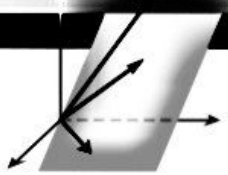
CHAPTER 9



Numerical Methods

CHAPTER CONTENTS

INTRODUCTION

This chapter is concerned with "numerical methods" of linear algebra, an area of study that encompasses techniques for solving large-scale linear systems and for finding numerical approximations of various kinds. It is not our objective to discuss algorithms and technical issues in fine detail since there are many excellent books on the subject. Rather, we will be concerned with introducing some of the basic ideas and exploring two important contemporary applications that rely heavily on numerical ideas—singular value decomposition and data compression. A computing utility such as MATLAB, *Mathematica*, or Maple is recommended for Sections 9.2 to 9.5.

9.1 *LU*-Decompositions

Up to now, we have focused on two methods for solving linear systems, Gaussian elimination (reduction to row echelon form) and Gauss–Jordan elimination (reduction to reduced row echelon form). While these methods are fine for the small-scale problems in this text, they are not suitable for large-scale problems in which computer roundoff error, memory usage, and speed are concerns. In this section we will discuss a method for solving a linear system of n equations in n unknowns that is based on factoring its coefficient matrix into a product of lower and upper triangular matrices. This method, called "LU-decomposition," is the basis for many computer algorithms in common use.

Solving Linear Systems by Factoring

Our first goal in this section is to show how to solve a linear system $A\mathbf{x} = \mathbf{b}$ of n equations in n unknowns by factoring the coefficient matrix A. We begin with some terminology.

DEFINITION 1 A factorization of a square matrix A as

$$A = LU \tag{1}$$

where L is lower triangular and U is upper triangular, is called an ***LU*-decomposition** (or ***LU*-factorization**) of A.

Before we consider the problem of obtaining an LU-decomposition, we will explain how such decompositions can be used to solve linear systems, and we will give an illustrative example.

The Method of LU-Decomposition

Step 1. Rewrite the system $A\mathbf{x} = \mathbf{b}$ as

$$LU\mathbf{x} = \mathbf{b} \tag{2}$$

Step 2. Define a new $n \times 1$ matrix $\mathbf{y}$ by

$$U\mathbf{x} = \mathbf{y} \tag{3}$$

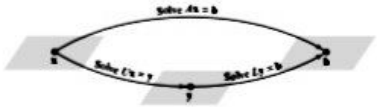
Step 3. Use (3) to rewrite (2) as $L\mathbf{y} = \mathbf{b}$ and solve this system for $\mathbf{y}$.

Step 4. Substitute $\mathbf{y}$ in (3) and solve for $\mathbf{x}$.

This procedure, which is illustrated in Figure 9.1.1, replaces the single linear system $A\mathbf{x} = \mathbf{b}$ by a pair of linear systems

$$U\mathbf{x} = \mathbf{y}$$
$$L\mathbf{y} = \mathbf{b}$$

that must be solved in succession. However, since each of these systems has a triangular coefficient matrix, it generally turns out to involve no more computation to solve the two systems than to solve the original system directly.



▶ Figure 9.1.1

▶ **EXAMPLE 1 Solving $A\mathbf{x} = \mathbf{b}$ by *LU*-Decomposition**

Later in this section we will derive the factorization

$$\underbrace{\begin{bmatrix} 2 & 6 & 2 \\ -3 & -8 & 0 \\ 4 & 9 & 2 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 2 & 0 & 0 \\ -3 & 1 & 0 \\ 4 & -3 & 7 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}}_{U} \tag{4}$$

Use this result to solve the linear system

$$\underbrace{\begin{bmatrix} 2 & 6 & 2 \\ -3 & -8 & 0 \\ 4 & 9 & 2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix}}_{\mathbf{b}}$$

From (4) we can rewrite this system as

$$\underbrace{\begin{bmatrix} 2 & 0 & 0 \\ -3 & 1 & 0 \\ 4 & -3 & 7 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix}}_{\mathbf{b}} \tag{5}$$

Historical Note In 1979 an important library of machine-independent linear algebra programs called LINPACK was developed at Argonne National Laboratories. Many of the programs in that library use the decomposition methods that we will study in this section. Variations of the LINPACK routines are used in many computer programs, including MATLAB, Mathematica, and Maple.

As specified in Step 2 above, let us define y_1, y_2, and y_3 by the equation

$$\underbrace{\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}}_{\mathbf{y}} \tag{6}$$

which allows us to rewrite (5) as

$$\underbrace{\begin{bmatrix} 2 & 0 & 0 \\ -3 & 1 & 0 \\ 4 & -3 & 7 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}}_{\mathbf{y}} = \underbrace{\begin{bmatrix} 2 \\ 2 \\ 3 \end{bmatrix}}_{\mathbf{b}} \tag{7}$$

or equivalently as

$$\begin{aligned} 2y_1 &= 2 \\ -3y_1 + y_2 &= 2 \\ 4y_1 - 3y_2 + 7y_3 &= 3 \end{aligned}$$

This system can be solved by a procedure that is similar to back substitution, except that we solve the equations from the top down instead of from the bottom up. This procedure, called ***forward substitution***, yields

$$y_1 = 1, \quad y_2 = 5, \quad y_3 = 2$$

(verify). As indicated in Step 4 above, we substitute these values into (6), which yields the linear system

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ 2 \end{bmatrix}$$

or, equivalently,

$$\begin{aligned} x_1 + 3x_2 + x_3 &= 1 \\ x_2 + 3x_3 &= 5 \\ x_3 &= 2 \end{aligned}$$

Solving this system by back substitution yields

$$x_1 = 2, \quad x_2 = -1, \quad x_3 = 2$$

(verify). ◀

Alan Mathison Turing (1912–1954)

Historical Note Although the ideas were known earlier, credit for popularizing the matrix formulation of the *LU*-decomposition is often given to the British mathematician Alan Turing for his work on the subject in 1948. Turing, one of the great geniuses of the twentieth century, is the founder of the field of artificial intelligence. Among his many accomplishments in that field, he developed the concept of an internally programmed computer before the practical technology had reached the point where the construction of such a machine was possible. During World War II Turing was secretly recruited by the British government's Code and Cypher School at Bletchley Park to help break the Nazi Enigma codes; it was Turing's statistical approach that provided the breakthrough. In addition to being a brilliant mathematician, Turing was a world-class runner who competed successfully with Olympic-level competition. Sadly, Turing, a homosexual, was tried and convicted of "gross indecency" in 1952, in violation of the then-existing British statutes. Depressed, he committed suicide at age 41 by eating an apple laced with cyanide.

[Image: © National Portrait Gallery]

Finding LU-Decompositions

The preceding example illustrates that once an *LU*-decomposition of A is obtained, a linear system $A\mathbf{x} = \mathbf{b}$ can be solved by one forward substitution and one backward substitution. The main advantage of this method over Gaussian and Gauss–Jordan elimination is that it "decouples" A from $\mathbf{b}$ so that for solving a *sequence* of linear systems with the same coefficient matrix A, say

$$A\mathbf{x} = \mathbf{b}_1, \quad A\mathbf{x} = \mathbf{b}_2, \ldots, \quad A\mathbf{x} = \mathbf{b}_k$$

the work in factoring A need only be performed once, after which it can be reused for each system in the sequence. Such sequences occur in problems in which the matrix A remains fixed but the matrix $\mathbf{b}$ varies with time.

Not every square matrix has an *LU*-decomposition. However, if it is possible to reduce a square matrix A to row echelon form by Gaussian elimination *without performing any row interchanges*, then A will have an *LU*-decomposition, though it may not be unique. To see why this is so, assume that A has been reduced to a row echelon form U using a sequence of row operations that does not include row interchanges. We know from Theorem 1.5.1 that these operations can be accomplished by multiplying A on the left by an appropriate sequence of elementary matrices, that is, there exist elementary matrices $E_1, E_2, \ldots, E_k$ such that

$$E_k \cdots E_2 E_1 A = U \tag{8}$$

Since elementary matrices are invertible, we can solve (8) for A as

$$A = E_1^{-1} E_2^{-1} \cdots E_k^{-1} U$$

or more briefly as

$$A = LU \tag{9}$$

where

$$L = E_1^{-1} E_2^{-1} \cdots E_k^{-1} \tag{10}$$

We now have all of the ingredients to prove the following result.

THEOREM 9.1.1 *If A is a square matrix that can be reduced to a row echelon form U by Gaussian elimination without row interchanges, then A can be factored as $A = LU$, where L is a lower triangular matrix.*

Proof Let L and U be the matrices in Formulas (10) and (8), respectively. The matrix U is upper triangular because it is a row echelon form of a square matrix (so all entries below its main diagonal are zero). To prove that L is lower triangular, it suffices to prove that each factor on the right side of (10) is lower triangular, since Theorem 1.7.1(*b*) will then imply that L itself is lower triangular. Since row interchanges are excluded, each E_j results either by adding a scalar multiple of one row of an identity matrix to a row below or by multiplying one row of an identity matrix by a nonzero scalar. In either case, the resulting matrix E_j is lower triangular and hence so is E_j^{-1} by Theorem 1.7.1(*d*). This completes the proof. ◀

▶ EXAMPLE 2 **An *LU*-Decomposition**

Find an *LU*-decomposition of

$$A = \begin{bmatrix} 2 & 6 & 2 \\ -3 & -8 & 0 \\ 4 & 9 & 2 \end{bmatrix}$$

Solution To obtain an LU-decomposition, $A = LU$, we will reduce A to a row echelon form U using Gaussian elimination and then calculate L from (10). The steps are as follows:

	Reduction to Row Echelon Form	Row Operation	Elementary Matrix Corresponding to the Row Operation	Inverse of the Elementary Matrix
	$\begin{bmatrix} 2 & 6 & 2 \\ -3 & -8 & 0 \\ 4 & 9 & 2 \end{bmatrix}$			
Step 1	$\begin{bmatrix} 1 & 3 & 1 \\ -3 & -8 & 0 \\ 4 & 9 & 2 \end{bmatrix}$	$\frac{1}{2} \times$ row 1	$E_1 = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$E_1^{-1} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Step 2	$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 3 \\ 4 & 9 & 2 \end{bmatrix}$	(3 × row 1) + row 2	$E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$E_2^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Step 3	$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 3 \\ 0 & -3 & -2 \end{bmatrix}$	(−4 × row 1) + row 3	$E_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix}$	$E_3^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix}$
Step 4	$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 7 \end{bmatrix}$	(3 × row 2) + row 3	$E_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 1 \end{bmatrix}$	$E_4^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix}$
Step 5	$\begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} = U$	$\frac{1}{7} \times$ row 3	$E_5 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{7} \end{bmatrix}$	$E_5^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 7 \end{bmatrix}$

and, from (10),

$$L = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 7 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 0 \\ -3 & 1 & 0 \\ 4 & -3 & 7 \end{bmatrix} \tag{11}$$

so

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

is an LU-decomposition of A. ◀

Bookkeeping As Example 2 shows, most of the work in constructing an LU-decomposition is expended in calculating L. However, *all* this work can be eliminated by some careful bookkeeping of the operations used to reduce A to U.

Because we are assuming that no row interchanges are required to reduce A to U, there are only two types of operations involved—multiplying a row by a nonzero constant, and adding a scalar multiple of one row to another. The first operation is used to introduce the leading 1's and the second to introduce zeros below the leading 1's.

In Example 2, a multiplier of $\frac{1}{2}$ was needed in Step 1 to introduce a leading 1 in the first row, and a multiplier of $\frac{1}{7}$ was needed in Step 5 to introduce a leading 1 in the third row. No actual multiplier was required to introduce a leading 1 in the second row because it was already a 1 at the end of Step 2, but for convenience let us say that the multiplier was 1. Comparing these multipliers with the successive diagonal entries of L, we see that these diagonal entries are precisely the reciprocals of the multipliers used to construct U:

$$L = \begin{bmatrix} ② & 0 & 0 \\ -3 & ① & 0 \\ 4 & -3 & ⑦ \end{bmatrix} \tag{12}$$

Also observe in Example 2 that to introduce zeros below the leading 1 in the first row, we used the operations

add 3 times the first row to the second

add −4 times the first row to the third

and to introduce the zero below the leading 1 in the second row, we used the operation

add 3 times the second row to the third

Now note in (11) that in each position below the main diagonal of L, the entry is the *negative* of the multiplier in the operation that introduced the zero in that position in U. This suggests the following procedure for constructing an LU-decomposition of a square matrix A, assuming that this matrix can be reduced to row echelon form without row interchanges.

Procedure for Constructing an LU-Decomposition

Step 1. Reduce A to a row echelon form U by Gaussian elimination without row interchanges, keeping track of the multipliers used to introduce the leading 1's and the multipliers used to introduce the zeros below the leading 1's.

Step 2. In each position along the main diagonal of L, place the reciprocal of the multiplier that introduced the leading 1 in that position in U.

Step 3. In each position below the main diagonal of L, place the negative of the multiplier used to introduce the zero in that position in U.

Step 4. Form the decomposition $A = LU$.

▶ **EXAMPLE 3 Constructing an LU-Decomposition**

Find an LU-decomposition of

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

Solution We will reduce A to a row echelon form U and at each step we will fill in an entry of L in accordance with the four-step procedure above.

$$A = \begin{bmatrix} 6 & -2 & 0 \\ 9 & -1 & 1 \\ 3 & 7 & 5 \end{bmatrix} \qquad \begin{bmatrix} \bullet & 0 & 0 \\ \bullet & \bullet & 0 \\ \bullet & \bullet & \bullet \end{bmatrix}$$

• denotes an unknown entry of L.

$$\begin{bmatrix} \textcircled{1} & -\frac{1}{3} & 0 \\ 9 & -1 & 1 \\ 3 & 7 & 5 \end{bmatrix} \leftarrow \text{multiplier} = \frac{1}{6} \qquad \begin{bmatrix} 6 & 0 & 0 \\ \bullet & \bullet & 0 \\ \bullet & \bullet & \bullet \end{bmatrix}$$

$$\begin{bmatrix} 1 & -\frac{1}{3} & 0 \\ \textcircled{0} & 2 & 1 \\ \textcircled{0} & 8 & 5 \end{bmatrix} \begin{matrix} \\ \leftarrow \text{multiplier} = -9 \\ \leftarrow \text{multiplier} = -3 \end{matrix} \qquad \begin{bmatrix} 6 & 0 & 0 \\ 9 & \bullet & 0 \\ 3 & \bullet & \bullet \end{bmatrix}$$

$$\begin{bmatrix} 1 & -\frac{1}{3} & 0 \\ 0 & \textcircled{1} & \frac{1}{2} \\ 0 & 8 & 5 \end{bmatrix} \leftarrow \text{multiplier} = \frac{1}{2} \qquad \begin{bmatrix} 6 & 0 & 0 \\ 9 & 2 & 0 \\ 3 & \bullet & \bullet \end{bmatrix}$$

$$\begin{bmatrix} 1 & -\frac{1}{3} & 0 \\ 0 & 1 & \frac{1}{2} \\ 0 & \textcircled{0} & 1 \end{bmatrix} \leftarrow \text{multiplier} = -8 \qquad \begin{bmatrix} 6 & 0 & 0 \\ 9 & 2 & 0 \\ 3 & 8 & \bullet \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & -\frac{1}{3} & 0 \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & \textcircled{1} \end{bmatrix} \leftarrow \text{multiplier} = 1 \qquad L = \begin{bmatrix} 6 & 0 & 0 \\ 9 & 2 & 0 \\ 3 & 8 & 1 \end{bmatrix}$$

No actual operation is performed here since there is already a leading 1 in the third row.

Thus, we have constructed the LU-decomposition

$$A = LU = \begin{bmatrix} 6 & 0 & 0 \\ 9 & 2 & 0 \\ 3 & 8 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{3} & 0 \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

We leave it for you to confirm this end result by multiplying the factors. ◀

LU-Decompositions Are Not Unique

In general, LU-decompositions are not unique. For example, if

$$A = LU = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

and L has nonzero diagonal entries (which will be true if A is invertible), then we can shift the diagonal entries from the left factor to the right factor by writing

$$A = \begin{bmatrix} 1 & 0 & 0 \\ l_{21}/l_{11} & 1 & 0 \\ l_{31}/l_{11} & l_{32}/l_{22} & 1 \end{bmatrix} \begin{bmatrix} l_{11} & 0 & 0 \\ 0 & l_{22} & 0 \\ 0 & 0 & l_{33} \end{bmatrix} \begin{bmatrix} 1 & u_{12} \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ l_{21}/l_{11} & 1 & 0 \\ l_{31}/l_{11} & l_{32}/l_{22} & 1 \end{bmatrix} \begin{bmatrix} l_{11} & l_{11}u_{12} & l_{11}u_{13} \\ 0 & l_{22} & l_{22}u_{23} \\ 0 & 0 & l_{33} \end{bmatrix}$$

which is another LU-decomposition of A.

LDU-Decompositions

The method we have given for computing LU-decompositions may result in an "asymmetry" in that the matrix U has 1's on the main diagonal but L need not. However, if it is preferred to have 1's on the main diagonal of both the lower triangular factor and the upper triangular factor, then we can "shift" the diagonal entries of L to a diagonal matrix D and write L as

$$L = L'D$$

where L' is a lower triangular matrix with 1's on the main diagonal. For example, a general 3×3 lower triangular matrix with nonzero entries on the main diagonal can be factored as

$$\underbrace{\begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix}}_{L} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ a_{21}/a_{11} & 1 & 0 \\ a_{31}/a_{11} & a_{32}/a_{22} & 1 \end{bmatrix}}_{L'} \underbrace{\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}}_{D}$$

Note that the columns of L' are obtained by dividing each entry in the corresponding column of L by the diagonal entry in the column. Thus, for example, we can rewrite (4) as

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

$$= \begin{bmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 1 & 0 \\ 2 & -3 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 7 \end{bmatrix} \begin{bmatrix} 1 & 3 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \qquad (13)$$

If desired, the diagonal matrix and the upper triangular matrix in (13) can be multiplied to produce an LU-decomposition in which the 1's are on the main diagonal of L rather than U.

One can prove that if A is an invertible matrix that can be reduced to row echelon form without row interchanges, then A can be factored *uniquely* as

$$A = LDU$$

where L is a lower triangular matrix with 1's on the main diagonal, D is a diagonal matrix, and U is an upper triangular matrix with 1's on the main diagonal. This is called the ***LDU-decomposition*** (or ***LDU-factorization***) of A.

PLU-Decompositions Many computer algorithms for solving linear systems perform row interchanges to reduce roundoff error, in which case the existence of an LU-decomposition is not guaranteed. However, it is possible to work around this problem by "preprocessing" the coefficient matrix A so that the row interchanges are performed *prior* to computing the LU-decomposition itself. More specifically, the idea is to create a matrix Q (called a ***permutation matrix***) by multiplying, in sequence, those elementary matrices that produce the row interchanges and then execute them by computing the product QA. This product can then be reduced to row echelon form *without* row interchanges, so it is assured to have an LU-decomposition

$$QA = LU \tag{14}$$

Because the matrix Q is invertible (being a product of elementary matrices), the systems $A\mathbf{x} = \mathbf{b}$ and $QA\mathbf{x} = Q\mathbf{b}$ will have the same solutions. But it follows from (14) that the latter system can be rewritten as $LU\mathbf{x} = Q\mathbf{b}$ and hence can be solved using LU-decomposition.

It is common to see Equation (14) expressed as

$$A = PLU \tag{15}$$

in which $P = Q^{-1}$. This is called a ***PLU-decomposition*** or (***PLU-factorization***) of A.

Exercise Set 9.1

1. Use the method of Example 1 and the LU-decomposition

$$\begin{bmatrix} 3 & -6 \\ -2 & 5 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}$$

to solve the system

$$\begin{aligned} 3x_1 - 6x_2 &= 0 \\ -2x_1 + 5x_2 &= 1 \end{aligned}$$

2. Use the method of Example 1 and the LU-decomposition

$$\begin{bmatrix} 3 & -6 & -3 \\ 2 & 0 & 6 \\ -4 & 7 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 \\ 2 & 4 & 0 \\ -4 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & -2 & -1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

to solve the system

$$\begin{aligned} 3x_1 - 6x_2 - 3x_3 &= -3 \\ 2x_1 + 6x_3 &= -22 \\ -4x_1 + 7x_2 + 4x_3 &= 3 \end{aligned}$$

▶ In Exercises 3–6, find an LU-decomposition of the coefficient matrix, and then use the method of Example 1 to solve the system. ◀

3. $\begin{bmatrix} 2 & 8 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$

4. $\begin{bmatrix} -5 & -10 \\ 6 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -10 \\ 19 \end{bmatrix}$

5. $\begin{bmatrix} 2 & -2 & -2 \\ 0 & -2 & 2 \\ -1 & 5 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -4 \\ -2 \\ 6 \end{bmatrix}$

6. $\begin{bmatrix} -3 & 12 & -6 \\ 1 & -2 & 2 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -33 \\ 7 \\ -1 \end{bmatrix}$

▶ In Exercises 7–8, an LU-decomposition of a matrix A is given.
(a) Compute L^{-1} and U^{-1}.
(b) Use the result in part (a) to find the inverse of A. ◀

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

$$A = LU = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 3 \\ 0 & 4 & -5 \\ 0 & 0 & 6 \end{bmatrix}$$

8. The LU-decomposition obtained in Example 2

9. Let

$$A = \begin{bmatrix} 2 & 1 & -1 \\ -2 & -1 & 2 \\ 2 & 1 & 0 \end{bmatrix}$$

(a) Find an LU-decomposition of A.
(b) Express A in the form $A = L_1DU_1$, where L_1 is lower triangular with 1's along the main diagonal, U_1 is upper triangular, and D is a diagonal matrix.
(c) Express A in the form $A = L_2U_2$, where L_2 is lower triangular with 1's along the main diagonal and U_2 is upper triangular.

10. (a) Show that the matrix

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

has no LU-decomposition.
(b) Find a PLU-decomposition of this matrix.

▶ In Exercises 11–12, use the given PLU-decomposition of A to solve the linear system $A\mathbf{x} = \mathbf{b}$ by rewriting it as $P^{-1}A\mathbf{x} = P^{-1}\mathbf{b}$ and solving this system by LU-decomposition. ◀

11. $\mathbf{b} = \begin{bmatrix} 2 \\ 1 \\ 5 \end{bmatrix}$; $A = \begin{bmatrix} 0 & 1 & 4 \\ 1 & 2 & 2 \\ 3 & 1 & 3 \end{bmatrix}$;

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & -5 & 17 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} = PLU$$

12. $\mathbf{b} = \begin{bmatrix} 3 \\ 0 \\ 6 \end{bmatrix}$; $A = \begin{bmatrix} 4 & 1 & 2 \\ 0 & 2 & 1 \\ 8 & 1 & 8 \end{bmatrix}$;

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 4 & 1 & 2 \\ 0 & -1 & 4 \\ 0 & 0 & 9 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{4} & \frac{1}{2} \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix} = PLU$$

▶ In Exercises 13–14, find the LDU-decomposition of A. ◀

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

14. $A = \begin{bmatrix} 3 & -12 & 6 \\ 0 & 2 & 0 \\ 6 & -28 & 13 \end{bmatrix}$

▶ In Exercises 15–16, find a PLU-decomposition of A, and use it to solve the linear system $A\mathbf{x} = \mathbf{b}$ by the method of Exercises 11 and 12. ◀

15. $A = \begin{bmatrix} 3 & -1 & 0 \\ 3 & -1 & 1 \\ 0 & 2 & 1 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} -2 \\ 1 \\ 4 \end{bmatrix}$

16. $A = \begin{bmatrix} 0 & 3 & -2 \\ 1 & 1 & 4 \\ 2 & 2 & 5 \end{bmatrix}$; $\mathbf{b} = \begin{bmatrix} 7 \\ 5 \\ -2 \end{bmatrix}$

17. Let $A\mathbf{x} = \mathbf{b}$ be a linear system of n equations in n unknowns, and assume that A is an invertible matrix that can be reduced to row echelon form without row interchanges. How many additions and multiplications are required to solve the system by the method of Example 1?

Working with Proofs

18. Let

$$A = \begin{bmatrix} a & b \\ c & \end{bmatrix}$$

(a) Prove: If $a \neq 0$, then the ma… decomposition with 1's along th…
(b) Find the LU-decomposition described in part (a).

19. Prove: If A is any $n \times n$ matrix, then A can be factored as $A = PLU$, where L is lower triangular, U is upper triangular, and P can be obtained by interchanging the rows of I_n appropriately. [*Hint*: Let U be a row echelon form of A, and let all row interchanges required in the reduction of A to U be performed first.]

True-False Exercises

TF. In parts (a)–(e) determine whether the statement is true or false, and justify your answer.

(a) Every square matrix has an LU-decomposition.

(b) If a square matrix A is row equivalent to an upper triangular matrix U, then A has an LU-decomposition.

(c) If $L_1, L_2, \ldots, L_k$ are $n \times n$ lower triangular matrices, then the product $L_1L_2 \cdots L_k$ is lower triangular.

(d) If an invertible matrix A has an LU-decomposition, then A has a unique LDU-decomposition.

(e) Every square matrix has a PLU-decomposition.

Working with Technology

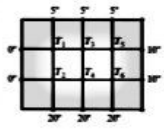
T1. Technology utilities vary in how they handle LU-decompositions. For example, many utilities perform row interchanges to reduce roundoff error and hence produce PLU-decompositions, even when asked for LU-decompositions. See what happens when you use your utility to find an LU-decomposition of the matrix A in Example 2.

T2. The accompanying figure shows a metal plate whose edges are held at the temperatures shown. It follows from thermodynamic principles that the temperature at each of the six interior nodes will eventually stabilize at a value that is approximately the average of the temperatures at the four neighboring nodes. These are called the ***steady-state temperatures*** at the nodes. Thus, for example, if we denote the steady-state temperatures at the interior nodes in

the accompanying figure as T_1, T_2, T_3, T_4, T_5, and T_6, then at the node labeled T_1 that temperature will be $T_1 = \frac{1}{4}(0 + 5 + T_2 + T_3)$ or, equivalently,

$$4T_1 - T_2 - T_3 = 5$$

Find a linear system whose solution gives the steady-state temperatures at the nodes, and use your technology utility to solve that system by LU-decomposition.



◀ Figure Ex-T2

9.2 The Power Method

The eigenvalues of a square matrix can, in theory, be found by solving the characteristic equation. However, this procedure has so many computational difficulties that it is almost never used in applications. In this section we will discuss an algorithm that can be used to approximate the eigenvalue with greatest absolute value and a corresponding eigenvector. This particular eigenvalue and its corresponding eigenvectors are important because they arise naturally in many iterative processes. The methods we will study in this section have recently been used to create Internet search engines such as Google.

The Power Method

There are many applications in which some vector $\mathbf{x}_0$ in R^n is multiplied repeatedly by an $n \times n$ matrix A to produce a sequence

$$\mathbf{x}_0, \quad A\mathbf{x}_0, \quad A^2\mathbf{x}_0, \ldots, \quad A^k\mathbf{x}_0, \ldots$$

We call a sequence of this form a *power sequence generated by A*. In this section we will be concerned with the convergence of power sequences and how such sequences can be used to approximate eigenvalues and eigenvectors. For this purpose, we make the following definition.

DEFINITION 1 **If the *distinct* eigenvalues of a matrix A are $\lambda_1, \lambda_2, \ldots, \lambda_k$, and if $|\lambda_1|$ is larger than $|\lambda_2|, \ldots, |\lambda_k|$, then λ_1 is called a *dominant eigenvalue* of A. Any eigenvector corresponding to a dominant eigenvalue is called a *dominant eigenvector* of A.**

▶ **EXAMPLE 1 Dominant Eigenvalues**

Some matrices have dominant eigenvalues and some do not. For example, if the distinct eigenvalues of a matrix are

$$\lambda_1 = -4, \quad \lambda_2 = -2, \quad \lambda_3 = 1, \quad \lambda_4 = 3$$

then $\lambda_1 = -4$ is dominant since $|\lambda_1| = 4$ is greater than the absolute values of all the other eigenvalues, but if the distinct eigenvalues of a matrix are

$$\lambda_1 = 7, \quad \lambda_2 = -7, \quad \lambda_3 = -2, \quad \lambda_4 = 5$$

then $|\lambda_1| = |\lambda_2| = 7$, so there is no eigenvalue whose absolute value is greater than the absolute value of all the other eigenvalues. ◀

The most important theorems about convergence of power sequences apply to $n \times n$ matrices with n linearly independent eigenvectors (symmetric matrices, for example), so we will limit our discussion to this case in this section.

THEOREM 9.2.1 ***Let A be a symmetric $n \times n$ matrix that has a positive* dominant eigenvalue λ. If $\mathbf{x}_0$ is a unit vector in R^n that is not orthogonal to the eigenspace corresponding to λ, then the normalized power sequence***

$$\mathbf{x}_0, \quad \mathbf{x}_1 = \frac{A\mathbf{x}_0}{\|A\mathbf{x}_0\|}, \quad \mathbf{x}_2 = \frac{A\mathbf{x}_1}{\|A\mathbf{x}_1\|}, \ldots, \quad \mathbf{x}_k = \frac{A\mathbf{x}_{k-1}}{\|A\mathbf{x}_{k-1}\|}, \ldots \tag{1}$$

converges to a unit dominant eigenvector, and the sequence

$$A\mathbf{x}_1 \cdot \mathbf{x}_1, \quad A\mathbf{x}_2 \cdot \mathbf{x}_2, \quad A\mathbf{x}_3 \cdot \mathbf{x}_3, \ldots, \quad A\mathbf{x}_k \cdot \mathbf{x}_k, \ldots \tag{2}$$

converges to the dominant eigenvalue λ.

Remark In the exercises we will ask you to show that (1) can also be

$$\mathbf{x}_0, \quad \mathbf{x}_1 = \frac{A\mathbf{x}_0}{\|A\mathbf{x}_0\|}, \quad \mathbf{x}_2 = \frac{A^2\mathbf{x}_0}{\|A^2\mathbf{x}_0\|}, \ldots, \quad \mathbf{x}_k = \frac{A^k}{\|A^k}$$

This form of the power sequence expresses each iterate in terms of the [illegible] **than in terms of its predecessor.**

516 / 802

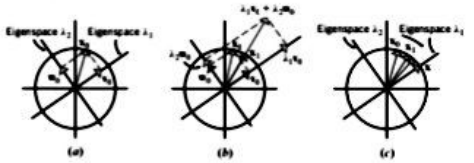
We will not prove Theorem 9.2.1, but we can make it plausible geometrically in the 2×2 case where A is a symmetric matrix with distinct positive eigenvalues, λ_1 and λ_2, one of which is dominant. To be specific, assume that λ_1 is dominant and

$$\lambda_1 > \lambda_2 > 0$$

Since we are assuming that A is symmetric and has distinct eigenvalues, it follows from Theorem 7.2.2 that the eigenspaces corresponding to λ_1 and λ_2 are perpendicular lines through the origin. Thus, the assumption that $\mathbf{x}_0$ is a unit vector that is not orthogonal to the eigenspace corresponding to λ_1 implies that $\mathbf{x}_0$ does not lie in the eigenspace corresponding to λ_2. To see the geometric effect of multiplying $\mathbf{x}_0$ by A, it will be useful to split $\mathbf{x}_0$ into the sum

$$\mathbf{x}_0 = \mathbf{v}_0 + \mathbf{w}_0 \tag{4}$$

where $\mathbf{v}_0$ and $\mathbf{w}_0$ are the orthogonal projections of $\mathbf{x}_0$ on the eigenspaces of λ_1 and λ_2, respectively (Figure 9.2.1*a*).



▶ Figure 9.2.1

This enables us to express $A\mathbf{x}_0$ as

$$A\mathbf{x}_0 = A\mathbf{v}_0 + A\mathbf{w}_0 = \lambda_1\mathbf{v}_0 + \lambda_2\mathbf{w}_0 \tag{5}$$

* If the dominant eigenvalue is not positive, sequence (2) will still converge to the dominant eigenvalue, but sequence (1) may not converge to a specific dominant eigenvector because of *alternation* (see Exercise 11). Nevertheless, each term of (1) will closely approximate some dominant eigenvector for sufficiently large values of k.

which tells us that multiplying $\mathbf{x}_0$ by A "scales" the terms $\mathbf{v}_0$ and $\mathbf{v}_0$ in (4) by λ_1 and λ_2, respectively. However, λ_1 is larger than λ_2, so the scaling is greater in the direction of $\mathbf{v}_0$ than in the direction of $\mathbf{w}_0$. Thus, multiplying $\mathbf{x}_0$ by A "pulls" $\mathbf{x}_0$ toward the eigenspace of λ_1, and normalizing produces a vector $\mathbf{x}_1 = A\mathbf{x}_0/\|A\mathbf{x}_0\|$, which is on the unit circle and is closer to the eigenspace of λ_1 than $\mathbf{x}_0$ (Figure 9.2.1*b*). Similarly, multiplying $\mathbf{x}_1$ by A and normalizing produces a unit vector $\mathbf{x}_2$ that is closer to the eigenspace of λ_1 than $\mathbf{x}_1$. Thus, it seems reasonable that by repeatedly multiplying by A and normalizing we will produce a sequence of vectors $\mathbf{x}_k$ that lie on the unit circle and converge to a unit vector $\mathbf{x}$ in the eigenspace of λ_1 (Figure 9.2.1*c*). Moreover, if $\mathbf{x}_k$ converges to $\mathbf{x}$, then it also seems reasonable that $A\mathbf{x}_k \cdot \mathbf{x}_k$ will converge to

$$A\mathbf{x} \cdot \mathbf{x} = \lambda_1 \mathbf{x} \cdot \mathbf{x} = \lambda_1 \|\mathbf{x}\|^2 = \lambda_1$$

which is the dominant eigenvalue of A.

The Power Method with Euclidean Scaling

Theorem 9.2.1 provides us with an algorithm for approximating the dominant eigenvalue and a corresponding unit eigenvector of a symmetric matrix A, provided the dominant eigenvalue is positive. This algorithm, called the ***power method with Euclidean scaling***, is as follows:

The Power Method with Euclidean Scaling

Step 0. Choose an arbitrary nonzero vector and normalize it, if need be, to obtain a unit vector $\mathbf{x}_0$.

Step 1. Compute $A\mathbf{x}_0$ and normalize it to obtain the first approximation $\mathbf{x}_1$ to a dominant unit eigenvector. Compute $A\mathbf{x}_1 \cdot \mathbf{x}_1$ to obtain the first approximation to the dominant eigenvalue.

Step 2. Compute $A\mathbf{x}_1$ and normalize it to obtain the second approximation $\mathbf{x}_2$ to a dominant unit eigenvector. Compute $A\mathbf{x}_2 \cdot \mathbf{x}_2$ to obtain the second approximation to the dominant eigenvalue.

Step 3. Compute $A\mathbf{x}_2$ and normalize it to obtain the third approximation $\mathbf{x}_3$ to a dominant unit eigenvector. Compute $A\mathbf{x}_3 \cdot \mathbf{x}_3$ to obtain the third approximation to the dominant eigenvalue.

Continuing in this way will usually generate a sequence of better and better approximations to the dominant eigenvalue and a corresponding unit eigenvector.*

▶ **EXAMPLE 2 The Power Method with Euclidean Scaling**

Apply the power method with Euclidean scaling to

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix} \quad \text{with} \quad \mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Stop at $\mathbf{x}_5$ and compare the resulting approximations to the exact values of the dominant eigenvalue and eigenvector.

*If the vector $\mathbf{x}_0$ happens to be orthogonal to the eigenspace of the dominant eigenvalue, then the hypotheses of Theorem 9.2.1 will be violated and the method may fail. However, the reality is that computer roundoff errors usually perturb $\mathbf{x}_0$ enough to destroy any orthogonality and make the algorithm work. This is one instance in which errors help to obtain correct results!

Solution We will leave it for you to show that the eigenvalues of A are $\lambda = 1$ and $\lambda = 5$ and that the eigenspace corresponding to the dominant eigenvalue $\lambda = 5$ is the line represented by the parametric equations $x_1 = t$, $x_2 = t$, which we can write in vector form as

$$\mathbf{x} = t\begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{6}$$

Setting $t = 1/\sqrt{2}$ yields the normalized dominant eigenvector

$$\mathbf{v}_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \approx \begin{bmatrix} 0.707106781187\ldots \\ 0.707106781187\ldots \end{bmatrix} \tag{7}$$

Now let us see what happens when we use the power method, starting with the unit vector $\mathbf{x}_0$.

$$A\mathbf{x}_0 = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \qquad \mathbf{x}_1 = \frac{A\mathbf{x}_0}{\|A\mathbf{x}_0\|} = \frac{1}{\sqrt{13}}\begin{bmatrix} 3 \\ 2 \end{bmatrix} \approx \frac{1}{3.60555}\begin{bmatrix} 3 \\ 2 \end{bmatrix} \approx \begin{bmatrix} 0.83205 \\ 0.55470 \end{bmatrix}$$

$$A\mathbf{x}_1 \approx \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} 0.83205 \\ 0.55470 \end{bmatrix} \approx \begin{bmatrix} 3.60555 \\ 3.32820 \end{bmatrix} \qquad \mathbf{x}_2 = \frac{A\mathbf{x}_1}{\|A\mathbf{x}_1\|} \approx \frac{1}{4.90682}\begin{bmatrix} 3.60555 \\ 3.32820 \end{bmatrix} \approx \begin{bmatrix} 0.73480 \\ 0.67828 \end{bmatrix}$$

$$A\mathbf{x}_2 \approx \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} 0.73480 \\ 0.67828 \end{bmatrix} \approx \begin{bmatrix} 3.56097 \\ 3.50445 \end{bmatrix} \qquad \mathbf{x}_3 = \frac{A\mathbf{x}_2}{\|A\mathbf{x}_2\|} \approx \frac{1}{4.99616}\begin{bmatrix} 3.56097 \\ 3.50445 \end{bmatrix} \approx \begin{bmatrix} 0.71274 \\ 0.70143 \end{bmatrix}$$

$$A\mathbf{x}_3 \approx \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} 0.71274 \\ 0.70143 \end{bmatrix} \approx \begin{bmatrix} 3.54108 \\ 3.52976 \end{bmatrix} \qquad \mathbf{x}_4 = \frac{A\mathbf{x}_3}{\|A\mathbf{x}_3\|} \approx \frac{1}{4.99985}\begin{bmatrix} 3.54108 \\ 3.52976 \end{bmatrix} \approx \begin{bmatrix} 0.70824 \\ 0.70597 \end{bmatrix}$$

$$A\mathbf{x}_4 \approx \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} 0.70824 \\ 0.70597 \end{bmatrix} \approx \begin{bmatrix} 3.53666 \\ 3.53440 \end{bmatrix} \qquad \mathbf{x}_5 = \frac{A\mathbf{x}_4}{\|A\mathbf{x}_4\|} \approx \frac{1}{4.99999}\begin{bmatrix} 3.53666 \\ 3.53440 \end{bmatrix} \approx \begin{bmatrix} 0.70733 \\ 0.70688 \end{bmatrix}$$

$$\lambda^{(1)} = (A\mathbf{x}_1) \cdot \mathbf{x}_1 = (A\mathbf{x}_1)^T\mathbf{x}_1 \approx \begin{bmatrix} 3.60555 & 3.32820 \end{bmatrix}\begin{bmatrix} 0.83205 \\ 0.55470 \end{bmatrix} \approx 4.84615$$

$$\lambda^{(2)} = (A\mathbf{x}_2) \cdot \mathbf{x}_2 = (A\mathbf{x}_2)^T\mathbf{x}_2 \approx \begin{bmatrix} 3.56097 & 3.50445 \end{bmatrix}\begin{bmatrix} 0.73480 \\ 0.67828 \end{bmatrix} \approx 4.99361$$

$$\lambda^{(3)} = (A\mathbf{x}_3) \cdot \mathbf{x}_3 = (A\mathbf{x}_3)^T\mathbf{x}_3 \approx \begin{bmatrix} 3.54108 & 3.52976 \end{bmatrix}\begin{bmatrix} 0.71274 \\ 0.70143 \end{bmatrix} \approx 4.99974$$

$$\lambda^{(4)} = (A\mathbf{x}_4) \cdot \mathbf{x}_4 = (A\mathbf{x}_4)^T\mathbf{x}_4 \approx \begin{bmatrix} 3.53666 & 3.53440 \end{bmatrix}\begin{bmatrix} 0.70824 \\ 0.70597 \end{bmatrix} \approx 4.99999$$

$$\lambda^{(5)} = (A\mathbf{x}_5) \cdot \mathbf{x}_5 = (A\mathbf{x}_5)^T\mathbf{x}_5 \approx \begin{bmatrix} 3.53576 & 3.53531 \end{bmatrix}\begin{bmatrix} 0.70733 \\ 0.70688 \end{bmatrix} \approx 5.00000$$

It is accidental that $\lambda^{(5)}$ (the fifth approximation) produced five decimal place accuracy. In general, n iterations need not produce n decimal place accuracy.

Thus, $\lambda^{(5)}$ approximates the dominant eigenvalue to five decimal place accuracy and $\mathbf{x}_5$ approximates the dominant eigenvector in (7) to three decimal place accuracy. ◀

The Power Method with Maximum Entry Scaling

There is a variation of the power method in which the iterates, rather than being normalized at each stage, are scaled to make the maximum entry 1. To describe this method, it will be convenient to denote the maximum *absolute value* of the entries in a vector $\mathbf{x}$ by

$\max(\mathbf{x})$. Thus, for example, if

$$\mathbf{x} = \begin{bmatrix} 5 \\ 3 \\ -7 \\ 2 \end{bmatrix}$$

then $\max(\mathbf{x}) = 7$. We will need the following variation of Theorem 9.2.1.

THEOREM 9.2.2 *Let A be a symmetric $n \times n$ matrix that has a positive dominant[1] eigenvalue λ. If $\mathbf{x}_0$ is a nonzero vector in $\mathbb{R}^n$ that is not orthogonal to the eigenspace corresponding to λ, then the sequence*

$$\mathbf{x}_0, \quad \mathbf{x}_1 = \frac{A\mathbf{x}_0}{\max(A\mathbf{x}_0)}, \quad \mathbf{x}_2 = \frac{A\mathbf{x}_1}{\max(A\mathbf{x}_1)}, \ldots, \quad \mathbf{x}_k = \frac{A\mathbf{x}_{k-1}}{\max(A\mathbf{x}_{k-1})}, \ldots \tag{8}$$

converges to an eigenvector corresponding to λ, and the sequence

$$\frac{A\mathbf{x}_1 \cdot \mathbf{x}_1}{\mathbf{x}_1 \cdot \mathbf{x}_1}, \quad \frac{A\mathbf{x}_2 \cdot \mathbf{x}_2}{\mathbf{x}_2 \cdot \mathbf{x}_2}, \quad \frac{A\mathbf{x}_3 \cdot \mathbf{x}_3}{\mathbf{x}_3 \cdot \mathbf{x}_3}, \ldots, \quad \frac{A\mathbf{x}_k \cdot \mathbf{x}_k}{\mathbf{x}_k \cdot \mathbf{x}_k}, \ldots \tag{9}$$

converges to λ.

Remark In the exercises we will ask you to show that (8) can be written in the alternative form

$$\mathbf{x}_0, \quad \mathbf{x}_1 = \frac{A\mathbf{x}_0}{\max(A\mathbf{x}_0)}, \quad \mathbf{x}_2 = \frac{A^2\mathbf{x}_0}{\max(A^2\mathbf{x}_0)}, \ldots, \quad \mathbf{x}_k = \frac{A^k\mathbf{x}_0}{\max(A^k\mathbf{x}_0)}, \ldots \tag{10}$$

which expresses the iterates in terms of the initial vector $\mathbf{x}_0$.

We will omit the proof of this theorem, but if we accept that (8) converges to an eigenvector of A, then it is not hard to see why (9) converges to the dominant eigenvalue. To see this, note that each term in (9) is of the form

$$\frac{A\mathbf{x} \cdot \mathbf{x}}{\mathbf{x} \cdot \mathbf{x}} \tag{11}$$

which is called a ***Rayleigh quotient*** of A. In the case where λ is an eigenvalue of A and $\mathbf{x}$ is a corresponding eigenvector, the Rayleigh quotient is

$$\frac{A\mathbf{x} \cdot \mathbf{x}}{\mathbf{x} \cdot \mathbf{x}} = \frac{\lambda\mathbf{x} \cdot \mathbf{x}}{\mathbf{x} \cdot \mathbf{x}} = \frac{\lambda(\mathbf{x} \cdot \mathbf{x})}{\mathbf{x} \cdot \mathbf{x}} = \lambda$$

Thus, if $\mathbf{x}_k$ converges to a dominant eigenvector $\mathbf{x}$, then it seems reasonable that

$$\frac{A\mathbf{x}_k \cdot \mathbf{x}_k}{\mathbf{x}_k \cdot \mathbf{x}_k} \text{ converges to } \frac{A\mathbf{x} \cdot \mathbf{x}}{\mathbf{x} \cdot \mathbf{x}} = \lambda$$

which is the dominant eigenvalue.

Theorem 9.2.2 produces the following algorithm, which is called the ***power method with maximum entry scaling***.

[1] As in Theorem 9.2.1, if the dominant eigenvalue is not positive, sequence (9) will still converge to the dominant eigenvalue, but sequence (8) may not converge to a *specific* dominant eigenvector. Nevertheless, each term of (8) will closely approximate *some* dominant eigenvector for sufficiently large values of k (see Exercise 11).

John William Strutt Rayleigh (1842–1919)

Historical Note The British mathematical physicist John Rayleigh won the Nobel prize in physics in 1904 for his discovery of the inert gas argon. Rayleigh also made fundamental discoveries in acoustics and optics, and his work in wave phenomena enabled him to give the first accurate explanation of why the sky is blue.

[Image: The Granger Collection, New York]

The Power Method with Maximum Entry Scaling

Step 0. Choose an arbitrary nonzero vector $\mathbf{x}_0$.

Step 1. Compute $A\mathbf{x}_0$ and multiply it by the factor $1/\max(A\mathbf{x}_0)$ to obtain the first approximation $\mathbf{x}_1$ to a dominant eigenvector. Compute the Rayleigh quotient of $\mathbf{x}_1$ to obtain the first approximation to the dominant eigenvalue.

Step 2. Compute $A\mathbf{x}_1$ and scale it by the factor $1/\max(A\mathbf{x}_1)$ to obtain the second approximation $\mathbf{x}_2$ to a dominant eigenvector. Compute the Rayleigh quotient of $\mathbf{x}_2$ to obtain the second approximation to the dominant eigenvalue.

Step 3. Compute $A\mathbf{x}_2$ and scale it by the factor $1/\max(A\mathbf{x}_2)$ to obtain the third approximation $\mathbf{x}_3$ to a dominant eigenvector. Compute the Rayleigh quotient of $\mathbf{x}_3$ to obtain the third approximation to the dominant eigenvalue.

Continuing in this way will generate a sequence of better and better approximations to the dominant eigenvalue and a corresponding eigenvector.

▶ **EXAMPLE 3 Example 2 Revisited Using Maximum Entry Scaling**

Apply the power method with maximum entry scaling to

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix} \quad \text{with} \quad \mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Stop at $\mathbf{x}_5$ and compare the resulting approximations to the exact values and to the approximations obtained in Example 2.

Solution We leave it for you to confirm that

$$A\mathbf{x}_0 = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \qquad \mathbf{x}_1 = \frac{A\mathbf{x}_0}{\max(A\mathbf{x}_0)} = \frac{1}{3}\begin{bmatrix} 3 \\ 2 \end{bmatrix} \approx \begin{bmatrix} 1.00000 \\ 0.66667 \end{bmatrix}$$

$$A\mathbf{x}_1 \approx \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} 1.00000 \\ 0.66667 \end{bmatrix} \approx \begin{bmatrix} 4.33333 \\ 4.00000 \end{bmatrix} \qquad \mathbf{x}_2 = \frac{A\mathbf{x}_1}{\max(A\mathbf{x}_1)} \approx \frac{1}{4.33333}\begin{bmatrix} 4.33333 \\ 4.00000 \end{bmatrix} \approx \begin{bmatrix} 1.00000 \\ 0.92308 \end{bmatrix}$$

$$A\mathbf{x}_2 \approx \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} 1.00000 \\ 0.92308 \end{bmatrix} \approx \begin{bmatrix} 4.84615 \\ 4.76923 \end{bmatrix} \qquad \mathbf{x}_3 = \frac{A\mathbf{x}_2}{\max(A\mathbf{x}_2)} \approx \frac{1}{4.84615}\begin{bmatrix} 4.84615 \\ 4.76923 \end{bmatrix} \approx \begin{bmatrix} 1.00000 \\ 0.98413 \end{bmatrix}$$

$$A\mathbf{x}_3 \approx \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} 1.00000 \\ 0.98413 \end{bmatrix} \approx \begin{bmatrix} 4.96825 \\ 4.95238 \end{bmatrix} \qquad \mathbf{x}_4 = \frac{A\mathbf{x}_3}{\max(A\mathbf{x}_3)} \approx \frac{1}{4.96825}\begin{bmatrix} 4.96825 \\ 4.95238 \end{bmatrix} \approx \begin{bmatrix} 1.00000 \\ 0.99681 \end{bmatrix}$$

$$A\mathbf{x}_4 \approx \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\begin{bmatrix} 1.00000 \\ 0.99681 \end{bmatrix} \approx \begin{bmatrix} 4.99361 \\ 4.99042 \end{bmatrix} \qquad \mathbf{x}_5 = \frac{A\mathbf{x}_4}{\max(A\mathbf{x}_4)} \approx \frac{1}{4.99361}\begin{bmatrix} 4.99361 \\ 4.99042 \end{bmatrix} \approx \begin{bmatrix} 1.00000 \\ 0.99936 \end{bmatrix}$$

$$\lambda^{(1)} = \frac{A\mathbf{x}_1 \cdot \mathbf{x}_1}{\mathbf{x}_1 \cdot \mathbf{x}_1} = \frac{(A\mathbf{x}_1)^T\mathbf{x}_1}{\mathbf{x}_1^T\mathbf{x}_1} \approx \frac{7.00000}{1.44444} \approx 4.84615$$

$$\lambda^{(2)} = \frac{A\mathbf{x}_2 \cdot \mathbf{x}_2}{\mathbf{x}_2 \cdot \mathbf{x}_2} = \frac{(A\mathbf{x}_2)^T\mathbf{x}_2}{\mathbf{x}_2^T\mathbf{x}_2} \approx \frac{9.24852}{1.85207} \approx 4.99361$$

$$\lambda^{(3)} = \frac{A\mathbf{x}_3 \cdot \mathbf{x}_3}{\mathbf{x}_3 \cdot \mathbf{x}_3} = \frac{(A\mathbf{x}_3)^T\mathbf{x}_3}{\mathbf{x}_3^T\mathbf{x}_3} \approx \frac{9.84203}{1.96851} \approx 4.99974$$

$$\lambda^{(4)} = \frac{A\mathbf{x}_4 \cdot \mathbf{x}_4}{\mathbf{x}_4 \cdot \mathbf{x}_4} = \frac{(A\mathbf{x}_4)^T\mathbf{x}_4}{\mathbf{x}_4^T\mathbf{x}_4} \approx \frac{9.96808}{1.99362} \approx 4.99999$$

$$\lambda^{(5)} = \frac{A\mathbf{x}_5 \cdot \mathbf{x}_5}{\mathbf{x}_5 \cdot \mathbf{x}_5} = \frac{(A\mathbf{x}_5)^T\mathbf{x}_5}{\mathbf{x}_5^T\mathbf{x}_5} \approx \frac{9.99360}{1.99872} \approx 5.00000$$

Whereas the power method with Euclidean scaling produces a sequence that approaches a *unit* dominant eigenvector, maximum entry scaling produces a sequence that approaches an eigenvector whose largest *component* is 1.

Thus, $\lambda^{(5)}$ approximates the dominant eigenvalue correctly to five decimal places and $\mathbf{x}_5$ closely approximates the dominant eigenvector

$$\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

that results by taking $t = 1$ in (6). ◀

Rate of Convergence If A is a symmetric matrix whose distinct eigenvalues can be arranged so that

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_k|$$

then the "rate" at which the Rayleigh quotients converge to the dominant eigenvalue λ_1 depends on the ratio $|\lambda_1|/|\lambda_2|$; that is, the convergence is slow when this ratio is near 1 and rapid when it is large—the greater the ratio, the more rapid the convergence. For example, if A is a 2×2 symmetric matrix, then the greater the ratio $|\lambda_1|/|\lambda_2|$, the greater the disparity between the scaling effects of λ_1 and λ_2 in Figure 9.2.1, and hence the greater the effect that multiplication by A has on pulling the iterates toward the eigenspace of λ_1. Indeed, the rapid convergence in Example 3 is due to the fact that $|\lambda_1|/|\lambda_2| = 5/1 = 5$, which is considered to be a large ratio. In cases where the ratio is close to 1, the convergence of the power method may be so slow that other methods must be used.

Stopping Procedures If λ is the exact value of the dominant eigenvalue, and if a power method produces the approximation $\lambda^{(k)}$ at the kth iteration, then we call

$$\left|\frac{\lambda - \lambda^{(k)}}{\lambda}\right| \tag{12}$$

the ***relative error*** in $\lambda^{(k)}$. If this is expressed as a percentage, then it is called the ***percentage error*** in $\lambda^{(k)}$. For example, if $\lambda = 5$ and the approximation after three iterations is $\lambda^{(3)} = 5.1$, then

$$\text{relative error in } \lambda^{(3)} = \left|\frac{\lambda - \lambda^{(3)}}{\lambda}\right| = \left|\frac{5 - 5.1}{5}\right| = |-0.02| = 0.02$$

$$\text{percentage error in } \lambda^{(3)} = 0.02 \times 100\% = 2\%$$

In applications one usually knows the relative error E that can be tolerated in the dominant eigenvalue, so the goal is to stop computing iterates once the relative error in the approximation to that eigenvalue is less than E. However, there is a problem in computing the relative error from (12) in that the eigenvalue λ is unknown. To circumvent this problem, it is usual to estimate λ by $\lambda^{(k)}$ and stop the computations when

$$\left|\frac{\lambda^{(k)} - \lambda^{(k-1)}}{\lambda^{(k)}}\right| < E \tag{13}$$

The quantity on the left side of (13) is called the ***estimated relative error*** in $\lambda^{(k)}$ and its percentage form is called the ***estimated percentage error*** in $\lambda^{(k)}$.

▶ **EXAMPLE 4 Estimated Relative Error**

For the computations in Example 3, find the smallest value of k for which the estimated percentage error in $\lambda^{(k)}$ is less than 0.1%.

Solution The estimated percentage errors in the approximations in Example 3 are as follows:

	APPROXIMATION	RELATIVE ERROR	PERCENTAGE ERROR
$\lambda^{(2)}$:	$\left\|\frac{\lambda^{(2)} - \lambda^{(1)}}{\lambda^{(2)}}\right\| \approx \left\|\frac{4.99361 - 4.84615}{4.99361}\right\|$	≈ 0.02953	$= 2.953\%$
$\lambda^{(3)}$:	$\left\|\frac{\lambda^{(3)} - \lambda^{(2)}}{\lambda^{(3)}}\right\| \approx \left\|\frac{4.99974 - 4.99361}{4.99974}\right\|$	≈ 0.00123	$= 0.123\%$
$\lambda^{(4)}$:	$\left\|\frac{\lambda^{(4)} - \lambda^{(3)}}{\lambda^{(4)}}\right\| \approx \left\|\frac{4.99999 - 4.99974}{4.99999}\right\|$	≈ 0.00005	$= 0.005\%$
$\lambda^{(5)}$:	$\left\|\frac{\lambda^{(5)} - \lambda^{(4)}}{\lambda^{(5)}}\right\| \approx \left\|\frac{5.00000 - 4.99999}{5.00000}\right\|$	≈ 0.00000	$= 0\%$

Thus, $\lambda^{(4)} = 4.99999$ is the first approximation whose estimated percentage error is less than 0.1%. ◀

Remark A rule for deciding when to stop an iterative process is called a ***stopping procedure***. In the exercises, we will discuss stopping procedures for the power method that are based on the dominant eigenvector rather than the dominant eigenvalue.

Exercise Set 9.2

▶ In Exercises 1–2, the distinct eigenvalues of a matrix are given. Determine whether A has a dominant eigenvalue, and if so, find it. ◀

1. (a) $\lambda_1 = 7,\ \lambda_2 = 3,\ \lambda_3 = -8,\ \lambda_4 = 1$

(b) $\lambda_1 = -5,\ \lambda_2 = 3,\ \lambda_3 = 2,\ \lambda_4 = 5$

2. (a) $\lambda_1 = 1,\ \lambda_2 = 0,\ \lambda_3 = -3,\ \lambda_4 = 2$

(b) $\lambda_1 = -3,\ \lambda_2 = -2,\ \lambda_3 = -1,\ \lambda_4 = 3$

▶ In Exercises 3–4, apply the power method with Euclidean scaling to the matrix A, starting with $\mathbf{x}_0$, and stopping at $\mathbf{x}_4$. Compare the resulting approximations to the exact values of the dominant eigenvalue and the corresponding unit eigenvector. ◀

3. $A = \begin{bmatrix} 5 & -1 \\ -1 & -1 \end{bmatrix};\ \mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

4. $A = \begin{bmatrix} 7 & -2 & 0 \\ -2 & 6 & -2 \\ 0 & -2 & 5 \end{bmatrix};\ \mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

▶ In Exercises 5–6, apply the power method with maximum entry scaling to the matrix A, starting with $\mathbf{x}_0$, and stopping at $\mathbf{x}_4$. Compare the resulting approximations to the exact values of the dominant eigenvalue and the corresponding scaled eigenvector. ◀

5. $A = \begin{bmatrix} 1 & -3 \\ -3 & 5 \end{bmatrix};\ \mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

6. $A = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 2 & 0 \\ 2 & 0 & 4 \end{bmatrix};\ \mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

7. Let

$$A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix};\ \mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

(a) Use the power method with maximum entry scaling to approximate a dominant eigenvector of A. Start with $\mathbf{x}_0$, round off all computations to three decimal places, and stop after three iterations.

(b) Use the result in part (a) and the Rayleigh quotient to approximate the dominant eigenvalue of A.

(c) Find the exact values of the eigenvector and eigenvalue approximated in parts (a) and (b).

(d) Find the percentage error in the approximation of the dominant eigenvalue.

8. Repeat the directions of Exercise 7 with

$$A = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 10 \end{bmatrix};\ \mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

▶ In Exercises 9–10, a matrix A with a dominant eigenvalue and a sequence $\mathbf{x}_0, A\mathbf{x}_0, \ldots, A^5\mathbf{x}_0$ are given. Use Formulas (9) and (10) to approximate the dominant eigenvalue and a corresponding eigenvector. ◀

9. $A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix};\ \mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ A\mathbf{x}_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix},\ A^2\mathbf{x}_0 = \begin{bmatrix} 5 \\ 4 \end{bmatrix},$

$A^3\mathbf{x}_0 = \begin{bmatrix} 13 \\ 14 \end{bmatrix},\ A^4\mathbf{x}_0 = \begin{bmatrix} 41 \\ 40 \end{bmatrix},\ A^5\mathbf{x}_0 = \begin{bmatrix} 121 \\ 122 \end{bmatrix}$

10. $A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$; $\mathbf{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $A\mathbf{x}_0 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $A^2\mathbf{x}_0 = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$, $A^3\mathbf{x}_0 = \begin{bmatrix} 14 \\ 13 \end{bmatrix}$, $A^4\mathbf{x}_0 = \begin{bmatrix} 40 \\ 41 \end{bmatrix}$, $A^5\mathbf{x}_0 = \begin{bmatrix} 122 \\ 121 \end{bmatrix}$

11. Consider matrices

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{x}_0 = \begin{bmatrix} a \\ b \end{bmatrix}$$

where $\mathbf{x}_0$ is a unit vector and $a \neq 0$. Show that even though the matrix A is symmetric and has a dominant eigenvalue, the power sequence (1) in Theorem 9.2.1 does not converge. This shows that the requirement in that theorem that the dominant eigenvalue be positive is essential.

12. Use the power method with Euclidean scaling to approximate the dominant eigenvalue and a corresponding eigenvector of A. Choose your own starting vector, and stop when the estimated percentage error in the eigenvalue approximation is less than 0.1%.

(a) $\begin{bmatrix} 1 & 3 & 3 \\ 3 & 4 & -1 \\ 3 & -1 & 10 \end{bmatrix}$ (b) $\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 2 & -1 & 1 \\ 1 & -1 & 4 & 1 \\ 1 & 1 & 1 & 8 \end{bmatrix}$

13. Repeat Exercise 12, but this time stop when all corresponding entries in two successive eigenvector approximations differ by less than 0.01 in absolute value.

14. Repeat Exercise 12 using maximum entry scaling.

Working with Proofs

15. Prove: If A is a nonzero $n \times n$ matrix, then A^TA and AA^T have positive dominant eigenvalues.

16. ***(For readers familiar with proofs by induction)*** Let A be an $n \times n$ matrix, let $\mathbf{x}_0$ be a unit vector in R^n, and define the sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k, \ldots$ by

$$\mathbf{x}_1 = \frac{A\mathbf{x}_0}{\|A\mathbf{x}_0\|}, \quad \mathbf{x}_2 = \frac{A\mathbf{x}_1}{\|A\mathbf{x}_1\|}, \ldots, \quad \mathbf{x}_k = \frac{A\mathbf{x}_{k-1}}{\|A\mathbf{x}_{k-1}\|}, \ldots$$

Prove by induction that $\mathbf{x}_k = A^k\mathbf{x}_0/\|A^k\mathbf{x}_0\|$.

17. ***(For readers familiar with proofs by induction)*** Let A be an $n \times n$ matrix, let $\mathbf{x}_0$ be a nonzero vector in R^n, and define the sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k, \ldots$ by

$$\mathbf{x}_1 = \frac{A\mathbf{x}_0}{\max(A\mathbf{x}_0)}, \quad \mathbf{x}_2 = \frac{A\mathbf{x}_1}{\max(A\mathbf{x}_1)}, \ldots, \quad \mathbf{x}_k = \frac{A\mathbf{x}_{k-1}}{\max(A\mathbf{x}_{k-1})}, \ldots$$

Prove by induction that

$$\mathbf{x}_k = \frac{A^k\mathbf{x}_0}{\max(A^k\mathbf{x}_0)}$$

Working with Technology

T1. Use your technology utility to duplicate the computations in Example 2.

T2. Use your technology utility to duplicate the computations in Example 3.

9.3 Comparison of Procedures for Solving Linear Systems

There is an old saying that "time is money." This is especially true in industry where the cost of solving a linear system is generally determined by the time it takes for a computer to perform the required computations. This typically depends both on the speed of the computer processor and on the number of operations required by the algorithm. Thus, choosing the right algorithm has important financial implication in an industrial or research setting. In this section we will discuss some of the factors that affect the choice of algorithms for solving large-scale linear systems.

Flops and the Cost of Solving a Linear System

In computer jargon, an arithmetic operation ($+, -, *, /$) on two real numbers is called a *flop*, which is an acronym for "floating-point operation."[1] The total number of flops required to solve a problem, which is called the *cost* of the solution, provides a convenient

[1] Real numbers are stored in computers as numerical approximations called *floating-point numbers*. In base 10, a floating-point number has the form $\pm .d_1d_2 \cdots d_n \times 10^m$, where m is an integer, called the *mantissa*, and n is the number of digits to the right of the decimal point. The value of n varies with the computer. In some literature the term *flop* is used as a measure of processing speed and stands for "floating-point operations *per second*." In our usage it is interpreted as a counting unit.

It is now common in computer jargon to write "FLOPs" to mean the number of "flops per second." However, we will write "flops" simply as the plural of "flop." When needed, we will write flops per second as flops/s.

way of choosing between various algorithms for solving the problem. When needed, the cost in flops can be converted to units of time or money if the speed of the computer processor and the financial aspects of its operation are known. For example, today's fastest computers are capable of performing in excess of 17 petaflops/s (1 petaflop = 10^{15} flops). Thus, an algorithm that costs 1,000,000 flops would be performed in 0.000000001 second. By contrast, today's personal computers can perform in excess of 80 gigaflops/s (1 gigaflop = 10^9 flops). Thus, an algorithm that costs 1,000,000 flops would be performed on a personal computer in 0.00001 25 second.

To illustrate how costs (in flops) can be computed, let us count the number of flops required to solve a linear system of n equations in n unknowns by Gauss–Jordan elimination. For this purpose we will need the following formulas for the sum of the first n positive integers and the sum of the squares of the first n positive integers:

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} \tag{1}$$

$$1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} \tag{2}$$

Let $A\mathbf{x} = \mathbf{b}$ be a linear system of n equations in n unknowns to be solved by Gauss–Jordan elimination (or, equivalently, by Gaussian elimination with back substitution). For simplicity, let us assume that A is invertible and that no row interchanges are required to reduce the augmented matrix $[A \mid \mathbf{b}]$ to row echelon form. The diagrams that accompany the following analysis provide a convenient way of counting the operations required to introduce a leading 1 in the first row and then zeros below it. In our operation counts, we will lump divisions and multiplications together as "multiplications," and we will lump additions and subtractions together as "additions."

Step 1. It requires n flops (multiplications) to introduce the leading 1 in the first row.

$$\left[\begin{array}{cccccc|c} 1 & \times & \times & \cdots & \times & \times & \times \\ \bullet & \bullet & \bullet & \cdots & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \cdots & \bullet & \bullet & \bullet \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ \bullet & \bullet & \bullet & \cdots & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \cdots & \bullet & \bullet & \bullet \end{array}\right]$$

[× denotes a quantity that is being computed. • denotes a quantity that is not being computed. The augmented matrix size is $n \times (n+1)$.]

Step 2. It requires n multiplications and n additions to introduce a zero below the leading 1, and there are $n - 1$ rows below the leading 1, so the number of flops required to introduce zeros below the leading 1 is $2n(n-1)$.

$$\left[\begin{array}{cccccc|c} 1 & \bullet & \bullet & \cdots & \bullet & \bullet & \bullet \\ 0 & \times & \times & \cdots & \times & \times & \times \\ 0 & \times & \times & \cdots & \times & \times & \times \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & \times & \times & \cdots & \times & \times & \times \\ 0 & \times & \times & \cdots & \times & \times & \times \end{array}\right]$$

Column 1. Combining Steps 1 and 2, the number of flops required for column 1 is

$$n + 2n(n-1) = 2n^2 - n$$

Column 2. The procedure for column 2 is the same as for column 1, except that now we are dealing with one less row and one less column. Thus, the number of flops required to introduce the leading 1 in row 2 and the zeros below it can be obtained by replacing n by $n-1$ in the flop count for the first column. Thus, the number of flops required for column 2 is

$$2(n-1)^2 - (n-1)$$

Column 3. By the argument for column 2, the number of flops required for column 3 is

$$2(n-2)^2 - (n-2)$$

Total. The pattern should now be clear. The total number of flops required to create the n leading 1's and the associated zeros is

$$(2n^2 - n) + [2(n-1)^2 - (n-1)] + [2(n-2)^2 - (n-2)] + \cdots + (2-1)$$

which we can rewrite as

$$2[n^2 + (n-1)^2 + \cdots + 1] - [n + (n-1) + \cdots + 1]$$

or on applying Formulas (1) and (2) as

$$2\frac{n(n+1)(2n+1)}{6} - \frac{n(n+1)}{2} = \frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{1}{6}n$$

Next, let us count the number of operations required to complete the backward phase (the back substitution).

Column n. It requires $n-1$ multiplications and $n-1$ additions to introduce zeros above the leading 1 in the nth column, so the total number of flops required for the column is $2(n-1)$.

$$\left[\begin{array}{ccccc|c} 1 & * & * & \cdots & * & 0 & \times \\ 0 & 1 & * & \cdots & * & 0 & \times \\ 0 & 0 & 1 & \cdots & * & 0 & \times \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & \times \\ 0 & 0 & 0 & \cdots & 0 & 1 & * \end{array}\right]$$

Column $(n-1)$. The procedure is the same as for Step 1, except that now we are dealing with one less row. Thus, the number of flops required for the $(n-1)$st column is $2(n-2)$.

$$\left[\begin{array}{ccccc|c} 1 & * & * & \cdots & 0 & 0 & \times \\ 0 & 1 & * & \cdots & 0 & 0 & \times \\ 0 & 0 & 1 & \cdots & 0 & 0 & \times \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & * \\ 0 & 0 & 0 & \cdots & 0 & 1 & * \end{array}\right]$$

Column $(n-2)$. By the argument for column $(n-1)$, the number of flops required for column $(n-2)$ is $2(n-3)$.

Total. The pattern should now be clear. The total number of flops to complete the backward phase is

$$2(n-1) + 2(n-2) + 2(n-3) + \cdots + 2(n-n) = 2[n^2 - (1 + 2 + \cdots + n)]$$

which we can rewrite using Formula (1) as

$$2\left(n^2 - \frac{n(n+1)}{2}\right) = n^2 - n$$

In summary, we have shown that for Gauss–Jordan elimination the number of flops required for the forward and backward phases is

$$\text{flops for forward phase} = \tfrac{2}{3}n^3 + \tfrac{1}{2}n^2 - \tfrac{1}{6}n \tag{3}$$

$$\text{flops for backward phase} = n^2 - n \tag{4}$$

Thus, the total cost of solving a linear system by Gauss–Jordan elimination is

$$\text{flops for both phases} = \tfrac{2}{3}n^3 + \tfrac{3}{2}n^2 - \tfrac{7}{6}n \tag{5}$$

Cost Estimates for Solving Large Linear Systems

It is a property of polynomials that for large values of the independent variable the term of highest power makes the major contribution to the value of the polynomial. Thus, for *large* linear systems we can use (3) and (4) to approximate the number of flops in the forward and backward phases as

$$\text{flops for forward phase} \approx \tfrac{2}{3}n^3 \tag{6}$$

$$\text{flops for backward phase} \approx n^2 \tag{7}$$

Thus shows that it is more costly to execute the forward phase than the backward phase for large linear systems. Indeed, the cost difference between the forward and backward phases can be enormous, as the next example shows.

We leave it as an exercise for you to confirm the results in Table 1.

Table 1

Approximate Cost for an $n \times n$ Matrix A with Large n	
Algorithm	**Cost in Flops**
Gauss–Jordan elimination (forward phase)	$\approx \frac{2}{3}n^3$
Gauss–Jordan elimination (backward phase)	$\approx n^2$
LU-decomposition of A	$\approx \frac{2}{3}n^3$
Forward substitution to solve $L\mathbf{y} = \mathbf{b}$	$\approx n^2$
Backward substitution to solve $U\mathbf{x} = \mathbf{y}$	$\approx n^2$
A^{-1} by reducing $[A \mid I]$ to $[I \mid A^{-1}]$	$\approx 2n^3$
Compute $A^{-1}\mathbf{b}$	$\approx 2n^3$

The cost in flops for Gaussian elimination is the same as that for the forward phase of Gauss–Jordan elimination.

▶ **EXAMPLE 1 Cost of Solving a Large Linear System**

Approximate the time required to execute the forward and backward phases of Gauss–Jordan elimination for a system of one million ($= 10^6$) equations in one million unknowns using a computer that can execute 10 petaflops per second (1 petaflop $= 10^{15}$ flops).

Solution We have $n = 10^6$ for the given system, so from (6) and (7) the number of petaflops required for the forward and backward phases is

$$\text{petaflops for forward phase} \approx \tfrac{2}{3}n^3 \times 10^{-15} = \tfrac{2}{3}(10^6)^3 \times 10^{-15} = \tfrac{2}{3} \times 10^3$$

$$\text{petaflops for backward phase} \approx n^2 \times 10^{-15} = (10^6)^2 \times 10^{-15} = 10^{-3}$$

Thus, at 10 petaflops/s the execution times for the forward and backward phases are

$$\text{time for forward phase} \approx \left(\tfrac{2}{3} \times 10^3\right) \times 10^{-1}\ \text{s} \approx 66.67\ \text{s}$$

$$\text{time for backward phase} \approx (10^{-3}) \times 10^{-1}\ \text{s} \approx 0.0001\ \text{s}$$ ◀

Considerations in Choosing an Algorithm for Solving a Linear System

For a *single* linear system $A\mathbf{x} = \mathbf{b}$ of n equations in n unknowns, the methods of LU-decomposition and Gauss–Jordan elimination differ in bookkeeping but otherwise involve the same number of flops. Thus, neither method has a cost advantage over the other. However, LU-decomposition has the following advantages that make it the method of choice:

- Gauss–Jordan elimination and Gaussian elimination both use the augmented matrix $[A \mid \mathbf{b}]$, so $\mathbf{b}$ must be known. In contrast, LU-decomposition uses only the matrix A, so once that decomposition is known it can be used with as many right-hand sides as are required.
- The LU-decomposition that is computed to solve $A\mathbf{x} = \mathbf{b}$ can be used to compute A^{-1}, if needed, with little additional work.
- For large linear systems in which computer memory is at a premium, one can dispense with the storage of the 1's and zeros that appear on or below the main diagonal of U, since those entries are known from the form of U. The space that this opens up can then be used to store the entries of L, thereby reducing the amount of memory required to solve the system.
- If A is a large matrix consisting mostly of zeros, and if the nonzero entries are concentrated in a "band" around the main diagonal, then there are techniques that can be used to reduce the cost of LU-decomposition, giving it an advantage over Gauss–Jordan elimination.

Exercise Set 9.3

1. A certain computer can execute 10 gigaflops per second. Use Formula (5) to find the time required to solve the system using Gauss–Jordan elimination.
   (a) A system of 1000 equations in 1000 unknowns
   (b) A system of 10,000 equations in 10,000 unknowns
   (c) A system of 100,000 equations in 100,000 unknowns

2. A certain computer can execute 100 gigaflops per second. Use Formula (5) to find the time required to solve the system using Gauss–Jordan elimination.
   (a) A system of 10,000 equations in 10,000 unknowns
   (b) A system of 100,000 equations in 100,000 unknowns
   (c) A system of 1,000,000 equations in 1,000,000 unknowns

3. A certain computer can execute 70 gigaflops per second. Use Table 1 to estimate the time required to perform the following operations on the invertible $10{,}000 \times 10{,}000$ matrix A.
   (a) Execute the forward phase of Gauss–Jordan elimination.
   (b) Execute the backward phase of Gauss–Jordan elimination.
   (c) LU-decomposition of A
   (d) Find A^{-1} by reducing $[A \mid I]$ to $[I \mid A^{-1}]$

4. The IBM Sequoia computer can operate at speeds in excess of 16 petaflops per second (1 petaflop = 10^{15} flops). Use Table 1 to estimate the time required to perform the following operations on an invertible $100{,}000 \times 100{,}000$ matrix A.
   (a) Execute the forward phase of Gauss–Jordan elimination.
   (b) Execute the backward phase of Gauss–Jordan elimination.
   (c) LU-decomposition of A.
   (d) Find A^{-1} by reducing $[A \mid I]$ to $[I \mid A^{-1}]$

5. (a) Approximate the time required to execute the forward phase of Gauss–Jordan elimination for a system of 100,000 equations in 100,000 unknowns using a computer that can execute 1 gigaflop per second. Do the same for the backward phase. (See Table 1.)
   (b) How many gigaflops per second must a computer be able to execute to find the LU-decomposition of a matrix of size $10{,}000 \times 10{,}000$ in less than 0.5 s? (See Table 1.)

6. About how many teraflops per second must a computer be able to execute to find the inverse of a matrix of size $100{,}000 \times 100{,}000$ in less than 0.5 s? (1 teraflop = 10^{12} flops.)

▶ In Exercises 7–10, A and B are $n \times n$ matrices and c is a real number. ◀

7. How many flops are required to compute cA?

8. How many flops are required to compute $A + B$?

9. How many flops are required to compute AB?

10. If A is a diagonal matrix and k is a positive integer, how many flops are required to compute A^k?

9.4 Singular Value Decomposition

In this section we will discuss an extension of the diagonalization theory for $n \times n$ symmetric matrices to general $m \times n$ matrices. The results that we will develop in this section have applications to compression, storage, and transmission of digitized information and form the basis for many of the best computational algorithms that are currently available for solving linear systems.

Decompositions of Square Matrices

We saw in Formula (2) of Section 7.2 that every symmetric matrix A can be expressed as

$$A = PDP^T \tag{1}$$

where P is an $n \times n$ orthogonal matrix of eigenvectors of A, and D is the diagonal matrix whose diagonal entries are the eigenvalues corresponding to the column vectors of P. In this section we will call (1) an ***eigenvalue decomposition*** of A (abbreviated EVD of A).

If an $n \times n$ matrix A is not symmetric, then it does not have an eigenvalue decomposition, but it does have a ***Hessenberg decomposition***

$$A = PHP^T$$

in which P is an orthogonal matrix and H is in upper Hessenberg form (Theorem 7.2.4). Moreover, if A has real eigenvalues, then it has a ***Schur decomposition***

$$A = PSP^T$$

in which P is an orthogonal matrix and S is upper triangular (Theorem 7.2.3).

The eigenvalue, Hessenberg, and Schur decompositions are important in numerical algorithms not only because the matrices D, H, and S have simpler forms than A, but also because the orthogonal matrices that appear in these factorizations do not magnify roundoff error. To see why this is so, suppose that $\tilde{\mathbf{x}}$ is a column vector whose entries are known exactly and that

$$\mathbf{x} = \tilde{\mathbf{x}} + \mathbf{e}$$

is the vector that results when roundoff error is present in the entries of $\tilde{\mathbf{x}}$. If P is an orthogonal matrix, then the length-preserving property of orthogonal transformations implies that

$$\|P\mathbf{x} - P\tilde{\mathbf{x}}\| = \|\mathbf{x} - \tilde{\mathbf{x}}\| = \|\mathbf{e}\|$$

which tells us that the error in approximating $P\tilde{\mathbf{x}}$ by $P\mathbf{x}$ has the same magnitude as the error in approximating $\tilde{\mathbf{x}}$ by $\mathbf{x}$.

There are two main paths that one might follow in looking for other kinds of decompositions of a general square matrix A: One might look for decompositions of the form

$$A = PJP^{-1}$$

in which P is invertible but not necessarily orthogonal, or one might look for decompositions of the form

$$A = U\Sigma V^T$$

in which U and V are orthogonal but not necessarily the same. The first path leads to decompositions in which J is either diagonal or a certain kind of block diagonal matrix, called a ***Jordan canonical form*** in honor of the French mathematician Camille Jordan (see p. 518). Jordan canonical forms, which we will not consider in this text, are important theoretically and in certain applications, but they are of lesser importance numerically because of the roundoff difficulties that result from the lack of orthogonality in P. In this section we will focus on the second path.

Singular Values

Since matrix products of the form A^TA will play an important role in our work, we will begin with two basic theorems about them.

THEOREM 9.4.1 *If A is an $m \times n$ matrix, then:*

(a) A and A^TA have the same null space.

(b) A and A^TA have the same row space.

(c) A^T and A^TA have the same column space.

(d) A and A^TA have the same rank.

We will prove part (a) and leave the remaining proofs for the exercises.

Proof (a) We must show that every solution of $A\mathbf{x} = \mathbf{0}$ is a solution of $A^TA\mathbf{x} = \mathbf{0}$, and conversely. If $\mathbf{x}_0$ is any solution of $A\mathbf{x} = \mathbf{0}$, then $\mathbf{x}_0$ is also a solution of $A^TA\mathbf{x} = \mathbf{0}$ since

$$A^TA\mathbf{x}_0 = A^T(A\mathbf{x}_0) = A^T\mathbf{0} = \mathbf{0}$$

Conversely, if $\mathbf{x}_0$ is any solution of $A^TA\mathbf{x} = \mathbf{0}$, then $\mathbf{x}_0$ is in the null space of A^TA and hence is orthogonal to all vectors in the row space of A^TA by part (*q*) of Theorem 4.8.8. However, A^TA is symmetric, so $\mathbf{x}_0$ is also orthogonal to every vector in the column space of A^TA. In particular, $\mathbf{x}_0$ must be orthogonal to the vector $(A^TA)\mathbf{x}_0$; that is,

$$\mathbf{x}_0 \cdot (A^TA)\mathbf{x}_0 = 0$$

Using the first formula in Table 1 of Section 3.2 and properties of the transpose operation we can rewrite this as

$$\mathbf{x}_0^T(A^TA)\mathbf{x}_0 = (A\mathbf{x}_0)^T(A\mathbf{x}_0) = (A\mathbf{x}_0) \cdot (A\mathbf{x}_0) = \|A\mathbf{x}_0\|^2 = 0$$

which implies that $A\mathbf{x}_0 = \mathbf{0}$, thereby proving that $\mathbf{x}_0$ is a solution of $A\mathbf{x}_0 = \mathbf{0}$. ◀

THEOREM 9.4.2 *If A is an $m \times n$ matrix, then:*

(a) A^TA is orthogonally diagonalizable.

(b) The eigenvalues of A^TA are nonnegative.

Proof (a) The matrix A^TA, being symmetric, is orthogonally diagonalizable by Theorem 7.2.1.

Proof (b) Since A^TA is orthogonally diagonalizable, there is an orthonormal basis for R^n consisting of eigenvectors of A^TA, say $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$. If we let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the corresponding eigenvalues, then for $1 \le i \le n$ we have

$$\begin{aligned} \|A\mathbf{v}_i\|^2 &= A\mathbf{v}_i \cdot A\mathbf{v}_i = \mathbf{v}_i \cdot A^TA\mathbf{v}_i \qquad \text{[Formula (26) of Section 3.2]} \\ &= \mathbf{v}_i \cdot \lambda_i\mathbf{v}_i = \lambda_i(\mathbf{v}_i \cdot \mathbf{v}_i) = \lambda_i\|\mathbf{v}_i\|^2 = \lambda_i \end{aligned}$$

It follows from this relationship that $\lambda_i \ge 0$. ◀

We will assume throughout this section that the eigenvalues of A^TA are named so that

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \ge 0$$

and hence that

$$\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n \ge 0$$

DEFINITION 1 If A is an $m \times n$ matrix, and if $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of A^TA, then the numbers

$$\sigma_1 = \sqrt{\lambda_1}, \quad \sigma_2 = \sqrt{\lambda_2}, \ldots, \quad \sigma_n = \sqrt{\lambda_n}$$

are called the ***singular values*** of A.

▶ **EXAMPLE 1 Singular Values**

Find the singular values of the matrix

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

530 / 802

Solution The first step is to find the eigenvalues of the matrix

$$A^TA = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

The characteristic polynomial of A^TA is

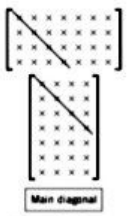
$$\lambda^2 - 4\lambda + 3 = (\lambda - 3)(\lambda - 1)$$

so the eigenvalues of A^TA are $\lambda_1 = 3$ and $\lambda_2 = 1$ and the singular values of A in order of decreasing size are

$$\sigma_1 = \sqrt{\lambda_1} = \sqrt{3}, \quad \sigma_2 = \sqrt{\lambda_2} = 1 \quad ◀$$

Singular Value Decomposition

Before turning to the main result in this section, we will find it useful to extend the notion of a "main diagonal" to matrices that are not square. We define the ***main diagonal*** of an $m \times n$ matrix to be the line of entries shown in Figure 9.4.1—it starts at the upper left corner and extends diagonally as far as it can go. We will refer to the entries on the main diagonal as the ***diagonal entries***.



▲ Figure 9.4.1

We are now ready to consider the main result in this section, which is concerned with a specific way of factoring a general $m \times n$ matrix A. This factorization, called ***singular value decomposition*** (abbreviated SVD) will be given in two forms, a brief form that captures the main idea, and an expanded form that spells out the details. The proof is given at the end of this section.

THEOREM 9.4.3 Singular Value Decomposition (Brief Form)

If A is an $m \times n$ matrix of rank k, then A can be expressed in the form $A = U\Sigma V^T$, where Σ has size $m \times n$ and can be expressed in partitioned form as

$$\Sigma = \left[\begin{array}{c|c} D & O_{k \times (n-k)} \\ \hline O_{(m-k) \times k} & O_{(m-k) \times (n-k)} \end{array}\right]$$

in which D is a diagonal $k \times k$ matrix whose successive entries are the first k singular values of A in nonincreasing order, U is an $m \times n$ orthogonal matrix, and V is an $n \times n$ orthogonal matrix.

Harry Bateman
(1882–1946)

Historical Note The term *singular value* is apparently due to the British-born mathematician Harry Bateman, who used it in a research paper published in 1908. Bateman emigrated to the United States in 1910, teaching at Bryn Mawr College, Johns Hopkins University, and finally at the California Institute of Technology. Interestingly, he was awarded his Ph.D. in 1913 by Johns Hopkins at which point in time he was already an eminent mathematician with 60 publications to his name.
[*Image: Courtesy of the Archives, California Institute of Technology*]

THEOREM 9.4.4 Singular Value Decomposition (Expanded Form)
If A is an $m \times n$ matrix of rank k, then A can be factored as

$$A = U\Sigma V^T = [\mathbf{u}_1 \ \mathbf{u}_2 \ \cdots \ \mathbf{u}_k \mid \mathbf{u}_{k+1} \ \cdots \ \mathbf{u}_m] \left[\begin{array}{ccc|c} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_k \\ \hline \multicolumn{3}{c|}{\mathbf{0}_{m-k,\,k}} & \mathbf{0}_{m-k,\,n-k} \end{array}\right] \left[\begin{array}{c} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_k^T \\ \hline \mathbf{v}_{k+1}^T \\ \vdots \\ \mathbf{v}_n^T \end{array}\right]$$

in which U, Σ, and V have sizes $m \times m$, $m \times n$, and $n \times n$, respectively, and in which:

(a) $V = [\mathbf{v}_1 \ \ \mathbf{v}_2 \ \ \cdots \ \ \mathbf{v}_n]$ *orthogonally diagonalizes* A^TA.

(b) *The nonzero diagonal entries of Σ are $\sigma_1 = \sqrt{\lambda_1}$, $\sigma_2 = \sqrt{\lambda_2}, \ldots, \sigma_k = \sqrt{\lambda_k}$, where $\lambda_1, \lambda_2, \ldots, \lambda_k$ are the nonzero eigenvalues of A^TA corresponding to the column vectors of V.*

(c) *The column vectors of V are ordered so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$.*

(d) $\mathbf{u}_i = \dfrac{A\mathbf{v}_i}{\|A\mathbf{v}_i\|} = \dfrac{1}{\sigma_i} A\mathbf{v}_i \quad (i = 1, 2, \ldots, k)$

(e) $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$ *is an orthonormal basis for* col(A).

(f) $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_m\}$ *is an extension of* $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$ *to an orthonormal basis for* R^m.

The vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k$ are called the *left singular vectors* of A, and the vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ are called the *right singular vectors of A*.

▶ **EXAMPLE 2 Singular Value Decomposition If A Is Not Square**

Find a singular value decomposition of the matrix

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Solution We showed in Example 1 that the eigenvalues of A^TA are $\lambda_1 = 3$ and $\lambda_2 = 1$ and that the corresponding singular values of A are $\sigma_1 = \sqrt{3}$ and $\sigma_2 = 1$. We leave it for you to verify that

$$\mathbf{v}_1 = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix} \quad \text{and} \quad \mathbf{v}_2 = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix}$$

are eigenvectors corresponding to λ_1 and λ_2, respectively, and that $V = [\mathbf{v}_1 \mid \mathbf{v}_2]$ orthogonally diagonalizes A^TA. From part (*d*) of Theorem 9.4.4, the vectors

$$\mathbf{u}_1 = \frac{1}{\sigma_1} A\mathbf{v}_1 = \frac{\sqrt{3}}{3} \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{6}}{3} \\ \frac{\sqrt{6}}{6} \\ \frac{\sqrt{6}}{6} \end{bmatrix}$$

$$\mathbf{u}_2 = \frac{1}{\sigma_2} A\mathbf{v}_2 = (1) \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}$$

are two of the three column vectors of U. Note that $\mathbf{u}_1$ and $\mathbf{u}_2$ are orthonormal, as expected. We could extend the set $\{\mathbf{u}_1, \mathbf{u}_2\}$ to an orthonormal basis for R^3. However, the computations will be easier if we first remove the messy radicals by multiplying $\mathbf{u}_1$ and $\mathbf{u}_2$ by appropriate scalars. Thus, we will look for a unit vector $\mathbf{u}_3$ that is orthogonal to

$$\sqrt{6}\mathbf{u}_1 = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \sqrt{2}\mathbf{u}_2 = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$$

To satisfy these two orthogonality conditions, the vector $\mathbf{u}_3$ must be a solution of the homogeneous linear system

$$\begin{bmatrix} 2 & 1 & 1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

We leave it for you to show that a general solution of this system is

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = t \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

Normalizing the vector on the right yields

$$\mathbf{u}_3 = \begin{bmatrix} -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \end{bmatrix}$$

Eugenio Beltrami
(1835–1900)

Camille Jordan
(1838–1922)

Hermann Klaus Weyl
(1885–1955)

Gene H. Golub
(1933–2007)

Historical Note The theory of singular value decompositions can be traced back to the work of five people: the Italian mathematician Eugenio Beltrami, the French mathematician Camille Jordan, the English mathematician James Sylvester (see p. 35), and the German mathematicians Erhard Schmidt (see p. 371) and the mathematician Herman Weyl. More recently, the pioneering efforts of the American mathematician Gene Golub produced a stable and efficient algorithm for computing it. Beltrami and Jordan were the progenitors of the decomposition—Beltrami gave a proof of the result for real, invertible matrices with distinct singular values in 1873. Subsequently, Jordan refined the theory and eliminated the unnecessary restrictions imposed by Beltrami. Sylvester, apparently unfamiliar with the work of Beltrami and Jordan, rediscovered the result in 1889 and suggested its importance. Schmidt was the first person to show that the singular value decomposition could be used to approximate a matrix by another matrix with lower rank, and, in so doing, he transformed it from a mathematical curiosity to an important practical tool. Weyl showed how to find the lower rank approximations in the presence of error.
[*Images: http://turnbull.mcs.st-andrews.ac.uk/history/PictDisplay/Beltrami.html* (Beltrami); *The Granger Collection, New York* (Jordan); *Courtesy Electronic Publishing Services, Inc., New York City* (Weyl); *Courtesy of Hector Garcia-Molina* (Golub)]

Thus, the singular value decomposition of A is

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$$

$$A = U \Sigma V^T$$

You may want to confirm the validity of this equation by multiplying out the matrices on the right side. ◀

OPTIONAL

We conclude this section with an optional proof of Theorem 9.4.4.

Proof of Theorem 9.4.4 For notational simplicity we will prove this theorem in the case where A is an $n \times n$ matrix. To modify the argument for an $m \times n$ matrix you need only make the notational adjustments required to account for the possibility that $m > n$ or $n > m$.

The matrix A^TA is symmetric, so it has an eigenvalue decomposition

$$A^TA = VDV^T$$

in which the column vectors of

$$V = [\mathbf{v}_1 \mid \mathbf{v}_2 \mid \cdots \mid \mathbf{v}_n]$$

are unit eigenvectors of A^TA, and D is a diagonal matrix whose successive diagonal entries $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of A^TA corresponding in succession to the column vectors of V. Since A is assumed to have rank k, it follows from Theorem 9.4.1 that A^TA also has rank k. It follows as well that D has rank k, since it is similar to A^TA and rank is a similarity invariant. Thus, D can be expressed in the form

$$D = \begin{bmatrix} \lambda_1 & & & & & 0 \\ & \lambda_2 & & & & \\ & & \ddots & & & \\ & & & \lambda_k & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix} \tag{2}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > 0$. Now let us consider the set of image vectors

$$\{A\mathbf{v}_1, A\mathbf{v}_2, \ldots, A\mathbf{v}_n\} \tag{3}$$

This is an orthogonal set, for if $i \neq j$, then the orthogonality of $\mathbf{v}_i$ and $\mathbf{v}_j$ implies that

$$A\mathbf{v}_i \cdot A\mathbf{v}_j = \mathbf{v}_i \cdot A^TA\mathbf{v}_j = \mathbf{v}_i \cdot \lambda_j\mathbf{v}_j = \lambda_j(\mathbf{v}_i \cdot \mathbf{v}_j) = 0$$

Moreover, the first k vectors in (3) are nonzero since we showed in the proof of Theorem 9.4.2(*b*) that $\|A\mathbf{v}_i\|^2 = \lambda_i$ for $i = 1, 2, \ldots, n$, and we have assumed that the first k diagonal entries in (2) are positive. Thus,

$$S = \{A\mathbf{v}_1, A\mathbf{v}_2, \ldots, A\mathbf{v}_k\}$$

is an orthogonal set of *nonzero* vectors in the column space of A. But the column space of A has dimension k since

$$\text{rank}(A) = \text{rank}(A^TA) = k$$

and hence S, being a linearly independent set of k vectors, must be an orthogonal basis for col(A). If we now normalize the vectors in S, we will obtain an orthonormal basis $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}$ for col(A) in which

$$\mathbf{u}_i = \frac{A\mathbf{v}_i}{\|A\mathbf{v}_i\|} = \frac{1}{\sqrt{\lambda_i}}A\mathbf{v}_i \quad (1 \leq i \leq k)$$

or, equivalently, in which

$$A\mathbf{v}_1 = \sqrt{\lambda_1}\mathbf{u}_1 = \sigma_1\mathbf{u}_1, \quad A\mathbf{v}_2 = \sqrt{\lambda_2}\mathbf{u}_2 = \sigma_2\mathbf{u}_2, \ldots, \quad A\mathbf{v}_k = \sqrt{\lambda_k}\mathbf{u}_k = \sigma_k\mathbf{u}_k \tag{4}$$

It follows from Theorem 6.3.6 that we can extend this to an orthonormal basis

$$\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_n\}$$

for R^n. Now let U be the orthogonal matrix

$$U = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_k \quad \mathbf{u}_{k+1} \quad \cdots \quad \mathbf{u}_n]$$

and let Σ be the diagonal matrix

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & & 0 \\ & \sigma_2 & & & & \\ & & \ddots & & & \\ & & & \sigma_k & & \\ & & & & 0 & \\ 0 & & & & & 0 \end{bmatrix}$$

It follows from (4), and the fact that $A\mathbf{v}_i = 0$ for $i > k$, that

$$\begin{aligned} U\Sigma &= [\sigma_1\mathbf{u}_1 \quad \sigma_2\mathbf{u}_2 \quad \cdots \quad \sigma_k\mathbf{u}_k \quad 0 \quad \cdots \quad 0] \\ &= [A\mathbf{v}_1 \quad A\mathbf{v}_2 \quad \cdots \quad A\mathbf{v}_k \quad A\mathbf{v}_{k+1} \quad \cdots \quad A\mathbf{v}_n] \\ &= AV \end{aligned}$$

which we can rewrite using the orthogonality of V as $A = U\Sigma V^T$. ◀

Exercise Set 9.4

▶ In Exercises 1–4, find the distinct singular values of A. ◀

1. $A = \begin{bmatrix} 1 & 2 & 0 \end{bmatrix}$

2. $A = \begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix}$

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

4. $A = \begin{bmatrix} \sqrt{2} & 0 \\ 1 & \sqrt{2} \end{bmatrix}$

▶ In Exercises 5–12, find a singular value decomposition of A. ◀

5. $A = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$

6. $A = \begin{bmatrix} -3 & 0 \\ 0 & -4 \end{bmatrix}$

7. $A = \begin{bmatrix} 4 & 6 \\ 0 & 4 \end{bmatrix}$

8. $A = \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}$

9. $A = \begin{bmatrix} -2 & 2 \\ -1 & 1 \\ 2 & -2 \end{bmatrix}$

10. $A = \begin{bmatrix} -2 & -1 & 2 \\ 2 & 1 & -2 \end{bmatrix}$

11. $A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 1 \end{bmatrix}$

12. $A = \begin{bmatrix} 6 & 4 \\ 0 & 0 \\ 4 & 0 \end{bmatrix}$

Working with Proofs

13. Prove: If A is an $m \times n$ matrix, then A^TA and AA^T have the same rank.

14. Prove part (*d*) of Theorem 9.4.1 by using part (*a*) of the theorem and the fact that A and A^TA have n columns.