LU Factoring of Non-Invertible Matrices

D. J. Jeffrey
Department of Applied Mathematics,
The University of Western Ontario,
London, Ontario, Canada N6A 5B7

Revised submission to Communications in Computer Algebra, 2010

Abstract

The definition of the LU factoring of a matrix usually requires that the matrix be invertible. Current software systems have extended the definition to non-square and rank-deficient matrices, but each has chosen a different extension. Two new extensions, both of which could serve as useful standards, are proposed here: the first combines LU factoring with full-rank factoring, and the second extension combines full-rank factoring with fraction-free methods. Amongst other applications, the extension to full-rank, fraction-free factoring is the basis for a fraction-free computation of the Moore—Penrose inverse.

1 Introduction

Mathematical software systems occasionally take the initiative away from mainstream mathematics and create extensions of mathematical theory. The example of interest here concerns the LU factoring of matrices. Many textbooks restrict their definitions to square invertible matrices, and early versions of Maple, Mathematica and Matlab followed the textbooks by implementing LU-factoring routines that gave error messages for non-square matrices, and also gave error messages if the square matrix were singular.

More recent versions of these software systems have been leading the way in extending the definition of LU factoring, however, they have been leading 'madly off in all directions' [18]. Recent versions of Matlab and Maple will now return results for all matrices, but not the same results. For example, two sets of LU factors for the same matrix are given below; the first line shows the factors returned by Matlab 7.9 and the second shows those returned by Maple 13.

$$\begin{pmatrix} 5 & 10 & 15 & 20 \\ -1 & -6 & -19 & -16 \\ 1 & 5 & 15 & 19 \\ 5 & 6 & -1 & -12 \\ 4 & 9 & 16 & 29 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -0.2 & 1 & 0 & 0 \\ 0.2 & -0.75 & 1 & 0 \\ 1.0 & 1.0 & 0 & 1 \\ 0.8 & -0.25 & 0 & -0.5 \end{pmatrix} \begin{pmatrix} 5 & 10 & 15 & 20 \\ 0 & -4 & -16 & -12 \\ 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & -20 \end{pmatrix}, \tag{1}$$

$$= \begin{pmatrix} 1 & 0 & 0 & \mathbf{0} & \mathbf{0} \\ -1/5 & 1/5 & 0 & \mathbf{0} & \mathbf{0} \\ 1/5 & -3/20 & -1/20 & \mathbf{0} & \mathbf{0} \\ 1 & 1/5 & 1/6 & \mathbf{1} & \mathbf{0} \\ 4/5 & -1/20 & -1/12 & \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} 5 & 10 & 15 & 20 \\ 0 & -20 & -80 & -60 \\ 0 & 0 & 0 & -120 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{2}$$

Note that the matrix was chosen so that the programs would not interchange the rows, and hence the permutation matrix is not shown, being an identity matrix.

Since the idea of LU factoring is not sufficiently well defined to give a unique definition, it is hardly surprising that different definitions have been adopted by different systems. In the face of an incomplete definition, standardization becomes desirable. However, even for invertible matrices, three variations are commonly used for LU factors, since for any diagonal matrix D, we have $LU = (LD^{-1})(DU)$. The three variations are named after Cholesky, Doolittle and Crout[1]. Historically, Cholesky [6] and Doolittle [11] considered only symmetric systems of equations, because

[1]For biographical notes on André-Louis Cholesky (1875–1918), see [6]; on Myrick Hascall Doolittle (1830–1913), see [13]; on Prescott Durand Crout (1907 – 1984), see [1].

both were interested in least-squares corrections used in surveying, and it was only later that Crout [9] included non-symmetric equations in his treatment. Turing wrote an influential paper [23] stating LU factoring in its modern matrix notation, and explicitly connecting it to Gaussian elimination. Dwyer [12] also credits Banachiewicz[3] with anticipating matrix factoring ideas[2].

The first part of this paper defines a form for LU factors that is more useful than the variations at present offered by software systems, or books. The second part takes up fraction-free algorithms developed for linear system solving [14, 4]. There have been various attempts to apply the fraction-free idea to LU factoring [7, 19, 24]. Here we follow [24] and define a combined full-rank and fraction-free form.

2 Full-rank factoring and LU

Given a rectangular matrix A that is $m \times n$ and has rank r, a full rank factoring of A consists of two matrices F and G having dimensions $m \times r$ and $r \times n$ such that $A = FG$. An introduction to the full-rank factoring of a matrix can be found in [21]. Clearly, full-rank factoring is not unique.

In the context of exact computation, an obvious way to find the rank of a matrix is to use Gaussian elimination to reduce the matrix to row-echelon form and then to count the nonzero rows. Numerical linear algebraists will immediately point out that rank is difficult to calculate using approximate arithmetic, and that methods other than Gaussian elimination are preferred for the estimation of rank. This is an important point for implementation, but the general proposal here can be presented using Gaussian elimination as the basis. Since Gaussian elimination is equivalent to LU factoring [23], it is natural to extend LU factoring to non-square or rank-deficient matrices by using Gaussian elimination to obtain lower- and upper-triangular matrices L and U, and then to discard all zero rows in U and corresponding columns of L. For example, using MAPLE's LU factors given in (2), we discard the elements shown in bold to obtain a full-rank LU factoring.

One characteristic of a rank-deficient matrix is the possibility that in the factor U an echelon form will replace the purely triangular form of an invertible matrix, as can be seen in (2). By reordering the columns of U we can recover the more convenient form, and if a computer is performing the factoring, we can reasonably request the software to complete this small extra task for us. This suggests the following theorem, which is stated in a form that would be equally suitable for exact computation or approximate computation, provided again, we note the difficulty of computing rank numerically.

Theorem 1 *Given a matrix A with dimensions $m \times n$ and rank r, there exists a factoring*

$$A = P_r L U P_c \ , \tag{3}$$

where P_r is an $m \times m$ permutation matrix, L is an $m \times r$ lower triangular matrix, U is an $r \times n$ upper triangular matrix and P_c is an $n \times n$ permutation matrix. Furthermore, the structure of the matrix L is

$$L = \begin{pmatrix} \mathcal{L} \\ \mathcal{M} \end{pmatrix} \ ,$$

where $\mathcal{L}$ is an $r \times r$ lower triangular invertible matrix, and the structure of U is

$$U = \begin{pmatrix} \mathcal{U} & \mathcal{V} \end{pmatrix} \ ,$$

where $\mathcal{U}$ is $r \times r$, upper triangular, and invertible. Both $\mathcal{M}$ and $\mathcal{V}$ might be null.

Proof. This theorem is a special case of theorem 2, and it is proved as a corollary to that theorem in §3.1. □

2.1 Application to a generalized inverse

The generalized inverse of a matrix A is any matrix X satisfying $AXA = A$ [5]. In terms of the factoring (3), a generalized inverse (it is not unique) is

$$X = P_c^T \begin{pmatrix} \mathcal{U}^{-1}\mathcal{L}^{-1} & 0 \\ 0 & 0 \end{pmatrix} P_r^T \ .$$

This also satisfies the equation $XAX = X$, which is an alternative definition of a generalized inverse.

[2] Some historical quotations are given in [16, Chap. 9].

2.2 Application to analyzing rectangular systems

An application of the factoring that exists in any first course on matrix theory is the standard topic is deciding how many solutions there are to a given system of equations. Most books begin by listing three possibilities [2], namely, a system can have no solution, one solution or an infinite number of solutions; after that, they treat particular examples by reducing an augmented matrix to row-echelon form, and then apply an *ad hoc* analysis. With the new LU factors, the analysis is quick. Suppose there are m equations in n unknowns in the usual form $Ax = b$, with A having rank r. We obtain the full-rank LU factors:

$$Ax = P_r L U P_c x = b \ .$$

We first separate the bound and free variables, by writing $P_c x = [x_b \ x_f]^T$, with x_b being the r bound variables and x_f the $n-r$ free variables. We also separate the right-hand side into corresponding constants: $P_r^{-1} b = [b_b \ b_c]^T$. Now we can decide whether solutions exist by checking the consistency condition,

$$\mathcal{M}\mathcal{L}^{-1} b_b = b_c \ . \tag{4}$$

If this equation is satisfied, then the system is consistent, and we can write the bound variables in terms of the free variables as

$$x_b = (\mathcal{L}\mathcal{U})^{-1} b_b - \mathcal{U}^{-1}\mathcal{V} x_f \ .$$

Otherwise, there is no solution.

3 Fraction-free methods and LU

Fraction-free methods for the solution of systems of linear equations are well-established [4, 14], but the combination of fraction-free methods and LU factoring is more recent [19, 7, 24]. Here, we combine the full-rank factoring of the previous section with a fraction-free form. We extend the form defined in [24], rather than that in [7], because the matrices have smaller entries. The main theorem is the following.

Theorem 2 *A rectangular matrix A with elements from an integral domain $\mathbb{I}$, having dimensions $m \times n$ and rank r, may be factored into matrices containing only elements from $\mathbb{I}$ in the form*

$$A = P_r L D^{-1} U P_c = P_r \begin{pmatrix} \mathcal{L} \\ \mathcal{M} \end{pmatrix} D^{-1} \begin{pmatrix} \mathcal{U} & \mathcal{V} \end{pmatrix} P_c \ , \tag{5}$$

where permutation matrix P_r is $m \times m$, permutation matrix P_c is $n \times n$, $\mathcal{L}$ is $r \times r$, lower triangular and invertible.

$$\mathcal{L} = \begin{pmatrix} p_1 & & & \\ l_{21} & p_2 & & \\ \vdots & \vdots & \ddots & \\ l_{r1} & l_{r2} & \cdots & p_r \end{pmatrix} \ ,$$

where the $p_i \neq 0$ are the pivots in a Gaussian elimination. $\mathcal{M}$ is $(m-r) \times r$ and could be null. D is $r \times r$ and diagonal:

$$D = \mathrm{diag}(p_1 \, , p_1 p_2 \, , p_2 p_3 \, , \ldots p_{r-2} p_{r-1} \, , p_{r-1} p_r) \ .$$

$\mathcal{U}$ is $r \times r$ and upper triangular, while $\mathcal{V}$ is $r \times (n-r)$ and could be null.

$$\mathcal{U} = \begin{pmatrix} p_1 & u_{12} & \cdots & u_{1r} \\ & p_2 & \cdots & u_{2r} \\ & & \ddots & \vdots \\ & & & p_r \end{pmatrix} \ .$$

Proof. The standard treatments of fraction-free methods use determinants to obtain explicit formulae for the quantities involved. Rather than reproduce that approach, we here modify the standard recursive treatment of Gaussian elimination. The presentation uses an enhanced version of block-matrix notation for additional clarity.

The usual notation does not distinguish visually between the shapes of the blocks, but it helps to have reminders of which blocks are rows and which columns. An $m \times n$ matrix A will be written as

$$A = \begin{pmatrix} a_{11} & a_{12} & \overline{\boldsymbol{a}}_{13} \\ a_{21} & a_{22} & \overline{\boldsymbol{a}}_{23} \\ \underline{\boldsymbol{a}}_{31} & \underline{\boldsymbol{a}}_{32} & \mathbf{A}_{33} \end{pmatrix} , \tag{6}$$

where the elements $a_{11}, a_{12}, a_{21}, a_{22}$ are 1×1 scalars; the column vectors $\underline{\boldsymbol{a}}_{31}, \underline{\boldsymbol{a}}_{32}$ are $(m-2) \times 1$; the row vectors $\overline{\boldsymbol{a}}_{13}, \overline{\boldsymbol{a}}_{23}$ are $1 \times (n-2)$ and the submatrix $\mathbf{A}_{33}$ is $(m-2) \times (n-2)$.

Gaussian elimination as multiplication by elementary matrices is well known and treated at various levels of sophistication in everything from introductory to advanced texts [15, 22, 10]. Here we present a fraction-free variation which uses cross multiplication to avoid fractions, together with an exact division to reduce the growth in the size of elements. The fraction-free process requires that two steps be carried out explicitly, in order to see the complete process. The steps in the iterative scheme will be denoted by superscripts in parentheses. Thus the initial matrix is A, or $A^{(1)}$ with elements a_{ij} or $a_{ij}^{(1)}$.

We can assume that row and column reordering has ensured that the first pivot is $p_1 = a_{11}$. Transformation of the first column is written

$$E_1 A = \begin{pmatrix} 1 & 0 & \overline{\mathbf{0}} \\ -a_{21}^{(1)} & p_1 & \overline{\mathbf{0}} \\ -\underline{\boldsymbol{a}}_{31}^{(1)} & \underline{\mathbf{0}} & p_1\mathbf{I} \end{pmatrix} \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \overline{\boldsymbol{a}}_{13}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \overline{\boldsymbol{a}}_{23}^{(1)} \\ \underline{\boldsymbol{a}}_{31}^{(1)} & \underline{\boldsymbol{a}}_{32}^{(1)} & \mathbf{A}_{33}^{(1)} \end{pmatrix} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \overline{\boldsymbol{a}}_{13}^{(1)} \\ 0 & a_{22}^{(2)} & \overline{\boldsymbol{a}}_{23}^{(2)} \\ \underline{\mathbf{0}} & \underline{\boldsymbol{a}}_{32}^{(2)} & \boldsymbol{A}_{33}^{(2)} \end{pmatrix} .$$

Here $\boldsymbol{A}_{33}^{(2)} = p_1 \boldsymbol{A}_{33}^{(1)} - \underline{\boldsymbol{a}}_{31}^{(1)} \overline{\boldsymbol{a}}_{13}^{(1)}$, and similar formulae for the other elements. The second pivot, again supposing that row and column permutations have been completed, is $p_2 = a_{22}^{(2)}$.

$$\begin{aligned} E_2 E_1 A &= \begin{pmatrix} 1 & 0 & \overline{\mathbf{0}} \\ & 1 & \overline{\mathbf{0}} \\ & -\underline{\boldsymbol{a}}_{32}^{(2)} & p_2\mathbf{I} \end{pmatrix} \begin{pmatrix} 1 & 0 & \overline{\mathbf{0}} \\ -a_{21}^{(1)} & p_1 & \overline{\mathbf{0}} \\ -\underline{\boldsymbol{a}}_{31}^{(1)} & \underline{\mathbf{0}} & p_1\mathbf{I} \end{pmatrix} \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \overline{\boldsymbol{a}}_{13}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \overline{\boldsymbol{a}}_{23}^{(1)} \\ \underline{\boldsymbol{a}}_{31}^{(1)} & \underline{\boldsymbol{a}}_{32}^{(1)} & \mathbf{A}_{33}^{(1)} \end{pmatrix} \\ &= \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \overline{\boldsymbol{a}}_{13}^{(1)} \\ & a_{22}^{(2)} & \overline{\boldsymbol{a}}_{23}^{(2)} \\ & & \hat{\boldsymbol{A}}_{33} \end{pmatrix} . \end{aligned} \tag{7}$$

Here $\hat{\boldsymbol{A}}_{33}$ is an intermediate form. The well-known exact division is now established by expressing $\hat{\boldsymbol{A}}_{33}$ using elements from $A = A^{(1)}$.

$$\hat{\boldsymbol{A}}_{33} = a_{11}\left[(a_{11}a_{22} - a_{12}a_{21})\boldsymbol{A}_{33} + (a_{21}\underline{\boldsymbol{a}}_{32} - a_{22}\underline{\boldsymbol{a}}_{31})\overline{\boldsymbol{a}}_{13} + (a_{12}\underline{\boldsymbol{a}}_{31} - a_{11}\underline{\boldsymbol{a}}_{32})\overline{\boldsymbol{a}}_{23}\right] .$$

Since $p_1 = a_{11}$, the division $\hat{\boldsymbol{A}}_{33}/p_1$ is exact and the result is denoted $\boldsymbol{A}_{33}^{(3)}$. The same exact division occurs on the left of equation (7), as is seen by multiplying the elementary matrices together to obtain

$$E_2 E_1 = \begin{pmatrix} 1 & 0 & \overline{\mathbf{0}} \\ -a_{21}^{(1)} & p_1 & \overline{\mathbf{0}} \\ a_{21}^{(1)}\underline{\boldsymbol{a}}_{32}^{(2)} - p_2\underline{\boldsymbol{a}}_{31}^{(1)} & -p_1\underline{\boldsymbol{a}}_{32}^{(2)} & p_1p_2\mathbf{I} \end{pmatrix} .$$

Since $a_{21}^{(1)}\underline{\boldsymbol{a}}_{32}^{(2)} - p_2\underline{\boldsymbol{a}}_{31}^{(1)} = p_1[a_{21}^{(1)}\underline{\boldsymbol{a}}_{32}^{(1)} - a_{22}^{(1)}\underline{\boldsymbol{a}}_{31}^{(1)}]$, we can remove a factor p_1 from the third row (in a partitioned sense) of the matrices in the equation. That is, we can multiply by D_2^{-1}, where

$$D_2 = \begin{pmatrix} 1 & & \\ & 1 & \\ & & p_1\mathbf{I} \end{pmatrix} .$$

Thus we get

$$D_2^{-1}E_2E_1A = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \overline{\boldsymbol{a}}_{13}^{(1)} \\ & a_{22}^{(2)} & \overline{\boldsymbol{a}}_{23}^{(2)} \\ & & \hat{\boldsymbol{A}}_{33}/p_1 \end{pmatrix} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \overline{\boldsymbol{a}}_{13}^{(1)} \\ & a_{22}^{(2)} & \overline{\boldsymbol{a}}_{23}^{(2)} \\ & & \boldsymbol{A}_{33}^{(3)} \end{pmatrix} . \tag{8}$$

The iterative procedure is more clearly seen if we write $D_1 = I$ and then (8) becomes

$$D_2^{-1} E_2 D_1^{-1} E_1 A \quad = \quad A^{(3)} \ . \tag{9}$$

In words, step k of a fraction-free Gaussian elimination consists of a pivoting step (not shown), a cross multiplication step and a division by the pivot from step $k-1$.

The connection with LU factoring is now made, in a way similar to standard treatments, by inverting the matrices E_k. We have

$$E_1^{-1} \quad = \quad \begin{pmatrix} 1 & & \\ -a_{21}^{(1)} & p_1 & \\ -\underline{\boldsymbol{a}}_{31}^{(1)} & \underline{\mathbf{0}} & p_1\mathbf{I} \end{pmatrix}^{-1} = \begin{pmatrix} 1 & & \\ a_{21}^{(1)}/p_1 & 1/p_1 & \\ \underline{\boldsymbol{a}}_{31}^{(1)}/p_1 & \underline{\mathbf{0}} & \mathbf{I}/p_1 \end{pmatrix} \ .$$

The key new idea is now to write this as

$$E_1^{-1} \quad = \quad \begin{pmatrix} p_1 & & \\ a_{21}^{(1)} & 1 & \\ \underline{\boldsymbol{a}}_{31}^{(1)} & 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} p_1 & & \\ & p_1 & \\ & & p_1\mathbf{I} \end{pmatrix}^{-1} \ .$$

Including now all the matrices from the first two steps, we obtain

$$\begin{pmatrix} p_1 & & \\ a_{21}^{(1)} & 1 & \\ \underline{\boldsymbol{a}}_{31}^{(1)} & 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} p_1 & & \\ & p_1 & \\ & & p_1\mathbf{I} \end{pmatrix}^{-1} \begin{pmatrix} 1 & & \\ & p_2 & \\ & \underline{\boldsymbol{a}}_{32}^{(2)} & \mathbf{I} \end{pmatrix} \begin{pmatrix} 1 & & \\ & p_2 & \\ & & p_2\mathbf{I} \end{pmatrix}^{-1} \begin{pmatrix} 1 & & \\ & 1 & \\ & & p_1\mathbf{I} \end{pmatrix} \ ,$$

and completing the multiplications, we see

$$E_1^{-1} D_1 E_2^{-2} D_2 \quad = \quad \begin{pmatrix} p_1 & 0 & 0 \\ a_{21} & p_2 & 0 \\ \underline{\boldsymbol{a}}_{31} & \underline{\boldsymbol{a}}_{32}^{(2)} & \mathbf{I} \end{pmatrix} \begin{pmatrix} p_1 & 0 & 0 \\ 0 & p_1 p_2 & 0 \\ 0 & 0 & p_2\mathbf{I} \end{pmatrix}^{-1} \ . \tag{10}$$

In this form, the delayed division that led to (8) is now seen as limiting the occurrences of each pivot to two entries in the diagonal inverse matrix. It is clear that we can now continue recursively for r steps to obtain

$$D_r^{-1} E_r \ldots D_1^{-1} E_1 A \quad = \quad A^{(r)} \ . \tag{11}$$

By assumption, $A^{(r)}$ has r non-zero rows, and the rest can be discarded. The extension of (10) gives the LD^{-1} of the theorem, after trimming to r columns. We have $l_{ij} = a_{ij}^{(j)}$ and $u_{ij} = a_{ij}^{(i)}$. Adding in the permutation matrices is no different from the usual treatments [22, 16].

3.1 Corollary

Theorem 1 is obtained by combining either LD^{-1} together as L to obtain Doolittle LU factors, or $D^{-1}U$ together as U to obtain Crout LU factors. If A is symmetric, then D can be combined symmetrically to obtain Cholesky factors.

3.2 Example

Returning to the example above, we now have the factoring

$$\begin{pmatrix} 5 & 10 & 15 & 20 \\ -1 & -6 & -19 & -16 \\ 1 & 5 & 15 & 19 \\ 5 & 6 & -1 & -12 \\ 4 & 9 & 16 & 29 \end{pmatrix} = \tag{12}$$

$$\begin{pmatrix} 5 & 0 & 0 \\ -1 & -20 & 0 \\ 1 & 15 & -120 \\ 5 & -20 & 400 \\ 4 & 5 & -200 \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 \\ 0 & -100 & 0 \\ 0 & 0 & 2400 \end{pmatrix}^{-1} \begin{pmatrix} 5 & 10 & 20 & 15 \\ 0 & -20 & -60 & -80 \\ 0 & 0 & -120 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \ .$$

The above result was obtained using standard partial pivoting from numerical linear algebra. For exact computation, a pivoting strategy based on looking for the smallest non-zero pivot is desirable [14]. This gives the factoring

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -1 & -1 & 0 \\ 5 & -15 & 120 \\ 5 & -19 & 164 \\ 4 & -11 & 80 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -120 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 5 & 19 & 15 \\ 0 & -1 & 3 & -4 \\ 0 & 0 & 120 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} .$$

3.3 Application to fraction-free solving of rectangular systems

As in §2.2, rectangular systems can be checked for consistency and solved. The new feature is that the first steps can be performed fraction-free. Given

$$Ax = P_r \begin{pmatrix} \mathcal{L} \\ \mathcal{M} \end{pmatrix} D^{-1} \begin{pmatrix} \mathcal{U} & \mathcal{V} \end{pmatrix} P_c x = b \ ,$$

We have a simple variation on the usual procedure. We solve first $\mathcal{L}D^{-1}y = b_b$, where b_b is defined in § 2.2. We can see two ways that $D\mathcal{L}^{-1}$ is fraction-free. One is the proof given in [24] using determinants. The other way is to note that $D\mathcal{L}^{-1}$ is just another way of writing the matrices in (11), suitably trimmed. The evaluation of $\mathcal{U}^{-1}y$ must introduce fractions.

3.4 Moore—Penrose inverse

The Moore—Penrose inverse can be expressed in terms of full-rank factors, and using the factors (5) we can arrange matters so that most of the calculation is fraction free. It is shown in [5, 20] that if a matrix A has full-rank factors $A = FG$, then the Moore-Penrose inverse A^+ is given by

$$A^+ = G^*(GG^*)^{-1}(F^*F)^{-1}F^* \ ,$$

where the asterisk denotes Hermitian transpose. If we set $F = P_rLD^{-1}$ and $G = UP_c$, we can write

$$A^+ = [P_rL(UP_cA^*P_rL)^{-1}UP_c]^* \ ,$$

where the D matrix has disappeared from the calculation. The matrix $UP_cA^*P_rL$ can be computed within the domain of A and its inverse computed by fraction-free methods. Using the second numerical example, we have

$$UP_cA^*P_rL = \begin{pmatrix} 6852 & -14568 & 112288 \\ -226 & 684 & -6312 \\ 22920 & -45000 & 330240 \end{pmatrix} .$$

A fraction-free inverse is equivalent to computing the adjoint and determinant of the matrix. The final result is

$$A^+ = \frac{1}{3552948} \begin{pmatrix} 103020 & 133534 & -60579 & 142423 & 101249 \\ 113640 & 91774 & -50787 & 186937 & 82889 \\ -60540 & -300574 & 99747 & 35633 & -174689 \\ 34020 & 137214 & -13797 & -113337 & 136899 \end{pmatrix} .$$

4 Concluding remarks

The starting point for this paper was the existence of various *ad hoc* extensions to LU factoring. New or extended mathematical definitions introduced by software systems present difficulties to the software users. They lack documentation. The output from the system might not be understood, because it will not be described in textbooks, and as a consequence users may be unwilling to accept the result. Moreover, the lack of help could mean that users will miss the point and advantages of the new form. The aims here have been to propose two standards and to provide some documentation for them.

The treatment here has been directed towards exact computation, but the factoring should be useful in approximate systems also, although it would not be implemented in the same way as in exact systems. Pivoting, in particular, will most likely be different. Earlier, the numerical software MATLAB was included in the discussion. At present

MATLAB's `rank` and `rref` commands will calculate the rank of a matrix using a tolerance, which can optionally be set by the user. The algorithms used by the commands are different, and so the same tolerance can yield different estimates of rank for the same matrix. The MATLAB `lu` command, however, returns LU factors without using a tolerance to force zero. The new factoring would require that the `lu` command act more like the `rref` command for computations.

For symbolic systems, the rank could depend upon symbolic expressions, thus posing the usual dilemmas for software designers [8]. However, in common with other matrix factorings [7], expressions that are potentially zero will appear visibly in the results, alerting the user to possible special cases. The critical expressions will be those appearing on the diagonals of the factors. It can be noted that the diagonal matrix appearing in (5) contains the same information as the L an U matrices, and does not need separate examination; indeed, a system might not bother to return it explicitly at all.

There is a useful comparison to be made between the treatment of LU factors defined here, and the treatment of QR factors for non-square matrices [22]. For a matrix A, $m \times n$ with $m > n$, it is common to define either a "full" QR factoring $A = QR$, with Q being $m \times m$, or an "economy sized" factoring, in which Q is $m \times n$. The possible factors differ in that the R factor for the full case is padded with $m - n$ rows of zeros. The additional $m - n$ columns for the full Q are chosen so that it is orthonormal, although since the additional columns multiply zeros, it remains a useful convention not required by the factoring alone. It should be noted that, in contrast to the present definitions, the dimensions of the Q and R matrices do not depend upon the rank of the matrix. A desire to work with invertible, and thus square, matrices seems to be part of extended definitions in a number of places, including [7, 17], where the L matrix is padded to make it invertible. The view here is that this is not necessary, and indeed misses an opportunity.

References

[1] Anonymous. Contributors to the proceedings of the I.R.E. *Proceedings of the I.R.E.*, page 1328, November 1947.

[2] Howard Anton and Chris Rorres. *Elementary Linear Algebra*. Wiley, 9th edition, 2005.

[3] T. Banachiewicz. *Cracow Observatory Reprint: Études d'analyse pratique*, volume 22. University of Cracow, 1938.

[4] Erwin H. Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of Computation*, 22(103):565 – 578, 1968.

[5] A. Ben-Israel and T.N.F. Greville. *Generalized Inverses: Theory and Applications*. Wiley, 1974.

[6] Claude Brezinski. The life and work of André Cholesky. *Numer. Algorithms*, 43:279–288, 2006.

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

[8] Robert M. Corless and David J. Jeffrey. Well ... it isn't quite that simple. *SIGSAM Bulletin*, 26(3):2–6, 1992.

[9] P.D. Crout. A short method for evaluating determinants and solving systems of linear equations with real or complex coefficients. *Trans. Amer. Institute Elec. Engng*, 60:1235–1240, 1941.

[10] James W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[11] P.S. Dwyer. The Doolittle technique. *Ann. Math. Stat.*, 12:449–458, 1941.

[12] P.S. Dwyer. *Linear Computations*. Wiley, New York, 1951.

[13] R.W. Farebrother. A memoir of the life of M.H. Doolittle. *Bulletin Inst. Math. Applic.*, 23:102, 1987.

[14] K.O. Geddes, G. Labahn, and S. Czapor. *Algorithms for Computer Algebra*. Kluwer, 1992.

[15] Gene Golub and Charles Van Loan. *Matrix Computations, 2nd edition*. Johns Hopkins Press, 1989.

[16] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial & Applied Mathematics, 1996.

[17] Leslie Hogben, editor. *Handbook of linear algebra*. Chapman and Hall/CRC, 2007.

[18] Stephen B. Leacock. *Nonsense Novels*, chapter Gertrude the Governess. Wildside Press, 1911.

[19] George C. Nakos, Peter R. Turner, and Robert M. Williams. Fraction-free algorithms for linear and polynomial equations. *SIGSAM Bull.*, 31(3):11–19, 1997.

[20] R. Piziak and P. L. Odell. Full rank factorization of matrices. *Mathematics Magazine*, 72(3):193–201, 1999.

[21] Robert Piziak and P.L. Odell. *Matrix Theory: From Generalized Inverses to Jordan Form*. Chapman & Hall/CRC, 2007.

[22] Lloyd N. Trefethen and David Bau III. *Numerical Linear Algebra*. Society for Industrial & Applied Mathematics, 1997.

[23] Alan M. Turing. Rounding-off errors in matrix processes. *Quart. J. Mech. Appl. Math.*, 1:287 – 308, 1948.

[24] Wenqin Zhou and D.J. Jeffrey. Fraction-free matrix factors: new forms for LU and QR factors. *Frontiers of Computer Science in China*, 2(1):67–80, 2008.