Chapter 14

Gram-Schmidt Orthonormalization

You should be familiar with

- The inner product
- Orthogonal vectors and matrices
- Rank
- Algorithm stability
- The determinant

Orthogonal vectors and matrices are of great importance in many fields of science and engineering; for instance, they play an important role in least-squares problems, analysis of waves, the finite element method for the numerical solution of partial differential equations, financial engineering, and quantum mechanics. We determined in Chapter 7 that orthogonal matrices preserve length, a property called orthogonal invariance; for example, in computer graphics, orthogonal matrices are used to rotate an image without altering its size. Also, an orthogonal matrix is used in a change of basis, such as constructing a basis for spherical or cylindrical coordinates. Many problems in dynamics involve a change of basis. We begin this chapter by reviewing some of properties of orthogonal vectors and matrices.

- If $x_1, x_2, \ldots, x_k$ are orthogonal, they are linearly independent.
- If A is a real symmetric matrix, then any two eigenvectors corresponding to different eigenvalues are orthogonal.
- If P is an orthogonal matrix, then $P^{-1} = P^{\mathrm{T}}$.
- If P is an orthogonal matrix, then $\|Px\|_2 = \|x\|_2$.
- Let P be a $n \times n$ real matrix. Then P is an orthogonal matrix if and only if the columns of P are orthogonal and have unit length.
- For any orthogonal matrices U and V, $\|UAV\|_2 = \|A\|_2$.

If we have a basis $v_1, v_2, \ldots, v_n$ for a subspace, it would be valuable to have the means of constructing an orthonormal basis $e_1, e_2, \ldots, e_n$ that spans the same subspace. We can then use the orthonormal basis to build an orthogonal matrix. The Gram-Schmidt process does exactly that, and leads to the QR decomposition of a general real $m \times n$ matrix.

14.1 THE GRAM-SCHMIDT PROCESS

The *Gram-Schmidt process* takes a set of linearly independent vectors $S = \{v_1, v_2, \ldots, v_n\} \in \mathbb{R}^m$, and transforms them into a set of orthonormal vectors $S' = \{e_1, e_2, \ldots, e_n\}$. The orthonormal set $S' = \{e_1 e_2, \ldots, e_n\}$ spans the same n-dimensional subspace as S. It follows that $m \geq n$ or the vectors would be linearly dependent.

Remark 14.1. Although we will deal with the inner product of vectors, v_i, $1 \leq i \leq n$, we can just as well have a set of functions that we transform into an orthonormal set relative to the L^2 norm

$$\langle v_i, v_j \rangle = \int_a^b v_i(t)\, v_j(t)\, \mathrm{d}t.$$

The Gram-Schmidt process works by successively subtracting orthogonal projections from vectors. The projection operator we now define is one of the most important uses of the inner product.

Definition 14.1. The orthogonal projection of vector v onto vector u is done by the projection operator $\mathrm{proj}_u(v) = \left(\langle v, u\rangle / \|u\|_2^2\right) u$. Figure 14.1 depicts the projection.

Numerical Linear Algebra with Applications. http://dx.doi.org/10.1016/B978-0-12-394435-1.00014-4

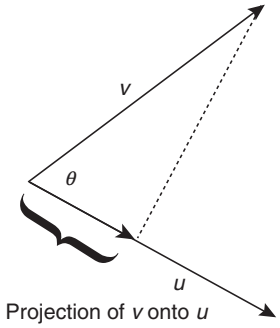

FIGURE 14.1 Vector orthogonal projection.

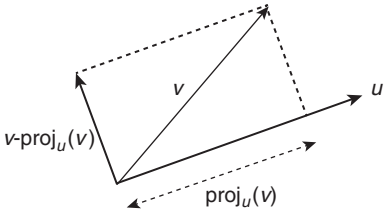

FIGURE 14.2 Removing the orthogonal projection.

The use of some geometry will help understand the projection operator. The length of the projection, d, of v onto u has the value

$$d = \|v\|_2 \cos\theta,$$

which can be negative if $\theta > \pi/2$. The inner product of v and u is $\langle v,\, u\rangle = \langle v,\, u\rangle = \|v\|_2\, \|u\|_2\, cos\theta$. As a result

$$d = \frac{\langle v,\, u\rangle}{\|u\|_2}.$$

Now create a vector with magnitude $|d|$ by multiplying the unit vector $u/\|u\|$ by d to obtain the vector

$$\text{proj}_u\,(v) = \left(\frac{\langle v,\, u\rangle}{\|u\|_2}\right)\frac{u}{\|u\|_2} = \left(\frac{\langle v,\, u\rangle}{\|u\|_2^2}\right)u.$$

Now comes the most critical point. If we remove the orthogonal projection of v onto u from v by computing $v - \text{proj}_u\,(v)$ we obtain a vector orthogonal to u, as depicted in Figure 14.2. We can verify this algebraically as follows:

$$u^T\left(v - \text{proj}_{\text{u}}\,(\text{v})\right) = u^T v - \left(\frac{u^T v}{\|u\|_2^2}\right)u^T u = u^T v - u^T v = 0$$

The Gram-Schmidt process works by successively removing the orthogonal projection of v_i from the orthonormal vectors $e_1, e_2, \ldots, e_{i-1}$ already built from $v_1, v_2, \ldots, v_{i-1}$. This will produce a vector orthogonal to $\{e_1, e_2, \ldots, e_{i-1}\}$ (verify).

Step 1: Begin by taking the first vector, v_1, normalize it to obtain a vector e_1, and define $r_{11} = \|v_1\|_2$. As our description of the process progresses, we will define a set, $\left\{r_{ij},\ 1 \le i,j \le n,\ j \ge i\right\}$. While not strictly necessary for building an orthonormal basis from $v_1, v_2, \cdots, v_k$, we will use these values to form an upper triangular matrix when we discuss the QR decomposition later in this chapter. Let $u_1 = v_1$, and then

$$e_1 = \frac{v_1}{\|v_1\|_2} = \frac{v_1}{r_{11}},\quad r_{11} = \|v_1\|_2\,.$$

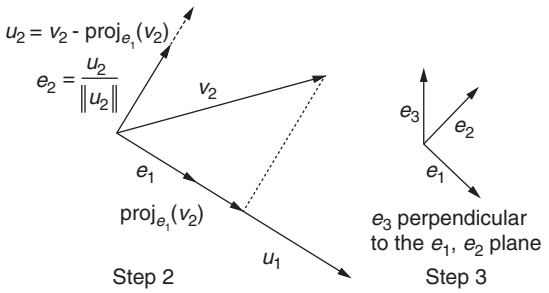

FIGURE 14.3 Result of the first three steps of Gram-Schmidt.

Step 2: Let $r_{12} = \langle v_2, e_1 \rangle$. Define vector, u_2, normal to e_1, by removing the orthogonal projection of v_2 onto e_1. Then normalize it to obtain e_2.

$$u_2 = v_2 - \text{proj}_{e_1}(v_2) = v_2 - \frac{\langle v_2, e_1 \rangle}{\|e_1\|_2^2} e_1 = v_2 - \langle v_2, e_1 \rangle e_1 = v_2 - r_{12} e_1.$$

Normalize the vector u_2, let $r_{22} = \|u\|_2$ and define

$$e_2 = \frac{u_2}{\|u_2\|_2} = \frac{u_2}{r_{22}}.$$

Step 3: Now obtain a vector u_3 normal to both e_1 and e_2 by removing the orthogonal projections of v_3 onto e_1 and e_2. Define $r_{13} = \langle v_3, e_1 \rangle$ and $r_{23} = \langle v_3, e_2 \rangle$.

$$u_3 = v_3 - \text{proj}_{e_1}(v_3) - \text{proj}_{e_2}(v_3) = v_3 - \langle v_3, e_1 \rangle e_1 - \langle v_3, e_2 \rangle e_2 = v_3 - r_{13} e_1 - r_{23} e_2.$$

Normalize u_3, and let $r_{33} = \|u_3\|_2$ to obtain

$$e_3 = \frac{u_3}{\|u_3\|_2} = \frac{u_3}{r_{33}}.$$

See Figure 14.3 showing the results of steps 1-3.

Continue in this fashion until all the vectors up to e_n are determined. The formula for computing a general vector, e_i, is:

Step i: Find

$$u_i = v_i - \sum_{j=1}^{i-1} r_{ji} e_j, \quad 1 \leq i \leq n, \tag{14.1}$$

where

$$r_{ji} = \langle v_i, e_j \rangle.$$

Let $r_{ii} = \|u_i\|_2$. Normalize u_i to obtain

$$e_i = \frac{u_i}{\|u_i\|_2} = \frac{u_i}{r_{ii}}.$$

The sequence $e_1, \ldots, e_k$ is the required set of orthonormal vectors, and the process is known as *Gram-Schmidt orthonormalization*.

This process can be explained geometrically. The vectors e_1 and e_2 are orthogonal (Figure 14.3). The vector u_3 is orthogonal to e_1 and e_2, since it is formed by removing their orthogonal projections from v_3. Continuing in this fashion, vector e_i is orthogonal to the vectors $e_1, e_2, \ldots, e_{i-1}$; furthermore, the vectors $e_1, e_2, \ldots, e_i$ span the same subspace as $v_1, v_2, \ldots, v_i$. To see this, let vector x be a linear combination of the vectors $v_1, v_2, \ldots, v_i$.

$$x = c_1 v_1 + c_2 v_2 + c_3 v_3 + \cdots + c_i v_i.$$

Each vector $v_i = u_i + \sum_{j=1}^{i-1} \text{proj}_{e_j}(v_i)$. Now, each $\text{proj}_{e_j}(v_i)$ is a scalar multiple of e_j, and $u_i = \|u_i\| e_i$, so x can be written as a linear combination of $e_1, e_2, \ldots, e_i$. Since $e_i, e_2, \ldots, e_i$ are orthonormal, they are linearly independent and form a basis for the same subspace as $v_1, v_2, \ldots, v_i$.

If the Gram-Schmidt process is applied to a linearly dependent sequence, then at least one v_i is a linear combination of the remaining vectors,

$$v_i = c_1 v_1 + c_2 v_2 + \cdots + c_{i-1} v_{i-1} + c_{i+1} v_{i+1} + \cdots + c_k v_k.$$

In this situation, on the ith step the result is $u_i = 0$. Discard u_i and v_i and continue the computation. If this happens again, do the same. The number of vectors output by the process will then be the dimension of the space spanned by the original vectors, less the dependent vectors. For the sake of simplicity, we will assume throughout this chapter that all the vectors we deal with are linearly independent.

Example 14.1. Consider the following set of three linearly independent vectors in $\mathbb{R}^3$.

$$S = \left\{ v_1 = \begin{bmatrix} 1 \\ -1 \\ 3 \end{bmatrix}, v_2 = \begin{bmatrix} 3 \\ 1 \\ 4 \end{bmatrix}, v_3 = \begin{bmatrix} 3 \\ 2 \\ 5 \end{bmatrix} \right\}$$

Execute Gram-Schmidt to obtain an orthonormal set of vectors.

$$\begin{aligned}
r_{11} &= \sqrt{11} = 3.3166, \quad e_1 = \frac{\begin{bmatrix} 1 \\ -1 \\ 3 \end{bmatrix}}{r_{11}} = \begin{bmatrix} 0.30151 \\ -0.30151 \\ 0.90453 \end{bmatrix}, \\
r_{12} &= \langle v_2, \quad e_1 \rangle = 4.2212, u_2 = v_2 - r_{12} e_1 = \begin{bmatrix} 1.7273 \\ 2.2727 \\ 0.18182 \end{bmatrix}, \quad e_2 = \frac{u_2}{\|u_2\|_2} = \begin{bmatrix} 0.60386 \\ 0.79455 \\ 0.063564 \end{bmatrix}, \\
r_{22} &= \|u_2\| = 2.8604, \\
r_{13} &= \langle v_3, e_1 \rangle = 4.8242, \quad r_{23} = \langle v_3, e_2 \rangle = 3.7185, \quad u_3 = v_3 - r_{13} e_1 - r_{23} e_2 = \begin{bmatrix} -0.7 \\ 0.5 \\ 0.4 \end{bmatrix}, \\
e_3 &= \frac{u_3}{\|u_3\|} = \begin{bmatrix} -0.737861 \\ 0.52705 \\ 0.42164 \end{bmatrix}, \quad r_{33} = \|u_3\| = 0.94868.
\end{aligned}$$

Summary

$$e_1 = \begin{bmatrix} 0.30151 \\ -0.30151 \\ 0.90453 \end{bmatrix}, \; e_2 = \begin{bmatrix} 0.60386 \\ 0.79455 \\ 0.063564 \end{bmatrix}, \; e_3 = \begin{bmatrix} -0.737861 \\ 0.52705 \\ 0.42164 \end{bmatrix},$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix} = \begin{bmatrix} 3.3166 & 4.2212 & 4.8242 \\ 0 & 2.8604 & 3.7185 \\ 0 & 0 & 0.94868 \end{bmatrix}.$$

■

Algorithm 14.1 describes the Gram-Schmidt process. The algorithm name, clgrsch, reflects the fact that the process we have described is called *classical Gram-Schmidt*. As we will see in Section 14.2, classical Gram-Schmidt has numerical stability problems, and we normally use a simple modification of the classical process.

Remark 14.2. The function clgrsch in Algorithm 14.1 takes an $m \times n$ matrix as input. It is important to understand that there are n vectors to be orthonormalized, and the dimension of each vector is m; however, the function does not compute the values r_{ij}. We will use them when developing the QR decomposition of a matrix.

NLALIB: The function `clgrsch` implements Algorithm 14.1.

14.2 NUMERICAL STABILITY OF THE GRAM-SCHMIDT PROCESS

During the execution of the Gram-Schmidt process, the vectors u_i are often not quite orthogonal, due to rounding errors. For the classical Gram-Schmidt process we have described, this loss of orthogonality is particularly bad. The computation also

Algorithm 14.1 Classical Gram-Schmidt

```
function CLGRSCH(V)
   % Converts a set of linearly independent vectors to a set
   % of orthonormal vectors spanning the same subspace
   % Input: An m × n matrix V whose columns are the vectors to be normalized.
   % Output: An m × n matrix E whose columns are an orthonormal set of
   % vectors spanning the same subspace as the columns of V.
   for i = 1:n do
      sumproj = 0
      for j = 1:i-1 do
         sumproj = sumproj + E(:, j)^T V(:, i) E(:, j)
      end for
      E(:, i) = V(:, i) − sumproj
      E(:, i) = E(:, i) / ‖E(:, i)‖_2
   end for
   return E
end function
```

yields poor results when some of the vectors are almost linearly dependent. For these reasons, it is said that the classical Gram-Schmidt process is numerically unstable.

The Gram-Schmidt process can be improved by a small modification. The computation of u_i using the formula

$$u_i = v_i - \sum_{j=1}^{i-1} r_{ji} e_j$$

removes the projections all at once. Split the computation into smaller parts by removing the projections one at a time.

$u_i^{(1)} = v_i - \langle v_i, e_1 \rangle e_1$ remove the projection of v_i onto e_1.

$u_i^{(2)} = u_i^{(1)} - \langle v_i, e_2 \rangle e_2$ remove the projection of v_i onto e_2.

$u_i^{(3)} = u_i^{(2)} - \langle v_i, e_3 \rangle e_3$ remove the projection of v_i onto e_3.

$\vdots$

$u_i^{i-1} = u_i^{(i-2)} - \langle v_i, e_{i-2} \rangle e_{i-2}$ remove the projection of v_i onto e_{i-2}

$u_i = u_i^{(i-1)} - \langle v_i, e_{i-1} \rangle e_{i-1}$ remove the projection of v_i onto e_{i-1}.

This approach (sometimes referred to as *modified Gram-Schmidt (MGS) process*) gives the same result as the original formula in exact arithmetic, but it introduces smaller roundoff errors when executed on a computer.

The following algorithm implements the MGS process. The input and output format is the same as for clgrsch.

NLALIB: The function `modgrsch` implements Algorithm 14.2.

Remark 14.3. The orthonormal basis obtained by modgrsch are the columns of the output matrix E, so E is an orthogonal matrix.

Example 14.2. Apply the function `modgrsch` to the vectors of Example 14.1, obtain an orthonormal basis. Verify that the columns in matrix E are orthonormal by computing $E^{\mathrm{T}}E$. ■

```
>> E = modgrsch(V)
E =
      0.30151      0.60386     -0.73786
     -0.30151      0.79455      0.52705
```

Algorithm 14.2 Modified Gram-Schmidt

```
function MODGRSCH(V)
  % Modified Gram-Schmidt process for converting a set of linearly independent vectors to a
set
  % of orthonormal vectors spanning the same subspace
  % Input: An m × n matrix V whose columns are the vectors to be normalized.
  % Output: An m × n matrix E whose columns are an orthonormal set of
  % vectors spanning the same subspace as the columns of V
  for i = 1:n do
    E(:,i) = V(:,i)
    for j = 1:i-1 do
      E(:, i) = E(:, i) − E(:, j)^T E(:, i) E(:, j)
    end for
    E(:, i) = E(:, i) / ||E(:, i)||_2
  end for
  return E
end function
```

```
      0.90453     0.063564      0.42164
>> E'*E
ans =
            1 -1.3878e-017  -4.996e-016
 -1.3878e-017            1  8.6736e-016
  -4.996e-016  8.6736e-016             1
```

The difference in results between the classical and MGS methods can be startling. For Example 14.3, the author is indebted to an example found on the MIT OpenCourseWare site for the course [44] 18.335J, Introduction to Numerical Methods.

Example 14.3. Choose $\epsilon = 10^{-8}$, and form the 4×3 matrix $A = \begin{bmatrix} 1 & 1 & 1 \\ \epsilon & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{bmatrix}$. Apply the classical and MGS methods to A and test for the orthogonality of columns.

```
>> epsilon = 1.0e-8;
>> A = [1 1 1;epsilon 0 0;0 epsilon 0;0 0 epsilon];
>> E1 = clgrsch(A);
>> E2 = modgrsch(A);
>> E1(:,2)'*E1(:,3)
ans =
   0.500000000000000
>> E2(:,2)'*E2(:,3)
ans =
    1.1102e-016
```

The huge difference between the results is caused by the fact that the columns are almost equal, and cancellation errors occur when using the classical method. ■

The concept of rank is very important in linear algebra. The best situation is when an $m \times n$ matrix has full rank.

Definition 14.2. The $m \times n$ real matrix is said to have *full rank* if $rank = \min(m, n)$. If $A \in R^{m\times n}$ and $m > n$ then to have full rank the n columns of A must be linearly independent. If $m < n$, then to be of full rank, the m rows must be linearly independent.

If $m > n$, the application of the Gram-Schmidt process to the column vectors of an $m \times n$ full rank matrix A while recording the values r_{ij} yields the QR decomposition, one of the major achievements in linear algebra. The QR decomposition is very important in the accurate, efficient, computation of eigenvalues and is very useful in least-squares problems.

14.3 THE *QR* DECOMPOSITION

Assume $A = \begin{bmatrix} v_1 & v_2 & \dots & v_{n-1} & v_n \end{bmatrix}$ is an $m \times n$ matrix with columns $v_1, v_2, \dots, v_n$. The Gram-Schmidt process can be used to factor A into a product $A = QR$, where $Q^{m\times n}$ has orthonormal columns, and $R^{n\times n}$ is an upper-triangular matrix. The decomposition comes directly from the Gram-Schmidt process by using the r_{ij} values we defined in the description of Gram-Schmidt. Arrange Equation 14.1 so the v_i are on the left-hand side.

Equation 1: $v_1 = e_1 r_{11}$

Equation 2: $v_2 = u_2 + r_{12}e_1$. Note that $e_2 = u_2/\|u_2\|_2$, so $u_2 = e_2\,\|u_2\|_2 = e_2\, r_{22}$, and we have

$$v_2 = e_1 r_{12} + e_2 r_{22.}$$

Equation 3: $v_3 = u_3 + r_{13}e_1 + r_{23}e_2$. We have $e_3 = u_3/\|u_3\|_2$, so $u_3 = r_{33}\, e_3$, and then

$$v_3 = e_1 r_{13} + e_2 r_{23} + e_3 r_{33}.$$

The general formula for the v_k is

$$v_k = \sum_{j=1}^{k-1} r_{jk} e_j + e_k r_{kk}, \quad k = 1, 2, \dots, n. \tag{14.2}$$

Let $Q = \begin{bmatrix} e_1 & e_2 & \dots & e_{n-1} & e_n \end{bmatrix}$, the matrix whose columns are the orthonormal vectors

$$e_1 = \begin{bmatrix} e_{11} \\ e_{21} \\ \vdots \\ e_{m-1,1} \\ e_{m1} \end{bmatrix}, e_2 = \begin{bmatrix} e_{12} \\ e_{22} \\ \vdots \\ e_{m-1,2} \\ e_{m2} \end{bmatrix}, \dots, e_n = \begin{bmatrix} e_{1n} \\ e_{2n} \\ \vdots \\ e_{m-1,n} \\ e_{mn} \end{bmatrix}$$

$$\dots$$

and

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & \dots & r_{1n} \\ 0 & r_{22} & r_{23} & r_{24} & \dots & r_{2n} \\ 0 & 0 & r_{33} & r_{34} & \dots & r_{3n} \\ 0 & 0 & 0 & \ddots & \dots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & r_{nn} \end{bmatrix}.$$

It is sufficient to show that $A = QR$ for a general 3×3 matrix. Let A be the 3×3 matrix

$$A = \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix},$$

and

$$v_1 = \begin{bmatrix} v_{11} \\ v_{21} \\ v_{31} \end{bmatrix}, \quad v_2 = \begin{bmatrix} v_{12} \\ v_{22} \\ v_{32} \end{bmatrix}, \quad v_3 = \begin{bmatrix} v_{13} \\ v_{23} \\ v_{33} \end{bmatrix}.$$

Then,

$$\begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix} = \begin{bmatrix} r_{11}e_{11} & r_{12}e_{11} + r_{22}e_{12} & r_{13}e_{11} + r_{23}e_{12} + r_{33}e_{13} \\ r_{11}e_{21} & r_{12}e_{21} + r_{22}e_{22} & r_{13}e_{21} + r_{23}e_{22} + r_{33}e_{23} \\ r_{11}e_{31} & r_{12}e_{31} + r_{22}e_{32} & r_{13}e_{31} + r_{23}e_{32} + r_{33}e_{33} \end{bmatrix}. \tag{14.3}$$

From Equation 14.2,

$$v_1 = r_{11}e_1, \quad v_2 = r_{12}e_1 + r_{22}e_2, \quad v_3 = r_{13}e_1 + r_{23}e_2 + r_{33}e_3. \tag{14.4}$$

Comparing Equations 14.3 and 14.4, we see that

$$A = QR,$$

where

$$Q = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}, \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{bmatrix}.$$

Theorem 14.1. *(**QR Decomposition**) If A is a full rank $m \times n$ matrix, $m \geq n$, then there exists an $m \times n$ matrix Q with orthonormal columns and an $n \times n$ upper-triangular matrix R such that $A = QR$.*

Remark 14.4. The decomposition is unique (Problem 14.10).

The MATLAB function `clqrgrsch` in the software distribution executes the classical Gram-Schmidt *QR* decomposition process. The application of the classical Gram-Schmidt process for finding the *QR* decomposition has the same problems as the classical Gram-Schmidt process for finding an orthonormal basis. Algorithm 14.3 specifies the MGS process to find the *QR* decomposition. The implementation is that of the MGS process with the added maintenance of the r_{ij}, $1 \leq i, j \leq n$, $j \geq i$, $r_{ij} = 0, j < i$.

Algorithm 14.3 Modified Gram-Schmidt *QR* Decomposition

```
function MODQRGRSCH(A)
   % Input: m × n matrix A.
   % Output: the QR decomposition A = QR, where
   % Q is an m × n matrix with orthonormal columns, and
   % R is an n × n upper-triangular matrix.
   for i = 1:n do
      Q(:, i) = A(:, i)
      for j = 1:i-1 do
         R(j, i) = Q(:, j)^T Q(:, i)
         Q(:, i) = Q(:, i) − R(j, i) Q(:, j)
      end for
      R(i, i) = ‖Q(:, i)‖
      Q(:, i) = Q(:, i) / R(i, i)
   end for
   return [Q, R]
end function
```

NLALIB: The function `modqrgrsch` implements Algorithm 14.3.

Example 14.4. Use MGS to compute the QR decomposition of the matrix

$$A = \begin{bmatrix} 1 & 6 & -1 & 4 & 7 \\ -7 & 0 & 12 & -8 & 2 \\ 14 & 4 & 5 & 3 & 35 \end{bmatrix}.$$

Check the results by computing $\|A - QR\|_2$.

```
>> [Q R] = modqrgrsch(A)

Q =
    0.0638    0.9531   -0.2960   -0.3841    0.2643
   -0.4463    0.2925    0.8457    0.5121   -0.7362
    0.8926    0.0782    0.4440    0.7682    0.6230

R =

   15.6844    3.9530   -0.9564    6.5033   30.7950
         0    6.0311    2.9481    1.7066    9.9929
         0         0   12.6647   -6.6178   15.1595
         0         0         0    0.0000    0.0000
         0         0         0         0    0.0000

>> norm(A - Q*R)

ans =
  9.9301e-016
```

■

Remark 14.5. The Gram-Schmidt QR algorithm produces an $m \times n$ matrix Q and an $n \times n$ matrix R. This is termed the *reduced QR decomposition*. MATLAB has a function `qr` that computes the QR decomposition of an $m \times n$ matrix. Normally, `qr` factors A into the product of an $m \times m$ orthogonal matrix Q and an $m \times n$ upper-triangular matrix R, called the *full QR decomposition*, and we will discuss this decomposition in Chapter 17. This decomposition gives useful information not provided by the reduced version. The term "reduced" derives from the fact that if $m > n$, the resulting matrices are smaller in the reduced decomposition. The MATLAB statement `[Q R] = qr(A,0)` gives the reduced QR decomposition.

14.3.1 Efficiency

We will determine the flop count for MGS. This analysis is somewhat more involved than earlier ones, so it is suggested that the reader refer to the steps in Algorithm 14.3 while reading the analysis.

The outer loop executes n times ($i = 1 : n$), and for each execution of the outer loop, an inner loop executes $i - 1$ times. Let's count the number of flops in the inner loop.

```
i = 1: inner loop executes (n-1) times
i = 2: inner loop executes (n-2) times
...
i = n-1: inner loop executes 1 time
i = n: inner loop does not execute
```

The statements in the inner loop thus execute $(n-1) + (n-2) + \cdots + 1 = n(n-1)/2$ times. The inner loop computes an inner product, requiring $2m$ flops, followed by an expression that performs a scalar multiplication and a vector subtraction. This expression requires $2m$ flops. After the inner loop, a 2-norm is computed, which costs $2m$ flops. Following the computation of the 2-norm, a vector is normalized, requiring m divisions. Now add this all up:

$$\text{Inner loop: } \tfrac{n(n-1)}{2}\,(2m+2m) = \tfrac{n(n-1)}{2}\,(4m)$$
$$\text{Statements after the inner loop: } n\,(2m+m) = n\,(3m) \text{ flops}$$
$$\text{TOTAL: } n\,(3m) + \tfrac{n(n-1)}{2}\,(4m) = 2mn^2 + mn \text{ flops.}$$

Assume the term $2mn^2$ will dominate mn, and we obtain an approximation of $2mn^2$ flops.

14.3.2 Stability

Before we can discuss the stability of the Gram-Schmidt QR decomposition, we need a definition of for the condition number of a general an $m \times n$ matrix. For a nonsingular square matrix, we know that

$$\kappa_2(A) = \|A\|_2 \left\|A^{-1}\right\|_2 = \frac{\sigma_1}{\sigma_n},$$

where σ_1 and σ_n are the largest and smallest singular values of A. A non-square matrix has no inverse, but it does have a largest and a smallest nonzero singular value, leading to the following definition.

Definition 14.3. If A is an $m \times n$ matrix, the 2-norm condition number of A is

$$\kappa_2(A) = \frac{\sigma_1}{\sigma_r},$$

where σ_1 and σ_r are the largest and smallest nonzero singular values of A.

The Gram-Schmidt process for computing the reduced QR decomposition is conceptually simpler than methods we will discuss in Chapter 17, but it is not as stable as those methods. The stability of the algorithm is linked to the condition number of the matrix, as we see in Theorem 14.2, whose proof can be found in Ref. [16, pp. 372-373]. This is not the case with other algorithms we will discuss for computing the QR decomposition. In reading the theorem, note that if Q is an $m \times n$ matrix with orthonormal columns, $m > n$, $Q^{\mathrm{T}}Q$ is the $n \times n$ identity matrix (Problem 14.4).

Theorem 14.2. *Suppose the MGS process is applied to the $m \times n$ matrix $A = \begin{bmatrix} a_1 & a_2 & \dots & a_{n-1} & a_n \end{bmatrix}$ having rank n, yielding an $m \times n$ matrix $\hat{Q}$ and $n \times n$ matrix $\hat{R}$. Then there are constants c_i, $i = 1, 2, 3$, depending on m and n, such that*

$$\begin{aligned} A + \Delta A_1 &= \hat{Q}\hat{R}, \quad \|\Delta A_1\|_2 \le c_1 \,\text{eps}\, \|A\|_2, \\ \left\|\hat{Q}^{\mathrm{T}}\hat{Q} - I\right\|_2 &\le c_2 \,\text{eps}\, \kappa_2(A) + O\left(\left(\text{eps}\,\kappa_2(A)\right)^2\right), \end{aligned}$$

and there exists a matrix Q with orthonormal columns such that

$$A + \Delta A_2 = Q\hat{R}, \quad \|\Delta A_2(:, j)\|_2 \le c_3 \,\text{eps}\, \left\|a_j\right\|_2, \quad 1 \le j \le n.$$

There are three observations we can obtain from Theorem 14.2.

a. The residual $A - \hat{Q}\hat{R}$ is small.
b. How close $\hat{Q}$ is to having orthonormal columns depends on the condition number of A. If A is well conditioned, then $\hat{Q}$ has close to orthonormal columns.
c. $\hat{R}$ is the exact upper-triangular factor of a matrix near to A.

14.4 APPLICATIONS OF THE *QR* DECOMPOSITION

There are numerous applications for the QR decomposition. Chapters 18 and 19 discuss the use of the QR decomposition to find eigenvalues and eigenvectors, a problem we know should never be done by finding the roots of the characteristic polynomial. We discussed polynomial fitting using least squares in Section 12.3. The general linear least-squares problem involves finding a vector $x \in \mathbb{R}^n$ that minimizes $\|Ax - b\|_2$, where A is an $m \times n$ matrix, and b an $m \times 1$ vector. The QR decomposition can be effectively used to solve such problems, as you will see in Chapter 16. In this section, we discuss two useful but simpler applications, computing the determinant, and finding the range of a matrix.

14.4.1 Computing the Determinant

We can use the QR decomposition to find the absolute value of the determinant of a square matrix A. We will assume that the columns of A are linearly independent. Since Q has orthonormal columns,

$$Q^{\mathrm{T}}Q = I, \text{ and } \det\left(Q^{\mathrm{T}}Q\right) = \det\left(Q^{\mathrm{T}}\right)\det(Q) = (\det(Q))^2 = 1\,,$$

so $|\det(Q)| = 1$. It follows that

$$|\det(A)| = |\det(QR)| = |\det(Q)|\,|\det(R)| = |r_{11}r_{22}r_{33}\ldots r_{nn}|\,,$$

since the determinant of an upper-triangular matrix is the product of its diagonal elements.

Example 14.5. Find the absolute value of the determinant of the matrix $A = \begin{bmatrix} 8.0 & 2.6 & 4.0 & 9.8 \\ 4.2 & 6.3 & -1.2 & 5.0 \\ -2.0 & 0.0 & 9.1 & 8.5 \\ 18.7 & 25.0 & -1.0 & 23.5 \end{bmatrix}$.

```
>> [Q R] = modqrgrsch(A);
>> abs(prod(diag(R)))

ans =
  519.8238

>> det(A)

ans =
 -519.8238
```

■

14.4.2 Finding an Orthonormal Basis for the Range of a Matrix

Recall that the range, $R(A)$, of an $m \times n$ matrix A is defined by

$$R(A) = \left\{y \in \mathbb{R}^m \mid Ax = y \text{ for some vector } x \in \mathbb{R}^n\right\}.$$

In other words, the range of A is the set of all vectors y for which the equation $Ax = y$ has a solution. If $m \geq n$, A has rank n, and v_i, $1 \leq i \leq n$ are the columns of A, then

$$Ax = x_1v_1 + x_2v_2 + \cdots + x_{n-1}v_{n-1} + x_nv_n,$$

so the v_i are a basis for the range of A. Another way of putting this is that $R(A)$ is the column space of A. Now suppose that $A = QR$ is a reduced decomposition of A, $m \geq n$, and the diagonal entries r_{ii} of R are nonzero. We do not make the assumption that the decomposition was found using the Gram-Schmidt algorithm; in fact, we will discuss two other algorithms for computing the reduced decomposition. It is reasonable to require $r_{ii} \neq 0$, since we know the QR decomposition can be done using Gram-Schmidt with $r_{ii} > 0$. The following theorem connects Q and $R(A)$.

Theorem 14.3. *If A is a full rank $m \times n$ matrix, $m \geq n$, and $A = QR$ is a reduced QR decomposition of A with $r_{ii} \neq 0$, the columns of Q are an orthonormal basis for the range of A.*

Proof. If $x \in \mathbb{R}^n$, then $Ax = Q(Rx)$. Rx is a vector in $\mathbb{R}^n$, so $\text{range}(A) \subseteq \text{range}(Q)$. If we can show that $\text{range}(Q) \subseteq \text{range}(A)$, then $\text{range}(A) = \text{range}(Q)$, and the columns of Q are an orthonormal basis for the range of A. Since R is upper diagonal with nonzero diagonal entries, it is invertible, and $AR^{-1} = Q$. Then, $Qx = A\left(R^{-1}x\right)$, and $\text{range}(Q) \subseteq \text{range}(A)$, completing the proof. □

14.5 CHAPTER SUMMARY

The Gram-Schmidt Process

The Gram-Schmidt process takes a set of k linearly independent vectors, v_i, $1 \leq i \leq k$, and builds an orthonormal basis that spans the same subspace. Compute the projection of vector v onto vector u using

$$\text{proj}_u(v) = \left(\frac{\langle v, u \rangle}{\|u\|_2^2} \right) u.$$

The vector $v - \text{proj}_u(v)$ is orthogonal to u, and this forms the basis for the Gram-Schmidt process. Begin with the first vector, v_1, normalize it, name it e_1, and form $u_2 = e_1 - \text{proj}_{e_1}(v_2)$, and let $e_2 = u_2/\|u\|_2$. Continue this process by subtracting all projections of v_i onto e_j, $1 \leq j \leq i-1$ to obtain a vector u_i orthogonal to the e_j and normalize it to obtain e_i. Continue until you have an orthonormal set $e_1, e_2, \ldots, e_k$. To prepare for constructing the QR decomposition, maintain the upper-triangular matrix entries $r_{ji} = \langle v_i, e_j \rangle$, $1 \leq j \leq i-1$ and $r_{ii} = \|u_i\|_2$.

Numerical Stability of the Gram-Schmidt Process

During the execution of the Gram-Schmidt process, the vectors u_i are often not quite orthogonal, due to rounding errors. For the classical Gram-Schmidt process just described, this loss of orthogonality is particularly bad. The computation also yields poor results when some of the vectors are almost linearly dependent. For these reasons, it is said that the classical Gram-Schmidt process is numerically unstable.

Subtracting the projections of v_i onto the e_j all at once causes the problem. Split the computation into smaller parts by removing the projections one at a time. This approach (referred to as *MGS* process) gives the same result as the original formula in exact arithmetic, but it introduces smaller roundoff errors when executed on a computer.

The *QR* Decomposition

An $m \times n$ real matrix A is said to have full rank if $\text{rank}(A) = \min(m, n)$. If $A \in R^{m \times n}$ and $m > n$, then to have full rank the n columns of A must be linearly independent. If $m < n$, then to be of full rank, the m rows must be linearly independent. If $m \geq n$, the application of the Gram-Schmidt process to the column vectors of an $m \times n$ full rank matrix A while recording the values r_{ij} yields the QR decomposition, $A = QR$, where Q has orthonormal columns and R is an $n \times n$ upper-triangular matrix.

The decomposition requires approximately $2mn^2$ flops, which is better than other methods we will discuss. However, the algorithm is not as stable. Unlike the other methods, how close the computed Q is to having orthonormal columns depends on the condition number of A.

Applications of the *QR* Decomposition

We will see in Chapter 16 that the QR decomposition can be very effectively used to solve linear least-squares problems. In Chapters 18 and 19, we will see that the QR decomposition is very important in the accurate, efficient, computation of eigenvalues. This chapter presents two simpler applications.

The QR decomposition can be used to determine the absolute value of a determinant, namely,

$$|\det A| = |r_{11} r_{22} \ldots r_{nn}|.$$

The matrix R computed by Gram-Schmidt has positive diagonal elements, and Theorem 14.3 tells us that the columns of Q are an orthonormal basis for the range of A.

14.6 PROBLEMS

14.1 The vectors $\begin{bmatrix} 1 & -1 & 5 \end{bmatrix}^T$ and $\begin{bmatrix} -1 & -\frac{3}{5} & \frac{2}{25} \end{bmatrix}^T$ are orthogonal. Divide them by their norms to produce orthonormal vectors e_1 and e_2. Create a matrix Q with e_1 and e_2 as its columns and calculate Q^TQ and QQ^T. Explain the results.

14.2 Compute the projection of v onto u and verify that $v - \text{proj}_u(v)$ is orthogonal to u.

a. $v = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}^T$, $u = \begin{bmatrix} -1 & 3 & -1 & 7 \end{bmatrix}^T$

b. $v = \begin{bmatrix} -1 & 6 & 2 & 0 & 1 \end{bmatrix}^T$, $u = \begin{bmatrix} 0 & 6 & -1 & -1 & 1 \end{bmatrix}^T$

14.3 Find an orthonormal basis e_1, e_2 for the subspace spanned by $v = \begin{bmatrix} 1 & -1 & 2 & 3 \end{bmatrix}^{\mathrm{T}}$ and $w = \begin{bmatrix} 2 & 0 & 1 & 6 \end{bmatrix}^{\mathrm{T}}$.

14.4 If Q is an $m \times n$ matrix with orthonormal columns, prove that $Q^{\mathrm{T}}Q$ is the $n \times n$ identity matrix.

For Problems 14.5 and 14.6, assume that $A = QR$ is the QR decomposition of an $n \times n$ matrix A.

14.5 Show that $A^{\mathrm{T}}A = R^{\mathrm{T}}R$.

14.6 Show that $\det\left(A^{\mathrm{T}}A\right) = (r_{11}r_{22}\ldots r_{nn})^2$.

14.7 **a.** Let $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, with $\det A = ad - bc > 0$. Find the QR decomposition of A.

b. Gram-Schmidt breaks down if the column vectors are linearly dependent. Using the result of part (a), show how the breakdown occurs.

14.8 Assume A is an $m \times n$ matrix, $m < n$, and rank $(A) = m$. prove that A can be written in the form $A = LQ$, where L is a $m \times m$ lower-triangular matrix and Q is an $m \times n$ matrix with orthonormal rows.

14.9 This problem takes another approach to developing the QR decomposition.

a. Show that $A^{\mathrm{T}}A = R^{\mathrm{T}}R$, where R is upper triangular with $r_{ii} > 0$, $1 \le i \le n$.

b. Show that R nonsingular so we can define $Q = AR^{-1}$, and thus $A = QR$.

c. Show that $Q^{\mathrm{T}}Q = I$, so Q has orthonormal columns.

14.10 This problem is a proof that the reduced decomposition is unique.

a. Assume that $A = \hat{Q}\hat{R}$ is another QR decomposition of A. Show that $A^{\mathrm{T}}A = \hat{R}^{\mathrm{T}}\hat{R}$.

b. Using the fact that the Cholesky decomposition of a positive definite matrix is unique, show that $\hat{Q} = Q$ and $\hat{R} = R$.

14.11 Assume that $A = QR$ is the QR decomposition of the $m \times n$ matrix A. In the proof of Theorem 14.3, we showed that $R(A) \subseteq R(Q)$. This problem provides an alternative approach to developing this subset relationship. Let $C = QR$. Show that each element c_{ki}, $1 \le k \le m$, in column i of C has the value

$$c_{ki} = \sum_{p=1}^{i} q_{kp}r_{pi}.$$

Now show that

$$c_i = \sum_{p=1}^{i} q_p r_{pi},$$

where c_i is column i of QR and q_p is column p of Q. Explain why this shows that $R(A) \subseteq R(Q)$.

14.6.1 MATLAB Problems

14.12

a. Show that the vectors $\begin{bmatrix} 1 \\ 5 \\ -1 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \\ 8 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix}$ are linearly independent by showing that $\det \begin{bmatrix} 1 & 2 & 0 \\ 5 & 3 & 1 \\ -1 & 8 & 3 \end{bmatrix} \neq 0$

b. Using `modqrgrsch`, find the QR decomposition of $A = \begin{bmatrix} 1 & 2 & 0 \\ 5 & 3 & 1 \\ -1 & 8 & 3 \end{bmatrix}$.

14.13

a. Find an orthonormal basis for the subspace spanned by the columns of $A = \begin{bmatrix} 1 & 4 & 7 \\ -1 & 2 & 3 \\ 9 & 1 & 0 \\ 4 & 1 & 8 \end{bmatrix}$.

b. Find an orthonormal basis for the subspace spanned by the rows of $B = \begin{bmatrix} 5 & -1 & 3 & 6 \\ 0 & 5 & -7 & 1 \\ 1 & 2 & -1 & 0 \end{bmatrix}$.

c. Our implementation of Gram-Schmidt assumes that the columns of the matrix are linearly independent. Will `modgrsch` apply to the columns of B in part (b)? Why or why not?

14.14 Find an orthonormal basis for the column space of A.

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

14.15 Use `modgrsch` to find an orthonormal basis for the columns of the matrix

$$A = \begin{bmatrix} 1 & 9 & 0 & 5 & 3 & 2 \\ -6 & 3 & 8 & 2 & -8 & 0 \\ 3 & 15 & 23 & 2 & 1 & 7 \\ 3 & 57 & 35 & 1 & 7 & 9 \\ 3 & 5 & 6 & 15 & 55 & 2 \\ 33 & 7 & 5 & 3 & 5 & 7 \end{bmatrix}$$

14.16 The QR decomposition using Gram-Schmidt gives a result if rank $(A) \neq \min(m, n)$.

a. Demonstrate this for the matrices

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

ii. $B = \begin{bmatrix} 1 & 8 & -1 & 3 & 2 \\ 5 & 7 & -9 & 1 & 4 \\ 13 & 71 & -17 & 25 & 20 \end{bmatrix}$

b. Explain why the results of part (a) are actually not QR decompositions.

c. Use the MATLAB function `qr` with the two matrices in part (a). Are these actually QR decompositions? Why?

14.17 Compute the absolute value of the determinant of the matrix in Problem 14.15 using `modqrgrsch`. Compare the computed value to that obtained from the MATLAB function `det`.

14.18 The QR decomposition can be used to solve a linear system. Let A be an $n \times n$ matrix, with $A = QR$. Then, the linear system $Ax = b$ can be written as

$$QRx = b.$$

The process goes as follows:
Solve $Qy = b$ for y.
Solve $Rx = y$ for x.

a. It is very easy to solve for y without using Gaussian elimination. Why?

b. The solution to $Rx = y$ can be done quickly why?

c. Develop a function `qrsolve` that solves an $n \times n$ system using the MATLAB reduced QR decomposition. If the matrix is singular, the QR decomposition will complete with no error. Make the return values `[x, resid]`, where resid is the residual $\|Ax - b\|_2$. A large residual will indicate that the matrix is singular or ill-conditioned.

d. Apply the function to solve each system.

i. $\begin{bmatrix} 1 & -1 & 0 \\ 2 & 4 & 5 \\ -7 & 1 & 3 \end{bmatrix} x = \begin{bmatrix} 1 \\ -1 \\ 8 \end{bmatrix}$

ii. $\begin{bmatrix} 21 & 3 & -4 & 8 \\ 1 & 3 & 59 & 0 \\ 1 & 2 & -22 & 35 \\ 3 & 78 & 100 & 3 \end{bmatrix} x = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 2 \end{bmatrix}$

iii. The rosser matrix `A = rosser`.

We have three methods available for computing the *QR* decomposition of a matrix:

a. `[Q, R] = clqrgrsch(A);`
b. `[Q, R] = modqrgrsch(A);`
c. `[Q, R] = qr(A);`

Problems 14.19 and 14.20 involve running numerical experiments using these functions.

14.19 A Hilbert matrix is square and has entries $H(i,j) = 1/(i+j-1)$, $1 \leq i,j \leq n$. Hilbert matrices are notoriously ill-conditioned. An interesting fact about a Hilbert matrix is that every Hilbert matrix has an inverse consisting entirely of large integers. For $n \leq 15$, the MATLAB command `H = invhilb(n)` returns the exact inverse of the $n \times n$ Hilbert matrix. Use methods 1-3 for *QR* decomposition to produce Q_1R_1, Q_2R_2, and Q_3R_3 for the inverse Hilbert matrix of order 12. Each Q_i should be orthogonal. Test this by computing $\|Q_i\|_2$ and $\left\|Q_i^{\mathrm{T}}Q_i - I\right\|$ for each method. Explain your results.

14.20 Let $A = \begin{bmatrix} -1 & 2 & 7 \\ -1 & 2 & 5 \\ 1 & -1 & 3 \end{bmatrix}$, and perform the following numerical experiment.

```
>> for i = 1:10
      [Q R] = modqrgrsch(A);
      A = R*Q;
   end
```

Examine R, and determine what it tells you about the original matrix A.

14.21 This problem tests the performance of MGS. Recall that Theorem 14.2 links the performance of MGS to the condition number of A.

a. Let A be the Rosser matrix by executing `A = rosser`. A is an 8×8 symmetric matrix with some very bad properties.
b. Find the condition number of `A`.
c. The function `hqr` in the software distribution computes the *QR* decomposition of an $m \times n$ matrix using what are termed Householder reflections. We will develop `hqr` in Chapter 17. Execute the following statements

```
>> [Q1, R1] = modqrgrsch(A);
>> [Q2, R2] = hqr(A);
>> norm(Q1'*Q1 - eye(8))
>> norm(Q2'*Q2 - eye(8))
```

and comment on the results.

d. We defined the Vandermonde matrix in Section 12.3, and it is implemented by the function `vandermonde` in the software distribution. Given a vector $x \in \mathbb{R}^n$ and an integer m, the Vandermonde matrix is of dimension $m \times (n+1)$. Normally, m is much larger than n. Explain the output of the following code:

```
>> x = rand(100,1);
>> V = vandermonde(x,25);
>> [Q1 R1] = modqrgrsch(V);
>> [Q2 R2] = qr(V,0);
>> norm(Q1'*Q1 - eye(26))
>> norm(Q2'*Q2 - eye(26))
```

14.22 Determining the rank of a matrix is a difficult problem, as will become evident in subsequent chapters. The *QR* decomposition can be used to determine the exact or approximate rank of a matrix. Column pivoting can be used in performing the *QR* decomposition, and the process is termed *rank revealing QR decomposition* [2, pp. 248-250]. A discussion of this method is beyond the scope of this book, but it is useful to know that it exists and to experiment with it. In MATLAB, there is a version of the function `qr` that has the calling sequence

```
[Q,R,E] = qr(A,0)
```

It produces a matrix `Q` with orthonormal columns, upper triangular `R`, and a permutation matrix `E` so that AE = QR. The rank of A is determined by counting the number of nonzero values on the diagonal of R. What is a nonzero value? Since roundoff error occurs, it is likely that an entry that should be exactly zero is small instead. We will assign a tolerance, tol, and any value less than or equal to tol will be considered zero. Portions of the following questions are taken from Ref. [23, Exercise 4.2.21, pp. 271-272].

a. The Kahan matrix $K_n(\theta)$ is an $n \times n$ upper triangular matrix whose entries are a function of θ. It provides an interesting problem in rank determination. Generate the 90×90 Kahan matrix with $\theta = 1.2$ using the MATLAB statement

```
K = gallery('kahan',90,1.2,0);
```

b. Enter

```
>> [Q R E] = qr(K,0);
>> tol = 1.0e-14;
>> rdiag = diag(R);
>> sum(rdiag > tol);
>> rank(K)
```

Is rank determination using the *QR* decomposition correct?

c. To ensure that the *QR* factorization with column pivoting does not interchange columns in the presence of rounding errors, the diagonal is perturbed by `pert*eps*diag([n:-1:1])`. Obtain a slightly perturbed version of K using the statement

```
>> K25 = gallery('kahan',90,1.2,25);
```

Enter the same sequence of statements as in part (b), replacing K by K25. Does the *QR* decomposition with column pivoting give the correct rank?

14.23 Example 14.3 provided evidence that the classical Gram-Schmidt method was error-prone. In this problem, you will repeat an expanded version of Example 14.3. Define $\epsilon = 0.5 \times 10^{-7}$ and build the matrix

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

Execute the following MATLAB command sequence. The function `givensqr` is in the software distribution and will be developed in Chapter 17.

```
>> [Q1 R1] = clqrgrsch(A);
>> [Q2 R2] = modqrgrsch(A);
>> [Q3 R3] = qr(A);
>> [Q4 R4] = givensqr(A);
```

For each decomposition, compute $\left\| Q_i^{\mathrm{T}} Q_i - I \right\|$. Comment on the results.

14.24 This problem investigates what happens when the matrix `A` given as an argument to `modqrgrsch` is complex. The two primary components in the implementation of MGS are the inner product $\langle u, v \rangle$ and the 2-norm $\|u\|_2$. The remaining operations are arithmetic.

a. Enter the complex vectors

$$x = \begin{bmatrix} 1-i & 2+i & 3-2i \end{bmatrix}^{\mathrm{T}}, \quad y = \begin{bmatrix} 8+i & i & 5+4i \end{bmatrix}^{\mathrm{T}}$$

and compute the expression $\sum_{i=1}^{3} x_i y_i$. MATLAB will compute the inner product using the function `dot(x,y)`. Use `dot` with x and y. Do you obtain the same results?

b. Compute $\sum_{i=1}^{3} \overline{x_i} y_i$, where $\overline{x_i}$ is the complex conjugate of x_i. Use the MATLAB function `conj` to compute the complex conjugate. Does your result agree with `dot(x,y)`?

c. Based upon the your answers in parts (a) and (b), what is the definition for the inner product of two vectors $x, y \in \mathbb{C}^n$?

d. Define the norm of a vector $x \in \mathbb{C}^n$. Use your definition to find the norm of x and compare the result with the MATLAB statement `norm(x)`.

e. Run `modqrgrsch` with a 3×3 and a 5×3 complex matrix of your choosing. In each case, compare your result with that of the reduced *QR* decomposition computed using MATLAB's `qr`. If the results are very close, explain why, and if not explain why `modqrgrsch` fails with complex matrices.

14.25 We will see in Chapters 21 and 22 that there are important algorithms which are subject to problems when the vectors they are generating begin to lose orthogonality. A standard solution is to perform *reorthogonalization*. At

point i in the algorithm, assume it has generated vectors $v_1, v_2, \ldots, v_i$ that should be orthogonal if exact arithmetic were used. Reorthogonalization performs operations on the vectors to improve orthogonality. This can be done with both the classical and MGS methods. In Ref. [45, p. 1071], the classical and MGS methods are discussed with and without reorthogonalization. Also see Ref. [23, pp. 231-233]. In this problem, we will only discuss the classical Gram-Schmidt method. The following listing is from the classical Gram-Schmidt method modified to compute the QR decomposition.

```
1:  for i = 1:n do
2:    Q(:,i) = A(:,i)
3:    sumproj = 0
4:    for j = 1:i-1 do
5:      R(j, i) = Q(:, j)^T Q(:, i)
6:      sumproj = sumproj + R(j, i) Q(:, j)
7:    end for
8:    Q(:, i) = Q(:, i) − sumproj
9:    R(i, i) = ‖Q(:, i)‖_2
10:    Q(:, i) = Q(:, i)/R(i, i)
11:  end for
```

After steps 3-8, in exact arithmetic $Q(:, i)$ is orthogonal to $Q(:, 1 : i)$; however with roundoff and cancellation error present, it likely is not. We need to start with the already computed $Q(:, i)$ and repeat steps 3-8, hopefully improving orthogonality. The inner loop computed the terms $R(j, i) = Q(:, j)^{\mathrm{T}} Q(:, i)$. When starting with $Q(:, i)$ and repeating steps 3-8, the values of $Q(:, j)^{\mathrm{T}} Q(:, i)$ will be small and serve as corrections to the original values of R. They must be assigned to another matrix, S, and then added to R. The following statements perform reorthogonalization when placed between statements 8 and 9.

```
sumproj = 0
for j = 1:i-1 do
   S(j, i) = Q(:, j)^T Q(:, i)
   sumproj = sumproj + S(j, i) Q(:, j)
end for
Q(:, i) = Q(:, i) − sumproj
R(1 : i − 1, i) = R(1 : i − 1, i) + S(1 : i − 1, i)
```

a. Modify clqrgrsch so it selectively performs reorthogonalization. Its calling format should be `[Q,R] = clqrgrsch(A,reorthog)`. When reorthog = 1, perform reorthogonalization; otherwise, don't. If the argument is omitted, the default value should be 1. If you don't know how to deal with variable input arguments, use MATLAB help for `nargin`.

b. The matrix `west0167` in the software distribution has dimension 167×167 and a condition number of 4.7852×10^{10}. It was involved in a chemical process simulation problem. Apply clqrgrsch to west0167 with and without reorthogonalization. In each case, compute $\|Q\|_2$ and $\left\|Q^{\mathrm{T}}Q - I\right\|_2$. Explain the results.

c. Modify your `clqrgrsch` so it returns an array of values orthog $= \left\|Q(:, 1 : i)^{\mathrm{T}} Q(:, 1 : i) - I^{i\times i}\right\|_2$, $1 \le i \le n$, in addition to Q and R. The array records how well orthogonality is maintained. Take the function `modqrgrsch` in the software distribution and do the same. Apply classical Gram-Schmidt, with and without reorthogonalization, and `modqrgrsch` to `west0167`. On the same set of axes, draw a `semilogy` graph of $m = 1, 2, \ldots, 167$ against `orthog` for each algorithm. Comment on the results.