Numerical Analysis Lecture Notes

Peter J. Olver

4. Gaussian Elimination

In this part, our focus will be on the most basic method for solving linear algebraic systems, known as *Gaussian Elimination* in honor of one of the all-time mathematical greats — the early nineteenth century German mathematician Carl Friedrich Gauss. As the father of linear algebra, his name will occur repeatedly throughout this text. Gaussian Elimination is quite elementary, but remains one of *the* most important algorithms in applied (as well as theoretical) mathematics. Our initial focus will be on the most important class of systems: those involving the same number of equations as unknowns — although we will eventually develop techniques for handling completely general linear systems. While the former typically have a unique solution, general linear systems may have either no solutions or infinitely many solutions. Since physical models require existence and uniqueness of their solution, the systems arising in applications often (but not always) involve the same number of equations as unknowns. Nevertheless, the ability to confidently handle all types of linear systems is a basic prerequisite for further progress in the subject. In contemporary applications, particularly those arising in numerical solutions of differential equations, in signal and image processing, and elsewhere, the governing linear systems can be huge, sometimes involving millions of equations in millions of unknowns, challenging even the most powerful supercomputer. So, a systematic and careful development of solution techniques is essential. Section 4.5 discusses some of the practical issues and limitations in computer implementations of the Gaussian Elimination method for large systems arising in applications.

4.1. Solution of Linear Systems.

Gaussian Elimination is a simple, systematic algorithm to solve systems of linear equations. It is the workhorse of linear algebra, and, as such, of absolutely fundamental importance in applied mathematics. In this section, we review the method in the most important case, in which there are the same number of equations as unknowns.

To illustrate, consider an elementary system of three linear equations

$$\begin{aligned} x + 2y + z &= 2, \\ 2x + 6y + z &= 7, \\ x + y + 4z &= 3, \end{aligned} \tag{4.1}$$

in three unknowns x, y, z. Linearity refers to the fact that the unknowns only appear to the first power, and there are no product terms like $x\,y$ or $x\,y\,z$. The basic solution method is to systematically employ the following fundamental operation:

Linear System Operation #1: Add a multiple of one equation to another equation.

Before continuing, you might try to convince yourself that this operation doesn't change the solutions to the system. Our goal is to judiciously apply the operation and so be led to a much simpler linear system that is easy to solve, and, moreover has the same solutions as the original. Any linear system that is derived from the original system by successive application of such operations will be called an *equivalent system*. By the preceding remark, *equivalent linear systems have the same solutions*.

The systematic feature is that we successively eliminate the variables in our equations in order of appearance. We begin by eliminating the first variable, x, from the second equation. To this end, we subtract twice the first equation from the second, leading to

$$\begin{aligned} x + 2y + z &= 2, \\ 2y - z &= 3, \\ x + y + 4z &= 3. \end{aligned} \tag{4.2}$$

Next, we eliminate x from the third equation by subtracting the first equation from it:

$$\begin{aligned} x + 2y + z &= 2, \\ 2y - z &= 3, \\ -y + 3z &= 1. \end{aligned} \tag{4.3}$$

The equivalent system (4.3) is already simpler than the original (4.1). Notice that the second and third equations do not involve x (by design) and so constitute a system of two linear equations for two unknowns. Moreover, once we have solved this subsystem for y and z, we can substitute the answer into the first equation, and we need only solve a single linear equation for x.

We continue on in this fashion, the next phase being the elimination of the second variable, y, from the third equation by adding $\frac{1}{2}$ the second equation to it. The result is

$$\begin{aligned} x + 2y + z &= 2, \\ 2y - z &= 3, \\ \tfrac{5}{2} z &= \tfrac{5}{2}, \end{aligned} \tag{4.4}$$

which is the simple system we are after. It is in what is called *triangular form*, which means that, while the first equation involves all three variables, the second equation only involves the second and third variables, and the last equation only involves the last variable.

Any triangular system can be straightforwardly solved by the method of *Back Substitution*. As the name suggests, we work backwards, solving the last equation first, which requires that $z = 1$. We substitute this result back into the penultimate equation, which becomes $2y - 1 = 3$, with solution $y = 2$. We finally substitute these two values for y and z into the first equation, which becomes $x + 5 = 2$, and so the solution to the triangular system (4.4) is

$$x = -3, \qquad y = 2, \qquad z = 1. \tag{4.5}$$

Moreover, since we only used our basic linear system operation to pass from (4.1) to the triangular system (4.4), this is also the solution to the original system of linear equations, as you can check. We note that the system (4.1) has a unique — meaning one and only one — solution, namely (4.5).

And that, barring a few minor complications that can crop up from time to time, is all that there is to the method of Gaussian Elimination! It is extraordinarily simple, but its importance cannot be overemphasized. Before exploring the relevant issues, it will help to reformulate our method in a more convenient matrix notation.

4.2. Gaussian Elimination — Regular Case.

With the basic matrix arithmetic operations in hand, let us now return to our primary task. The goal is to develop a systematic method for solving linear systems of equations. While we could continue to work directly with the equations, matrices provide a convenient alternative that begins by merely shortening the amount of writing, but ultimately leads to profound insight into the structure of linear systems and their solutions.

We begin by replacing the system (3.2) by its matrix constituents. It is convenient to ignore the vector of unknowns, and form the *augmented matrix*

$$M = \left(A \mid \mathbf{b} \right) = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} & b_m \end{array} \right) \tag{4.6}$$

which is an $m \times (n+1)$ matrix obtained by tacking the right hand side vector onto the original coefficient matrix. The extra vertical line is included just to remind us that the last column of this matrix plays a special role. For example, the augmented matrix for the system (4.1), i.e.,

$$\begin{aligned} x + 2y + z &= 2, \\ 2x + 6y + z &= 7, \\ x + y + 4z &= 3, \end{aligned} \qquad \text{is} \qquad M = \left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 2 & 6 & 1 & 7 \\ 1 & 1 & 4 & 3 \end{array} \right). \tag{4.7}$$

Note that one can immediately recover the equations in the original linear system from the augmented matrix. Since operations on equations also affect their right hand sides, keeping track of everything is most easily done through the augmented matrix.

For the time being, we will concentrate our efforts on linear systems that have the same number, n, of equations as unknowns. The associated coefficient matrix A is square, of size $n \times n$. The corresponding augmented matrix $M = \left(A \mid \mathbf{b} \right)$ then has size $n \times (n+1)$.

The matrix operation that assumes the role of Linear System Operation #1 is:

Elementary Row Operation #1:

Add a scalar multiple of one row of the augmented matrix to another row.

For example, if we add -2 times the first row of the augmented matrix (4.7) to the second row, the result is the row vector

$$-2 \left(1 \quad 2 \quad 1 \quad 2 \right) + \left(2 \quad 6 \quad 1 \quad 7 \right) = \left(0 \quad 2 \quad -1 \quad 3 \right).$$

The result can be recognized as the second row of the modified augmented matrix

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & 2 & -1 & 3 \\ 1 & 1 & 4 & 3 \end{array}\right) \tag{4.8}$$

that corresponds to the first equivalent system (4.2). When elementary row operation #1 is performed, it is critical that the result replaces the row being added to — *not* the row being multiplied by the scalar. Notice that the elimination of a variable in an equation — in this case, the first variable in the second equation — amounts to making its entry in the coefficient matrix equal to zero.

We shall call the $(1,1)$ entry of the coefficient matrix the *first pivot*. The precise definition of pivot will become clear as we continue; the one key requirement is that a pivot be *nonzero*. Eliminating the first variable x from the second and third equations amounts to making all the matrix entries in the column below the pivot equal to zero. We have already done this with the $(2,1)$ entry in (4.8). To make the $(3,1)$ entry equal to zero, we subtract (that is, add -1 times) the first row from the last row. The resulting augmented matrix is

$$\left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & 2 & -1 & 3 \\ 0 & -1 & 3 & 1 \end{array}\right),$$

which corresponds to the system (4.3). The *second pivot* is the $(2,2)$ entry of this matrix, which is 2, and is the coefficient of the second variable in the second equation. Again, the pivot must be nonzero. We use the elementary row operation of adding $\frac{1}{2}$ of the second row to the third row to make the entry below the second pivot equal to 0; the result is the augmented matrix

$$N = \left(\begin{array}{ccc|c} 1 & 2 & 1 & 2 \\ 0 & 2 & -1 & 3 \\ 0 & 0 & \frac{5}{2} & \frac{5}{2} \end{array}\right)$$

that corresponds to the triangular system (4.4). We write the final augmented matrix as

$$N = \big(\, U \mid \mathbf{c} \,\big), \qquad \text{where} \qquad U = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & 0 & \frac{5}{2} \end{pmatrix}, \qquad \mathbf{c} = \begin{pmatrix} 2 \\ 3 \\ \frac{5}{2} \end{pmatrix}.$$

The corresponding linear system has vector form

$$U\mathbf{x} = \mathbf{c}. \tag{4.9}$$

Its coefficient matrix U is *upper triangular*, which means that all its entries below the main diagonal are zero: $u_{ij} = 0$ whenever $i > j$. The three nonzero entries on its diagonal, $1, 2, \frac{5}{2}$, including the last one in the $(3,3)$ slot, are the three pivots. Once the system has been reduced to triangular form (4.9), we can easily solve it by Back Substitution, as before.

Gaussian Elimination — Regular Case

```
start
    for $j = 1$ to $n$
        if $m_{jj} = 0$, stop; print "$A$ is not regular"
        else for $i = j + 1$ to $n$
            set $l_{ij} = m_{ij} / m_{jj}$
            add $-l_{ij}$ times row $j$ of $M$ to row $i$ of $M$
        next $i$
    next $j$
end
```

The preceding algorithm for solving a linear system of n equations in n unknowns is known as *regular Gaussian Elimination*. A square matrix A will be called *regular*[†] if the algorithm successfully reduces it to upper triangular form U with all non-zero pivots on the diagonal. In other words, for regular matrices, as the algorithm proceeds, each successive pivot appearing on the diagonal must be nonzero; otherwise, the matrix is not regular. We then use the pivot row to make all the entries lying in the column below the pivot equal to zero through elementary row operations. The solution is found by applying Back Substitution to the resulting triangular system.

Let us state this algorithm in the form of a program, written in a general "pseudocode" that can be easily translated into any specific language, e.g., C++, FORTRAN, JAVA, MAPLE, MATHEMATICA or MATLAB. By convention, the same letter $M = (m_{ij})$ will be used to denote the current augmented matrix at each stage in the computation, keeping in mind that its entries will change as the algorithm progresses. We initialize $M = \big(A \mid \mathbf{b} \big)$. The final output of the program, assuming A is regular, is the augmented matrix $M = \big(U \mid \mathbf{c} \big)$, where U is the upper triangular matrix whose diagonal entries are the pivots, while $\mathbf{c}$ is the resulting vector of right hand sides in the triangular system $U\mathbf{x} = \mathbf{c}$.

Elementary Matrices

A key observation is that elementary row operations can, in fact, be realized by matrix multiplication. To this end, we introduce the first type of "elementary matrix". (Later we will meet two other types of elementary matrix, corresponding to two other kinds of elementary row operation.)

Definition 4.1. The *elementary matrix E* associated with an elementary row operation for m–rowed matrices is the matrix obtained by applying the row operation to the

[†] Strangely, there is no commonly accepted term to describe these kinds of matrices. For lack of a better alternative, we propose to use the adjective "regular" in the sequel.

$m \times m$ identity matrix I_m.

For example, applying the elementary row operation that adds -2 times the first row to the second row of the 3×3 identity matrix $\mathrm{I} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ results in the corresponding elementary matrix $E_1 = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. We claim that, if A is *any* 3–rowed matrix, then multiplying $E_1 A$ has the same effect as the given elementary row operation. For example,

$$\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 1 & 1 & 4 \end{pmatrix},$$

which you may recognize as the first elementary row operation we used to solve our illustrative example. If we set

$$E_1 = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad E_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}, \qquad E_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \end{pmatrix}, \tag{4.10}$$

then multiplication by E_1 will subtract twice the first row from the second row, multiplication by E_2 will subtract the first row from the third row, and multiplication by E_3 will add $\frac{1}{2}$ the second row to the third row — precisely the row operations used to place our original system in triangular form. Therefore, performing them in the correct order (and using the associativity of matrix multiplication), we conclude that when

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix}, \qquad \text{then} \qquad E_3 E_2 E_1 A = U = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & 0 & \frac{5}{2} \end{pmatrix}. \tag{4.11}$$

The reader is urged to check this by directly multiplying the indicated matrices.

In general, then, an $m \times m$ *elementary matrix* E *of the first type* will have all 1's on the diagonal, one nonzero entry c in some off-diagonal position (i, j), with $i \neq j$, and all other entries equal to zero. If A is any $m \times n$ matrix, then the matrix product $E A$ is equal to the matrix obtained from A by the elementary row operation adding c times row j to row i. (Note that the order of i and j is reversed.)

To undo the operation of adding c times row j to row i, we must perform the inverse row operation that subtracts c (or, equivalently, adds $-c$) times row j from row i. The corresponding *inverse elementary matrix* again has 1's along the diagonal and $-c$ in the (i, j) slot. Let us denote the inverses of the particular elementary matrices (4.10) by L_i, so that, according to our general rule,

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \qquad L_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 1 \end{pmatrix}. \tag{4.12}$$

Note that the products

$$L_1 E_1 = L_2 E_2 = L_3 E_3 = \mathrm{I} \tag{4.13}$$

yield the 3×3 identity matrix, reflecting the fact that the matrices represent mutually inverse row operations.

The product of the latter three elementary matrices (4.12) is equal to

$$L = L_1 L_2 L_3 = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -\frac{1}{2} & 1 \end{pmatrix}. \tag{4.14}$$

The matrix L is called a *special lower triangular* matrix, where "lower triangular" means that all the entries above the main diagonal are 0, while "special" indicates that all the entries on the diagonal are equal to 1. Observe that the entries of L below the diagonal are the same as the corresponding nonzero entries in the L_i. This is a general fact that holds when the lower triangular elementary matrices are multiplied in the correct order. More generally, the following elementary consequence of the laws of matrix multiplication will be used extensively.

Lemma 4.2. *If L and $\widehat{L}$ are lower triangular matrices of the same size, so is their product $L\widehat{L}$. If they are both special lower triangular, so is their product. Similarly, if $U, \widehat{U}$ are (special) upper triangular matrices, so is their product $U\widehat{U}$.*

The LU Factorization

We have almost arrived at our first important result. Let us compute the product of the matrices L and U in (4.11), (4.14). Using associativity of matrix multiplication, equations (4.13), and the basic property of the identity matrix I, we conclude that

$$\begin{aligned} LU &= (L_1L_2L_3)(E_3E_2E_1A) = L_1L_2(L_3E_3)E_2E_1A = L_1L_2\,\mathrm{I}\,E_2E_1A \\ &= L_1(L_2E_2)E_1A = L_1\,\mathrm{I}\,E_1A = (L_1E_1)A = \mathrm{I}\,A = A. \end{aligned}$$

In other words, we have *factored* the coefficient matrix $A = LU$ into a product of a special lower triangular matrix L and an upper triangular matrix U with the nonzero pivots on its main diagonal. By similar reasoning, the same holds true for almost all square matrices.

Theorem 4.3. *A matrix A is regular if and only if it can be factored*

$$A = LU, \tag{4.15}$$

where L is a special lower triangular matrix, having all 1's on the diagonal, and U is upper triangular with nonzero diagonal entries, which are the pivots of A. The nonzero off-diagonal entries l_{ij} for $i > j$ appearing in L prescribe the elementary row operations that bring A into upper triangular form; namely, one subtracts l_{ij} times row j from row i at the appropriate step of the Gaussian Elimination process.

In practice, to find the LU factorization of a square matrix A, one applies the regular Gaussian Elimination algorithm to reduce A to its upper triangular form U. The entries of L can be filled in during the course of the calculation with the negatives of the multiples used in the elementary row operations. If the algorithm fails to be completed, which happens whenever zero appears in any diagonal pivot position, then the original matrix is *not* regular, and does *not* have an LU factorization.

Example 4.4. Let us compute the LU factorization of the matrix $A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & 5 & 2 \\ 2 & -2 & 0 \end{pmatrix}$. Applying the Gaussian Elimination algorithm, we begin by adding -2 times the first row to the second row, and then adding -1 times the first row to the third. The result is the matrix $\begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & -3 & -1 \end{pmatrix}$. The next step adds the second row to the third row, leading to the upper triangular matrix $U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix}$, whose diagonal entries are the pivots. The corresponding lower triangular matrix is $L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix}$; its entries lying below the main diagonal are the *negatives* of the multiples we used during the elimination procedure. For instance, the $(2,1)$ entry indicates that we added -2 times the first row to the second row, and so on. The reader might wish to verify the resulting factorization

$$\begin{pmatrix} 2 & 1 & 1 \\ 4 & 5 & 2 \\ 2 & -2 & 0 \end{pmatrix} = A = LU = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix}.$$

Forward and Back Substitution

Once we know the LU factorization of a regular matrix A, we are able to solve any associated linear system $A\mathbf{x} = \mathbf{b}$ in two easy stages:

(1) First, solve the lower triangular system

$$L\mathbf{c} = \mathbf{b} \tag{4.16}$$

for the vector $\mathbf{c}$ by *Forward Substitution*. This is the same as Back Substitution, except one solves the equations for the variables in the direct order — from first to last. Explicitly,

$$c_1 = b_1, \qquad c_i = b_i - \sum_{j=1}^{i-1} l_{ij}\, c_j, \qquad \text{for} \qquad i = 2, 3, \ldots, n, \tag{4.17}$$

noting that the previously computed values of $c_1, \ldots, c_{i-1}$ are used to determine c_i.

(2) Second, solve the resulting upper triangular system

$$U\mathbf{x} = \mathbf{c} \tag{4.18}$$

by *Back Substitution*. The values of the unknowns

$$x_n = \frac{c_n}{u_{nn}}\,, \qquad x_i = \frac{1}{u_{ii}}\left(c_i \;-\; \sum_{j=i+1}^{n} u_{ij}\,x_j \right), \qquad \text{for} \qquad i = n-1,\ldots,2,1, \tag{4.19}$$

are successively computed, but now in reverse order. It is worth pointing out that the requirement that each pivot $u_{ii} \neq 0$ is essential here, as otherwise we would not be able to solve for the corresponding variable x_i.

Note that the combined algorithm does indeed solve the original system, since if

$$U\mathbf{x} = \mathbf{c} \qquad \text{and} \qquad L\,\mathbf{c} = \mathbf{b}, \qquad \text{then} \qquad A\,\mathbf{x} = L\,U\,\mathbf{x} = L\,\mathbf{c} = \mathbf{b}.$$

Example 4.5. With the LU decomposition

$$\begin{pmatrix} 2 & 1 & 1 \\ 4 & 5 & 2 \\ 2 & -2 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

found in Example 4.4, we can readily solve any linear system with the given coefficient matrix by Forward and Back Substitution. For instance, to find the solution to

$$\begin{pmatrix} 2 & 1 & 1 \\ 4 & 5 & 2 \\ 2 & -2 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix},$$

we first solve the lower triangular system

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}, \qquad \text{or, explicitly,} \qquad \begin{aligned} a \qquad\quad &= 1, \\ 2\,a + b \qquad &= 2, \\ a - b + c &= 2. \end{aligned}$$

The first equation says $a = 1$; substituting into the second, we find $b = 0$; the final equation yields $c = 1$. We then use Back Substitution to solve the upper triangular system

$$\begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \qquad \text{which is} \qquad \begin{aligned} 2\,x + y + z &= 1, \\ 3\,y \qquad &= 0, \\ -\,z &= 1. \end{aligned}$$

We find $z = -1$, then $y = 0$, and then $x = 1$, which is indeed the solution.

Thus, once we have found the LU factorization of the coefficient matrix A, the Forward and Back Substitution processes quickly produce the solution to any system $A\,\mathbf{x} = \mathbf{b}$. Moreover, they can be straightforwardly programmed on a computer. In practice, to solve a system from scratch, it is a matter of taste whether you work directly with the augmented matrix, or first determine the LU factorization of the coefficient matrix, and then apply Forward and Back Substitution to compute the solution.

4.3. Pivoting and Permutations.

The method of Gaussian Elimination presented so far applies only to regular matrices. But not every square matrix is regular; a simple class of examples is matrices whose upper left, i.e., $(1,1)$, entry is zero, and so cannot serve as the first pivot. More generally, the algorithm cannot proceed whenever a zero entry appears in the current pivot position on the diagonal. What then to do? The answer requires revisiting the source of the method.

Consider, as a specific example, the linear system

$$\begin{aligned} 2y + z &= 2, \\ 2x + 6y + z &= 7, \\ x + y + 4z &= 3. \end{aligned} \tag{4.20}$$

The augmented coefficient matrix is

$$\left(\begin{array}{ccc|c} 0 & 2 & 1 & 2 \\ 2 & 6 & 1 & 7 \\ 1 & 1 & 4 & 3 \end{array}\right).$$

In this case, the $(1,1)$ entry is 0, and so is not a legitimate pivot. The problem, of course, is that the first variable x does not appear in the first equation, and so we cannot use it to eliminate x in the other two equations. But this “problem” is actually a bonus — we already have an equation with only two variables in it, and so we only need to eliminate x from one of the other two equations. To be systematic, we rewrite the system in a different order,

$$\begin{aligned} 2x + 6y + z &= 7, \\ 2y + z &= 2, \\ x + y + 4z &= 3, \end{aligned}$$

by interchanging the first two equations. In other words, we employ

Linear System Operation #2: Interchange two equations.

Clearly, this operation does not change the solution and so produces an equivalent linear system. In our case, the augmented coefficient matrix

$$\left(\begin{array}{ccc|c} 2 & 6 & 1 & 7 \\ 0 & 2 & 1 & 2 \\ 1 & 1 & 4 & 3 \end{array}\right),$$

can be obtained from the original by performing the second type of row operation:

Elementary Row Operation #2: Interchange two rows of the matrix.

The new nonzero upper left entry, 2, can now serve as the first pivot, and we may continue to apply elementary row operations of Type #1 to reduce our matrix to upper triangular form. For this particular example, we eliminate the remaining nonzero entry in the first column by subtracting $\frac{1}{2}$ the first row from the last:

$$\left(\begin{array}{rrr|r} 2 & 6 & 1 & 7 \\ 0 & 2 & 1 & 2 \\ 0 & -2 & \frac{7}{2} & -\frac{1}{2} \end{array}\right).$$

The $(2,2)$ entry serves as the next pivot. To eliminate the nonzero entry below it, we add the second to the third row:

$$\left(\begin{array}{ccc|c} 2 & 6 & 1 & 7 \\ 0 & 2 & 1 & 2 \\ 0 & 0 & \frac{9}{2} & \frac{3}{2} \end{array}\right).$$

We have now placed the system in upper triangular form, with the three pivots $2, 2$, and $\frac{9}{2}$ along the diagonal. Back Substitution produces the solution $x = \frac{5}{6}$, $y = \frac{5}{6}$, $z = \frac{1}{3}$.

The row interchange that is required when a zero shows up in the diagonal pivot position is known as *pivoting*. Later, in Section 4.5, we will discuss practical reasons for pivoting even when a diagonal entry is nonzero. Let us distinguish the class of matrices that can be reduced to upper triangular form by Gaussian Elimination with pivoting. These matrices will prove to be of fundamental importance throughout linear algebra.

Definition 4.6. A square matrix is called *nonsingular* if it can be reduced to upper triangular form with all non-zero elements on the diagonal — the pivots — by elementary row operations of Types 1 and 2.

In contrast, a *singular* square matrix cannot be reduced to such upper triangular form by such row operations, because at some stage in the elimination procedure the diagonal entry and all the entries below it are zero. Every regular matrix is nonsingular, but, as we just saw, not every nonsingular matrix is regular. Uniqueness of solutions is the key defining characteristic of nonsingularity.

Theorem 4.7. *A linear system* $A\mathbf{x} = \mathbf{b}$ *has a unique solution for* every *choice of right hand side* $\mathbf{b}$ *if and only if its coefficient matrix* A *is square and nonsingular.*

We are able to prove the "if" part of this theorem, since nonsingularity implies reduction to an equivalent upper triangular form that has the same solutions as the original system. The unique solution to the system is then found by Back Substitution. The "only if" part will be proved later.

The revised version of the Gaussian Elimination algorithm, valid for all nonsingular coefficient matrices, is implemented by the accompanying pseudocode program. The starting point is the augmented matrix $M = \left(A \mid \mathbf{b} \right)$ representing the linear system $A\mathbf{x} = \mathbf{b}$. After successful termination of the program, the result is an augmented matrix in upper triangular form $M = \left(U \mid \mathbf{c} \right)$ representing the equivalent linear system $U\mathbf{x} = \mathbf{c}$. One then uses Back Substitution to determine the solution $\mathbf{x}$ to the linear system.

Gaussian Elimination — Nonsingular Case

```
start
    for $j = 1$ to $n$
        if $m_{kj} = 0$ for all $k \geq j$, stop; print "$A$ is singular"
        if $m_{jj} = 0$ but $m_{kj} \neq 0$ for some $k > j$, switch rows $k$ and $j$
        for $i = j + 1$ to $n$
            set $l_{ij} = m_{ij}/m_{jj}$
            add $-l_{ij}$ times row $j$ to row $i$ of $M$
        next $i$
    next $j$
end
```

Permutation Matrices

As with the first type of elementary row operation, row interchanges can be accomplished by multiplication by a second type of elementary matrix, which is found by applying the row operation to the identity matrix of the appropriate size. For instance, interchanging rows 1 and 2 of the 3×3 identity matrix produces the elementary interchange matrix $P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. The result PA of multiplying any 3–rowed matrix A on the left by P is the same as interchanging the first two rows of A. For instance,

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

Multiple row interchanges are accomplished by combining such elementary interchange matrices. Each such combination of row interchanges corresponds to a unique permutation matrix.

Definition 4.8. A *permutation matrix* is a matrix obtained from the identity matrix by any combination of row interchanges.

In particular, applying a row interchange to a permutation matrix produces another permutation matrix. The following result is easily established.

Lemma 4.9. *A matrix P is a permutation matrix if and only if each row of P contains all* 0 *entries except for a single* 1, *and, in addition, each column of P also contains all* 0 *entries except for a single* 1.

In general, if a permutation matrix P has a 1 in position (i,j), then the effect of multiplication by P is to move the j^{th} row of A into the i^{th} row of the product PA.

Example 4.10. There are six different 3×3 permutation matrices, namely

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}. \tag{4.21}$$

These have the following effects: if A is a matrix with row vectors $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$, then multiplication on the left by each of the six permutation matrices produces, respectively,

$$\begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{pmatrix}, \quad \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{r}_3 \\ \mathbf{r}_1 \end{pmatrix}, \quad \begin{pmatrix} \mathbf{r}_3 \\ \mathbf{r}_1 \\ \mathbf{r}_2 \end{pmatrix}, \quad \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{r}_1 \\ \mathbf{r}_3 \end{pmatrix}, \quad \begin{pmatrix} \mathbf{r}_3 \\ \mathbf{r}_2 \\ \mathbf{r}_1 \end{pmatrix}, \quad \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_3 \\ \mathbf{r}_2 \end{pmatrix}.$$

Thus, the first permutation matrix, which is the identity, does nothing. The fourth, fifth and sixth represent row interchanges. The second and third are non-elementary permutations, and can be realized by a pair of successive row interchanges.

An elementary combinatorial argument proves that there are a total of

$$n! = n\,(n-1)\,(n-2)\ \cdots\ 3 \cdot 2 \cdot 1 \tag{4.22}$$

different permutation matrices of size $n \times n$. Moreover, the product $P = P_1 P_2$ of any two permutation matrices is also a permutation matrix. An important point is that multiplication of permutation matrices is *noncommutative* — the order in which one permutes makes a difference. Switching the first and second rows, and then switching the second and third rows *does not* have the same effect as first switching the second and third rows and then switching the first and second rows!

The Permuted LU Factorization

As we now know, any nonsingular matrix A can be reduced to upper triangular form by elementary row operations of types #1 and #2. The row interchanges merely reorder the equations. If one performs all of the required row interchanges in advance, then the elimination algorithm can proceed without requiring any further pivoting. Thus, the matrix obtained by permuting the rows of A in the prescribed manner is regular. In other words, if A is a nonsingular matrix, then there is a permutation matrix P such that the product PA is regular, and hence admits an LU factorization. As a result, we deduce the general *permuted LU factorization*

$$PA = LU, \tag{4.23}$$

where P is a permutation matrix, L is special lower triangular, and U is upper triangular with the pivots on the diagonal. For instance, in the preceding example, we permuted the first and second rows, and hence equation (4.23) has the explicit form

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2} & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 6 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & \frac{9}{2} \end{pmatrix}.$$

We have now established the following generalization of Theorem 4.3.

Theorem 4.11. *Let A be an $n \times n$ matrix. Then the following conditions are equivalent*:

(i) *A is nonsingular.*
(ii) *A has n nonzero pivots.*
(iii) *A admits a permuted LU factorization*: $P A = L U$.

A practical method to construct a permuted LU factorization of a given matrix A would proceed as follows. First set up $P = L = \mathrm{I}$ as $n \times n$ identity matrices. The matrix P will keep track of the permutations performed during the Gaussian Elimination process, while the entries of L below the diagonal are gradually replaced by the negatives of the multiples used in the corresponding row operations of type #1. Each time two rows of A are interchanged, the same two rows of P will be interchanged. Moreover, any pair of entries that both lie *below* the diagonal in these same two rows of L must also be interchanged, while entries lying on and above its diagonal need to stay in their place. At a successful conclusion to the procedure, A will have been converted into the upper triangular matrix U, while L and P will assume their final form. Here is an illustrative example.

Example 4.12. Our goal is to produce a permuted LU factorization of the matrix

$$A = \begin{pmatrix} 1 & 2 & -1 & 0 \\ 2 & 4 & -2 & -1 \\ -3 & -5 & 6 & 1 \\ -1 & 2 & 8 & -2 \end{pmatrix}.$$

To begin the procedure, we apply row operations of type #1 to eliminate the entries below the first pivot. The updated matrices[†] are

$$A = \begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 1 & 3 & 1 \\ 0 & 4 & 7 & -2 \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ -3 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

where L keeps track of the row operations, and we initialize P to be the identity matrix. The $(2,2)$ entry of the new A is zero, and so we interchange its second and third rows, leading to

$$A = \begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & -1 \\ 0 & 4 & 7 & -2 \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

We interchanged the same two rows of P, while in L we only interchanged the already computed entries in its second and third rows that lie in its first column below the diagonal.

† Here, we are adopting computer programming conventions, where updates of a matrix are all given the same name.

We then eliminate the nonzero entry lying below the $(2,2)$ pivot, leading to

$$A = \begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -5 & -6 \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ -1 & 4 & 0 & 1 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

A final row interchange places the matrix in upper triangular form:

$$U = A = \begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & -5 & -6 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ -1 & 4 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Again, we performed the same row interchange on P, while only interchanging the third and fourth row entries of L that lie below the diagonal. You can verify that

$$PA = \begin{pmatrix} 1 & 2 & -1 & 0 \\ -3 & -5 & 6 & 1 \\ -1 & 2 & 8 & -2 \\ 2 & 4 & -2 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ -1 & 4 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & -5 & -6 \\ 0 & 0 & 0 & -1 \end{pmatrix} = LU, \tag{4.24}$$

as promised. Thus, by rearranging the equations in the order first, third, fourth, second, as prescribed by P, we obtain an equivalent linear system with regular coefficient matrix PA.

Once the permuted LU factorization is established, the solution to the original system $A\mathbf{x} = \mathbf{b}$ is obtained by applying the same Forward and Back Substitution algorithm presented above. Explicitly, we first multiply the system $A\mathbf{x} = \mathbf{b}$ by the permutation matrix, leading to

$$PA\mathbf{x} = P\mathbf{b} = \widehat{\mathbf{b}}, \tag{4.25}$$

whose right hand side $\widehat{\mathbf{b}}$ has been obtained by permuting the entries of $\mathbf{b}$ in the same fashion as the rows of A. We then solve the two triangular systems

$$L\mathbf{c} = \widehat{\mathbf{b}} \qquad \text{and} \qquad U\mathbf{x} = \mathbf{c} \tag{4.26}$$

by, respectively, Forward and Back Substitution.

Example 4.12. (*continued*) Suppose we wish to solve the linear system

$$\begin{pmatrix} 1 & 2 & -1 & 0 \\ 2 & 4 & -2 & -1 \\ -3 & -5 & 6 & 1 \\ -1 & 2 & 8 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 3 \\ 0 \end{pmatrix}.$$

In view of the $PA = LU$ factorization established in (4.24), we need only solve the two auxiliary lower and upper triangular systems (4.26). The lower triangular system is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ -1 & 4 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 0 \\ -1 \end{pmatrix};$$

whose right hand side was obtained by applying the permutation matrix P to the right hand side of the original system. Its solution, namely $a = 1$, $b = 6$, $c = -23$, $d = -3$, is obtained through Forward Substitution. The resulting upper triangular system is

$$\begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 3 & 1 \\ 0 & 0 & -5 & -6 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} 1 \\ 6 \\ -23 \\ -3 \end{pmatrix}.$$

Its solution, $w = 3$, $z = 1$, $y = 0$, $x = 2$, which is also the solution to the original system, is easily obtained by Back Substitution.

4.4. Gauss–Jordan Elimination.

The principal algorithm used to compute the inverse of a nonsingular matrix is known as *Gauss–Jordan Elimination*, in honor of Gauss and Wilhelm Jordan, a nineteenth century German engineer. A key fact is that we only need to solve the right inverse equation

$$A X = \mathrm{I} \tag{4.27}$$

in order to compute $X = A^{-1}$. The left inverse equation in (3.7), namely $X A = \mathrm{I}$, will then follow as an automatic consequence. In other words, for square matrices, a right inverse is automatically a left inverse, and conversely! A proof will appear below.

The reader may well ask, then, why use both left and right inverse conditions in the original definition? There are several good reasons. First of all, a non-square matrix may satisfy one of the two conditions — having either a left inverse or a right inverse — but can never satisfy both. Moreover, even when we restrict our attention to square matrices, starting with only one of the conditions makes the logical development of the subject considerably more difficult, and not really worth the extra effort. Once we have established the basic properties of the inverse of a square matrix, we can then safely discard the superfluous left inverse condition. Finally, when we generalize the notion of an inverse to linear operators, then, unlike square matrices, we *cannot* dispense with either of the conditions.

Let us write out the individual columns of the right inverse equation (4.27). The j^{th} column of the $n \times n$ identity matrix I is the vector $\mathbf{e}_j$ that has a single 1 in the j^{th} slot and 0's elsewhere, so

$$\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \qquad \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \qquad \ldots \qquad \mathbf{e}_n = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}. \tag{4.28}$$

According to (3.6), the j^{th} column of the matrix product $A X$ is equal to $A\mathbf{x}_j$, where $\mathbf{x}_j$ denotes the j^{th} column of the inverse matrix X. Therefore, the single matrix equation (4.27) is equivalent to n linear systems

$$A\mathbf{x}_1 = \mathbf{e}_1, \qquad A\mathbf{x}_2 = \mathbf{e}_2, \qquad \ldots \qquad A\mathbf{x}_n = \mathbf{e}_n, \tag{4.29}$$

all having the same coefficient matrix. As such, to solve them we should form the n augmented matrices $M_1 = \left(A \mid \mathbf{e}_1 \right), \ldots, M_n = \left(A \mid \mathbf{e}_n \right)$, and then apply our Gaussian Elimination algorithm to each. But this would be a waste of effort. Since the coefficient matrix is the same, we will end up performing *identical* row operations on each augmented matrix. Clearly, it will be more efficient to combine them into one large augmented matrix $M = \left(A \mid \mathbf{e}_1 \ \ldots \ \mathbf{e}_n \right) = \left(A \mid \mathrm{I} \right)$, of size $n \times (2n)$, in which the right hand sides $\mathbf{e}_1, \ldots, \mathbf{e}_n$ of our systems are placed into n different columns, which we then recognize as reassembling the columns of an $n \times n$ identity matrix. We may then simultaneously apply our elementary row operations to reduce, if possible, the large augmented matrix so that its first n columns are in upper triangular form.

Example 4.13. For example, to find the inverse of the matrix $A = \begin{pmatrix} 0 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix}$, we form the large augmented matrix

$$\left(\begin{array}{ccc|ccc} 0 & 2 & 1 & 1 & 0 & 0 \\ 2 & 6 & 1 & 0 & 1 & 0 \\ 1 & 1 & 4 & 0 & 0 & 1 \end{array}\right).$$

Applying the same sequence of elementary row operations as in Section 4.3, we first interchange the rows

$$\left(\begin{array}{ccc|ccc} 2 & 6 & 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 1 & 1 & 4 & 0 & 0 & 1 \end{array}\right),$$

and then eliminate the nonzero entries below the first pivot,

$$\left(\begin{array}{ccc|ccc} 2 & 6 & 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & -2 & \frac{7}{2} & 0 & -\frac{1}{2} & 1 \end{array}\right).$$

Next we eliminate the entry below the second pivot:

$$\left(\begin{array}{ccc|ccc} 2 & 6 & 1 & 0 & 1 & 0 \\ 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & \frac{9}{2} & 1 & -\frac{1}{2} & 1 \end{array}\right).$$

At this stage, we have reduced our augmented matrix to the form $\left(U \mid C \right)$ where U is upper triangular. This is equivalent to reducing the original n linear systems $A\,\mathbf{x}_i = \mathbf{e}_i$ to n upper triangular systems $U\,\mathbf{x}_i = \mathbf{c}_i$. We can therefore perform n back substitutions to produce the solutions $\mathbf{x}_i$, which would form the individual columns of the inverse matrix $X = (\mathbf{x}_1 \ \ldots \ \mathbf{x}_n)$. In the more common version of the Gauss–Jordan scheme, one instead continues to employ elementary row operations to fully reduce the augmented matrix. The goal is to produce an augmented matrix $\left(\mathrm{I} \mid X \right)$ in which the left hand $n \times n$ matrix has become the identity, while the right hand matrix is the desired solution $X = A^{-1}$. Indeed, $\left(\mathrm{I} \mid X \right)$ represents the n trivial linear systems $\mathrm{I}\,\mathbf{x} = \mathbf{x}_i$ whose solutions $\mathbf{x} = \mathbf{x}_i$ are the columns of the inverse matrix X.

Now, the identity matrix has 0's below the diagonal, just like U. It also has 1's along the diagonal, whereas U has the pivots (which are all nonzero) along the diagonal. Thus, the next phase in the reduction process is to make all the diagonal entries of U equal to 1. To proceed, we need to introduce the last, and least, of our linear systems operations.

Linear System Operation #3: Multiply an equation by a nonzero constant.

This operation clearly does not affect the solution, and so yields an equivalent linear system. The corresponding elementary row operation is:

Elementary Row Operation #3: Multiply a row of the matrix by a nonzero scalar.

Dividing the rows of the upper triangular augmented matrix $\big(U \mid C \big)$ by the diagonal pivots of U will produce a matrix of the form $\big(V \mid B \big)$ where V is *special upper triangular*, meaning it has all 1's along the diagonal. In our particular example, the result of these three elementary row operations of Type #3 is

$$\left(\begin{array}{ccc|ccc} 1 & 3 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 1 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} \end{array}\right),$$

where we multiplied the first and second rows by $\frac{1}{2}$ and the third row by $\frac{2}{9}$.

We are now over halfway towards our goal. We need only make the entries above the diagonal of the left hand matrix equal to zero. This can be done by elementary row operations of Type #1, but now we work backwards. First, we eliminate the nonzero entries in the third column lying above the $(3,3)$ entry by subtracting one half the third row from the second and also from the first:

$$\left(\begin{array}{ccc|ccc} 1 & 3 & 0 & -\frac{1}{9} & \frac{5}{9} & -\frac{1}{9} \\ 0 & 1 & 0 & \frac{7}{18} & \frac{1}{18} & -\frac{1}{9} \\ 0 & 0 & 1 & \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} \end{array}\right).$$

Finally, we subtract 3 times the second row from the first to eliminate the remaining nonzero off-diagonal entry, thereby completing the Gauss–Jordan procedure:

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & -\frac{23}{18} & \frac{7}{18} & \frac{2}{9} \\ 0 & 1 & 0 & \frac{7}{18} & \frac{1}{18} & -\frac{1}{9} \\ 0 & 0 & 1 & \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} \end{array}\right).$$

The left hand matrix is the identity, and therefore the final right hand matrix is our desired inverse:

$$A^{-1} = \begin{pmatrix} -\frac{23}{18} & \frac{7}{18} & \frac{2}{9} \\ \frac{7}{18} & \frac{1}{18} & -\frac{1}{9} \\ \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} \end{pmatrix}. \tag{4.30}$$

The reader may wish to verify that the final result does satisfy both inverse conditions $A\,A^{-1} = \mathrm{I} = A^{-1}A$.

We are now able to complete the proofs of the basic results on inverse matrices. First, we need to determine the elementary matrix corresponding to an elementary row operation of type #3. Again, this is obtained by performing the row operation in question on the identity matrix. Thus, the elementary matrix that multiplies row i by the nonzero scalar c is the diagonal matrix having c in the i^{th} diagonal position, and 1's elsewhere along the diagonal. The inverse elementary matrix is the diagonal matrix with $1/c$ in the i^{th} diagonal position and 1's elsewhere on the main diagonal; it corresponds to the inverse operation that divides row i by c. For example, the elementary matrix that multiplies the second row of a 3–rowed matrix by 5 is $E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, and has inverse $E^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{5} & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

In summary:

Lemma 4.14. *Every elementary matrix is nonsingular, and its inverse is also an elementary matrix of the same type.*

The Gauss–Jordan method tells us how to reduce any nonsingular square matrix A to the identity matrix by a sequence of elementary row operations. Let $E_1, E_2, \ldots, E_N$ be the corresponding elementary matrices. The elimination procedure that reduces A to I amounts to multiplying A by a succession of elementary matrices:

$$E_N E_{N-1} \,\cdots\, E_2 E_1 A = \mathrm{I}\,. \tag{4.31}$$

We claim that the product matrix

$$X = E_N E_{N-1} \,\cdots\, E_2 E_1 \tag{4.32}$$

is the inverse of A. Indeed, formula (4.31) says that $X A = \mathrm{I}$, and so X is a left inverse. Furthermore, each elementary matrix has an inverse, and so by (3.11), X itself is invertible, with

$$X^{-1} = E_1^{-1} E_2^{-1} \,\cdots\, E_{N-1}^{-1} E_N^{-1}. \tag{4.33}$$

Therefore, multiplying formula (4.31), namely $X A = \mathrm{I}$, on the left by X^{-1} leads to $A = X^{-1}$. Lemma 3.5 implies $X = A^{-1}$. We have thus proved

Theorem 4.15. *A square matrix A has an inverse if and only if it is nonsingular.*

Consequently, an $n \times n$ matrix will have an inverse if and only if it can be reduced to upper triangular form, with n nonzero pivots on the diagonal, by a combination of elementary row operations. Indeed, "invertible" is often used as a synonym for "nonsingular". All other matrices are singular and do not have an inverse as defined above. Before attempting to prove Theorem 4.15, we need to first become familiar with some elementary properties of matrix inverses.

Finally, equating $A = X^{-1}$ to the product (4.33), and invoking Lemma 4.14, we have established the following result.

Proposition 4.16. *Every nonsingular matrix A can be written as the product of elementary matrices.*

Example 4.17. The 2×2 matrix $A = \begin{pmatrix} 0 & -1 \\ 1 & 3 \end{pmatrix}$ is converted into the identity matrix by first interchanging its rows, $\begin{pmatrix} 1 & 3 \\ 0 & -1 \end{pmatrix}$, then scaling the second row by -1, $\begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix}$, and, finally, subtracting 3 times the second row from the first to obtain $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathrm{I}$. The corresponding elementary matrices are

$$E_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad E_2 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad E_3 = \begin{pmatrix} 1 & -3 \\ 0 & 1 \end{pmatrix}.$$

Therefore, by (4.32),

$$A^{-1} = E_3 E_2 E_1 = \begin{pmatrix} 1 & -3 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 1 \\ -1 & 0 \end{pmatrix},$$

while

$$A = E_1^{-1} E_2^{-1} E_3^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 3 \end{pmatrix}.$$

As an application, let us prove that the inverse of a nonsingular triangular matrix is also triangular. Specifically:

Proposition 4.18. *If L is a lower triangular matrix with all nonzero entries on the main diagonal, then L is nonsingular and its inverse L^{-1} is also lower triangular. In particular, if L is special lower triangular, so is L^{-1}. A similar result holds for upper triangular matrices.*

Proof: It suffices to note that if L has all nonzero diagonal entries, one can reduce L to the identity by elementary row operations of Types #1 and #3, whose associated elementary matrices are all lower triangular. Lemma 4.2 implies that the product (4.32) is then also lower triangular. If L is special, then all the pivots are equal to 1. Thus, no elementary row operations of Type #3 are required, and so L can be reduced to the identity matrix by elementary row operations of Type #1 alone. Therefore, its inverse is a product of special lower triangular matrices, and hence is itself special lower triangular. A similar argument applies in the upper triangular cases. *Q.E.D.*

Solving Linear Systems with the Inverse

The primary motivation for introducing the matrix inverse is that it provides a compact formula for the solution to any linear system with an invertible coefficient matrix.

Theorem 4.19. *If A is nonsingular, then $\mathbf{x} = A^{-1}\mathbf{b}$ is the unique solution to the linear system $A\mathbf{x} = \mathbf{b}$.*

Proof: We merely multiply the system by A^{-1}, which yields $\mathbf{x} = A^{-1}A\mathbf{x} = A^{-1}\mathbf{b}$. Moreover, $A\mathbf{x} = A A^{-1}\mathbf{b} = \mathbf{b}$, proving that $\mathbf{x} = A^{-1}\mathbf{b}$ is indeed the solution. *Q.E.D.*

For example, let us return to the linear system (4.20). Since we computed the inverse of its coefficient matrix in (4.30), a "direct" way to solve the system is to multiply the right hand side by the inverse matrix:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -\frac{23}{18} & \frac{7}{18} & \frac{2}{9} \\ \frac{7}{18} & \frac{1}{18} & -\frac{1}{9} \\ \frac{2}{9} & -\frac{1}{9} & \frac{2}{9} \end{pmatrix} \begin{pmatrix} 2 \\ 7 \\ 3 \end{pmatrix} = \begin{pmatrix} \frac{5}{6} \\ \frac{5}{6} \\ \frac{1}{3} \end{pmatrix},$$

reproducing our earlier solution.

However, while æsthetically appealing, the solution method based on the inverse matrix is hopelessly inefficient as compared to direct Gaussian Elimination, and, despite what you may have learned, *should not be used in practical computations*. (A complete justification of this dictum will be provided in Section 4.5.) On the other hand, the inverse does play a useful role in theoretical developments, as well as providing insight into the design of practical algorithms. But the principal message of applied linear algebra is that LU decomposition and Gaussian Elimination are fundamental; matrix inverses are to be avoided in all but the most elementary computations.

Remark: The reader may have learned a version of the Gauss–Jordan algorithm for solving a single linear system that replaces the Back Substitution step by a complete reduction of the coefficient matrix to the identity. In other words, to solve $A\mathbf{x} = \mathbf{b}$, we start with the augmented matrix $M = \begin{pmatrix} A \mid \mathbf{b} \end{pmatrix}$ and use all three types of elementary row operations to produce (assuming nonsingularity) the fully reduced form $\begin{pmatrix} \mathrm{I} \mid \mathbf{d} \end{pmatrix}$, representing the trivially soluble, equivalent system $\mathbf{x} = \mathbf{d}$, which is the solution to the original system. However, Back Substitution is more efficient, and remains the method of choice in practical computations.

The LDV Factorization

The second phase of the Gauss–Jordan process leads to a slightly more detailed version of the LU factorization. Let D denote the diagonal matrix having the same diagonal entries as U; in other words, D contains the pivots on its diagonal and zeros everywhere else. Let V be the special upper triangular matrix obtained from U by dividing each row by its pivot, so that V has all 1's on the diagonal. We already encountered V during the course of the Gauss–Jordan procedure. It is easily seen that $U = DV$, which implies the following result.

Theorem 4.20. *A matrix A is regular if and only if it admits a factorization*

$$A = LDV, \tag{4.34}$$

where L is a special lower triangular matrix, D is a diagonal matrix having the nonzero pivots on the diagonal, and V is a special upper triangular matrix.

For the matrix appearing in Example 4.4, we have $U = DV$, where

$$U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \qquad D = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix}, \qquad V = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

This leads to the factorization

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & 5 & 2 \\ 2 & -2 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = L\,D\,V.$$

Proposition 4.21. *If $A = LU$ is regular, then the factors L and U are uniquely determined. The same holds for the $A = L\,D\,V$ factorization.*

Proof: Suppose $LU = \widetilde{L}\,\widetilde{U}$. Since the diagonal entries of all four matrices are nonzero, Proposition 4.18 implies that they are invertible. Therefore,

$$\widetilde{L}^{-1}L = \widetilde{L}^{-1}L\,U\,U^{-1} = \widetilde{L}^{-1}\widetilde{L}\,\widetilde{U}\,U^{-1} = \widetilde{U}\,U^{-1}. \tag{4.35}$$

The left hand side of the matrix equation (4.35) is the product of two special lower triangular matrices, and so, by Lemma 4.2, is itself special lower triangular. The right hand side is the product of two upper triangular matrices, and hence is upper triangular. But the only way a special lower triangular matrix could equal an upper triangular matrix is if they both equal the diagonal identity matrix. Therefore, $\widetilde{L}^{-1}L = \mathrm{I} = \widetilde{U}\,U^{-1}$, and so $\widetilde{L} = L$ and $\widetilde{U} = U$, proving the first result. The $L\,D\,V$ version is an immediate consequence. *Q.E.D.*

As you may have guessed, the more general cases requiring one or more row interchanges lead to a permuted $L\,D\,V$ factorization in the following form.

Theorem 4.22. *A matrix A is nonsingular if and only if there is a permutation matrix P such that*

$$P\,A = L\,D\,V, \tag{4.36}$$

where L, D, V are, respectively, special lower triangular, diagonal, and special upper triangular matrices.

Uniqueness does not hold for the more general permuted factorizations (4.23), (4.36), since there may be several permutation matrices that place a matrix in regular form. Moreover, unlike regular elimination, the pivots, i.e., the diagonal entries of U, are no longer uniquely defined, but depend on the particular combination of row interchanges employed during the course of the computation.

The $L\,D\,V$ factorization of a nonsingular matrix takes a particularly simple form if the matrix also happens to be symmetric. This result will form the foundation of some significant later developments.

Theorem 4.23. *A symmetric matrix A is regular if and only if it can be factored as*

$$A = L\,D\,L^T, \tag{4.37}$$

where L is a special lower triangular matrix and D is a diagonal matrix with nonzero diagonal entries.

Proof: We already know, according to Theorem 4.20, that we can factor

$$A = L\,D\,V. \tag{4.38}$$

We take the transpose of both sides of this equation:

$$A^T = (L\,D\,V)^T = V^T D^T L^T = V^T D\,L^T, \tag{4.39}$$

since diagonal matrices are automatically symmetric: $D^T = D$. Note that V^T is special lower triangular, and L^T is special upper triangular. Therefore (4.39) is the $L\,D\,V$ factorization of A^T.

In particular, if A is symmetric, then

$$L\,D\,V = A = A^T = V^T D\,L^T.$$

Uniqueness of the $L\,D\,V$ factorization implies that

$$L = V^T \qquad \text{and} \qquad V = L^T$$

(which are two versions of the same equation). Replacing V by L^T in (4.38) establishes the factorization (4.37). *Q.E.D.*

Remark: If $A = L\,D\,L^T$, then A is necessarily symmetric. Indeed,

$$A^T = (L\,D\,L^T)^T = (L^T)^T D^T L^T = L\,D\,L^T = A.$$

However, not every symmetric matrix has an $L\,D\,L^T$ factorization. A simple example is the irregular but nonsingular 2×2 matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

Example 4.24. The problem is to find the $L\,D\,L^T$ factorization of the particular symmetric matrix $A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix}$. This requires performing the usual Gaussian Elimination algorithm. Subtracting twice the first row from the second and also the first row from the third produces the matrix $\begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & -1 & 3 \end{pmatrix}$. We then add one half of the second row of the latter matrix to its third row, resulting in the upper triangular form

$$U = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & 0 & \frac{5}{2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & \frac{5}{2} \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & -\frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix} = D\,V,$$

which we further factor by dividing each row of U by its pivot. On the other hand, the special lower triangular matrix associated with the preceding row operations is $L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -\frac{1}{2} & 1 \end{pmatrix}$, which, as guaranteed by Theorem 4.23, is the transpose of $V = L^T$.

Therefore, the desired $A = LU = LDL^T$ factorizations of this particular symmetric matrix are

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -\frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & 0 & \frac{5}{2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & -\frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & \frac{5}{2} \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & -\frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}.$$

Example 4.25. Let us look at a general 2×2 symmetric matrix $A = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$. Regularity requires that the first pivot be $a \neq 0$. A single row operation will place A in upper triangular form $U = \begin{pmatrix} a & c \\ 0 & \dfrac{ac - b^2}{a} \end{pmatrix}$. The associated lower triangular matrix is $L = \begin{pmatrix} 1 & 0 \\ \dfrac{b}{a} & 1 \end{pmatrix}$. Thus, $A = LU$. Finally, $D = \begin{pmatrix} a & 0 \\ 0 & \dfrac{ac - b^2}{a} \end{pmatrix}$ is just the diagonal part of U, and we find $U = DL^T$, so that the LDL^T factorization is explicitly given by

$$\begin{pmatrix} a & b \\ b & c \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \dfrac{b}{a} & 1 \end{pmatrix} \begin{pmatrix} a & 0 \\ 0 & \dfrac{ac - b^2}{a} \end{pmatrix} \begin{pmatrix} 1 & \dfrac{b}{a} \\ 0 & 1 \end{pmatrix}. \tag{4.40}$$

4.5. Practical Linear Algebra.

For pedagogical and practical reasons, the examples and exercises we have chosen to illustrate the algorithms are all based on relatively small matrices. When dealing with matrices of moderate size, the differences between the various approaches to solving linear systems (Gauss, Gauss–Jordan, matrix inverse, etc.) are relatively unimportant, particularly if one has a decent computer or even hand calculator to do the tedious parts. However, real-world applied mathematics deals with much larger linear systems, and the design of efficient algorithms is a must. For example, numerical solution schemes for ordinary differential equations will typically lead to matrices with thousands of entries, while numerical schemes for partial differential equations arising in fluid and solid mechanics, weather prediction, image and video processing, quantum mechanics, molecular dynamics, chemical processes, etc., will often require dealing with matrices with more than a million entries. It is not hard for such systems to tax even the most sophisticated supercomputer. Thus, it is essential that we understand the computational details of competing methods in order to compare their efficiency, and thereby gain some experience with the issues underlying the design of high performance numerical algorithms.

The most basic question is: how many arithmetic operations[†] are required to complete an algorithm? The number will directly influence the time spent running the algorithm on a computer. We shall keep track of additions and multiplications separately, since the

[†] For simplicity, we will only count basic arithmetic operations. But it is worth noting that other issues, such as the number of I/O operations, may also play a role in estimating the computational complexity of a numerical algorithm.

latter typically take longer to process. But we shall not distinguish between addition and subtraction, nor between multiplication and division, as these typically rely on the same floating point algorithm. We shall also assume that the matrices and vectors we deal with are *generic*, with few, if any, zero entries. Modifications of the basic algorithms for *sparse matrices*, meaning those that have lots of zero entries, are an important topic of research, since these include many of the large matrices that appear in applications to differential equations. We refer the interested reader to more advanced treatments of numerical linear algebra, such as [**13**, **25**, **47**, **54**], for further developments.

First, when multiplying an $n \times n$ matrix A and an $n \times 1$ column vector $\mathbf{b}$, each entry of the product $A\mathbf{b}$ requires n multiplications of the form $a_{ij}\, b_j$ and $n-1$ additions to sum the resulting products. Since there are n entries, this means a total of n^2 multiplications and $n(n-1) = n^2 - n$ additions. Thus, for a matrix of size $n = 100$, one needs about $10,000$ distinct multiplications and a similar number of additions. If $n = 1,000,000 = 10^6$, then $n^2 = 10^{12}$, which is phenomenally large, and the total time required to perform the computation becomes a significant issue.

Let us next look at the (regular) Gaussian Elimination algorithm, referring back to our pseudocode program for the notational details. First, we count how many arithmetic operations are based on the j^{th} pivot m_{jj}. For each of the $n-j$ rows lying below it, we must perform one division to compute the factor $l_{ij} = m_{ij}/m_{jj}$ used in the elementary row operation. The entries in the column below the pivot will be set to zero automatically, and so we need only compute the updated entries lying strictly below and to the right of the pivot. There are $(n-j)^2$ such entries in the coefficient matrix and an additional $n-j$ entries in the last column of the augmented matrix. Let us concentrate on the former for the moment. For each of these, we replace m_{ik} by $m_{ik} - l_{ij}\, m_{jk}$, and so must perform one multiplication and one addition. For the j^{th} pivot, there are a total of $(n-j)(n-j+1)$ multiplications — including the initial $n-j$ divisions needed to produce the l_{ij} — and $(n-j)^2$ additions needed to update the coefficient matrix. Therefore, to reduce a regular $n \times n$ matrix to upper triangular form requires a total of

$$\begin{aligned} &\sum_{j=1}^{n} (n-j)(n-j+1) = \frac{n^3 - n}{3} \qquad && \text{multiplications and} \\ &\sum_{j=1}^{n} (n-j)^2 = \frac{2n^3 - 3n^2 + n}{6} \qquad && \text{additions.} \end{aligned} \tag{4.41}$$

Thus, when n is large, both involve approximately $\frac{1}{3}n^3$ operations.

We should also be keeping track of the number of operations on the right hand side of the system. No pivots appear there, and so there are

$$\sum_{j=1}^{n} (n-j) = \frac{n^2 - n}{2} \tag{4.42}$$

multiplications and the same number of additions required to produce the right hand side in the resulting triangular system $U\mathbf{x} = \mathbf{c}$. For large n, this count is considerably smaller than the coefficient matrix totals (4.41). We note that the Forward Substitution equations

(4.17) require precisely the same number of arithmetic operations to solve $L\mathbf{c} = \mathbf{b}$ for the right hand side of the upper triangular system. Indeed, the j^{th} equation

$$c_j = b_j - \sum_{k=1}^{j-1} l_{jk} c_k$$

requires $j-1$ multiplications and the same number of additions, giving a total of

$$\sum_{j=1}^{n} j = \frac{n^2 - n}{2}$$

operations of each type. Therefore, to reduce a linear system to upper triangular form, it makes no difference in computational efficiency whether one works directly with the augmented matrix or employs Forward Substitution after the LU factorization of the coefficient matrix has been established.

The Back Substitution phase of the algorithm can be similarly analyzed. To find the value of

$$x_j = \frac{1}{u_{jj}} \left(c_j - \sum_{k=j+1}^{n} u_{jk} x_k \right)$$

once we have computed $x_{j+1}, \ldots, x_n$, requires $n-j+1$ multiplications/divisions and $n-j$ additions. Therefore, the Back Substitution phase of the algorithm requires

$$\begin{aligned} \sum_{j=1}^{n} (n-j+1) &= \frac{n^2+n}{2} && \text{multiplications, along with} \\ \sum_{j=1}^{n} (n-j) &= \frac{n^2-n}{2} && \text{additions.} \end{aligned} \tag{4.43}$$

For n large, both of these are approximately equal to $\frac{1}{2} n^2$. Comparing the counts, we conclude that the bulk of the computational effort goes into the reduction of the coefficient matrix to upper triangular form.

Combining the two counts (4.42–43), we discover that, once we have computed the $A = LU$ decomposition of the coefficient matrix, the Forward and Back Substitution process requires n^2 multiplications and $n^2 - n$ additions to solve a linear system $A\mathbf{x} = \mathbf{b}$. This is exactly the *same* as the number of multiplications and additions needed to compute the product $A^{-1}\mathbf{b}$. Thus, even if we happen to know the inverse of A, it is still *just as efficient* to use Forward and Back Substitution to compute the solution!

On the other hand, the computation of A^{-1} is decidedly more inefficient. There are two possible strategies. First, we can solve the n linear systems

$$A\mathbf{x} = \mathbf{e}_i, \qquad i = 1, \ldots, n, \tag{4.44}$$

for the individual columns of A^{-1}. This requires first computing the LU decomposition, which uses about $\frac{1}{3} n^3$ multiplications and a similar number of additions, followed by applying Forward and Back Substitution to each of the systems, using $n \cdot n^2 = n^3$ multiplications and $n(n^2 - n) \approx n^3$ additions, for a grand total of about $\frac{4}{3} n^3$ operations of each type in order to compute A^{-1}. Gauss–Jordan Elimination fares no better (in fact, slightly worse), also requiring about the same number, $\frac{4}{3}n^3$, of each type of arithmetic operation. Both algorithms can be made more efficient by exploiting the fact that there are lots of zeros on the right hand sides of the systems (4.44). Designing the algorithm to avoid adding or subtracting a preordained 0, or multiplying or dividing by a preordained ± 1, reduces the total number of operations required to compute A^{-1} to exactly n^3 multiplications and $n(n-1)^2 \approx n^3$ additions. And don't forget we still need to multiply $A^{-1}\mathbf{b}$ to solve the original system. As a result, solving a linear system with the inverse matrix requires approximately *three* times as many arithmetic operations, and so would take three times as long to complete, as the more elementary Gaussian Elimination and Back Substitution algorithm. This justifies our earlier contention that matrix inversion is inefficient, and, except in very special situations, should never be used for solving linear systems in practice.

Tridiagonal Matrices

Of course, in special cases, the actual arithmetic operation count might be considerably reduced, particularly if A is a sparse matrix with many zero entries. A number of specialized techniques have been designed to handle sparse linear systems. A particularly important class are the *tridiagonal matrices*

$$A = \begin{pmatrix} q_1 & r_1 & & & & & \\ p_1 & q_2 & r_2 & & & & \\ & p_2 & q_3 & r_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & p_{n-2} & q_{n-1} & r_{n-1} \\ & & & & p_{n-1} & q_n \end{pmatrix} \tag{4.45}$$

with all entries zero except for those on the main diagonal, namely $a_{ii} = q_i$, the *subdiagonal*, meaning the $n-1$ entries $a_{i+1,i} = p_i$ immediately below the main diagonal, and the *superdiagonal*, meaning the entries $a_{i,i+1} = r_i$ immediately above the main diagonal. (Blank entries indicate a 0.) Such matrices arise in the numerical solution of ordinary differential equations and the spline fitting of curves for interpolation and computer graphics. If $A = LU$ is regular, it turns out that the factors are lower and upper *bidiagonal matrices*, of the form

$$L = \begin{pmatrix} 1 & & & & & \\ l_1 & 1 & & & & \\ & l_2 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & l_{n-2} & 1 & \\ & & & & l_{n-1} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} d_1 & u_1 & & & & \\ & d_2 & u_2 & & & \\ & & d_3 & u_3 & & \\ & & & \ddots & \ddots & \\ & & & & d_{n-1} & u_{n-1} \\ & & & & & d_n \end{pmatrix}. \tag{4.46}$$

Multiplying out LU and equating the result to A leads to the equations

$$\begin{array}{lll}
d_1 = q_1, & u_1 = r_1, & l_1 d_1 = p_1, \\
l_1 u_1 + d_2 = q_2, & u_2 = r_2, & l_2 d_2 = p_2, \\
\quad\vdots & \quad\vdots & \quad\vdots \\
l_{j-1} u_{j-1} + d_j = q_j, & u_j = r_j, & l_j d_j = p_j, \\
\quad\vdots & \quad\vdots & \quad\vdots \\
l_{n-2} u_{n-2} + d_{n-1} = q_{n-1}, & u_{n-1} = r_{n-1}, & l_{n-1} d_{n-1} = p_{n-1}, \\
l_{n-1} u_{n-1} + d_n = q_n. & &
\end{array} \tag{4.47}$$

These elementary algebraic equations can be successively solved for the entries of L and U in the following order: $d_1, u_1, l_1, d_2, u_2, l_2, d_3, u_3 \ldots$. The original matrix A is regular provided none of the diagonal entries $d_1, d_2, \ldots$ are zero, which allows the recursive procedure to successfully proceed to termination.

Once the LU factors are in place, we can apply Forward and Back Substitution to solve the tridiagonal linear system $A\mathbf{x} = \mathbf{b}$. We first solve the lower triangular system $L\mathbf{c} = \mathbf{b}$ by Forward Substitution, which leads to the recursive equations

$$c_1 = b_1, \qquad c_2 = b_2 - l_1 c_1, \qquad \ldots \qquad c_n = b_n - l_{n-1} c_{n-1}. \tag{4.48}$$

We then solve the upper triangular system $U\mathbf{x} = \mathbf{c}$ by Back Substitution, again recursively:

$$x_n = \frac{c_n}{d_n}, \qquad x_{n-1} = \frac{c_{n-1} - u_{n-1} x_n}{d_{n-1}}, \qquad \ldots \qquad x_1 = \frac{c_1 - u_1 x_2}{d_1}. \tag{4.49}$$

As you can check, there are a total of $5n - 4$ multiplications/divisions and $3n - 3$ additions/subtractions required to solve a general tridiagonal system of n linear equations — a striking improvement over the general case.

Example 4.26. Consider the $n \times n$ tridiagonal matrix

$$A = \begin{pmatrix}
4 & 1 & & & & & \\
1 & 4 & 1 & & & & \\
 & 1 & 4 & 1 & & & \\
 & & 1 & 4 & 1 & & \\
 & & & \ddots & \ddots & \ddots & \\
 & & & & 1 & 4 & 1 \\
 & & & & & 1 & 4
\end{pmatrix}$$

in which the diagonal entries are all $q_i = 4$, while the entries immediately above and below the main diagonal are all $p_i = r_i = 1$. According to (4.47), the tridiagonal factorization (4.46) has $u_1 = u_2 = \ldots = u_{n-1} = 1$, while

$$d_1 = 4, \qquad l_j = 1/d_j, \qquad d_{j+1} = 4 - l_j, \qquad j = 1, 2, \ldots, n-1.$$

The computed values are

j	1	2	3	4	5	6	7
d_j	4.0	3.75	3.733333	3.732143	3.732057	3.732051	3.732051
l_j	.25	.266666	.267857	.267942	.267948	.267949	.267949

These converge rapidly to

$$d_j \longrightarrow 2+\sqrt{3} = 3.732051\ldots, \qquad l_j \longrightarrow 2-\sqrt{3} = .267949\ldots,$$

which makes the factorization for large n almost trivial. The numbers $2\pm\sqrt{3}$ are the roots of the quadratic equation $x^2-4x+1=0$, and are characterized as the fixed points of the nonlinear iterative system $d_{j+1} = 4 - 1/d_j$.

Pivoting Strategies

Let us now investigate the practical side of pivoting. As we know, in the irregular situations when a zero shows up in a diagonal pivot position, a row interchange is required to proceed with the elimination algorithm. But even when a nonzero pivot element is in place, there may be good numerical reasons for exchanging rows in order to install a more desirable element in the pivot position. Here is a simple example:

$$.01\,x + 1.6\,y = 32.1, \qquad x + .6\,y = 22. \tag{4.50}$$

The exact solution to the system is easily found:

$$x = 10, \qquad y = 20.$$

Suppose we are working with a very primitive calculator that only retains 3 digits of accuracy. (Of course, this is not a very realistic situation, but the example could be suitably modified to produce similar difficulties no matter how many digits of accuracy our computer retains.) The augmented matrix is

$$\left(\begin{array}{cc|c} .01 & 1.6 & 32.1 \\ 1 & .6 & 22 \end{array}\right).$$

Choosing the $(1,1)$ entry as our pivot, and subtracting 100 times the first row from the second produces the upper triangular form

$$\left(\begin{array}{cc|c} .01 & 1.6 & 32.1 \\ 0 & -159.4 & -3188 \end{array}\right).$$

Since our calculator has only three–place accuracy, it will round the entries in the second row, producing the augmented coefficient matrix

$$\left(\begin{array}{cc|c} .01 & 1.6 & 32.1 \\ 0 & -159.0 & -3190 \end{array}\right).$$

The solution by Back Substitution gives

$$y = 3190/159 = 20.0628\ldots \simeq 20.1, \qquad \text{and then}$$
$$x = 100\,(32.1 - 1.6\,y) = 100\,(32.1 - 32.16) \simeq 100\,(32.1 - 32.2) = -10.$$

Gaussian Elimination With Partial Pivoting

```
start
    for $i = 1$ to $n$
        set $\rho(i) = i$
    next $i$
    for $j = 1$ to $n$
        if $m_{\rho(i),j} = 0$ for all $i \geq j$, stop; print "$A$ is singular"
        choose $i > j$ such that $m_{\rho(i),j}$ is maximal
        interchange $\rho(i) \longleftrightarrow \rho(j)$
        for $i = j + 1$ to $n$
            set $l_{\rho(i)j} = m_{\rho(i)j} / m_{\rho(j)j}$
            for $k = j + 1$ to $n + 1$
                set $m_{\rho(i)k} = m_{\rho(i)k} - l_{\rho(i)j} m_{\rho(j)k}$
            next $k$
        next $i$
    next $j$
end
```

The relatively small error in y has produced a very large error in x — not even its sign is correct!

The problem is that the first pivot, .01, is much smaller than the other element, 1, that appears in the column below it. Interchanging the two rows before performing the row operation would resolve the difficulty — even with such an inaccurate calculator! After the interchange, we have

$$\left(\begin{array}{cc|c} 1 & .6 & 22 \\ .01 & 1.6 & 32.1 \end{array}\right),$$

which results in the rounded-off upper triangular form

$$\left(\begin{array}{cc|c} 1 & .6 & 22 \\ 0 & 1.594 & 31.88 \end{array}\right) \quad \simeq \quad \left(\begin{array}{cc|c} 1 & .6 & 22 \\ 0 & 1.59 & 31.9 \end{array}\right).$$

The solution by Back Substitution now gives a respectable answer:

$$y = 31.9/1.59 = 20.0628\ldots \ \simeq\ 20.1, \qquad x = 22 - .6\,y = 22 - 12.06 \ \simeq\ 22 - 12.1 = 9.9.$$

The general strategy, known as *Partial Pivoting*, says that at each stage, we should use the largest (in absolute value) legitimate (i.e., in the pivot column on or below the diagonal) element as the pivot, even if the diagonal element is nonzero. Partial pivoting can help suppress the undesirable effects of round-off errors during the computation.

In a computer implementation of pivoting, there is no need to waste processor time physically exchanging the row entries in memory. Rather, one introduces a separate array of pointers that serve to indicate which original row is currently in which permuted position. More concretely, one initializes n row pointers $\rho(1) = 1, \ldots, \rho(n) = n$. Interchanging row i and row j of the coefficient or augmented matrix is then accomplished by merely interchanging $\rho(i)$ and $\rho(j)$. Thus, to access a matrix element that is currently in row i of the augmented matrix, one merely retrieves the element that is in row $\rho(i)$ in the computer's memory. An explicit implementation of this strategy is provided in the accompanying pseudocode program.

Partial pivoting will solve most problems, although there can still be difficulties. For instance, it does not accurately solve the system

$$10\,x + 1600\,y = 3210, \qquad x + .6\,y = 22,$$

obtained by multiplying the first equation in (4.50) by 1000. The tip-off is that, while the entries in the column containing the pivot are smaller, those in its row are much larger. The solution to this difficulty is *Full Pivoting*, in which one also performs column interchanges — preferably with a column pointer — to move the largest legitimate element into the pivot position. In practice, a column interchange amounts to reordering the variables in the system, which, as long as one keeps proper track of the order, also doesn't change the solutions. Thus, switching the order of x, y leads to the augmented matrix $\left(\begin{array}{cc|c} 1600 & 10 & 3210 \\ .6 & 1 & 22 \end{array}\right)$ in which the first column now refers to y and the second to x. Now Gaussian Elimination will produce a reasonably accurate solution to the system.

Finally, there are some matrices that are hard to handle even with sophisticated pivoting strategies. Such *ill-conditioned* matrices are typically characterized by being "almost" singular. A famous example of an ill-conditioned matrix is the $n \times n$ *Hilbert matrix*

$$H_n = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \cdots & \frac{1}{n+2} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \cdots & \frac{1}{n+3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \frac{1}{n+3} & \cdots & \frac{1}{2n-1} \end{pmatrix}. \tag{4.51}$$

It can be shown that H_n is nonsingular for all n. However, the solution of a linear system whose coefficient matrix is a Hilbert matrix H_n, even for moderately large n, is a very challenging problem, even using high precision computer arithmetic. This is because the larger n is, the closer H_n is, in a sense, to being singular.

The reader is urged to try the following computer experiment. Fix a moderately large value of n, say 20. Choose a column vector $\mathbf{x}$ with n entries chosen at random. Compute

$\mathbf{b} = H_n \mathbf{x}$ directly. Then try to solve the system $H_n \mathbf{x} = \mathbf{b}$ by Gaussian Elimination, and compare the result with the original vector $\mathbf{x}$. If you obtain an accurate solution with $n = 20$, try $n = 50$ or 100. This will give you a good indicator of the degree of arithmetic precision used by your computer hardware, and the accuracy of the numerical solution algorithm(s) in your software.