The Turing factorization of a rectangular matrix

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

SIGSAM Bulletin, vol 31(3), September 1997, pp 20–28

Dedicated to the memory of M. A. H. Nerenberg

Abstract

The Turing factorization is a generalization of the standard LU factoring of a square matrix. Among other advantages, it allows us to meet demands that arise in a symbolic context. For a rectangular matrix A, the generalized factors are written $PA = LDUR$, where R is the row-echelon form of A. For matrices with symbolic entries, the $LDUR$ factoring is superior to the standard reduction to row-echelon form, because special case information can be recorded in a natural way. Special interest attaches to the continuity properties of the factors, and it is shown that conditions for discontinuous behaviour can be given using the factor D. We show that this is important, for example, in computing the Moore-Penrose inverse of a matrix containing symbolic entries.

We also give a separate generalization of LU factoring to fraction-free Gaussian elimination.

1 Introduction

It is now fifty years since Alan M. Turing wrote the seminal paper [16], in which he introduced several ideas that have since had a profound effect on numerical analysis, through their popularization and completion by Wilkinson [8]. One of the simplest, and yet most far-reaching, ideas was his recognition that direct methods for solving linear systems $Ax = b$ can be written as matrix factorizations, and in particular Turing proved that, once the 'partial pivoting' strategy is chosen, every square nonsingular matrix A can be written uniquely (up to ties for the choice of pivots) as the following 'resolution into the product of matrices'

$$PA = LDU \,, \tag{1}$$

where P is a permutation matrix, L is unit lower triangular with entries of magnitude bounded by 1, D is diagonal with nonzero entries, and U is unit upper triangular.

There are several 'standard' variants of LU factorization for square matrices; the main two group the LD factors together or instead group the DU factors together. Turing's choice to separate out the diagonal entries did not, unfortunately, become 'the' standard, which is a pity because it has some advantages, such as greater symmetry if A is symmetric. Incidentally, his use of the word 'resolution' where a modern author might use 'factorization' or 'decomposition' interchangeably, is another aspect of his paper that has been forgotten, but in this case we doubt whether any harm has been done. We shall use 'factorization' or 'factoring', in the main.

In this paper we generalize Turing's factorization to the case of rectangular matrices, and show that this helps to deal with the so-called specialization problem[1], which we now briefly discuss in the context of linear algebra.

Some of the standard operations of linear algebra, in particular the reduction to row-echelon form, present difficulties when implemented in computer algebra systems, because continuous input can lead to discontinuous output. We give an example below in which discontinuous behaviour is missed by a computer algebra system.

We emphasize that it is *discontinuous matrix properties* that are the ultimate source of the difficulty. Properties such as rank or nonsingularity are sometimes discontinuous, even when the matrix is continuous, and this presents problems both for exact and for floating point computation. The theory of 'conditioning' (initiated also by Turing in [16]) was developed to deal with discontinuity in the floating-point context. Perhaps because most computer algebraic systems take a formally algebraic viewpoint, no similarly useful and satisfactory theory has yet been developed for the case of exact computation.

The difficulty is, however, well known in algebra systems, and a number of approaches to it have been pro-

[1]We would be interested in printed references to this usage. The term 'specialization problem' appears to be well-known but we do not know a single paper, other than our own, which uses it. Further, there are other kinds of 'specialization problems' than the ones caused by removable discontinuities in expressions (see e.g. [15]), but we restrict ourselves to this type in this paper.

posed and discussed (see e.g. [3]). Many of the proposals are based on appropriate modifications of the user interface to the individual systems, but for the problems studied here, an alternative is to modify the mathematical setting so that special cases can be identified efficiently. We do this by formally replacing the notion of the 'row echelon reduction' with a factorization which preserves special-case information.

We also take the opportunity to write fraction-free Gaussian elimination (see e.g. [5]) as a factorization. This offers similar advantages in the case of parameters, but is more useful conceptually in that we feel this is a simpler presentation of the fraction-free algorithm, which avoids determinants and Sylvester's identity.

2 Row Echelon Reduction

We begin by discussing the difficulties associated with the traditional row-echelon reduction. This is taught to students as the transformation of a given matrix A into a succession of equivalent matrices by the use of row operations. The transformations stop when the unique row-echelon form R is reached. The shortcomings of this approach are illustrated by the following problem, adapted from a widely used textbook [12].

P.1) The matrix below is the augmented matrix for some system of equations:

$$\begin{pmatrix} 1 & -2 & 3 & \sin x \\ 1 & 4\cos x & 3 & 3\sin x \\ -1 & 3 & \cos x - 3 & \cos x \end{pmatrix} . \qquad (2)$$

Find the values of the parameter x for which the system has no, one, and infinitely many solutions.

Every CAS has a command to obtain the reduced row-echelon form of a matrix. In Maple, the command is `linalg[rref](A)`. In Maple V Release 3, this gave the error message 'matrix entries must be rational polynomials'. In Maple V Release 4, this gave the error message 'unable to find a provably non-zero pivot'.

This message shows the dichotomy between the analytic and algebraic viewpoints. If you replace $\cos x$ with c and $\sin x$ with s, then Maple V Release 4 will compute the row echelon form without a murmur, taking the algebraic viewpoint that the user is working over rational polynomials in indeterminates s and c. But given the information that s was $\sin x$ and c was $\cos x$, then the code recognized that the problem is analytic in nature, not algebraic, and refused to give a possibly wrong answer. This behaviour is not a very satisfactory way of handling the difficulty of discontinuities, and thus was changed for Release 5.

In Maple V Release 5, `rref(A)` gives

$$R(x) = \begin{bmatrix} 1 & 0 & 0 & \frac{2\cos(x)\sin(x) - 3\sin(x) - 6\cos(x) - 3}{2\cos(x)+1} \\ 0 & 1 & 0 & \frac{\sin(x)}{2\cos(x)+1} \\ 0 & 0 & 1 & \frac{2\cos(x)+2\sin(x)+1}{2\cos(x)+1} \end{bmatrix} . \qquad (3)$$

So now Maple takes the algebraic viewpoint that $\sin x$ and $\cos x$ are 'indeterminates'.

If, however, the user takes the analytic viewpoint, difficulties arise. By inspection it is clear that $2\cos x + 1 = 0$ is a special case. For illustrative purposes, only one root of $1 + 2\cos x = 0$ need be considered, so substituting $x = 2\pi/3$ into (2) and re-issuing the command `rref` gives

$$R(2\pi/3) = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & -1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \qquad (4)$$

Inspection of $R(x)$ in (3), however, gives no clue that there is a second special case, $x = \theta$, for which the row-echelon form is

$$R(\theta) = \begin{bmatrix} 1 & 0 & 3 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} . \qquad (5)$$

Identifying the appropriate value θ, which we will need in a later section, is left as an exercise for the reader. Note that this last row echelon form is necessary to answer the last part of the original textbook question.

Turing's idea of resolution of matrices into factors is now a mainstay of the modern view of linear algebra, with most of the important constructs being expressed this way (e.g. the FFT [17]). Row reduction stands out as an exception in being a transformation. Therefore, potentially zero divisors can be seen only by inspecting the intermediate working. The central difficulty, however, is the normalization required by row-echelon form in order to make it unique; this is a key property because many proofs in linear algebra rely on it. One way to tackle the problem would be to change the definition of row-echelon form to remove the need to normalize to 1, for example, by changing to the Hermite normal form for matrices of integers or univariate polynomials (see e.g. [11, p. 15]). However, the Hermite normal form is not defined for the general algebraic objects we wish to consider, such as our example above. In any event, the same difficulty of 'losing' a division by a parameter can be constructed for this normal form by considering polynomials with parameters in the coefficients.

By generalizing LU factoring to include the row-echelon form, we can resolve the difficulties described above. Any terms that would be divisors in a reduction to row-echelon form will appear somewhere in the factors, and therefore a computer system does not have to watch during the reduction to identify special cases for the user. Moreover, all information about the original matrix is preserved in a convenient form for the investigation of special cases.

3 The Turing factorization

Theorem 1: If A is an $m \times n$ matrix over $\mathbb{C}$, there exists a permutation matrix P, a unit lower triangular matrix L, a *nonsingular* diagonal matrix D, and a unit upper triangular matrix U such that

$$PA = LDUR \ , \tag{6}$$

where R is the (unique) row-echelon form of A. Though it is not *very* different from standard LU factoring of square matrices, we feel that the differences are important enough to give equation (6) a name. We call this the *Turing factorization* of A.

Proof: Noble and Daniel [12] prove that there exists a nonsingular matrix F such that

$$A = FR \ .$$

Apply Turing's LDU resolution [6, 16] to F, so we have

$$PF = LDU \ .$$

Note that since F is nonsingular, D is nonsingular. The Turing factorization (6) follows. ♮

We remark that if A is invertible then $R = I$. In this case, if we combine the factors D and U, then (one kind of) standard LU decomposition is obtained. If instead we combine the factors L and D, we get another standard. In the first combination, the nonsingularity requirement for D becomes a nonsingularity requirement for $U_1 = DU$ and this is what is implemented in the Maple routine `LUdecomp`[2]. It is worth noting that LU factoring requires no more computation than row-echelon reduction itself; all that is required is that the steps in the calculation be stored.

Our main theorem addresses the continuity of the row-echelon form. To motivate it, we return to the example problem and examine the row-echelon forms (2)–(4). If we take any point $x = a$ which is removed from the special cases, we clearly have

$$\lim_{x\to a} R(x) = R(a) \ .$$

[2]The routine `LUdecomp` was written by Michael Buckley, and was based in part on the program RowEchelon [4] from the share library. The paper [4] was never published in a journal, owing in part to the untimely death of Paddy Nerenberg. This present paper updates and corrects [4].

If, however, $a = \theta = \pi/2$ (this is the answer to our earlier exercise for the reader) we find

$$\lim_{x\to\pi/2} R(x) = \begin{pmatrix} 1 & 0 & 0 & -6 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 3 \end{pmatrix} \ ,$$

which clearly does not equal $R(\pi/2)$, which is given in equation (5). Thus the row-echelon form is discontinuous at this special case. Moreover, this is the essential feature we are looking for. Because the leading elements of each row have been normalized to 1, any change in the qualitative nature of the system, such as its rank, will be equivalent to a discontinuity. Finally, we notice that the original matrix is continuous at $x = \pi/2$. We thus need a theorem characterizing when the row-echelon form of a matrix is discontinuous, even though the matrix itself is continuous.

Definition. A matrix $A(x)$ is continuous at $x = a$ if each of its elements is continuous at $x = a$.

Theorem 2: Let $A(x)$ be a matrix depending upon one or more variables or parameters x, and let A be continuous at a point $x = a$. For any fixed x let $A(x)$ have the Turing factorization $P(x)A(x) = L(x)D(x)U(x)R(x)$. If $\det D(x) \neq 0$ in some neighbourhood of $x = a$, then $R(x)$, $L(x)$, $D(x)$, $U(x)$ are all continuous at $x = a$ and moreover $P(x)$ may be taken constant in a neighbourhood of $x = a$.

Proof. We may choose $P(x)$ so that all denominators of $L(x)$ introduced by the factorization—that is, not present already in $A(x)$—are factors of $\det D(x)$. This is a consequence of the construction of P, L, D, and U by Gaussian elimination of $F(x)$ with partial pivoting.

Since $\det D(x) \neq 0$ in a neighbourhood of $x = a$ by hypothesis, D^{-1} exists and is continuous in a (possibly smaller) neighbourhood of $x = a$ because D^{-1} is made up of rational polynomial combinations of elements of $A(x)$, which are continuous at $x = a$ and hence in some neighbourhood of $x = a$. Similarly since $\det L(x) = \det U(x) = 1$, we have that L^{-1} and U^{-1} exist and are continuous in a neighbourhood of $x = a$.

Now $P(x)$ may be taken constant in a neighbourhood of $x = a$ because the pivots that arise in Gaussian elimination with partial pivoting can have only a finite number of zeros in a small enough neighbourhood of $x = a$, because otherwise by continuity of $A(x)$ we would have $\det D(a) = 0$ which contradicts the hypothesis.

Therefore we have

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

and everything on the right hand side is continuous at $x = a$. ♮.

Corollary 1: The only places where R can be discontinuous are those where $\det D = 0$.

We now illustrate some of the theorem's consequences.

3.1 Examples

3.1.1 Example 1.

Parameterized matrices arise naturally in eigenvalue problems, such as the following. The example is chosen to illustrate that, even if an individual element of D is zero, it is only when $\det D = 0$ that discontinuous changes in the row echelon form may occur.

For ease of typography in this two-column format we combine D and U and use $U_1 = DU$.

Consider

$$B = \begin{pmatrix} 2 & 0 & 0 \\ 2 & 3 & 1 \\ -4 & -2 & 0 \end{pmatrix}$$

To find its eigenvalues and eigenvectors, we examine the row-echelon decomposition of $A(\lambda) = B - \lambda I$. We have $IA(\lambda) =$

$$\begin{pmatrix} 1 & 0 & 0 \\ \frac{2}{2-\lambda} & 1 & 0 \\ \frac{4}{\lambda-2} & \frac{2}{\lambda-3} & 1 \end{pmatrix} \begin{pmatrix} 2-\lambda & 0 & 0 \\ 0 & 3-\lambda & 1 \\ 0 & 0 & \frac{\lambda^2-3\lambda+2}{3-\lambda} \end{pmatrix} I.$$

In the case $\lambda = 2$, the first diagonal element of U is zero, and when $\lambda = 3$ the second diagonal element is zero. However, to find eigenvalues, we must test $\det A(\lambda) = \det U_1 = (1-\lambda)(2-\lambda)^2$. Thus $\lambda = 3$ is not an eigenvalue and not a case in which the row-echelon form changes; instead, when $\lambda = 3$, a row exchange could be introduced when calculating the decomposition to obtain the same row-echelon form as above, but with non-zero diagonal entries in U_1. For each distinct eigenvalue we must re-factorize. Further, the row echelon form allows immediate calculation of the desired eigenvectors, because $(A - \lambda I)x = 0$ if and only if $Rx = 0$, since P, L, and U_1 are nonsingular. For $\lambda = 2$ the row-echelon form of $A(\lambda)$ is

$$R(2) = \begin{pmatrix} 1 & 1/2 & 1/2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

showing that there are two linearly independent eigenvectors for this eigenvalue. The case $\lambda = 1$ is treated similarly. This example was constructed using the method of [13].

3.1.2 Example 2.

This example shows that even when $\det U_1 = 0$ the row-echelon form might still be continuous. Consider

$$I \cdot A(a,b,c) = \begin{pmatrix} 1 & a \\ 1 & b \\ 1 & c \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & \frac{a-c}{a-b} & 1 \end{pmatrix} \begin{pmatrix} 1 & a & 0 \\ 0 & b-a & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

Clearly $\det U_1 = 0$ when $b = a$. When, however, we recalculate the decomposition, we find $P_{23}A(a,a,c) =$

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & a & 0 \\ 0 & c-a & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix},$$

where

$$P_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

The row-echelon form is unchanged *unless also* $c = a$, in which case we get

$$I \cdot A(a,a,a) = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & a \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

This example shows that our theorem is as strong as possible, in that parameter values rendering U_1 singular can only be said to "need further investigation".

4 The Moore-Penrose Inverse

Different methods for computing generalized inverses are discussed in [10]. We begin our treatment by following [6]. Given the SVD of the m by n matrix A, namely

$$A = U\Sigma V^* , \tag{7}$$

with $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r, 0, \ldots, 0)$ we may define the n by m *pseudo-inverse* A^+ by

$$A^+ = V\Sigma^+U^* \tag{8}$$

where $\Sigma^+ = \mathrm{diag}(1/\sigma_1, 1/\sigma_2, \ldots, 1/\sigma_r, 0, \ldots, 0)$. If $\mathrm{rank}(A) = n$, then $A^+ = (A^TA)^{-1}A^T$, while if $m = n = \mathrm{rank}(A)$ then $A^+ = A^{-1}$.

The pseudo-inverse satisfies (and is in fact defined by) the four Moore-Penrose conditions:

$$\begin{aligned} AA^+A &= A & (9) \\ A^+AA^+ &= A^+ & (10) \\ (AA^+)^* &= AA^+ & (11) \\ (A^+A)^* &= A^+A\,. & (12) \end{aligned}$$

The definition (8) is very useful for computing the Moore-Penrose inverse of a matrix A containing numerical entries. But it is of occasional interest to compute A^+ for a matrix containing symbolic entries, and the computation of explicit formulas for the entries of the SVD of a symbolic matrix is usually impossible.

The following formula, which is based on a regularization procedure associated with the name of Tihonov [7], is more suited to symbolic computation (although as we will see it has severe limitations of its own):

$$A^+ = \lim_{t\to 0} A^*(AA^* + tI)^{-1} . \tag{13}$$

This formula follows easily from (8), because the right hand side of (13) may be written

$$V\Sigma U^* \left(U(\Sigma^2 + tI)U^*\right)^{-1} \tag{14}$$

and from here it becomes

$$V \mathrm{diag}\left(\sigma_1/(\sigma_1^2+t), \ldots, \sigma_r/(\sigma_r^2+t), 0, \ldots, 0\right) U^* . \tag{15}$$

and it is clear that $t \to 0$ in this will give A^+.

Remark 1. The above computation shows that the limit does not have to be taken one-sided, because all the entries in $\Sigma^+(t)$ are well-defined for $|t| < \sigma_r$.

Remark 2. It is very important to notice that this formulation starts with a *constant* matrix A. But the pseudo-inverse is *not continuous*, and so if we try to find the pseudo-inverse of a parameterized family of matrices, $A(x)$ say, the family of inverses might contain gaps or 'holes'.

Consider the following simple example:

$$A = \left[\begin{array}{cc} 1 & e \\ e & 1 \end{array}\right] . \tag{16}$$

The Moore-Penrose inverse is, if $e \neq 1$,

$$A^+ = \frac{1}{1-e^2}\left[\begin{array}{cc} 1 & -e \\ -e & 1 \end{array}\right] , \tag{17}$$

which is of course just the ordinary inverse of A. But if $e = 1$, we have

$$A^+ = \frac{1}{4}\left[\begin{array}{cc} 1 & 1 \\ 1 & 1 \end{array}\right] . \tag{18}$$

Notice that the limit as e goes to 1 of the general form (17) does not exist. This example demonstrates that

$$\left(\lim_{e\to 1} A\right)^+ \neq \lim_{e\to 1} A^+ . \tag{19}$$

Therefore, the Moore-Penrose inverse of a symbolic matrix will have 'specialization' problems—we may expect that a routine that computes the general form will give us an answer that is not necessarily correct for all values of the parameters. The following theorem addresses this issue in a constructive way, using provisos.

Theorem 3: The following are equivalent.

1. $A^+(x)$ (where $A : D \subset \mathbb{R}^\ell \to M_{mn}(\mathbb{R})$) is continuous at $x = a \in D$.
2. rank(A) is constant in a neighbourhood of $x = a$, and the A^+ given by formula (13) is correct in this neighbourhood.
3. $\det(D(x)) \neq 0$ in a neighbourhood of $x = a$, where $A(x) = PL(x)D(x)U(x)R(x)$ is the Turing factorization of $A(x)$. Again, the A^+ given by formula (13) is correct in this neighbourhood.

Proof: Consider the Singular Value Decomposition of $A(x)$, namely $A = U(x)\Sigma(x)V^*(x)$. The only requirement for the Tihonov formula to be correct, by reasoning similarly to the constant matrix case, is that $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_r > 0 = \sigma_{r+1} = \cdots \sigma_n$. This is equivalent to saying that rank $A(x) = r$. Finally, rank(A) is constant whenever $\det(D) \neq 0$ by the Turing factorization theorem. ♮

4.1 Maple Code

The following Maple subroutine computes the Moore-Penrose inverse and its proviso.

```
MoorePenrose := proc(A::matrix, proviso::name)
  local T,t, Ah;
  Ah := map(r->evalc(conjugate(r)),
             linalg[transpose](A));
  linalg[LUdecomp](A, 'det'=proviso);
  T := evalm( Ah &* (A &* Ah + t*&*() )^(-1) );
  map(limit, eval(T), t=0, right)
end:
```

It uses the long names `linalg[transpose]` and `linalg[LUdecomp]` so the routine will work even if `linalg` is not loaded in. Names are assumed to be real-valued by `evalc`, by default. The shorthand notation `&*()` is Maple's syntax for an appropriately-sized identity matrix.

4.2 Examples

It turns out to be convenient to verify that our computed Moore-Penrose inverse satisfies the the four Moore-Penrose conditions, by using the following procedure. [It was very useful in debugging, for example.]

```
MoorePenroseConditions :=
  proc( A::matrix, Ap::matrix )
  local herm;
  herm := m -> map(r->evalc(conjugate(r)),
                      linalg[transpose](m));
  print(map(normal,evalm( A &* Ap &* A - A )));
  print(map(normal,evalm( Ap&*A&*Ap - Ap)));
  print(map(normal@evalc,
             evalm( herm(A&*Ap) - A &* Ap)));
  print(map(normal@evalc,
             evalm( herm(Ap&*A) - Ap &* A)));
end:
```

The idea is that this procedure prints out four possibly differently-shaped zero matrices; if any of these matrices contains a nonzero entry, there may be a bug (but most probably just a weakness in zero-recognition). The following Maple session explores these routines.

```
> a := alpha[1] + I*alpha[2];
```

$$a := \alpha_1 + I\,\alpha_2$$

```
> b := beta[1] + I*beta[2];
```

$$b := \beta_1 + I\,\beta_2$$

```
> A := matrix(1,2,[a,b]);
```

$$A := \begin{bmatrix} \alpha_1 + I\,\alpha_2 & \beta_1 + I\,\beta_2 \end{bmatrix}$$

```
> M := MoorePenrose(A, 'prA');
```

$$M := \begin{bmatrix} -\dfrac{-\alpha_1 + I\,\alpha_2}{{\alpha_1}^2 + {\alpha_2}^2 + {\beta_1}^2 + {\beta_2}^2} \\ -\dfrac{-\beta_1 + I\,\beta_2}{{\alpha_1}^2 + {\alpha_2}^2 + {\beta_1}^2 + {\beta_2}^2} \end{bmatrix}$$

```
> MoorePenroseConditions(A, M);
```

$$\begin{bmatrix} 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
> B := matrix(2,2,[1,e,e,1]);
```

$$B := \begin{bmatrix} 1 & e \\ e & 1 \end{bmatrix}$$

```
> M := MoorePenrose(B,'prB');
```

$$M := \begin{bmatrix} -\dfrac{1}{-1+e^2} & \dfrac{e}{-1+e^2} \\ \dfrac{e}{-1+e^2} & -\dfrac{1}{-1+e^2} \end{bmatrix}$$

```
> prB;
```

$$1 - e^2$$

```
> MoorePenroseConditions(B,M);
```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
> C := subs(e=1,eval(B));
```

$$C := \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

```
> MC := MoorePenrose(C,'prC');
```

$$MC := \begin{bmatrix} \dfrac{1}{4} & \dfrac{1}{4} \\ \dfrac{1}{4} & \dfrac{1}{4} \end{bmatrix}$$

```
> prC;
```

$$1$$

```
> MoorePenroseConditions(C,MC);
```

This yields four 4 by 4 matrices containing only zeros.

4.3 Limitations

This routine uses symbolic inversion (of $AA^* + tI$) as a tool to compute the symbolic Moore-Penrose inverse. As is well-known, exact arithmetic solutions, and even more so exact symbolic solutions, lead very quickly to computationally intractable problems. Nonetheless if your problem contains only one or two parameters, and isn't of too high a dimension, then efficiency and insight can be gained by using a symbolic inverse or Moore-Penrose inverse. For more discussion and examples, see [2].

5 Fraction-free LU factoring

In this section we generalize the $PA = LU$ factorization to the fraction-free case, which appears not to have been done before. The paper [9] is based on a poster presented at ECCAD 97, in which they gave something they called (in quotes) a fraction-free 'factorization'; we modify this here to give a true factorization.

However, after writing down the true factorization (20 below), one realizes that the information in the extra factors F_1 and F_2 are duplicated in the U factor, and hence there is no need to form them explicitly except for conceptual understanding, and thus we see that while they do not give a true factorization, the treatment of [9] is complete and practical.

Nonetheless, while no new information is discovered this way, this true factorization approach has some advantages. First, it avoids Sylvester's Identity, and indeed avoids determinants altogether. It also, in our view, gives a clearer explanation of just why we can pull common integer factors out of certain submatrices, which is the key to the whole algorithm.

We first see a theorem and then work out an example in detail. The proof of the theorem follows the reasoning in the example and is thus omitted. Maple code for the fraction-free factorization, which works for matrices with entries from arbitrary integral domains, can be found in [2].

Theorem 4: *Fraction-Free Factorization.* Consider the rectangular matrix $A \in \mathbb{Z}^{n\times m}$. Then we may write

$$F_1 PA = LF_2 U \; , \qquad (20)$$

where $F_1 = \mathrm{diag}(1, p_1, p_1p_2, \ldots, p_1p_2\cdots p_{n-1})$, P is a permutation matrix, $L \in \mathbb{Z}^{n\times n}$ is unit lower triangular, $F_2 = \mathrm{diag}(1, 1, p_1, p_1p_2, \ldots, p_1p_2\cdots p_{n-2})$, and $U \in \mathbb{Z}^{n\times m}$ is upper triangular. The p_i are the pivots that arise.

Remarks.

- This factorization (or rather, its construction by algorithm) gives a simple proof of divisibility of the submatrices by p_1, p_2, and so on. It becomes clear that since we put the factors in, with F_1, and we ought to be able to take them out again, with F_2.

- We may use either the one-step, or the two-step, fraction free algorithm (see [5] for details) to construct the factorization, which is the same in either case. Since the two-step method is asymptotically more efficient than the one-step method, we should use that. For clarity, we do not.

- Formation of the factors in F_i is not actually necessary. We may simply record the pivots p_1, p_2, and in fact even this is not necessary, since the pivots are already recorded in the diagonal entries of U.

- The determinants of each side are
$$p_1^{n-1}p_2^{n-2}\cdots p_{n-1}\det(A)$$
and
$$p_1^{n-2}p_2^{n-3}\cdots p_{n-2}\det(U)$$
respectively. This shows that
$$\det(U) = p_1p_2\cdots p_{n-1}\det(A) \; .$$

- If zero pivots are encountered, or indeed if we wish to select the smallest pivot so as to encourage the least growth in the integers that arise (this heuristic is well-known and works very well), then we must do row exchanges. By the usual trick, we may pretend that we knew ahead of time which rows would be exchanged, and do them all to start with. This is why the P factor is next to the A factor in the statement of the theorem. The details of the percolation of the permutations (via column exchanges and row exchanges) through all the various factors are left as an exercise.

- The factor U is different from that of the Turing factorization, in that it is not unit upper triangular.

- As with the Turing factorization, once the pivoting strategy has been chosen then the factorization is (up to ties for pivots) unique.

- The integral domain $\mathbb{Z}$ is not special; this factorization works for matrices over any integral domain.

- We may combine the Fraction-Free Factorization with the Turing factorization to get $F_1PA = LF_2UF_3H$, which gives a a factorization for the Hermite normal form H.

5.1 Example

We use example 9.1 from [5], which has the augmented matrix

$$A = \begin{bmatrix} 3 & 4 & -2 & 1 & -2 \\ 1 & -1 & 2 & 2 & 7 \\ 4 & -3 & 4 & -3 & 2 \\ -1 & 1 & 6 & -1 & 1 \end{bmatrix} . \tag{21}$$

In what follows we appear to temporarily allow divisions. This is a notational device only, for exposition, and it should be clear how to avoid ever forming any fractions even temporarily.

Applying one elementary matrix step of ordinary $PA = LU$ factorization to this matrix would give $A =$

$$\begin{bmatrix} 1 & & & \\ 1/3 & 1 & & \\ 4/3 & & 1 & \\ -1/3 & & & 1 \end{bmatrix} \begin{bmatrix} 3 & 4 & -2 & 1 & -2 \\ & -7/3 & 8/3 & 5/3 & \frac{23}{3} \\ & -\frac{25}{3} & \frac{20}{3} & -13/3 & 14/3 \\ & 7/3 & 16/3 & -2/3 & 1/3 \end{bmatrix} \tag{22}$$

To remove the fractions, we may rewrite the identity matrix as

$$I = \begin{bmatrix} 1 & & & \\ & 1/3 & & \\ & & 1/3 & \\ & & & 1/3 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 3 & & \\ & & 3 & \\ & & & 3 \end{bmatrix} \tag{23}$$

and insert these two factors in between the two factors of A we have found so far. Since multiplication by a diagonal matrix on the right multiplies columns, the 1's on the diagonal of the L factor all become 1/3. Once this happens, we may factor 1/3 out of each row, giving

$$A = \begin{bmatrix} 1 & & & \\ & 1/3 & & \\ & & 1/3 & \\ & & & 1/3 \end{bmatrix} \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ 4 & & 1 & \\ -1 & & & 1 \end{bmatrix} \times \begin{bmatrix} 3 & 4 & -2 & 1 & -2 \\ & -7 & 8 & 5 & 23 \\ & -25 & 20 & -13 & 14 \\ & 7 & 16 & -2 & 1 \end{bmatrix}$$

Of course, multiplying both sides by $\mathrm{diag}(1, 3, 3, 3)$ will remove the fractions completely. So far, we have not captured the essence of the Bareiss-Jordan fraction-free algorithm; all we have done is cleared fractions in ordinary $PA = LU$ factorization. Indeed, this is just the beginning of what is called 'division-free Gaussian elimination' in [5]. [Apparently, 'division-free' is the accepted name for a slightly different algorithm, which is less clever than the 'fraction-free' algorithm. We will not have cause to refer to 'division-free' elimination again.] We need to do one more step before the idea of 'fraction-free factorization' becomes clear. Call the last factor in the above equation, $A^{(1)}$. We will work just with $A^{(1)}$, for easy typography.

We start as before by pretending to use ordinary rational LU factorization steps. We may write $A^{(1)} =$

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & \frac{25}{7} & 1 & \\ & -1 & & 1 \end{bmatrix} \begin{bmatrix} 3 & 4 & -2 & 1 & -2 \\ & -7 & 8 & 5 & 23 \\ & & -\frac{60}{7} & -\frac{216}{7} & -\frac{477}{7} \\ & & 24 & 3 & 24 \end{bmatrix} \tag{24}$$

and again we will wish to clear the fractions (accidentally, the last multiplier was also 7 and so the entries in the last row are integers, but in general this will not happen). In actual fact the pivot was -7, so we'll adjust the minus signs above, and use a similar rewriting of the identity (this time as $\mathrm{diag}(1, 1, -1/7, -1/7)\mathrm{diag}(1, 1, -7, -7)$) to arrive at

$$A^{(1)} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & -1/7 & \\ & & & -1/7 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & -25 & 1 & \\ & 7 & & 1 \end{bmatrix} \times \begin{bmatrix} 3 & 4 & -2 & 1 & -2 \\ & -7 & 8 & 5 & 23 \\ & & 60 & 216 & 477 \\ & & -168 & -21 & -168 \end{bmatrix}$$

The percolation of the $\mathrm{diag}(1, 1, -1/7, -1/7)$ factor through the left-hand factors already obtained is left as an exercise. What concerns us now is the fact that every element of the remaining non-triangular submatrix, namely

$$\begin{bmatrix} 60 & 216 & 477 \\ -168 & -21 & -168 \end{bmatrix} , \tag{25}$$

is divisible by 3.

A moment's thought shows why this must be so. We introduced the factor 3 into the entire submatrix when we multiplied by $\mathrm{diag}(1, 3, 3, 3)$. It was needed for the elimination of the first column, but is not needed here. We may now factor it out, to keep the size of the elements down.

It is important to note that we will not need to do GCD calculations to discover this divisibility; we will know ahead of time that this divisibility will happen. Thus we may cheaply take advantage of it.

Writing this observation as a factorization, we have that $\mathrm{diag}(1,1,-7,-7)A^{(1)}$

$$= \begin{bmatrix} 1 & & & \\ & 1 & & \\ & -25 & 1 & \\ & 7 & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 3 & \\ & & & 3 \end{bmatrix} \times \begin{bmatrix} 3 & 4 & -2 & 1 & -2 \\ & -7 & 8 & 5 & 23 \\ & & 20 & 72 & 159 \\ & & -56 & -7 & -56 \end{bmatrix}.$$

Continuing with one more step, and rearranging all the factors that we have so far found, we get at last that

$$\begin{bmatrix} 1 & & & \\ & 3 & & \\ & & -21 & \\ & & & -420 \end{bmatrix} A = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ -28 & -25 & 1 & \\ 140 & 140 & -56 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 3 & \\ & & & -21 \end{bmatrix} \begin{bmatrix} 3 & 4 & -2 & 1 & -2 \\ & -7 & 8 & 5 & 23 \\ & & 20 & 72 & 159 \\ & & & -556 & -1112 \end{bmatrix}.$$

6 Concluding Remarks

Recent releases of Maple offer a command `LUdecomp` that returns the generalized LU factoring described here. The proviso $\det(U_1) \neq 0$ allows the user (at least in principle) to identify the values of the parameters that give exceptional row echelon forms, and the program can be called again with these special values of the parameters.

This idea of a row echelon decomposition may be useful numerically, as well, if we make the usual shift to looking at the condition number of U_1 instead of the determinant. The reader will be interested to note that the idea of condition number, and the definition of 'ill-conditioned', was also introduced by Turing, in the same paper [16]. Thus we hope that the name 'Turing factorization' for the result of Theorem 1 is accepted.

Further, the Turing factorization can be used to solve all the major computational problems encountered in a first linear algebra course, such as solution of linear equations, positive definiteness of quadratic forms, Gram-Schmidt orthogonalization (see [14]), and eigenvalues. This elegant unity is surely a consequence of Turing's mathematical taste; even in the simple fields, he was attracted to the key ideas.

References

[1] B. W. Char, K. O. Geddes, G. H. Gonnet, M. B. Monagan, and S. M. Watt, *The Maple V Language Reference Manual*, Springer, (1991).

[2] Robert M. Corless, *Symbolic Recipes*, Volume I: Exact Computation, Springer, to appear (1998).

[3] Robert M. Corless and David J. Jeffrey, "Well, it isn't quite that simple", SIGSAM BULLETIN, Vol. 26, no. 3, issue 101, *pp.* 2–6 (1992).

[4] R. M. Corless, D. J. Jeffrey, & M. A. H. Nerenberg, "The Row Echelon Decomposition", TR AM-91-01, Dept. Applied Math, U. W. O. (1991).

[5] K.O. Geddes, G. Labahn, S. Czapor, *Algorithms for Computer Algebra*, Kluwer, (1992).

[6] Gene Golub and Charles Van Loan. *Matrix Computations.* Johns Hopkins, 2nd edition, (1989).

[7] C. W. Groetch, *Generalized inverses of linear operators*, Monographs and texts in pure & applied mathematics, Vol. 37, Marcel Dekker, New York, (1977).

[8] Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, (1996).

[9] G. C. Nakos, P. R. Turner, & R. M. Williams, "Fraction-Free algorithms for linear and polynomial equations", this BULLETIN, *pp.* 11–19, (1997).

[10] M.-T. Noda, I. Makino, & T. Saito, "Algebraic Methods for Computing a Generalized Inverse", this BULLETIN, *pp.* 51–52, (1997).

[11] Morris Newman, *Integral Matrices*, Academic Press, (1972).

[12] Ben Noble and James W. Daniel, *Applied Linear Algebra*, 3rd ed., Prentice-Hall, (1988).

[13] J. M. Ortega, Letters to the Editor, American Mathematical Monthly **92**, *p.* 526 (1985).

[14] Lyle Pursell and S. Y. Trimble, "Gram-Schmidt Orthogonalization by Gauss Elimination", American Mathematical Monthly **98** no. 6, *pp.* 544–549 (1991).

[15] David R. Stoutemyer, "Crimes and Misdemeanors in the Computer Algebra Trade", Notices of the A. M. S. **38** No. 7, *pp.* 778–785 (1991).

[16] Alan M. Turing, "Rounding-off errors in matrix processes", Quart. J. Mech. Appl. Math. 1, *pp.* 287–308, (1948).

[17] Charles F. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, 1992.