 & 0 & 0 & 0 \\ -5 & 5905 & 11006 & 0 & 0 \\ 134 & -4098 & 0 & 11006 & 0 \\ -73 & 5905 & 11006 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 68 & 0 & 0 & 0 & 0 \\ 0 & -278664 & 0 & 0 & 0 \\ 0 & 0 & -45102588 & 0 & 0 \\ 0 & 0 & 0 & 121132036 & 0 \\ 0 & 0 & 0 & 0 & 11006 \end{bmatrix}$$

$$\bar{U} = \begin{bmatrix} 68 & 25 & 11 & 26 & 55 \\ 0 & -4098 & -2902 & -5184 & -2474 \\ 0 & 0 & 11006 & 532491 & 300715 \\ 0 & 0 & 0 & 11006 & 0 \\ 0 & 0 & 0 & 0 & 11006 \end{bmatrix}$$

The permuted kernels are:

$$R = \begin{bmatrix} -51585 & -36105 \\ 363161 & 206307 \\ -532491 & -300715 \\ 11006 & 0 \\ 0 & 11006 \end{bmatrix} \quad \text{and} \quad S = \begin{bmatrix} -11006 & 11006 \\ -11006 & 0 \\ 0 & -11006 \\ 11006 & 0 \\ 0 & 11006 \end{bmatrix}$$

and one solution is obtained as: $x = \begin{bmatrix} -14110 & 108710 & -154840 & 0 & 0 \end{bmatrix}^T$ with $d = 11006$.

5 Conclusions

This article proposes an extension of complete fraction-free algorithms, first to square singular matrices in an integral domain, $A \in \mathcal{Z}^{n \times n}$ with $\det A = 0$, providing: (i) the factorization of a regularized matrix $\bar{A} \in \mathcal{Z}^{n \times n}$ whose inverse is a pseudo-inverse of A, and (ii) the fraction-free determination of the kernels of A and of A^T. Second, the case with rectangular matrices $A \in \mathcal{Z}^{n \times m}$ with $m > n$, and exhibiting a rank deficiency, is derived as an other extension. As for the original algorithm, all computations are exact in any integral domain and singular linear systems can be solved within their input domain.

A Pseudo-code for complete fraction-free factorization

The pseudo-codes for dense matrices are recalled in algorithms 1, 2 and 3. Supplementary information from reference [33] are: the matrices L and U are overwritten on A during the factorization (the diagonal of L is not stored), the vector y is overwritten on b during the forward substitution (where b has been permuted if needed), and the vector x is overwritten on y during the backward substitution; the permutations are stored in a list P giving indexes for row permutations (consistent with the standard LAPACK [1] factorization routine `dgetf2`, and BLAS [6] substitution routine `dtrsm`); and a typo correction has been fixed with respect to the original article (line 8 in algorithm 2). All the involved divisions are proved to be exact in $\mathcal{Z}$, and the tests are strict [30, 33].

B Pseudo-code for singular complete fraction-free factorization with kernel extraction

The additional codes for singular cases are given in algorithms 4 and 6. Note that the last one, for kernel extraction, deals only with a square matrix, which can be the first n-by-n block of a permuted factorization of a n-by-m ($m > n$) rectangular matrix. The algorithm 5 is a modified forward substitution with a transpose left hand side. All the divisions are exact in $\mathcal{Z}$ due to the property $(DL^{-1})^T \in \mathcal{Z}^n$.

Algorithm 1 Completely fraction-free factorization of dense A

Inputs: A
$(n, m) \leftarrow \text{size}(A)$
oldpivot $\leftarrow 1$
for $k = 1, 2, \ldots, n$ **do**
  $P(k) \leftarrow k$
  if $A(k, k) = 0$ **then**
    Search for the first non null pivot $A(\text{kpivot}, k)$, for kpivot $\in [k + 1, n]$
    if No pivot found **then**
      Error: Matrix is rank deficient
    else
      {Row interchange}
      $A(k, 1 : m) \leftrightarrow A(\text{kpivot}, 1 : m)$, $P(k) \leftarrow$ kpivot
    end if
  end if
  pivot $\leftarrow A(k, k)$
  for $i = k + 1, \ldots, n$ **do**
    $Aik \leftarrow A(i, k)$
    for $j = k + 1, \ldots, m$ **do**
      {Integer exact division}
      $A(i, j) \leftarrow (\text{pivot} * A(i, j) - Aik * A(k, j))/\text{oldpivot}$
    end for
  end for
  oldpivot $\leftarrow$ pivot
end for
Return: modified A (storing L, D and U), P

Algorithm 2 Forward substitution: solve y from $LD^{-1}y = b$ for a permuted b

Inputs: A (storing L and D), b
$n \leftarrow \text{size}(b)$
oldpivot $\leftarrow 1$
for $k = 1, 2, \ldots, n - 1$ **do**
  pivot $\leftarrow A(k, k)$
  for $i = k + 1, \ldots, n$ **do**
    {Integer exact division}
    $b(i) \leftarrow (\text{pivot} * b(i) - A(i, k) * b(k))/\text{oldpivot}$
  end for
  oldpivot $\leftarrow$ pivot
end for
Return: modified b (storing y)

Algorithm 3 Backward substitution: solve x from $Ux = d\,y$ with $d = \det A$

Inputs: A (storing U), y
$n \leftarrow \text{size}(y)$
$d \leftarrow A(n, n)$
for $i = n, n - 1, \ldots, 1$ **do**
  {Integer exact division}
  $y(i) \leftarrow (d * y(i) - \sum_{k=i+1}^{n} A(i, k) * y(k))/A(i, i)$
end for
Return: modified y (storing x), scaling factor d

Algorithm 4 Completely fraction-free factorization of a dense regularization of A with full pivoting

Inputs: A
$(n, m) \leftarrow \text{size}(A)$
$n_p \leftarrow 0$
oldpivot $\leftarrow 1$
for $k = 1, 2, \ldots, n$ **do**
  $P(k) \leftarrow k$, $Q(k) \leftarrow k$
  if $A(k, k) = 0$ **then**
    Search for the first non null pivot $A(\text{kpivot}, \text{lpivot})$, for kpivot $\in [k+1, n]$ and lpivot $\in [k+1, n]$
    if No pivot found **then**
      {Matrix is rank deficient}
      $n_p \leftarrow n_p + 1$, $ln_p(n_p) \leftarrow k$
      {Apply a suited regularization}
      $A(k, k) \leftarrow$ oldpivot
    else
      {Row interchange}
      $A(k, 1 : m) \leftrightarrow A(\text{kpivot}, 1 : m)$, $P(k) \leftarrow$ kpivot
      {Column interchange}
      $A(1 : n, k) \leftrightarrow A(1 : n, \text{lpivot})$, $Q(k) \leftarrow$ kpivot
    end if
  end if
  {No more singularity here: Apply CFF transformation}
  pivot $\leftarrow A(k, k)$
  for $i = k + 1, \ldots, n$ **do**
    $Aik \leftarrow A(i, k)$
    for $j = k + 1, \ldots, m$ **do**
      {Integer exact division}
      $A(i, j) \leftarrow (\text{pivot} * A(i, j) - Aik * A(k, j))/\text{oldpivot}$
    end for
  end for
  oldpivot $\leftarrow$ pivot
end for
Return: modified A (storing L, D and U of the regularized matrix $\bar{A}$), P, Q and ln_p

Algorithm 5 Modified forward substitution: solve y from $L^T D^{-1} y = b$

Inputs: A (storing L and D), b
$n \leftarrow \text{size}(b)$
for $i = n$, $n - 1$,..., 1 **do**
  if $i = 1$ **then**
    oldpivot $\leftarrow 1$
  else
    oldpivot $\leftarrow A(i - 1, i - 1)$
  end if
  {Integer exact division}
  $b(i) \leftarrow \text{oldpivot} * b(i) - (\sum_{k=i+1}^{n} A(k, i) * b(k))/A(i, i)$
end for
Return: modified b (storing y)

Algorithm 6 Find kernels from a fraction-free factorization of a regularized square matrix A

Inputs: A (storing L, D, U), ln_p (for original nul pivot locations)
$n \leftarrow \text{size}(A, 1)$
$n_p \leftarrow \text{length}(ln_p)$
$d \leftarrow A(n, n)$
{Initialize right-hand-sides for kernel extractions}
for $i = 1, 2, \ldots, n_p$ **do**
    $R(ln_p(i), i) \leftarrow d$
    $S(ln_p(i), i) \leftarrow 1$
end for
Solve $U\,R = d\,b$ where the n_p right-hand-sides b are stored in R on input, with fraction-free backward substitution (algorithm 3)
Solve $L^T\,D^{-1}\,S = b$ where the n_p right-hand-sides b are stored in S on input, with fraction-free forward modified substitution (algorithm 5)
Return: kernels R and S

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[2] D.-O. Azulay and I. Gambini. Which pivot to solve linear systems? *Journal of Computational and Applied Mathematics*, 147(2):471–484, 2002.

[3] E. H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of Computation*, 22:565–578, 1968.

[4] E. H. Bareiss. Computational solutions of matrix problems over an integral domain. *Journal of the Institute of Mathematics and Its Applications*, 10:68–104, 1972.

[5] D. Bini and V. Pan. *Polynomial and matrix computations*. Birkhäuser, 1994.

[6] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley. An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28(2):135–151, 2002.

[7] S. F. Chang and W. J. Kennedy. Error-free computer solution of certain systems of linear equations. *Journal of Computational and Applied Mathematics*, 18(3):279–287, 1987.

[8] R. M. Corless and D. J. Jeffrey. The Turing factorization of a rectangular matrix. *SIGSAM Bulletin*, 31(3):20–30, 1997.

[9] F. M. Dopico, C. R. Johnson, and J. M. Molera. Multiple LU factorizations of a singular matrix. *Linear Algebra and its Applications*, 419:24–36, 2006.

[10] J.-G. Dumas, P. Giorgi, and C. Pernet. Dense linear algebra over word-size prime fields: the FFLAS and FFPACK packages. *ACM Transactions on Mathematical Software*, 35(3):19:1–19:42, 2008.

[11] D. Dureisseix and H. Bavestrello. Information transfer between incompatible finite element meshes: Application to coupled thermo-viscoelasticity. *Computer Methods in Applied Mechanics and Engineering*, 195(44-47):6523–6541, 2006.

[12] C. Farhat and D. Rixen. *Encyclopedia of Vibration*, chapter Linear Algebra, pages 710–720. Academic Press, 2002.

[13] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software*, 33(2):13, 2007.

[14] P. Giorgi, C.-P. Jeannerod, and G. Villard. On the complexity of polynomial matrix computations. In R. Sendra, editor, *International Symposium on Symbolic and Algebraic Computation*, pages 135–142, Philadelphia, Pennsylvania, USA, 2003. ACM Press.

[15] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

[16] C. Gotsman and S. Toledo. On the computation of null spaces of sparse rectangular matrices. *SIAM Journal on Matrix Analysis and Applications*, 30(2):445–463, 2008.

[17] T. Granlund. GMP, the GNU multiple precision arithmetic library. Technical report, TMG Datakonsult, 2001.

[18] D. Halperin and E. Packer. Iterated snap rounding. *Computational Geometry: Theory and Applications*, 23(2):209–225, 2002.

[19] T.-M. Hwang, W.-W. Lin, and E. K. Yang. Rank revealing LU factorizations. *Linear Algebra and its Applications*, 175:115–141, 1992.

[20] O. H. Ibarra, S. Moran, and R. Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *Journal of Algorithms*, 3(1):45–56, 1982.

[21] H. R. Lee and B. D. Saunders. Fraction-free gaussian elimination for sparse matrices. *Journal of Symbolic Computation*, 19(5):393–402, 1995.

[22] T. Mulders. A generalized Sylvester identity and fraction-free random Gaussian elimination. *Journal of Symbolic Computation*, 31:447–460, 2001.

[23] B. Noble. A method for computing the generalized inverse of a matrix. *SIAM Journal on Numerical Analysis*, 3(4):582–584, 1966.

[24] A. M. Odlyzko. Discrete logarithms: The past and the future. *Designs, Codes, and Cryptography*, 19:129–145, 2000.

[25] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, second edition, 1998.

[26] J. M. Peña. LDU decompositions with L and U well conditioned. *Electronic Transactions on Numerical Analysis*, 18:198–208, 2004.

[27] S. Schirra. *Robustness and precision issues in geometric computation*, chapter 14, pages 597–632. Handbook of Computational Geometry. Elsevier Science Publishers B.V., North-Holland, Amsterdam, 2000.

[28] A. Storjohann. The shifted number system for fast linear algebra on integer matrices. *Journal of Complexity*, 21(4):609–650, 2005.

[29] SymPy Development Team. *SymPy: Python library for symbolic mathematics*, 2011.

[30] P. R. Turner. A simplified fraction-free integer Gauss elimination algorithm. Report NAWCADPAW–96-196-TR, Naval Air Warfare Center, Warminster, PA, 1995.

[31] P. R. Turner. Fraction-free RNS algorithms for solving linear systems. In *IEEE Symposium on Computer Arithmetic*, pages 218–224, Los Alamitos, CA, USA, 1997. IEEE Computer Society.

[32] Z. Wan. An algorithm to solve integer linear systems exactly using numerical methods. *Journal of Symbolic Computation*, 41(6):621–632, 2006.

[33] W. Zhou and D. J. Jeffrey. Fraction-free matrix factors: new forms for LU and QR factors. *Frontiers of Computer Science in China*, 2(1):1–13, 2008.

[34] W. Zhou, D. J. Jeffrey, and R. M. Corless. Fraction-free forms of LU matrix factoring. In *Transgressive Computing*, pages 443–446, 2006.