in running time—the total computation time is 1.003 seconds, almost all of which would be taken by the elimination step. The next example shows a third point. While the back-substitution time may sometimes be negligible, it may factor into an important calculation.

► **EXAMPLE 2.3** On a particular computer, back substitution of a 5000 × 5000 triangular matrix takes 0.1 seconds. Estimate the time needed to solve a general system of 3000 equations in 3000 unknowns by Gaussian elimination.

The computer can carry out $(5000)^2$ operations in 0.1 seconds, or $(5000)^2(10) = 2.5 \times 10^8$ operations/second. Solving a general (nontriangular) system requires about $2(3000)^3/3$ operations, which can be done in approximately

$$\frac{2(3000)^3/3}{(5000)^2(10)} \approx 72 \text{ sec.}$$

◄

2.1 Exercises

1. Use Gaussian elimination to solve the systems:

$$\text{(a)} \begin{aligned} 2x - 3y &= 2 \\ 5x - 6y &= 8 \end{aligned} \qquad \text{(b)} \begin{aligned} x + 2y &= -1 \\ 2x + 3y &= 1 \end{aligned} \qquad \text{(c)} \begin{aligned} -x + y &= 2 \\ 3x + 4y &= 15 \end{aligned}$$

2. Use Gaussian elimination to solve the systems:

$$\text{(a)} \begin{aligned} 2x - 2y - z &= -2 \\ 4x + y - 2z &= 1 \\ -2x + y - z &= -3 \end{aligned} \qquad \text{(b)} \begin{aligned} x + 2y - z &= 2 \\ 3y + z &= 4 \\ 2x - y + z &= 2 \end{aligned} \qquad \text{(c)} \begin{aligned} 2x + y - 4z &= -7 \\ x - y + z &= -2 \\ -x + 3y - 2z &= 6 \end{aligned}$$

3. Solve by back substitution:

$$\text{(a)} \begin{aligned} 3x - 4y + 5z &= 2 \\ 3y - 4z &= -1 \\ 5z &= 5 \end{aligned} \qquad \text{(b)} \begin{aligned} x - 2y + z &= 2 \\ 4y - 3z &= 1 \\ -3z &= 3 \end{aligned}$$

4. Solve the tableau form

$$\text{(a)} \left[\begin{array}{ccc|c} 3 & -4 & -2 & 3 \\ 6 & -6 & 1 & 2 \\ -3 & 8 & 2 & -1 \end{array}\right] \qquad \text{(b)} \left[\begin{array}{ccc|c} 2 & 1 & -1 & 2 \\ 6 & 2 & -2 & 8 \\ 4 & 6 & -3 & 5 \end{array}\right]$$

5. Use the approximate operation count $2n^3/3$ for Gaussian elimination to estimate how much longer it takes to solve n equations in n unknowns if n is tripled.

6. Assume that your computer completes a 5000 equation back substitution in 0.005 seconds. Use the approximate operation counts n^2 for back substitution and $2n^3/3$ for elimination to estimate how long it will take to do a complete Gaussian elimination of this size. Round your answer to the nearest second.

7. Assume that a given computer requires 0.002 seconds to complete back substitution on a 4000 × 4000 upper triangular matrix equation. Estimate the time needed to solve a general system of 9000 equations in 9000 unknowns. Round your answer to the nearest second.

8. If a system of 3000 equations in 3000 unknowns can be solved by Gaussian elimination in 5 seconds on a given computer, how many back substitutions of the same size can be done per second?

2.1 Computer Problems

1. Put together the code fragments in this section to create a MATLAB program for "naive" Gaussian elimination (meaning no row exchanges allowed). Use it to solve the systems of Exercise 2.

2. Let H denote the $n \times n$ Hilbert matrix, whose (i, j) entry is $1/(i + j - 1)$. Use the MATLAB program from Computer Problem 1 to solve $Hx = b$, where b is the vector of all ones, for (a) $n = 2$ (b) $n = 5$ (c) $n = 10$.

2.2 THE LU FACTORIZATION

Carrying the idea of tableau form one step farther brings us to the matrix form of a system of equations. Matrix form will save time in the long run by simplifying the algorithms and their analysis.

2.2.1 Matrix form of Gaussian elimination

The system (2.1) can be written as $Ax = b$ in matrix form, or

$$\begin{bmatrix} 1 & 1 \\ 3 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}. \tag{2.10}$$

We will usually denote the **coefficient matrix** by A and the **right-hand-side** vector as b. In the matrix form of the systems of equations, we interpret x as a column vector and Ax as matrix-vector multiplication. We want to find x such that the vector Ax is equal to the vector b. Of course, this is equivalent to having Ax and b agree in all components, which is exactly what is required by the original system (2.1).

The advantage of writing systems of equations in matrix form is that we can use matrix operations, like matrix multiplication, to keep track of the steps of Gaussian elimination. The LU factorization is a matrix representation of Gaussian elimination. It consists of writing the coefficient matrix A as a product of a lower triangular matrix L and an upper triangular matrix U. The LU factorization is the Gaussian elimination version of a long tradition in science and engineering—breaking down a complicated object into simpler parts.

DEFINITION 2.2 An $m \times n$ matrix L is **lower triangular** if its entries satisfy $l_{ij} = 0$ for $i < j$. An $m \times n$ matrix U is **upper triangular** if its entries satisfy $u_{ij} = 0$ for $i > j$. ❐

► **EXAMPLE 2.4** Find the LU factorization for the matrix A in (2.10).

The elimination steps are the same as for the tableau form seen earlier:

$$\begin{bmatrix} 1 & 1 \\ 3 & -4 \end{bmatrix} \xrightarrow{\text{subtract } 3 \times \text{ row } 1 \text{ from row } 2} \begin{bmatrix} 1 & 1 \\ 0 & -7 \end{bmatrix} = U. \tag{2.11}$$

The difference is that now we store the multiplier 3 used in the elimination step. Note that we have defined U to be the upper triangular matrix showing the result of Gaussian elimination. Define L to be the 2×2 lower triangular matrix with 1's on the main diagonal and the multiplier 3 in the (2,1) location:

$$\begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}.$$

an extra back substitution that was not part of classical Gaussian elimination, these "extra" calculations exactly replace the calculations that were saved during elimination because the right-hand-side b was absent.

If all b_i were available at the outset, we could solve all k problems simultaneously in the same number of operations. But in typical applications, we are asked to solve some of the $Ax = b_i$ problems before other b_i's are available. The LU approach allows efficient handling of all present and future problems that involve the same coefficient matrix A.

► EXAMPLE 2.8 Assume that it takes one second to factorize the 300×300 matrix A into $A = LU$. How many problems $Ax = b_1, \ldots, Ax = b_k$ can be solved in the next second?

The two back substitutions for each b_i require a total of $2n^2$ operations. Therefore, the approximate number of b_i that can be handled per second is

$$\frac{\frac{2n^3}{3}}{2n^2} = \frac{n}{3} = 100. \quad \blacktriangleleft$$

The LU factorization is a significant step forward in our quest to run Gaussian elimination efficiently. Unfortunately, not every matrix allows such a factorization.

► EXAMPLE 2.9 Prove that $A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ does not have an LU factorization.

The factorization must have the form

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix} \begin{bmatrix} b & c \\ 0 & d \end{bmatrix} = \begin{bmatrix} b & c \\ ab & ac+d \end{bmatrix}.$$

Equating coefficients yields $b = 0$ and $ab = 1$, a contradiction. ◄

The fact that not all matrices have an LU factorization means that more work is required before we can declare the LU factorization a general Gaussian elimination algorithm. The related problem of swamping is described in the next section. In Section 2.4, the PA = LU factorization is introduced, which will overcome both problems.

2.2 Exercises

1. Find the LU factorization of the given matrices. Check by matrix multiplication.

   (a) $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ (b) $\begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}$ (c) $\begin{bmatrix} 3 & -4 \\ -5 & 2 \end{bmatrix}$

2. Find the LU factorization of the given matrices. Check by matrix multiplication.

   (a) $\begin{bmatrix} 3 & 1 & 2 \\ 6 & 3 & 4 \\ 3 & 1 & 5 \end{bmatrix}$ (b) $\begin{bmatrix} 4 & 2 & 0 \\ 4 & 4 & 2 \\ 2 & 2 & 3 \end{bmatrix}$ (c) $\begin{bmatrix} 1 & -1 & 1 & 2 \\ 0 & 2 & 1 & 0 \\ 1 & 3 & 4 & 4 \\ 0 & 2 & 1 & -1 \end{bmatrix}$

3. Solve the system by finding the LU factorization and then carrying out the two-step back substitution.

   (a) $\begin{bmatrix} 3 & 7 \\ 6 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -11 \end{bmatrix}$ (b) $\begin{bmatrix} 2 & 3 \\ 4 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$

4. Solve the system by finding the LU factorization and then carrying out the two-step back substitution.

   (a) $\begin{bmatrix} 3 & 1 & 2 \\ 6 & 3 & 4 \\ 3 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix}$ (b) $\begin{bmatrix} 4 & 2 & 0 \\ 4 & 4 & 2 \\ 2 & 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$

5. Solve the equation $Ax = b$, where

   $$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 3 & 1 & 0 \\ 4 & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } b = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 0 \end{bmatrix}.$$

6. Given the 1000×1000 matrix A, your computer can solve the 500 problems $Ax = b_1, \ldots, Ax = b_{500}$ in exactly one minute, using $A = LU$ factorization methods. How much of the minute was the computer working on the $A = LU$ factorization? Round your answer to the nearest second.

7. Assume that your computer can solve 1000 problems of type $Ux = c$, where U is an upper-triangular 500×500 matrix, per second. Estimate how long it will take to solve a full 5000×5000 matrix problem $Ax = b$. Answer in minutes and seconds.

8. Assume that your computer can solve a 2000×2000 linear system $Ax = b$ in 0.1 second. Estimate the time required to solve 100 systems of 8000 equations in 8000 unknowns with the same coefficient matrix, using the LU factorization method.

9. Let A be an $n \times n$ matrix. Assume that your computer can solve 100 problems $Ax = b_1, \ldots, Ax = b_{100}$ by the LU method in the same amount of time it takes to solve the first problem $Ax = b_0$. Estimate n.

2.2 Computer Problems

1. Use the code fragments for Gaussian elimination in the previous section to write a MATLAB script to take a matrix A as input and output L and U. No row exchanges are allowed—the program should be designed to shut down if it encounters a zero pivot. Check your program by factoring the matrices in Exercise 2.

2. Add two-step back substitution to your script from Computer Problem 1, and use it to solve the systems in Exercise 4.

2.3 SOURCES OF ERROR

There are two major potential sources of error in Gaussian elimination as we have described it so far. The concept of ill-conditioning concerns the sensitivity of the solution to the input data. We will discuss condition number, using the concepts of backward and forward error from Chapter 1. Very little can be done to avoid errors in computing the solution of ill-conditioned matrix equations, so it is important to try to recognize and avoid ill-conditioned matrices when possible. The second source of error is swamping, which can be avoided in the large majority of problems by a simple fix called partial pivoting, the subject of Section 2.4.

1. **Exact solution.** In tableau form, Gaussian elimination proceeds as

$$\left[\begin{array}{cc|c} 10^{-20} & 1 & 1 \\ 1 & 2 & 4 \end{array}\right] \longrightarrow \begin{array}{c}\text{subtract } 10^{20}\times \text{ row 1} \\ \text{from row 2}\end{array} \longrightarrow \left[\begin{array}{cc|c} 10^{-20} & 1 & 1 \\ 0 & 2-10^{20} & 4-10^{20} \end{array}\right].$$

The bottom equation is

$$(2-10^{20})x_2 = 4-10^{20} \longrightarrow x_2 = \frac{4-10^{20}}{2-10^{20}},$$

and the top equation yields

$$\begin{aligned} 10^{-20}x_1 + \frac{4-10^{20}}{2-10^{20}} &= 1 \\ x_1 = 10^{20}\left(1 - \frac{4-10^{20}}{2-10^{20}}\right) & \\ x_1 = \frac{-2\times 10^{20}}{2-10^{20}}. & \end{aligned}$$

The exact solution is

$$[x_1, x_2] = \left[\frac{2\times 10^{20}}{10^{20}-2}, \frac{4-10^{20}}{2-10^{20}}\right] \approx [2, 1].$$

2. **IEEE double precision.** The computer version of Gaussian elimination proceeds slightly differently:

$$\left[\begin{array}{cc|c} 10^{-20} & 1 & 1 \\ 1 & 2 & 4 \end{array}\right] \longrightarrow \begin{array}{c}\text{subtract } 10^{20}\times \text{ row 1} \\ \text{from row 2}\end{array} \longrightarrow \left[\begin{array}{cc|c} 10^{-20} & 1 & 1 \\ 0 & 2-10^{20} & 4-10^{20} \end{array}\right].$$

In IEEE double precision, $2 - 10^{20}$ is the same as -10^{20}, due to rounding. Similarly, $4 - 10^{20}$ is stored as -10^{20}. Now the bottom equation is

$$-10^{20}x_2 = -10^{20} \longrightarrow x_2 = 1.$$

The machine arithmetic version of the top equation becomes

$$10^{-20}x_1 + 1 = 1,$$

so $x_1 = 0$. The computed solution is exactly

$$[x_1, x_2] = [0, 1].$$

This solution has large relative error compared with the exact solution.

3. **IEEE double precision, after row exchange.** We repeat the computer version of Gaussian elimination, after changing the order of the two equations:

$$\left[\begin{array}{cc|c} 1 & 2 & 4 \\ 10^{-20} & 1 & 1 \end{array}\right] \longrightarrow \begin{array}{c}\text{subtract } 10^{-20}\times \text{ row 1} \\ \text{from row 2}\end{array} \longrightarrow \left[\begin{array}{cc|c} 1 & 2 & 4 \\ 0 & 1-2\times 10^{-20} & 1-4\times 10^{-20} \end{array}\right].$$

In IEEE double precision, $1 - 2 \times 10^{-20}$ is stored as 1 and $1 - 4 \times 10^{-20}$ is stored as 1. The equations are now

$$\begin{aligned} x_1 + 2x_2 &= 4 \\ x_2 &= 1, \end{aligned}$$

which yield the computed solution $x_1 = 2$ and $x_2 = 1$. Of course, this is not the exact answer, but it is correct up to approximately 16 digits, which is the most we can ask from a computation that uses 52-bit floating point numbers.

The difference between the last two calculations is significant. Version 3 gave us an acceptable solution, while version 2 did not. An analysis of what went wrong with version 2 leads to considering the multiplier 10^{20} that was used for the elimination step. The effect of subtracting 10^{20} times the top equation from the bottom equation was to overpower, or "swamp," the bottom equation. While there were originally two independent equations, or sources of information, after the elimination step in version 2, there are essentially two copies of the top equation. Since the bottom equation has disappeared, for all practical purposes, we cannot expect the computed solution to satisfy the bottom equation; and it does not.

Version 3, on the other hand, completes elimination without swamping, because the multiplier is 10^{-20}. After elimination, the original two equations are still largely existent, slightly changed into triangular form. The result is an approximate solution that is much more accurate. ◀

The moral of Example 2.13 is that multipliers in Gaussian elimination should be kept as small as possible to avoid swamping. Fortunately, there is a simple modification to naive Gaussian elimination that forces the absolute value of multipliers to be no larger than 1. This new protocol, which involves judicious row exchanges in the tableau, is called partial pivoting, the topic of the next section.

2.3 Exercises

1. Find the norm $||A||_\infty$ of each of the following matrices:

$$\text{(a)}\quad A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \text{(b)}\quad A = \begin{bmatrix} 1 & 5 & 1 \\ -1 & 2 & -3 \\ 1 & -7 & 0 \end{bmatrix}.$$

2. Find the (infinity norm) condition number of

$$\text{(a)}\quad A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \text{(b)}\quad A = \begin{bmatrix} 1 & 2.01 \\ 3 & 6 \end{bmatrix} \qquad \text{(c)}\quad A = \begin{bmatrix} 6 & 3 \\ 4 & 2 \end{bmatrix}.$$

3. Find the forward and backward errors, and the error magnification factor (in the infinity norm) for the following approximate solutions x_a of the system in Example 2.11: (a) $[-1, 3]$ (b) $[0, 2]$ (c) $[2, 2]$ (d) $[-2, 4]$ (e) $[-2, 4.0001]$.

4. Find the forward and backward errors and error magnification factor for the following approximate solutions of the system $x_1 + 2x_2 = 1, 2x_1 + 4.01x_2 = 2$: (a) $[-1, 1]$ (b) $[3, -1]$ (c) $[2, -1/2]$.

5. Find the relative forward and backward errors and error magnification factor for the following approximate solutions of the system $x_1 - 2x_2 = 3, 3x_1 - 4x_2 = 7$: (a) $[-2, -4]$ (b) $[-2, -3]$ (c) $[0, -2]$ (d) $[-1, -1]$ (e) What is the condition number of the coefficient matrix?

6. Find the relative forward and backward errors and error magnification factor for the following approximate solutions of the system $x_1 + 2x_2 = 3, 2x_1 + 4.01x_2 = 6.01$: (a) $[-10, 6]$ (b) $[-100, 52]$ (c) $[-600, 301]$ (d) $[-599, 301]$ (e) What is the condition number of the coefficient matrix?

7. Find the norm $||H||_\infty$ of the 5×5 Hilbert matrix.

8. (a) Find the condition number of the coefficient matrix in the system $\begin{bmatrix} 1 & 1 \\ 1+\delta & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2+\delta \end{bmatrix}$ as a function of $\delta > 0$. (b) Find the error magnification factor for the approximate root $x_a = [-1, 3+\delta]$.

9. (a) Prove that the infinity norm $||x||_\infty$ is a vector norm. (b) Prove that the 1-norm $||x||_1$ is a vector norm.

10. (a) Prove that the infinity norm $||A||_\infty$ is a matrix norm. (b) Prove that the 1-norm $||A||_1$ is a matrix norm.

11. Prove that the matrix infinity norm is the operator norm of the vector infinity norm.

12. Prove that the matrix 1-norm is the operator norm of the vector 1-norm.

13. For the matrices in Exercise 1, find a vector x satisfying $||A||_\infty = ||Ax||_\infty / ||x||_\infty$.

14. For the matrices in Exercise 1, find a vector x satisfying $||A||_1 = ||Ax||_1 / ||x||_1$.

15. Find the LU factorization of

$$A = \begin{bmatrix} 10 & 20 & 1 \\ 1 & 1.99 & 6 \\ 0 & 50 & 1 \end{bmatrix}.$$

What is the largest magnitude multiplier l_{ij} needed?

2.3 Computer Problems

1. For the $n \times n$ matrix with entries $A_{ij} = 5/(i + 2j - 1)$, set $x = [1, \ldots, 1]^T$ and $b = Ax$. Use the MATLAB program from Computer Problem 2.1.1 or MATLAB's backslash command to compute x_c, the double precision computed solution. Find the infinity norm of the forward error and the error magnification factor of the problem $Ax = b$, and compare it with the condition number of A: (a) $n = 6$ (b) $n = 10$.

2. Carry out Computer Problem 1 for the matrix with entries $A_{ij} = 1/(|i - j| + 1)$.

3. Let A be the $n \times n$ matrix with entries $A_{ij} = |i - j| + 1$. Define $x = [1, \ldots, 1]^T$ and $b = Ax$. For $n = 100, 200, 300, 400$, and 500, use the MATLAB program from Computer Problem 2.1.1 or MATLAB's backslash command to compute x_c, the double precision computed solution. Calculate the infinity norm of the forward error for each solution. Find the five error magnification factors of the problems $Ax = b$, and compare with the corresponding condition numbers.

4. Carry out the steps of Computer Problem 3 for the matrix with entries $A_{ij} = \sqrt{(i-j)^2 + n/10}$.

5. For what values of n does the solution in Computer Problem 1 have no correct significant digits?

6. Use the MATLAB program from Computer Problem 2.1.1 to carry out double precision implementations of versions 2 and 3 of Example 2.13, and compare with the theoretical results found in the text.

2.4 THE PA = LU FACTORIZATION

The form of Gaussian elimination considered so far is often called "naive," because of two serious difficulties: encountering a zero pivot and swamping. For a nonsingular matrix, both can be avoided with an improved algorithm. The key to this improvement is an efficient protocol for exchanging rows of the coefficient matrix, called partial pivoting.

2.4.1 Partial pivoting

At the start of classical Gaussian elimination of n equations in n unknowns, the first step is to use the diagonal element a_{11} as a pivot to eliminate the first column. The **partial pivoting** protocol consists of comparing numbers before carrying out each elimination step. The largest entry of the first column is located, and its row is swapped with the pivot row, in this case the top row.

In other words, at the start of Gaussian elimination, partial pivoting asks that we select the pth row, where

$$|a_{p1}| \geq |a_{i1}| \tag{2.21}$$

for all $1 \leq i \leq n$, and exchange rows 1 and p. Next, elimination of column 1 proceeds as usual, using the "new" version of a_{11} as the pivot. The multiplier used to eliminate a_{i1} will be

$$m_{i1} = \frac{a_{i1}}{a_{11}}$$

and $|m_{i1}| \leq 1$.

The same check is applied to every choice of pivot during the algorithm. When deciding on the second pivot, we start with the current a_{22} and check all entries directly below. We select the row p such that

$$|a_{p2}| \geq |a_{i2}|$$

for all $2 \leq i \leq n$, and if $p \neq 2$, rows 2 and p are exchanged. Row 1 is never involved in this step. If $|a_{22}|$ is already the largest, no row exchange is made.

The protocol applies to each column during elimination. Before eliminating column k, the p with $k \leq p \leq n$ and largest $|a_{pk}|$ is located, and rows p and k are exchanged if necessary before continuing with the elimination. Note that using partial pivoting ensures that all multipliers, or entries of L, will be no greater than 1 in absolute value. With this minor change in the implementation of Gaussian elimination, the problem of swamping illustrated in Example 2.13 is completely avoided.

► EXAMPLE 2.14 Apply Gaussian elimination with partial pivoting to solve the system (2.1).

The equations can be written in tableau form as

$$\left[\begin{array}{cc|c} 1 & 1 & 3 \\ 3 & -4 & 2 \end{array}\right].$$

According to partial pivoting, we compare $|a_{11}| = 1$ with all entries below it, in this case the single entry $a_{21} = 3$. Since $|a_{21}| > |a_{11}|$, we must exchange rows 1 and 2. The new tableau is

$$\left[\begin{array}{cc|c} 3 & -4 & 2 \\ 1 & 1 & 3 \end{array}\right] \xrightarrow{\substack{\text{subtract } \frac{1}{3} \times \text{ row 1} \\ \text{from row 2}}} \left[\begin{array}{cc|c} 3 & -4 & 2 \\ 0 & \frac{7}{3} & \frac{7}{3} \end{array}\right].$$

Starting at the bottom,

$$\begin{aligned} 8x_3 = 8 &\Rightarrow x_3 = 1 \\ 2x_2 + 2(1) = 6 &\Rightarrow x_2 = 2 \\ 4x_1 + 4(2) - 4(1) = 0 &\Rightarrow x_1 = -1. \end{aligned} \tag{2.25}$$

Therefore, the solution is $x = [-1, 2, 1]$. ◄

► **EXAMPLE 2.18** Solve the system $2x_1 + 3x_2 = 4, 3x_1 + 2x_2 = 1$ using the PA = LU factorization with partial pivoting.

In matrix form, this is the equation

$$\begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}.$$

We begin by ignoring the right-hand-side b. According to partial pivoting, rows 1 and 2 must be exchanged (because $a_{21} > a_{11}$). The elimination step is

$$A = \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix} \xrightarrow{P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}} \xrightarrow{\text{exchange rows 1 and 2}} \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}$$

$$\xrightarrow{\text{subtract } \frac{2}{3} \times \text{ row 1 from row 2}} \begin{bmatrix} 3 & 2 \\ \textcircled{\frac{2}{3}} & \frac{5}{3} \end{bmatrix}.$$

Therefore, the PA = LU factorization is

$$\underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 \\ \frac{2}{3} & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 3 & 2 \\ 0 & \frac{5}{3} \end{bmatrix}}_{U}.$$

The first back substitution $Lc = Pb$ is

$$\begin{bmatrix} 1 & 0 \\ \frac{2}{3} & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}.$$

Starting at the top, we have

$$\begin{aligned} c_1 &= 1 \\ \frac{2}{3}(1) + c_2 = 4 &\Rightarrow c_2 = \frac{10}{3}. \end{aligned}$$

The second back substitution $Ux = c$ is

$$\begin{bmatrix} 3 & 2 \\ 0 & \frac{5}{3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{10}{3} \end{bmatrix}.$$

Starting at the bottom, we have

$$\begin{aligned} \frac{5}{3}x_2 = \frac{10}{3} &\Rightarrow x_2 = 2 \\ 3x_1 + 2(2) = 1 &\Rightarrow x_1 = -1. \end{aligned} \tag{2.26}$$

Therefore, the solution is $x = [-1, 2]$. ◄

Every $n \times n$ matrix has a PA = LU factorization. We simply follow the partial pivoting rule, and if the resulting pivot is zero, it means that all entries that need to be eliminated are already zero, so the column is done.

All of the techniques described so far are implemented in MATLAB. The most sophisticated form of Gaussian elimination we have discussed is the PA = LU factorization. MATLAB's `lu` command accepts a square coefficient matrix A and returns P, L, and U. The following MATLAB script defines the matrix of Example 2.16 and computes its factorization:

```
>> A=[2 1 5; 4 4 -4; 1 3 1];
>> [L,U,P]=lu(A)

L=

    1.0000         0         0
    0.2500    1.0000         0
    0.5000   -0.5000    1.0000

U=

     4     4    -4
     0     2     2
     0     0     8

P=

     0     1     0
     0     0     1
     1     0     0
```

2.4 Exercises

1. Find the PA = LU factorization (using partial pivoting) of the following matrices:

   (a) $\begin{bmatrix} 1 & 3 \\ 2 & 3 \end{bmatrix}$ (b) $\begin{bmatrix} 2 & 4 \\ 1 & 3 \end{bmatrix}$ (c) $\begin{bmatrix} 1 & 5 \\ 5 & 12 \end{bmatrix}$ (d) $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

2. Find the PA = LU factorization (using partial pivoting) of the following matrices:

   (a) $\begin{bmatrix} 1 & 1 & 0 \\ 2 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix}$ (b) $\begin{bmatrix} 0 & 1 & 3 \\ 2 & 1 & 1 \\ -1 & -1 & 2 \end{bmatrix}$ (c) $\begin{bmatrix} 1 & 2 & -3 \\ 2 & 4 & 2 \\ -1 & 0 & 3 \end{bmatrix}$ (d) $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ -2 & 1 & 0 \end{bmatrix}$

3. Solve the system by finding the PA = LU factorization and then carrying out the two-step back substitution.

   (a) $\begin{bmatrix} 3 & 7 \\ 6 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -11 \end{bmatrix}$ (b) $\begin{bmatrix} 3 & 1 & 2 \\ 6 & 3 & 4 \\ 3 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix}$

4. Solve the system by finding the PA = LU factorization and then carrying out the two-step back substitution.

   (a) $\begin{bmatrix} 4 & 2 & 0 \\ 4 & 4 & 2 \\ 2 & 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$ (b) $\begin{bmatrix} -1 & 0 & 1 \\ 2 & 1 & 1 \\ -1 & 2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -2 \\ 17 \\ 3 \end{bmatrix}$

5. Write down a 5×5 matrix P such that multiplication of another matrix by P on the left causes rows 2 and 5 to be exchanged.

6. (a) Write down the 4×4 matrix P such that multiplying a matrix on the left by P causes the second and fourth rows of the matrix to be exchanged. (b) What is the effect of multiplying on the *right* by P? Demonstrate with an example.

7. Change four entries of the leftmost matrix to make the matrix equation correct:

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

8. Find the PA = LU factorization of the matrix A in Exercise 2.3.15. What is the largest multiplier l_{ij} needed?

9. (a) Find the PA = LU factorization of $A = \begin{bmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$. (b) Let A be the $n \times n$ matrix of the same form as in (a). Describe the entries of each matrix of its PA = LU factorization.

10. (a) Assume that A is an $n \times n$ matrix with entries $|a_{ij}| \leq 1$ for $1 \leq i, j \leq n$. Prove that the matrix U in its PA = LU factorization satisfies $|u_{ij}| \leq 2^{n-1}$ for all $1 \leq i, j \leq n$. See Exercise 9(b). (b) Formulate and prove an analogous fact for an arbitrary $n \times n$ matrix A.

Reality Check 2 The Euler–Bernoulli Beam

The Euler–Bernoulli beam is a fundamental model for a material bending under stress. Discretization converts the differential equation model into a system of linear equations. The smaller the discretization size, the larger is the resulting system of equations. This example will provide us an interesting case study of the roles of system size and ill-conditioning in scientific computation.

The vertical displacement of the beam is represented by a function $y(x)$, where $0 \leq x \leq L$ along the beam of length L. We will use MKS units in the calculation: meters, kilograms, seconds. The displacement $y(x)$ satisfies the Euler–Bernoulli equation

$$EIy'''' = f(x) \tag{2.27}$$

where E, the Young's modulus of the material, and I, the area moment of inertia, are constant along the beam. The right-hand-side $f(x)$ is the applied load, including the weight of the beam, in force per unit length.

Techniques for discretizing derivatives are found in Chapter 5, where it will be shown that a reasonable approximation for the fourth derivative is

$$y''''(x) \approx \frac{y(x-2h) - 4y(x-h) + 6y(x) - 4y(x+h) + y(x+2h)}{h^4} \tag{2.28}$$

for a small increment h. The discretization error of this approximation is proportional to h^2 (see Exercise 5.1.21.). Our strategy will be to consider the beam as the union of many segments of length h, and to apply the discretized version of the differential equation on each segment.

For a positive integer n, set $h = L/n$. Consider the evenly spaced grid $0 = x_0 < x_1 < \ldots < x_n = L$, where $h = x_i - x_{i-1}$ for $i = 1, \ldots, n$. Replacing the differential equation (2.27) with the difference approximation (2.28) to get the system of linear equations for the displacements $y_i = y(x_i)$ yields

$$y_{i-2} - 4y_{i-1} + 6y_i - 4y_{i+1} + y_{i+2} = \frac{h^4}{EI} f(x_i). \tag{2.29}$$

We will develop n equations in the n unknowns $y_1, \ldots, y_n$. The coefficient matrix, or structure matrix, will have coefficients from the left-hand side of this equation. However, notice that we must alter the equations near the ends of the beam to take the boundary conditions into account.

A diving board is a beam with one end clamped at the support, and the opposite end free. This is called the **clamped-free** beam or sometimes the **cantilever** beam. The boundary conditions for the clamped (left) end and free (right) end are

$$y(0) = y'(0) = y''(L) = y'''(L) = 0.$$

In particular, $y_0 = 0$. Note that finding y_1, however, presents us with a problem, since applying the approximation (2.29) to the differential equation (2.27) at x_1 results in

$$y_{-1} - 4y_0 + 6y_1 - 4y_2 + y_3 = \frac{h^4}{EI} f(x_1), \tag{2.30}$$

and y_{-1} is not defined. Instead, we must use an alternate derivative approximation at the point x_1 near the clamped end. Exercise 5.1.22(a) derives the approximation

$$y''''(x_1) \approx \frac{16y(x_1) - 9y(x_1+h) + \frac{8}{3}y(x_1+2h) - \frac{1}{4}y(x_1+3h)}{h^4} \tag{2.31}$$

which is valid when $y(x_0) = y'(x_0) = 0$.

Calling the approximation "valid," for now, means that the discretization error of the approximation is proportional to h^2, the same as for equation (2.28). In theory, this means that the error in approximating the derivative in this way will decrease toward zero in the limit of small h. This concept will be the focal point of the discussion of numerical differentiation in Chapter 5. The result for us is that we can use approximation (2.31) to take the endpoint condition into account for $i = 1$, yielding

$$16y_1 - 9y_2 + \frac{8}{3}y_3 - \frac{1}{4}y_4 = \frac{h^4}{EI} f(x_1).$$

The free right end of the beam requires a little more work because we must compute y_i all the way to the end of the beam. Again, we need alternative derivative approximations at the last two points x_{n-1} and x_n. Exercise 5.1.22 gives the approximations

$$y''''(x_{n-1}) \approx \frac{-28y_n + 72y_{n-1} - 60y_{n-2} + 16y_{n-3}}{17h^4} \tag{2.32}$$

$$y''''(x_n) \approx \frac{72y_n - 156y_{n-1} + 96y_{n-2} - 12y_{n-3}}{17h^4} \tag{2.33}$$

which are valid under the assumption $y''(x_n) = y'''(x_n) = 0$.