9 Linear Systems of Equations and LU Factorization

Many computational problems require the solution of linear systems of equations

$$A\mathbf{x} = \mathbf{b} \tag{1}$$

with an $n \times n$ matrix

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n-1} & a_{2,n} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

and a right-hand side

$$\mathbf{b} = [b_1, b_2, \ldots, b_n]^T.$$

The matrix A typically represents a model and the right-hand side $\mathbf{b}$ contains data to which one seeks to fit the model.

The number of equations, n, can be large in many scientific applications. For instance, problems in structural analysis frequently give rise to linear systems of equations of order about 10^6. These systems are commonly solved on computers with several processors. Some applications require the solution of linear systems of equations with 10^8 or more unknowns. The solution of so large systems is a challenge even on the fastest computers available. This lecture discusses methods for the solution of linear systems of equations based on Gaussian elimination.

9.1 Gaussian elimination and LU factorization

The most commonly used methods for solving linear systems of equations are based on Gaussian elimination. Gaussian elimination is a method for transforming a linear system of equations (1) to an equivalent system of equations

$$U\mathbf{x} = \tilde{\mathbf{b}} \tag{2}$$

with an upper triangular matrix

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n-1} & u_{1,n} \\ & u_{2,2} & \cdots & u_{2,n-1} & u_{2,n} \\ & & \ddots & \vdots & \vdots \\ & & & u_{n-1,n-1} & u_{n-1,n} \\ & & & & u_{n,n} \end{bmatrix}.$$

and a modified right-hand side vector $\tilde{\mathbf{b}}$. Linear systems with an upper triangular matrix are easy to solve by back substitution.

Frequently, one would like to apply the same model to more than one set of measured data vectors (right-hand sides). One then needs to solve the system (1) for several right-hand sides, all of which may not be available at the same time. The upper triangular matrix U is independent of the right-hand side $\mathbf{b}$ in (1), however, the vector $\tilde{\mathbf{b}}$ in the upper triangular system (2) is not. We would like to determine the vector $\tilde{\mathbf{b}}$ associated with each right-hand side $\mathbf{b}$ without having to recompute U. It turns out that this is possible

and computationally inexpensive. It can be achieved by storing some auxiliary quantities, computed during the evaluation of U from A, in an $n \times n$ lower triangular matrix with unit diagonal,

$$L = \begin{bmatrix} 1 & & & & & \\ \ell_{2,1} & 1 & & & & \\ \ell_{3,1} & \ell_{3,2} & 1 & & & \\ \vdots & \vdots & & \ddots & & \\ \ell_{n-1,1} & \ell_{n-1,2} & & \cdots & 1 & \\ \ell_{n,1} & \ell_{n,2} & & \cdots & & 1 \end{bmatrix}.$$

Then we have

$$A = LU. \tag{3}$$

This factorization is called the *LU factorization* of A.

Given the LU factorization of the matrix A, we can solve the linear system (1) in two steps: substitute (3) into (1) to obtain

$$LU\mathbf{x} = \mathbf{b},$$

and then solve the triangular systems, in order,

$$L\mathbf{y} = \mathbf{b}, \tag{4}$$
$$U\mathbf{x} = \mathbf{y}. \tag{5}$$

The first system is solved by computing the components of $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$ in the order of increasing index, i.e., y_1 is computed from the first equation of (4), then y_2 is computed from the second equation of (4), and so on, until the last component y_n is computed from the last equation in (4). This solution method is known as forward substitution. One can show that the vector $\mathbf{y}$ obtained in this manner is the vector $\tilde{\mathbf{b}}$ in (2).

9.2 Computation of the LU factorization

We illustrate the computation of the triangular factors L and U with a 4×4 matrix. Let

$$A = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix}.$$

The first step of Gaussian elimination entails subtracting suitable multiples of the first row from rows 2 through 4, so that the $(2,1)$, $(3,1)$, and $(4,1)$ entries in the matrix so obtained vanish. We store these multiples in the first column below the diagonal of a lower triangular matrix L with unit diagonal. Thus, we subtract 2 times the first row from the second row and store the multiple 2 in the $(2,1)$-entry of L. This gives the matrices

$$A' = \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & 3 & 5 & 5 \\ & 4 & 6 & 8 \end{bmatrix}, \qquad L = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 4 & & 1 & \\ 3 & & & 1 \end{bmatrix}.$$

To proceed, we subtract suitable multiples of the second row of the matrix A' from rows three and four and store these multiples in the second column below the diagonal of L. Specifically, we subtract 3 times row two from row three and 4 times row two from row four and obtain

$$A'' = \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & 2 & 4 \end{bmatrix}, \qquad L = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 4 & 3 & 1 & \\ 3 & 4 & & 1 \end{bmatrix}.$$

Finally, we subtract row three of A'' from row four and record the multiple 1 in the $(4,3)$-entry of L. This yields the upper triangular matrix U as well as the lower triangular matrix L,

$$U = \begin{bmatrix} 2 & 1 & 1 & 0 \\ & 1 & 1 & 1 \\ & & 2 & 2 \\ & & & 2 \end{bmatrix}, \qquad L = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 4 & 3 & 1 & \\ 3 & 4 & 1 & 1 \end{bmatrix}.$$

One can verify by direct computation that L and U are the desired factors in the LU factorization (3) of A. To summarize, the upper triangular matrix U is determined in the standard way by Gaussian elimination of A, and the lower triangular matrix L is obtained by recording the multiples used in the subtractions in Gaussian elimination. The following algorithm computes the LU factorization of an $n \times n$ matrix A.

Algorithm 9.1: Computation of the LU factorization

INPUT: $n \times n$ matrix $A = [a_{j,k}]_{j,k=1}^{n}$.

OUTPUT: Lower triangular $n \times n$ matrix $L = [\ell_{j,k}]_{j,k=1}^{n}$, upper triangular $n \times n$ matrix $U = [u_{j,k}]_{j,k=1}^{n}$.

Let L be the $n \times n$ identity I. Let $U := A$.

For $k = 1, 2, \ldots, n-1$,

  For $j = k+1, k+2, \ldots, n$,

    $\ell_{j,k} := \frac{u_{j,k}}{u_{k,k}}$.

    For $i = k+1, k+2, \ldots, n$,

      $u_{j,i} := u_{j,i} - \ell_{j,k} u_{k,i}$.

    End i

  End k

End j

Exercise 9.1

Determine the LU factorization of the matrix

$$A = \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix}.$$

□

Exercise 9.2

Solve $A\mathbf{x} = \mathbf{b}$, where A is the matrix in Exercise 9.1 and $\mathbf{b} = [3, 5]^T$, by using the LU factorization from Exercise 9.1. □

Exercise 9.3

Write a MATLAB/Octave function for computing the LU factorization of a general $n \times n$ matrix. The function call should look like

function [L,U]=lufactors(A)

Test the code on a few examples. □

Exercise 9.4

Write a MATLAB/Octave function for the solution of the lower triangular system (4) by forward substitution. The function call should look like

function [y]=forwardsubst(L,b)

Test the code on a few examples. □

Exercise 9.5

Write a MATLAB or Octave function for the solution of the upper triangular system (5) by back substitution. The function call should look like

function [x]=backsubst(U,y)

Test the code on a few examples. □

Exercise 9.6

How many arithmetic floating point operations are required to compute the upper triangular factor U in the LU factorization of an $n \times n$ matrix A? Provide a rough estimate. □

Exercise 9.7

How many arithmetic floating point operations are required for the solution of (4) by forward substitution, when L is an $n \times n$ lower triangular matrix? Provide a rough estimate. □

Exercise 9.8

How many arithmetic floating point operations are required for the solution of (4) by back substitution, when U is an $n \times n$ upper triangular matrix? Provide a rough estimate. □

9.3 LU factorization with pivoting

Consider the solution of a linear system of equations (1) with

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}. \tag{6}$$

The matrix has determinant 1. It therefore is nonsingular and the linear system of equations (1) has a unique solution. In fact, it is easy to verify that the solution is $\mathbf{x} = [2, 3]^T$.

Apply Algorithm 9.1 to determine the LU factorization of the matrix A. You will see that the algorithm breaks down already for $k = 1$, because $u_{1,1} = 0$ causes division by zero. We conclude that LU factorization as described by Algorithm 9.1 cannot be applied to the solution of all linear systems of equations with a nonsingular matrix.

As you already may have noticed, the system of equations (1) can be solved quite easily without LU factorization by applying back substitution in an appropriate order. We first determine the solution component x_2 from the first equation of (1), $1 \cdot x_2 = 2$, and compute x_1 from the second equation, $-1 \cdot x_1 + 1 \cdot x_2 = 1$. Note that your MATLAB/Octave function *backsubst* from Exercise 9.5 cannot be applied to carry out this variant of back substitution.

There is a simple remedy that allows the application of your MATLAB function *backsubst* to the solution of (1), namely to interchange the rows of the matrix and right-hand side before solution. This gives the matrix and right-hand side vector

$$\tilde{A} = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad \tilde{\mathbf{b}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

The matrix $\tilde{A}$ is upper triangular and therefore the linear system of equations

$$\tilde{A}\mathbf{x} = \tilde{\mathbf{b}},$$

can be solved by standard back substitution. We recall that interchanging rows of a linear system of equations does not change the solution.

The interchange of rows is commonly referred to as *pivoting* and the divisors $u_{k,k}$ in Algorithm 9.1 as *pivot elements* or simply *pivots*. Pivoting has to be employed whenever a pivot $u_{k,k}$ in Algorithm 9.1 vanishes. One can show that all linear systems of equations with a square nonsingular matrix can be solved by Gaussian elimination with pivoting, or equivalently, by LU factorization with pivoting. The factor L is not lower triangular when pivoting is employed.

Example 9.1

The function *lu* in MATLAB and Octave determines the LU factorization of a matrix A with pivoting. When applied to the matrix (6), it produces

$$L = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad U = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Thus, L is not lower triangular. The matrix L can be thought of as a lower triangular matrix with the rows interchanged. More details on the function *lu* are provided in Exercise 9.9. □

In exact arithmetic, we can compute the LU factorization of any nonsingular 2×2 matrix with a nonvanishing $(1, 1)$ element. However, in floating point arithmetic, the computed factors might not be accurate.

Example 9.2

Apply a MATLAB or Octave implementation of Algorithm 9.1 on a standard PC to the matrix

$$A = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}.$$

We obtain the factors

$$L = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix},$$

where the entry $u_{2,2}$ of U is computed as $1 - 10^{20} \cdot 1$, which is stored as -10^{20}.

The relative error in $u_{2,2}$ is

$$\frac{-10^{20} - (1 - 10^{20})}{1 - 10^{20}} \approx 10^{-20},$$

which is tiny; however, the absolute error in $u_{2,2}$, given by $-10^{20} - (1 - 10^{20}) = -1$, is not. Multiplying L and U yields

$$LU = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 0 \end{bmatrix},$$

which is not close to the matrix A. The error in the matrix LU is caused by round-off errors during the computations of $u_{2,2}$ and by the fact that there are intermediate quantities of very large magnitude formed during the computations. The difficulties are caused by the quantity -10^{20}. □

Example 9.3

A row interchange in the matrix of the above example remedies the accuracy problems encountered. Let

$$\tilde{A} = \begin{bmatrix} 1 & 1 \\ 10^{-20} & 1 \end{bmatrix}.$$

Then a MATLAB or Octave implementation of Algorithm 9.1 determines the LU factorization

$$\tilde{L} = \begin{bmatrix} 1 & 0 \\ 10^{-20} & 1 \end{bmatrix}, \qquad \tilde{U} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Multiplication of $\tilde{L}$ and $\tilde{U}$ gives the matrix $\tilde{A}$ in floating point arithmetic. The high accuracy depends on that the matrices $\tilde{L}$ and $\tilde{U}$ do not contain entries of large magnitude. □

The above example illustrates that row interchange can improve the accuracy significantly in the computed LU factors, also when in exact arithmetic no row interchanges are required. The row interchange gave a lower triangular factor $\tilde{L}$ with entries of significantly smaller magnitude than the magnitude of the entries of the lower triangular matrix L of Example 9.2. Looking at Algorithm 9.1, we see that the magnitude of all subdiagonal entries $\ell_{j,k}$ of L is bounded by one, if $|u_{k,k}|$ is larger than or equal to

$$\max\{|u_{k+1,k}|, |u_{k+2,k}|, \ldots, |u_{n,k}|\}$$

for $k = 1, 2, \ldots, n-1$. This suggests that we interchange the rows of the U-matrix during the factorization process to achieve that

$$\frac{|u_{j,k}|}{|u_{k,k}|} \leq 1, \qquad j = k+1, k+2, \ldots, n. \tag{7}$$

LU factorization with the rows (re)ordered so that (7) holds is commonly referred as *LU factorization with partial pivoting*. There are also other pivoting strategies. We will comment on some of them in Exercises 9.13 and 9.14.

When we solve a linear system of equations and interchange the rows of the matrix, we also need to interchange the corresponding rows of the right-hand side in order to obtain the correct solution. LU

factorization with partial pivoting may be carried out without access to the right-hand side. We have to keep track of the row interchanges carried out during the factorization, so that we can apply them to the right-hand side when the latter is available. We do this by applying all the row interchanges carried out during the LU factorization to the identity matrix. This gives a *permutation matrix* P. Permutation matrices are matrices obtained be interchanging rows or columns in the identity matrix. They have precisely one nonvanishing entry (one) in each row and column. A permutation matrix is an example of a matrix, whose transpose is its inverse. Thus, a permutation matrix is an orthogonal matrix.

Example 9.4

Let

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Then its transpose is given by

$$P^T = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

and it is easy to verify that $P^T P = P P^T = I$. □

In order to illustrate LU factorization with partial pivoting, we apply the method to the matrix

$$A = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{bmatrix},$$

which we factored in Section 9.2 without partial pivoting pivoting. We denote the 4×4 permutation matrix, which keeps track of the row interchanges by P; it is initialized as the identity matrix and so is the lower triangular matrix L in the factorization. We set $U = A$. The first step of factorization process is to determine the entry of largest magnitude in column 1. This is the entry 8 in row 3. We therefore swap rows 1 and 3 of the matrices U and P to obtain,

$$U = \begin{bmatrix} 8 & 7 & 9 & 5 \\ 4 & 3 & 3 & 1 \\ 2 & 1 & 1 & 0 \\ 6 & 7 & 9 & 8 \end{bmatrix}, \qquad P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We then subtract suitable multiples of row 1 of U from rows 2 through 4, to create zeros in the first column of U below the diagonal. The multiples are stored in the subdiagonal entries of the first column of the matrix L. This gives the matrices

$$U = \begin{bmatrix} 8 & 7 & 9 & 5 \\ 0 & -1/2 & -3/2 & -3/2 \\ 0 & -3/4 & -5/4 & -5/4 \\ 0 & 7/4 & 9/4 & 1/4 \end{bmatrix}, \qquad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 1 & 0 & 0 \\ 1/4 & 0 & 1 & 0 \\ 3/4 & 0 & 0 & 1 \end{bmatrix}, \qquad P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We now proceed to the entries 2 through 4 of column 2 of U, and note that the last entry is of largest magnitude. Hence, we interchange rows 2 and 4 of the matrices U and P. We also need to swap the subdiagonal entries of rows 2 and 4 of L. This gives

$$U = \begin{bmatrix} 8 & 7 & 9 & 5 \\ 0 & 7/4 & 9/4 & 1/4 \\ 0 & -3/4 & -5/4 & -5/4 \\ 0 & -1/2 & -3/2 & -3/2 \end{bmatrix}, \quad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3/4 & 1 & 0 & 0 \\ 1/4 & 0 & 1 & 0 \\ 1/2 & 0 & 0 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

We subtract suitable multiples of row 2 from rows 3 and 4 to obtain zeros in the second column of U below the diagonal. The multiples are stored in the subdiagonal entries of the second column of the matrix L. This yields the matrices

$$U = \begin{bmatrix} 8 & 7 & 9 & 5 \\ 0 & 7/4 & 9/4 & 1/4 \\ 0 & 0 & -2/7 & 4/7 \\ 0 & 0 & -6/7 & -2/7 \end{bmatrix}, \quad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3/4 & 1 & 0 & 0 \\ 1/4 & -3/7 & 1 & 0 \\ 1/2 & -2/7 & 0 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

We turn to the last 2 entries of column 3 of U, and notice that the last entry has the largest magnitude. Hence, we swap rows 3 and 4 of the matrices U and P, and we interchange the subdiagonal entries in rows 3 and 4 of L to obtain

$$U = \begin{bmatrix} 8 & 7 & 9 & 5 \\ 0 & 7/4 & 9/4 & 1/4 \\ 0 & 0 & -6/7 & -2/7 \\ 0 & 0 & -2/7 & 4/7 \end{bmatrix}, \quad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3/4 & 1 & 0 & 0 \\ 1/2 & -2/7 & 1 & 0 \\ 1/4 & -3/7 & 0 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

The final step of the factorization is to subtract $1/3$ times row 3 of U from row 4 and store $1/3$ in the subdiagonal entry of column 3 of L. This yields

$$U = \begin{bmatrix} 8 & 7 & 9 & 5 \\ 0 & 7/4 & 9/4 & 1/4 \\ 0 & 0 & -6/7 & -2/7 \\ 0 & 0 & 0 & 2/3 \end{bmatrix}, \quad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3/4 & 1 & 0 & 0 \\ 1/2 & -2/7 & 1 & 0 \\ 1/4 & -3/7 & 1/3 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

and we can verify that

$$PA = LU. \tag{8}$$

When applying this factorization to the solution of a linear system of equations (1), we first multiply the right-hand side $\mathbf{b}$ by the permutation matrix P to obtain

$$LU\mathbf{x} = PA\mathbf{x} = P\mathbf{b}.$$

We then solve $L\mathbf{y} = P\mathbf{b}$ by forward substitution and $U\mathbf{x} = \mathbf{y}$ by back substitution.

Assume that we apply LU factorization with partial pivoting to an $n \times n$ matrix A and find that for some k all entries $u_{j,k}$, $j = k, k+1, \ldots, n$ of U vanish. Then pivoting does not help us to proceed and LU factorization with partial pivoting breaks down. One can show that this situation only can occur when A is singular. Thus, LU factorization with partial pivoting can be applied to solve all linear systems of equations with a nonsingular matrix.

Partial pivoting secures that the entries of L are all bounded by one in magnitude. However, the entries of U may become large; see Exercise 9.15 for an illustration. Let ρ denote the largest magnitude of any of the entries $u_{i,j}$ generated during the execution of a variant of Algorithm 9.1 with partial pivoting. This quantity is referred to as the growth factor. The accuracy of the computed lower and upper triangular factors $\mathrm{fl}(L)$ and $\mathrm{fl}(U)$, respectively, of PA depend on ρ. One can show that

$$\frac{\|\mathrm{fl}(L)\mathrm{fl}(U) - A\|}{\|A\|} = \rho\mathcal{O}(\mathsf{eps}).$$

Thus, a large growth factor ρ may destroy the accuracy of the LU factorization. However, for many matrices ρ is modest and LU factorization with partial pivoting yields an accurate factorization in the presence of round-off errors. LU factorization without partial pivoting may yield poor accuracy.

Exercise 9.9

(a) Apply the MATLAB/Octave function *lu* to the matrix (6) by using the call $[L, U] = lu(A)$. What is L? Read the help file for a description of L. (b) Use instead the function call $[L, U, P] = lu(A)$. What are L, U, and P? □

Exercise 9.10

Determine the LU factorization with partial pivoting of the matrix

$$A = \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix}$$

by hand computations. □

Exercise 9.11

Solve $A\mathbf{x} = \mathbf{b}$, where A is the matrix in Exercise 9.10 and $\mathbf{b} = [3, 5]^T$, by using the LU factorization from Exercise 9.10 □

Exercise 9.12

The storage of the permutation matrix P is clumsy. After all, we only need to keep track of the row interchanges carried out and this does not require storage of an $n \times n$ matrix with most entries zero. Describe a more compact way to represent the information. □

Exercise 9.13

If an entry $u_{k,k}$ in Algorithm 9.1 vanishes, then we could interchange columns instead of rows to obtain a nonvanishing replacement for $u_{k,k}$. Mention a difficulty with this approach that is not present when interchanging rows. Hint: How does the solution change when we interchange rows and columns? □

Exercise 9.14

In LU factorization with complete pivoting rows as well as columns are interchanged so as to maximize the denominator $u_{k,k}$ in each step. Discuss possible advantages and disadvantages of complete pivoting? □

Exercise 9.15

Determine the growth factor for LU factorization with partial pivoting of the almost lower triangular $n \times n$ matrix

$$A = \begin{bmatrix} 1 & & & & & 1 \\ -1 & 1 & & & & 1 \\ -1 & -1 & 1 & & & 1 \\ -1 & -1 & -1 & 1 & & 1 \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ -1 & -1 & -1 & \cdots & & 1 \end{bmatrix}.$$

□