Chapter 18

The Algebraic Eigenvalue Problem

You should be familiar with

- Ordinary differential equations for Sections 18.1.1 and 18.1.3.
- Eigenvalues, eigenvectors, and their basic properties covered in Chapter 5.
- Vector and matrix norms and the matrix condition number $\kappa(A)$.
- QR decomposition, both full and reduced.
- Householder reflections and Givens rotations.
- Reduction to upper triangular form using orthogonal matrices.

This chapter discusses the computation of eigenvalues and eigenvectors of nonsymmetric matrices. As we discussed in Chapter 10, finding the roots of the characteristic polynomial is not acceptable, since the problem of finding roots of polynomials is unstable. Since the eigenvalue problem is extremely important in areas such as engineering, physics, chemistry, statistics, and economics, we need to know how to solve the problem accurately. The chapter provides three examples where the eigenvalue problem arises, vibration analysis, the Leslie model for population ecology, and buckling of a column.

Some applications only need the largest or smallest eigenvalue of a matrix, so the iterative power and inverse power methods are often used.

Application of the QR decomposition is the most commonly used method for computing eigenvalues. Multiple versions of the QR algorithm are discussed. The basic QR algorithm for computing eigenvalues and eigenvectors is very simple to implement; however, is not efficient. Recall that similar matrices have the same eigenvalues. The basic QR algorithm can be greatly improved by first reducing the matrix to what is termed upper Hessenberg form using a transformation $H = P^{\mathrm{T}}AP$, where H is upper Hessenberg and P is orthogonal. The application of the basic algorithm finds the eigenvalues of H, which are the same as those of A (Theorem 5.1). In practice, the basic QR algorithm is improved by shifting the problem to computing the eigenvalue of a nearby matrix and, as a result, better isolate an eigenvalue from nearby eigenvalues.

For computing the eigenvalues and eigenvectors of a small matrix ($n < 1000$), the algorithm of choice for many years has been the implicit QR iteration, often called the Francis algorithm. This method involves a transformation to upper Hessenberg form, followed by a series of transformations that reduce the Hessenberg matrix to upper triangular form using what are termed single and double shifts. These shifts are done implicitly to save significant computation. In this chapter, we will present the algorithm that uses the double shift to compute the eigenvalues for a general matrix. The double shift is necessary to find complex conjugate eigenvalues. Chapter 19 discusses the algorithm for a symmetric matrix, in which only single shifts are used.

It may be that a particular eigenvalue is known and a corresponding eigenvector is required. In this case, the Hessenberg inverse iteration can be used to compute the eigenvector. In Section 18.10, we build a general eigenvalue/eigenvector solver that uses transformation to upper Hessenberg form, followed by the Francis algorithm to compute the eigenvalues, and the Hessenberg inverse iteration to compute corresponding eigenvectors.

We discussed perturbation theory for solving the linear system $Ax = b$, and there are similar results for the computation of eigenvalues. In particular, we can define a condition number for a particular eigenvalue and use it to estimate the difficulty of computing the eigenvalue.

18.1 APPLICATIONS OF THE EIGENVALUE PROBLEM

The applications of the eigenvalue problem in engineering and science are vast, including such areas as the theory of vibration, analysis of buckling beams, principle component analysis in statistics, economic models, and quantum physics. In this section, we will present three applications.

Numerical Linear Algebra with Applications. http://dx.doi.org/10.1016/B978-0-12-394435-1.00018-1

FIGURE 18.1 Tacoma Narrows Bridge collapse. *Source: University of Washington Libraries, Special Collections, UW 21422.*

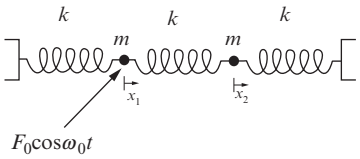

FIGURE 18.2 Mass-spring system.

18.1.1 Vibrations and Resonance

One application of eigenvalues and eigenvectors is in the analysis of vibration problems. You have probably heard of the collapse of the Tacoma Narrows Bridge (Figure 18.1) in the state of Washington. For a period of time, the bridge would move in small waves and became a tourist attraction. One day, the wind was approximately 40 miles/hr, and the oscillations of the bridge increased to the point where the bridge tore apart and crashed into the water. There is still debate as to the cause of the collapse, but one explanation is that the frequency of the wind was close to the fundamental frequency of the bridge. The fundamental frequency of the bridge is the magnitude of the smallest eigenvalue of a system that mathematically models the bridge. The lowest frequency is the most dangerous for a structure or machine because that mode corresponds to the largest displacement.

Consider the vibration of two objects of mass m attached to each other and the walls by three springs with spring constant k. There is no damping, and the springs cannot move vertically. A driving force $F_0 \cos(\beta t)$ acts on the left mass (Figure 18.2). The equations of motion for the displacement of the masses are:

$$\begin{aligned} m\frac{\mathrm{d}^2x_1}{\mathrm{d}t^2} &= -2kx_1 + kx_2 + F_0 \cos\omega_0 t \\ m\frac{\mathrm{d}^2x_2}{\mathrm{d}t^2} &= kx_1 - 2kx_2 \end{aligned}$$

In matrix form, we have

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = \frac{1}{m}Kx + \frac{F}{m}, \tag{18.1}$$

where

$$K = \begin{bmatrix} -2k & k \\ k & -2k \end{bmatrix}, \quad F = \begin{bmatrix} F_0 \cos\omega_0 t \\ 0 \end{bmatrix}.$$

where K is the *stiffness* matrix and F is the *force vector*. The general solution to Equation 18.1 is a linear combination of the solution to the homogeneous system

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} - \frac{K}{m}x = 0 \tag{18.2}$$

and a particular solution to Equation 18.1. Try a solution to the homogeneous equation of the form $x = v\mathrm{e}^{\omega t}$, where ω is a frequency. Substituting $x = v\mathrm{e}^{\omega t}$ into Equation 18.2 gives

$$\omega^2 v\mathrm{e}^{\omega t} = \frac{K}{m}v\mathrm{e}^{\omega t},$$

and we have the eigenvalue problem

$$Kv = \left(m\omega^2\right)v. \tag{18.3}$$

The coefficient matrix is symmetric, so it has real eigenvalues and corresponding real linearly independent eigenvectors (Lemma 7.3 and Theorem 7.6). The characteristic equation is

$$\lambda^2 + 4k\lambda + 3k^2 = (\lambda + 3k)(\lambda + k) = 0.$$

The eigenvalues and corresponding normalized eigenvectors are

$$\begin{aligned} \lambda_1 &= -3k \quad v_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 \\ 1 \end{bmatrix} \\ \lambda_2 &= -k \quad v_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned}$$

Now, using Equation 18.3,

$$\omega_1^2 = -3\frac{k}{m}, \quad \omega_2^2 = -\frac{k}{m},$$

and

$$\omega_1 = \mathrm{i}\sqrt{\frac{3k}{m}}, \quad \omega_2 = \mathrm{i}\sqrt{\frac{k}{m}}.$$

The corresponding solutions are

$$v_1\mathrm{e}^{\mathrm{i}\sqrt{\frac{3k}{m}}t}, \quad v_2\mathrm{e}^{\mathrm{i}\sqrt{\frac{k}{m}}t}.$$

Applying Euler's formula (Equation A.2) and taking the real and imaginary parts, the general solution to the homogeneous equation is

$$x_{\mathrm{h}}(t) = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 \\ 1 \end{bmatrix}\left(c_1\cos\sqrt{\frac{3k}{m}}t + c_2\sin\sqrt{\frac{3k}{m}}t\right) + \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\left(c_3\cos\sqrt{\frac{k}{m}}t + c_4\sin\sqrt{\frac{k}{m}}t\right).$$

The frequencies $\sqrt{\frac{3k}{m}}$ and $\sqrt{\frac{k}{m}}$ are known as the *natural frequencies* of the system. Without a driving force, the system vibrates with these frequencies.

We now determine a particular solution, $x_{\mathrm{p}}(t)$, by using complex variables and replacing the driving force $F = \begin{bmatrix} F_0\cos\omega_0 t \\ 0 \end{bmatrix}$ by $\begin{bmatrix} F_0 \\ 0 \end{bmatrix}\mathrm{e}^{\mathrm{i}\omega_0 t}$. After finding a imaginary particular solution, the real part is the particular solution we are looking for.

Using the method of undetermined coefficients, the trial solution is $x_{\mathrm{p}}(t) = D\mathrm{e}^{\mathrm{i}\omega_0 t}$. Substitute it into Equation 18.1 to obtain

$$\left(\frac{K}{m} + \omega_0^2 I\right)D = -\frac{1}{m}\begin{bmatrix} F_0 \\ 0 \end{bmatrix}.$$

If $\left(\frac{K}{m} + \omega_0^2 I\right)$ is invertible

$$D = -\frac{1}{m}\left(\frac{K}{m} + \omega_0^2 I\right)^{-1}\begin{bmatrix} F_0 \\ 0 \end{bmatrix},$$

and

$$x_{\mathrm{p}}(t) = D\cos\omega_0 t,$$

giving the general solution $x(t) = x_{\mathrm{h}}(t) + x_{\mathrm{p}}(t)$.
Now, let $\det\left(\frac{K}{m} + \omega_0^2 I\right) = 0$.

$$\begin{aligned}\det\left(\frac{K}{m} + \omega_0^2 I\right) &= \det\left(\begin{bmatrix} \omega_0^2 - \frac{2k}{m} & \frac{k}{m} \\ \frac{k}{m} & \omega_0^2 - \frac{2k}{m} \end{bmatrix}\right) \\ &= \left(\omega_0^2 - \frac{3k}{m}\right)\left(\omega_0^2 - \frac{k}{m}\right).\end{aligned}$$

When

$$\omega_0 = \sqrt{\frac{k}{m}}, \quad \omega_0 = \sqrt{\frac{3k}{m}},$$

$\left(\frac{K}{m} + \omega_0^2 I\right)$ is not invertible. Thus, as ω_0 approaches either natural frequency $\sqrt{\frac{k}{m}}$ or $\sqrt{\frac{3k}{m}}$, $D\left(\frac{K}{m} + \omega_0^2 I\right)$ is close to singular. Example 18.1 investigates what happens when ω_0 approaches these frequencies.

Example 18.1. The initial conditions determine the constants c_i, $1 \le i \le 4$. Assume at time $t = 0$ the spring system is at rest, so that $x(0) = x'(0) = 0$, and that the driving force starts motion. We will leave ω_0 as a variable and investigate what happens as ω_0 varies. The constants are determined by solving the system of equations

$$\begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & -\sqrt{\frac{k}{2m}} & 0 & \sqrt{\frac{3k}{2m}} \\ 0 & \sqrt{\frac{k}{2m}} & 0 & \sqrt{\frac{3k}{2m}} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} -d_1 \\ -d_2 \\ 0 \\ 0 \end{bmatrix},$$

where d_1 and d_2 are the components of D. The solution $x(t) = x_{\mathrm{h}}(t) + x_{\mathrm{p}}(t)$ can be computed when we have values for m, k, and F_0, and we assume $m = 1$, $k = 1$, and $F_0 = 5$. Figure 18.3 shows the behavior of $x_1(t)$, $x_2(t)$, $0 \le t \le 20$ for $\omega_0 = 2.0$ and $\omega_0 = \sqrt{3} - 0.001$. Note the large oscillations for the latter value of ω_0. This caused by the fact that ω_0 is close to the natural frequency $\sqrt{\frac{3k}{m}}$, and the system approaches *resonance*. Resonance is the tendency of a system to oscillate at a greater amplitude at some frequencies than at others. Note that resonance occurs in our example when the frequency of the driving force approaches $\sqrt{\frac{3k}{m}}$, one of the natural frequencies of the system. ■

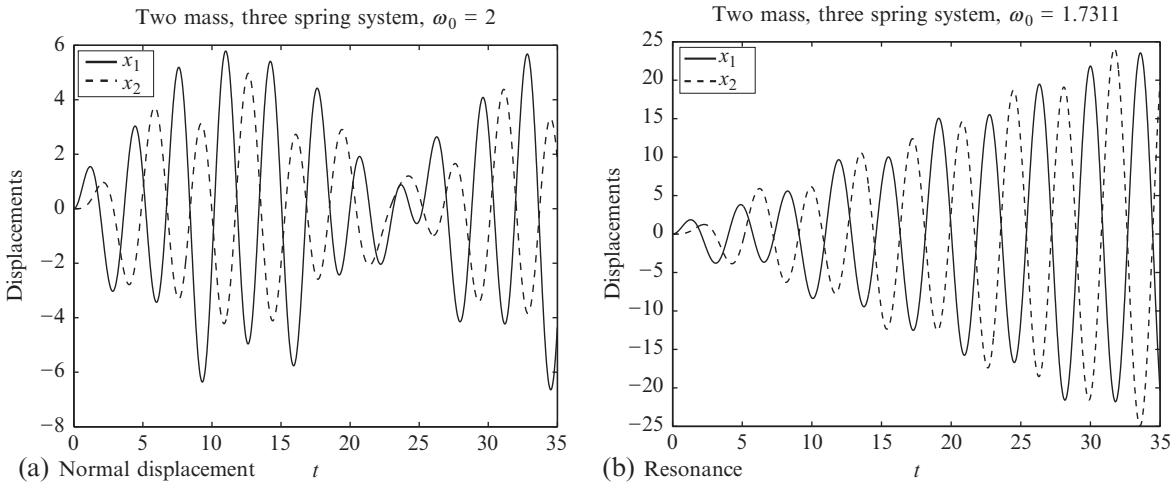

FIGURE 18.3 Solution to a system of ordinary differential equations.

18.1.2 The Leslie Model in Population Ecology

The model of Leslie is a heavily used tool in population ecology. It is a model of an age-structured population which predicts how distinct populations change over time. The model runs for n units of time, beginning at 0. For simplicity of modeling, we sort individuals into discrete age classes, denoted $n_i(t)$. We further simplify our model by assuming that there is an approximately 50:50 male to female ratio, and that the number of offspring per year depends primarily on the number of females. Therefore, we only consider the females present in the population. There are m age classes, $1, \ldots, m$, where m is the maximum reproductive age of an individual. The model uses the following parameters for the groups:

p_i	probability of surviving from age i to $i+1$
f_i	average number of offspring surviving to age 1 in class i (fertility function)
$n_i(t)$	number of females of age i at time t (the number of i-year olds at time t)

In specifying f_i, the mortality rate of offspring and parents is included.

The number of individuals in age class i at time $t+1$ depends on the number of individuals surviving from age class $i-1$ at time t. The age class $i=1$ at time t consists of new borns. First, consider how many individuals in age class i survive from t to $t+1$, and do not consider births. This is given by the equation

$$n_i(t+1) = p_{i-1} n_{i-1}(t), \quad 2 \le i \le m$$

Now consider age class 1, the newly born individuals. The number individuals in age class 1 at time $t+1$ is the number of offspring born to existing individuals in the population at time t:

$$n_1(t+1) = \sum_{i=1}^{m} f_i n_i(t)$$

Now place all of the $n_i(t)$ values into a column vector $N(t)$ of dimension m and build the $m \times m$ *Leslie matrix* L as follows:

The first row of L consists of the fertility values f_i, $1 \le i \le m$, and the subdiagonal contains the probabilities of survival. The remaining entries are zero.

$$L = \begin{bmatrix} f_1 & f_2 & \cdots & f_{m-1} & f_m \\ p_1 & 0 & 0 & \cdots & 0 \\ 0 & p_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & & p_{m-1} & 0 \end{bmatrix}$$

The product of L and $N(t)$ is

$$\begin{aligned} L(N(t)) &= \begin{bmatrix} f_1 & f_2 & \cdots & f_{m-1} & f_m \\ p_1 & 0 & 0 & \cdots & 0 \\ 0 & p_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & & p_{m-1} & 0 \end{bmatrix} \begin{bmatrix} n_1(t) \\ n_2(t) \\ \vdots \\ n_{m-1}(t) \\ n_m(t) \end{bmatrix} \\ &= \begin{bmatrix} \sum_{i=1}^{m} f_i n_i(t) \\ p_1 n_1(t) \\ p_2 n_2(t) \\ \vdots \\ p_{m-1} n_{m-1}(t) \end{bmatrix} = N(t+1), \end{aligned}$$

so

$$N(t+1) = L(N(t)).$$

Thus,

$$N(t+2) = L(N(t+1)) = L^2 N(t),$$

and in general for each time $t+k$

$$N(t+k) = L^k N(t). \tag{18.4}$$

Equation 18.4 says that we can project into the future by computing matrix powers.

The digraph produced by L is strongly connected, so L is irreducible. By Theorem 5.6, L has an eigenvector with strictly positive entries. Thus, we can assume that v is an eigenvector of L, with $v_1 \neq 0$. Divide by v_1 so that

$$v = \begin{bmatrix} 1 \\ v_2 \\ \vdots \\ v_{m-1} \\ v_m \end{bmatrix}.$$

Our aim is to determine the characteristic equation for L, from which we will obtain valuable information. Let $Lv = \lambda v$ to obtain

$$\begin{bmatrix} f_1 + f_2 v_2 + \cdots + f_m v_m \\ p_1 \\ p_2 v_2 \\ \vdots \\ \vdots \\ p_{m-1} v_{m-1} \end{bmatrix} = \begin{bmatrix} \lambda \\ \lambda v_2 \\ \lambda v_3 \\ \vdots \\ \lambda v_{m-1} \\ \lambda v_m \end{bmatrix}. \tag{18.5}$$

Equate entries 2 through m to get

$$\begin{aligned} p_1 &= \lambda v_2 \\ p_2 v_2 &= \lambda v_3 \\ &\vdots \\ p_{m-1} v_{m-1} &= \lambda v_m. \end{aligned}$$

From these equations, we have $v_2 = \left(\frac{1}{\lambda}\right) p_1$ and $v_3 = \left(\frac{1}{\lambda}\right) p_2 v_2$, so $v_3 = \left(\frac{1}{\lambda}\right)^2 p_1 p_2$. In general,

$$v_k = \left(\frac{1}{\lambda}\right)^{k-1} p_1 p_2 \ldots p_{k-1}, \quad 2 \leq k \leq m.$$

The first components of Equation 18.5 must be equal, so

$$f_1 + f_2 v_2 + \cdots + f_m v_m = \lambda,$$

and

$$f_1 + f_2 \left[\left(\frac{1}{\lambda}\right) p_1\right] + f_3 \left[\left(\frac{1}{\lambda}\right)^2 p_1 p_2\right] + \cdots + f_k \left[\left(\frac{1}{\lambda}\right)^k p_1 p_2 \ldots p_k\right] + \cdots + f_m \left[\left(\frac{1}{\lambda}\right)^{m-1} p_1 p_2 \ldots p_{m-1}\right] = \lambda.$$

Divide by λ to obtain

$$\left(\frac{1}{\lambda}\right) f_1 + f_2 \left[\left(\frac{1}{\lambda}\right)^2 p_1\right] + f_3 \left[\left(\frac{1}{\lambda}\right)^3 p_1 p_2\right] + \cdots + f_k \left[\left(\frac{1}{\lambda}\right)^{k+1} p_1 p_2 \ldots p_k\right] + \cdots + f_m \left[\left(\frac{1}{\lambda}\right)^m p_1 p_2 \ldots p_{m-1}\right] = 1. \tag{18.6}$$

Let $l_1 = 1$ and for $k = 2, 3, \ldots, m$, define $l_k = p_1 p_2 \ldots p_{k-1}$. Then Equation 18.6 becomes

$$f_1 l_1 \lambda^{-1} + f_2 l_2 \lambda^{-2} + f_3 l_3 \lambda^{-3} + \cdots + \lambda^{-m} f_m l_m = 1.$$

Using summation notation, we have

$$\sum_{k=1}^{m} f_k l_k \lambda^{-k} - 1 = 0, \tag{18.7}$$

which is known as the *Euler-Lotka equation*. The value l_k is the fraction of 1-year olds that survive to age k. The characteristic equation can be derived by evaluating $det\,(L-\lambda I)$ using expansion by minors or the elementary row operation of multiplying a row by a constant and subtracting from another row, and the result is

$$\lambda^m\left(1-\left[\frac{f_1l_1}{\lambda}+\frac{f_2l_2}{\lambda^2}+\ldots+\frac{f_ml_m}{\lambda^m}\right]\right)=0,\lambda\neq 0.$$

Both the Euler-Lotka equation and the characteristic equation have the same roots. Let

$$f(\lambda)=\sum_{k=1}^{m}f_kl_k\lambda^{-k}-1=\frac{f_1l_1}{\lambda}+\frac{f_2l_2}{\lambda^2}+\ldots+\frac{f_ml_m}{\lambda^m}-1,$$

and we have

$$f'(\lambda)=-\frac{f_1l_1}{\lambda^2}-\frac{2f_2l_2}{\lambda^3}-\ldots-\frac{mf_ml_m}{\lambda^{m+1}}<0,\ \lambda>0.$$

The function f is decreasing for $\lambda>0$, and we know there is a value of λ such that $f(\lambda)=0$. The second derivative

$$f''(\lambda)=\frac{2f_1l_1}{\lambda^3}+\frac{6f_2l_2}{\lambda^4}+\ldots+\frac{m(m+1)f_ml_m}{\lambda^{m+2}}>0,\ \lambda>0,$$

so f is concave upward, and there is one positive real root, λ_1, the dominant eigenvalue. All the other eigenvalues are negative or imaginary.

If we assume that L has m distinct eigenvalues, then there is a basis of eigenvectors $u_1,u_2,\ldots,u_m$ such that

$$L=UDU^{-1},$$

where $U=(u_1,u_2,\ldots,u_m)$ and $D=\begin{bmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & \lambda_m\end{bmatrix}$ is the matrix of eigenvalues. As a result (Section 5.3.1),

$$L^k=U\begin{bmatrix}\lambda_1^k & 0 & \ldots & 0\\ 0 & \lambda_2^k & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & \lambda_n^k\end{bmatrix}U^{-1}$$

From Equation 18.4,

$$N(t+k)=U\begin{bmatrix}\lambda_1^k & 0 & \ldots & 0\\ 0 & \lambda_2^k & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & \lambda_n^k\end{bmatrix}U^{-1}N(t).$$

If we let $U^{-1}N(t)=c(t)$, then

$$N(t+k)=U\begin{bmatrix}\lambda_1^k & 0 & \ldots & 0\\ 0 & \lambda_2^k & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & \lambda_m^k\end{bmatrix}c(t),$$

and

$$N(t+k)=c_1(t)\lambda_1^ku_1+c_2(t)\lambda_2^ku_2+\cdots+c_m(t)\lambda_m^ku_m.$$

Since λ_1 is the largest eigenvalue in magnitude,

$$N(t+k)\approx c_1(t)\lambda_1^ku_1.$$

This means that as time increases, the age distribution vector tends to a scalar multiple of the eigenvector associated with the largest eigenvalue of the Leslie matrix. In other words, at equilibrium the proportion of individuals belonging to each age class will remain constant, and the number of individuals will increase by λ_1 times each period. Divide the eigenvector, u_1, with all positive entries, associated with λ_1 by the sum of its components $\left(u_1=\left(\frac{1}{\sum_{i=1}^{m}u_{1i}}\right)u_1\right)$. The components of the new vector have the same relative proportions as those in the original eigenvector, and they determine the percentage of females in each age class after an extended period of time.

Example 18.2. Suppose a population has four age classes, and that the following table specifies the data for the population.

Age Class	n_i (0)	f_i	p_i
1	8	0	0.60
2	10	6	0.45
3	12	3	0.25
4	7	2	

The Leslie matrix for this population is

$$L = \begin{bmatrix} 0 & 6 & 3 & 2 \\ 0.60 & 0 & 0 & 0 \\ 0 & 0.45 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \end{bmatrix}$$

Using MATLAB, find the eigenvalues and eigenvectors of L.

```
>> [U D] = eig(L);
>> diag(D)

ans =
       2.0091
      -1.7857
     -0.11171 +    0.15858i
     -0.11171 -    0.15858i

>> u1 = U(:,1);
>> u1 = (u1/sum(u1))*100

u1 =
       72.788
       21.737
       4.8687
      0.60582
```

The dominant eigenvalue is 2.0091, so the number of individuals at equilibrium will increase by 2.0091 each time period. The percentage of females in each age class after an extended period of time is given by the components of `u1`. Now, we will look at the situation graphically. Let $n_0 = \begin{bmatrix} 8 & 10 & 12 & 7 \end{bmatrix}^T$, and compute the age distribution vector over a 10-year period. Store the initial distribution in column 1 of a 4×11 matrix N, and use Equation 18.5 to compute vectors of the age distribution for years 1-10. The vectors grow exponentially, so graph the results for each class using a logarithmic scale of the vertical (population) axis (Figure 18.4).

```
>> format shortg
>> n0 =[8 10 12 7]';
>> N = zeros(4,11);
>> N(:,1) = n0;
>> for k = 2:11
      N(:,k) = L*N(:,k-1);
   end
>> t = 0:10;
>> semilogy(t,N);
>> xlabel('Time');
>> ylabel('log_{10}(population)');
>> legend('Age class 1', 'Age class 2', 'Age class 3', 'Age class 4',...
          'Location','NorthWest');
```

■

18.1.3 Buckling of a Column

This example shows that eigenvalues can be associated with functions as well as matrices.

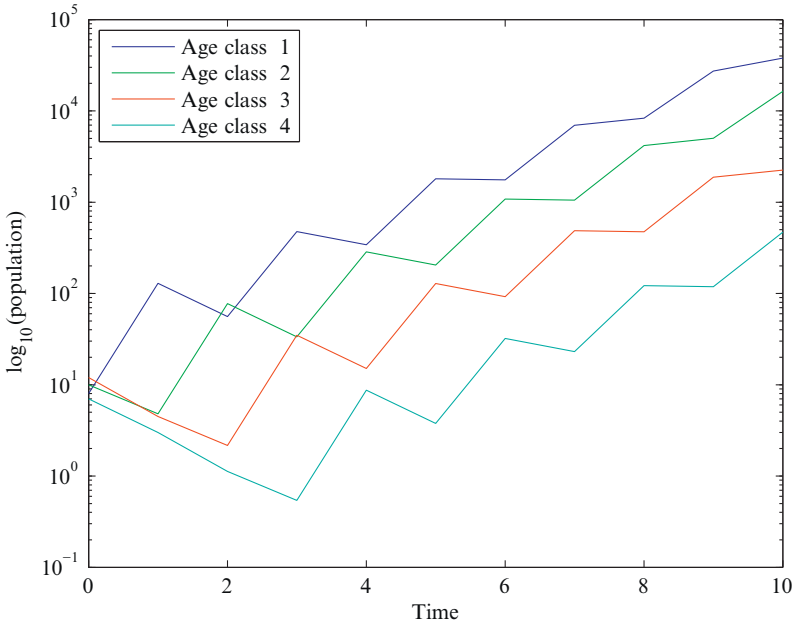

FIGURE 18.4 Populations using the Leslie matrix.

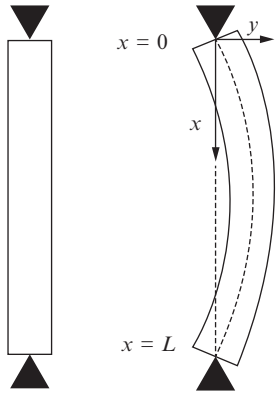

FIGURE 18.5 Column buckling.

Apply a compressive axial force, or load, P, to the top of a vertical thin elastic column of uniform cross-section having length L (Figure 18.5). The column will buckle, and the deflection $y(x)$ satisfies the differential equation

$$EI\frac{\mathrm{d}^2y}{\mathrm{d}x^2} = -Py,$$

where E is Young's modulus of elasticity and I is the area moment of inertia of column cross-section. Assume that the column is hinged at both ends so that $y(0) = y(L) = 0$, and we have the boundary value problem

$$EI\frac{\mathrm{d}^2y}{\mathrm{d}x^2} + Py = 0, \quad y(0) = 0,\ y(L) = 0.$$

Note that $y = 0$ is a solution to the problem, and this corresponds to the situation where the load P is not large enough to cause deflection. We wish to determine values of P that cause the column to buckle; in other words, for what values of P does the boundary value problem have nontrivial solutions?

Let $\lambda = \frac{P}{EI}$, and we have the problem

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} + \lambda y = 0, \quad y(0) = 0,\ y(L) = 0.$$

Let $\lambda = \beta^2$, $\beta > 0$. The equation is homogeneous, so try a solution of the form $y = e^{px}$. After substitution into the equation, we have

$$p^2 + \beta^2 = 0,$$

and $p = \pm\beta i$, yielding complex solutions $e^{\pm\beta ix}$. These give rise to the real solutions $y = c_1 \cos\beta x + c_2 \sin\beta x$. Apply the boundary conditions:

$$\begin{aligned} y(0) &= c_1 = 0, \\ y(L) &= c_2 \sin\beta L = 0. \end{aligned}$$

If $\sin\beta L = 0$, we can choose c_2 to be any nonzero value, so let $c_2 = 1$. We have $\sin\beta L = 0$ when $\beta L = n\pi$, $n = 1, 2, 3, \ldots$, and so our values of β must be

$$\beta_n = \frac{n\pi}{L}, \quad n = 1, 2, 3, \ldots.$$

Since $\lambda = \beta^2$, it follows that:

$$\lambda_1 = \frac{\pi^2}{L^2}, \quad \lambda_2 = \frac{4\pi^2}{L^2}, \quad \lambda_3 = \frac{9\pi^2}{L^2}, \ \ldots, \ \lambda_n = \frac{n^2\pi^2}{L^2}, \ldots$$

and the sequence of functions

$$y_n(x) = k \sin\left(\frac{n\pi}{L}x\right), \quad n = 1, 2, 3, \ldots$$

where k is a constant are nontrivial solutions to the boundary value problem. These functions are called *eigenfunctions* with corresponding eigenvalues λ_n. For our column buckling equation, we have eigenvalues $\lambda_n = \frac{P_n}{EI} = \frac{n^2\pi^2}{L^2}$, $n = 1, 2, 3, \ldots$, and loads

$$P_n = \frac{EI\pi^2 n^2}{L^2}, \quad n = 1, 2, 3, \ldots$$

The column will buckle only when the compressive force is one of these values. This sequence of forces are called *critical loads*. The deflection function corresponding to the smallest critical load $P_1 = \frac{EI\pi^2}{L^2}$ is termed the *first buckling mode*.

Example 18.3. For a thin 2.13 m column of aluminum, $E = 69 \times 10^9\ \mathrm{N/m^2}$, and $I = 3.26 \times 10^{-4}\ \mathrm{m^4}$. Compute the critical loads and graph the deflection curves for $n = 1, 2, 3$ (Figure 18.6).

```
The first buckling mode = 4.89336e+07
The second critical load = 1.95734e+08
The third critical load = 4.40402e+08
```

If the column has a physical restraint on it at $x = L/2$, then the smallest critical load will be $P_2 = 1.69 \times 10^7$and the deflection curve is $\sin\left(\frac{2\pi x}{L}\right)$. If restraints are placed on the column at $x = L/3$ and at $2L/3$, then the column will not buckle until the column is subjected to the critical load $P_3 = 3.81 \times 10^7$, and the deflection curve is $\sin\left(\frac{3\pi x}{L}\right)$ [53, pp. 167-169]. ■

Remark 18.1. See Problem 18.43 for another approach to the buckling problem.

18.2 COMPUTATION OF SELECTED EIGENVALUES AND EIGENVECTORS

In some applications, it is only necessary to compute a few of the largest or the smallest eigenvalues and their corresponding eigenvectors. Examples include:

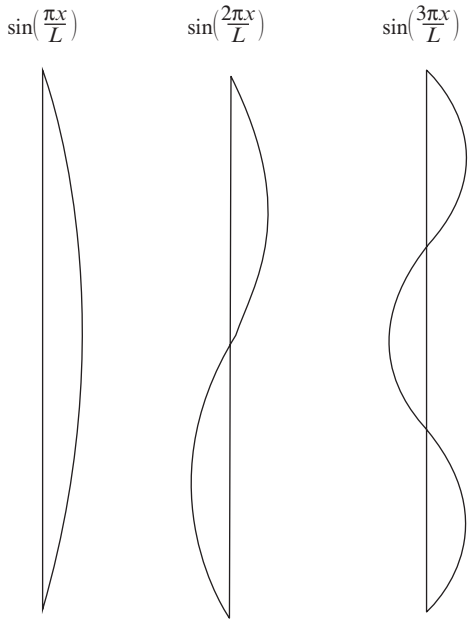

FIGURE 18.6 Deflection curves for critical loads P_1, P_2, and P_3.

- The Leslie matrix.
- The buckling problem. The most important eigenvalue is the smallest.
- Vibration of structures. The most important eigenvalues are a few of the smallest ones.
- Statistical applications. Only the first few of the largest eigenvalues need to be computed.

The power and inverse power methods compute the largest and smallest eigenvector, respectively. If v is an eigenvector of A, then $Av = \lambda v$, so $\langle Av, v\rangle = \lambda \langle v, v\rangle$, and

$$\lambda = \frac{v^{\mathrm{T}}(Av)}{v^{\mathrm{T}}v}. \tag{18.8}$$

Equation 18.8 is called the *Rayleigh quotient*. Given any eigenvector of A, Equation 18.8 computes the corresponding eigenvalue. This will be useful to us, since the power and inverse power methods compute an eigenvector, and the Rayleigh quotient finds the corresponding eigenvalue.

If the eigenvalues of an $n \times n$ matrix A are such that

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| > \cdots \geq |\lambda_n|, \tag{18.9}$$

the eigenvalue λ_1 is said to be the *dominant eigenvalue* of A. Thus, if we know that v_{d} is an eigenvector of the dominant eigenvalue, the eigenvalue is $\lambda_1 = \frac{v_{\mathrm{d}}^{\mathrm{T}}(Av_{\mathrm{d}})}{v_{\mathrm{d}}^{\mathrm{T}}v_{\mathrm{d}}}$. Not all matrices have a dominant eigenvalue. For instance, let $A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. Its eigenvalues are $\lambda_1 = 1$ and $\lambda_2 = -1$. The matrix $B = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ has eigenvalues $\lambda_1 = \lambda_2 = 2, \lambda_3 = 1$.

18.2.1 Additional Property of a Diagonalizable Matrix

Theorem 5.3 states that if the $n \times n$ matrix A has n linearly independent eigenvectors $v_1, v_2, \ldots, v_n$, then A can be diagonalized by the matrix the *eigenvector matrix* $X = (v_1 v_2 \ldots v_n)$. The converse of Theorem 5.3 is also true; that is, if a matrix can be diagonalized, it must have n linearly independent eigenvectors. We need this result for the purposes of developing the power method in Section 18.2.2.

Theorem 18.1. *If A is a real $n \times n$ matrix that is diagonalizable, it must have n linearly independent eigenvectors.*

Proof. We know there is an invertible matrix V such that $V^{-1}AV = D$, where $D = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$ is a diagonal matrix, and let $v_1, v_2, \ldots, v_n$ be the columns of V. Since V is invertible, the v_i are linearly independent. The relationship $V^{-1}AV = D$ gives $AV = VD$, and using matrix column notation we have

$$A \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix} = \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}.$$

Column i of $A \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix}$ is Av_i, and column i of $\begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$ is $\lambda_i v_i$, so $Av_i = \lambda_i v_i$.

Thus, the linearly independent set $v_1, v_2, \ldots, v_n$ are eigenvectors of A corresponding to eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. □

18.2.2 The Power Method for Computing the Dominant Eigenvalue

We now develop the *power method*, a simple iteration, for computing the dominant eigenvalue of a matrix, if it has one. Make an initial guess for the eigenvector, usually $v_0 = \begin{bmatrix} 1 & \ldots & 1 \end{bmatrix}^{\mathrm{T}}$, and normalize it by assigning $v_0 = \frac{v_0}{\|v_0\|_2}$. Compute $v_1 = Av_0$, and then normalize v_1. Repeat this process until satisfying a convergence criterion. Example 18.4 demonstrates this algorithm.

Example 18.4. Let $A = \begin{bmatrix} 1 & 3 & 0 \\ 2 & 5 & 1 \\ -1 & 2 & 3 \end{bmatrix}$, $x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

$k = 1$

$$x_1 = Ax_0 = \begin{bmatrix} 4 \\ 8 \\ 4 \end{bmatrix}, \quad x_1 = x_1 / \|x_1\|_2 = \begin{bmatrix} 0.4082 \\ 0.8165 \\ 0.4082 \end{bmatrix}$$

$k = 2$

$$x_2 = Ax_1 = \begin{bmatrix} 2.8577 \\ 5.3072 \\ 2.4495 \end{bmatrix}, \quad x_2 = x_2 / \|x_2\|_2 = \begin{bmatrix} 0.4392 \\ 0.8157 \\ 0.3765 \end{bmatrix}$$

$k = 3$

$$x_3 = Ax_2 = \begin{bmatrix} 2.8863 \\ 5.3334 \\ 2.3216 \end{bmatrix}, \quad x_3 = x_3 / \|x_3\|_2 = \begin{bmatrix} 0.4445 \\ 0.8213 \\ 0.3575 \end{bmatrix}$$

$k = 4$

$$x_4 = Ax_3 = \begin{bmatrix} 2.9085 \\ 5.3532 \\ 2.2708 \end{bmatrix}, \quad x_4 = x_4 / \|x_4\|_2 = \begin{bmatrix} 0.4473 \\ 0.8233 \\ 0.3493 \end{bmatrix}$$

$$\lambda_1 = \frac{(Ax_4) \cdot x_4}{x_4 \cdot x_4} = 6.5036$$

The eigenvalues of A are 6.5050, -0.4217, 2.9166, so the relative error in computing the largest eigenvalue is 2.1856×10^{-4}. ■

In Algorithm 18.1 (the power method), we assume that the real matrix A is diagonalizable. Theorem 18.1 guarantees there exists a basis of eigenvectors for $\mathbb{R}^n$. This assumption will allow us to prove the power method converges under condition 18.9. The convergence test is to compute the Rayleigh quotient after normalization ($\lambda_k = x_k^{\mathrm{T}}(Ax_k)$) at each iteration and determine if $\|Ax_k - \lambda x_k\|_2 < \text{tol}$; in other words, is λx_k sufficiently close to Ax_k?

Algorithm 18.1 The Power Method

```
function LARGEEIG(A,x0,tol,numiter)
  % Use the power method to find the dominant eigenvalue and
  % the corresponding eigenvector of real diagonalizable matrix A.
  % [lambda x iter] = largeeig(A,x0,n,tol) computes the largest eigenvalue
  % lambda in magnitude and corresponding eigenvector x of real matrix A.
  % x0 is the initial approximation, tol is the desired error tolerance,
  % and maxiter is the maximum number of iterations to perform.
  % If the algorithm converges, iter contains the number of iterations required.
  % If the method does not converge, iter=-1.

  x0 = x0/||x0||2
  for k=1:numiter do
    xk = Axk-1
    xk = xk/||xk||2
    λ = xkTAxk
    error=||Axk − λxk||2
    if error<tol then
      x = xk
      iter=k
      return [λ, x, iter]
    end if
  end for
  x=xk
  iter=-1
  return [λ, x, iter]
end function
```

NLALIB: The function `largeeig` implements Algorithm 18.1.
Theorem 18.2 specifies conditions under which the power iteration is guaranteed to converge.

Theorem 18.2. *Assume that A is diagonalizable, with real eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and associated real eigenvectors $v_1, v_2, \ldots, v_n$, and that λ_1 is a simple eigenvalue with the largest magnitude, i.e.,*

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n| . \tag{18.10}$$

Then, the power method converges to a normalized eigenvector corresponding to eigenvalue λ_1.

Proof. Let x_0 be the initial approximation. By Theorem 18.1, the eigenvectors are linearly independent, and so $x_0 = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$. We can assume that $c_1 \neq 0$, or otherwise rearrange the linear combination so it is. Now,

$$\begin{aligned}
x_1 &= Ax_0, \quad x_1 = \frac{Ax_0}{\|Ax_0\|_2} \\
x_2 &= A\left(\frac{Ax_0}{\|Ax_0\|_2}\right) = \frac{A^2x_0}{\|Ax_0\|_2}, \quad x_2 = \frac{A^2x_0}{\|Ax_0\|_2}\left(\frac{\|Ax_0\|_2}{\|A^2x_0\|_2}\right) = \frac{A^2x_0}{\|A^2x_0\|_2} \\
x_k &= \frac{A^kx_0}{\|A^kx_0\|_2}, \quad \ldots
\end{aligned} \tag{18.11}$$

Since $x_0 = c_1v_1 + c_2v_2 + \cdots + c_nv_n$,

$$\begin{aligned} A^k x_0 &= A^k (c_1v_1 + c_2v_2 + \cdots + c_nv_n) = c_1\lambda_1^k v_1 + c_2\lambda_2^k v_2 + \cdots + c_n\lambda_n^k v_n \\ &= \lambda_1^k \left(c_1v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + c_3 \left(\frac{\lambda_3}{\lambda_1}\right)^k v_3 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k v_n \right). \end{aligned} \tag{18.12}$$

By substituting Equation 18.12 into Equation 18.11, we obtain

$$\begin{aligned} x_k &= \frac{\lambda_1^k \left(c_1v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + c_3 \left(\frac{\lambda_3}{\lambda_1}\right)^k v_3 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k v_n \right).}{\left\| \lambda_1^k \left(c_1v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + c_3 \left(\frac{\lambda_3}{\lambda_1}\right)^k v_3 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k v_n \right) \right\|_2} = \\ &\pm \frac{\left(c_1v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + c_3 \left(\frac{\lambda_3}{\lambda_1}\right)^k v_3 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k v_n \right).}{\left\| \left(c_1v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + c_3 \left(\frac{\lambda_3}{\lambda_1}\right)^k v_3 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k v_n \right) \right\|_2}. \end{aligned} \tag{18.13}$$

Because $c_1 \neq 0$, the denominator in Equation 18.13 is never 0. Since $|\lambda_1| > |\lambda_i|$, $i \geq 2$,

$$\lim_{k\to\infty} x_k = \pm \frac{v_1}{\|v_1\|_2},$$

a normalized eigenvector of A corresponding to the largest eigenvalue λ_1. □

Remark 18.2.

a. If we assume that $|\lambda_1| > |\lambda_2| > |\lambda_3| > \cdots > |\lambda_n|$, then Theorem 5.2 guarantees that there are n linearly independent eigenvectors. Our assumption that A is diagonalizable allows us to require only condition 18.9.

b. The assumption that $c_1 \neq 0$ is critical to the proof. A randomly chosen x_0 will normally guarantee this with high probability.

c. From the proof of Theorem 18.2, we see that the rate of convergence is determined by the ratio $\left|\frac{\lambda_2}{\lambda_1}\right|$, where λ_2 is the second largest eigenvalue in magnitude. The power method will converge quickly if $\left|\frac{\lambda_2}{\lambda_1}\right|$ is small and slowly if $\left|\frac{\lambda_2}{\lambda_1}\right|$ is close to 1.

Example 18.5.

a. Estimate the largest eigenvalue and the corresponding eigenvector for the Leslie matrix $L = \begin{bmatrix} 0 & 6 & 3 & 2 \\ 0.60 & 0 & 0 & 0 \\ 0 & 0.45 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \end{bmatrix}$.

The MATLAB code runs `largeeig` twice with an error tolerance of 1.0×10^{-10}. The first call uses `maxiter = 100`, and the iteration failed to attain the tolerance. With `maxiter = 300` for the second call, the tolerance was attained in 200 iterations. The two largest eigenvalues of L in magnitude are 2.0091 and 1.7857, and the ratio $\frac{\lambda_2}{\lambda_1} = 0.88879$, so we expect slow convergence.

```
>> x0 = ones(4,1);
>> [lambda x iter] = largeeig(L,x0,1.0e-10,100)

lambda =
       2.0091
x =
      0.95619
      0.28556
     0.063958
    0.0079585

iter =
    -1
```

```
>> [lambda x iter] = largeeig(L,x0,1.0e-10,300)
lambda =
       2.0091
x =
      0.95619
      0.28555
     0.063958
    0.0079584
iter =
   200
```

b. Let

$$A = \begin{bmatrix} 10.995 & -1.6348 & 1.2323 & 3.055 \\ 2.2256 & 5.4029 & 1.3355 & 5.4342 \\ 1.5794 & 1.1108 & 2.2963 & -0.46759 \\ -3.7347 & -2.447 & -1.3122 & -2.6939 \end{bmatrix}.$$

The eigenvalues of A to eight significant figures are

$$\{10.000373, 3.0000000, 2.0000200, 0.99991723\},$$

and the ratio $\frac{\lambda_2}{\lambda_1} = 0.29998782$, so the power method should converge reasonably quickly. In fact, after 27 iterations, the approximate eigenvalue is 10.000373. ■

18.2.3 Computing the Smallest Eigenvalue and Corresponding Eigenvector

Assume that A is nonsingular and has eigenvalue λ with corresponding eigenvector v. If $Av = \lambda v$, then $A^{-1}v = \frac{1}{\lambda}v$ and $\frac{1}{\lambda}$ is an eigenvalue of A^{-1} with corresponding eigenvector v. (Recall that an invertible matrix cannot have a zero eigenvalue.) Thus, the eigenvalues of A^{-1} are the reciprocals of those for A, but the eigenvectors are the same. If we assume that

$$|\lambda_1| \geq |\lambda_2| \geq |\lambda_3| > \cdots \geq |\lambda_{n-1}| > |\lambda_n|,$$

the smallest eigenvalue of A is the largest eigenvalue of A^{-1}. The power method applied to A^{-1} will yield the smallest eigenvalue of A and is termed the *inverse power method*.

As we have discussed, computing A^{-1} is fraught with problems, so we should avoid the iteration $x_k = A^{-1}x_{k-1}$; however, this is equivalent to solving $Ax_k = x_{k-1}$. Use Gaussian elimination to compute $PA = LU$ and solve the linear systems $Ax_k = x_{k-1}$ using forward and backward substitution. As the iteration progresses, x_k approaches a normalized eigenvector corresponding to eigenvalue $\frac{1}{\lambda_n}$, so $A^{-1}x_k \approx \left(\frac{1}{\lambda_n}\right)x_k$. Therefore, $Ax_k \approx \lambda_n x_k$ and x_k are approximate eigenvectors for the smallest eigenvalue, λ_n, of A. If we keep track of the Rayleigh quotient $\lambda_k = x_k^{\mathrm{T}}Ax_k$, then we can apply the convergence test $\|Ax_k - \lambda_k x_k\|_2 < \text{tol}$, at each step and the algorithm is the same as the power method with $x_k = Ax_{k-1}$ replaced by $x_k = \text{lusolve}\,(L, U, P, x_{k-1})$. The function `smalleig` in the software distribution implements the inverse power method.

Example 18.6. This problem constructs a matrix with known eigenvalues by generating a random integer matrix, zeroing out all the elements from the diagonal and below using triu, and then adding a diagonal matrix containing the eigenvalues. There is an eigenvalue, 9, of largest magnitude with multiplicity three, and a smallest eigenvalue in magnitude, -1, well separated from the others. As expected, the power method fails, and the inverse power method is successful.

```
>> A = randi([-100,100],10,10);
>> A = triu(A,1);
>> d = [-1 3 5 9 9 9 6 2 -7 4]';
>> A = A + diag(d);
>> iterations = [50 100 1000 10000];
>> for i = 1:length(iterations)
[lambda,v,iter] = largeeig(A,ones(10,1),1.0e-10,iterations(i));
lambda
```

```
iter
end

lambda =
   9.471799775358845
iter =
    -1

lambda =
   9.204649150708208
iter =
    -1

lambda =
   9.018221954378124
iter =
    -1

lambda =
   9.001802197342364
iter =
    -1

>> [lambda,v,iter] = smalleig(A,rand(10,1),1.0e-14,75);
>> lambda

lambda =
  -1.000000000000044

>> iter

iter =
    52
```

■

18.3 THE BASIC *QR* ITERATION

The *QR* method in its various forms is the most-used algorithm for computing all the eigenvalues of a matrix. We will consider a real matrix A whose eigenvalues satisfy

$$|\lambda_1| > |\lambda_2| > |\lambda_3| > \cdots > |\lambda_n| > 0 \tag{18.14}$$

Such a matrix cannot have imaginary eigenvalues, since each member of a complex conjugate pair has the same modulus. For in-depth coverage of the general eigenvalue problem, including matrices with entries having a nonzero imaginary part the interested reader should see Refs. [2, 9]. A real matrix satisfying Equation 18.14 is invertible, since Theorem 5.2 implies the eigenvectors are linearly independent. In addition, Theorem 5.4 guarantees that A is diagonalizable.

We will first discuss what is termed the *basic QR iteration* for computing all the eigenvalues of a matrix satisfying Equation 18.14. This algorithm should be viewed as a starting point, and we will discuss techniques to enhance its performance in later sections. The basic *QR* iteration is very simple to perform. Let $A_0 = A$, apply the *QR* decomposition to A_0 and obtain $A_0 = Q_1R_1$, and form $A_1 = R_1Q_1$. Now compute $A_1 = Q_2R_2$ and let $A_2 = R_2Q_2$. Continuing in this fashion, we obtain $A_{k-1} = Q_kR_k$ and $A_k = R_kQ_k$. Under the correct conditions, the sequence approaches an upper triangular matrix. What is on the diagonal of this upper triangular matrix? Since $A = A_0 = Q_1R_1$ and $A_1 = R_1Q_1$, it follows that $A_1 = Q_1^{\mathrm{T}}AQ_1$. Now, $A_1 = Q_2R_2$ and $A_2 = R_2Q_2$, so $A_2 = Q_2^{\mathrm{T}}A_1Q_2 = Q_2^{\mathrm{T}}Q_1^{\mathrm{T}}AQ_1Q_2$. If we continue this process k times, we have

$$\begin{aligned} A_k &= Q_k^{\mathrm{T}}Q_{k-1}^{\mathrm{T}}\cdots Q_2^{\mathrm{T}}Q_1^{\mathrm{T}}AQ_1Q_2\cdots Q_{k-1}Q_k \\ &= (Q_1Q_2\cdots Q_{k-1}Q_k)^{\mathrm{T}}\,A\,(Q_1Q_2\cdots Q_{k-1}Q_k) \\ &= \overline{Q}_k^{\mathrm{T}}A\overline{Q}_k, \end{aligned}$$

where $\overline{Q}_k = Q_1 Q_2 \dots Q_k$ is an orthogonal matrix. A and A_k are similar matrices, and by Theorem 5.1 have the same eigenvalues. Since A_k is approximately an upper triangular matrix, estimates for its eigenvalues lie on the diagonal. Here is an outline of the basic QR algorithm.

Outline of the Basic *QR* Iteration

```
A0 = A
for k = 1, 2, ... do
   Ak−1 = QkRk
   Ak = RkQk
end for
```

- Compute the QR decomposition, multiply the factors Q and R together in the reverse order RQ, and repeat.
- Under suitable conditions, this simple process converges to an upper triangular matrix, so if k is sufficiently large,

$$A_k \approx \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1,n-1} & r_{1n} \\ & r_{22} & \dots & r_{2,n-1} & r_{2n} \\ & & \ddots & \ddots & \vdots \\ & & & r_{n-1,n-1} & \vdots \\ & & & & r_{nn} \end{bmatrix}.$$

- The eigenvalues of A are approximately $r_{11}, r_{22}, \dots, r_{nn}$.

Remark 18.3. It can be shown that the eigenvalues occur on the diagonal in decreasing order of magnitude, so in the outline of the basic QR iteration $\lambda_1 = r_{11}, \lambda_2 = r_{22}, \dots, \lambda_n = r_{nn}$.

Example 18.7. Let $A = \begin{bmatrix} 5 & 1 & 4 \\ -1 & 3 & 1 \\ 3 & -1 & 2 \end{bmatrix}$. Run the following MATLAB script.

```
>> B = A;
>> for i = 1:15
      [Q R] = qr(A);
      A = R*Q;
   end
>> A
>> eig(B)
```

Note that A is transformed into an upper triangular matrix with the eigenvalues of A on its diagonal in descending order of magnitude. ■

A proof of the following result can be found in Ref. [27, pp. 209-212]. Recall when reading the theorem that a real matrix satisfying Equation 18.14 can be diagonalized. Also, it is assumed that the LU decomposition can be done without row exchanges.

Theorem 18.3. *Let A be a real matrix satisfying Equation 18.14. Assume also that P^{-1} has an LU decomposition, where P is the matrix of eigenvectors of A, i.e., $A = P\text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n) P^{-1}$. Then the sequence $\{A_k\}$, $k \geq 1$, generated by the basic QR iteration, converges to an upper triangular matrix whose diagonal entries are the eigenvalues of A.*

The implementation of the basic QR iteration is in the function `eigqrbasic` of the software distribution.

18.4 TRANSFORMATION TO UPPER HESSENBERG FORM

Prior to presenting a more effective approach to eigenvalue computation than the basic QR iteration, we need two definitions.

Definition 18.1. An *orthogonal similarity transformation* is a decomposition of form $B = P^{\mathrm{T}}AP$, where P is an orthogonal matrix. Since $P^{\mathrm{T}} = P^{-1}$, matrices A and B are similar and thus have the same eigenvalues.

We will have many occasions to use Hessenberg matrices in this and the remaining chapters because they have a simpler form than a general matrix.

Definition 18.2. A square matrix H is *upper Hessenberg* if $h_{ij} = 0$ for all $i > j+1$. The transpose of an upper Hessenberg matrix is a *lower Hessenberg* matrix ($h_{ij} = 0$ for all $j > i+1$). The upper and lower Hessenberg matrices are termed "almost triangular." For instance, the matrix

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

is upper Hessenberg, and

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

is lower Hessenberg.

The basic QR iteration is not practical for large matrices. It requires $O\left(n^3\right)$ flops for each QR decomposition, so k iterations cost $kO\left(n^3\right)$ flops. If $k = n$, the algorithm becomes $O\left(n^4\right)$. Our approach is to split the problem into two parts:

a. Transform A into an upper Hessenberg matrix using orthogonal similarity transformations.
b. Make use of the simpler form of an upper Hessenberg matrix to quickly and accurately compute the eigenvalues.

We will develop an algorithm, hhess, that creates the orthogonal similarity transformation $H = P^{\mathrm{T}}AP$, where H is an upper Hessenberg matrix. Theorem 5.1 guarantees that H will have the same eigenvalues as A. We will then convert the upper Hessenberg matrix into upper triangular form using Givens rotations, and the eigenvalues of A will be on the diagonal.

The transformation to upper Hessenberg form uses Householder reflections. The idea is similar to the use of Householder reflections to perform the QR decomposition, but we must leave a diagonal below the main diagonal (a subdiagonal), and a similarity transformation must be used to maintain the same eigenvalues; in other words, we must use a transformation $A_k = H_{u_k}A_{k-1}H_{u_k}^{\mathrm{T}}$ rather than $A_k = H_{u_k}A_{k-1}$. Assuming a 5×5 matrix, the process proceeds as follows:

$$A = A_0 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}, A_1 = H_{u_1}AH_{u_1}^{\mathrm{T}} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}$$

$$A_2 = H_{u_2}A_1H_{u_2}^{\mathrm{T}} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix}, A_3 = H_{u_3}A_2H_{u_3}^{\mathrm{T}} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

In each case, $H_{u_i}A_{i-1}$ zeroes out the elements at indices $(i+2,i),(i+3,i),\ldots,(n,i)$ of A_{i-1}. After $k-1$ steps,

$$A_{k-1}=\begin{bmatrix} * & * & * & * & \ldots & * & * & * \\ * & * & * & * & \cdots & * & * & * \\ 0 & * & \ddots & * & \ldots & * & * & * \\ 0 & 0 & \ddots & \overline{a_{k-1,k-1}} & * & \ldots & * & * \\ \vdots & \vdots & \ddots & \overline{a_{k,k-1}} & \overline{a_{kk}} & \ldots & * & * \\ 0 & 0 & \ddots & 0 & \overline{a_{k+1,k}} & \ldots & * & * \\ \vdots & \vdots & \cdots & \vdots & \overline{a_{k+2,k}} & \ldots & * & * \\ 0 & 0 & \ldots & 0 & \vdots & \ldots & * & * \\ 0 & 0 & \ldots & 0 & \overline{a_{nk}} & \ldots & * & * \end{bmatrix}.$$

To find the next Householder matrix, let $y=\begin{bmatrix} \overline{a_{k+1,k}} \\ \overline{a_{k+2,k}} \\ \vdots \\ \overline{a_{nk}} \end{bmatrix}$. Determine the $(n-k)\times(n-k)$ Householder matrix $\widetilde{H}_{u_k}$ so that $\widetilde{H}_{u_k}y=\begin{bmatrix} \hat{a}_{k+1,k} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ by using the techniques of Section 17.8.1. We must develop a matrix H_{u_k} containing the submatrix $\widetilde{H}_{u_k}$ such that when forming $H_{u_k}A_{k-1}$ the elements $\begin{bmatrix} \overline{a_{k+1,k}} \\ \overline{a_{k+2,k}} \\ \vdots \\ \overline{a_{nk}} \end{bmatrix}$ become $\begin{bmatrix} \hat{a}_{k+1,k} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ without destroying any work we have already done. We choose Householder matrices H_{u_k} of the form

$$H_{u_k}=\begin{bmatrix} I_k & 0 \\ 0 & \widetilde{H}_{u_k} \end{bmatrix}, \tag{18.15}$$

where I_k is the $k\times k$ identity matrix, and $\widetilde{H}_{u_k}$ is the $n-k$ Householder matrix. The submatrix $\begin{bmatrix} I_k \\ 0 \end{bmatrix}$ protects the entries already having their final value, while $\begin{bmatrix} 0 \\ \widetilde{H}_{k_j} \end{bmatrix}$ makes the required modifications. For instance, suppose one elimination step has executed. The matrix A_1 has the form

$$\begin{bmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & \mathbf{X} & X & X & X \\ 0 & \mathbf{X} & X & X & X \\ 0 & \mathbf{X} & X & X & X \end{bmatrix},$$

and we need to zero out the elements at indices $(4,2)$ and $(5,2)$. Let

$$H_{u_2}=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & h_{u_2}^{(11)} & h_{u_2}^{(12)} & h_{u_2}^{(13)} \\ 0 & 0 & h_{u_2}^{(21)} & h_{u_2}^{(22)} & h_{u_2}^{(23)} \\ 0 & 0 & h_{u_2}^{(31)} & h_{u_2}^{(32)} & h_{u_2}^{(33)} \end{bmatrix},$$

and

$$H_{u_2}A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & h_{u_2}^{(11)} & h_{u_2}^{(12)} & h_{u_2}^{(13)} \\ 0 & 0 & h_{u_2}^{(21)} & h_{u_2}^{(22)} & h_{u_2}^{(23)} \\ 0 & 0 & h_{u_2}^{(31)} & h_{u_2}^{(32)} & h_{u_2}^{(33)} \end{bmatrix} \begin{bmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & \mathbf{X} & X & X & X \\ 0 & \mathbf{X} & X & X & X \\ 0 & \mathbf{X} & X & X & X \end{bmatrix} = \begin{bmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & Y & Y & Y & Y \\ 0 & 0 & Y & Y & Y \\ 0 & 0 & Y & Y & Y \end{bmatrix},$$

Verify that forming the product $H_{u_2}A_1H_{u_2}^{\mathrm{T}}$ maintains the zeros created by $H_{u_2}A_1$. Example 18.6 illustrates the process in detail for a 5×5 matrix.

Example 18.8. Let $A = \begin{bmatrix} 9 & 5 & 1 & 2 & 1 \\ 9 & 7 & 10 & 5 & 8 \\ 1 & 7 & 2 & 4 & 3 \\ 4 & 3 & 2 & 10 & 5 \\ 6 & 5 & 4 & 10 & 6 \end{bmatrix}$ and let $A_0 = A$.

Step 1: Compute $\widetilde{H}_{u_1}$ so that $\widetilde{H}_{u_1}\begin{bmatrix} 9 \\ 1 \\ 4 \\ 6 \end{bmatrix} = \begin{bmatrix} -11.5758 \\ 0 \\ 0 \\ 0 \end{bmatrix}$. Using Equation 18.15,

$$H_{u_1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & -0.7775 & -0.0864 & -0.3455 & -0.5183 \\ 0 & -0.0864 & 0.9958 & -0.0168 & -0.0252 \\ 0 & -0.3455 & -0.0168 & 0.9328 & -0.1008 \\ 0 & -0.5183 & -0.0252 & -0.1008 & 0.8489 \end{bmatrix},$$

and

$$H_{u_1}A_0 = \begin{bmatrix} 9.0000 & 5.0000 & 1.0000 & 2.0000 & 1.0000 \\ -11.5758 & -9.6753 & -10.7120 & -12.8716 & -11.3167 \\ 0 & 6.1896 & 0.9934 & 3.1314 & 2.0612 \\ 0 & -0.2417 & -2.0265 & 6.5257 & 1.2448 \\ 0 & 0.1374 & -2.0397 & 4.7886 & 0.3672 \end{bmatrix}$$

$$A_1 = H_{u_1}A_0H_{u_1}^{\mathrm{T}} = \begin{bmatrix} 9.0000 & -5.1832 & 0.5051 & 0.0204 & -1.9695 \\ -11.5758 & 18.7612 & -9.3299 & -7.3435 & -3.0245 \\ 0 & -7.0485 & 0.3500 & 0.5579 & -1.7991 \\ 0 & -2.5371 & -2.1380 & 6.0795 & 0.5754 \\ 0 & -1.7756 & -2.1327 & 4.4167 & -0.1907 \end{bmatrix}.$$

Note that $H_{u_1}A_0$ affected rows 2:5 and columns 1:5, and multiplying on the right by$H_{u_1}^{\mathrm{T}}$ affected rows 1:5, columns 2:5; in other words, the first column of $H_{u_1}A_0$ was not modified, so the work done to zero out the elements at indices $(3, 1)$, $(4, 1)$, and $(5, 1)$ was not modified.

Step 2: Using the vector $\begin{bmatrix} -7.0485 \\ -2.5371 \\ -1.7756 \end{bmatrix}$, form

$$H_{u_2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -0.9155 & -0.3296 & -0.2306 \\ 0 & 0 & -0.3296 & 0.9433 & -0.0397 \\ 0 & 0 & -0.2306 & -0.0397 & 0.9722 \end{bmatrix}$$

from which we obtain

$$H_{u_2}A_1 = \begin{bmatrix} 9.0000 & -5.1832 & 0.5051 & 0.0204 & -1.9695 \\ -11.5758 & 18.7612 & -9.3299 & -7.3435 & -3.0245 \\ 0 & 7.6988 & 0.8760 & -3.5329 & 1.5015 \\ 0 & 0 & -2.0475 & 5.3757 & 1.1433 \\ 0 & 0 & -2.0693 & 3.9241 & 0.2067 \end{bmatrix},$$

$$A_2 = H_{u_2}A_1H_{u_2}^{\mathrm{T}} = \begin{bmatrix} 9.0000 & -5.1832 & -0.0149 & -0.0691 & -2.0321 \\ -11.5758 & 18.7612 & 11.6595 & -3.7325 & -0.4973 \\ 0 & 7.6988 & 0.0160 & -3.6809 & 1.3979 \\ 0 & 0 & -0.1607 & 5.7003 & 1.3704 \\ 0 & 0 & 0.5537 & 4.3754 & 0.5225 \end{bmatrix}.$$

$H_{u_2}A_1$ affected rows 3:5 and columns 2:5, and multiplying on the right by $H_{u_2}^{\mathrm{T}}$ affected rows 1:5, columns 3:5.

Step 3: Compute H_{u_3} using the vector $\begin{bmatrix} -0.1607 \\ 0.5537 \end{bmatrix}$, from which there results

$$H_{u_3}A_2 = \begin{bmatrix} 9.0000 & -5.1832 & -0.0149 & -0.0691 & -2.0321 \\ -11.5758 & 18.7612 & 11.6595 & -3.7325 & -0.4973 \\ 0 & 7.6988 & 0.0160 & -3.6809 & 1.3979 \\ 0 & 0 & 0.5765 & 2.6136 & 0.1200 \\ 0 & 0 & 0 & 6.6938 & 1.4618 \end{bmatrix},$$

$$A_3 = H_{u_3}A_2H_{u_3}^{\mathrm{T}} = \begin{bmatrix} 9.0000 & -5.1832 & -0.0149 & -1.9323 & -0.6326 \\ -11.5758 & 18.7612 & 11.6595 & 0.5625 & -3.7232 \\ 0 & 7.6988 & 0.0160 & 2.3683 & -3.1455 \\ 0 & 0 & 0.5765 & -0.6131 & 2.5435 \\ 0 & 0 & 0 & -0.4614 & 6.8360 \end{bmatrix}.$$

$H_{u_3}A_2$ affected rows 4:5 and columns 3:5, and multiplying on the right by$H_{u_3}^{\mathrm{T}}$ altered rows 1:5, columns 4:5.
The eigenvalues of both A and A_3 are $\{\ 25.8275\ \ -4.9555\ \ -0.1586\ \ 6.4304\ \ 6.8562\ \}$. ■

In practice, the Householder matrices H_{u_k} are often not determined explicitly, and Example 18.7 builds them to assist with understanding the process. In practice, use Equations 17.11 and 17.12,

$$H_uA = A - \beta uu^{\mathrm{T}}A$$
$$AH_u = A - \beta Auu^{\mathrm{T}}$$

to compute the product $H_{u_k}A_{k-1}H_{u_k}^{\mathrm{T}}$ with submatrix operations. From Example 18.6, we see that $H_{u_k}A_{k-1}$ affects rows $k+1:n$, columns $k:n$, and post multiplying by $H_{u_k}^{\mathrm{T}}$ modifies rows $1:n$, columns $k+1:n$, so we can use the following statements:

$$H(k+1:n,k:n) = H(k+1:n,k:n) - \beta\left(uu^{\mathrm{T}}\right)H(k+1:n,k:n)$$
$$H(1:n,k+1:n) = H(1:n,k+1:n) - \beta H(1:n,k+1:n)\left(uu^{\mathrm{T}}\right).$$

In order to compute eigenvectors from their corresponding eigenvalues using the Hessenberg inverse iteration in Section 18.8.1, we need to compute the orthogonal matrix P such that $A = PHP^{\mathrm{T}}$. Begin with $P = I$ and continue post multiplying P by the current Householder reflection.

Algorithm 18.2 specifies the transformation to upper Hessenberg form. It uses a function $\begin{bmatrix} u & \beta \end{bmatrix} = \text{house}(A,i,j)$ that computes the vector u and the scalar β required to form a Householder reflection that will zero out $A(i+1:n,j)$.

Algorithm 18.2 Transformation to Upper Hessenberg Form

```
function HHESS(A)
  % Given the n × n matrix A,
  % [P H]=hhess(A) returns an upper Hessenberg matrix H and
  % orthogonal matrix P such that A = PHP^T using Householder reflections.
  % H and A have the same eigenvalues.

  H=A
  P=I
  for k=1:n-2 do
    [ u β ] = house(H, k+1, k)
    H(k+1 : n, k : n) = H(k+1 : n, k : n) − β(uu^T)H(k+1 : n, k : n)
    H(1 : n, k+1 : n) = H(1 : n, k+1 : n) − βH(1 : n, k+1 : n)(uu^T)
    P(1 : n, k+1 : n) = P(1 : n, k+1 : n) − βP(1 : n, k+1 : n)(uu^T)
    H(k+2 : n, k) = zeros(n − k − 1, 1)
  end for
  return [H]
end function
```

NLALIB: The function `hhess` implements Algorithm 18.2.

Example 18.9. Let A be the matrix of Example 18.8. The MATLAB commands use `hhess` to transform A to upper Hessenberg form and then verify that $PAP^{\mathrm{T}} = H$ within expected roundoff error.

```
>> [P H] = hhess(A);
>> H

H =
            9      -5.1832    -0.014905      -1.9323     -0.63263
      -11.576       18.761       11.659       0.5625      -3.7232
            0       7.6988      0.01596       2.3683      -3.1455
            0            0      0.57652     -0.61311       2.5435
            0            0            0     -0.46141        6.836

>> norm(P*H*P' - A)

ans =
  2.1495e-014
```

■

18.4.1 Efficiency and Stability

Algorithm 18.2 requires $\frac{10}{3}n^3$ flops for the computation of H. To build the orthogonal matrix P requires an additional $4n^3/3$ flops.

The stability of this algorithm is very satisfactory. The computed upper Hessenberg matrix $\hat{H}$ satisfies $\hat{H} = Q^{\mathrm{T}}(A + E)Q$, were Q is orthogonal and $\|E\|_F \leq cn^2 \text{ eps } \|A\|_F$, where c is a small constant [9, pp. 350-351].

18.5 THE UNSHIFTED HESSENBERG *QR* ITERATION

After transforming matrix A into an upper Hessenberg matrix H having the same eigenvalues as A, we can apply the basic QR iteration and transform H into an upper triangular matrix with the eigenvalues of A on its diagonal. Before presenting the transformation of an upper Hessenberg matrix to upper triangular form, we need the concept of an unreduced upper Hessenberg matrix.

Definition 18.3. An upper Hessenberg matrix whose subdiagonal entries $h_{i+1,i}$, $1 \leq i \leq n-1$ are all nonzero is said to be *unreduced* or *proper*.

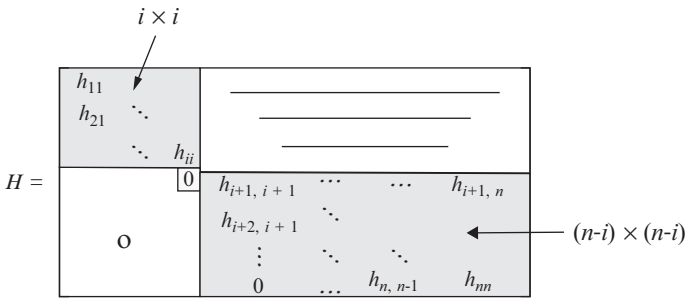

FIGURE 18.7 Reduced Hessenberg matrix.

Figure 18.7 shows a reduced Hessenberg matrix, H.
View the matrix in the form

$$\begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix},$$

where H_{11} is $i \times i$, H_{22} is $(n-i) \times (n-i)$, and both are upper Hessenberg. The eigenvalues of H are those of H_{11} and H_{22} (Problem 18.8). If either of these submatrices has a zero on its subdiagonal, split it into two submatrices, and so forth. As a result, we only need to consider unreduced matrices.

Remark 18.4. For the sake of simplicity, we will only deal with unreduced upper Hessenberg matrices in the book.

During the QR iteration, we want the intermediate matrices $A_i = R_{i-1}Q_{i-1}$ to remain upper Hessenberg, and Theorem 18.7 guarantees this.

Theorem 18.4. *If the $n \times n$ unreduced upper Hessenberg matrix H_k has full column rank and $H_k = Q_k R_k$ is its reduced QR decomposition, then $H_{k+1} = R_k Q_k$ is also an upper Hessenberg matrix.*

Proof. Apply $n-1$ Givens rotations to transform H_k into upper triangular matrix R_k:

$$J_{n-1}(n-1, n, c_{n-1}, s_{n-1}) J_{n-2}(n-2, n-1, c_{n-2}, s_{n-2}) \ldots J_2(2, 3, c_2, s_2) J_1(1, 2, c_1, s_1) H_k = R_k.$$

Thus, $H_k = Q_k R_k$, where

$$\begin{aligned} Q_k &= J_1(1, 2, c_1, s_1)^{\mathrm{T}} J_2(2, 3, c_2, s_2)^{\mathrm{T}} \ldots J_{n-2}(n-2, n-1, c_{n-2}, s_{n-2})^{\mathrm{T}} J_{n-1}(n-1, n, c_{n-1}, s_{n-1})^{\mathrm{T}} \\ &= \begin{bmatrix} c_1 & -s_1 & & & & \\ s_1 & c_1 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & & & \\ & c_2 & -s_2 & & & \\ & s_2 & c_2 & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & c_{n-1} & -s_{n-1} \\ & & & & s_{n-1} & c_{n-1} \end{bmatrix}, \end{aligned}$$

and Q_k is upper Hessenberg. By the results of Problem 17.13, Q_k is unique. Now, since R_k is upper triangular and Q_k is upper Hessenberg, $R_k Q_k$ must be upper Hessenberg. □

Theorem 18.4 shows us that each new matrix H_{k+1} is upper Hessenberg and that the QR decomposition of an upper Hessenberg matrix H_k is accomplished using $n-1$ Givens rotations that eliminate the subdiagonal entries. The cost of the decomposition is $O(n^2)$ (Problem 18.12), much better than the $O(n^3)$ flops required for a general square matrix.

In order to implement the algorithm, there must be a criterion for terminating the iteration. It can be shown that as the iteration moves forward, the entry $h_{n,n-1}$ converges to zero rapidly. As suggested in Ref. [2, p. 391], stop the iterations when $|h_{n,n-1}| < \text{tol}\left(|h_{n-1,n-1}| + |h_{nn}|\right)$, accept h_{nn} as the approximate eigenvalue, and set $h_{n,n-1}$ to zero. The algorithm then works with the $(n-1) \times (n-1)$ submatrix and repeats the process. The final matrix has dimension 2×2, and its diagonal contains the final two eigenvalues. This method is termed *deflation*, and the eigenvalues are on the diagonal of H.

We need to justify deflation by showing that it will yield the same eigenvalues we would obtain by dealing with the whole matrix. Assume we have executed the QR iteration and have reduced the $k \times k$ submatrix, T_k, in the lower right-hand corner to upper triangular form so we now have

$$\overline{H} = \begin{bmatrix} H_{n-k} & X \\ 0 & T_k \end{bmatrix}.$$

H has the same eigenvalues as A, and the QR iteration is an orthogonal similarity transformation, so the eigenvalues of $\overline{H}$ are the same as those of H. $\overline{H}$ is a reduced upper Hessenberg matrix, so its eigenvalues are those of T_k and H_{n-k} (see Problem 18.8). Assuming that the Givens QR decomposition of an upper Hessenberg matrix is implemented in the function `givenshessqr`, Algorithm 18.3 specifies the unshifted Hessenberg QR iteration.

Algorithm 18.3 Unshifted Hessenberg QR Iteration

```
function EIGQR(A,tol,maxiter)
   % Hessenberg QR iteration for computing all the eigenvalues of
   % a real matrix whose eigenvalues have distinct magnitudes.
   % E=eigqr(A,tol,maxiter), where the tol is error tolerance
   % desired, and maxiter is the maximum number of iterations for
   % computing any single eigenvalue. Vector E contains the eigenvalues.
   % If the desired tolerance is not obtained for any particular eigenvalue,
   % a warning message is printed and computation continues
   % for the remaining eigenvalues.

   H=hhess(A)
   for k=n:-1:2 do
      iter=0
      while |h_{k,k-1}| ≥ tol(|h_{k-1,k-1}| + |h_{kk}|) do
         iter=iter+1
         if iter>maxiter then
            print 'Failure of convergence.'
            print 'Current eigenvalue approximation <value of h_{kk}>'
            break out of inner loop.
         end if
         [ Q_k  R_k ] = givenshessqr(H(1 : k, 1 : k))
         H(1 : k, 1 : k) = R_k Q_k
      end while
      H_{k,k-1} = 0
   end for
   E=diag(H)
   return E
end function
```

NLALIB: The function `eigqr`, supported by `givenshessqr`, implements Algorithm 18.3. If $|h_{k-1,k-1}| + |h_{kk}| = 0$, the algorithm fails. In the MATLAB implementation, additional code handles this case by changing the convergence criterion to $|h_{k,k-1}| < \text{tol}\,\|H\|_F$.

Example 18.10. This example demonstrates the behavior of `eigqr` for the matrix

$$A = \begin{bmatrix} 8 & 6 & 10 & 10 \\ 9 & 1 & 10 & 5 \\ 1 & 3 & 1 & 8 \\ 10 & 6 & 10 & 1 \end{bmatrix}.$$

After reduction to upper Hessenberg form,

$$H = \begin{bmatrix} 8 & -12.156 & -9.0829 & 2.3919 \\ -13.491 & 8.0714 & 13.61 & 0.60082 \\ 0.0000 & 7.4288 & -0.64678 & -3.2612 \\ 0.0000 & 0.0000 & 0.66746 & -4.4246 \end{bmatrix}.$$

The first *QR* iteration ends with the matrix

$$\begin{bmatrix} 24.348 & 9.2602 & -0.59021 & -5.7314 \\ 1.0076e-006 & -7.5299 & -4.1391 & -2.705 \\ 0 & 0.0010417 & -4.9789 & -0.77024 \\ 0 & 0 & 0 & -0.83907 \end{bmatrix}.$$

Now shift to the upper 3×3 submatrix

$$\begin{bmatrix} 24.348 & 9.2602 & -0.59021 \\ 1.0076e-006 & -7.5299 & -4.1391 \\ 0 & 0.0010417 & -4.9789 \end{bmatrix}.$$

The second *QR* iteration ends with

$$\begin{bmatrix} 24.348 & 9.2604 & -0.58643 \\ 0 & -7.5282 & -4.1402 \\ 0 & 0 & -4.9806 \end{bmatrix}.$$

The last *QR* iteration begins with the 2×2 submatrix

$$\begin{bmatrix} 24.348 & 9.2604 \\ 0 & -7.5282 \end{bmatrix},$$

and finishes with

$$\begin{bmatrix} 24.348 & 9.2604 \\ 0 & -7.5282 \end{bmatrix}$$

The algorithm finds the eigenvalues in the order $\{-0.83907, -4.9806, -7.5282, 24.348\}$. ■

18.5.1 Efficiency

Reduction to upper Hessenberg form requires $O\left(n^3\right)$ flops. It can be shown that the *QR* iteration applied to an upper Hessenberg matrix requires $O\left(n^2\right)$ flops [5, p. 92]. First reducing A to upper Hessenberg form and applying the *QR* iteration to the Hessenberg matrix costs $O\left(n^3\right) + O\left(n^2\right)$ flops, clearly superior to the basic *QR* iteration.

18.6 THE SHIFTED HESSENBERG *QR* ITERATION

The rate at which the subdiagonal entries of the Hessenberg matrix converge to zero depends on the ratio $\left|\frac{\lambda_1}{\lambda_2}\right|^p$, where $\lambda_1, \lambda_2, |\lambda_1| < |\lambda_2|$ are the two smallest eigenvalues in the current $k \times k$ submatrix H_k. If λ_2 is close in magnitude to λ_1, convergence will be slow. As a result, we employ a strategy that produces more accurate results. We transform the problem of finding the eigenvalue to finding an eigenvalue of another matrix in which that eigenvalue is more isolated. A lemma that will form the basis for our improved approach.

Lemma 18.1.

1. *If σ is a real number, then the eigenvalues of $A - \sigma I$ are $(\lambda_i - \sigma)$, $1 \leq i \leq n$, and the eigenvector, v_i, corresponding to eigenvalue λ_i of A is also an eigenvector corresponding to the eigenvalue $(\lambda_i - \sigma)$ of $A - \sigma I$.*
2. *If $\overline{\lambda_i}$ is an eigenvalue of $A - \sigma I$, then $\overline{\lambda_i} + \sigma$ is an eigenvalue of A.*

Proof. For (1), pick any λ_i with eigenvector v_i. Then

$$(A - \sigma I)\, v_i = Av_i - \sigma v_i = \lambda_i v_i - \sigma v_i = (\lambda_i - \sigma)\, v_i,$$

and $\lambda_i - \sigma$ is an eigenvalue of $A - \sigma I$ with corresponding eigenvector v_i.

For (b), if $\overline{\lambda_i}$ is an eigenvalue of $A - \sigma I$ with corresponding eigenvector w_i, then

$$(A - \sigma I)\, w_i = \overline{\lambda} w_i$$
$$Aw_i = \left(\overline{\lambda} + \sigma\right) w_i,$$

and $\overline{\lambda_i} + \sigma$ is an eigenvalue of A. □

A solution to the problem of nearly equal eigenvalues and to improve convergence in general is to perform what is called a *shift*. There are two types of shifts, single and double. A single shift is used when computing a real eigenvalue, and a double shift is used when computing a pair of complex conjugate eigenvalues or a pair of real eigenvalues.

18.6.1 A Single Shift

Use a *single shift* to create a new matrix $\overline{H_k} = H_k - \sigma_1 I$, where σ_1 is close to λ_1. By Lemma 18.1, $\lambda_1 - \sigma_1$ and $\lambda_2 - \sigma_1$ are eigenvalues of $H_k - \sigma_1 I$. The rate of convergence to the smallest eigenvalue of $H_k - \sigma_1 I$ then depends on $\left|\frac{\lambda_1 - \sigma_1}{\lambda_2 - \sigma_1}\right|^p$. The numerator $\lambda_1 - \sigma_1$ is small relative to $\lambda_2 - \sigma_2$, and the rate of convergence is improved. Also by Lemma 18.1, if $\overline{\lambda}$ is an eigenvalue of $H_k - \sigma_1 I$ then $\overline{\lambda} + \sigma_1$ is an eigenvalue of H_k. Our strategy is to compute the QR decomposition of the matrix $H_k - \sigma_1 I = Q_k R_k$ and let $H_{k+1} = R_k Q_k + \sigma_1 I$. The reasoning for this is as follows:

$$\begin{aligned} H_{k+1} &= R_k Q_k + \sigma_1 I \\ &= Q_k^T \left(Q_k R_k\right) Q_k + \sigma_i I \\ &= Q_k^T \left(H_k - \sigma_1 I\right) Q_k + \sigma_1 I \\ &= Q_k^T H_k Q_k, \end{aligned}$$

and H_{k+1} is similar to H_k and to A. The problem is to choose an optimal σ_1. It practice, the shift is often chosen as $\sigma_1 = H_k(k, k)$, since H_{kk} converges to λ_1. This is known as the *Rayleigh quotient shift.*

Example 18.11. Let $A = \begin{bmatrix} 8 & 1 \\ -1 & 5 \end{bmatrix}$, which is already in upper Hessenberg form. The example shows the detailed results of three iterations of the single-shifted Hessenberg QR method.

$$\sigma_1 = 5.0000, \overline{H_1} = H_1 - \begin{bmatrix} 8 & 1 \\ -1 & 5 \end{bmatrix} - \sigma_1 I = \begin{bmatrix} 3 & 1 \\ -1 & 0 \end{bmatrix}.$$

$$\left[\, Q_1,\ R_1 \,\right] = \left[\begin{bmatrix} 0.9487 & 0.3162 \\ -0.3162 & 0.9487 \end{bmatrix} \begin{bmatrix} 3.1623 & 0.9487 \\ 0 & 0.3162 \end{bmatrix} \right] = qr\left(\overline{H_1}\right),$$

$$H_1 = R_1 Q_1 + \sigma_1 I = \begin{bmatrix} 7.7000 & 1.9000 \\ -0.1000 & 5.3000 \end{bmatrix}$$

$$\sigma_1 = 5.3000, \overline{H_1} = H_1 - \sigma_1 I = \begin{bmatrix} 2.4000 & 1.9000 \\ -0.1000 & 0 \end{bmatrix}$$

$$\left[\, Q_1,\ R_1 \,\right] = \left[\begin{bmatrix} 0.9991 & 0.0416 \\ -0.0416 & 0.9991 \end{bmatrix} \begin{bmatrix} 2.4021 & 1.8984 \\ 0 & 0.0791 \end{bmatrix} \right] = qr\left(\overline{H_1}\right)$$

$$H_1 = R_1 Q_1 + \sigma_1 I = \begin{bmatrix} 7.6210 & 1.9967 \\ -0.0033 & 5.3790 \end{bmatrix}$$

One more iteration using $\sigma_1 = 5.3790$ gives $H_1 = \begin{bmatrix} 7.6180 & 2.0000 \\ 0.0000 & 5.3820 \end{bmatrix}$. The eigenvalues accurate to four decimal places are $\left\{\ 7.6180,\ \ 5.3820\ \right\}$. ■

Example 18.12. Let

$$A = \begin{bmatrix} -7 & 2 & -1 & 7 & -8 \\ 6 & -5 & -9 & 1 & 10 \\ -4 & 3 & -6 & 10 & -10 \\ 1 & 4 & 9 & -9 & 6 \\ -7 & 5 & -7 & -1 & 7 \end{bmatrix}.$$

Use the algorithm hhess and transform A to the upper Hessenberg matrix

$$H = \begin{bmatrix} -7 & -7.8222 & -0.085538 & -5.4477 & 5.2085 \\ -10.1 & -9.6569 & 1.0505 & 0.77123 & 3.0882 \\ 0 & 11.451 & -1.3692 & 13.96 & -5.7396 \\ 0 & 0 & 16.694 & -2.5483 & -4.2948 \\ 0 & 0 & 0 & 4.5531 & 0.57437 \end{bmatrix}.$$

Using the shift $\sigma = 0.57437$, initially apply the *QR* iteration to the matrix

$$H - \sigma I = \begin{bmatrix} -7.5744 & -7.8222 & -0.085538 & -5.4477 & 5.2085 \\ -10.1 & -10.231 & 1.0505 & 0.77123 & 3.0882 \\ 0 & 11.451 & -1.9435 & 13.96 & -5.7396 \\ 0 & 0 & 16.694 & -3.1227 & -4.2948 \\ 0 & 0 & 0 & 4.5531 & 0 \end{bmatrix}.$$

After a total of five iterations that involve the additional shifts $\sigma = \{\ 1.6065 \ \ 2.3777 \ \ 2.3664 \ \ 2.3663\ \}$, we obtain the matrix

$$\begin{bmatrix} -21.978 & -3.1496 & -2.8609 & 0.45197 & 5.5643 \\ 1.2298 & -7.0975 & 6.9562 & 8.87 & -0.39277 \\ 0 & 7.9804 & 10.4 & -7.8376 & -9.725 \\ 0 & 0 & 0.2591 & -3.6907 & 0.67703 \\ 0 & 0 & 0 & 0 & 2.3663 \end{bmatrix}.$$

Now choose the shift $\sigma = -3.6907$, and continue with the submatrix

$$R_2 = \begin{bmatrix} -21.978 & -3.1496 & -2.8609 & 0.45197 \\ 1.2298 & -7.0975 & 6.9562 & 8.87 \\ 1.2842e-016 & 7.9804 & 10.4 & -7.8376 \\ -4.6588e-016 & -1.5213e-016 & 0.2591 & -3.6907 \end{bmatrix}.$$

After a total of 14 iterations, we obtain the eigenvalue approximations $\lambda_5 = -21.746$, $\lambda_4 = 13.035$, $\lambda_3 = -9.856$, $\lambda_2 = -3.7993$, and $\lambda_1 = 2.3663$. ■

The single-shift Hessenberg *QR* iteration is implemented by the function `eigqrshift` in the software distribution. Each iteration of a $k \times k$ submatrix during deflation terminates when

$$\left|h_{i,i-1}\right| \leq \text{tol}\left(\left|h_{ii}\right| + \left|h_{i-1,i-1}\right|\right),$$

where tol is larger than the unit roundoff. The only difference between eigqr and eigqrshift is that the statements

```
[Q1, R1] = givenshessqr(H(1:k,1:k));
 H(1:k,1:k) = R1*Q1;
```

are replaced by

```
sigma = H(k,k);
[Q1, R1] = givenshessqr(H(1:k,1:k) - sigma*I);
H(1:k,1:k) = R1*Q1 + sigma*I;
```

Remark 18.5. The function eigqrshift only applies to a real matrix with eigenvalues having distinct magnitudes. Also, it cannot be used if the eigenvalues consist of complex conjugate pairs. We remedy these problems by developing the implicit double-shift Francis algorithm in Section 18.8.

18.7 SCHUR'S TRIANGULARIZATION

We develop another matrix decomposition called *Schur's triangularization* that involves orthogonal matrices. The decomposition is very useful theoretically because any square matrix can be factored, including singular ones. It is also a good lead-in to the Francis method in Section 18.8. We will restrict ourselves to real matrices with real eigenvalues. The proof involves the use of mathematical induction, and the reader unfamiliar with this proof technique should consult Appendix B.

Theorem 18.5 (Schur's triangularization). *Every $n \times n$ real matrix A with real eigenvalues can be factored into $A = PTP^{\mathrm{T}}$, where P is an orthogonal matrix, and T is an upper triangular matrix.*

Summary:

If the result is true, then $AP = PT$. Proceeding like we did with the Cholesky decomposition, see what relationships must hold if $AP = PT$ for $n \times n$ orthogonal matrix P. When a pattern evolves, we use mathematical induction to verify the theorem. If the reader chooses to skip the details of the proof, be sure to study Section 19.1, where Schur's triangularization is used to very easily prove the spectral theorem.

Proof. The proof uses construction and induction. If $A = PTP^{\mathrm{T}}$, then $AP = PT$. Let us investigate what we can conclude from this. Equation 18.16 depicts the equation.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ 0 & t_{22} & \cdots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & t_{nn} \end{bmatrix} \tag{18.16}$$

From Equation 18.16, the first column of AP is

$$A \begin{bmatrix} p_{11} \\ p_{21} \\ \vdots \\ p_{n1} \end{bmatrix}, \tag{18.17}$$

and the first column of PT is

$$\begin{bmatrix} p_{11}t_{11} \\ p_{21}t_{11} \\ \vdots \\ p_{n1}t_{11} \end{bmatrix} = t_{11} \begin{bmatrix} p_{11} \\ p_{21} \\ \vdots \\ p_{n1} \end{bmatrix}. \tag{18.18}$$

If we let $\alpha = \begin{bmatrix} p_{11} \\ p_{21} \\ \vdots \\ p_{n1} \end{bmatrix}$, then Equations 18.17 and 18.18 give

$$A\alpha = t_{11}\alpha, \tag{18.19}$$

and t_{11} is an eigenvalue of A with associated normalized eigenvector α (P is orthogonal). The remaining $(n-1)$ columns of P cannot be eigenvectors of A. Pick any other $(n-1)$ linearly independent vectors $w_2, \ldots, w_n$ so that $\{\alpha, w_2, \ldots, w_n\}$ are a basis for $\mathbb{R}^n$. Apply the Gram-Schmidt process to $\{\alpha, w_2, \ldots, w_n\}$ to obtain an orthonormal basis $\{v_1, v_2, \ldots, v_n\}$. The Gram-Schmidt process will not change the first vector if it is already a unit vector, so $v_1 = \alpha$. Let P be the $n \times n$ matrix $P = [v_1 v_2 \ldots v_n]$, where the v_i are the columns of P. P^{T} can be written as $\begin{bmatrix} v_1^{\mathrm{T}} \\ v_2^{\mathrm{T}} \\ \vdots \\ v_n^{\mathrm{T}} \end{bmatrix}$, where the v_i^{T} are the rows of P^{T}. Now form

$$P^{\mathrm{T}}AP = \begin{bmatrix} v_1^{\mathrm{T}} \\ v_2^{\mathrm{T}} \\ \vdots \\ v_n^{\mathrm{T}} \end{bmatrix} A \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} \tag{18.20}$$

$$= \begin{bmatrix} v_1^{\mathrm{T}} \\ v_2^{\mathrm{T}} \\ \vdots \\ v_n^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} Av_1 & Av_2 & \cdots & Av_n \end{bmatrix} \tag{18.21}$$

$$= \begin{bmatrix} v_1^{\mathrm{T}}Av_1 & v_1^{\mathrm{T}}Av_2 & \cdots & v_1^{\mathrm{T}}Av_n \\ v_2^{\mathrm{T}}Av_1 & v_2^{\mathrm{T}}Av_2 & \cdots & v_2^{\mathrm{T}}Av_n \\ \vdots & \vdots & \ddots & \vdots \\ v_n^{\mathrm{T}}Av_1 & v_n^{\mathrm{T}}Av_2 & \cdots & v_n^{\mathrm{T}}Av_n \end{bmatrix} \tag{18.22}$$

$$= \begin{bmatrix} t_{11}v_1^{\mathrm{T}}v_1 & v_1^{\mathrm{T}}Av_2 & \cdots & v_1^{\mathrm{T}}Av_n \\ t_{11}v_2^{\mathrm{T}}v_1 & v_2^{\mathrm{T}}Av_2 & \cdots & v_2^{\mathrm{T}}Av_n \\ \vdots & \vdots & \ddots & \vdots \\ t_{11}v_n^{\mathrm{T}}v_1 & v_n^{\mathrm{T}}Av_2 & \cdots & v_n^{\mathrm{T}}Av_n \end{bmatrix} \tag{18.23}$$

$$= \begin{bmatrix} v_1^{\mathrm{T}}t_{11}v_1 & \cdots & \cdots \\ 0 & \cdots & \cdots \\ \vdots & \cdots & \cdots \\ 0 & \cdots & \cdots \end{bmatrix}. \tag{18.24}$$

The first column of Equation 18.24 depends on the inner products $\langle v_1, v_1\rangle_1, \langle v_1, v_2\rangle, \ldots, \langle v_1, v_n\rangle$. Since $v_1, v_2, \ldots, v_n$ are orthonormal, the only nonzero inner product is $\langle v_1, v_1\rangle = 1$. If we exclude row 1 and column 1, the remaining submatrix has size $(n-1)\times(n-1)$, and we will designate it by A_2. $P^{\mathrm{T}}AP$ has the structure shown in Figure 18.8.

Now apply induction on n. If $n = 1$, the theorem is obviously true, since $A = (1)\,A\,(1)$. Now assume that Schur's triangularization applies to any $(n-1)\times(n-1)$ matrix. It then applies to A_2, so $A_2 = P_2T_2P_2^{\mathrm{T}}$, where P_2 is orthogonal and T_2 is upper triangular. We will show that this assumption implies that the theorem is true for an $n\times n$ matrix. In Figure 18.9, A_2 has been replaced by its decomposition.

Now, we must take this representation of $P^{\mathrm{T}}AP$ and produce the triangularization of A. Using block matrix notation, we can write the matrix in Figure 18.9 as follows:

$$\begin{bmatrix} t_{11} & \cdots & \cdots \\ 0 & & \\ 0 & P_2T_2P_2^{\mathrm{T}} & \\ \vdots & & \\ 0 & & \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & P_2 \end{bmatrix} \begin{bmatrix} t_{11} & \cdots \\ 0 & T_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & P_2 \end{bmatrix}^{\mathrm{T}}.$$

$$P^{\mathrm{T}}AP = \begin{bmatrix} t_{11} & \cdots & \cdots & \cdots \\ 0 & & & \\ \vdots & & A_2 & \\ 0 & & & \end{bmatrix}$$

FIGURE 18.8 Inductive step in Schur's triangularization.

$$P^{\mathrm{T}}AP = \begin{bmatrix} t_{11} & \cdots & \cdots & \cdots \\ 0 & & & \\ \vdots & & P_2T_2P_2^{\mathrm{T}} & \\ 0 & & & \end{bmatrix}$$

FIGURE 18.9 Schur's triangularization.

Let $\overline{Q} = \begin{bmatrix} 1 & 0 \\ 0 & P_2 \end{bmatrix}$ and $T = \begin{bmatrix} t_{11} & \cdots \\ 0 & T_2 \end{bmatrix}$. This gives $P^{\mathrm{T}}AP = \overline{Q}T\overline{Q}^{\mathrm{T}}$, so $A = P\overline{Q}T\overline{Q}^{\mathrm{T}}P^{\mathrm{T}} = \left(P\overline{Q}\right)T\left(P\overline{Q}\right)^{\mathrm{T}}$. If we show that $P\overline{Q}$ is an orthogonal matrix, the proof is complete. Now,

$$\overline{Q}^{\mathrm{T}}\overline{Q} = \begin{bmatrix} 1 & 0 \\ 0 & P_2 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 1 & 0 \\ 0 & P_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & P_2^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & P_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & P_2^{\mathrm{T}}P_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & I \end{bmatrix} = I,$$

so $\overline{Q}$ is an orthogonal matrix. Then, $\left(P\overline{Q}\right)^{\mathrm{T}}\left(P\overline{Q}\right) = \overline{Q}^{\mathrm{T}}P^{\mathrm{T}}P\overline{Q} = \overline{Q}^{\mathrm{T}}I\overline{Q} = \overline{Q}^{\mathrm{T}}\overline{Q} = I$, so $P\overline{Q}$ is an orthogonal matrix. If we let $Q = P\overline{Q}$, we have

$$A = QTQ^{\mathrm{T}},$$

and the proof is complete. □

Example 18.13. Let $A = \begin{bmatrix} 1 & 3 \\ -1 & 5 \end{bmatrix}$. Build the Schur form by starting with Equation 18.19. Find an eigenvalue t_{11} and normalized eigenvector v_1 such that

$$Av_1 = t_{11}v_1$$

and obtain $t_{11} = 2$, $v_1 = \begin{bmatrix} -0.94868 \\ -0.31623 \end{bmatrix}$. Now extend v_1 to an orthonormal basis v_1, v_2 for $\mathbb{R}^2$. Let $v_2 = \begin{bmatrix} 1 & 1 \end{bmatrix}^{\mathrm{T}}$, which is linearly independent of v_1, and apply Gram-Schmidt to obtain the orthonormal basis

$$\begin{bmatrix} -0.94868 \\ -0.31623 \end{bmatrix}, \quad \begin{bmatrix} -0.31623 \\ 0.94868 \end{bmatrix}$$

and matrix

$$P = \begin{bmatrix} -0.94868 & -0.31623 \\ -0.31623 & 0.94868 \end{bmatrix}.$$

Form

$$P^{\mathrm{T}}AP = \begin{bmatrix} 2 & -4 \\ 0 & 4 \end{bmatrix},$$

as indicated in Figure 18.8. In Figure 18.8, $P_2 = 1$, $T_2 = 4$, and $P_2^{\mathrm{T}} = 1$. Follow the remainder of the proof, and you will see that $\overline{Q} = I$, $T = \begin{bmatrix} 2 & -4 \\ 0 & 4 \end{bmatrix}$, and the final value of the orthogonal matrix is $P = \begin{bmatrix} -0.94868 & -0.31623 \\ -0.31623 & 0.94868 \end{bmatrix}$. A Schur's triangularization for A is

$$A = \begin{bmatrix} -0.94868 & -0.31623 \\ -0.31623 & 0.94868 \end{bmatrix} \begin{bmatrix} 2 & -4 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} -0.94868 & -0.31623 \\ -0.31623 & 0.94868 \end{bmatrix}.$$ ■

Remark 18.6.

- Since t_{11} is any eigenvalue of A, and many choices are possible for performing the extension to an orthonormal basis for $\mathbb{R}^n$, there is no unique Schur's triangularization (Problem 18.21).
- The MATLAB function `[U, T] = schur(A)` computes the Schur's triangularization of the square matrix A.
- Our proof of Schur's triangularization theorem involved knowing the eigenvalues of the matrix, so it will not help us to compute eigenvalues. However, it gives us a reason to suspect that it is possible to reduce any matrix to upper triangular form using orthogonality similarity transformations. We will see in Section 18.9 that the Francis iteration of degree one or two does exactly that.

The theorem is actually true for any matrix $A \in \mathbb{C}^{n\times n}$. If A is complex, then P and T are complex. If A is real and has complex eigenvalues, it is possible to find a real orthogonal matrix P if T is replaced by a real *quasi-triangular matrix* and obtain what is termed the *real Schur form*. T is a block upper triangular matrix of the form

$$T = \begin{bmatrix} T_{11} & T_{12} & T_{13} & \dots & T_{1k} \\ 0 & T_{22} & T_{23} & \dots & T_{2k} \\ 0 & 0 & T_{33} & \dots & T_{3k} \\ \vdots & \vdots & \vdots & & \\ 0 & 0 & 0 & \dots & T_{kk} \end{bmatrix},$$

where the diagonal blocks are of size 1×1 or 2×2. The 1×1 blocks contain the real eigenvalues of A, and the eigenvalues of the 2×2 diagonal blocks contain complex conjugate eigenvalues or two real eigenvalues. For a proof, see Ref. [2, pp. 376-377].

Example 18.14. Compute the real Schur form of a 7×7 matrix and note the 2×2 and 1×1 diagonal blocks that contain the eigenvalues.

```
[U,T] = schur(A);
>> T

T =
    141.6    -86.367    -85.826    -91.634    -14.548    -57.656    -31.229
        0    -167.75     12.805     12.646     -32.73    -14.815     74.447
        0          0     54.834    -93.417     71.143     27.484     21.421
        0          0     80.735     54.834     2.7022     144.71    -2.5564
        0          0          0          0    -9.4734    -117.66     28.839
        0          0          0          0          0    -67.023     130.67
        0          0          0          0          0    -42.112    -67.023

>> eig(A)

ans =
        141.6 +          0i
       54.834 +     86.845i
       54.834 -     86.845i
      -167.75 +          0i
      -9.4734 +          0i
      -67.023 +     74.181i
      -67.023 -     74.181i
```

The diagonal blocks are

$$\begin{bmatrix} -67.023 & 130.67 \\ -42.112 & -67.023 \end{bmatrix}, \quad \begin{bmatrix} -9.4734 \end{bmatrix}, \quad \begin{bmatrix} 54.834 & -93.417 \\ 80.735 & 54.834 \end{bmatrix}, \quad \begin{bmatrix} 141.6 & -86.367 \\ 0 & -167.75 \end{bmatrix}.$$

Block $\begin{bmatrix} -67.023 & 130.67 \\ -42.112 & -67.023 \end{bmatrix}$ gives eigenvalues -67.023 ± 74.181i, block $\begin{bmatrix} 54.834 & -93.417 \\ 80.735 & 54.834 \end{bmatrix}$ gives eigenvalues 54.834 ± 86.845i, and block $\begin{bmatrix} 141.6 & -86.367 \\ 0 & -167.75 \end{bmatrix}$ returns the real eigenvalues 141.6 and -167.75. The 1×1 block $\begin{bmatrix} -9.4734 \end{bmatrix}$ produces eigenvalue -9.4734. ■

18.8 THE FRANCIS ALGORITHM

The *Francis algorithm*, also known as the *implicit QR algorithm*, has been a staple in computing the eigenvalues and eigenvectors of a small- to medium-size general matrix for many years [54, 55]. It begins by reducing A to upper Hessenberg form so that $Q^{\mathrm{T}}AQ = H$, where Q is an orthogonal matrix (Algorithm 18.2). Then the algorithm transforms H to upper triangular form by using a succession of orthogonal similarity transformations rather than directly using shifts and the QR decomposition, and this is the origin of the term implicit. There are two versions of the Francis algorithm, the *single shift* and *double shift*. The single-shift version can be used to determine the eigenvalues of a real matrix whose eigenvalues are real, and this includes symmetric matrices. Chapter 19 discusses eigenvalue computation of a symmetric matrix and uses the single-shift algorithm. The double-shift version computes all eigenvalues, real or complex conjugate pairs, of a nonsymmetric matrix without using complex arithmetic. The development of the double-shift version is complicated, and can be omitted if desired, but the reader should study Section 18.8.1.

18.8.1 Francis Iteration of Degree One

The first step is to apply orthogonal similarity transformations that reduce the matrix A to upper Hessenberg form, $H = Q^{\mathrm{T}}AQ$. With the explicit shifted QR algorithm, we perform a series of QR factorizations on $H - \sigma I$, each of which requires

$O\left(n^2\right)$ flops. Instead, the Francis algorithm executes a sequence of orthogonal similarity transformations to form the QR factorizations.

Recall that an explicit single shift requires executing the following two statements in succession:

$$H - \sigma I = Q_1 R_1, \quad H_1 = R_1 Q_1 + \sigma I,$$

so

$$\begin{aligned} Q_1^{\mathrm{T}} H - \sigma_1 Q_1^{\mathrm{T}} &= R_1 \\ H_1 Q_1^{\mathrm{T}} - \sigma_1 Q_1^{\mathrm{T}} &= R_1, \end{aligned}$$

and

$$H_1 = Q_1^{\mathrm{T}} H Q_1. \tag{18.25}$$

See Section 18.6.1 for an alternative derivation of Equation 18.25. The trick is to form H_1 without having to directly compute Q_1 from the QR decomposition. The approach is based upon the *implicit Q theorem*. For a proof see Ref. [2, p. 381].

Theorem 18.6 (The implicit Q theorem). *Let* $Q = \begin{bmatrix} q_1 & q_3 & \dots & q_{n-1} & q_n \end{bmatrix}$ *and* $V = \begin{bmatrix} v_1 & v_3 & \dots & v_{n-1} & v_n \end{bmatrix}$ *be orthogonal matrices with the property that both* $Q^{\mathrm{T}} A Q = H$ *and* $V^{\mathrm{T}} A V = K$ *are unreduced upper Hessenberg, where* $A \in \mathbb{R}^{n \times n}$. *If* $q_1 = v_1$, *then* $q_i = \pm v_i$ *and* $\left|h_{i,i-1}\right| = \left|k_{i,i-1}\right|$ *for* $2 \leq i \leq n$. *In other words, H and K are essentially the same matrix.*

The theorem applies because we are going to execute a series of steps

$$H_{i+1} = Q_i^{\mathrm{T}} H_i Q_i.$$

in lieu of performing a QR decomposition, where Q_i is a orthogonal matrix with an appropriately chosen first column.

Preparation for Understanding the Iteration

Suppose that

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ \underline{\mathbf{h_{31}}} & h_{32} & h_{33} & h_{34} \\ 0 & 0 & h_{43} & h_{44} \end{bmatrix}$$

is upper Hessenberg, except for a nonzero entry at position (3, 1). It is important to understand that the element to be zeroed out by $J(i, j, c, s)$ does not have to be a_{ji}. Let c and s be determined by

$$\begin{bmatrix} c, & s \end{bmatrix} = \text{givensparms}\left(h_{21}, h_{31}\right),$$

and consider the following:

$$\begin{aligned} J(2,3,c,s)\, H &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & s & 0 \\ 0 & -s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ \underline{\mathbf{h_{31}}} & h_{32} & h_{33} & h_{34} \\ 0 & 0 & h_{43} & h_{44} \end{bmatrix} \\ &= \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ ch_{21} + sX & ch_{22} + sh_{32} & ch_{23} + sh_{33} & ch_{24} + sh_{34} \\ \underline{-\mathbf{sh_{21}} + \mathbf{ch_{31}}} & -sh_{22} + ch_{32} & -sh_{23} + ch_{33} & -sh_{24} + ch_{34} \\ 0 & 0 & h_{43} & h_{44} \end{bmatrix} \\ &= \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ ch_{21} + sX & ch_{22} + sh_{32} & ch_{23} + sh_{33} & ch_{24} + sh_{34} \\ \underline{\mathbf{0}} & -sh_{22} + ch_{32} & -sh_{23} + ch_{33} & -sh_{24} + ch_{34} \\ 0 & 0 & h_{43} & h_{44} \end{bmatrix} \end{aligned}$$

The action zeroed out position $(3, 1)$. If we multiply by $J\,(2, 3, c, s)^{\mathrm{T}}$ on the right, here is the form of the product:

$$J\,(2,3,c,s)\,HJ\,(2,3,c,s)^{\mathrm{T}} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & * & * & * \\ 0 & \mathbf{X} & * & * \end{bmatrix}.$$

By computing $J\,(2, 3, c, s)$, we remove a nonzero item at position $(3, 1)$, but introduce another at position $(4, 2)$, and the matrix is not upper Hessenberg. We have "chased" a nonzero element from $(3, 1)$ to $(4, 2)$ with an orthogonal similarity transformation.

Demonstration of the Francis Iteration of Degree One

We use a 5×5 matrix to illustrate a step of the Francis algorithm:

$$H = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

Step 1: Assume a real shift, σ, and form a Givens rotation

$$J_1 = J\,(1,2,c_1,s_1) = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix},$$

where c and s are determined from the vector

$$\begin{bmatrix} h_{11} - \sigma \\ h_{21} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Note that the computation of c and s using a vector with first element $h_{11} - \sigma$ is critical. It will be a important link to Theorem 18.5 that justifies the implicit approach to the single-shift algorithm. The product $J_1 H J_1^{\mathrm{T}}$ and the resulting matrix have the form

$$H_1 = J_1 H J_1^{\mathrm{T}} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ + & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

The product disturbs the upper Hessenberg form, leaving a nonzero element at $(3, 1)$. The element at $(3, 1)$ is called a bulge. *Our job is to chase the bulge down to the right and off the matrix, leaving the resulting matrix in upper Hessenberg form.*

Step 2: Let c_2 and s_2 be formed from the elements $H_1\,(2, 1)$ and $H_1\,(3, 1)$. The rotation $J_2 = \begin{bmatrix} 1 & & & & \\ & c_2 & s_2 & & \\ & -s_2 & c_2 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}$ applied to H_1 eliminates the bulge at $(3, 1)$, but after postmultiplication by J_2^{T}, another bulge appears at $(4, 2)$:

$$H_2 = J_2 J_1 H J_1^{\mathrm{T}} J_2^{\mathrm{T}} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & + & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

Step 3: Let c_3 and s_3 be formed from the elements $H_2(3,2)$ and $H_2(4,2)$. The rotation $J_3 = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & c_3 & s_3 & \\ & & -s_3 & c_3 & \\ & & & & 1 \end{bmatrix}$ *applied to* H_2 *eliminates the bulge at* $(4,2)$*, but after postmultiplication by* J_3^{T} *another bulge appears at* $(5,3)$.

$$H_3 = J_3 J_2 J_1 H_2 J_1^{\mathrm{T}} J_2^{\mathrm{T}} J_3^{\mathrm{T}} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & + & * & * \end{bmatrix},$$

Step 4: Let c_4 and s_4 be formed from the elements $H_3(4,3)$ and $H_3(5,3)$. The rotation $J_4 = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & c_4 & s_4 \\ & & & -s_4 & c_4 \end{bmatrix}$ *applied to* H_3 *eliminates the bulge at* $(5,3)$*, and after postmultiplication by* J_4^{T} *no more bulges remain, and the matrix has upper Hessenberg form*

$$H_4 = J_4 J_3 J_2 J_1 (H - \sigma I) J_1^{\mathrm{T}} J_2^{\mathrm{T}} J_3^{\mathrm{T}} J_4^{\mathrm{T}} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

In general, each iteration chases the bulge through indices $(3,1), (4,2), (5,3), (6,4), \ldots, (n, n-2)$.
If we let K be the orthogonal matrix

$$K = \left(J_1^{\mathrm{T}} J_2^{\mathrm{T}} J_3^{\mathrm{T}} J_4^{\mathrm{T}}\right)^{\mathrm{T}},$$

then

$$K^{\mathrm{T}} H K = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

is an unreduced upper Hessenberg matrix, and a check will show that K has the form

$$K = \begin{bmatrix} c_1 & * & * & * & * \\ s_1 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

If Givens rotations are used to compute the QR decomposition of $H - \sigma I$, the first column of Q will be the same as that of K. By the implicit Q theorem, the matrices K and the matrix Q are essentially the same matrix, and K can be used to perform the single shift given by Equation 18.16. Except for signs, the matrix after the bulge chase is the matrix we would obtain by doing an explicit shift. Algorithm 18.4 implements the bulge chase. It uses a function givensmult(H,i,j,c,s) that computes $HJ(i,j,c,s)^{\mathrm{T}}$. An implementation of the function is in the book software distribution.

Algorithm 18.4 Single Shift Using the Francis Iteration of Degree One

```
function CHASE(H)
   % Bulge chase in the Francis algorithm.
   % [Q, H1]=chase(H) returns an orthogonal matrix Q such that Q^T HQ = H_1,
   % where H and H1 are upper Hessenberg matrices, and Q is an orthogonal matrix.
   Q=I
   % use h_nn as the shift.
   σ = h_nn
   % case (1,2), (2,1) using shift σ.
   [c, s] = givensparms(h_11 − σ, h_21)
   H=givensmul(H,1,2,c,s)
   H=givensmult(H,1,2,c,s)
   Q=givensmul(Q,1,2,c,s)

   % chase the bulge.
   for i=1:n-2 do
      [c, s] = givensparms(h_{i+1,i}, h_{i+2,i})
      H=givensmul(H,i+1,i+2,c,s)
      H=givensmult(H,i+1,i+2,c,s)
      h_{i+2,i} = 0
      Q=givensmul(Q,i+1,i+2,c,s)
   end for
   H1=H
   return [Q,H1]
end function
```

NLALIB: The function `chase` implements Algorithm 18.4.

The single-shift strategy is only practical when we know that all eigenvalues are real, and this is true for a symmetric matrix. Section 19.5 uses the implicit single shift to build a general symmetric matrix eigenvalue/eigenvector solver.

18.8.2 Francis Iteration of Degree Two

When computing a complex conjugate pair of eigenvalues, it is desirable to avoid complex arithmetic, since complex arithmetic requires about four times as many flops. By performing a double shift, we can compute both eigenvalues using only real arithmetic, and the double shift can also be used to approximate two real eigenvalues. The idea is to use the eigenvalues of the 2×2 lower right-hand corner submatrix as the shifts (Figure 18.10).

Let H be the upper Hessenberg matrix whose 2×2 submatrix has eigenvalues σ_1 and σ_2. Shift first by σ_1 to obtain

$$H - \sigma_1 I = Q_1 R_1, \quad H_1 = R_1 Q_1 + \sigma_1 I, \tag{18.26}$$

and then shift H_1 by σ_2 and we have

$$H_1 - \sigma_2 I = Q_2 R_2, \quad H_2 = R_2 Q_2 + \sigma_2 I. \tag{18.27}$$

The initial matrix H is real, but if σ_1 and $\sigma_2 = \overline{\sigma_1}$ are complex conjugates, the double shift as given requires complex arithmetic. By some clever operations, we can perform the shifts entirely with real arithmetic. When dealing with a complex matrix, we use the *Hermitian transpose*, or the *conjugate transpose*, that is, the complex conjugate of the elements in the transpose, indicated by A^*. For instance

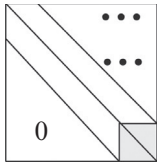

FIGURE 18.10 Eigenvalues of a 2×2 matrix as shifts.

$$\begin{bmatrix} 1-i & 2+3i \\ 5-2i & 6+9i \end{bmatrix}^* = \begin{bmatrix} 1+\mathrm{i} & 5+2i \\ 2-3i & 6-9i \end{bmatrix}.$$

From Equation 18.26, we have

$$\begin{aligned} Q_1^* H - \sigma_1 Q_1^* &= R_1 \\ H_1 Q_1^* - \sigma_1 Q_1^* &= R_1, \end{aligned}$$

so

$$Q_1^* H Q_1 = H_1. \tag{18.28}$$

Similarly, Equation 18.27 gives

$$Q_2^* H_1 Q_2 = H_2, \tag{18.29}$$

and by substituting Equation 18.28 into Equation 18.29, we have

$$H_2 = (Q_1 Q_2)^* H (Q_1 Q_2) \tag{18.30}$$

Thus, H and H_2 are orthogonally similar and have the same eigenvalues. These relationships allow us to prove the following lemma that leads to an algorithm for performing the double shift using real arithmetic.

Lemma 18.2. *There exist orthogonal matrices Q_1 and Q_2 such that*

a. *$Q_1 Q_2$ is real,*
b. *H_2 is real*

Proof. From Equations 18.26 and 18.27,

$$Q_2 R_2 = H_1 - \sigma_2 I = R_1 Q_1 + (\sigma_1 - \sigma_2) I,$$

and

$$\begin{aligned} Q_1 Q_2 R_2 R_1 &= Q_1 (R_1 Q_1 + (\sigma_1 - \sigma_2) I) R_1 && (18.31) \\ &= Q_1 R_1 Q_1 R_1 + (\sigma_1 - \sigma_2) Q_1 R_1 && (18.32) \\ &= (H - \sigma_1 I)(H - \sigma_1 I) + (\sigma_1 - \sigma_2)(H - \sigma_1 I) && (18.33) \\ &= H^2 - 2\sigma_1 H + \sigma_1^2 I + \sigma_1 H - \sigma_1^2 I - \sigma_2 H + (\sigma_1 \sigma_2) I && (18.34) \\ &= H^2 - (\sigma_1 + \sigma_2) H + \sigma_1 \sigma_2 I && (18.35) \\ &= (H - \sigma_1 I)(H - \sigma_2 I) = S && (18.36) \end{aligned}$$

If $\sigma_1 = a + \mathrm{i}b$ and $\sigma_2 = a - \mathrm{i}b$ are complex conjugates, Equation 18.36 takes the form

$$H^2 - 2\mathrm{Real}(\sigma_1) H + |\sigma_1|^2 I.$$

The matrix S is real, and $S = (Q_1 Q_2)(R_2 R_1)$ is a QR decomposition of S, so both $Q_1 Q_2$ and $R_1 R_2$ can be chosen to be real. By Equation 18.30, H_2 is real. □

Now let's see how all this fits together to perform a double shift. The lower right-hand 2×2 matrix is

$$M = \begin{bmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{nn} \end{bmatrix},$$

whose eigenvalues are

$$\lambda = \frac{(h_{n-1,n-1} + h_{nn}) \pm \sqrt{(h_{n-1,n-1} + h_{nn})^2 - 4(h_{n-1,n-1} h_{nn} - h_{n,n-1} h_{n-1,n})}}{2}.$$

The matrix S of Equation 18.36 has the value (Problem 18.18)

$$\begin{aligned} S &= H^2 - (h_{n-1,n-1} + h_{nn}) H + (h_{n-1,n-1} h_{nn} - h_{n,n-1} h_{n-1,n}) I && (18.37) \\ &= H^2 - \mathrm{trace}(M) + \det(M). && (18.38) \end{aligned}$$

Find the QR decomposition of $S = QR = (Q_1Q_2)(R_2R_1)$ and compute

$$H_2 = Q^{\mathrm{T}}HQ.$$

In summary,

Computation of a Double Shift

1. Form the real matrix $S = H^2 - \left(h_{n-1,n-1} + h_{nn}\right)H + \left(h_{n-1,n-1}h_{nn} - h_{n,n-1}h_{n-1,n}\right)I$.
2. Find the QR decomposition $S = QR$.
3. Compute $H_2 = Q^{\mathrm{T}}HQ$.

It would appear we can simply use the double shift as we have developed it to find complex conjugate eigenvalues or two real eigenvalues. We call this an *explicit double shift*. There is a serious problem! Each execution of step 2 costs $O\left(n^3\right)$ flops, so n applications of the double shift will cost $O\left(n^4\right)$ flops, and this is not at all satisfactory. We want to compute the upper Hessenberg matrix $H_2 = (Q)^{\mathrm{T}}H(Q)$ without first performing a QR decomposition of S. By using an implicit process, we can compute a double shift using only $O\left(n^2\right)$ flops. We will proceed like we did for the implicit single shift, except the rotations used will be a little more complex.

We will find Q and thus H_2 by using the implicit Q theorem. We know that $Q^{\mathrm{T}}HQ = H_2$, where $QR = H^2 - (\sigma_1 + \sigma_2)H + \sigma_1\sigma_2 I$. The implicit Q theorem now tells us that we essentially get H_2 using any orthogonal similarity transformation $Z^{\mathrm{T}}HZ$ provided that $Z^{\mathrm{T}}HZ$ is upper Hessenberg, and Q and Z have the same first column or $Qe_1 = Ze_1$.

A calculation shows that the first column of S is (Problem 18.19)

$$C = \begin{bmatrix} (h_{11} - \sigma_1)(h_{11} - \sigma_2) + h_{12}h_{21} \\ h_{21}((h_{11} + h_{22}) - (\sigma_1 + \sigma_2)) \\ h_{21}h_{32} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{18.39}$$

We will use a 7×7 matrix to illustrate the sequence of orthogonal transformations. Form Householder reflection H_{u_1} by letting $\overline{H}_{u_1}$ be the 3×3 reflection that zeros out $h_{21}((h_{11} + h_{22}) - (\sigma_1 + \sigma_2))$ and $h_{21}h_{32}$ and adding the 4×4 identity matrix to form

$$H_{u_1} = \begin{bmatrix} \overline{H}_{u_1} & \\ & I \end{bmatrix} = \begin{bmatrix} * & * & * & & & & \\ * & * & * & & & & \\ * & * & * & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}.$$

Recalling that a Householder reflection is symmetric, now compute

$$H_{u_1}HH_{u_1} = \left[\begin{array}{ccc|cccc} * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ + & * & * & * & * & * & * \\ \hline + & + & * & * & * & * & * \\ & & & * & * & * & * \\ & & & & * & * & * \\ & & & & & * & * \end{array}\right],$$

and obtain a upper Hessenberg matrix with the exception of a 2×2 bulge. We must chase this bulge down to the right and out. Create a 3×3 Householder reflection $\overline{H}_{u_2}$ that is designed to zero out indices (3, 1) and (4, 1) and form the reflection

$$H_{u_2} = \begin{bmatrix} 1 & & \\ & \overline{H}_{u_2} & \\ & & I \end{bmatrix} = \begin{bmatrix} 1 & & & & & & \\ & * & * & * & & & \\ & * & * & * & & & \\ & * & * & * & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}.$$

A premultiplication by H_{u_2} zeroes out indices (3, 1) and (4, 1). Compute

$$H_{u_2}H_{u_1}HH_{u_1}H_{u_2} = \left[\begin{array}{c|ccc|ccc} * & * & * & * & * & * & * \\ \hline * & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & + & * & * & * & * & * \\ \hline & + & + & * & * & * & * \\ & & & & * & * & * \\ & & & & & * & * \end{array}\right],$$

and we have chased the bulge down one row and one column to the right. Create a 3×3 reflection $\overline{H}_{u_3}$ that zeros out indices (4, 2) and (5, 2), and form

$$H_{u_3} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \overline{H}_{u_3} & \\ & & & I \end{bmatrix} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & * & * & * & & \\ & & * & * & * & & \\ & & * & * & * & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}.$$

The computation

$$H_{u_3}H_{u_2}H_{u_1}HH_{u_1}H_{u_2}H_{u_3} = \left[\begin{array}{cc|ccc|cc} * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ \hline & * & * & * & * & * & * \\ & & * & * & * & * & * \\ & & + & * & * & * & * \\ \hline & & + & + & * & * & * \\ & & & & & * & * \end{array}\right],$$

again moves the bulge down. Build a 3×3 reflection $\overline{H}_{u_4}$ that zeros out indices (5, 3) and (6, 3), and form

$$H_{u_4} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \overline{H}_{u_4} & \\ & & & & I \end{bmatrix} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & * & * & * & \\ & & & * & * & * & \\ & & & * & * & * & \\ & & & & & & 1 \end{bmatrix},$$

and compute

$$H_{u_4}H_{u_3}H_{u_2}H_{u_1}HH_{u_1}H_{u_2}H_{u_3}H_{u_4} = \left[\begin{array}{ccc|ccc|c} * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ \hline & & * & * & * & * & * \\ & & & * & * & * & * \\ & & & + & * & * & * \\ \hline & & & + & + & * & * \end{array}\right],$$

For the next step, build a 3×3 reflection $\overline{H}_{u_5}$ that zeros out indices (6, 4) and (7, 4), and form

$$H_{u_5} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & \overline{H}_{u_5} \end{bmatrix} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & * & * & * \\ & & & & * & * & * \\ & & & & * & * & * \end{bmatrix},$$

and then

$$H_{u_5}H_{u_4}H_{u_3}H_{u_2}H_{u_1}HH_{u_1}H_{u_2}H_{u_3}H_{u_4}H_{u_5} = \left[\begin{array}{cccc|ccc} * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & & * & * & * & * & * \\ \hline & & & * & * & * & * \\ & & & & * & * & * \\ & & & & + & * & * \end{array}\right].$$

To complete the chase, construct a 2×2 Householder reflection, $\overline{H}_{u_6}$ that zeros out index (7, 5), form

$$H_{u_6} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & \overline{H}_{u_6} \end{bmatrix} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & * & * \\ & & & & & * & * \end{bmatrix},$$

and the final result is an upper Hessenberg matrix

$$\begin{aligned} H_{u_6}H_{u_5}H_{u_4}H_{u_3}H_{u_2}H_{u_1}HH_{u_1}H_{u_2}H_{u_3}H_{u_4}H_{u_5}H_{u_6} &= QHQ^{\mathrm{T}} \\ &= \left[\begin{array}{ccccc|cc} * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & & * & * & * & * & * \\ & & & * & * & * & * \\ \hline & & & & * & * & * \\ & & & & & * & * \end{array}\right], \end{aligned}$$

where $Q = H_{u_6}H_{u_5}H_{u_4}H_{u_3}H_{u_2}H_{u_1}$.

The 7×7 example motivates the general sequence of $n-1$ Householder reflections:

$$\begin{bmatrix} I^{(i-1)\times(i-1)} & & \\ & \overline{H}_{u_i}^{3\times 3} & \\ & & I^{(n-i-2)\times(n-i-2)} \end{bmatrix}, \quad 1 \le i \le n-2$$

$$\begin{bmatrix} I^{(n-2)\times(n-2)} & \\ & \overline{H}_{u_{n-1}}^{2\times 2} \end{bmatrix}, \quad i = n-1,$$

which give rise to

$$QHQ^{\mathrm{T}} = \overline{H},$$

where $Q = H_{u_{n-1}}H_{u_{n-2}} \ldots H_{u_2}H_{u_1}$ and $\overline{H}$ is an upper Hessenberg matrix. Now,

$$Qe_1 = \left(H_{u_{n-1}} \ldots H_{u_3}H_{u_2}\right)H_{u_1}e_1.$$

The product $\left(H_{u_{n-1}} \ \ldots \ H_{u_3} H_{u_2}\right)$ does not affect column 1 of $H_{u_1} e_1$, so

$$Qe_1 = H_{u_1} e_1.$$

In forming the QR decomposition, $S = ZR$, using Householder reflections, the first column of Z is $H_{u_1} e_1$. By the implicit Q theorem, Q^{T} is a matrix such that

$$H_2 = \left(Q^{\mathrm{T}}\right)^{\mathrm{T}} H \left(Q\right)^{\mathrm{T}},$$

as desired.

The algorithm requires $O\left(n\right)$ flops to apply each reflector, and $n-1$ reflectors are required, so the cost of an iteration is $O\left(n^2\right)$ flops.

Example 18.15. Let

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

Form the 2×2 matrix $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $h_{n-1,n-1} + h_{nn} = 5$, $h_{n-1,n-1} h_{nn} - h_{n,n-1} h_{n-1,n} = -2$, so

$$S = H^2 - 5H - 2I = \begin{bmatrix} -8 & 2 & 2 \\ 6 & -6 & 6 \\ 18 & 0 & 0 \end{bmatrix}.$$

The QR decomposition of S gives

$$Q = \begin{bmatrix} -0.3885 & 0.1757 & 0.9045 \\ 0.2914 & -0.9078 & 0.3015 \\ 0.8742 & 0.3807 & 0.3015 \end{bmatrix},$$

and

$$H_2 = Q^{\mathrm{T}} H Q = \begin{bmatrix} 4.2642 & -1.6270 & 1.9328 \\ 1.6896 & -0.9005 & -4.1500 \\ 0 & 1.8719 & 6.6364 \end{bmatrix}.$$

Now we will perform the double shift using the implicit algorithm.

$$v_1 = \begin{bmatrix} -8 \\ 6 \\ 18 \end{bmatrix} \qquad H_{u_1} = \begin{bmatrix} -0.3885 & 0.2914 & 0.8742 \\ .2914 & 0.9389 & -0.1834 \\ 0.8742 & -0.1834 & 0.4497 \end{bmatrix}$$

$$H_{u_1} H = \begin{bmatrix} -0.1943 & 3.3024 & 3.6909 \\ 7.0900 & 0.0971 & 1.4353 \\ 3.2701 & 0.2914 & 2.3059 \end{bmatrix} \qquad H_{u_1} H H_{u_1} = \begin{bmatrix} 4.2642 & 2.3668 & 0.8840 \\ -1.4716 & 1.8938 & 6.8254 \\ 0.8302 & 0.8034 & 3.8420 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} -1.4716 \\ 0.8302 \end{bmatrix} \qquad H_{u_2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.8710 & 0.4913 \\ 0 & 0.4913 & 0.8710 \end{bmatrix}$$

$$H_{u_2} H_{u_1} H H_{u_1} = \begin{bmatrix} 4.2642 & 2.3668 & 0.8840 \\ 1.6896 & -1.2547 & -4.0570 \\ 0 & 1.6303 & 6.6998 \end{bmatrix} \quad H_{u_2} H_{u_1} H H_{u_1} H_{u_2} = \begin{bmatrix} 4.2642 & -1.6270 & 1.9328 \\ 1.6896 & -0.9005 & -4.1500 \\ 0 & 1.8719 & 6.6364 \end{bmatrix} = H_2$$

Observe that the product $H_{u_2} H_{u_1} H H_{u_1} H_{u_2}$ removed the bulge at (3, 1), and that the implicit computation produced the same value for H_2. Also, compute the orthogonal matrix

$$Q^{\mathrm{T}} = H_{u_1} H_{u_2} = \begin{bmatrix} -0.3885 & 0.1757 & 0.9045 \\ 0.2914 & -0.9078 & 0.3015 \\ 0.8742 & 0.3807 & 0.3015 \end{bmatrix},$$

which is the same matrix obtained by the QR decomposition of S. ■

Building an $n \times n$ Householder reflection like we did in Example 18.15 is not efficient. We should take advantage of the identity and zero block matrices in H_u and do each computation $H_u H H_u$using submatrix operations.

$$
\begin{aligned}
H_{u_i} H &= \begin{bmatrix} I^{(i-1)\times(i-1)} & 0^{(i-1)\times 3} & 0^{(i-1)\times(n-i-2)} \\ 0^{3\times(i-1)} & \overline{H}_{u_i}^{3\times 3} & 0^{3\times(n-i-2)} \\ 0^{(n-i-2)\times(i-1)} & 0^{(n-i-2)\times 3} & I^{(n-i-2)\times(n-i-2)} \end{bmatrix} \begin{bmatrix} H_1^{(i-1)\times(i-1)} & H_2^{(i-1)\times 3} & H_3^{(i-1)\times(n-i-2)} \\ H_4^{3\times(i-1)} & H_5^{3\times 3} & H_6^{3\times(n-i-2)} \\ H_7^{(n-i-2)\times(i-1)} & H_8^{(n-i-2)\times 3} & H_9^{(n-i-2)\times(n-i-2)} \end{bmatrix} \\
&= \begin{bmatrix} H_1^{(i-1)\times(i-1)} & H_2^{(i-1)\times 3} & H_3^{(i-1)\times(n-i-2)} \\ \overline{H}_{u_i}^{3\times 3} H_4^{3\times(i-1)} & \overline{H}_{u_i}^{3\times 3} H_5^{3\times 3} & \overline{H}_{u_i}^{3\times 3} H_6^{3\times(n-i-2)} \\ H_7^{(n-i-2)\times(i-1)} & H_8^{(n-i-2)\times 3} & H_9^{(n-i-2)\times(n-i-2)} \end{bmatrix}
\end{aligned}
$$

The only computation that need be done involves rows $i : i+2$, and columns $1 : n$. The same type of analysis shows that the product $\left(H_{u_i} H\right) H_{u_i}$ is done using rows $1 : n$ and columns $i : i+2$. The final preproduct product using a 2×2 Householder reflection involves rows $n-1 : n$ and columns $1 : n$, followed by the postproduct using rows $1 : n$ and columns $n-1 : n$.

The following algorithm implements the implicit double-shift QR algorithm.

Algorithm 18.5 Implicit Double-Shift *QR*

```
function IMPDSQR(H)
  % H is an n × n upper Hessenberg matrix.
  % Apply one iteration of the double shift implicit QR algorithm,
  % and return upper Hessenberg matrix H_2 and orthogonal
  % matrix Q such that H_2 = QHQ^T.
  Q = I

  % trace of lower right-hand 2 × 2 matrix.
  trce = h_{n-1,n-1} + h_{nn}
  % determinant of lower right-hand 2 × 2 matrix.
  determ = h_{n-1,n-1} h_{nn} - h_{n,n-1} h_{n-1,n}
  % first nonzero entries of column 1 of
  x = h_{11}^2 + h_{12} h_{21} - h_{11} × trce + determ
  y = h_{11} h_{21} + h_{21} (h_{22} - trce)
  z = h_{21} h_{32}

  for i=0:n-3 do
    [ u  β ] = houseparms([ x  y  z ]^T)
    lengu=length(u)
    H̄_u = I^{lengu×lengu} - β u u^T
    H(i+1 : i+3, 1 : n) = H̄_u H(i+1 : i+3, 1 : n)
    H(1 : n, i+1 : i+3) = H(1 : n, i+1 : i+3) H̄_u
    Q(i+1 : i+3, 1 : n) = H̄_u Q(i+1 : i+3, 1 : n)
    x = h_{i+2,i+1}
    y = h_{i+3,i+1}
    if i<n-3 then
      z = h_{i+4,i+1}
    end if
  end for

  [ u  β ] = houseparms([ x  y ]^T)
  lengu=length(u)
  H̄_u = I^{lengu×lengu} - β u u^T
  H(n-1 : n, 1 : n) = H̄_u H(n-1 : n, 1 : n)
  H(1 : n, n-1 : n) = H(1 : n, n-1 : n) H̄_u
  H_2 = H
  return [ Q,  H_2 ]
end function
```

Now, how do we use *impdsqr* to compute all the eigenvalues, real and complex conjugates, of a nonsymmetric matrix? The answer is to use deflation and apply the function *impdsqr* to the current $k \times k$ deflated matrix. The matrix we build is precisely the quasitriangular matrix of the real Schur form presented in Section 18.7.

We need to know when to terminate the iteration and deflate again. To do this we must have a means of detecting what type of blocks are on the diagonal. If the convergence is indicated by a value tol there are two criteria we must consider. See Problem 18.41 for experiments that exhibit the criteria. If $h_{k-1,k-2} <$ tol, there are two real eigenvalues or a complex conjugate pair in the 2×2 submatrix $T(k-1:k,\ k-1:k)$.

$$\begin{array}{|ccc|} \hline & \ddots & \\ \hline & \ddots & \\ \hline \sim 0 & h_{k-1,k-1} & h_{k-1,k} \\ \hline & h_{k,k-1} & h_{kk} \\ \hline \end{array} \quad \left[\ \lambda_{k_1} \quad \lambda_{k_2}\ \right] = \text{eig}\,(H(k-1:k, k-1:k))$$

If $h_{k,k-1} <$ tol, there is a real eigenvalue at $T(k,\ k)$. This is similar to the criteria we used to determine convergence of the shifted Hessenberg iteration in Section 18.6.1.

$$\begin{array}{|ccc|} \hline \ddots & & \\ \hline & \ddots & \\ \hline & h_{k-1,k-1} & h_{k-1,k} \\ \hline & \sim 0 & h_{kk} \\ \hline \end{array} \quad \lambda_k = h_{kk}$$

Remark 18.7. In Ref. [1], it is stated that on the average only two implicit *QR* iterations per eigenvalue are needed for convergence for most matrices. However, it is possible for convergence to fail (Problem 18.27), and when this happens, a special shift every 10 shifts is applied in production versions of the algorithm, such as the `eig` function in MATLAB. The iterations may converge slowly (see Problem 18.27), but this is rare. If the iteration appears to be converging slowly, an iteration with random shifts can be used to keep convergence on track.

A study of the mechanisms involved in the Francis algorithm can be found in Ref. [23, Chapter 6]. The analysis involves integrating a study of the power method, similarity transformations, upper Hessenberg form, and Krylov subspaces. Krylov subspaces are discussed in Chapter 21 when we develop iterative methods for large, sparse matrix problems.

Remark 18.8. The algorithm requires $10n^3$ flops if eigenvalues only are required. It is possible to calculate the corresponding eigenvectors by accumulating transformations (see Ref. [23, pp. 387-389]). In this case, $27n^3$ flops are required.

18.9 COMPUTING EIGENVECTORS

We have discussed ways to compute the eigenvalues of a real matrix, but have not discussed a general method of computing the corresponding eigenvectors. To find an eigenvector corresponding to a given eigenvalue, we use the *shifted inverse iteration*, a variation on the inverse power method for computing the smallest eigenvalue of a matrix. First, we need a lemma that provides a tool needed to develop the inverse iteration algorithm.

Lemma 18.3. *If $(\lambda_i,\ v_i)$ are the eigenvalue/eigenvector pairs of A, $1 \le i \le n$, then $(A-\sigma I)^{-1}$ has eigenvalue/eigenvector pairs $\left(\frac{1}{\lambda_i - \sigma}, v_i\right)$.*

Proof. If A is an $n \times n$ nonsingular matrix, the eigenvalues of A^{-1} are the reciprocals of those for A, and the eigenvectors remain the same. Thus, the result follows immediately from Lemma 18.1. □

If $(\lambda,\ v)$ is an eigenvalue/eigenvector pair of A, and σ is an approximate eigenvalue, then $(\lambda - \sigma)^{-1}$ is likely to be much larger than $\left(\hat{\lambda} - \sigma\right)^{-1}$ for any eigenvalue $\hat{\lambda} \neq \lambda$. Thus, $(\lambda - \sigma)^{-1}$ is the largest eigenvalue of $(A - \sigma I)^{-1}$. Apply the power method to $(A - \sigma I)^{-1}$ to approximate eigenvector v. This is termed the inverse iteration for computing an eigenvector corresponding to an approximate eigenvalue σ.

Just as in Algorithm 18.2, we compute x_{i+1} by solving the system $(A - \sigma I)\, x_{i+1} = x_i$; however, there appears to be a problem. If σ is very close to an eigenvalue λ, and $v \neq 0$ is close to an eigenvector corresponding to λ, then $(A - \sigma I)\, v$ is

close to zero, and $(A - \sigma I)$ is close to being singular. As a result, there may be serious error in the computation of x_{i+1}. As it turns out, the near singularity of $A - \sigma I$ is a good thing. The error at each iteration causes the approximate eigenvector to move closer and closer to the direction of the actual eigenvector. See Ref. [23, pp. 324-325].

Example 18.16. Let A be the matrix of Example 18.7 that has an eigenvalue 2.2518. Choose $\sigma = 2.2000$, $x_0 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}}$, and execute three shifted inverse iterations. The norm of the difference between the computed and actual result is approximately 0.0018431.

```
>> sigma = 2.2000;
>> x0 = ones(3,1);
>> [L U P] = lu(A - sigma*eye(3));
>> v = x0;
>> for i = 1:3
      v = lusolve(L,U,P,v);
      v = v/norm(v);
   end
>> v'

ans =
      0.25979      0.87965     -0.39841
```

■

Now that we have intuitively explained the algorithm and given an example, a proof that it works is in order.

Theorem 18.7. *If the eigenvalues of A satisfy Equation 18.14, the sequence $\{x_k\}$ in the shifted inverse iteration converges to an approximate eigenvector corresponding to the approximate eigenvalue σ.*

Proof. The eigenvalues of $(A - \sigma I)^{-1}$ are $(\lambda_1 - \sigma)^{-1}$, $(\lambda_2 - \sigma)^{-1}$, $\ldots$, $(\lambda_n - \sigma)^{-1}$, and the eigenvectors are the same as A. Assume our approximate eigenvalue σ approximates λ_1 and that λ_1 corresponds to eigenvector v_1, Just as in the proof of Theorem 18.2, we have

$$\begin{aligned}\left((A - \sigma I)^{-1}\right)^k x_0 &= \frac{c_1}{(\lambda_1 - \sigma)^k} v_1 + \frac{c_2}{(\lambda_2 - \sigma)^k} v_2 + \ldots + \frac{c_n}{(\lambda_n - \sigma)^k} v_n \\ &= \frac{1}{(\lambda_1 - \sigma)^k}\left(c_1 v_1 + c_2 \left(\frac{\lambda_1 - \sigma}{\lambda_2 - \sigma}\right)^k v_2 + c_3 \left(\frac{\lambda_1 - \sigma}{\lambda_3 - \sigma}\right)^k v_3 + \ldots + c_n \left(\frac{\lambda_1 - \sigma}{\lambda_n - \sigma}\right)^k v_n\right).\end{aligned}$$

Since σ approximates λ_1, $\frac{1}{|\lambda_1 - \sigma|} > \frac{1}{|\lambda_i - \sigma|}$, $2 \le i \le n$, it follows that $\left|\frac{\lambda_1 - \sigma}{\lambda_i - \sigma}\right| < 1$. The terms $\left(\frac{\lambda_1 - \sigma}{\lambda_i - \sigma}\right)^k$, $2 \le i \le n$ become small as k increases. If we normalize the sequence as in Equation 18.14, $\left((A - \sigma I)^{-1}\right)^k x_0$ converges to a multiple of v_1. □

A function `inverseiter` that implements inverse iteration is an easy modification of `smalleig` and is left to the exercises.

Remark 18.9. Given an accurate approximation to an eigenvalue, the inverse iteration converges very quickly.

18.9.1 Hessenberg Inverse Iteration

If we have an isolated approximation to an eigenvalue σ, the *shifted inverse iteration* can be used to compute an approximate eigenvector. However, if we use the Francis iteration to compute all the eigenvalues of an upper Hessenberg matrix H, we should take advantage of the upper Hessenberg structure of the matrix to find the corresponding eigenvectors. H has the same eigenvalues as A but not the same eigenvectors. However, we can use the orthogonal matrix P in the transformation to upper Hessenberg form to compute an eigenvector of A.

Let u be an eigenvector of $H = P^{\mathrm{T}}AP$ corresponding to eigenvalue λ of A. Then $Hu = \lambda u$, so $P^{\mathrm{T}}APu = \lambda u$ and $A(Pu) = \lambda(Pu)$. Thus, Pu is an eigenvector of A corresponding to eigenvalue λ.

Use shifted inverse iteration with matrix H to obtain eigenvector u, and then $v = Pu$ is an eigenvector of A. Since the inverse iteration requires repeatedly solving a linear system, we use the LU decomposition first. The normal LU decomposition with partial pivoting requires $O\left(n^3\right)$ flops, but we can take advantage of the upper Hessenberg form of H to perform the decomposition more efficiently. Begin by comparing $|h_{11}|$ and $|h_{21}|$ and exchange rows 1 and 2, if necessary,

to place the largest element in magnitude at h_{11}. In general, compare $|h_{ii}|$ and $\left|h_{i+1,i}\right|$ and swap rows if necessary. During the process, maintain the lower triangular matrix

$$L = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ & l_{32} & \ddots & & \\ & & \ddots & 1 & \\ & & & l_{n,n-1} & 1 \end{bmatrix}$$

and the permutation matrix P. The algorithm requires $(n-1)$ divisions $\left(\frac{h_{i+1,i}}{h_{ii}}\right)$ and $2\left[(n-1)+(n-2)+\cdots+1\right] = n(n-1)$ multiplications and subtractions, for a total of n^2-1 flops. Since the algorithm is very similar to ludecomp (Algorithm 11.2), we will not provide a formal specification. The MATLAB function `luhess` in the software distribution implements the algorithm.

Example 18.17. The matrix `EX18_17` is a 500×500 upper Hessenberg matrix. Time its LU decomposition using `ludecomp` developed in Chapter 11, and then time its decomposition using `luhess`. The function `ludecomp` performs general LU decomposition with pivoting, so it does not take advantage of the upper Hessenberg structure. The execution time of `luhess` is approximately 13 times faster than that of `ludecomp`.

```
>> tic;[L1, U1, P1] = ludecomp(EX18_17);toc
Elapsed time is 0.421030 seconds.
>> tic;[L2, U2, P2] = luhess(EX18_17);toc;
Elapsed time is 0.032848 seconds.
```

■

The algorithm eigvechess uses luhess with inverse iteration to compute an eigenvector of an upper Hessenberg matrix with known eigenvalue σ.

Algorithm 18.6 Inverse Iteration to Find Eigenvector of an Upper Hessenberg Matrix

```
function EIGVECHESS(H,σ,x0,tol,maxiter)
   % Computes an eigenvector corresponding to the approximate
   % eigenvalue sigma of the upper Hessenberg matrix H
   % using the inverse iteration.
   % [x iter]=eigvechess(H,sigma,x0,tol,maxiter)
   % sigma is the approximate eigenvalue,
   % x0 is the initial approximation to the eigenvector,
   % tol is the desired error tolerance, and maxiter is
   % the maximum number of iterations allowed.
   % iter=-1 if the method did not converge.

   [L U P]=luhess(A − σI)
   for i=1:maxiter do
      x̄i =lusolve(L,U,P,xi−1)
      xi = x̄i/‖x̄i‖2
      if ‖(A − σI)xi‖2 < tol then
         iter=i
         v=xi
         return [v,i]
      end if
   end for
   iter=-1
   v=xi
   return [v,-1]
end function
```

NLALIB: The function `eigvechess` implements Algorithm 18.6. If A has a multiple eigenvalue σ, Hessenberg inverse iteration can result in vector entries `NaN` or `Inf`. The next section discusses a method that attempts to solve this problem.

Remark 18.10. Although it involves complex arithmetic, `eigvechess` will compute a complex eigenvector when given a complex eigenvalue σ. There is a way to perform inverse iteration with complex σ using real arithmetic (see Ref. [9, p. 630]).

18.10 COMPUTING BOTH EIGENVALUES AND THEIR CORRESPONDING EIGENVECTORS

We have developed the implicit double-shift QR iteration for computing eigenvalues and the Hessenberg inverse iteration for computing eigenvectors. It is now time to develop a function, `eigb`, that computes both. The function applies the Francis iteration of degree two to compute the eigenvalues, followed by the use of the shifted inverse Hessenberg iteration to determine an eigenvector corresponding to each eigenvalue. Inverse iteration is economical because we do not have to accumulate transformations during the Francis iteration. Inverse iteration deals with $H - \sigma I$ using $O\left(n^2\right)$ flops, and normally one or two iterations will produce a suitable eigenvector [2, pp. 394-395]. If A has a multiple eigenvalue, σ, Hessenberg inverse iteration can result in vector entries NaN or Inf. The MATLAB implementation checks for this, perturbs σ slightly, and executes `eigvechess` again with the perturbed eigenvalue. It is hoped this will produce another eigenvector corresponding to σ. All the pieces are in place, so we will not state the formal algorithm that is implemented in the software distribution. Note that it uses the MATLAB features of variable input arguments and variable output to make its use more flexible.

The function can be called in the following ways:

- `[V D] = eigb(A,tol);`
  - Returns a diagonal matrix `D` of eigenvalues and a matrix `V` whose columns are the corresponding normalized eigenvectors so that $AV = VD$. `tol` is the error tolerance for the computations.
    - If `tol` is not present, the default is 1.0×10^{-8}.
- `E = eigb(A,tol);`
  - Assigns `E` a column vector containing the eigenvalues. The default for `tol` is the same as for the previous calling sequence.
- `eigb(A,tol)`
  - Returns a vector of eigenvalues. The default value of `tol` is as before.

Example 18.18. Generate a 75×75 random real matrix that most certainly will have imaginary eigenvalues and use `eigb` to compute its eigenvalues and eigenvectors. A function, `checkeigb`, in the software distribution computes the minimum and maximum values of $\|Av_i - \lambda_i v_i\|_2$, $1 \leq i \leq n$.

```
>> A = randn(75,75);
[V D] = eigb(A,1.0e-12);
[min max] = checkeigb(A,V,D)

min =
   7.4402e-15

max =
   4.3307e-12

>> E = diag(D);
E(1:6)

ans =
      0.25436 +          0i
      0.62739 +          0i
       1.2352 +     1.8955i
       1.2352 -     1.8955i
       2.3356 +    0.90619i
       2.3356 -    0.90619i

>> E(70:75)

ans =
       6.8625 -     1.1636i
```

```
2.7378 +     7.5951i
2.7378 -     7.5951i
7.2777 +     5.6828i
7.2777 -     5.6828i
9.4579 +          0i
```

■

18.11 SENSITIVITY OF EIGENVALUES TO PERTURBATIONS

The eigenvalues of matrix A are the roots of the characteristic equation $p(\lambda) = \det(A - \lambda I)$. If A is the 2×2 matrix

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix},$$

the eigenvalues are roots of the polynomial

$$p(\lambda) = \lambda^2 - (a_{11} + a_{22})\lambda + (a_{11}a_{22} - a_{12}a_{21}).$$

We can view p as a continuous function of the four variables a_{11}, a_{12}, a_{21}, a_{22}, so the roots of the characteristic equation depend continuously on the matrix coefficients. See Refs. [1, 9] for further discussion and proofs. Under the right conditions this continuity means that if the perturbations of A and δA are small, we can control the perturbations in the eigenvalues. The *Bauer-Fike theorem* [91] is a well-known result that deals with eigenvalue perturbations for diagonalizable matrices. For a proof, see Ref. [23, pp. 472-473].

Theorem 18.8. *Assume that $A \in \mathbb{R}^n$ is diagonalizable, so that there exists a nonsingular matrix X such that $X^{-1}AX = D$, where $D = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$. If δA is a matrix of perturbations, then for any eigenvalue $\hat{\lambda}_i$ of $A + \delta A$, there is an eigenvalue λ_i of A such that*

$$\left|\hat{\lambda}_i - \lambda_i\right| \leq \kappa(X) \|\delta A\|_2,$$

where X is a subordinate matrix norm.

Theorem 18.8 has consequences for the conditioning of the eigenproblem for A. If $\|X\|_2 \left\|X^{-1}\right\|_2 = \kappa(X)$ is large, then a computed eigenvalue $\hat{\lambda}_i$ (an eigenvalue for $A + \delta A$) can be very different from the actual eigenvalue λ_i. The more ill-conditioned X is, the more ill-conditioned the eigenproblem for A will be.

Example 18.19. Let $A = \begin{bmatrix} 5.0000 & 0 & 0 \\ 2.0000 & 1.0000 & -7.0000 \\ 3.0000 & 0 & 0.9900 \end{bmatrix}$. The eigenvalues of A are

$$\lambda_1 = 5.000, \quad \lambda_2 = 1.0000, \quad \lambda_3 = 0.9900$$

so A is diagonalizable. The matrix $X = \begin{bmatrix} 0 & 0 & 0.67198 \\ 1 & 1 & -0.54379 \\ 0 & 0.0014286 & 0.50273 \end{bmatrix}$. The condition number of X is 1922.6, so there could be a conditioning problem with the eigenvalues of A. If we let $\delta A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1.0 \times 10^{-5} & 0 \end{bmatrix}$, the eigenvalues of $A + \delta A$ are

$$\lambda_1 = 5.0000, \quad \lambda_2 = 0.9950 + 0.0067082i, \quad \lambda_3 = 0.9950 - 0.0067082i,$$

so the high condition number produces instability. But why were only the eigenvalues λ_2 and λ_3 affected? That question is answered by considering the conditioning of individual eigenvalues. ■

Theorem 18.8 gives one bound for all the eigenvalues, and Example 18.19 demonstrates that some eigenvalues are well conditioned and some are not, so it makes sense to develop a criterion for the conditioning of an individual eigenvalue. To do so requires the concept of a left eigenvector.

Definition 18.4. The vector y is a *left eigenvector* of A if

$$y^{\mathrm{T}}A = \lambda y^{\mathrm{T}}.$$

Remark 18.11. Recall that the determinant of a matrix and its transpose are equal. As a result, the characteristic equations $\det(A - \lambda I) = 0$ and $\det\left(A^{\mathrm{T}} - \lambda I\right) = 0$ have the same roots, so A and A^{T} have the same eigenvalues. Since $y^{\mathrm{T}}A = \lambda y^{\mathrm{T}}$, it follows that $A^{\mathrm{T}}y = \lambda y$, and a left eigenvector of A is an eigenvector of A^{T} corresponding to the eigenvalue λ of A.

Assume that all the eigenvalues are distinct, and that during computation an error is made in A causing perturbation δA. The eigenvalue we actually compute, $\hat{\lambda} = \lambda + \delta\lambda$, is an eigenvalue of $A + \delta A$. Let f be the function mapping δA to $\hat{\lambda}$; in other words, as δA varies, $f(\delta A)$ is the resulting eigenvalue $\hat{\lambda}$. Since the eigenvalues are simple, it can be shown that they depend continuously on the entries of the matrix. As a result, the function f is continuous and

$$\lim_{\delta A \to 0} f(\delta A) = \lambda.$$

This says that as a small error δA occurs, the eigenvalue of $A + \delta A$ will be close to λ. It also follows that if $x + \delta x$ is an eigenvector corresponding to $\lambda + \delta\lambda$, as δA becomes small, δx will have a small norm.

This reasoning leads to the following theorem.

Theorem 18.9. *Assume λ is a simple eigenvalue of the $n \times n$ matrix A having associated right and left eigenvectors x and y, respectively, with $\|x\|_2 = 1$ and $\|y\|_2 = 1$. Let $\lambda + \delta\lambda$ be the corresponding eigenvalue of $A + \delta A$. Then,*

$$|\delta\lambda| \leq \frac{\|\delta A\|_2}{\left|y^{\mathrm{T}}x\right|} + O\left(\|\delta A\|_2^2\right).$$

Proof. We have the two equations

$$Ax = \lambda x \tag{18.40}$$

$$(A + \delta A)(x + \delta x) = (\lambda + \delta\lambda)(x + \delta x) \tag{18.41}$$

Subtract Equation 18.40 from Equation 18.41 and obtain

$$A(\delta x) + (\delta A)x + (\delta A)(\delta x) = \lambda(\delta x) + (\delta\lambda)x + (\delta\lambda)(\delta x).$$

Since we know that if $\|\delta A\|_2$ is small $|\delta\lambda|$ and $\|\delta x\|_2$ will be small, we will ignore the terms $(\delta A)(\delta x)$ and $(\delta\lambda)(\delta x)$ involving products of small factors to arrive at

$$A(\delta x) + (\delta A)x \cong \lambda(\delta x) + (\delta\lambda)x \tag{18.42}$$

Multiply both sides of Equation 18.42 by $y^{\mathrm{T}}$to obtain

$$y^{\mathrm{T}}A(\delta x) + y^{\mathrm{T}}(\delta A)x \cong \lambda y^{\mathrm{T}}(\delta x) + y^{\mathrm{T}}(\delta\lambda)x. \tag{18.43}$$

Since y is a left eigenvector of A, $y^{\mathrm{T}}A = \lambda y^{\mathrm{T}}$, and we have $y^{\mathrm{T}}A(\delta x) = \lambda y^{\mathrm{T}}(\delta x)$. The terms $\lambda y^{\mathrm{T}}(\delta x)$ cancel out in Equation 18.43, leaving

$$y^{\mathrm{T}}(\delta A)x \simeq (\delta\lambda)y^{\mathrm{T}}x,$$

and so

$$\delta\lambda \simeq \frac{y^{\mathrm{T}}(\delta A)x}{y^{\mathrm{T}}x}.$$

It follows that:

$$|\delta\lambda| \leq \left\|y^{\mathrm{T}}\right\|_2 \frac{\|\delta A\|_2}{\left|y^{\mathrm{T}}x\right|}\|x\|_2 = (1)\frac{\|\delta A\|_2}{\left|y^{\mathrm{T}}x\right|}(1) = \frac{\|\delta A\|_2}{\left|y^{\mathrm{T}}x\right|}.$$

Throughout the proof, we have used the notation "$\simeq$" to account for ignoring the terms $(\delta A)(\delta x)$ and $(\delta\lambda)(\delta x)$. δy and δx depend on δA so

$$|\delta\lambda| \leq \frac{\|\delta A\|_2}{\left|y^{\mathrm{T}}x\right|} + O\left(\|\delta A\|_2^2\right).$$

□

Theorem 18.7 indicates that the quantity $\frac{1}{|y^Y x|}$ affects the accuracy of the computation of a particular eigenvalue λ, leading us to the definition of the condition number for λ.

Definition 18.5. If λ is a simple eigenvalue of matrix A, and x, y are normalized right and left eigenvectors of λ, respectively, then

$$\kappa(\lambda) = \frac{1}{|y^T x|}$$

is called the *condition number* of the eigenvalue λ.

Now let us determine how to compute $\kappa(\lambda)$. If A has n distinct eigenvectors, it can be diagonalized, so there is a matrix X such that $D = X^{-1}AX$, where $X = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$ and the v_i are eigenvectors of A corresponding to eigenvalues λ_i. $Av_i = \lambda_i v_i$, and so $x_i = \frac{v_i}{\|v_i\|_2}$ is a normalized right eigenvector of A. By Remark 18.11, if y_i is a normalized right eigenvector of A^T corresponding to eigenvalue λ_i, y_i^T is a left eigenvector of A. Now, $A = XDX^{-1}$, so $A^T = \left(X^T\right)^{-1} DX^T$ and $\left(\left(X^T\right)^{-1}\right)^{-1} A^T \left(X^T\right)^{-1} = D$. The proof of Theorem 18.1 shows that column i of $\left(X^{-1}\right)^T$ is an eigenvector of A^T corresponding to λ_i.

Algorithm 18.7 shows how to compute the condition number of every eigenvalue of matrix A, assuming the eigenvalues are distinct.

Algorithm 18.7 Compute the Condition Number of the Eigenvalues of a Matrix

```
function EIGCOND(A)
   % Compute the condition number of the distinct eigenvalues
   % of the n × n matrix A.
   % [c λ]=eigcond(A) returns a vectors c and λ such that c_i
   % is the condition number of eigenvalue λ_i.

   [ X  D ] = eig(A)
   λ = diag(D)
   invXT = X^-1
   for i=1:n do
      x = X(:,i) / ||X(:,i)||_2
      y = invXT(:,i) / ||invXT(:,i)||_2
      c_i = 1 / |y^T x|
   end for
   return [ c  lambda ]
end function
```

NLALIB: The function `eigcond` implements Algorithm 18.7. The MATLAB function `condeig` also computes eigenvalue condition numbers.

Example 18.20. In Example 18.19, we found that the eigenvalues $\lambda_2 = 1.0000$ and $\lambda_3 = 0.9900$ were sensitive to perturbations. The use of `eigcond` shows why. The condition number of the eigenvalue 5.0000 is 1.4881, and it will not be sensitive to perturbations.

```
>> [c lambda] = eigcond(A)

c =
  874.7007
  874.2160
    1.4881

lambda =
    1.0000
    0.9900
    5.0000
```

■

Example 18.21. The 25×25 matrix EIGBTEST in the book software distribution has distinct integer eigenvalues and a condition number of 4.7250×10^8. The eigenvalues are sensitive to perturbations, as the output shows.

```
>> [V D] = eigb(EIGBTEST);
norm(EIGBTEST*V - V*D)

ans =
  4.0684e-09

>> eigcond(EIGBTEST)

ans =
       412.74
       452.16
         2120
       2389.6
        22744
        33208
        10269
       4115.1
       6047.9
       283.35
   4.2367e+05
    8.313e+06
   1.4719e+07
   1.0591e+07
   3.6274e+06
   2.2256e+05
       3557.7
   2.0129e+06
   2.0363e+06
    1.623e+05
   7.4567e+05
   2.8465e+06
   4.1147e+06
   6.4555e+06
   4.5237e+06

>> [min, max] = checkeigb(EIGBTEST,V,D)

min =
    9.244e-14

max =
    4.066e-09
```

■

Remark 18.12. If λ is a simple eigenvalue of A, then a large condition number implies that A is near a matrix with multiple eigenvalues. If λ is a repeated eigenvalue of a nonsymmetric matrix, the conditioning question is more complicated. The interested reader should consult Ref. [9].

18.11.1 Sensitivity of Eigenvectors

Perturbation analysis for eigenvectors is significantly more complex than that for eigenvalues. The perturbation of a particular eigenvector, v_i, corresponding to eigenvalue, λ_i, is determined by the condition number of the eigenvectors λ_k, $k \neq i$ and $|\lambda_i - \lambda_k|$, $k \neq i$. See Refs. [19, pp. 319-320] and [23, pp. 477-481]. To put it more simply, if the eigenvalues are well separated and well conditioned, then the eigenvectors will be well conditioned. However, if there is a multiple eigenvalue or there is an eigenvalue close to another eigenvalue, then there will be some ill-conditioned eigenvectors. Problem 18.22 deals with eigenvector sensitivity theoretically, and Problems 18.37 and 18.38 are numerical experiments dealing with the problem.

18.12 CHAPTER SUMMARY

Applications of the Eigenvalue Problem

Resonance is the tendency of a system to oscillate at a greater amplitude at some frequencies than at others. With little or no damping, if a periodic driving force matches a natural frequency of vibration, oscillations of large amplitude can occur. The are many types of resonance, including mechanical resonance, acoustic resonance, electromagnetic resonance, and nuclear magnetic resonance. The phenomenon is illustrated by a mass-spring system that is modeled by a system of second-order ordinary differential equations. The solution to the system with no driving force (homogeneous system) depends on eigenvalue and eigenvector computations, that give rise to the natural frequencies of vibration. After adding a driving force and solving the nonhomogeneous system, resonance is illustrated by choosing the frequency of the driving force to be close to a natural frequency.

The model of Leslie is a heavily used tool in population ecology. It models an age-structured population which predicts how distinct populations change over time. The heart of the model is the Leslie matrix, which is irreducible and has an eigenvector with strictly positive entries. Its characteristic function has exactly one real positive root, λ_1, the largest eigenvalue of the Leslie matrix in magnitude. All the other eigenvalues are negative or imaginary. By studying powers of the Leslie matrix, we find that at equilibrium the proportion of individuals belonging to each age class will remain constant, and the number of individuals will increase by λ_1 times each period. The eigenvector with all positive entries can be used to determine the percentage of females in each age class after an extended period of time.

The buckling of a elastic column is determined by solving a boundary value problem. The column will not buckle unless subjected to a critical load, in which case the deflection curve is of the form $k \sin\left(\frac{n\pi x}{L}\right)$, where L is the length of the column and k is a constant. The functions are termed eigenfunctions, and the corresponding eigenvalues are $\lambda_n = \frac{n^2\pi^2}{L^2}$. Associated with the eigenvalues are the critical loads $P_n = \frac{EI\pi^2 n^2}{L^2}$, the only forces that will cause buckling.

Computation of Selected Eigenvalues and Eigenvectors

There are problems for which only selected eigenvalues and associated eigenvectors are needed. If a real matrix has a simple eigenvalue of largest magnitude, the sequence $x_k = Ax_{k-1}$ converges to the eigenvector corresponding to the largest eigenvalue, where x_0 is a normalized initial approximation, and all subsequent x_k are normalized. This is known as the power method. After k iterations, the corresponding eigenvalue is approximately $\lambda_1 = v_k^{\mathrm{T}}(Av_k)$.

Assume A has a simple eigenvalue of smallest magnitude. Since the eigenvalues of A^{-1} are the reciprocals of the eigenvalues of A, the smallest eigenvalue of A is the largest eigenvalue of A^{-1}, and we can compute it by applying the power method to A^{-1}. This is not done by using the iteration $x_k = A^{-1}x_{k-1}$ but by solving the system $Ax_k = x_{k-1}$ for each k.

The Basic *QR* Iteration

The discovery of the *QR* iteration is one of the great accomplishments in numerical linear algebra. Under the right conditions, the following sequence converges to an upper triangular matrix whose diagonal consists of the eigenvalues in decreasing order of magnitude:

$$\begin{aligned} A_{k-1} &= Q_k R_k \\ A_k &= R_k Q_k. \end{aligned}$$

The execution of one *QR* iteration requires $O\left(n^3\right)$ flops, so k iterations requires $O\left(kn^3\right)$ flops. If $k = n$, this is an $O\left(n^4\right)$ algorithm, and this is not satisfactory. However, there are ways to greatly speed it up.

Transformation to Upper Hessenberg Form

Transformation to upper Hessenberg form is the initial step in most algorithms for the computation of eigenvalues. It is accomplished by using orthogonal similarity transformations of the form $A_k = H_{u_k} A_{k-1} H_{u_k}^{\mathrm{T}}$, where H_{u_k} is a Householder reflection. The end result is an upper Hessenberg matrix $H = P^{\mathrm{T}}AP$, where P is an orthogonal matrix comprised of products of Householder matrices. The eigenvalues of H are the same as those of A.

The Unshifted Hessenberg *QR* Iteration

The reduction of a matrix A to upper Hessenberg form requires approximately $\frac{10}{3}n^3$ flops for the computation of H. To build the orthogonal matrix P requires an additional $4n^3/3$ flops. The reduction of H to upper triangular form using the QR iteration with deflation requires only $O\left(n^2\right)$ flops. By an initial reduction to upper Hessenberg form followed by the Hessenberg QR iteration, we can compute eigenvalues with $O\left(n^3\right) + O\left(n^2\right)$ flops.

The Shifted Hessenberg *QR* Iteration

The QR iteration is more effective when it is applied to compute an eigenvalue isolated from other eigenvalues. Let H be an upper Hessenberg matrix. Choose a shift σ close to λ_k, and form $H_k - \sigma I$ that has eigenvalues $\lambda_i - \sigma$, $1 \leq i \leq n$. The eigenvalue $\lambda_k - \sigma$ is smaller than all the other eigenvalues, and the QR iteration applied to $H_k - \sigma I$ is very accurate. Compute the QR decomposition of the matrix $H_k - \sigma_1 I = Q_k R_k$ and let $H_{k+1} = R_k Q_k + \sigma_1 I$. Repeat this process until $\left|h_{k,k-1}\right|$ is sufficiently small. This technique significantly speeds up the computation of eigenvalues.

The Francis Algorithm

The Francis algorithm has for many years been the staple for eigenvalue computation. By using a double shift, it enables the computation of complex conjugate pairs of eigenvalues without using complex arithmetic. The algorithm is also known as the implicit QR iteration because it indirectly computes a single and a double shift in an upper Hessenberg matrix without actually computing the QR decomposition. This is done using orthogonal similarity transformations that introduce bulges, disturbing the upper Hessenberg form. By moving down the matrix, the transformations chase the bulge down and off the Hessenberg matrix. The algorithm makes the computation of all the eigenvalues of a matrix run in $O\left(n^3\right)$ flops.

Computing Eigenvectors

If λ_k is an accurate eigenvalue of matrix A, apply the inverse power iteration to the matrix $H_k - \lambda_k I$ to find a normalized eigenvector v_k, which is also an eigenvector of H_k. This means solving linear systems of the form $\overline{H} x_k = x_{k-1}$. Since $\overline{H}$ is upper Hessenberg, a simple modification of the LU decomposition enables very rapid computation of $P\overline{H} = LU$.

Computing Both Eigenvalues and Their Corresponding Eigenvectors

An eigenproblem solver, eigb, is easy to build using the algorithms developed in the book. First, reduce matrix A to upper Hessenberg form $H = P^{\mathrm{T}} A P$ and compute the eigenvalues by applying the Francis double-shift QR iteration to H. Now find corresponding eigenvectors of H using inverse iteration. For each eigenvector v of H, Pv is an eigenvector of A.

Sensitivity of Eigenvalues and Eigenvectors to Perturbations

Assume that A is diagonalizable so that $X^{-1} A X = D$. The Bauer-Fike theorem says that if δA is a matrix of perturbations, then for any eigenvalue $\hat{\lambda_i}$ of $A + \delta A$, there is an eigenvalue λ_i of A such that

$$\left|\hat{\lambda_i} - \lambda_i\right| \leq \kappa\left(X\right) \|\delta A\|_2$$

This is a global estimate, meaning it applies to all eigenvalues. If $\kappa\left(X\right)$ is large, it is possible that one or more eigenvalues are ill-conditioned.

The condition number of an individual eigenvalue λ is defined by $\kappa\left(\lambda\right) = \frac{1}{|y^{\mathrm{T}} x|}$, where x is a right and y is a left normalized eigenvector, respectively. If λ is a simple eigenvalue of A, then a large condition number implies that A is near a matrix with multiple eigenvalues.

If the eigenvalues are well separated and well conditioned, then the eigenvectors will be well conditioned. However, if there is a multiple eigenvalue or there is an eigenvalue close to another eigenvalue, then there will be some ill-conditioned eigenvectors.

18.13 PROBLEMS

18.1 Consider the coupled mass vibration problem consisting of springs having the same mass attached to each other and fixed outer walls by springs of spring constants k_1 and k_2 (Figure 18.11).

The displacements x_1 and x_2 satisfy the system of second-order differential equations

$$m\frac{d^2x_1}{dt^2} + k_1x_1 - k_2(x_2 - x_1) = F_0 \sin \omega_0 t$$
$$m\frac{d^2x_2}{dt^2} + k_1x_2 + k_2(x_2 - x_1) = 0$$

Leaving ω_0 variable, solve the system given $m = 1\,\text{kg}$, $k_1 = 1\,\text{N/m}$, $k_2 = 2\,\text{N/m}$, $F_0 = 2\,\text{N}$, and initial conditions $x_1(0) = 3, x_2(0) = 1, x_1'(0) = x_2'(0) = 0$. Demonstrate resonance by making ω_0 near one of the natural frequencies of the system and graphing $x_1(t)$, $x_2(t)$ for $0 \le x \le 5$.

18.2 Using paper and pencil, execute the first three iterations of the power method for computing the largest eigenvalue of $A = \begin{bmatrix} 1 & 3 \\ 1 & 1 \end{bmatrix}$.

18.3 Using paper and pencil, execute the first three iterations of the inverse power method for computing the smallest eigenvalue of the matrix in Problem 18.2.

18.4 Explain the slow rate of convergence of the power method for the matrices

a. $A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 10 & 0 \\ 1 & 1 & 9.8 \end{bmatrix}$.

b. $\begin{bmatrix} 2.9910 & 1.2104 & 0.7912 \\ -1.4082 & 0.5913 & -2.8296 \\ -0.1996 & -0.3475 & 2.0678 \end{bmatrix}$.

18.5 Explain what happens when the inverse power method is applied to the matrix

$$A = \begin{bmatrix} 0.7674 & 0.2136 & 3.3288 \\ -0.7804 & -0.9519 & -0.4240 \\ 0.5086 & -0.1812 & 2.1899 \end{bmatrix}.$$

18.6 **a.** Prove that the left eigenvectors of a symmetric matrix are eigenvectors.

b. Let λ and μ be two distinct eigenvalues of A. Show that all left eigenvectors associated with λ are orthogonal to all right eigenvectors associated with μ.

18.7 Let $A = \begin{bmatrix} 3.8865 & 0.29072 & 1.6121 \\ -1.6988 & 2.6922 & -4.3539 \\ -0.50715 & -0.04681 & 1.4712 \end{bmatrix}$. Use MATLAB to compute its eigenvalues. Do you think using the shift strategy will enable more rapid computation of the eigenvalues? Explain.

18.8 If an upper Hessenberg matrix has a zero on the subdiagonal, the problem must be split into two eigenvalue problems. Assume a split has occurred and the matrix has the form

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix},$$

where A_{11} is $i \times i$, A_{22} is $(n - i) \times (n - i)$, and both are upper Hessenberg. Show that the eigenvalues of H are those of A_{11} and A_{22}. For the sake of simplicity, assume that A_{11} and A_{22} have no common eigenvalues and that if λ is an eigenvalue of A_{22} then the matrix $A_{11} - I$ is nonsingular.

18.9 Let A be an $n \times n$ matrix.

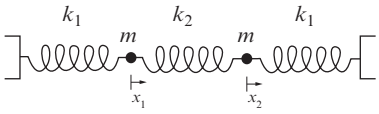

FIGURE 18.11 Springs problem.

a. If λ is an eigenvalue of A and $v = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}^{\mathrm{T}}$ is an associated eigenvector, use $Av = \lambda v$ to show that

$$(\lambda - a_{ii})\, v_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} v_j, \quad i = 1, 2, \dots, n.$$

b. Let v_k be the largest component of v in absolute value. Using part (a), show that

$$|\lambda - a_{kk}| \leq \sum_{\substack{j=1 \\ j \neq k}}^{n} \left|a_{kj}\right|.$$

c. Using part (b), show that each eigenvalue of A satisfies at least one of the inequalities

$$|\lambda - a_{ii}| \leq r_i, \quad 1 \leq i \leq n,$$

where

$$r_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} \left|a_{ij}\right|.$$

d. Argue this means that all the eigenvalues of A lie in the union of the disks $\{z \mid |z - a_{ii}| \leq r_i\}$, $i = 1, 2, \dots, n$ in the complex plane. This result is one of *Gergorin's disk theorems* that are used in perturbation theory for eigenvalues.

e. Let

$$A = \begin{bmatrix} 9 & 5 & 4 \\ 10 & 3 & 1 \\ 5 & 9 & 8 \end{bmatrix}.$$

Draw the three Gergorin disks and verify that the three eigenvalues lie in their union.

18.10 A square matrix is strictly row diagonally dominant if $|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}$. Use Gergorin's disk theorem developed in Problem 18.9 to prove that a strictly row diagonally dominant matrix is nonsingular.

18.11 In this problem, you are to show that if Equation 18.14 holds, as the QR iterations progress, the elements of $A_k = \overline{Q}_{k-1}^{\mathrm{T}} A \overline{Q}_{k-1}$ at index (1, 1) converge to the largest eigenvalue λ_1 by the power method.

a. Let $Q^{(k)} = Q_1 Q_2 \dots Q_{k-1} Q_k$, and show that $AQ^{(k-1)} = Q^{(k)} R_k$. Hint:

$$\begin{aligned} (Q_1 Q_2 \dots Q_{k-1} Q_k)\, R_k &= Q_1 Q_2 \dots Q_{k-1}\, (Q_k R_k) \\ &= Q_1 Q_2 \dots Q_{k-1} A_{k-1} \\ &= Q_1 Q_2 \dots Q_{k-1}\, (R_{k-1} Q_{k-1}) = \cdots \end{aligned}$$

b. $Q^{(k-1)}$ has n orthonormal columns, so $Q^{(k-1)} = \begin{bmatrix} q_1^{(k-1)} & q_2^{(k-1)} & \dots & q_{n-1}^{(k-1)} & q_n^{(k-1)} \end{bmatrix}$. Let $r_{11}^{(k)}$ be entry $R_k\,(1,1)$ and equate the first columns on both sides of the result from part (a) to develop the relation

$$A q_1^{(k-1)} = r_{11}^{(k)} q_1^{(k)}.$$

c. Argue that the result of part (b) shows that $q_1^{(k)}$ converges by the power method to an eigenvector of A with associated eigenvalue $r_{11}^{(k)}$.

18.12 Show that the QR decomposition of an upper Hessenberg matrix using Givens rotations costs $O\left(n^2\right)$ flops.

18.13 Given $A = \begin{bmatrix} 1 & 1 \\ 0 & 1+\epsilon \end{bmatrix}$, find the eigenvector matrix S such that $S^{-1}AS$ is diagonal. Use the Bauer-Fike theorem to show that the eigenvalues are ill-conditioned.

18.14 Show that an orthogonal similarity transformation maintains the condition number of an eigenvalue.

A real matrix is said to be *normal* if $A^{\mathrm{T}}A = AA^{\mathrm{T}}$. Problems 18.15-18.17 deal with properties of a normal matrix.

18.15 There are matrices that can be diagonalized, but are not normal. Let $A = \begin{bmatrix} -1 & 3 \\ 0 & 2 \end{bmatrix}$ and $P = \begin{bmatrix} 1 & \frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$.

a. P has normalized columns. Is it orthogonal?
b. Show that A can be diagonalized using the matrix P.
c. Show that A is not normal.

18.16 Prove that
a. a symmetric matrix is normal.
b. an orthogonal matrix is normal.
c. if A is *skew symmetric* $\left(A^{\mathrm{T}} = -A\right)$, then A is normal.
d. If there is a polynomial p such that $A^{\mathrm{T}} = p(A)$, then A is normal.

18.17 Using the Bauer-Fike theorem, show that if A is normal, and λ is an eigenvalue of $A + \delta A$, then

$$\min_{1 \le i \le n} |\lambda_i - \lambda| \le \|\delta A\|_2 .$$

What does this say about the conditioning of the eigenvalues?

18.18 Show that Equation 18.37 is correct.

18.19 Show that Equation 18.39 is correct.

18.20 What are the condition numbers for the eigenvalues of a symmetric matrix?

18.21 Using the matrix of Example 18.13, show that the Schur's triangularization is not unique.

18.22 In Watkins [23, p. 480], it is shown that if λ_k is a simple eigenvalue of A, then

$$\max_{\substack{1 \le i \le n \\ i \ne k}} |\lambda_i - \lambda_k|^{-1} = 1 / \min_{\substack{1 \le i \le n \\ i \ne k}} |\lambda_i - \lambda_k|$$

is a lower bound for the condition number of an eigenvector corresponding to λ_k. Under what conditions will this formula indicate that an eigenvector is ill-conditioned?

18.23 Given two matrices A and B, the generalized eigenvalue problem is to find nonzero vectors v and a number λ such that $Av = \lambda Bv$. The matrix $A - \lambda B$ is called a *matrix pencil*, usually designated by (A, B). The characteristic equation for the pencil (A, B) is $\det(A - \lambda B) = 0$, and the generalized eigenvalues are the roots of the characteristic polynomial. See Ref. [19, Chapter 11] or [23, pp. 526-541] for a discussion of the problem.
a. How does this problem relate to the standard eigenvalue problem?
b. Show that the degree of the characteristic equation is less than or equal to n. Give an example where the degree is less than n. In general, when is the degree guaranteed to be less than n?
c. Assume that B is nonsingular. Show that λ is an eigenvalue of (A, B) with associated eigenvector v if and only if λ is an eigenvalue of $B^{-1}A$ with associated eigenvector v.
d. Show that the nonzero eigenvalues of (B, A) are the reciprocals of the nonzero eigenvalues of (A, B).
e. Find the generalized eigenvalues for (A, B) by hand. Using `eig`, find corresponding eigenvectors.

$$A = \begin{bmatrix} 1 & 2 \\ -1 & 5 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 5 \\ 1 & 2 \end{bmatrix}$$

18.13.1 MATLAB Problems

18.24 Suppose a population has five age classes, and that the following table specifies the data for the population.

Age Class	$n_i(0)$	f_i	p_i
1	8	0	0.60
2	10	6	0.45
3	12	4	0.25
4	7	3	0.15
5	5	2	

a. Find the Leslie matrix, L, for this population.
b. Find the eigenvalues and eigenvectors of L.
c. Find the rate of increase of the number of individuals at equilibrium for each time period.
d. Graph the age population for each age group over a 10-year period using a logarithmic scale on the vertical axis.

18.25 Compute the largest eigenvalue of the matrix

$$A = \begin{bmatrix} -68 & 20 & -10 & 65 & -79 \\ 59 & -48 & -84 & 8 & 93 \\ -38 & 31 & -54 & 100 & -100 \\ 6 & 38 & 83 & -85 & 55 \\ -67 & 50 & -70 & -12 & 64 \end{bmatrix}$$

using the power method.

18.26 In this problem, you will compare the execution time and accuracy of `eigqrbasic`, `eigqr`, `eigqrshift`, and `eigb`.

a. Execute the following:

```
A = gallery('gearmat',35);
tic;E1 = eigqrbasic(A,5000);toc;
tic;E2 = eigqr(A,1.0e-10,1000);toc
tic;E3 = eigqrshift(A,1.0e-10,50);toc;
tic;E4 = eigb(A,1.0e-10,50);toc;
```

Comment on the results.

b. Using the MATLAB `sort` command, sort `E1`, `E2`, `E3`, `E4`, and compute `E = sort(eig(A))`. Compute the accuracy of each of the four functions against that of `eig` by computing $\|E - E_i\|_2$, $i = 1, 2, 3, 4$. Explain the results.

18.27 **a.** Except for signs, no elements of the $n \times n$ matrix

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

change for $m < n$ executions of the Francis iteration of degree two [9]. If n is large, this creates a very slow algorithm. When lack of convergence is detected, the situation can be resolved by replacing the Francis double shift by an exceptional shift after 10-20 iterations [56]. Build a 25×25 version of this matrix, apply `impdsqr` 24 times, and then execute `diag(A,-1)` and `A(1,25)`. Repeat the experiment by applying `impdsqr` 35 times and 100 times. Comment on the results.

b. The function `eigb` terminates if any 2×2 or 1×1 eigenvalue block fails to converge in a default of 1000 iterations. In Ref. [57], it is noted that there are small sets of matrices where the Francis algorithm fails to converge in a reasonable number of iterations. Let

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

and apply `eigb` to A for $h = 1.0 \times 10^{-6}$. Comment on the result. Does `eig` give results? If so, why?

18.28 The function `trid` in the book software distribution takes three vectors $a^{(n-1)\times 1}$, $b^{n\times 1}$, and $c^{(n-1)\times 1}$ and builds a tridiagonal matrix with subdiagonal a, diagonal b, and superdiagonal c. Use `trid` to construct the 200×200 matrix

$$A = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & 0 & \cdots & -1 & 2 \end{bmatrix}.$$

a. Is A symmetric?

b. Is A positive definite?

c. Does A have distinct eigenvalues?
d. Apply the power method and the inverse power method to estimate the largest and smallest eigenvalues of A.
e. Is A ill-conditioned?
f. Find the condition number for each eigenvalue of A. Explain the result.

18.29 For this problem, we will use the MATLAB command `eig`. The 20×20 Wilkinson upper bidiagonal matrix has diagonal entries 20, 19, …, 1. The superdiagonal entries are all equal to 20. The aim of this problem is to compute the condition number of the Wilkinson matrix and study the sensitivity of its eigenvalues by executing the series of steps:
a. Build W.
b. Compute $\kappa\,(W)$.
c. Find the eigenvalues, $\lambda_1, \lambda_2, \ldots, \lambda_{20}$, and the matrix of eigenvectors X.
d. Compute $\kappa\,(X)$.
e. Find c, the vector of eigenvalue condition numbers.
f. Let $\hat{W}$ be the matrix obtained from W by assigning $w_{20,1} = 1.0 \times 10^{-10}$ and recompute the eigenvalues $\hat{\lambda_1}, \hat{\lambda_2}, \ldots, \hat{\lambda_{20}}$. Sort the eigenvalues in descending order.
g. For $i = 1, 2, \ldots, 20$ print $\quad \lambda_i, \quad \hat{\lambda_i}, \quad \left|\lambda_i - \hat{\lambda_i}\right|, \quad c_i$

h. Explain the results.

18.30
a. Write a MATLAB function `wilkcondeig(n)` that builds the upper Hessenberg matrix

$$\begin{bmatrix} n & n-1 & \ldots & 2 & 1 \\ n-1 & n-1 & n-2 & \ddots & 1 \\ & n-2 & \ddots & \ddots & \\ & & \ddots & 2 & 1 \\ & & & 1 & 1 \end{bmatrix}.$$

b. Compute the eigenvalue condition numbers of `W = wilkcondeig(20)`.
c. Compute the eigenvalues of W.
d. Let $\hat{W}$ be the matrix W with the value at index (20,1) set to 1.0×10^{-10}. Compute the eigenvalues of $\hat{W}$. Do the perturbations correspond to what you would expect from the condition numbers?
Problem 18.40 studies this matrix in more detail.

18.31 Let A be the matrix of Problem 18.7. Perform the following tests.
a. Execute the statements and explain the results.

```
>> E1 = eigqr(A,1.0e-10,10)
>> E2 = eigqrshift(A,1.0e-10,10)
>> eig(A)
```

b. Beginning with maxiter = 500, determine the smallest value of maxiter so that

```
>> E = eigqr(A,1.0e-10,maxiter)
```

terminates successfully.
c. Beginning with 1, find the smallest value of maxiter so that

```
>> E = eigqrshift(A,1.0e-10,maxiter)
```

terminates successfully.
d. Now execute

```
>> E = eigqrshift(A,1.0e-14,10);
>> EM = eig(A)
>> norm(E-EM)
```

e. Make a general statement about what you have learned from this problem.

18.32 Consider the matrix $A = \begin{bmatrix} 8 & 3 & 0 \\ 0 & 7.9999 & 0 \\ 2 & 5 & 10 \end{bmatrix}$.

a. Diagonalize A to obtain the eigenvector matrix X and the diagonal matrix of eigenvalues, D. Now let $E = 10^{-6}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$, and find the eigenvalues of $A + E$. What happens?

b. Compute $\kappa(X)$ and $\kappa(X)\|E\|_2$. Why is the behavior in part (a) predicted by the Bauer-Fike theorem? Which eigenvalues are actually ill-conditioned, and why?

18.33 Let

$$A = \begin{bmatrix} -3 & 1/2 & 1/3 \\ -36 & 8 & 6 \\ 30 & -15/2 & -6 \end{bmatrix}, \quad \delta A = \begin{bmatrix} 0.005 & 0 & 0 \\ 0 & 0.005 & 0.005 \\ 0 & 0.005 & 0 \end{bmatrix}, \quad \text{and} \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

a. Compute the eigenvalues of A and the solution to $Ax = b$.

b. Let $\hat{A} = A + \delta A$. Compute the eigenvalues of $\hat{A}$ and the solution of $\hat{A}x = b$.

c. Explain the results.

18.34 Use MATLAB to compute the left eigenvectors of the matrix

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

18.35 `A = gallery('clement',50)` returns a nonsymmetric 50×50 tridiagonal matrix with zeros on its main diagonal. The eigenvalues are $\pm 49, \pm 47, \pm 45, \ldots, \pm 3, \pm 1$. Since $|\lambda_1| = |\lambda_2| > |\lambda_3| = |\lambda_4| > \cdots > |\lambda_{49}| = |\lambda_{50}|$, the basic and the unshifted Hessenberg QR iterations may not converge. Fill-in the table and, if a method does not converge, indicate this in the last column. Sort the eigenvalues using `sort` prior to computing $\|E - E_i\|_2$, $i = 1, 2, 3, 4$. Discuss the results.

Number	Function	Time	$\|E - E_i\|_2$
	`E = eig(A);`		
1	`E1 = eigqrbasic(A,1000);`		
2	`E2 = eigqr(A,1.0e-12,500);`		
3	`E3 = eigqrshift(A,1.0e-12,500);`		
4	`E4 = eigb(A,1.0e-12,500);`		

18.36 Implement the shifted inverse iteration with a function

```
[x, iter] = inverseiter(A, sigma, x0, tol, maxiter)
```

where A is any real matrix with distinct eigenvalues. Use it to find eigenvectors corresponding to the given eigenvalues.

a. $A = \begin{bmatrix} 1 & 7 & 8 \\ 3 & 1 & 5 \\ 0 & -1 & -8 \end{bmatrix}$, $\lambda = 5.1942,\ \lambda_2 = -3.5521.$

b. $A = \begin{bmatrix} -1 & 3 & 5 & 3 \\ 22 & -1 & 8 & 2 \\ 9 & 12 & 3 & 7 \\ 25 & 33 & 55 & 35 \end{bmatrix}$,

$\lambda = 50.7622,\ \lambda_2 = 0.8774,\ \lambda_3 = -6.7865,\ \lambda_4 = -8.8531$

c. $A = \begin{bmatrix} 2 & 7 & 5 & 9 & 2 \\ 8 & 3 & 1 & 6 & 10 \\ 4 & 7 & 3 & 10 & 1 \\ 6 & 7 & 10 & 1 & 8 \\ 2 & 8 & 2 & 5 & 9 \end{bmatrix}$, $\lambda_1 = -9.4497,\ \lambda_2 = -1.8123.$

18.37 Input the 8×8 matrix `EIGVECTST.mat` in the software distribution. Perform these actions in the order given:

a. Compute the eigenvector matrix `V1` and the eigenvalue matrix `D1` for `EIGVECTST`.

b. Output the eigenvalues of `EIGVECTST`.

c. Create a matrix perturbation as follows:

```
deltaEIGVECTST = ones(8,8);
for i = 1:8
   for j = 1:8
      deltaEIGVECTST(i,j) = rand*1.e-6*deltaEIGVECTST(i,j);
   end
end
```

Perturb `EIGVECTST` by computing `EIGVECTST_hat = EIGVECTST + deltaEIGVECTST;`
Compute $\|EIGVECTST - EIGVECTST\_hat\|_2$.
Find the eigenvector matrix `V2` and the eigenvalue matrix `D2` for `EIGVECTST_hat`.
Output the eigenvalues of `EIGVECTST_hat`.
Output $\|V1 - V2\|_2$.
Explain the results.

18.38 Let $A = \begin{bmatrix} 1 & 0 \\ 0 & 1+\epsilon \end{bmatrix}$. A is symmetric and has two nearly equal eigenvalues if ϵ is small.

a. For $\epsilon = 10^{-3}$, 10^{-5}, 10^{-8}, do the following:

i. Compute the eigenvalues and eigenvectors of A.

ii. Let $\delta A = 10^{-8}$ randn (2), where randn(2) is a 2×2 matrix of random entries drawn from the standard normal distribution. Compute the eigenvalues and eigenvectors of $A + \delta A$.

b. Perform the same numerical experiment as in part (a) using the matrix $A = \begin{bmatrix} 1+\epsilon & 0 & 0 \\ 0 & 1-\epsilon & 0 \\ 0 & 0 & 2 \end{bmatrix}$.

c. Make a general statement about the conditioning of eigenvalues and eigenvectors of a symmetric matrix.

18.39 This problem assumes access to the MATLAB Symbolic Math Toolbox. Let

$$A = \begin{bmatrix} 3+\epsilon & 5 \\ 0 & 3 \end{bmatrix}.$$

a. Show that the condition number of both eigenvalues of A is

$$c_1 = c_2 = \sqrt{\frac{25}{\epsilon^2} + 1}$$

by executing the following statements

```
>> syms A epsilon X invX x y c
>> A = sym([3+epsilon 5;0 3])
>> [X D] = eig(A)
>> invX = inv(X)
>> x = X(:,1)/norm(X(:,1))
>> y = invX(:,1)/norm(invX(:,1))
>> c1 = 1/abs(y'*x)
>> x = X(:,2)/norm(X(:,2))
>> y = invX(:,2)/norm(invX(:,2))
>> c2 = 1/abs(y'*x)
```

b. Using the formula, compute the condition number of the eigenvalues for $\epsilon = 1.0 \times 10^{-6}$.

c. By entering

$$B = \begin{bmatrix} 3+\epsilon & 5 \\ 0 & 3 \end{bmatrix},$$

compute the condition number of the eigenvalues using floating point arithmetic.

d. Explain why the matrix A has ill-conditioned eigenvalues.

18.40 This problem is motivated by a discussion in the classic book by Wilkinson [9, pp. 92-93]. Consider the class of matrices

$$B_n = \begin{bmatrix} n & (n-1) & (n-2) & \dots & 3 & 2 & 1 \\ (n-1) & (n-1) & (n-2) & \dots & 3 & 2 & 1 \\ & (n-2) & (n-2) & \dots & 3 & 2 & 1 \\ & & \dots & & & & \\ & & & & 2 & 2 & 1 \\ & & & & & 1 & 1 \end{bmatrix}.$$

a. If you have not done Problem 18.30, write a MATLAB function, `wilkcondeig`, that builds B_n.
b. Using MATLAB, let $n = 20$ and show that the largest eigenvalues of B_{20} are well-conditioned, while the smallest ones are quite ill-conditioned.
c. Show that the determinant of B_n is one for all n.
d. If we assign $B_n(1, n) = 1 + \epsilon$, the determinate becomes

$$1 \pm (n-1)!\epsilon.$$

Let $n = 20$, $B(1, 20) = 1 + 1.0 \times 10^{-10}$, and compute the determinant.
e. Prove that the determinant of a matrix is equal to the product of its eigenvalues.
f. Explain why there must be large eigenvalues in the perturbed B_n?

18.41 This problem illustrates convergence of the double shift implicit QR iteration to a single real or a complex conjugate pair of eigenvalues.
a. Create the upper Hessenberg Toeplitz matrix

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 2 & 3 & 4 \\ 0 & 1 & 1 & 2 & 3 \\ 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

as follows:

```
>> H =  toeplitz([1 1 0 0 0], [1 2 3 4 5]);
```

b. Compute its eigenvalues using `eig`.
c. Execute

```
>> [~, H] = impdsqr(H)
```

six times and observe what happens in the lower right-hand corner of H. Comment.
d. Execute

```
>> [~, H(1:4,1:4)] = impdsqr(H(1:4,1:4))
```

six times and then apply the statement

```
eig(H(3:4,3:4))
```

Comment.

18.42 The execution of the MATLAB command `help gallery` contains the following entry

"gallery(5) is an interesting eigenvalue problem. Try to find its EXACT eigenvalues and eigenvectors."

Let

$$A = \text{gallery}(5) = \begin{bmatrix} -9 & 11 & -21 & 63 & -252 \\ 70 & -69 & 141 & -421 & 1684 \\ -575 & 575 & -1149 & 3451 & -13801 \\ 3891 & -3891 & 7782 & -23345 & 93365 \\ 1024 & -1024 & 2048 & -6144 & 24572 \end{bmatrix},$$

and request double precision output using `format long`.
a. Execute `eig(A)`, and compute $\|Av_i - \lambda_i v_i\|_2$, $1 \le i \le 5$.

b. Compute `det(A)` and `rank(A)`. Why does the output of both commands conflict with the results of (a)?
c. Approximate the characteristic polynomial of A using `p = poly(A)`. What do you believe is the actual characteristic polynomial, p?
d. The Cayley-Hamilton theorem states that every matrix satisfies its characteristic equation. Compute $p(A)$, where p is the assumed characteristic polynomial from part (c). Is the result close to zero?
e. Explain why this matrix causes so much trouble for eigenvalue solvers.

18.43 This problem takes another approach to the column buckling application in Section 18.1.3.
a. If the beam is hinged at both ends, we obtain the boundary value problem

$$EI\frac{\mathrm{d}^2y}{\mathrm{d}x^2} + Py = 0, \quad y(0) = 0,\ y(L) = 0,$$

whose terms are defined in Section 18.1.3. Approach the problem using finite differences, as we did for the heat equation in Section 12.2. Divide the interval $0 \le x \le L$ into n subintervals of length $h = \frac{L}{n}$ with points of division $x_1 = 0, x_2 = h, x_3 = 2h, \ldots, x_n = L - h$, and $x_{n+1} = L$. Use the central finite difference approximation

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2}(x_i) \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}, \quad 2 \le i \le n$$

along with the boundary conditions, to show that the result is a matrix eigenvalue problem of the form

$$Ay = \lambda y,$$

where A is a symmetric tridiagonal matrix. What is the relationship between λ and the critical loads?
b. For copper, $E = 117 \times 10^9\,\mathrm{N/m^2}$. Assume $I = .0052\,\mathrm{m}^4$ and that $L = 1.52\,\mathrm{m}$. Using $h = 1.52/25$ find the smallest three values of λ, compute the critical loads, and graph (x_i, y_i) for each value of λ on the same axes. Relate the results to the discussion in Section 18.1.3.
Note: Suppose that EV is a 26×3 matrix containing the eigenvectors corresponding to λ_1, λ_2, and λ_3 as well as the zero values at the endpoints. Each row has the format $\begin{bmatrix} 0 & y_2 & y_3 & \ldots & y_{24} & y_{25} & 0 \end{bmatrix}^{\mathrm{T}}$. To create a good looking graph place these statements in your program.

```
minval = min(min(EV));
maxval = max(max(EV));
axis equal;
axis([0 L minval maxval]);
hold on;
graph each displacement using different line styles
add a legend
hold off;
```

18.44 Problem 18.23 introduced the generalized eigenvalue problem. MATLAB can solve these problems using `eig(A,B)` that finds matrices V and D such that

$$AV = BVD.$$

The diagonal of D contains the generalized eigenvectors and the columns of V are the associated eigenvectors.
a. Solve the following generalized eigenvalue problem and verify that $AV = BVD$.

$$\begin{bmatrix} 1 & -7 & 3 \\ 0 & 1 & 5 \\ 3 & 7 & 8 \end{bmatrix} v = \lambda \begin{bmatrix} 1 & 3 & 7 \\ -1 & -8 & 1 \\ 4 & 3 & 1 \end{bmatrix}.$$

b. The basis for the computing the generalized eigenvalues and eigenvectors is the QZ iteration that is analogous to the QR iteration for eigenvalues. Look up the function `qz` in the MATLAB documentation and use it to solve the problem in part (a).