Chapter 17

Implementing the *QR* Decomposition

You should be familiar with

- Rank, orthonormal basis
- Orthogonal matrices
- *QR* decomposition using Gram-Schmidt
- Computation with submatrices

In Chapter 14, we developed the *QR* decomposition of an arbitrary $m \times n$ matrix, $m \geq n$, into a product of an $m \times n$ matrix Q with orthonormal columns and an $n \times n$ upper triangular matrix R such that $A = QR$. We constructed the decomposition using the Gram-Schmidt classical and modified algorithms. Chapter 16 discussed the solution of least-squares problems using the *QR* decomposition, and the decomposition plays a very important role in eigenvalue computation. We must be able to compute it accurately. There are other approaches to the *QR* decomposition that are numerically superior to Gram-Schmidt, *Givens method* and *Householder's Method*, and this chapter presents both methods.

17.1 REVIEW OF THE *QR* DECOMPOSITION USING GRAM-SCHMIDT

We begin this section with a formal statement of the *QR* decomposition theorem from Chapter 14.

Theorem 17.1. *If A is an $m \times n$ matrix, $m \geq n$, with linearly independent columns, then A can be factored as $A = QR$ where Q is an $m \times n$ matrix with orthonormal columns and R is an $n \times n$ upper triangular matrix.*

In Chapter 14, we built the matrices Q and R by using the Gram-Schmidt process, and developed algorithms for both the classical (clqrgrsch) and modified Gram-Schmidt process (modqrgrsch). The modified Gram-Schmidt algorithm avoided possibly costly cancelation errors and is always to be preferred. For the sake of brevity, we will use the acronym MGS to refer to the algorithm. We mentioned in the introduction that the *QR* decomposition using MGS was not as good numerically compared to the Givens or Householder methods. The columns of Q tend to lose orthogonality in proportion to the $\kappa(A)$ [51, 52], and MGS can have problems when the matrix A is ill-conditioned. Example 17.1 clearly demonstrates this.

Example 17.1.

```
>> H = hilb(5);
>> [Q,R] = modqrgrsch(H);
>> norm(eye(5)-Q'*Q)
ans =
   1.1154e-011

>> [Q,R] = clqrgrsch(H);
>> norm(eye(5)-Q'*Q)
ans =
  5.7917e-008

>> H = hilb(15);
>> [Q,R] = modqrgrsch(H);
>> norm(eye(15)-Q'*Q)

ans =
    0.9817   % indicates severe effects from ill-conditioned H
```

The reason the MGS was so poor with the 15×15 Hilbert matrix H is that $\text{Cond}_2(H) = 2.5699 \times 10^{17}$. ■

Remark 17.1. The use of Gram-Schmidt can be improved by a process known as reorthogonalization. See Problem 14.25. This issue of losing orthogonality will become a major concern in Chapters 21 and 22 when we discuss methods for solving systems and computing eigenvalues for large sparse matrices.

Gram-Schmidt requires $2mn^2$ flops to find both Q and R. As we will see, this flop count is better than those for the Givens or Householder methods when both Q and R are desired. However, its possible instability usually dictates the use of other methods. This does not mean that Gram-Schmidt is unimportant. In fact, using it to orthonormalize a set of vectors is a basis for the Arnoldi and Lanczos methods we will study in conjunction with large sparse matrix problems in Chapters 21 and 22.

17.2 GIVENS ROTATIONS

Recall we developed the LU decomposition in Chapter 11 by applying a sequence of elementary matrices to the left side of A. In the resulting decomposition LU, L is the product of the elementary matrices and U is an upper triangular matrix. We will employ this same idea to the transformation of A into the product QR by applying on the left a sequence of orthogonal matrices called Givens matrices that transform A into an upper triangular matrix R. Each Givens matrix product zeros out a matrix element in the creation of R. Here is the idea:

Apply $n-1$ Givens matrices, $J_{i1}, 2 \le i \le n$, on the left of A so that $a_{21}, a_{31}, \dots, a_{n1}$ are zeroed out, and

$$\left(J_{n1}J_{n-1,1}, \dots J_{31}J_{21}\right) A \rightarrow \begin{bmatrix} X & X & \dots & X \\ 0 & X & \cdots & X \\ 0 & X & \dots & X \\ \vdots & \vdots & \vdots & \vdots \\ 0 & X & \dots & X \end{bmatrix}.$$

Now use $n-2$ Givens matrices to zero out the elements at indices $(3,2), (4,2), \dots, (n,2)$, and we have

$$\left(J_{n2}, \dots J_{42}J_{32}\right)\left(J_{n1}J_{n-1,1}, \dots J_{31}J_{21}\right) A \rightarrow \begin{bmatrix} X & X & \dots & X \\ 0 & X & \cdots & X \\ 0 & 0 & \dots & X \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & X \end{bmatrix}.$$

Let J_i be the product of Givens matrices acting on column i. Continue this process $n-3$ more times until we have

$$\left(J_{n-1}J_{n-2}\dots J_2J_1\right) A \rightarrow \begin{bmatrix} X & X & \dots & X & X \\ 0 & X & \dots & X & X \\ 0 & 0 & X & \dots & X \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & X \end{bmatrix} = R,$$

and

$$A = \left(J_{n-1}J_{n-2}\dots J_2J_1\right)^{\mathrm{T}} R = QR.$$

Q is a product of orthogonal matrices, and so it is orthogonal.

We start with the definition of the general $n \times n$ *Givens matrix*.

Definition 17.1. A matrix of the form (Figure 17.1) is a Givens matrix. The value c is on the diagonal at indices (i,i) and (j,j), $i < j$. The value $-s$ is at index (j,i), and s is at index (i,j). The remaining diagonal entries are 1, and all off-diagonal elements at indices other than (j,i) and (i,j) are zero.

We want a Givens matrix to be orthogonal ($J(i,j,c,s)^{\mathrm{T}} J(i,j,c,s) = I$), and clearly the columns in the definition are orthogonal; however, each column must have unit length. This requires that $c^2 + s^2 = 1$. For clarity, Example 17.2 illustrates the form of $J(i,j,c,s)^{\mathrm{T}} J(i,j,c,s)$ for a 3×3 matrix.

Example 17.2. Let $J(1,3,c,s) = \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix}$. Then,

$$J(i,j,c,s) = \begin{array}{c|cccccccc|cl}
 & & & & & & i\text{th} & i\text{th} & & \text{Columns} & \\
 & & & & & & \downarrow & \downarrow & & & \\
\hline
 & 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 & & \\
 & 0 & 1 & 0 & \dots & \dots & \dots & \dots & 0 & & \\
 & \vdots & \vdots & \vdots & & & & & & & \\
 & 0 & 0 & 0 & \dots & c & s & \dots & 0 & \leftarrow & i\text{th} \\
 & \vdots & \vdots & \vdots & & & & & & & \quad\text{Rows} \\
 & 0 & 0 & 0 & \cdots & -s & c & \dots & 0 & \leftarrow & j\text{th} \\
 & \vdots & \vdots & \vdots & & & & & & & \\
 & 0 & 0 & 0 & \dots & \dots & 0 & \dots & 1 & & \\
\hline
\end{array}$$

FIGURE 17.1 Givens matrix.

$$J(1,3,c,s)^{\mathrm{T}} J(1,3,c,s) = \begin{bmatrix} c & 0 & -s \\ 0 & 1 & 0 \\ s & 0 & c \end{bmatrix} \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix} = \begin{bmatrix} c^2+s^2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & c^2+s^2 \end{bmatrix},$$

so if $c^2 + s^2 = 1$, $J(1,3,c,s)$ is orthogonal. ■

In the $n \times n$ case,

$$\begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ & \ddots & & & & & \\ & & c & & -s & & \\ & & \vdots & \ddots & \vdots & & \\ & & s & \dots & c & \vdots & \\ & & & & & \ddots & \\ 0 & 0 & \dots & \dots & \dots & \dots & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ & \ddots & & & & & \\ & & c & & s & & \\ & & \vdots & \ddots & \vdots & & \\ & & -s & \dots & c & \vdots & \\ & & & & & \ddots & \\ 0 & 0 & \dots & \dots & \dots & \dots & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ & \ddots & & & & & \\ & & c^2+s^2 & & 0 & & \\ & & \vdots & \ddots & \vdots & & \\ & & 0 & \dots & c^2+s^2 & \vdots & \\ & & & & & \ddots & \\ 0 & 0 & \dots & \dots & \dots & \dots & 1 \end{bmatrix}.$$

Require that $c^2 + s^2 = 1$ and recall the identity $\sin^2\theta + \cos^2\theta = 1$. For instance, the matrix

$$J(4,6,c,s) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\left(\frac{\pi}{6}\right) & 0 & \sin\left(\frac{\pi}{6}\right) \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\sin\left(\frac{\pi}{6}\right) & 0 & \cos\left(\frac{\pi}{6}\right) \end{bmatrix},$$

where $i = 4, j = 6, c = \cos\left(\frac{\pi}{6}\right), s = \sin\left(\frac{\pi}{6}\right)$ is orthogonal since $\cos^2\left(\frac{\pi}{6}\right) + \sin^2\left(\frac{\pi}{6}\right) = 1$. In general, we can choose any angle θ, let $c = cos(\theta)$, $s = sin(\theta)$ and always obtain a Givens matrix. When we make such a choice, we will use the notation $J(i,j,\theta)$. Such a matrix is actually a rotation matrix that rotates a pair of coordinate axes through an angle θ in the (i,j) plane, so it is also known as a *Givens* rotation. In $\mathbb{R}^2$, the Givens matrix $J(1,2,\theta)$ is $\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$, and you will recognize this as a rotation matrix, a topic we discussed in Chapter 1. It rotates a vector clockwise through the angle θ.

Remark 17.2. Operations with Givens matrices can be done implicitly; in other words, it is not necessary to actually build the Givens matrix. However, to reinforce understanding of exactly how these matrices operate, we will explicitly build them until Section 17.4.

17.2.1 Zeroing a Particular Entry in a Vector

The Givens *QR* decomposition algorithm relies on being able to place zeros at specified locations in a matrix. Let $x = \begin{bmatrix} x_1 & x_2 & \dots & x_{n-1} & x_n \end{bmatrix}^{\mathrm{T}}$, and form the product $J(i,j,c,s)\,x$.

$$J(i,j,c,s)\,x = \begin{bmatrix} 1 & 0 & 0 & \ldots & \ldots & & \ldots & \ldots & 0 \\ 0 & 1 & 0 & \ldots & \ldots & & \ldots & \ldots & 0 \\ \vdots & \vdots & \vdots & & & & & & \vdots \\ 0 & 0 & 0 & \ldots & c & \ldots & s & \cdots & 0 \\ \vdots & \vdots & \vdots & & & & & & \vdots \\ 0 & 0 & 0 & \ldots & -s & \ldots & c & \cdots & 0 \\ \vdots & \vdots & \vdots & & & & & \ldots & \vdots \\ 0 & 0 & 0 & \ldots & \ldots & \ldots & 0 & \ldots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_j \\ \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ cx_i + sx_j \\ \vdots \\ -sx_i + cx_j \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}$$

Note that the product changes only the components i and j of x.

Example 17.3. Let $J(2,4,c,s) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & 0 & s \\ 0 & 0 & 1 & 0 \\ 0 & -s & 0 & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1 \\ cx_2 + sx_4 \\ x_3 \\ -sx_2 + cx_4 \end{bmatrix}$ ■

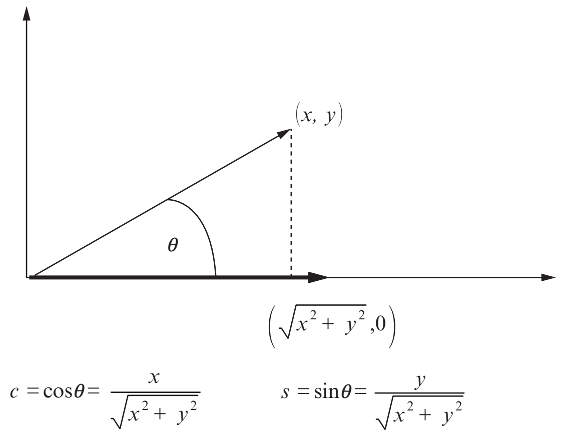
Assume we have a vector $x \in \mathbb{R}^n$ and want to zero out $x_j, j > 1$, as illustrated in Equation 17.1.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{bmatrix}, \quad J(i,j,c,s)\,x = \begin{bmatrix} x_1 \\ X \\ \vdots \\ 0 \\ \vdots \\ X \end{bmatrix} \tag{17.1}$$

We must determine the values of i, j, c, and s to use, and then form the Givens rotation by creating the identity matrix, inserting c in locations (i,i) and (j,j), $-s$ in location (j,i), and s into position (i,j). Since multiplication by a Givens matrix only affects components i and j of the vector, we can reduce finding c and s to a two-dimensional problem. Create a Givens matrix $\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$ that rotates vector $\begin{bmatrix} x \\ y \end{bmatrix}$ in $\mathbb{R}^2$ onto the x-axis. In that way, we zero out the y-coordinate. From Figure 17.2, we have $\cos\theta = \frac{x}{\sqrt{x^2+y^2}}$, $\sin\theta = \frac{y}{\sqrt{x^2+y^2}}$, so

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} \\ -\frac{y}{\sqrt{x^2+y^2}} & \frac{x}{\sqrt{x^2+y^2}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2} \\ 0 \end{bmatrix}.$$

By looking at Figure 17.2, this is exactly what we should get. The vector $\begin{bmatrix} x \\ y \end{bmatrix}$ after rotation will have an x-coordinate of $\sqrt{x^2+y^2}$ and a y-coordinate of 0.



(x, y)
θ
$\left(\sqrt{x^2+y^2}, 0\right)$
$c = \cos\theta = \frac{x}{\sqrt{x^2+y^2}}$
$s = \sin\theta = \frac{y}{\sqrt{x^2+y^2}}$

FIGURE 17.2 Givens rotation.

Summary

To zero out entry j of vector x, choose index $i < j$ and compute the values

$$c = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \quad s = \frac{x_j}{\sqrt{x_i^2 + x_j^2}}.$$

Let $J(i,j,c,s) = I$, followed by

$$\begin{aligned} J(i,i,c,s) &= J(j,j,c,s) = c \\ J(j,i,c,s) &= -s \\ J(i,j,c,s) &= s \end{aligned}$$

and compute $J(i,j,c,s)\,x$.

Example 17.4. Suppose we want to zero out component 2 of the 3×1 vector $x = \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix}$.

Choose $i = 1, j = 2$ and compute $c = \frac{-1}{\sqrt{5}}$, $s = \frac{2}{\sqrt{5}}$. The Givens rotation matrix is

$$J(1,2,c,s) = \begin{bmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{-1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 \\ -\frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Now form the product $J(1,2,c,s)\,x$ to obtain

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

Let us look at a more complex example.

We want to zero out component 3 of the 4×1 vector $x = \begin{bmatrix} 3 & -7 & -1 & 5 \end{bmatrix}^{\mathrm{T}}$. Choose $i = 1, j = 3$, and then

$$c = \frac{3}{\sqrt{10}}, \quad s = \frac{-1}{\sqrt{10}},$$

and form

$$J(1,3,c,s)\,x = \begin{bmatrix} \frac{3}{\sqrt{10}} & 0 & \frac{-1}{\sqrt{10}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{10}} & 0 & \frac{3}{\sqrt{10}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ -7 \\ -1 \\ 5 \end{bmatrix} = \begin{bmatrix} \sqrt{10} \\ -7 \\ 0 \\ 5 \end{bmatrix}$$

It is necessary to choose $i < j$, and any index $i < j$ can be used. Choose $i = 2$ and $j = 3$, form $J(2,3,c,s)$, and verify that $J(2,3,c,s)\,x$ zeros out x_3. ■

17.3 CREATING A SEQUENCE OF ZEROS IN A VECTOR USING GIVENS ROTATIONS

Given a vector $x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^{\mathrm{T}}$, we can use a product of Givens matrices to zero out the $n-i$ elements of x below entry x_i. To zero out x_{i+1}, compute $J(i,i+1,c_{i+1},s_{i+1})\,x = \overline{x_{i+1}}$. To zero out x_{i+2}, compute $J(i,i+2,c_{i+2},s_{i+2})\,\overline{x_{i+1}} = \overline{x_{i+2}}$, and continue the process until computing $J(i,n,c_n,s_n)\,\overline{x_{n-1}} = \overline{x_n}$. In summary, the product

$$J(i,n,c_n,s_n)\dots J(i,i+2,c_{i+2},s_{i+2})\,J(i,i+1,c_{i+1},s_{i+1})\,x$$

transforms x into a vector of the form $\begin{bmatrix} x_1 & x_2 & \dots & x_{i-1} & * & 0 & \dots & 0 \end{bmatrix}^{\mathrm{T}}$.

Example 17.5. Let $x = \begin{bmatrix} 5 \\ -1 \\ 3 \end{bmatrix}$ and zero out the second and third components of x using Givens rotations.

$$\overline{x_2} = J(1,2,c_2,s_2)\,x = \begin{bmatrix} \frac{5}{\sqrt{26}} & -\frac{1}{\sqrt{26}} & 0 \\ \frac{1}{\sqrt{26}} & \frac{5}{\sqrt{26}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ -1 \\ 3 \end{bmatrix} = \begin{bmatrix} \sqrt{26} \\ 0 \\ 3 \end{bmatrix},$$

$$\overline{x_3} = J(1,3,c_3,s_3)\,\overline{x_2} = \begin{bmatrix} \frac{\sqrt{26}}{\sqrt{35}} & 0 & \frac{3}{\sqrt{35}} \\ 0 & 1 & 0 \\ -\frac{3}{\sqrt{35}} & 0 & \frac{\sqrt{26}}{\sqrt{35}} \end{bmatrix} \begin{bmatrix} \sqrt{26} \\ 0 \\ 3 \end{bmatrix} = \begin{bmatrix} \sqrt{35} \\ 0 \\ 0 \end{bmatrix}$$ ■

17.4 PRODUCT OF A GIVENS MATRIX WITH A GENERAL MATRIX

If A is an $m \times n$ matrix, a huge advantage when dealing with Givens matrices is the fact that you can compute a product $J(i,j,c,s)\,A$ without ever constructing $J(i,j,c,s)$.

Example 17.6. If $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$, then

$$J(1,3,c,s)\,A = \begin{bmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ -s & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix} = \begin{bmatrix} ca_{11}+sa_{31} & ca_{12}+sa_{32} & ca_{13}+sa_{33} \\ a_{21} & a_{22} & a_{23} \\ ca_{31}-sa_{11} & ca_{32}-sa_{12} & ca_{33}-sa_{13} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

The product only affects rows 1 and 3. ■

In the general case where $J(i,j,c,s)$ is $m \times m$ and A is $m \times n$ the product $J(i,j,c,s)\,A$ only affects rows i and j. Using Example 17.6 as a guide, convince yourself that rows i and j, $i < j$, have the following form:

$$\text{Row } i: \quad ca_{i1} + sa_{j1}, ca_{i2} + sa_{j2}, \ldots, ca_{in} + sa_{jn}, \tag{17.2}$$

$$\text{Row } j: \quad ca_{j1} - sa_{i1}, ca_{j2} - sa_{i2}, \ldots, ca_{jn} - sa_{in}. \tag{17.3}$$

This makes the product $J(i,j,c,s)\,A$ very efficient to compute, requiring only $6n$ flops. Just change rows i and j according to Equations 17.2 and 17.3.

Algorithm 17.1 Product of a Givens Matrix J with a General Matrix A

```
function GIVENSMUL(A,i,j,c,s)
  % Multiplication by a Givens matrix.
  % Input: An $m \times n$ matrix A, matrix indices i, j,
  % and Givens parameters c, s.
  % Output: The $m \times n$ matrix J(i,j,c,s)A.
  a = A(i,:)
  b = A(j,:)
  A(i,:)=ca+sb
  A(:,j)=-sa+cb
end function
```

NLALIB: The function `givensmul` implements Algorithm 17.1.

17.5 ZEROING-OUT COLUMN ENTRIES IN A MATRIX USING GIVENS ROTATIONS

We know how to zero out all the entries of a vector x below an entry x_i, and we know how to very simply form the product of a Givens matrix with another matrix. Now we will learn how to zero out all the entries in column i below a matrix entry $A(i, i)$. This is what we must do to create an upper triangular matrix.

First, consider the problem of computing the product $J(i,j,c,s)A, j > i$, so that the result will have a zero at index (j, i) and only rows i and j of A will change. Think of column i, $\begin{bmatrix} a_{1i} & a_{2i} & \dots & a_{ii} & \dots & a_{ji} & \dots & a_{mi} \end{bmatrix}^{\mathrm{T}}$, as a vector and find c and s using a_{ii} and a_{ji} so the product will zero out a_{ji}. Using `givensmul`, implicitly form a Givens matrix with c at indices (i, i) and (j, j), $-s$ at index (j, i) and s at index (i, j) and compute $J(i, j, c, s) A$ using givensmul. Entries in both rows i and j will change. This makes no difference, since we are only interested in zeroing out a_{ji}.

$$\underset{J(i,j,c,s)^{m\times m}}{\begin{array}{c|cccccc|} & & i & & j & & \\ \hline & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ & \vdots & \ddots & \cdots & & & \vdots \\ i & & c & \cdots & s & & 0 \\ & \vdots & & \ddots & & & \\ j & & -s & & c & & 0 \\ & \vdots & & & & & \vdots \\ & 0 & & & & & 1 \\ \hline \end{array}} \quad \underset{A^{m\times n}}{\begin{array}{c|cccccc|} & & i & & j & & \\ \hline & a_{11} & a_{1i} & \cdots & \cdots & \cdots & a_{1n} \\ & \vdots & \ddots\ \vdots & \cdots & \cdots & \cdots & \vdots \\ i & a_{i1} & a_{ii} & & a_{ij} & \cdots & a_{in} \\ & \vdots & \vdots & \ddots & & & \\ j & a_{j1} & a_{ji} & & a_{jj} & \cdots & a_{jn} \\ & \vdots & \vdots & & & \ddots & \\ & a_{m1} & a_{mi} & & & & a_{mn} \\ \hline \end{array}} = \begin{array}{c|ccccccc|} & & & i & & j & & \\ \hline & a_{11} & \cdots & a_{i1} & \cdots & a_{1j} & \cdots & a_{1n} \\ & \vdots & & & & & & \\ i & \overline{a_{i1}} & \cdots & X & \cdots & X & \cdots & \overline{a_{in}} \\ & \vdots & & \vdots & \ddots & \vdots & & \vdots \\ j & \overline{a_{j1}} & & 0 & \cdots & X & \cdots & \overline{a_{jn}} \\ & \vdots & & \vdots & & & & \vdots \\ & a_{m1} & \cdots & a_{mi} & \cdots & a_{mj} & \cdots & a_{mn} \\ \hline \end{array}$$

To zero out every element in column i below a_{ii}, compute the sequence

$$\overline{A_{i+1}} = J(i, i+1, c_{i+1}, s_{i+1}) A, \overline{A_{i+2}} = J(i, i+2, c_{i+2}, s_{i+2}) \overline{A_{i+1}}, \dots, \overline{A_m} = J(i, m, c_m, d_m) \overline{A_{m-1}}.$$

Example 17.7. Let $A = \begin{bmatrix} 1 & 3 & -6 & -1 \\ 4 & 8 & 7 & 3 \\ 2 & 3 & 4 & 5 \\ -9 & 6 & 3 & 2 \end{bmatrix}$

Zero out all entries in column 1 below a_{11}. To form $\overline{A_2}$, implicitly multiply by

$$J(1, 2, c_2, s_2) = \begin{bmatrix} 0.2425 & 0.9701 & 0 & 0 \\ -0.9701 & 0.2425 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$\begin{aligned} \overline{A_2} &= J(1, 2, c_2, s_2) A \\ &= \begin{bmatrix} 0.2425(1) + 0.9701(4) & 0.2425(3) + 0.9701(8) & 0.2425(-6) + 0.9701(7) & 0.2425(-1) + 0.9701(3) \\ -0.9701(1) + 0.2425(4) & -0.9701(3) + 0.2425(8) & (-0.9701)(-6) + 0.2425(7) & (-0.9701)(-1) + 0.2425(3) \\ 2 & 3 & 4 & 5 \\ -9 & 6 & 3 & 2 \end{bmatrix} \\ &= \begin{bmatrix} 4.1231 & 8.4887 & 5.3358 & 2.6679 \\ 0 & -0.9701 & 7.5186 & 1.6977 \\ 2 & 3 & 4 & 5 \\ -9 & 6 & 3 & 2 \end{bmatrix} \end{aligned}$$

Implicitly multiply by

$$J(1, 3, c_3, c_4) = \begin{bmatrix} 0.8997 & 0 & 0.4364 & 0 \\ 0 & 1 & 0 & 0 \\ -0.4364 & 0 & 0.8997 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned}\overline{A_3} &= J(1,3,c_3,s_3)\,\overline{A_2}\\ &= \begin{bmatrix} 4.5826 & 8.9469 & 6.5465 & 4.5826\\ 0 & -0.9701 & 7.5186 & 1.6977\\ 0 & -1.0056 & 1.2702 & 3.3343\\ -9 & 6 & 3 & 2\end{bmatrix}\end{aligned}$$

Implicitly multiply by

$$J(1,4,c_4,s_4) = \begin{bmatrix} -0.4537 & 0 & 0 & 0.8911\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ -0.8911 & 0 & 0 & -0.4537\end{bmatrix}$$

$$\begin{aligned}\overline{A_4} &= J(1,4,c_4,s_4)\,\overline{A_3}\\ &= \begin{bmatrix} -10.0995 & 1.2872 & -0.2970 & -0.2970\\ 0 & -0.9701 & 7.5186 & 1.6977\\ 0 & -1.0056 & 1.2702 & 3.3343\\ 0 & -10.6954 & -7.1951 & -4.9912\end{bmatrix}\end{aligned}$$

Form

$$P = J(1,4,c_4,s_4)\,J(1,3,c_3,s_3)\,J(1,2,c_2,s_2) = \begin{bmatrix} -0.0990 & -0.3961 & -0.1980 & 0.8911\\ -0.9701 & 0.2425 & 0 & 0\\ -0.1059 & -0.4234 & 0.8997 & 0\\ -0.1945 & -0.7778 & -0.3889 & -0.4537\end{bmatrix},$$

and $PA = \overline{A_4}$. ■

The matrix P in Example 17.7 is the product of orthogonal matrices, and so P is orthogonal, as proved in Lemma 17.1.

Lemma 17.1. *If matrix $P = P_kP_{k-1}\dots P_2P_1$, where each matrix P_i is orthogonal, so is P.*

Proof. $P^{-1} = P_1^{-1}P_2^{-1}\dots P_{k-1}^{-1}P_k^{-1} = P_1^{\mathrm{T}}P_2^{\mathrm{T}}\dots P_{k-1}^{\mathrm{T}}P_k^{\mathrm{T}} = (P_kP_{k-1}\dots P_2P_1)^{\mathrm{T}}$, so $P^{-1} = P^{\mathrm{T}}$, and P is orthogonal. □

Note that we can continue with $\overline{A_4}$ in Example 17.8 and use Givens rotations to zero out the elements in column 2 below $\overline{A_4}(2,2)$. By zeroing out the element at index $(4,3)$, we will have an upper triangular matrix.

It is critical that c and s be computed as accurately as possible when implicitly multiplying by $J(i,j,c,s)$. As it turns out, this is not a simple matter.

17.6 ACCURATE COMPUTATION OF THE GIVENS PARAMETERS

The computation of c and s can have problems with overflow or underflow. We saw in Sections 8.4.1 and 8.4.2 that in order to minimize computation errors, it may be necessary to rearrange the way we perform a computation. The following algorithm provides improvement in overall accuracy [19, pp. 195-196] by cleverly employing the normalization procedure described in Section 8.4.1. The algorithm takes care of the case where $x_j = 0$ by assigning $c = 1$ and $s = 0$ so that the Givens rotation is the identity matrix. The signs of c and s may be different from those obtained from $c = \frac{x_i}{\sqrt{x_i^2+x_j^2}}$, $s = \frac{x_j}{\sqrt{x_i^2+x_j^2}}$, but that does not change the rotation's effect.

NLALIB: The function `givensparms` implements Algorithm 17.2.

Algorithm 17.2 requires five flops and a square root.

Algorithm 17.2 Computing the Givens Parameters

```
function GIVENSPARMS($x_i$, $x_j$)
  % Input: value $x_i$ at index i and $x_j$ at index j>i of a vector x.
  % Output: the Givens parameters for x.

  if $x_j = 0$ then
    c=1
    s=0
  else if $|x_j| > |x_i|$ then
    t= $x_i/x_j$
    $s = \frac{1}{\sqrt{1+t^2}}$
    c=st
  else
    t= $x_j/x_i$
    $c = \frac{1}{\sqrt{1+t^2}}$
    s=ct;
  end if
    return [ c, s ]
end function
```

17.7 THE GIVENS ALGORITHM FOR THE *QR* DECOMPOSITION

All the necessary tools are now in place to construct Q and R using Givens rotations. First, we formalize our understanding of the term *upper triangular matrix* A.

Definition 17.2. An $m \times n$ matrix $A = [a_{ij}]$ is upper triangular if $a_{ij} = 0$ for $i > j$. Another way of putting it is that all entries below a_{ii} are 0.

Example 17.8. The matrices A and B are upper triangular.

$$A = \begin{bmatrix} 1 & -1 & 7 & 12 & 1 & 3 \\ 0 & 2 & 8 & 2 & 4 & 1 \\ 0 & 0 & 3 & -9 & 10 & 6 \\ 0 & 0 & 0 & 4 & 2 & 1 \\ 0 & 0 & 0 & 0 & 5 & 18 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 2 & 6 & -1 \\ 0 & 4 & -2 & 8 \\ 0 & 0 & -1 & -9 \\ 0 & 0 & 0 & -4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$ ■

The decomposition algorithm uses Givens rotations to zero out elements below the diagonal element until arriving at an upper triangular matrix. The question is “How many steps, k, should we execute?” This depends on the dimension of A. Clearly, if A is an $n \times n$ matrix, we need to execute the process $k = n - 1$ times. Now consider two examples where $m \neq n$.

m > n: Look at a 5×3 matrix $A_1 = \begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \\ X & X & X \\ X & X & X \end{bmatrix}$. Here are the transformations that occur to A_1.

$$A_1 = \begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \\ X & X & X \\ X & X & X \end{bmatrix} \Longrightarrow \begin{bmatrix} X & X & X \\ 0 & X & X \\ 0 & X & X \\ 0 & X & X \\ 0 & X & X \end{bmatrix} \Longrightarrow \begin{bmatrix} X & X & X \\ 0 & X & X \\ 0 & 0 & X \\ 0 & 0 & X \\ 0 & 0 & X \end{bmatrix} \Longrightarrow \begin{bmatrix} X & X & X \\ 0 & X & X \\ 0 & 0 & X \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

We executed three steps until we could not continue to zero out elements below the diagonal, so $k = 3 = n$.

m < n: Look at a 4 × 6 matrix

$$A_2 = \begin{bmatrix} X & X & X & X & X & X \\ X & X & X & X & X & X \\ X & X & X & X & X & X \\ X & X & X & X & X & X \end{bmatrix} \Longrightarrow \begin{bmatrix} X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & X & X & X & X & X \end{bmatrix} \Longrightarrow \begin{bmatrix} X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & 0 & X & X & X & X \\ 0 & 0 & X & X & X & X \end{bmatrix}$$

$$\Longrightarrow \begin{bmatrix} X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & 0 & X & X & X & X \\ 0 & 0 & 0 & X & X & X \end{bmatrix}.$$

We executed three steps until we could not continue to zero out elements below the diagonal, so $k = 3 = m - 1$.

From these examples, we see that the sequence of steps in the Givens QR decomposition algorithm is $k = \min(m-1, n)$. We are now ready to provide the algorithm for the Givens QR decomposition. To compute Q, start with $Q = I$. As the algorithm progresses, build Q by successive uses of givensmul. Note also that when the rotations have produced R, $QA = R$, and so $A = Q^{\mathrm{T}}R$. Thus, the Q returned is the transpose of the one built during the construction of R.

Algorithm 17.3 Givens *QR* Decomposition

```
function GIVENSQR(A)
   % Computes the QR decomposition of A
   % Input: $m \times n$ matrix A
   % Output: $m \times n$ orthogonal matrix Q
   % and $m \times n$ upper triangular matrix R
   $Q = I$

   for i=1:min(m-1,n) do
      for j=i+1:m do
         [c s]=givensparms($a_{ii}$, $a_{ji}$)
         A=givensmul(A,i,j,c,s)
         Q=givensmul(Q,i,j,c,s)
      end for
   end for

   R=A;
   Q= $Q^T$
end function
```

NLALIB: The function `givensqr` implements Algorithm 17.3.

The construction we developed that is realized in Algorithm 17.3 proves the following theorem.

Theorem 17.2 (*QR* decomposition). *If A is an $m \times n$ matrix, it can be expressed in the form $A = QR$, where Q is an $m \times m$ orthogonal matrix and R is an $m \times n$ upper triangular matrix.*

Now let us do Example 17.1 again, this time using the Givens QR decomposition. Compare the results to those of produced by the modified Gram-Schmidt algorithm.

Example 17.9.
```
>> H = hilb(5);
>> [Q R] = givensqr(H);
>> norm(eye(5) - Q'*Q)

ans =
  5.6595e-016

>> H = hilb(15);
>> [Q R] = givensqr(H);
>> norm(eye(15) - Q'*Q)

ans =

  1.0601e-015
```

The columns of the matrix Q in the Givens algorithm do not lose orthogonality like the columns of Q do when using MGS. ■

17.7.1 The Reduced *QR* Decomposition

Gram-Schmidt produces the reduced *QR* decomposition. Without knowledge of Gram-Schmidt, we can prove the existence of this decomposition using Theorem 17.2 and, at the same time, determine how to compute it using Givens rotations.

If A is has full rank, $m \geq n$, let $A^{m\times n} = Q^{m\times m}R^{m\times n}$ be the full *QR* decomposition of A. Let the columns of Q be denoted by q_i, $1 \leq i \leq m$ so that we can write

$$Q = \left[\underbrace{q_1 \;\; q_2 \;\; \dots \;\; q_n}\;\; \underbrace{q_{n+1} \;\; q_{n+2} \;\; \dots \;\; q_m} \right].$$

Let q_i, $1 \leq i \leq n$ form the matrix Q_1, and q_i, $n+1 \leq i \leq m$ form Q_2 so that

$$Q = \left[\; Q_1^{m\times n} \;\; Q_2^{m\times(m-n)} \; \right]$$

The upper triangular matrix

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1,n} \\ 0 & r_{22} & \dots & r_{2,n} \\ \vdots & & \ddots & \ddots \\ 0 & & & r_{n,n} \\ 0 & & & 0 \\ \vdots & & & \vdots \\ 0 & & & 0 \end{bmatrix}$$

can be written as

$$R = \begin{bmatrix} R_1^{n\times n} \\ 0^{(m-n)\times n} \end{bmatrix},$$

and so

$$A = \left[\; Q_1^{m\times n} \;\; Q_2^{m\times(m-n)} \; \right] \begin{bmatrix} R_1^{n\times n} \\ 0^{(m-n)\times n} \end{bmatrix}.$$

This implies that $A = Q_1R_1$.

It is actually not necessary that A have full rank. We will still have $A = Q_1R_1$, and the matrix R_1 has the form

$$\begin{bmatrix} r_{11} & r_{12} & \dots & & \dots & & r_{1n} \\ 0 & r_{22} & \dots & & \dots & & r_{2n} \\ 0 & 0 & \ddots & & & & \vdots \\ \vdots & \vdots & 0 & r_{kk} & r_{k,k+1} & \dots & r_{kn} \\ \vdots & \vdots & \vdots & \vdots & 0 & & \\ & & & & \vdots & & \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix}.$$

Remark 17.3. If m is much larger than n, the reduced *QR* decomposition saves significant memory.

17.7.2 Efficiency

We will assume $m \geq n$. If $m > n$, the outer loop executes n times, and if $m = n$, it executes $n - 1$ times. Since $m > n$ is the more general case, assume the outer loop executes n times. Each call to givensparms requires five flops, and the call to givensmul(A,i,j,c,s) requires $6n$ flops. The matrix Q is of size $m \times m$, so givensmul(Q,i,j,c,s) executes $6m$ flops. The total flop count is

$$(5 + 6n + 6m) \sum_{i=1}^{n} (m - i)\,. \tag{17.4}$$

After some work, Equation 17.4 expands to

$$3n\left(mn + 2m^2 - n^2\right) + \text{lower order terms.}$$

Disregarding the lower-order terms, we have

$$\text{Flop count} \cong 3n\left(mn + 2m^2 - n^2\right). \tag{17.5}$$

If $m = n$, Equation 17.5 gives $6n^3$ flops, so in this case the algorithm is $O\left(n^3\right)$.

The Givens QR algorithm is stable [16, pp. 365-368].

17.8 HOUSEHOLDER REFLECTIONS

The use of Householder reflections is an alternative to Givens rotations for computing the QR decomposition of a matrix. A Givens rotation zeros out one element at a time, and a sequence of rotations is required to transform the matrix into upper triangular form. The Householder algorithm for the QR decomposition requires about two-thirds the flop count of the Givens algorithm because it zeros out all the elements under a diagonal element a_{ii} in one multiplication. However, it should be noted that Givens rotations lend themselves to parallelization. Also, if a matrix has a particular structure, it may be efficient to zero out one element at a time, as we will see in Chapters 18, 19, and 21. Givens rotations are perfect for that purpose.

A Householder reflection is an $n \times n$ orthogonal matrix formed from a vector in $\mathbb{R}^n$.

Definition 17.3. A *Householder reflection* (or Householder transformation) H_u is a transformation that takes a vector u and reflects it about a plane in $\mathbb{R}^n$. The transformation has the form

$$H_u = I - \frac{2uu^{\mathrm{T}}}{u^{\mathrm{T}}u}, \quad u \neq 0.$$

Clearly, H_u is an $n \times n$ matrix, since uu^{T} is a matrix of dimension $n \times n$. The Householder transformation has a geometric interpretation (Figure 17.3).

A Householder reflection applied to u gives $-u$. As a result, $H_u\,(cu) = -cu$, where c is a scalar. For all other vectors w, let $v = w - \mathrm{proj}_u\,(w)$. We know from our work with Gram-Schmidt that v is orthogonal to u. The vector w is a linear combination of the vectors u and v (Figure 17.4).

$$\begin{aligned} w &= c_1 u + c_2 v \\ c_1 &= \langle w,\, v\rangle \,/\, \|v\|_2^2 \\ c_2 &= \langle w,\, v\rangle \,/\, \|v\|_2^2 \end{aligned}$$

$H_u\,(w)$ is a reflection of w in the plane through 0 perpendicular to u. We will show that these claims are true.

$$H_u\,(u) = \left(I - \frac{2uu^{\mathrm{T}}}{u^{\mathrm{T}}u}\right) u = u - \frac{2uu^{\mathrm{T}}u}{u^{\mathrm{T}}u} = u - \frac{2u\,\|u\|_2^2}{\|u\|_2^2} = u - 2u = -u.$$

Now,

$$\begin{aligned} H_u\,(c_1 u + c_2 v) &= \left(I - 2\frac{uu^{\mathrm{T}}}{u^{\mathrm{T}}u}\right)(c_1 u + c_2 v) \\ &= c_1 u + c_2 v - 2\left(\frac{uu^{\mathrm{T}}}{u^{\mathrm{T}}u}\right)(c_1 u + c_2 v) = c_1 u + c_2 v - 2c_1 u - 2c_2\left(\frac{uu^{\mathrm{T}}}{u^{\mathrm{T}}u}\right) v \end{aligned}$$

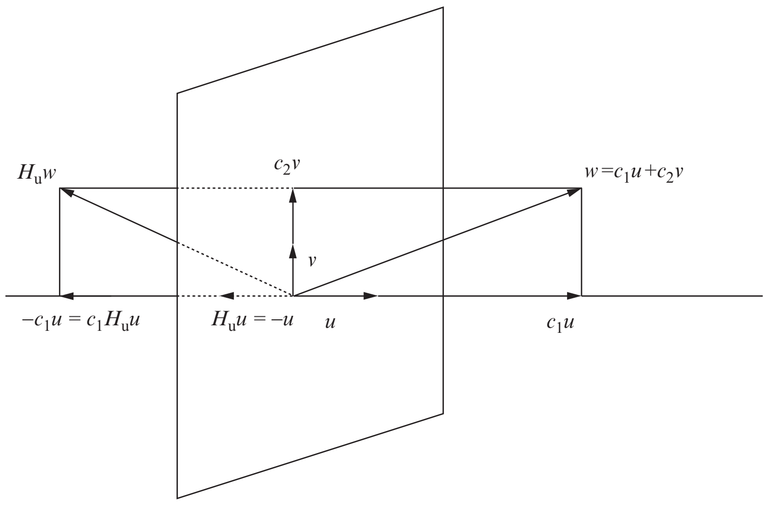


FIGURE 17.3 Householder reflection.

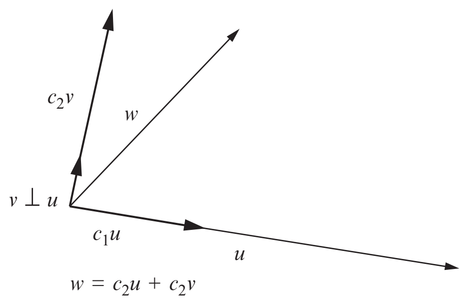


FIGURE 17.4 Linear combination associated with Householder reflection.

$$= -c_1 u + c_2 v - 2c_2 \left(\frac{u}{u^{\mathrm{T}} u}\right) \langle u, v\rangle = -c_1 u + c_2 v.$$

The Householder transformation also has a number of other interesting and useful properties.

Theorem 17.3. *Let H_u be a Householder reflection with vector $u \in \mathbb{R}^n$. Then*

1. *H_u is symmetric.*
2. *H_u is orthogonal.*
3. *$H_u^2 = I$*
4. *$H_u v = v$ if $\langle v, u\rangle = 0$.*

Proof. 1. $H_u^{\mathrm{T}} = \left(I - \frac{2uu^{\mathrm{T}}}{u^{\mathrm{T}}u}\right)^{\mathrm{T}} = I - \left(\frac{2uu^{\mathrm{T}}}{u^{\mathrm{T}}u}\right)^{\mathrm{T}} = I - \frac{2uu^{\mathrm{T}}}{u^{\mathrm{T}}u} = H_u.$

2. This is the most important property of H_u. Let $\beta = \frac{2}{u^{\mathrm{T}}u} = \frac{2}{\|u\|_2^2}$. Now, $H_u^{\mathrm{T}} H_u = H_u H_u = \left(I - \beta uu^{\mathrm{T}}\right)\left(I - \beta uu^{\mathrm{T}}\right)$, since H_u is symmetric.

$$\left(I - \beta uu^{\mathrm{T}}\right)\left(I - \beta uu^{\mathrm{T}}\right) = I - 2\beta uu^{\mathrm{T}} + \beta^2\left(uu^{\mathrm{T}}uu^{\mathrm{T}}\right) = I - 2\beta uu^{\mathrm{T}} + \beta^2 \|u\|_2^2 uu^{\mathrm{T}} = I - 2\left(\frac{2}{\|u\|_2^2}\right) uu^{\mathrm{T}}$$
$$+ \left(\frac{2}{\|u\|_2^2}\right)^2 \|u\|_2^2 uu^{\mathrm{T}} = I,$$

so H_u is orthogonal.

The proofs of properties 3 and 4 are left to the problems. □

Example 17.10. Let $u = \begin{bmatrix} 3 & -1 & 2 \end{bmatrix}^{\mathrm{T}}$.

$$\begin{aligned} H_u &= I - 2\frac{uu^{\mathrm{T}}}{u^{\mathrm{T}}u} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{2}{3^2 + (-1)^2 + 2^2} \begin{bmatrix} 3 \\ -1 \\ 2 \end{bmatrix} \begin{bmatrix} 3 & -1 & 2 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{1}{7} \begin{bmatrix} 9 & -3 & 6 \\ -3 & 1 & -2 \\ 6 & -2 & 4 \end{bmatrix} = \begin{bmatrix} -0.2857 & 0.4286 & -0.8571 \\ 0.4286 & 0.8571 & 0.2857 \\ -0.8571 & 0.2857 & 0.4286 \end{bmatrix} \end{aligned}$$

Note that H_u is symmetric. Other properties of H_u are easier to illustrate using MATLAB.

```
>> u = [3 -1 2]';
>> Hu = eye(3) - 2*(u*u')/(u'*u);
Hu*u % Hu*u = -u

ans =
   -3.0000
    1.0000
   -2.0000

>> w = [-1 5 2]';
>> v = w - ((w'*u)/(u'*u))*u;
v'*u % v orthogonal to u

ans =
  -8.8818e-16

>> c1 = (w'*u)/(u'*u);
>> c2 = (w'*v)/(v'*v);
>> norm((c1*u + c2*v)-w) % w = c1*u + c2*v

ans =
   2.2204e-16

>> Hu*w % should be -c1*u + c2*v

ans =
    0.7143
    4.4286
    3.1429

>> -c1*u + c2*v

ans =
    0.7143
    4.4286
    3.1429

>> Hu'*Hu  % Hu is orthogonal

ans =
    1.0000    0.0000    0.0000
    0.0000    1.0000    0.0000
    0.0000    0.0000    1.0000

>> Hu^2  % Hu*Hu = I

ans =

    1.0000    0.0000    0.0000
    0.0000    1.0000    0.0000
    0.0000    0.0000    1.0000

>> z = [-1 -1 1]';
z'*u % z is orthogonal to u
```

```
ans =
     0

>> Hu*z  % should be z

ans =
   -1.0000
   -1.0000
    1.0000
```

■

17.8.1 Matrix Column Zeroing Using Householder Reflections

A single Householder reflection can zero out all the elements $a_{i+1,i}a_{i+2,i}, \dots, a_{mi}$ below a diagonal element a_{ii}. By applying a sequence of Householder reflections, we can transform a matrix A into an upper triangular matrix. For simplicity, we will begin by developing a Householder matrix H_u that will zero out the elements $a_{21}, a_{31}, \dots, a_{m1}$ below a_{11}. Thus, our goal is to transform

$$A = \begin{bmatrix} X & X & X & X & X & X & X & \cdots & X \\ X & X & X & X & X & X & X & \cdots & X \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ X & X & X & X & X & X & X & \cdots & X \\ X & X & X & X & X & X & X & \cdots & X \end{bmatrix}$$

into the form

$$H_uA = \begin{bmatrix} X & X & X & X & X & X & X & \cdots & X \\ 0 & X & X & X & X & X & X & \cdots & X \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & X & X & X & X & X & X & \cdots & X \\ 0 & X & X & X & X & X & X & \cdots & X \end{bmatrix}.$$

Let x be the first column of A. Then, H_ux will be the first column of H_uA (Equation 2.3). If $x \neq ke_1 = \begin{bmatrix} k & 0 & \dots & 0 \end{bmatrix}^{\mathrm{T}}$, we want to choose u so that H_ux has zeros everywhere in column 1 except at location $(1, 1)$. Elements in the remaining columns will also be affected, but that is of no importance.

The process of choosing u can be viewed geometrically. We know that H_ux reflects x through the plane perpendicular to u, and that $\|H_ux\|_2 = \|x\|$ because H_u is an orthogonal matrix. We want to determine u in such a way that H_ux reflects x to a vector $\pm \|x\|_2 e_1$, i.e., $H_ux = \pm \|x\|_2 e_1$. Figure 17.5 helps in developing an approach. Reflect x through a hyperplane that bisects the angle between x and e_1. This can be done by choosing $u = x - \|x\|_2 e_1$, as Figure 17.5 indicates. A direct computation verifies the result. Begin with

$$H_ux = \left(I - 2\frac{uu^{\mathrm{T}}}{u^{\mathrm{T}}u}\right)x = x - 2\frac{(x - \|x\|_2 e_1)\left(x^{\mathrm{T}} - \|x\|_2 e_1^{\mathrm{T}}\right)x}{\|u\|_2^2}. \tag{17.6}$$

$$\|u\|_2^2 = \|x - \|x\|_2 e_1\|_2^2 = 2\|x\|_2\left(\|x\|_2 - x^{\mathrm{T}}e_1\right) = 2\|x\|_2(\|x\|_2 - x_1) \tag{17.7}$$

Evaluating the numerator of Equation 17.6 using the results $xe_1^{\mathrm{T}}x = x_1x$ and $e_1e_1^{\mathrm{T}}x = x_1e_1$, we have

$$(x - \|x\|_2 e_1)\left(x^{\mathrm{T}} - \|x\|_2 e_1^{\mathrm{T}}\right)x = \|x\|_2\left[(\|x\|_2 - x_1)x - \|x\|_2(\|x\|_2 - x_1)e_1\right]. \tag{17.8}$$

Using Equations 17.6–17.8, there results

$$H_ux = x - \frac{2\|x\|_2\left[(\|x\|_2 - x_1)x - \|x\|_2(\|x\|_2 - x_1)e_1\right]}{2\|x\|_2(\|x\|_2 - x_1)} = x - (x - \|x\|_2 e_1) = \|x\|_2 e_1.$$

A similar calculation shows that if $u = x + \|x\|_2 e_1$, $H_ux = -\|x\|_2 e_1$. Thus H_ux will eliminate all entries of x except that at index 1 by choosing either $u = x - \|x\|_2 e_1$ or $u = x + \|x\|_2 e_1$.

The sign in $u = x \pm \|x\|_2 e_1$ must be chosen carefully to avoid cancelation error. Now,

$$\begin{bmatrix} x_1 & x_2 & \dots & x_{m-1} & x_m \end{bmatrix}^{\mathrm{T}} \pm \|x\|_2 e_1 = \begin{bmatrix} x_1 \pm \|x\|_2 & x_2 & \dots & x_{m-1} & x_m \end{bmatrix}^{\mathrm{T}}$$

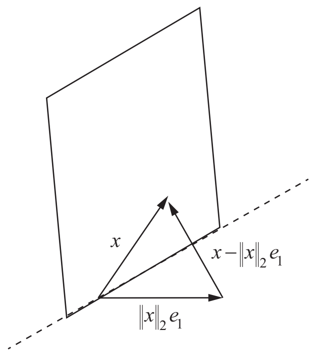


FIGURE 17.5 Householder reflection to a multiple of e_1.

so the only component different from x is the first component of u. To avoid subtraction and possible cancelation error, choose the sign to be that of x_1 so an addition is done instead of a subtraction.

$$u = \begin{cases} x + \|x\|_2 e_1, & x_1 > 0 \\ x - \|x\|_2 e_1, & x_1 < 0 \\ x + \|x\|_2 e_1, & x_1 = 0 \end{cases}. \tag{17.9}$$

Another possible problem is overflow or underflow when computing $\|x\|_2$. Section 8.4.1 presents a strategy to avoid overflow during the calculation.

1. colmax $= \max(|x_2|, |x_2|, \ldots, |x_m|)$.
2. $\overline{x} = x/\text{colmax}$.
3. $\|x\|_2 = \text{colmax}\, \|\overline{x}\|_2$

One of the components of $\overline{x}$ will have absolute value 1, and the remaining components will have absolute value less than or equal to 1, so there is no possibility of overflow. For computing u from Equation 17.9, we only execute steps 1 and 2. Then, $1 \leq \|\overline{x}\| \leq \sqrt{m}$, and there can be no overflow or underflow in computing $\overline{x}$. Let

$$\overline{u} = \left(\frac{x}{\text{colmax}}\right) + \left\|\frac{x}{\text{colmax}}\right\|_2 e_1$$

and use the Householder reflection $H_{\overline{u}}$. It follows that $\overline{u} = \left(\frac{1}{\text{colmax}}\right) u$ (Problem 17.4), and as Lemma 17.2 shows, multiplying u by a constant does not change the value of H_u.

Lemma 17.2. *If H_u is a Householder reflection and k is a constant, then $H_{ku} = H_u$.*

Proof.

$$\begin{aligned} H_{ku} &= \left(I - \frac{2(ku)(ku)^{\mathrm{T}}}{\|ku\|_2^2}\right) \\ &= I - \left(\frac{2k^2 uu^{\mathrm{T}}}{k^2 \|u\|_2^2}\right) = H_u \end{aligned}$$ □

Example 17.11. For $A = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 4 & 1 \\ -3 & 1 & 2 \end{bmatrix}$, find and apply a Householder reflection to zero out entries a_{21} and a_{31}. We will do this in four steps, retaining four decimal places.

1. *Determine x.* The maximum element in magnitude in column 1 is −3, so

$$x = \frac{1}{3}\begin{bmatrix} 1 & -1 & -3 \end{bmatrix} = \begin{bmatrix} 0.3333 & -0.3333 & -1.0000 \end{bmatrix}^{\mathrm{T}}.$$

2. *Compute u and* β. Let

$$u = x \pm \|x\|_2 e_1 = \begin{bmatrix} 0.3333 \\ -0.3333 \\ -1.0000 \end{bmatrix} \pm \left\| \begin{matrix} 0.3333 \\ -0.3333 \\ -1.0000 \end{matrix} \right\| \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.3333 \\ -0.3333 \\ -1.0000 \end{bmatrix} \pm 1.1055 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Choose the sign + because $x_1 = 0.3333 \geq 0$, and

$$u = \begin{bmatrix} 0.3333 \\ -0.3333 \\ -1.0000 \end{bmatrix} + \begin{bmatrix} 1.1055 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1.4388 \\ -0.3333 \\ -1.0000 \end{bmatrix}, \quad \beta = \frac{2}{u^{\mathrm{T}} u} = 0.6287.$$

3. *The Householder reflection is*

$$H_u = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \beta \begin{bmatrix} 1.4388 \\ -0.3333 \\ -1.0000 \end{bmatrix} \begin{bmatrix} 1.4388 \\ -0.3333 \\ -1.0000 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} -0.3015 & 0.3015 & 0.9045 \\ 0.3015 & 0.9301 & -0.2095 \\ 0.9045 & -0.2095 & 0.3713 \end{bmatrix}.$$

4. *Form*

$$H_u A = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 4 & 1 \\ -3 & 1 & 2 \end{bmatrix} - \beta u u^{\mathrm{T}} \begin{bmatrix} 1 & 2 & 0 \\ -1 & 4 & 1 \\ -3 & 1 & 2 \end{bmatrix} = \begin{bmatrix} -3.3166 & 1.5076 & 2.1106 \\ 0.0000 & 4.1141 & 0.5111 \\ 0.0000 & 1.3422 & 0.5332 \end{bmatrix}.$$ ■

17.8.2 Implicit Computation with Householder Reflections

As was the case with Givens rotations, multiplication by a Householder reflection H_u does not require construction of the matrix. If H_u is an $m \times m$ Householder matrix, x is a vector in $\mathbb{R}^m$, and A is an $m \times n$ matrix, then the products $H_u x$ and $H_u A$ can be computed by a simple formula. If H_u is an $n \times n$ matrix, then the same is true for AH_u.

Let $\beta = \frac{2}{u^{\mathrm{T}} u}$.

$$H_u v = \left(I - \beta u u^{\mathrm{T}}\right) v = v - \beta u \left(u^{\mathrm{T}} v\right) \tag{17.10}$$

$$H_u A = \left(I - \beta u u^{\mathrm{T}}\right) A = A - \beta u u^{\mathrm{T}} A \tag{17.11}$$

$$A H_u = A \left(I - \beta u u^{\mathrm{T}}\right) = A - \beta A u u^{\mathrm{T}} \tag{17.12}$$

We are now prepared to formally state the algorithm, hzero1, for zeroing out all elements below a_{11} using a Householder reflection. The function returns the vector u because it will be needed to form Q during the Householder QR decomposition algorithm.

Algorithm 17.4 Zero Out Entries in the First Column of a Matrix using a Householder Reflection

```
function HZERO1(A)
   % Zero out all elements $a_{21} \dots a_{m1}$ in the
   % $m \times n$ matrix A using a Householder reflection $H_u$
   % [A u] - hzero1(A) returns u and a new matrix A implicitly
   % premultiplied by the Householder matrix Hu.

   x=A(:,1)
   colmax=max$\left( \left[\ |x_1|,\ |x_2|,\ \dots,\ |x_{m-1}|,\ |x_m|\ \right]\right)$
   x=x/colmax
   colnorm$= \|x\|_2$
   u=x
   if $u_1 \geq 0$ then
      $u_1 = u_1 + colnorm$
   else
      $u_1 = u_1 - colnorm$
   end if

   % implicitly form $H_uA$.
   unorm$= \|u\|_2$
   if $unorm \neq 0$ then
      $\beta = 2/unorm$
   else
      $\beta = 0$
   end if
   $A = A - (\beta u)\left(u^T A\right)$
   return $\left[\ A,\ u\ \right]$
end function
```

NLALIB: The function `hzero1` implements Algorithm 17.4.

17.9 COMPUTING THE *QR* DECOMPOSITION USING HOUSEHOLDER REFLECTIONS

To transform an $m \times n$ matrix into upper triangular form, we must zero out all the elements below the diagonal entries $a_{11}, a_{22}, \dots, a_{kk}$, where $k = \min(m-1, n)$. We know how to do this for a_{11} (Algorithm 17.4), and now we will demonstrate how to zero out the elements below the remaining diagonal entries. This is done by implicitly creating a sequence of Householder matrices that deal with submatrix blocks, as illustrated in Figure 17.6.

Zeroing out all the elements below a_{11} using a Householder matrix gives matrix A_1. Now we must deal with the submatrices having a diagonal element in their upper left-hand corner. Assume we have zeroed out all the elements below diagonal indices $(1,1)$ through $(i-1, i-1)$ (Figure 17.7) by computing a sequence of matrices $A_1, A_2, \dots, A_{i-1}$. We must find a Householder reflection that zeros out $\overline{a_{i+1,i}}, \dots, \overline{a_{mi}}$ and only modifies elements in the submatrix denoted in Figure 17.7. Consider this $(m-i+1) \times (n-i+1)$ matrix as the matrix whose first column must be transformed to $\left[\ X\ \ 0\ \ \dots\ \ 0\ \right]^{\mathrm{T}}$ by using hzero1. The vector x used for the formation of H_u is $\left[\ \overline{a_{ii}}\ \ \overline{a_{i+1,i}}\ \ \dots\ \ \overline{a_{mi}}\ \right]^{\mathrm{T}}$. Imagine we build the matrix in Figure 17.8, say $\tilde{H}_u$, and compute $\tilde{H}_u A_{i-1}$. $\tilde{H}_u$ is orthogonal and is structured so it only affects the elements of the submatrix shown in Figure 17.7. To compute R we only have to carry out the calculations that modify the $(m-i+1) \times (n-i+1)$ matrix. Do this using Equation 17.13.

$$\left[\ A\,(i:m, i:n)\ \ u\ \right] = \text{hzero1}(A(i:m, i:n)). \tag{17.13}$$

The application of Equation 17.13 for $i = 1, 2,\ k = \min(m-1, n)$ determines R in a highly efficient fashion, since it only deals with the submatrices that must be modified, and the submatrices become smaller with each step.

$$A=\begin{bmatrix}X&X&X&X&X&X&X&X\\X&X&X&X&X&X&X&X\\X&X&X&X&X&X&X&X\\X&X&X&X&X&X&X&X\\X&X&X&X&X&X&X&X\\X&X&X&X&X&X&X&X\\X&X&X&X&X&X&X&X\\X&X&X&X&X&X&X&X\end{bmatrix}\rightarrow \overset{A_1}{H_{u_1}}=\begin{bmatrix}X&X&X&X&X&X&X&X\\0&X&X&X&X&X&X&X\\0&X&X&X&X&X&X&X\\0&X&X&X&X&X&X&X\\0&X&X&X&X&X&X&X\\0&X&X&X&X&X&X&X\\0&X&X&X&X&X&X&X\\0&X&X&X&X&X&X&X\end{bmatrix}\rightarrow$$

$$\overset{A_2}{H_{u_2}H_{u_1}A}=\left[\begin{array}{c|ccccccc|}\multicolumn{1}{c}{X}&X&X&X&X&X&X&\multicolumn{1}{c}{X}\\ \cline{2-8} 0&X&X&X&X&X&X&X\\0&0&X&X&X&X&X&X\\0&0&X&X&X&X&X&X\\0&0&X&X&X&X&X&X\\0&0&X&X&X&X&X&X\\0&0&X&X&X&X&X&X\\0&0&X&X&X&X&X&X\\ \cline{2-8}\end{array}\right]\rightarrow \overset{A_3}{H_{u_3}H_{u_2}H_{u_1}}=\left[\begin{array}{cc|cccccc|}X&\multicolumn{1}{c}{X}&X&X&X&X&X&\multicolumn{1}{c}{X}\\0&\multicolumn{1}{c}{X}&X&X&X&X&X&\multicolumn{1}{c}{X}\\ \cline{3-8} 0&0&X&X&X&X&X&X\\0&0&0&X&X&X&X&X\\0&0&0&X&X&X&X&X\\0&0&0&X&X&X&X&X\\0&0&0&X&X&X&X&X\\0&0&0&X&X&X&X&X\\ \cline{3-8}\end{array}\right]\rightarrow$$

$$\overset{A_4}{H_{u_4}H_{u_3}H_{u_2}H_{u_1}}=\left[\begin{array}{ccc|ccccc|}X&X&\multicolumn{1}{c}{X}&X&X&X&X&\multicolumn{1}{c}{X}\\0&X&\multicolumn{1}{c}{X}&X&X&X&X&\multicolumn{1}{c}{X}\\0&0&\multicolumn{1}{c}{X}&X&X&X&X&\multicolumn{1}{c}{X}\\ \cline{4-8} 0&0&0&X&X&X&X&X\\0&0&0&0&X&X&X&X\\0&0&0&0&X&X&X&X\\0&0&0&0&X&X&X&X\\0&0&0&0&X&X&X&X\\ \cline{4-8}\end{array}\right]$$

FIGURE 17.6 Transforming an $m \times n$ matrix to upper triangular form using householder reflections.

$$A_{i+1}=\left[\begin{array}{cccc|ccccc|}X&X&\ldots&\multicolumn{1}{c}{X}&X&X&X&\ldots&\multicolumn{1}{c}{X}\\0&X&\ldots&\multicolumn{1}{c}{X}&X&X&X&\ldots&\multicolumn{1}{c}{X}\\0&0&\ddots&\multicolumn{1}{c}{X}&X&X&X&\ldots&\multicolumn{1}{c}{X}\\&&&\multicolumn{1}{c}{X}&X&X&X&\ldots&\multicolumn{1}{c}{X}\\ \cline{5-9} &&&0&\bar{a}_{ii}&\bar{a}_{i,i+1}&\bar{a}_{i,i+2}&\ldots&\bar{a}_{in}\\ \vdots&\vdots&\vdots&\vdots&\bar{a}_{i+1,i}&\bar{a}_{i+1,i+1}&\bar{a}_{i+1,i+2}&\ldots&\bar{a}_{i+1,n}\\&&&&\bar{a}_{i+2,i}&\bar{a}_{i+2,i+1}&\bar{a}_{i+2,i+2}&\ldots&\bar{a}_{i+2,n}\\&&&&\vdots&\vdots&\vdots&\ddots&\vdots\\0&0&\ldots&0&\bar{a}_{mi}&\bar{a}_{m,i+1}&\bar{a}_{m,i+2}&\ldots&\bar{a}_{mn}\\ \cline{5-9}\end{array}\right]$$

FIGURE 17.7 Householder reflections and submatrices.

$$\begin{array}{|c|c|}\hline i-1\left\{\overbrace{\begin{matrix}1&&&&0\\&1&&&\\&&1&&\\&&&\ddots&\\0&&&&1\end{matrix}}^{i-1}\right. & 0\\ \hline 0 & \begin{matrix}H_u \text{ of size}\\(m-i+1)\times(m-i+1)\end{matrix}\\ \hline\end{array}$$

FIGURE 17.8 Householder reflection for a submatrix.

Example 17.12. Let $A = \begin{bmatrix} 1 & 5 & -1 & 8 & 3 \\ -1 & 4 & 12 & 6 & -9 \\ 0 & 3 & 16 & -1 & -6 \\ 8 & 1 & 4 & 9 & -2 \\ 1 & 2 & 7 & 8 & 0 \\ 15 & 22 & 17 & -1 & 5 \\ 23 & -7 & 1 & 7 & 9 \end{bmatrix}$. After zeroing out elements below the entries at indices (1,1) and (2,2), we have the matrix

$$H_{u_2}H_{u_1}A = \begin{bmatrix} -1.2458 & -6.2820 & -10.6097 & -7.9573 & -9.7023 \\ 0 & -1.4375 & -17.4780 & 1.4939 & 2.3462 \\ 0 & 0 & 12.7841 & -1.5443 & -4.8219 \\ 0 & 0 & 3.5983 & 5.0658 & -6.2295 \\ 0 & 0 & 4.9398 & 7.1680 & 0.2076 \\ 0 & 0 & -5.3266 & -12.0282 & 4.9730 \\ 0 & 0 & 10.4307 & -2.5192 & -7.0378 \end{bmatrix} = A_2.$$

The task is to zero out all the elements in column 3 below 12.7841. Start with

$$x = \begin{bmatrix} 12.7841 \\ 3.5983 \\ 4.9398 \\ -5.3266 \\ 10.4307 \end{bmatrix}$$

and apply the steps we have described.

$$x = x/12.7841 = \begin{bmatrix} 1.0000 \\ 0.2815 \\ 0.3864 \\ -0.4167 \\ 0.8159 \end{bmatrix}, \quad u = x + \|x\| \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2.4380 \\ 0.2815 \\ 0.3864 \\ -0.4167 \\ 0.8159 \end{bmatrix}$$

$$\beta = \frac{2}{u^{\mathrm{T}}u} = 0.2852, \quad A_3\,(3:7, 3:5) = A_2\,(3:7, 3:5) - \beta u u^{\mathrm{T}} A_2\,(3:7, 3:5),$$

$$A_3 = \begin{bmatrix} -1.2458 & -6.2820 & -10.6097 & -7.9573 & -9.7023 \\ 0 & -1.4375 & -17.4780 & 1.4939 & 2.3462 \\ 0 & 0 & -1.4380 & -3.8995 & 9.9508 \\ 0 & 0 & 0 & 4.7939 & -4.5240 \\ 0 & 0 & 0 & 6.7947 & 2.5490 \\ 0 & 0 & 0 & -11.6257 & 2.4483 \\ 0 & 0 & 0 & -3.3074 & -2.0939 \end{bmatrix}$$

We can now develop the QR decomposition using Householder reflections. It may be that we are only interested in computing R. For instance,

$$A^{\mathrm{T}}A = (QR)^{\mathrm{T}}(QR) = R^{\mathrm{T}}Q^{\mathrm{T}}QR = R^{\mathrm{T}}R,$$

and R is the factor in the Cholesky decomposition of $A^{\mathrm{T}}A$. Not forming Q saves computing time. If we require Q, it will be necessary to implicitly compute the $\tilde{H}_i$ (Figure 17.8) as we transform A into the upper triangular matrix R. Then we have

$$\begin{aligned}\tilde{H}_k\tilde{H}_{k-1}\tilde{H}_{k-2}\ldots\tilde{H}_2\tilde{H}_1A &= R\\ A &= \left(\tilde{H}_k\tilde{H}_{k-1}\tilde{H}_{k-2}\ldots\tilde{H}_2\tilde{H}_1\right)^{\mathrm{T}}R\\ &= \left(\tilde{H}_1\tilde{H}_2\tilde{H}_3\ldots\tilde{H}_{k-1}\tilde{H}_k\right)R\end{aligned}$$

and

$$Q = \tilde{H}_1\tilde{H}_2\tilde{H}_3\ldots\tilde{H}_{k-1}\tilde{H}_k.$$

Start with the $m \times m$ identity matrix $Q = I$. Implicitly compute $Q(1:m, 1:m) = IH_{u_1}$, which changes entries in all rows and columns of I. Now compute $Q(1:m, 2:m) = Q(1:m, 2:m)H_{u_2}$, which only affects columns 2 through m. Continue this k times. Algorithm 17.5 implements the full *QR* decomposition of an $m \times n$ matrix.

Algorithm 17.5 Computation of *QR* Decomposition Using Householder Reflections

```
function HQR(A)
   % Compute the QR decomposition of matrix A using Householder reflections.
   % Input: $m \times n$ matrix A. There are no restrictions on the values of m and n.
   % Output: $m \times m$ matrix Q and $m \times n$ upper triangular matrix R
   % such that A=QR.
   R=A
   Q=I
   $k = \min(m-1, n)$
   for i=1:k do
      $\left[\,R(i:m, i:n),\ u\,\right] = hzero1(R(i:m, i:n))$
      $Q(1:m, i:m) = Q(1:m, i:m) - \left(2/\|u\|_2^2\right)Q(1:m, i:m)\left(uu^T\right)$
   end for
end function
```

NLALIB: The function hqr implements Algorithm 17.5.

This is generally the preferred method for computing the *QR* decomposition, since it is faster than using Givens rotations. However, as we have stated, the *QR* decomposition using Givens rotations can be parallelized, and this is a advantage in an era of multicore processors and GPU computing. In addition, we will require the use of Givens rotations when computing eigenvalues and the singular value decomposition (SVD).

Example 17.13.

Let x be the vector of values beginning at -1.0 and ending at 1.0 in steps of 0.01. Use x to build a Vandermonde matrix, V, using $m = 20$, and apply both hqr and qr to V.

```
>> x = -1.0:0.01:1.0;
>> V = vandermonde(x,20);
>> cond(V)

ans =
  1.7067e+007

>> [Q R] = hqr(V);
>> [QM RM] = qr(V);
>> norm(V - Q*R)

ans =
  5.9967e-014

>> norm(V - QM*RM)

ans =
  9.5622e-015
```

```
>> norm(Q'*Q - eye(201))

ans =
  2.6553e-015

>> norm(QM'*QM - eye(201))

ans =
  1.7922e-015
```
■

17.9.1 Efficiency and Stability

Our analysis will not include the computation of Q. Including this calculation more than doubles the flop count. Assume $m > n$ so $k = n$. If we determine the flop count $\text{flop}_R(i)$ for the statement

$$\boxed{\left[\; R(i:m,\, i:n) \quad u \;\right] = \text{hzero1}\,(R(i:m,\, i:n))}$$

then $\sum_{i=1}^{n} \text{flop}_R(i)$ is the flop count for building R.

Before determining the flop count, we need to consider the computation $A - \beta uu^{\mathrm{T}}A$, since the order of evaluation drastically effects its flop count (Problem 17.5). We will assume that the expression is computed in the following order:

1. $v = \beta u^{\mathrm{T}}$: $(m-i+1)$ flops
2. $v = vA$: $(1 \times (m-i+1))$ matrix times $(m-i+1) \times (n-i+1)$ matrix requires $2(m-i+1)(n-i+1)$ flops.
3. $A - uv$: A is an $(m-i+1) \times (n-i+1)$ and uv is an $(m-i+1) \times (n-i+1)$ matrix. Using nested loops, this calculation can be done in $2(m-i+1) \times (n-i+1)$ flops, so we can compute $A - \beta uu^{\mathrm{T}}A$ in $(m-i+1) + 4(m-i+1)(n-i+1)$ flops.

Each instance of Algorithm 17.4 works with a matrix A of dimension $(m-i+1) \times (n-i+1)$. For each i, list the computations required, the flop count for each, and then form the total sum, $\text{flop}_R(i)$.

Expression	Flop Count
$x = x/\text{colmax}$	$m-i+1$
$\text{colnorm} = \lVert x \rVert_2$	$2(m-i+1)$
$u_1 \pm \text{colnorm}$	1
$\text{unorm} = \lVert u \rVert_2$	$2(m-i+1)$
$\beta = 2/\text{unorm}^2$	2
$A - u\left(\left(\beta u^{\mathrm{T}}\right)A\right)$	$(m-i+1) + 4(m-i+1)(n-i+1)$

$$\text{flop}_R(i) = 6(m-i+1) + 4(m-i+1)(n-i+1) + 3$$

The total flop count for computing R is

$$\sum_{i=1}^{n} \left[6(m-i+1) + 4(m-i+1)(n-i+1) + 3\right]$$

Discard $\sum_{i=1}^{n} \left[6(m-i+1) + 3\right]$, since it contributes only low-order terms. Using standard summation formulas and some algebra, we obtain

$$\begin{aligned} 4\sum_{i=1}^{n} (m-i+1)(n-i+1) &= \frac{2}{3}n(n+1)(3m-n+1) \\ &= 2\left(mn^2 - \frac{n^3}{3}\right) + 2mn + \frac{2}{3}n \end{aligned}$$

Again, exclude low-order terms, and we have

$$\text{flop count} \approx 2\left(mn^2 - \frac{n^3}{3}\right).$$

For the Givens algorithm, we computed the flop count for determining both R and Q. If we do the analysis and exclude Q, we have

$$\text{Givens flop count} = (5 + 6n) \sum_{i=1}^{n} (m - i) \approx 3n^2 (2m - n)$$

If we consider an $n \times n$ matrix, the Givens algorithm is approximately twice as costly as the Householder algorithm. Of course, this is because hqr zeros out all the elements in column i below a_{ii} in one reflection, whereas the Givens algorithm requires $(m - i)$ rotations.

The Householder algorithm for computing the QR decomposition is stable [16, pp. 357-360].

17.10 CHAPTER SUMMARY

Gram-Schmidt *QR* Decomposition

The modified Gram-Schmidt process (never use classical Gram-Schmidt unless you perform reorthogonalization) gives a reduced QR decomposition, and its algorithm for orthonormalization of set of linearly independent vectors has other applications. Its flop count of $2mn^2$ is superior to the Givens and Householder when both Q and R are required. However, its stability depends on the condition number of the matrix, so it is not as stable as the other methods.

Givens *QR* Decomposition

Assume A is an $m \times n$ matrix. If c and s are constants, an $m \times m$ Givens matrix $J(i, j, c, s)\ i < j$, also called a Givens rotation, places c at indices (i, i) and (j, j), $-s$ at (j, i), and s at (i, j) in the identify matrix. $J(i, j, c, s)$ is orthogonal, and by a careful choice of constants (Algorithm 17.2), $J(i, j, c, s)\, A$ affects only rows i and j of A and zeros out a_{ji}. The product is performed implicitly by changing just rows i and j, so it is rarely necessary to build a Givens matrix. By zeroing out all the elements below the main diagonal, the Givens QR algorithm produces the upper triangular matrix R in the decomposition. By maintaining a product of Givens matrices, Q can also be found. The algorithm is stable, and its perturbation analysis does not involve the condition number of A. While the Givens QR decomposition is efficient and stable, Householder reflections are normally used for the QR decomposition. However, because premultiplication by a Givens matrix can zero out a particular element, these matrices are very useful when A has a structure that lends itself to zeroing out one element at a time (Problem 17.12).

Householder Decomposition

The use of Householder matrices, also termed a Householder reflections, is the most commonly used method for performing the QR decomposition. If u is an $m \times 1$ vector, the Householder matrix defined by

$$H_u = I - \left(\frac{2}{u^{\mathrm{T}} u}\right) u u^{\mathrm{T}}$$

is orthogonal and symmetric. Products $H_u v$, $H_u A$, and $A H_u$, where A is an $m \times n$ matrix and v is an $m \times 1$ vector can be computed implicitly without the need to build H_u. By a proper choice of u (Equation 17.9), $H_u A$ zeros out all the elements below a diagonal element a_{ii}, and so it is an ideal tool for the QR decomposition. The Householder QR decomposition is stable and, like the Givens QR process, its perturbation analysis does not depend on the condition number of A. It is this "all at once" feature of Householder matrices that makes them so useful for matrix decompositions. They will be very important in our study of eigenvalue computation in Chapters 18 and 19.

17.11 PROBLEMS

17.1 Using pencil and paper, compute the QR decomposition of $A = \begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix}$ using a Givens rotation that you explicitly build.

17.2 Using pencil and paper, compute the QR decomposition of $A = \begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix}$ using a Householder reflection that you explicitly build.

17.3 Show that in the full QR decomposition of the full rank $m \times n$ matrix $A, m \geq n$, the vectors $q_{n+1}, \ldots, q_m$ are an orthonormal basis for the null space of A^{T}. Hint: Using block matrix notation, write

$$Q = \begin{bmatrix} Q_1^{m\times n} & Q_2^{(m-n)\times n} \end{bmatrix},$$

where Q_1 consists of the first n columns of Q, and the columns of Q_2 are the remaining $m - n$ columns. Write R in block matrix notation also.

17.4 If $u = x + \|x\|_2\, e_1$, show that if $\overline{u} = kx + \|kx\|_2\, e_1$, where $k \geq 0$ is a constant, then $\overline{u} = ku$.

17.5 If A is an $m \times n$ matrix, and $u \in \mathbb{R}^m$, $v \in \mathbb{R}^m$, the amount of work to evaluate $A - uv^{\mathrm{T}}A$ depends dramatically on the order in which the operations are performed.

a. How many flops are required to compute it in the order

$$\begin{aligned} T_1 &= uv^{\mathrm{T}} \\ T_2 &= T_1 A \\ T_3 &= A - T_2 \end{aligned}$$

b. Show that computing $A - \left(uv^{\mathrm{T}}\right) A$ can be done in approximately $2m^2n + 2mn$ flops. Hint: Determine the structure of each row of uv^{T}, and consider each row of $\left(uv^{\mathrm{T}}\right) A$ as the inner product of a row of $uv^{\mathrm{T}}$with all columns of A.

c. Show that the $A - uv^{\mathrm{T}}A$ can be done in $4mn$ flops by first computing $v^{\mathrm{T}}A$ and then $A - u\left(v^{\mathrm{T}}A\right)$.

17.6 The QR decomposition can be used to solve an $n \times n$ linear system $Ax = b$ using the following steps:

Find the QR decomposition of A: $A = QR$.
Form $b' = Q^{\mathrm{T}}b$.
Solve $Rx = b'$.

Assume that a full QR decomposition of an $n \times n$ matrix requires $4\left(m^2n - mn^2 + \frac{n^3}{3}\right)$ using Householder reflections.

a. Determine the flop count for the solution.

b. Compare your flop count with that of Gaussian elimination. Which method is generally preferable?

17.7 Prove these properties of Householder matrices

a. $H_u^2 = I$

b. $H_u v = v$ if $\langle v, u\rangle = 0$.

17.8 Prove that if u is chosen to be parallel to vector $x - y$, where $x \neq y$ but $\|x\|_2 = \|y\|_2$, then $H_u x = y$. Hint: Let $u = k\,(x - y)$, where k is a constant, and note that

$$x = \frac{1}{2}\,(x+y) + \frac{1}{2}\,(x-y)\,.$$

Compute Hx and apply the relation $H_u\,(u) = -u$. Show that $\langle x+y, x-y\rangle = 0$ and apply Theorem 17.3, part 4.

17.9 Let $A \in \mathbb{R}^{m\times n}, m \geq n$, have reduced QR decomposition $A = QR$. Show that $\|A\|_2 = \|R\|_2$.

17.10 If u and v are $n \times 1$ vectors, the $n \times n$ matrix uv^{T} has rank 1 (Problem 10.3). If A is an $n \times n$ matrix, we say that the matrix $B = A + uv^{\mathrm{T}}$ is a *rank 1 update* of A. Let $A = QR$ be the QR decomposition of A. Show that

$$A + uv^{\mathrm{T}} = Q\left(R + wv^{\mathrm{T}}\right),$$

where $w = Q^{\mathrm{T}}u$.

17.11 Problem 17.10 defines a rank 1 update of a matrix. A Householder reflection, $H_u = I - \left(\frac{2}{\|u\|_2^2}\right) uu^{\mathrm{T}}$, is a rank 1 update of the identity matrix. We know a Householder matrix is symmetric, orthogonal, and is its own inverse ($H_u^2 = I$). This problem investigates a more general rank 1 update of the identity,

$$R_1 = I - uv^{\mathrm{T}}.$$

a. Prove that R_1 is nonsingular if and only if $\langle v, u\rangle \neq 1$.

b. If R_1 is nonsingular, show that $R_1^{-1} = I - \beta uv^{\mathrm{T}}$. Do this by finding a formula for β.

17.12 In this problem, you will investigate a special type of square matrix called an upper Hessenberg matrix. Such a matrix has the property $a_{ij} = 0,\ i > j + 1$, and is often called almost upper triangular.

a. Give an example of a 4×4 and a 5×5 upper Hessenberg matrix.
b. Develop an algorithm for computing the *QR* decomposition of an upper Hessenberg matrix. Hint: Use Givens rotations. How many will be necessary?
c. Show that the flop count is $O\left(n^2\right)$.

17.13 Suppose that $A \in \mathbb{R}^{m \times n}$ has full column rank. Prove that the reduced decomposition

$$A = QR$$

is unique where $Q \in \mathbb{R}^{m \times n}$ has orthonormal columns and $R^{n \times n}$ is upper triangular with positive diagonal entries. Do this in steps.
a. Show that $A^{\mathrm{T}}A = R^{\mathrm{T}}R$.
b. Prove that $A^{\mathrm{T}}A$ is symmetric positive definite, and apply Theorem 13.3 to show that R is the unique Cholesky factor of $A^{\mathrm{T}}A$.
c. Show that Q must be unique.
d. Give an example to show that there is no guarantee of uniqueness if A does not have full column rank.

17.14 We have studied the *LU*, *QR*, and the SVDs. There are many more, and the book will construct additional ones in later chapters. This problem develops two variants of the *QR* decomposition, the *QL* and the *RQ*.
An $m \times m$ matrix of the form

$$K_m = \begin{bmatrix} & & & & 1 \\ & & & 1 & \\ & & \ddots & & \\ & 1 & & & \\ 1 & & & & \end{bmatrix} = \left[k_{ij}\right],$$

where $k_{i,m-i+1} = 1,\ 1 \le i \le m$, and all other entries are zero is termed a *reversal matrix* and sometimes the *reverse identity matrix*.
a. Show that $K_m^2 = I$ (a very handy feature).
b. If A is an $m \times n$ matrix, $m \ge n$, what is the action of K_mA? What about AK_n?
c. If R is upper triangular $n \times n$ matrix, what is the form of the product K_nRK_n?
d. Let $AK_n = \hat{Q}\hat{R}$ be the reduced *QR* decomposition of AK_n, $m \ge n$. Show that $A = \left(\hat{Q}K_n\right)\left(K_n\hat{R}K_n\right)$, and from that deduce the decomposition

$$A = QL,$$

where Q is an $m \times n$ matrix with orthogonal columns, and L is an $n \times n$ lower triangular matrix. This is a *reduced QL decomposition*.
e. If $m < n$, show to form an *RQ* decomposition, $A = RQ$, where R is $m \times m$ and Q is $m \times n$.

17.15 If $A \in \mathbb{R}^{m \times n}$, there exists an $m \times n$ lower triangular matrix and an $n \times n$ orthogonal matrix Q such that $A = LQ$.
a. Given the 1×2 vector $\begin{bmatrix} x & y \end{bmatrix}$, show there is a Givens rotation, J, such that $\begin{bmatrix} x & y \end{bmatrix} J = \begin{bmatrix} * & 0 \end{bmatrix}$. This type of rotation eliminates elements from columns.
b. Develop an algorithm using Givens rotations that computes an *LQ* decomposition of A for any $m \times n$ matrix. Hint: Write a function givensmulpost(A,i,j,c,s) that affects only columns i and j and zeros out A(i,j) in row i. The function givensparms does not change.

17.16 In a series of steps, this problem develops the result:

Assume $J(i,j,c,s)$ is a Givens rotation, A is a symmetric matrix, and define $B = J^{\mathrm{T}}(i,j,c,s)AJ(i,j,c,s)$. Then,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2.$$

Recall the following relationships:

- $\|X\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}^2 = \text{trace}\left(X^{\mathrm{T}}X\right)$
- $\text{trace}\,(XY) = \text{trace}\,(YX)$

a. Show that $\|B\|_F^2 = \text{trace}\left(J^{\mathrm{T}}(i,j,c,s)\, A^{\mathrm{T}} A J(i,j,c,s)\right)$.
b. Show that trace $\left(J^{\mathrm{T}}(i,j,c,s)\, A^{\mathrm{T}} A J(i,j,c,s)\right) = \text{trace}\left(AA^{\mathrm{T}}\right)$
c. Conclude that $\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2$.

17.17 Find the eigenvalues of a Householder reflection. Hint: Run some numerical experiments to look for a pattern. If H_u is a Householder reflection, $H_u u = -u$. Starting with u, build a basis $u, v_1, v_2, \ldots, v_n$, and take note of part 4 in Theorem 17.3.

17.11.1 MATLAB Problems

17.18 Given $v = \begin{bmatrix} -1 & 3 & 7 \end{bmatrix}^{\mathrm{T}}$, build the Givens matrix $J(1,2,c,s)$ such that the second component of $J(1,2,c,s)\,v$ is zero. Build the Givens matrix $J(1,3,c,s)$ such that $J(1,3)\,J(1,2,c,s)\,v$ has the form $\begin{bmatrix} * & 0 & 0 \end{bmatrix}^{\mathrm{T}}$. Use a sequence like the following to explicitly build a Givens matrix.

```
>> [c s] = givensparms(xi,xj);
>> J = eye(n);
>> J(i,i) = c;
>> J(j,j) = c;
>> J(j,i) = -s;
>> J(i,j) = s;
```

17.19 Explicitly build the Givens rotations that transform A to upper triangular form. Use the product of the Givens matrices to compute Q. You might want to use the MATLAB statements given in Problem 17.18. Verify that the decomposition is correct by computing $\|A - QR\|_2$. Also compute the decomposition using `qr`.

$$A = \begin{bmatrix} 1 & -1 & 2 \\ 9 & 3 & 4 \\ 3 & -8 & 1 \\ 12 & 10 & 5 \end{bmatrix}$$

17.20 To reinforce your understanding of using Householder reflections to find the QR decomposition, it is helpful to execute the algorithm step-by-step. Here are MATLAB statements that explicitly perform the QR decomposition of the matrix

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

The function `houseparms` in the book software distribution computes u and β for the Householder reflection $H_u(x)$ that zeros out all the elements of x except $x(1)$.

```
A = [1 -1 1;2 1 0;3 -1 1;4 5 3];
R = A;
[m,n] = size(A);

Q = eye(m);

[u, beta] = houseparms(R(:,1));
Hu1 = eye(m) - beta*u*u';
R = Hu1*R;
Q(1:m,1:m) = Q(1:m,1:m) - beta*Q(1:m,1:m)*(u*u');

[u,beta] = houseparms(R(2:m,2));
Hu2 = eye(m-1) - beta*u*u';
R(2:m,2:n) = Hu2*R(2:m,2:n);
Q(1:m,2:m) = Q(1:m,2:m) - beta*Q(1:m,2:m)*(u*u');

[u,beta] = houseparms(R(3:m,3));
Hu3 = eye(m-2) - beta*u*u';
R(3:m,3:n) = Hu3*R(3:m,3:n);
Q(1:m,3:m) = Q(1:m,3:m) - beta*Q(1:m,3:m)*(u*u');
```

Study the code, and the explicitly construct the *QR* decomposition of the matrix

$$A = \begin{bmatrix} 1 & 4 & 6 & 2 & -1 \\ 3 & 6 & 1 & 9 & 10 \\ -6 & 7 & 8 & 1 & 0 \\ 3 & -4 & 1 & -1 & 2 \\ 9 & 12 & 15 & 1 & 5 \\ 35 & 1 & 2 & 3 & 4 \end{bmatrix}.$$

Compute $\|A - QR\|_2$.

17.21 Let

$$A = \begin{bmatrix} 7 & 1 & 6 \\ 9 & 10 & -8 \\ -8 & 10 & -2 \\ 9 & -7 & 9 \\ 3 & 10 & 6 \\ -8 & 10 & 10 \\ -5 & 0 & 3 \end{bmatrix}$$

have *QR* decomposition $A = QR$. Note Problem 17.3 and Theorem 14.3.

a. Use the *QR* decomposition to find the rank of A, and verify that rank $(A) = n$. Find an orthonormal basis for the range of A.

b. Find an orthonormal basis for the null space of A^{T}.

17.22 Find the *QR* decomposition of each matrix using the modified Gram-Schmidt process, Givens rotations, and Householder reflections. In each case compute $\left\|Q^{\mathrm{T}}Q - I\right\|_2$.

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

b. $B = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -1 & 0 & 1 & 0 \end{bmatrix}$. Explain the results.

c. The Frank matrix of order 5 using the statement "`C = gallery('frank', 5);`". Explain the results.

d. The 30 × 20 Chebyshev Vandermonde matrix using the statement "`D = gallery('chebvand',30,20);`". Explain the results.

17.23 Given the four vectors $\begin{bmatrix} 1 & -1 & 4 & 3 & 8 \end{bmatrix}^{\mathrm{T}}, \begin{bmatrix} 9 & -1 & 3 & 0 & 12 \end{bmatrix}^{\mathrm{T}}, \begin{bmatrix} 2 & 1 & -1 & 3 & 5 \end{bmatrix}^{\mathrm{T}}$, and $\begin{bmatrix} 0 & 6 & 7 & -1 & 5 \end{bmatrix}^{\mathrm{T}}$, find an orthonormal basis that spans the same subspace using the Givens *QR* decomposition.

17.24 Develop a MATLAB function

```
function [Q R] = myhqr(A)
```

using Householder reflections that returns one of a set of possibilities according to the format of the calling sequence.

```
%MYHQR Executes the full or reduced QR decomposition to return
%both Q and R or, optionally, just R.
%
%   Full decomposition:
%     [Q, R] = myhqr(A) returns an orthogonal m x m matrix Q
%     and an upper triangular m x n matrix R such that A = QR.
%
%   Reduced decomposition
%     If m > n, [Q, R] = myhqr(A,0) returns an m x n matrix Q with
%     orthonormal columns and an upper triangular n x n matrix R
%     such that A = QR. If m <= n, returns the full QR decomposition.
%
%   R = myhqr(A) returns the m x n upper triangular matrix
%   from the full QR decomposition.
%
%   R = myhqr(A,0) returns the n x n upper triangular matrix
%   from the reduced QR decomposition as long as m > n; otherwise,
%   it returns the m x n upper triangular matrix of the full
%   QR decomposition
%
%   myhqr(A) returns the m x n upper triangular matrix
%   from the full QR decomposition.
%
%   myhqr(A,0) returns the n x n upper triangular matrix
%   from the reduced QR decomposition as long as m > n; otherwise,
%   it returns the m x n upper triangular matrix of the full
%   QR decomposition
```

Use the MATLAB constructs

```
varargout, nargout, varargin, nargin.
```

If you are not familiar with these, consult the MATLAB documentation. Test your function by constructing a 3×2 and a 2×3 matrix and using all the options on each.

17.25 Problem 17.14 presented the QL decomposition of an $m \times n$ matrix $A, m \geq n$. Implement the algorithm in a function `ql` and test it on at two matrices of different sizes. Hint: If I is the $m \times m$ identity matrix,

```
rot90(I)
```

gives the reversal matrix K_m by rotating I 90° counterclockwise.

17.26 Use the result of Problem 17.15 to develop a function `lq` that performs the LQ decomposition and test it with random matrices of sizes 10×7 and 50×75.

17.27 In this problem, we will use computation to motivate a theoretical result.

a. Build a series of Givens rotations, and find the eigenvalues and corresponding eigenvectors for each. A pattern emerges.

b. Propose a formula for the eigenvalues and corresponding eigenvectors of a Givens rotation, and prove you are correct.