Chapter 5

Orthogonalization and Least Squares

This chapter is primarily concerned with the least squares solution of overdetermined systems of equations, i.e., the minimization of $\| Ax - b \|_2$ where $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ and $b \in \mathbb{R}^m$. The most reliable solution procedures for this problem involve the reduction of A to various canonical forms via orthogonal transformations. Householder reflections and Givens rotations are central to this process and we begin the chapter with a discussion of these important transformations. In §5.2 we discuss the computation of the factorization $A = QR$ where Q is orthogonal and R is upper triangular. This amounts to finding an orthonormal basis for ran(A). The QR factorization can be used to solve the full rank least squares problem as we show in §5.3. The technique is compared with the method of normal equations after a perturbation theory is developed. In §5.4 and §5.5 we consider methods for handling the difficult situation when A is rank deficient (or nearly so). QR with column pivoting and the SVD are featured. In §5.6 we discuss several steps that can be taken to improve the quality of a computed least squares solution. Some remarks about square and underdetermined systems are offered in §5.7.

Before You Begin

Chapters 1, 2, and 3 and §§4.1-4.3 are assumed. Within this chapter there are the following dependencies:

$$\begin{tikzcd}
 & & & & \S5.6 \\
\S5.1 \arrow[r] & \S5.2 \arrow[r] & \S5.3 \arrow[r] & \S5.4 \arrow[r] & \S5.5 \arrow[u] \arrow[d] \\
 & & & & \S5.7
\end{tikzcd}$$

Complementary references include Lawson and Hanson (1974), Farebrother (1987), and Björck (1996). See also Stewart (1973), , Hager (1988), Stewart and Sun (1990), Watkins (1991), Gill, Murray, and Wright (1991), Higham (1996), Trefethen and Bau (1996), and Demmel (1996). Some MATLAB functions important to this chapter are qr, svd, pinv, orth, rank, and the "backslash" operator "\." LAPACK connections include

LAPACK: Householder/Givens Tools	
_LARFG	Generates a Householder matrix
_LARF	Householder times matrix
_LARFX	Small n Householder times matrix
_LARFB	Block Householder times matrix
_LARFT	Computes $I - VTV^H$ block reflector representation
_LARTG	Generates a plane rotation
_LARGV	Generates a vector of plane rotations
_LARTV	Applies a vector of plane rotations to a vector pair
_LASR	Applies rotation sequence to a matrix
CSROT	Real rotation times complex vector pair
CROT	Complex rotation (c real) times complex vector pair
CLACGV	Complex rotation (s real) times complex vector pair

LAPACK: Orthogonal Factorizations	
_GEQRF	$A = QR$
_GEQPF	$A\Pi = QR$
_ORMQR	Q (factored form) times matrix (real case)
_UNMQR	Q (factored form) times matrix (complex case)
_ORGQR	Generates Q (real case)
_UNGQR	Generates Q (complex case)
_GERQF	$A = RQ =$ (upper triangular)(orthogonal)
_GELQF	$A = QL =$ (orthogonal)(lower triangular)
_GEQLF	$A = LQ =$ (lower triangular)(orthogonal)
_TZRQF	$A = RQ$ where A is upper trapezoidal
_GESVD	$A = U\Sigma V^T$
_BDSQR	SVD of real bidiagonal matrix
_GEBRD	Bidiagonalization of general matrix
_ORGBR	Generates the orthogonal transformations
_GBBRD	Bidiagonalization of band matrix

LAPACK: Least Squares	
_GELS	Full rank min $\\| AX - B \\|_{2F}$ or min $\\| A^H X - B \\|_F$
_GELSS	SVD solution to min $\\| AX - B \\|_F$
_GELSX	Complete orthogonal decomposition solution to min $\\| AX - B \\|_F$.
_GEEQU	Equilibrates general matrix to reduce condition

5.1 Householder and Givens Matrices

Recall that $Q \in \mathbb{R}^{n\times n}$ is *orthogonal* if $Q^TQ = QQ^T = I_n$. Orthogonal matrices have an important role to play in least squares and eigenvalue computations. In this section we introduce the key players in this game: Householder reflections and Givens rotations.

5.1.1 A 2-by-2 Preview

It is instructive to examine the geometry associated with rotations and reflections at the $n = 2$ level. A 2-by-2 orthogonal matrix Q is a *rotation* if it has the form

$$Q = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}.$$

If $y = Q^Tx$, then y is obtained by rotating x counterclockwise through an angle θ.

A 2-by-2 orthogonal matrix Q is a *reflection* if it has the form

$$Q = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{bmatrix}.$$

If $y = Q^Tx = Qx$, then y is obtained by reflecting the vector x across the line defined by

$$S = \text{span}\left\{ \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2) \end{bmatrix} \right\}.$$

Reflections and rotations are computationally attractive because they are easily constructed and because they can be used to introduce zeros in a vector by properly choosing the rotation angle or the reflection plane.

Example 5.1.1 Suppose $x = [\,1,\ \sqrt{3}\,]^T$. If we set

$$Q = \begin{bmatrix} \cos(-60^\circ) & \sin(-60^\circ) \\ -\sin(-60^\circ) & \cos(-60^\circ) \end{bmatrix} = \begin{bmatrix} 1/2 & -\sqrt{3}/2 \\ \sqrt{3}/2 & 1/2 \end{bmatrix}$$

then $Q^Tx = [\,2,\ 0\,]^T$. Thus, a rotation of -60° zeros the second component of x. If

$$Q = \begin{bmatrix} \cos(30^\circ) & \sin(30^\circ) \\ \sin(30^\circ) & -\cos(30^\circ) \end{bmatrix} = \begin{bmatrix} \sqrt{3}/2 & 1/2 \\ 1/2 & -\sqrt{3}/2 \end{bmatrix}$$

then $Q^Tx = [\,2,\ 0\,]^T$. Thus, by reflecting x across the 30° line we can zero its second component.

5.1.2 Householder Reflections

Let $v \in \mathbb{R}^n$ be nonzero. An n-by-n matrix P of the form

$$P = I - \frac{2}{v^T v} v v^T \qquad (5.1.1)$$

is called a *Householder reflection.* (Synonyms: Householder matrix, Householder transformation.) The vector v is called a *Householder vector.* If a vector x is multiplied by P, then it is reflected in the hyperplane $\text{span}\{v\}^{\perp}$. It is easy to verify that Householder matrices are symmetric and orthogonal.

Householder reflections are similar in two ways to Gauss transformations, which we introduced in §3.2.1. They are rank-1 modifications of the identity and they can be used to zero selected components of a vector. In particular, suppose we are given $0 \neq x \in \mathbb{R}^n$ and want Px to be a multiple of $e_1 = I_n(:,1)$. Note that

$$Px = \left(I - \frac{2vv^T}{v^Tv}\right)x = x - \frac{2v^Tx}{v^Tv}v$$

and $Px \in \text{span}\{e_1\}$ imply $v \in \text{span}\{x, e_1\}$. Setting $v = x + \alpha e_1$ gives

$$v^Tx = x^Tx + \alpha x_1$$

and

$$v^Tv = x^Tx + 2\alpha x_1 + \alpha^2,$$

and therefore

$$Px = \left(1 - 2\frac{x^Tx + \alpha x_1}{x^Tx + 2\alpha x_1 + \alpha^2}\right)x - 2\alpha\frac{v^Tx}{v^Tv}e_1 .$$

In order for the coefficient of x to be zero, we set $\alpha = \pm\| x \|_2$ for then

$$v = x \pm \| x \|_2 e_1 \Rightarrow Px = \left(I - 2\frac{vv^T}{v^Tv}\right)x = \mp\| x \|_2 e_1. \qquad (5.1.2)$$

It is this simple determination of v that makes the Householder reflection so useful.

Example 5.1.2 If $x = [3,\ 1,\ 5,\ 1]^T$ and $v = [9,\ 1,\ 5,\ 1]^T$, then

$$P = I - 2\frac{vv^T}{v^Tv} = \frac{1}{54}\begin{bmatrix} -27 & -9 & -45 & -9 \\ -9 & 53 & -5 & -1 \\ -45 & -5 & 29 & -5 \\ -9 & -1 & -5 & 53 \end{bmatrix}$$

has the property that $Px = [-6,\ 0,\ 0,\ 0,\ ]^T$.

5.1.3 Computing the Householder Vector

There are a number of important practical details associated with the determination of a Householder matrix, i.e., the determination of a Householder vector. One concerns the choice of sign in the definition of v in (5.1.2). Setting

$$v_1 = x_1 - \| x \|_2$$

has the nice property that Px is a positive multiple of e_1. But this recipe is dangerous if x is close to a positive multiple of e_1 because severe cancellation would occur. However, the formula

$$v_1 = x_1 - \| x \|_2 = \frac{x_1^2 - \| x \|_2^2}{x_1 + \| x \|_2} = \frac{-(x_2^2 + \cdots + x_n^2)}{x_1 + \| x \|_2}$$

suggested by Parlett (1971) does not suffer from this defect in the $x_1 > 0$ case.

In practice, it is handy to normalize the Householder vector so that $v(1) = 1$. This permits the storage of $v(2{:}n)$ where the zeros have been introduced in x, i.e., $x(2{:}n)$. We refer to $v(2{:}n)$ as the *essential part* of the Householder vector. Recalling that $\beta = 2/v^Tv$ and letting **length**(x) specify vector dimension, we obtain the following encapsulation:

Algorithm 5.1.1 (Householder Vector) Given $x \in \mathbb{R}^n$, this function computes $v \in \mathbb{R}^n$ with $v(1) = 1$ and $\beta \in \mathbb{R}$ such that $P = I_n - \beta vv^T$ is orthogonal and $Px = \| x \|_2 e_1$.

```
function: $[v, \beta] = \mathbf{house}(x)$
    $n = \mathbf{length}(x)$
    $\sigma = x(2{:}n)^T x(2{:}n)$
    $v = \begin{bmatrix} 1 \\ x(2{:}n) \end{bmatrix}$
    if $\sigma = 0$
        $\beta = 0$
    else
        $\mu = \sqrt{x(1)^2 + \sigma}$
        if $x(1) <= 0$
            $v(1) = x(1) - \mu$
        else
            $v(1) = -\sigma/(x(1) + \mu)$
        end
        $\beta = 2v(1)^2/(\sigma + v(1)^2)$
        $v = v/v(1)$
    end
```

This algorithm involves about $3n$ flops and renders a computed Householder matrix that is orthogonal to machine precision, a concept discussed below.

A production version of Algorithm 5.1.1 may involve a preliminary scaling of the x vector ($x \leftarrow x/\| x \|$) to avoid overflow.

5.1.4 Applying Householder Matrices

It is critical to exploit structure when applying a Householder reflection to a matrix. If $A \in \mathbb{R}^{m\times n}$ and $P = I - \beta vv^T \in \mathbb{R}^{m\times m}$, then

$$PA = \left(I - \beta vv^T\right) A = A - vw^T$$

where $w = \beta A^T v$. Likewise, if $P = I - \beta vv^T \in \mathbb{R}^{n\times n}$, then

$$AP = A\left(I - \beta vv^T\right) = A - wv^T$$

where $w = \beta Av$. Thus, an m-by-n Householder update involves a matrix-vector multiplication and an outer product update. It requires $4mn$ flops. Failure to recognize this and to treat P as a general matrix increases work by an order of magnitude. *Householder updates never entail the explicit formation of the Householder matrix.*

Both of the above Householder updates can be implemented in a way that exploits the fact that $v(1) = 1$. This feature can be important in the computation of PA when m is small and in the computation of AP when n is small.

As an example of a Householder matrix update, suppose we want to overwrite $A \in \mathbb{R}^{m\times n}$ ($m \geq n$) with $B = Q^TA$ where Q is an orthogonal matrix chosen so that $B(j+1{:}m, j) = 0$ for some j that satisfies $1 \leq j \leq n$. In addition, suppose $A(j{:}m, 1{:}j-1) = 0$ and that we want to store the essential part of the Householder vector in $A(j+1{:}m, j)$. The following instructions accomplish this task:

```
$[v, \beta]$ = house($A(j{:}m, j)$)
$A(j{:}m, j{:}n) = (I_{m-j+1} - \beta vv^T)A(j{:}m, j{:}n)$
$A(j+1{:}m, j) = v(2{:}m-j+1)$
```

From the computational point of view, we have applied an order $m-j+1$ Householder matrix to the bottom $m-j+1$ rows of A. However, mathematically we have also applied the m-by-m Householder matrix

$$\tilde{P} = \begin{bmatrix} I_{j-1} & 0 \\ 0 & P \end{bmatrix} = I_m - \beta\tilde{v}\tilde{v}^T \qquad \tilde{v} = \begin{bmatrix} 0 \\ v \end{bmatrix}$$

to A in its entirety. Regardless, the "essential" part of the Householder vector can be recorded in the zeroed portion of A.

5.1.5 Roundoff Properties

The roundoff properties associated with Householder matrices are very favorable. Wilkinson (1965, pp. 152-62) shows that **house** produces a Householder vector $\hat{v}$ very near the exact v. If $\hat{P} = I - 2\hat{v}\hat{v}^T/\hat{v}^T\hat{v}$ then

$$\| \hat{P} - P \|_2 = O(\mathbf{u})$$

meaning that $\hat{P}$ is *orthogonal to machine precision*. Moreover, the computed updates with $\hat{P}$ are close to the exact updates with P :

$$\begin{aligned} fl(\hat{P}A) &= P(A+E) \qquad \| E \|_2 = O(\mathbf{u}\| A \|_2) \\ fl(A\hat{P}) &= (A+E)P \qquad \| E \|_2 = O(\mathbf{u}\| A \|_2) \end{aligned}$$

5.1.6 Factored Form Representation

Many Householder based factorization algorithms that are presented in the following sections compute products of Householder matrices

$$Q = Q_1Q_2\cdots Q_r \qquad Q_j = I - \beta_j v^{(j)}v^{(j)^T} \tag{5.1.3}$$

where $r \le n$ and each $v^{(j)}$ has the form

$$v^{(j)} = (\underbrace{0,\ 0, \cdots 0,}_{j-1}\ 1\ v_{j+1}^{(j)},\ \cdots\ , v_n^{(j)})^T .$$

It is usually not necessary to compute Q explicitly even if it is involved in subsequent calculations. For example, if $C \in \mathbb{R}^{n\times q}$ and we wish to compute Q^TC , then we merely execute the loop

```
for $j = 1:r$
    $C = Q_jC$
end
```

The storage of the Householder vectors $v^{(1)} \cdots v^{(r)}$ and the corresponding β_j (if convenient) amounts to a *factored form* representation of Q. To illustrate the economies of the factored form representation, suppose that we have an array A and that $A(j+1:n,j)$ houses $v^{(j)}(j+1:n)$, the essential part of the jth Householder vector. The overwriting of $C \in \mathbb{R}^{n\times q}$ with Q^TC can then be implemented as follows:

```
for $j = 1:r$
    $v(j:n) = \begin{bmatrix} 1 \\ A(j+1:n,j) \end{bmatrix}$        (5.1.4)
    $C(j:n,:) = (I - \beta_j v(j:n)v(j:n)^T)C(j:n,:)$
end
```

This involves about $2qr(2n - r)$ flops. If Q is explicitly represented as an n-by-n matrix, Q^TC would involve $2n^2q$ flops.

Of course, in some applications, it is necessary to explicitly form Q (or parts of it). Two possible algorithms for computing the Householder product matrix Q in (5.1.3) are *forward accumulation*,

```
$Q = I_n$
for $j = 1{:}r$
    $Q = QQ_j$
end
```

and *backward accumulation*,

```
$Q = I_n$
for $j = r{:} - 1{:}1$
    $Q = Q_jQ$
end
```

Recall that the leading $(j-1)$-by-$(j-1)$ portion of Q_j is the identity. Thus, at the beginning of backward accumulation, Q is "mostly the identity" and it gradually becomes full as the iteration progresses. This pattern can be exploited to reduce the number of required flops. In contrast, Q is full in forward accumulation after the first step. For this reason, backward accumulation is cheaper and the strategy of choice:

```
$Q = I_n$
for $j = r{:} - 1{:}1$
    $v(j{:}n) = \begin{bmatrix} 1 \\ A(j+1{:}n, j) \end{bmatrix}$                (5.1.5)
    $Q(j{:}n, j{:}n) = (I - \beta_j v(j{:}n)v(j{:}n)^T)Q(j{:}n, j{:}n)$
end
```

This involves about $4(n^2r - nr^2 + r^3/3)$ flops.

5.1.7 A Block Representation

Suppose $Q = Q_1 \cdots Q_r$ is a product of n-by-n Householder matrices as in (5.1.3). Since each Q_j is a rank-one modification of the identity, it follows from the structure of the Householder vectors that Q is a rank-r modification of the identity and can be written in the form

$$Q = I + WY^T \tag{5.1.6}$$

where W and Y are n-by-r matrices. The key to computing the *block representation* (5.1.6) is the following lemma.

Lemma 5.1.1 *Suppose $Q = I + WY^T$ is an n-by-n orthogonal matrix with $W, Y \in \mathbb{R}^{n \times j}$. If $P = I - \beta vv^T$ with $v \in \mathbb{R}^n$ and $z = -\beta Qv$, then*

$$Q_+ = QP = I + W_+Y_+^T$$

where $W_+ = [\, W \; z \,]$ and $Y_+ = [\, Y \; v \,]$ are each n-by-$(j+1)$.

Proof.

$$\begin{aligned} QP &= (I + WY^T)(I - \beta vv^T) = I + WY^T - \beta Qvv^T \\ &= I + WY^T + zv^T = I + [\, W \; z \,][\, Y \; v \,]^T \quad \Box \end{aligned}$$

By repeatedly applying the lemma, we can generate the block representation of Q in (5.1.3) from the factored form representation as follows:

Algorithm 5.1.2 Suppose $Q = Q_1 \cdots Q_r$ is a product of n-by-n Householder matrices as described in (5.1.3). This algorithm computes matrices $W, Y \in \mathbb{R}^{n \times r}$ such that $Q = I + WY^T$.

```
$Y = v^{(1)}$
$W = -\beta_1 v^{(1)}$
for $j = 2{:}r$
    $z = -\beta_j (I + WY^T) v^{(j)}$
    $W = [W \; z]$
    $Y = [\, Y \; v^{(j)} \,]$
end
```

This algorithm involves about $2r^2n - 2r^3/3$ flops if the zeros in the $v^{(j)}$ are exploited. Note that Y is merely the matrix of Householder vectors and is therefore unit lower triangular. Clearly, the central task in the generation of the WY representation (5.1.6) is the computation of the W matrix.

The block representation for products of Householder matrices is attractive in situations where Q must be applied to a matrix. Suppose $C \in \mathbb{R}^{n \times q}$. It follows that the operation

$$C \leftarrow Q^TC = (I + WY^T)^TC = C + Y(W^TC)$$

is rich in level-3 operations. On the other hand, if Q is in factored form, Q^TC is just rich in the level-2 operations of matrix-vector multiplication and outer product updates. Of course, in this context the distinction between level-2 and level-3 diminishes as C gets narrower.

We mention that the "WY" representation is not a generalized Householder transformation from the geometric point of view. True block reflectors have the form $Q = I - 2VV^T$ where $V \in \mathbb{R}^{n \times r}$ satisfies $V^TV = I_r$. See Schreiber and Parlett (1987) and also Schreiber and Van Loan (1989).

Example 5.1.3 If $n = 4$, $r = 2$, and $[\, 1, \; .6, \; 0, \; .8 \,]^T$ and $[\, 0, \; 1, \; .8, \; .6 \,]^T$ are the

Householder vectors associated with Q_1 and Q_2 respectively, then

$$Q_1Q_2 = I_4 + WY^T \equiv I_4 + \begin{bmatrix} -1 & 1.080 \\ -.6 & -.352 \\ 0 & -.800 \\ -.8 & .264 \end{bmatrix} \begin{bmatrix} 1 & .6 & 0 & .8 \\ 0 & 1 & .8 & .6 \end{bmatrix}.$$

5.1.8 Givens Rotations

Householder reflections are exceedingly useful for introducing zeros on a grand scale, e.g., the annihilation of all but the first component of a vector. However, in calculations where it is necessary to zero elements more selectively, *Givens rotations* are the transformation of choice. These are rank-two corrections to the identity of the form

$$G(i,k,\theta) = \begin{array}{c} \left[\begin{array}{ccccccc} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{array}\right] \begin{array}{c} \\ \\ i \\ \\ k \\ \\ \\ \end{array} \\ \begin{array}{ccccccc} \quad & & i & & k & & \end{array} \end{array} \qquad (5.1.7)$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$ for some θ. Givens rotations are clearly orthogonal.

Premultiplication by $G(i,k,\theta)^T$ amounts to a counterclockwise rotation of θ radians in the (i,k) coordinate plane. Indeed, if $x \in \mathbb{R}^n$ and $y = G(i,k,\theta)^T x$, then

$$y_j = \begin{cases} cx_i - sx_k & j = i \\ sx_i + cx_k & j = k \\ x_j & j \neq i,k \end{cases}.$$

From these formulae it is clear that we can force y_k to be zero by setting

$$c = \frac{x_i}{\sqrt{x_i^2 + x_k^2}} \qquad s = \frac{-x_k}{\sqrt{x_i^2 + x_k^2}} \qquad (5.1.8)$$

Thus, it is a simple matter to zero a specified entry in a vector by using a Givens rotation. In practice, there are better ways to compute c and s than (5.1.8). The following algorithm, for example, guards against overflow.

Algorithm 5.1.3 Given scalars a and b, this function computes $c = \cos(\theta)$ and $s = \sin(\theta)$ so

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}.$$

```
function: $[c, s] = \text{givens}(a, b)$
    if $b = 0$
        $c = 1;\ s = 0$
    else
        if $|b| > |a|$
            $\tau = -a/b;\ s = 1/\sqrt{1+\tau^2};\ c = s\tau$
        else
            $\tau = -b/a;\ c = 1/\sqrt{1+\tau^2};\ s = c\tau$
        end
    end
```

This algorithm requires 5 flops and a single square root. Note that it does not compute θ and so it does not involve inverse trigonometric functions.

Example 5.1.4 If $x = [1,\ 2,\ 3,\ 4]^T$, $\cos(\theta) = 1/\sqrt{5}$, and $\sin(\theta) = -2/\sqrt{5}$, then $G(2,4,\theta)x = [1,\ \sqrt{20},\ 3,\ 0]^T$.

5.1.9 Applying Givens Rotations

It is critical that the simple structure of a Givens rotation matrix be exploited when it is involved in a matrix multiplication. Suppose $A \in \mathbb{R}^{m\times n}$, $c = \cos(\theta)$, and $s = \sin(\theta)$. If $G(i,k,\theta) \in \mathbb{R}^{m\times m}$, then the update $A \leftarrow G(i,k,\theta)^T A$ effects just two rows of A,

$$A([i,k],:) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T A([i,k],:)$$

and requires just $6n$ flops:

```
for $j = 1{:}n$
    $\tau_1 = A(i,j)$
    $\tau_2 = A(k,j)$
    $A(1,j) = c\tau_1 - s\tau_2$
    $A(2,j) = s\tau_1 + c\tau_2$
end
```

Likewise, if $G(i,k,\theta) \in \mathbb{R}^{n\times n}$, then the update $A \leftarrow AG(i,k,\theta)$ effects just two columns of A,

$$A(:,[i,k]) = A(:,[i,k]) \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

and requires just $6m$ flops:

```
for $j = 1{:}m$
    $\tau_1 = A(j,i)$
    $\tau_2 = A(j,k)$
    $A(j,i) = c\tau_1 - s\tau_2$
    $A(j,k) = s\tau_1 + c\tau_2$
end
```

5.1.10 Roundoff Properties

The numerical properties of Givens rotations are as favorable as those for Householder reflections. In particular, it can be shown that the computed $\hat{c}$ and $\hat{s}$ in **givens** satisfy

$$\begin{aligned} \hat{c} &= c(1+\epsilon_c) & \epsilon_c &= O(\mathbf{u}) \\ \hat{s} &= s(1+\epsilon_s) & \epsilon_s &= O(\mathbf{u}). \end{aligned}$$

If $\hat{c}$ and $\hat{s}$ are subsequently used in a Givens update, then the computed update is the exact update of a nearby matrix:

$$\begin{aligned} fl[\hat{G}(i,k,\theta)^T A] &= G(i,k,\theta)^T(A+E) & \|E\|_2 &\approx \mathbf{u}\|A\|_2 \\ fl[A\hat{G}(i,k,\theta)] &= (A+E)G(i,k,\theta) & \|E\|_2 &\approx \mathbf{u}\|A\|_2. \end{aligned}$$

A detailed error analysis of Givens rotations may be found in Wilkinson (1965, pp. 131-39).

5.1.11 Representing Products of Givens Rotations

Suppose $Q = G_1 \cdots G_t$ is a product of Givens rotations. As we have seen in connection with Householder reflections, it is more economical to keep the orthogonal matrix Q in factored form than to compute explicitly the product of the rotations. Using a technique demonstrated by Stewart (1976), it is possible to do this in a very compact way. The idea is to associate a single floating point number ρ with each rotation. Specifically, if

$$Z = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \qquad c^2 + s^2 = 1$$

then we define the scalar ρ by

```
if $c = 0$
    $\rho = 1$
elseif $|s| < |c|$
    $\rho = \text{sign}(c)s/2$                                    (5.1.9)
else
    $\rho = 2\text{sign}(s)/c$
end
```

Essentially, this amounts to storing $s/2$ if the sine is smaller and $2/c$ if the cosine is smaller. With this encoding, it is possible to reconstruct $\pm Z$ as follows:

```
if $\rho = 1$
        $c = 0;\ s = 1$
elseif $|\rho| < 1$
        $s = 2\rho;\ c = \sqrt{1 - s^2}$                                  (5.1.10)
else
        $c = 2/\rho;\ s = \sqrt{1 - c^2}$
end
```

That $-Z$ may be generated is usually of no consequence for if Z zeros a particular matrix entry, so does $-Z$. The reason for essentially storing the smaller of c and s is that the formula $\sqrt{1 - x^2}$ renders poor results if x is near unity. More details may be found in Stewart (1976). Of course, to "reconstruct" $G(i, k, \theta)$ we need i and k in addition to the associated ρ. This usually poses no difficulty as we discuss in §5.2.3.

5.1.12 Error Propagation

We offer some remarks about the propagation of roundoff error in algorithms that involve sequences of Householder/Givens updates. To be precise, suppose $A = A_0 \in \mathbb{R}^{m \times n}$ is given and that matrices $A_1, \ldots, A_p = B$ are generated via the formula

$$A_k = fl(\hat{Q}_k A_{k-1} \hat{Z}_k) \qquad k = 1{:}p\,.$$

Assume that the above Householder and Givens algorithms are used for both the generation and application of the $\hat{Q}_k$ and $\hat{Z}_k$. Let Q_k and Z_k be the orthogonal matrices that would be produced in the absence of roundoff. It can be shown that

$$B \;=\; (Q_p \cdots Q_1)(A + E)(Z_1 \cdots Z_p), \tag{5.1.11}$$

where $\| E \|_2 \le c\mathbf{u} \| A \|_2$ and c is a constant that depends mildly on n, m, and p. In plain English, B is an exact orthogonal update of a matrix near to A.

5.1.13 Fast Givens Transformations

The ability to introduce zeros in a selective fashion makes Givens rotations an important zeroing tool in certain structured problems. This has led to the development of "fast Givens" procedures. The fast Givens idea amounts to a clever representation of Q when Q is the product of Givens rotations.

In particular, Q is represented by a matrix pair (M, D) where $M^TM = D = \text{diag}(d_i)$ and each d_i is positive. The matrices Q, M, and D are connected through the formula

$$Q = MD^{-1/2} = M\text{diag}(1/\sqrt{d_i}).$$

Note that $(MD^{-1/2})^T(MD^{-1/2}) = D^{-1/2}DD^{-1/2} = I$ and so the matrix $MD^{-1/2}$ is orthogonal. Moreover, if F is an n-by-n matrix with $F^TDF = D_{new}$ diagonal, then $M_{new}^TM_{new} = D_{new}$ where $M_{new} = MF$. Thus, it is possible to update the fast Givens representation (M, D) to obtain (M_{new}, D_{new}). For this idea to be of practical interest, we must show how to give F zeroing capabilities subject to the constraint that it "keeps" D diagonal.

The details are best explained at the 2-by-2 level. Let $x = [x_1\ x_2]^T$ and $D = \text{diag}(d_1, d_2)$ be given and assume that d_1 and d_2 are positive. Define

$$M_1 = \begin{bmatrix} \beta_1 & 1 \\ 1 & \alpha_1 \end{bmatrix} \tag{5.1.12}$$

and observe that

$$M_1^Tx = \begin{bmatrix} \beta_1x_1 + x_2 \\ x_1 + \alpha_1x_2 \end{bmatrix}$$

and

$$M_1^TDM_1 = \begin{bmatrix} d_2 + \beta_1^2d_1 & d_1\beta_1 + d_2\alpha_1 \\ d_1\beta_1 + d_2\alpha_1 & d_1 + \alpha_1^2d_2 \end{bmatrix} \equiv D_1 .$$

If $x_2 \neq 0$, $\alpha_1 = -x_1/x_2$, and $\beta_1 = -\alpha_1d_2/d_1$, then

$$M_1^Tx = \begin{bmatrix} x_2(1+\gamma_1) \\ 0 \end{bmatrix}$$

$$M_1^TDM_1 = \begin{bmatrix} d_2(1+\gamma_1) & 0 \\ 0 & d_1(1+\gamma_1) \end{bmatrix}$$

where $\gamma_1 = -\alpha_1\beta_1 = (d_2/d_1)(x_1/x_2)^2$.

Analogously, if we assume $x_1 \neq 0$ and define M_2 by

$$M_2 = \begin{bmatrix} 1 & \alpha_2 \\ \beta_2 & 1 \end{bmatrix} \tag{5.1.13}$$

where $\alpha_2 = -x_2/x_1$ and $\beta_2 = -(d_1/d_2)\alpha_2$, then

$$M_2^Tx = \begin{bmatrix} x_1(1+\gamma_2) \\ 0 \end{bmatrix}$$

and

$$M_2^TDM_2 = \begin{bmatrix} d_1(1+\gamma_2) & 0 \\ 0 & d_2(1+\gamma_2) \end{bmatrix} \equiv D_2,$$

where $\gamma_2 = -\alpha_2\beta_2 = (d_1/d_2)(x_2/x_1)^2$.

It is easy to show that for either $i = 1$ or 2, the matrix $J = D^{1/2}M_iD_i^{-1/2}$ is orthogonal and that it is designed so that the second component of $J^T(D^{-1/2}x)$ is zero. (J may actually be a reflection and thus it is half-correct to use the popular term "fast Givens.")

Notice that the γ_i satisfy $\gamma_1\gamma_2 = 1$. Thus, we can always select M_i in the above so that the "growth factor" $(1 + \gamma_i)$ is bounded by 2. Matrices of the form

$$M_1 = \left[\begin{array}{cc} \beta_1 & 1 \\ 1 & \alpha_1 \end{array}\right] \qquad M_2 = \left[\begin{array}{cc} 1 & \alpha_2 \\ \beta_2 & 1 \end{array}\right]$$

that satisfy $-1 \le \alpha_i\beta_i \le 0$ are 2-by-2 *fast Givens transformations*. Notice that premultiplication by a fast Givens transformation involves half the number of multiplies as premultiplication by an "ordinary" Givens transformation. Also, the zeroing is carried out without an explicit square root.

In the n-by-n case, everything "scales up" as with ordinary Givens rotations. The "type 1" transformations have the form

$$F(i,k,\alpha,\beta) = \begin{array}{cc} \left[\begin{array}{ccccccc} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \beta & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & \alpha & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{array}\right] & \begin{array}{c} \\ \\ i \\ \\ k \\ \\ \\ \end{array} \\ \begin{array}{ccccccc} & & i & & k & & \end{array} & \end{array} \qquad (5.1.14)$$

while the "type 2" transformations are structured as follows:

$$F(i,k,\alpha,\beta) = \begin{array}{cc} \left[\begin{array}{ccccccc} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 1 & \cdots & \alpha & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & \beta & \cdots & 1 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{array}\right] & \begin{array}{c} \\ \\ i \\ \\ k \\ \\ \\ \end{array} \\ \begin{array}{ccccccc} & & i & & k & & \end{array} & \end{array} \qquad (5.1.15)$$

Encapsulating all this we obtain

Algorithm 5.1.4 Given $x \in \mathbb{R}^2$ and positive $d \in \mathbb{R}^2$, the following algorithm computes a 2-by-2 fast Givens transformation M such that the

second component of M^Tx is zero and $M^TDM = D_1$ is diagonal where $D = \text{diag}(d_1, d_2)$. If *type* = 1 then M has the form (5.1.12) while if *type* = 2 then M has the form (5.1.13). The diagonal elements of D_1 overwrite d.

```
function: [$\alpha$, $\beta$, type] = fast.givens($x$, $d$)
    if $x(2) \neq 0$
        $\alpha = -x(1)/x(2)$; $\beta = -\alpha d(2)/d(1)$; $\gamma = -\alpha\beta$
        if $\gamma \leq 1$
            type = 1
            $\tau = d(1)$; $d(1) = (1+\gamma)d(2)$; $d(2) = (1+\gamma)\tau$
        else
            type = 2
            $\alpha = 1/\alpha$; $\beta = 1/\beta$; $\gamma = 1/\gamma$
            $d(1) = (1+\gamma)d(1)$; $d(2) = (1+\gamma)d(2)$
        end
    else
        type = 2
        $\alpha = 0$; $\beta = 0$
    end
```

The application of fast Givens transformations is analogous to that for ordinary Givens transformations. Even with the appropriate type of transformation used, the growth factor $1+\gamma$ may still be as large as two. Thus, 2^s growth can occur in the entries of D and M after s updates. This means that the diagonal D must be monitored during a fast Givens procedure to avoid overflow. See Anda and Park (1994) for how to do this efficiently.

Nevertheless, element growth in M and D is controlled because at all times we have $MD^{-1/2}$ orthogonal. The roundoff properties of a fast givens procedure are what we would expect of a Givens matrix technique. For example, if we computed $\hat{Q} = fl(\hat{M}\hat{D}^{-1/2})$ where $\hat{M}$ and $\hat{D}$ are the computed M and D, then $\hat{Q}$ is orthogonal to working precision: $\| \hat{Q}^T\hat{Q} - I \|_2 \approx \mathbf{u}$.

Problems

P5.1.1 Execute **house** with $x = [\,1,\ 7,\ 2,\ 3,\ -1\,]^T$.

P5.1.2 Let x and y be nonzero vectors in $\mathbb{R}^n$. Give an algorithm for determining a Householder matrix P such that Px is a multiple of y.

P5.1.3 Suppose $x \in \mathbb{C}^n$ and that $x_1 = |x_1|e^{i\theta}$ with $\theta \in \mathbb{R}$. Assume $x \neq 0$ and define $u = x + e^{i\theta}\| x \|_2 e_1$ Show that $P = I - 2uu^H/u^Hu$ is unitary and that $Px = -e^{i\theta}\| x \|_2 e_1$.

P5.1.4 Use Householder matrices to show that $\det(I + xy^T) = 1 + x^Ty$ where x and y are given n-vectors.

P5.1.5 Suppose $x \in \mathbb{C}^2$. Give an algorithm for determining a unitary matrix of the form

$$Q = \begin{bmatrix} c & s \\ -\bar{s} & c \end{bmatrix} \qquad c \in \mathbb{R},\ c^2 + |s|^2 = 1$$

such that the second component of Q^Hx is zero.

P5.1.6 Suppose x and y are unit vectors in $\mathbb{R}^n$. Give an algorithm using Givens transformations which computes an orthogonal Q such that $Q^T x = y$.

P5.1.7 Determine $c = \cos(\theta)$ and $s = \sin(\theta)$ such that

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T \begin{bmatrix} 5 \\ 12 \end{bmatrix} = \begin{bmatrix} 13 \\ 0 \end{bmatrix}.$$

P5.1.8 Suppose that $Q = I + YTY^T$ is orthogonal where $Y \in \mathbb{R}^{n \times j}$ and $T \in \mathbb{R}^{j \times j}$ is upper triangular. Show that if $Q_+ = QP$ where $P = I - 2vv^T/v^Tv$ is a Householder matrix, then Q_+ can be expressed in the form $Q_+ = I + Y_+T_+Y_+^T$ where $Y_+ \in \mathbb{R}^{n \times (j+1)}$ and $T_+ \in \mathbb{R}^{(j+1)\times(j+1)}$ is upper triangular.

P5.1.9 Give a detailed implementation of Algorithm 5.1.2 with the assumption that $v^{(j)}(j+1{:}n)$, the essential part of the the jth Householder vector, is stored in $A(j+1{:}n, j)$. Since Y is effectively represented in A, your procedure need only set up the W matrix.

P5.1.10 Show that if S is skew-symmetric ($S^T = -S$), then $Q = (I + S)(I - S)^{-1}$ is orthogonal. (Q is called the *Cayley transform* of S.) Construct a rank-2 S so that if x is a vector then Qx is zero except in the first component.

P5.1.11 Suppose $P \in \mathbb{R}^{n \times n}$ satisfies $\| P^TP - I_n \|_2 = \epsilon < 1$. Show that all the singular values of P are in the interval $[1-\epsilon, 1+\epsilon]$ and that $\| P - UV^T \|_2 \le \epsilon$ where $P = U\Sigma V^T$ is the SVD of P.

P5.1.12 Suppose $A \in \mathbb{R}^{2\times 2}$. Under what conditions is the closest rotation to A closer than the closest reflection to A?

Notes and References for Sec. 5.1

Householder matrices are named after A.S. Householder, who popularized their use in numerical analysis. However, the properties of these matrices have been known for quite some time. See

H.W. Turnbull and A.C. Aitken (1961). *An Introduction to the Theory of Canonical Matrices*, Dover Publications, New York, pp. 102–5.

Other references concerned with Householder transformations include

A.R. Gourlay (1970). "Generalization of Elementary Hermitian Matrices," *Comp. J. 13*, 411–12.
B.N. Parlett (1971). "Analysis of Algorithms for Reflections in Bisectors," *SIAM Review 13*, 197–208.
N.K. Tsao (1975). "A Note on Implementing the Householder Transformations." *SIAM J. Num. Anal. 12*, 53–58.
B. Danloy (1976). "On the Choice of Signs for Householder Matrices," *J. Comp. Appl. Math. 2*, 67–69.
J.J.M. Cuppen (1984). "On Updating Triangular Products of Householder Matrices," *Numer. Math. 45*, 403–410.
L. Kaufman (1987). "The Generalized Householder Transformation and Sparse Matrices," *Lin. Alg. and Its Applic. 90*, 221–234.

A detailed error analysis of Householder transformations is given in Lawson and Hanson (1974, 83–89).

The basic references for block Householder representations and the associated computations include

C.H. Bischof and C. Van Loan (1987). "The WY Representation for Products of Householder Matrices," *SIAM J. Sci. and Stat. Comp. 8*, s2–s13.

R. Schreiber and B.N. Parlett (1987). "Block Reflectors: Theory and Computation," *SIAM J. Numer. Anal. 25*, 189-205.

B.N. Parlett and R. Schreiber (1988). "Block Reflectors: Theory and Computation," *SIAM J. Num. Anal. 25*, 189–205.

R.S. Schreiber and C. Van Loan (1989). "A Storage-Efficient WY Representation for Products of Householder Transformations," *SIAM J. Sci. and Stat. Comp. 10*, 52–57.

C. Puglisi (1992). "Modification of the Householder Method Based on the Compact WY Representation," *SIAM J. Sci. and Stat. Comp. 13*, 723–726.

X. Sun and C.H. Bischof (1995). "A Basis-Kernel Representation of Orthogonal Matrices," *SIAM J. Matrix Anal. Appl. 16*, 1184–1196.

Givens rotations, named after W. Givens, are also referred to as Jacobi rotations. Jacobi devised a symmetric eigenvalue algorithm based on these transformations in 1846. See §8.4. The Givens rotation storage scheme discussed in the text is detailed in

G.W. Stewart (1976). "The Economical Storage of Plane Rotations," *Numer. Math. 25*, 137–38.

Fast Givens transformations are also referred to as "square-root-free" Givens transformations. (Recall that a square root must ordinarily be computed during the formation of Givens transformation.) There are several ways fast Givens calculations can be arranged. See

M. Gentleman (1973). "Least Squares Computations by Givens Transformations without Square Roots," *J. Inst. Math. Appl. 12*, 329–36.

C.F. Van Loan (1973). "Generalized Singular Values With Algorithms and Applications," Ph.D. thesis, University of Michigan, Ann Arbor.

S. Hammarling (1974). "A Note on Modifications to the Givens Plane Rotation," *J. Inst. Math. Appl. 13*, 215–18.

J.H. Wilkinson (1977). "Some Recent Advances in Numerical Linear Algebra," in *The State of the Art in Numerical Analysis*, ed. D.A.H. Jacobs, Academic Press, New York, pp. 1–53.

A.A. Anda and H. Park (1994). "Fast Plane Rotations with Dynamic Scaling," *SIAM J. Matrix Anal. Appl. 15*, 162–174.

5.2 The QR Factorization

We now show how Householder and Givens transformations can be used to compute various factorizations, beginning with the QR factorization. The QR factorization of an m-by-n matrix A is given by

$$A = QR$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $R \in \mathbb{R}^{m\times n}$ is upper triangular. In this section we assume $m \geq n$. We will see that if A has full column rank, then the first n columns of Q form an orthonormal basis for ran(A). Thus, calculation of the QR factorization is one way to compute an orthonormal basis for a set of vectors. This computation can be arranged in several ways. We give methods based on Householder, block Householder, Givens, and fast Givens transformations. The Gram-Schmidt orthogonalization process and a numerically more stable variant called modified Gram-Schmidt are also discussed.

5.2.1 Householder QR

We begin with a QR factorization method that utilizes Householder transformations. The essence of the algorithm can be conveyed by a small example. Suppose $m = 6$, $n = 5$, and assume that Householder matrices H_1 and H_2 have been computed so that

$$H_2H_1A = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \boxtimes & \times & \times \\ 0 & 0 & \boxtimes & \times & \times \\ 0 & 0 & \boxtimes & \times & \times \\ 0 & 0 & \boxtimes & \times & \times \end{bmatrix}.$$

Concentrating on the highlighted entries, we determine a Householder matrix $\tilde{H}_3 \in \mathbb{R}^{4\times 4}$ such that

$$\tilde{H}_3 \begin{bmatrix} \boxtimes \\ \boxtimes \\ \boxtimes \\ \boxtimes \end{bmatrix} = \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

If $H_3 = \text{diag}(I_2, \tilde{H}_3)$, then

$$H_3H_2H_1A = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}.$$

After n such steps we obtain an upper triangular $H_nH_{n-1}\cdots H_1A = R$ and so by setting $Q = H_1\cdots H_n$ we obtain $A = QR$.

Algorithm 5.2.1 (Householder QR) Given $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, the following algorithm finds Householder matrices $H_1, \ldots, H_n$ such that if $Q = H_1\cdots H_n$, then $Q^TA = R$ is upper triangular. The upper triangular part of A is overwritten by the upper triangular part of R and components $j+1{:}m$ of the jth Householder vector are stored in $A(j+1{:}m, j), j < m$.

```
for $j = 1{:}n$
    $[v, \beta] = \textbf{house}(A(j{:}m, j))$
    $A(j{:}m, j{:}n) = (I_{m-j+1} - \beta vv^T)A(j{:}m, j{:}n)$
    if $j < m$
        $A(j+1{:}m, j) = v(2{:}m-j+1)$
    end
end
```

This algorithm requires $2n^2(m - n/3)$ flops.

To clarify how A is overwritten, if

$$v^{(j)} = [\underbrace{0,\ldots,0}_{j-1},1,v_{j+1}^{(j)},\ldots,v_m^{(j)}]^T$$

is the jth Householder vector, then upon completion

$$A = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} \\ v_2^{(1)} & r_{22} & r_{23} & r_{24} & r_{25} \\ v_3^{(1)} & v_3^{(2)} & r_{33} & r_{34} & r_{35} \\ v_4^{(1)} & v_4^{(2)} & v_4^{(3)} & r_{44} & r_{45} \\ v_5^{(1)} & v_5^{(2)} & v_5^{(3)} & v_5^{(4)} & r_{55} \\ v_6^{(1)} & v_6^{(2)} & v_6^{(3)} & v_6^{(4)} & v_6^{(5)} \end{bmatrix}.$$

If the matrix $Q = H_1 \cdots H_n$ is required, then it can be accumulated using (5.1.5). This accumulation requires $4(m^2n - mn^2 + n^3/3)$ flops.

The computed upper triangular matrix $\hat{R}$ is the exact R for a nearby A in the sense that $Z^T(A+E) = \hat{R}$ where Z is some exact orthogonal matrix and $\| E \|_2 \approx \mathbf{u}\| A \|_2$.

5.2.2 Block Householder QR Factorization

Algorithm 5.2.1 is rich in the level-2 operations of matrix-vector multiplication and outer product updates. By reorganizing the computation and using the block Householder representation discussed in §5.1.7 we can obtain a level-3 procedure. The idea is to apply clusters of Householder transformations that are represented in the WY form of §5.1.7.

A small example illustrates the main idea. Suppose $n = 12$ and that the "blocking parameter" r has the value $r = 3$. The first step is to generate Householders H_1, H_2, and H_3 as in Algorithm 5.2.1. However, unlike Algorithm 5.2.1 where the H_i are applied to all of A, we only apply H_1, H_2, and H_3 to $A(:,1{:}3)$. After this is accomplished we generate the block representation $H_1H_2H_3 = I + W_1Y_1^T$ and then perform the level-3 update

$$A(:,4{:}12) = (I + WY^T)A(:,4{:}12).$$

Next, we generate H_4, H_5, and H_6 as in Algorithm 5.2.1. However, these transformations are not applied to $A(:,7{:}12)$ until their block representation $H_4H_5H_6 = I + W_2Y_2^T$ is found. This illustrates the general pattern.

```
$\lambda = 1$; $k = 0$
while $\lambda \le n$
    $\tau = \min(\lambda + r - 1, n)$; $k = k + 1$
    Using Algorithm 5.2.1, upper triangularize $A(\lambda{:}m, \lambda{:}n)$
        generating Householder matrices $H_\lambda, \ldots, H_\tau$.        (5.2.1)
    Use Algorithm 5.1.2 to get the block representation
        $I + W_k Y_k = H_\lambda, \ldots, H_\tau$.
    $A(\lambda{:}m, \tau + 1{:}n) = (I + W_k Y_k^T)^T A(\lambda{:}m, \tau + 1{:}n)$
    $\lambda = \tau + 1$
end
```

The zero-nonzero structure of the Householder vectors that define the matrices $H_\lambda, \ldots, H_\tau$ implies that the first $\lambda - 1$ rows of W_k and Y_k are zero. This fact would be exploited in a practical implementation.

The proper way to regard (5.2.1) is through the partitioning

$$A = [\, A_1, \ldots, A_N \,] \qquad N = \text{ceil}(n/r)$$

where block column A_k is processed during the kth step. In the kth step of (5.2.1), a block Householder is formed that zeros the subdiagonal portion of A_k. The remaining block columns are then updated.

The roundoff properties of (5.2.1) are essentially the same as those for Algorithm 5.2.1. There is a slight increase in the number of flops required because of the W-matrix computations. However, as a result of the blocking, all but a small fraction of the flops occur in the context of matrix multiplication. In particular, the level-3 fraction of (5.2.1) is approximately $1 - 2/N$. See Bischof and Van Loan (1987) for further details.

5.2.3 Givens QR Methods

Givens rotations can also be used to compute the QR factorization. The 4-by-3 case illustrates the general idea:

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(2,3)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(1,2)}$$

$$\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{(2,3)} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{(3,4)} R$$

Here we have highlighted the 2-vectors that define the underlying Givens rotations. Clearly, if G_j denotes the jth Givens rotation in the reduction, then $Q^T A = R$ is upper triangular where $Q = G_1 \cdots G_t$ and t is the total number of rotations. For general m and n we have:

Algorithm 5.2.2 (Givens QR) Given $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, the following algorithm overwrites A with $Q^TA = R$, where R is upper triangular and Q is orthogonal.

```
for $j = 1{:}n$
    for $i = m{:} - 1{:}j + 1$
        $[c, s] = \mathbf{givens}(A(i-1, j), A(i, j))$
        $A(i-1{:}i, j{:}n) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T A(i-1{:}i, j{:}n)$
    end
end
```

This algorithm requires $3n^2(m - n/3)$ flops. Note that we could use (5.1.9) to encode (c, s) in a single number ρ which could then be stored in the zeroed entry $A(i, j)$. An operation such as $x \leftarrow Q^Tx$ could then be implemented by using (5.1.10), taking care to reconstruct the rotations in the proper order.

Other sequences of rotations can be used to upper triangularize A. For example, if we replace the for statements in Algorithm 5.2.2 with

```
for $i = m{:} - 1{:}2$
    for $j = 1{:}\min\{i - 1, n\}$
```

then the zeros in A are introduced row-by-row.

Another parameter in a Givens QR procedure concerns the planes of rotation that are involved in the zeroing of each a_{ij}. For example, instead of rotating rows $i - 1$ and i to zero a_{ij} as in Algorithm 5.2.2, we could use rows j and i:

```
for $j = 1{:}n$
    for $i = m{:} - 1{:}j + 1$
        $[c, s] = \mathbf{givens}(A(j, j), A(i, j))$
        $A([\,j\ i\,], j{:}n) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T A([\,j\ i\,], j{:}n)$
    end
end
```

5.2.4 Hessenberg QR via Givens

As an example of how Givens rotations can be used in structured problems, we show how they can be employed to compute the QR factorization of an upper Hessenberg matrix. A small example illustrates the general idea.

Suppose $n = 6$ and that after two steps we have computed

$$G(2,3,\theta_2)^T G(1,2,\theta_1)^T A = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}.$$

We then compute $G(3,4,\theta_3)$ to zero the current (4,3) entry thereby obtaining

$$G(3,4,\theta_3)^T G(2,3,\theta_2)^T G(1,2,\theta_1)^T A = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}.$$

Overall we have

Algorithm 5.2.3 (Hessenberg QR) If $A \in \mathbb{R}^{n\times n}$ is upper Hessenberg, then the following algorithm overwrites A with $Q^T A = R$ where Q is orthogonal and R is upper triangular. $Q = G_1 \cdots G_{n-1}$ is a product of Givens rotations where G_j has the form $G_j = G(j, j+1, \theta_j)$.

```
for $j = 1{:}n-1$
    $[\, c\ s\,] = \mathbf{givens}(A(j,j), A(j+1,j))$
    $A(j{:}j+1, j{:}n) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}^T A(j{:}j+1, j{:}n)$
end
```

This algorithm requires about $3n^2$ flops.

5.2.5 Fast Givens QR

We can use the fast Givens transformations described in §5.1.13 to compute an (M, D) representation of Q. In particular, if M is nonsingular and D is diagonal such that $M^T A = T$ is upper triangular and $M^T M = D$ is diagonal, then $Q = MD^{-1/2}$ is orthogonal and $Q^T A = D^{-1/2}T \equiv R$ is upper triangular. Analogous to the Givens QR procedure we have:

Algorithm 5.2.4 (Fast Givens QR) Given $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, the following algorithm computes nonsingular $M \in \mathbb{R}^{m\times m}$ and positive $d(1{:}m)$ such that $M^T A = T$ is upper triangular, and $M^T M = \text{diag}(d_1, \ldots, d_m)$. A is overwritten by T. Note: $A = (MD^{-1/2})(D^{1/2}T)$ is a QR factorization of A.

```
for $i = 1{:}m$
    $d(i) = 1$
end
for $j = 1{:}n$
    for $i = m{:}-1{:}j+1$
        $[\,\alpha,\ \beta,\ type\,] = $ fast.givens$(A(i-1{:}i, j), d(i-1{:}i))$
        if $type = 1$
            $A(i-1{:}i, j{:}n) = \begin{bmatrix} \beta & 1 \\ 1 & \alpha \end{bmatrix}^T A(i-1{:}i, j{:}n)$
        else
            $A(i-1{:}i, j{:}n) = \begin{bmatrix} 1 & \alpha \\ \beta & 1 \end{bmatrix}^T A(i-1{:}i, j{:}n)$
    end
end
```

This algorithm requires $2n^2(m - n/3)$ flops. As we mentioned in the previous section, it is necessary to guard against overflow in fast Givens algorithms such as the above. This means that M, D, and A must be periodically scaled if their entries become large.

If the QR factorization of a narrow band matrix is required, then the fast Givens approach is attractive because it involves no square roots. (We found LDL^T preferable to Cholesky in the narrow band case for the same reason; see §4.3.6.) In particular, if $A \in \mathbb{R}^{m\times n}$ has upper bandwidth q and lower bandwidth p, then $Q^TA = R$ has upper bandwidth $p + q$. In this case Givens QR requires about $O(np(p+q))$ flops and $O(np)$ square roots. Thus, the square roots are a significant portion of the overall computation if $p, q \ll n$.

5.2.6 Properties of the QR Factorization

The above algorithms "prove" that the QR factorization exists. Now we relate the columns of Q to ran(A) and ran$(A)^\perp$ and examine the uniqueness question.

Theorem 5.2.1 *If $A = QR$ is a QR factorization of a full column rank $A \in \mathbb{R}^{m\times n}$ and $A = [\,a_1,\ldots,a_n\,]$ and $Q = [\,q_1,\ldots,q_m\,]$ are column partitionings, then*

$$\text{span}\{a_1,\ldots,a_k\} = \text{span}\{q_1,\ldots,q_k\} \qquad k = 1{:}n\,.$$

In particular, if $Q_1 = Q(1{:}m, 1{:}n)$ and $Q_2 = Q(1{:}m, n+1{:}m)$ then

$$\begin{aligned} \text{ran}(A) &= \text{ran}(Q_1) \\ \text{ran}(A)^\perp &= \text{ran}(Q_2) \end{aligned}$$

and $A = Q_1R_1$ with $R_1 = R(1{:}n, 1{:}n)$.

Proof. Comparing kth columns in $A = QR$ we conclude that

$$a_k = \sum_{i=1}^{k} r_{ik} q_i \in \text{span}\{q_1, \ldots, q_k\} . \tag{5.2.2}$$

Thus, $\text{span}\{a_1, \ldots, a_k\} \subseteq \text{span}\{q_1, \ldots, q_k\}$. However, since $\text{rank}(A) = n$ it follows that $\text{span}\{a_1, \ldots, a_k\}$ has dimension k and so must equal $\text{span}\{q_1, \ldots, q_k\}$ The rest of the theorem follows trivially. □

The matrices $Q_1 = Q(1{:}m, 1{:}n)$ and $Q_2 = Q(1{:}m, n+1{:}m)$ can be easily computed from a factored form representation of Q.

If $A = QR$ is a QR factorization of $A \in \mathbb{R}^{m \times n}$ and $m \geq n$, then we refer to $A = Q(:, 1{:}n)R(1{:}n, 1{:}n)$ as the *thin QR factorization*. The next result addresses the uniqueness issue for the *thin* QR factorization

Theorem 5.2.2 *Suppose $A \in \mathbb{R}^{m \times n}$ has full column rank. The thin QR factorization*

$$A = Q_1 R_1$$

is unique where $Q_1 \in \mathbb{R}^{m \times n}$ has orthonormal columns and R_1 is upper triangular with positive diagonal entries. Moreover, $R_1 = G^T$ where G is the lower triangular Cholesky factor of $A^T A$.

Proof. Since $A^T A = (Q_1 R_1)^T (Q_1 R_1) = R_1^T R_1$ we see that $G = R_1^T$ is the Cholesky factor of $A^T A$. This factor is unique by Theorem 4.2.5. Since $Q_1 = A R_1^{-1}$ it follows that Q_1 is also unique. □

How are Q_1 and R_1 affected by perturbations in A? To answer this question we need to extend the notion of condition to rectangular matrices. Recall from §2.7.3 that the 2-norm condition of a square nonsingular matrix is the ratio of the largest and smallest singular values. For rectangular matrices with full column rank we continue with this definition:

$$A \in \mathbb{R}^{m \times n}, \text{rank}(A) = n \implies \kappa_2(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)} .$$

If the columns of A are nearly dependent, then $\kappa_2(A)$ is large. Stewart (1993) has shown that $O(\epsilon)$ relative error in A induces $O(\epsilon \kappa_2(A))$ relative error in R and Q_1.

5.2.7 Classical Gram-Schmidt

We now discuss two alternative methods that can be used to compute the thin QR factorization $A = Q_1 R_1$ directly. If $\text{rank}(A) = n$, then equation (5.2.2) can be solved for q_k:

$$q_k = \left(a_k - \sum_{i=1}^{k-1} r_{ik} q_i \right) \Big/ r_{kk} .$$

Thus, we can think of q_k as a unit 2-norm vector in the direction of

$$z_k = a_k - \sum_{i=1}^{k-1} r_{ik} q_i$$

where to ensure $z_k \in \text{span}\{q_1, \ldots, q_{k-1}\}^\perp$ we choose

$$r_{ik} = q_i^T a_k \qquad i = 1{:}k-1 .$$

This leads to the *classical Gram-Schmidt* (CGS) algorithm for computing $A = Q_1 R_1$.

```
$R(1,1) = \| A(:,1) \|_2$
$Q(:,1) = A(:,1)/R(1,1)$
for $k = 2{:}n$
    $R(1{:}k-1, k) = Q(1{:}m, 1{:}k-1)^T A(1{:}m, k)$
    $z = A(1{:}m, k) - Q(1{:}m, 1{:}k-1) R(1{:}k-1, k)$                (5.2.3)
    $R(k,k) = \| z \|_2$
    $Q(1{:}m, k) = z/R(k,k)$
end
```

In the kth step of CGS, the kth columns of both Q and R are generated.

5.2.8 Modified Gram-Schmidt

Unfortunately, the CGS method has very poor numerical properties in that there is typically a severe loss of orthogonality among the computed q_i. Interestingly, a rearrangement of the calculation, known as *modified Gram-Schmidt* (MGS), yields a much sounder computational procedure. In the kth step of MGS, the kth column of Q (denoted by q_k) and the kth row of R (denoted by r_k^T) are determined. To derive the MGS method, define the matrix $A^{(k)} \in \mathbb{R}^{m \times (n-k+1)}$ by

$$A - \sum_{i=1}^{k-1} q_i r_i^T = \sum_{i=k}^{n} q_i r_i^T = [\, 0 \; A^{(k)} \,] . \tag{5.2.4}$$

It follows that if

$$A^{(k)} = \begin{array}{cc} [\; z & B \;] \\ 1 & n-k \end{array}$$

then $r_{kk} = \| z \|_2$, $q_k = z/r_{kk}$ and $(r_{k,k+1} \cdots r_{kn}) = q_k^T B$. We then compute the outer product $A^{(k+1)} = B - q_k (r_{k,k+1} \cdots r_{kn})$ and proceed to the next step. This completely describes the kth step of MGS.

Algorithm 5.2.5 (Modified Gram-Schmidt) Given $A \in \mathbb{R}^{m \times n}$ with $\text{rank}(A) = n$, the following algorithm computes the factorization $A = Q_1 R_1$ where $Q_1 \in \mathbb{R}^{m \times n}$ has orthonormal columns and $R_1 \in \mathbb{R}^{n \times n}$ is upper triangular.

```
for $k = 1:n$
        $R(k,k) = \| A(1:m,k) \|_2$
        $Q(1:m,k) = A(1:m,k)/R(k,k)$
        for $j = k+1:n$
                $R(k,j) = Q(1:m,k)^T A(1:m,j)$
                $A(1:m,j) = A(1:m,j) - Q(1:m,k)R(k,j)$
        end
end
```

This algorithm requires $2mn^2$ flops. It is not possible to overwrite A with both Q_1 and R_1. Typically, the MGS computation is arranged so that A is overwritten by Q_1 and the matrix R_1 is stored in a separate array.

5.2.9 Work and Accuracy

If one is interested in computing an orthonormal basis for ran(A), then the Householder approach requires $2mn^2 - 2n^3/3$ flops to get Q in factored form and another $2mn^2 - 2n^3/3$ flops to get the first n columns of Q. (This requires "paying attention" to just the first n columns of Q in (5.1.5).) Therefore, for the problem of finding an orthonormal basis for ran(A), MGS is about twice as efficient as Householder orthogonalization. However, Björck (1967) has shown that MGS produces a computed $\hat{Q}_1 = [\,\hat{q}_1,\ldots,\hat{q}_n\,]$ that satisfies

$$\hat{Q}_1^T\hat{Q}_1 = I + E_{MGS} \qquad \| E_{MGS} \|_2 \approx \mathbf{u}\kappa_2(A)$$

whereas the corresponding result for the Householder approach is of the form

$$\hat{Q}_1^T\hat{Q}_1 = I + E_H \qquad \| E_H \|_2 \approx \mathbf{u}\,.$$

Thus, if orthonormality is critical, then MGS should be used to compute orthonormal bases only when the vectors to be orthogonalized are fairly independent.

We also mention that the computed triangular factor $\hat{R}$ produced by MGS satisfies $\| A - \hat{Q}\hat{R} \| \approx \mathbf{u}\| A \|$ and that there exists a Q with perfectly orthonormal columns such that $\| A - Q\hat{R} \| \approx \mathbf{u}\| A \|$. See Higham (1996, p.379).

Example 5.2.1 If modified Gram-Schmidt is applied to

$$A = \begin{bmatrix} 1 & 1 \\ 10^{-3} & 0 \\ 0 & 10^{-3} \end{bmatrix} \qquad \kappa_2(A) \approx 1.4 \cdot 10^3$$

with 6-digit decimal arithmetic, then

$$[\hat{q}_1\ \hat{q}_2] = \begin{bmatrix} 1.00000 & 0 \\ .001 & -.707107 \\ 0 & .707100 \end{bmatrix}.$$

5.2.10 A Note on Complex QR

Most of the algorithms that we present in this book have complex versions that are fairly straight forward to derive from their real counterparts. (This is *not* to say that everything is easy and obvious at the implementation level.) As an illustration we outline what a complex Householder QR factorization algorithm looks like.

Starting at the level of an individual Householder transformation, suppose $0 \neq x \in \mathbb{C}^n$ and that $x_1 = re^{i\theta}$ where $r, \theta \in \mathbb{R}$. If $v = x \pm e^{i\theta}\| x \|_2 e_1$ and $P = I_n - \beta vv^H$, $\beta = 2/v^H v$, then $Px = \mp e^{i\theta}\| x \|_2 e_1$. (See P5.1.3.) The sign can be determined to maximize $\| v \|_2$ for the sake of stability.

The upper triangularization of $A \in \mathbb{R}^{m\times n}$, $m \geq n$, proceeds as in Algorithm 5.2.1. In step j we zero the subdiagonal portion of $A(j{:}m, j)$:

```
for $j = 1{:}n$
    $x = A(j{:}m, j)$
    $v = x \pm e^{i\theta}\| x \|_2 e_1$ where $x_1 = re^{i\theta}$.
    $\beta = 2/v^H/v$
    $A(j{:}m, j{:}n) = (I_{m-j+1} - \beta vv^H)A(j{:}m, j{:}n)$
end
```

The reduction involves $8n^2(m - n/3)$ real flops, four times the number required to execute Algorithm 5.2.1. If $Q = P_1 \cdots P_n$ is the product of the Householder transformations, then Q is unitary and $Q^T A = R \in \mathbb{R}^{m\times n}$ is complex and upper triangular.

Problems

P5.2.1 Adapt the Householder QR algorithm so that it can efficiently handle the case when $A \in \mathbb{R}^{m\times n}$ has lower bandwidth p and upper bandwidth q.

P5.2.2 Adapt the Householder QR algorithm so that it computes the factorization $A = QL$ where L is lower triangular and Q is orthogonal. Assume that A is square. This involves rewriting the Householder vector function $v = \text{house}(x)$ so that $(I - 2vv^T/v^Tv)x$ is zero everywhere but its bottom component.

P5.2.3 Adapt the Givens QR factorization algorithm so that the zeros are introduced by diagonal. That is, the entries are zeroed in the order $(m, 1)$, $(m-1, 1)$, $(m, 2)$, $(m-2, 1)$, $(m-1, 2)$, $(m, 3)$, etc.

P5.2.4 Adapt the fast Givens QR factorization algorithm so that it efficiently handles the case when A is n-by-n and tridiagonal. Assume that the subdiagonal, diagonal, and superdiagonal of A are stored in $e(1{:}n-1)$, $a(1{:}n)$, $f(1{:}n-1)$ respectively. Design your algorithm so that these vectors are overwritten by the nonzero portion of T.

P5.2.5 Suppose $L \in \mathbb{R}^{m\times n}$ with $m \geq n$ is lower triangular. Show how Householder matrices $H_1 \ldots H_n$ can be used to determine a lower triangular $L_1 \in \mathbb{R}^{n\times n}$ so that

$$H_n \cdots H_1 L = \begin{bmatrix} L_1 \\ 0 \end{bmatrix}$$

Hint: The second step in the 6-by-3 case involves finding H_2 so that

$$H_2 \begin{bmatrix} \times & 0 & 0 \\ \times & \times & 0 \\ \times & \times & \times \\ \times & \times & 0 \\ \times & \times & 0 \\ \times & \times & 0 \end{bmatrix} = \begin{bmatrix} \times & 0 & 0 \\ \times & \times & 0 \\ \times & \times & \times \\ \times & 0 & 0 \\ \times & 0 & 0 \\ \times & 0 & 0 \end{bmatrix}$$

with the property that rows 1 and 3 are left alone.

P5.2.6 Show that if

$$A = \begin{array}{c} \left[\begin{array}{cc} R & w \\ 0 & v \end{array}\right] \\ \begin{array}{cc} k & n-k \end{array} \end{array} \begin{array}{c} k \\ m-k \end{array} \qquad b = \left[\begin{array}{c} c \\ d \end{array}\right] \begin{array}{c} k \\ m-k \end{array}$$

and A has full column rank, then $\min \| Ax - b \|_2^2 = \| d \|_2^2 - \left(v^T d / \| v \|_2\right)^2$.

P5.2.7 Suppose $A \in \mathbb{R}^{n \times n}$ and $D = \text{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n \times n}$. Show how to construct an orthogonal Q such that $Q^T A - DQ^T = R$ is upper triangular. Do not worry about efficiency—this is just an exercise in QR manipulation.

P5.2.8 Show how to compute the QR factorization of the product $A = A_p \cdots A_2 A_1$ without explicitly multiplying the matrices $A_1, \ldots, A_p$ together. Hint: In the $p = 3$ case, write $Q_3^T A = Q_3^T A_3 Q_2 Q_2^T A_2 Q_1 Q_1^T A_1$ and determine orthogonal Q_i so that $Q_i^T (A_i Q_{i-1})$ is upper triangular. $(Q_0 = I)$.

P5.2.9 Suppose $A \in \mathbb{R}^{n \times n}$ and let E be the permutation obtained by reversing the order of the rows in I_n. (This is just the exchange matrix of §4.7.) (a) Show that if $R \in \mathbb{R}^{n \times n}$ is upper triangular, then $L = ERE$ is lower triangular. (b) Show how to compute an orthogonal $Q \in \mathbb{R}^{n \times n}$ and a lower triangular $L \in \mathbb{R}^{n \times n}$ so that $A = QL$ assuming the availability of a procedure for computing the QR factorization.

P5.2.10 MGS applied to $A \in \mathbb{R}^{m \times n}$ is numerically equivalent to the first step in Householder QR applied to

$$\tilde{A} = \begin{bmatrix} O_n \\ A \end{bmatrix}$$

where O_n is the n-by-n zero matrix. Verify that this statement is true after the first step of each method is completed.

P5.2.11 Reverse the loop orders in Algorithm 5.2.5 (MGS QR) so that R is computed column-by-column.

P5.2.12 Develop a complex version of the Givens QR factorization. Refer to P5.1.5. where complex Givens rotations are the theme. Is it possible to organize the calculations so that the diagonal elements of R are nonnegative?

Notes and References for Sec. 5.2

The idea of using Householder transformations to solve the LS problem was proposed in

A.S. Householder (1958). "Unitary Triangularization of a Nonsymmetric Matrix," *J. ACM. 5*, 339–42.

The practical details were worked out in

P. Businger and G.H. Golub (1965). "Linear Least Squares Solutions by Householder Transformations," *Numer. Math.* 7, 269–76. See also Wilkinson and Reinsch (1971,111–18).

G.H. Golub (1965). "Numerical Methods for Solving Linear Least Squares Problems," *Numer. Math.* 7, 206–16.

The basic references on QR via Givens rotations include

W. Givens (1958). "Computation of Plane Unitary Rotations Transforming a General Matrix to Triangular Form," *SIAM J. App. Math. 6*, 26–50.
M. Gentleman (1973). "Error Analysis of QR Decompositions by Givens Transformations," *Lin. Alg. and Its Appl. 10*, 189–97.

For a discussion of how the QR factorization can be used to solve numerous problems in statistical computation, see

G.H. Golub (1969). "Matrix Decompositions and Statistical Computation," in *Statistical Computation* , ed. R.C. Milton and J.A. Nelder, Academic Press, New York, pp. 365–97.

The behavior of the Q and R factors when A is perturbed is discussed in

G.W. Stewart (1977). "Perturbation Bounds for the QR Factorization of a Matrix," *SIAM J. Num. Anal. 14*, 509–18.
H. Zha (1993). "A Componentwise Perturbation Analysis of the QR Decomposition," *SIAM J. Matrix Anal. Appl. 4*, 1124–1131.
G.W. Stewart (1993). "On the Perturbation of LU Cholesky, and QR Factorizations," *SIAM J. Matrix Anal. Appl. 14*, 1141–1145.
A. Barrlund (1994). "Perturbation Bounds for the Generalized QR Factorization," *Lin. Alg. and Its Applic. 207*, 251–271.
J.-G. Sun (1995). "On Perturbation Bounds for the QR Factorization," *Lin. Alg. and Its Applic. 215*, 95–112.

The main result is that the changes in Q and R are bounded by the condition of A times the relative change in A. Organizing the computation so that the entries in Q depend continuously on the entries in A is discussed in

T.F. Coleman and D.C. Sorensen (1984). "A Note on the Computation of an Orthonormal Basis for the Null Space of a Matrix," *Mathematical Programming 29*, 234–242.

References for the Gram-Schmidt process include include

J.R. Rice (1966). "Experiments on Gram-Schmidt Orthogonalization," *Math. Comp. 20*, 325–28.
A. Björck (1967). "Solving Linear Least Squares Problems by Gram-Schmidt Orthogonalization," *BIT 7*, 1–21.
N.N. Abdelmalek (1971). "Roundoff Error Analysis for Gram-Schmidt Method and Solution of Linear Least Squares Problems," *BIT 11*, 345–68.
J. Daniel, W.B. Gragg, L.Kaufman, and G.W. Stewart (1976). "Reorthogonalization and Stable Algorithms for Updating the Gram-Schmidt QR Factorization," *Math. Comp. 30*, 772–795.
A. Ruhe (1983). "Numerical Aspects of Gram-Schmidt Orthogonalization of Vectors," *Lin. Alg. and Its Applic. 52/53*, 591–601.
W. Jalby and B. Philippe (1991). "Stability Analysis and Improvement of the Block Gram-Schmidt Algorithm," *SIAM J. Sci. Stat. Comp. 12*, 1058–1073.
Å. Björck and C.C. Paige (1992). "Loss and Recapture of Orthogonality in the Modified Gram-Schmidt Algorithm," *SIAM J. Matrix Anal. Appl. 13*, 176–190.
A. Björck (1994). "Numerics of Gram-Schmidt Orthogonalization," *Lin. Alg. and Its Applic. 197/198*, 297–316.

The QR factorization of a structured matrix is usually structured itself. See

A.W. Bojanczyk, R.P. Brent, and F.R. de Hoog (1986). "QR Factorization of Toeplitz Matrices," *Numer. Math. 49*, 81–94.

S. Qiao(1986). "Hybrid Algorithm for Fast Toeplitz Orthogonalization," *Numer. Math.* *53*, 351–366.

C.J. Demeure (1989). "Fast QR Factorization of Vandermonde Matrices," *Lin. Alg. and Its Applic.* *122/123/124*, 165–194.

L. Reichel (1991). "Fast QR Decomposition of Vandermonde-Like Matrices and Polynomial Least Squares Approximation," *SIAM J. Matrix Anal. Appl.* *12*, 552–564.

D.R. Sweet (1991). "Fast Block Toeplitz Orthogonalization," *Numer. Math.* *58*, 613–629.

Various high-performance issues pertaining to the QR factorization are discussed in

B. Mattingly, C. Meyer, and J. Ortega (1989). "Orthogonal Reduction on Vector Computers," *SIAM J. Sci. and Stat. Comp.* *10*, 372–381.

P.A. Knight (1995). "Fast Rectangular Matrix Multiplication and the QR Decomposition," *Lin. Alg. and Its Applic.* *221*, 69–81.

5.3 The Full Rank LS Problem

Consider the problem of finding a vector $x \in \mathbb{R}^n$ such that $Ax = b$ where the *data matrix* $A \in \mathbb{R}^{m\times n}$ and the *observation vector* $b \in \mathbb{R}^m$ are given and $m \geq n$. When there are more equations than unknowns, we say that the system $Ax = b$ is *overdetermined*. Usually an overdetermined system has no exact solution since b must be an element of $\text{ran}(A)$, a proper subspace of $\mathbb{R}^m$.

This suggests that we strive to minimize $\| Ax - b \|_p$ for some suitable choice of p. Different norms render different optimum solutions. For example, if $A = [\,1,\ 1,\ 1\,]^T$ and $b = [\,b_1,\ b_2,\ b_3\,]^T$ with $b_1 \geq b_2 \geq b_3 \geq 0$, then it can be verified that

$$\begin{array}{lclcl} p = 1 & \Rightarrow & x_{opt} & = & b_2 \\ p = 2 & \Rightarrow & x_{opt} & = & (b_1 + b_2 + b_3)/3 \\ p = \infty & \Rightarrow & x_{opt} & = & (b_1 + b_3)/2. \end{array}$$

Minimization in the 1-norm and ∞ -norm is complicated by the fact that the function $f(x) = \| Ax - b \|_p$ is not differentiable for these values of p. However, much progress has been made in this area, and there are several good techniques available for 1-norm and ∞-norm minimization. See Coleman and Li (1992), Li (1993), and Zhang (1993).

In contrast to general p-norm minimization, the *least squares* (LS) problem

$$\min_{x\in\mathbb{R}^n} \| Ax - b \|_2 \qquad (5.3.1)$$

is more tractable for two reasons:

- $\phi(x) = \frac{1}{2}\| Ax - b \|_2^2$ is a differentiable function of x and so the minimizers of ϕ satisfy the gradient equation $\nabla\phi(x) = 0$. This turns out to be an easily constructed symmetric linear system which is positive definite if A has full column rank.

- The 2-norm is preserved under orthogonal transformation. This means that we can seek an orthogonal Q such that the equivalent problem of minimizing $\| (Q^TA)x - (Q^Tb) \|_2$ is "easy" to solve.

In this section we pursue these two solution approaches for the case when A has full column rank. Methods based on normal equations and the QR factorization are detailed and compared.

5.3.1 Implications of Full Rank

Suppose $x \in \mathbb{R}^n$, $z \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$ and consider the equality

$$\| A(x + \alpha z) - b \|_2^2 \; = \; \| Ax - b \|_2^2 \; + 2\alpha z^T A^T (Ax - b) + \alpha^2 \| Az \|_2^2$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. If x solves the LS problem (5.3.1) then we must have $A^T(Ax - b) = 0$. Otherwise, if $z = -A^T(Ax - b)$ and we make α small enough, then we obtain the contradictory inequality $\| A(x + \alpha z) - b \|_2 < \| Ax - b \|_2$. We may also conclude that if x *and* $x + \alpha z$ are LS minimizers, then $z \in \text{null}(A)$.

Thus, if A has full column rank, then there is a unique LS solution x_{LS} and it solves the symmetric positive definite linear system

$$A^T A x_{LS} = A^T b\,.$$

These are called the *normal equations*. Since $\nabla\phi(x) = A^T(Ax - b)$ where $\phi(x) = \frac{1}{2}\| Ax - b \|_2^2$, we see that solving the normal equations is tantamount to solving the gradient equation $\nabla\phi = 0$. We call

$$r_{LS} \; = \; b - Ax_{LS}$$

the *minimum residual* and we use the notation

$$\rho_{LS} \; = \; \| Ax_{LS} - b \|_2$$

to denote its size. Note that if ρ_{LS} is small, then we can "predict" b with the columns of A.

So far we have been assuming that $A \in \mathbb{R}^{m \times n}$ has full column rank. This assumption is dropped in §5.5. However, even if $\text{rank}(A) = n$, then we can expect trouble in the above procedures if A is nearly rank deficient.

When assessing the quality of a computed LS solution $\hat{x}_{LS}$, there are two important issues to bear in mind:

- How close is $\hat{x}_{LS}$ to x_{LS}?
- How small is $\hat{r}_{LS} = b - A\hat{x}_{LS}$ compared to $r_{LS} = b - Ax_{LS}$?

The relative importance of these two criteria varies from application to application. In any case it is important to understand how x_{LS} and r_{LS} are affected by perturbations in A and b. Our intuition tells us that if the columns of A are nearly dependent, then these quantities may be quite sensitive.

Example 5.3.1 Suppose

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-6} \\ 0 & 0 \end{bmatrix}, \ \delta A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 10^{-8} \end{bmatrix}, \ b = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \ \delta b = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

and that x_{LS} and $\hat{x}_{LS}$ minimize $\| Ax - b \|_2$ and $\| (A + \delta A)x - (b + \delta b) \|_2$ respectively. Let r_{LS} and $\hat{r}_{LS}$ be the corresponding minimum residuals. Then

$$x_{LS} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \hat{x}_{LS} = \begin{bmatrix} 1 \\ .9999 \cdot 10^{4} \end{bmatrix}, \ r_{LS} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \ \hat{r}_{LS} = \begin{bmatrix} 0 \\ -.9999 \cdot 10^{-2} \\ .9999 \cdot 10^{0} \end{bmatrix}.$$

Since $\kappa_2(A) = 10^6$ we have

$$\frac{\| \hat{x}_{LS} - x_{LS} \|_2}{\| x_{LS} \|_2} \approx .9999 \cdot 10^4 \le \kappa_2(A)^2 \frac{\| \delta A \|_2}{\| A \|_2} = 10^{12} \cdot 10^{-8}$$

and

$$\frac{\| \hat{r}_{LS} - r_{LS} \|_2}{\| b \|_2} \approx .7070 \cdot 10^{-2} \le \kappa_2(A) \frac{\| \delta A \|_2}{\| A \|_2} = 10^{6} \cdot 10^{-8}.$$

The example suggests that the sensitivity of x_{LS} depends upon $\kappa_2(A)^2$. At the end of this section we develop a perturbation theory for the LS problem and the $\kappa_2(A)^2$ factor will return.

5.3.2 The Method of Normal Equations

The most widely used method for solving the full rank LS problem is the method of normal equations.

Algorithm 5.3.1 (Normal Equations) Given $A \in \mathbb{R}^{m \times n}$ with the property that $\text{rank}(A) = n$ and $b \in \mathbb{R}^m$, this algorithm computes the solution x_{LS} to the LS problem $\min \| Ax - b \|_2$ where $b \in \mathbb{R}^m$.

```
Compute the lower triangular portion of $C = A^T A$.
$d = A^T b$
Compute the Cholesky factorization $C = GG^T$.
Solve $Gy = d$ and $G^T x_{LS} = y$.
```

This algorithm requires $(m + n/3)n^2$ flops. The normal equation approach is convenient because it relies on standard algorithms: Cholesky factorization, matrix-matrix multiplication, and matrix-vector multiplication. The compression of the m-by-n data matrix A into the (typically) much smaller n-by-n cross-product matrix C is attractive.

Let us consider the accuracy of the computed normal equations solution $\hat{x}_{LS}$. For clarity, assume that no roundoff errors occur during the formation of $C = A^TA$ and $d = A^Tb$. (On many computers inner products are accumulated in double precision and so this is not a terribly unfair assumption.) It follows from what we know about the roundoff properties of the Cholesky factorization (cf. §4.2.7) that

$$(A^TA + E)\hat{x}_{LS} \;=\; A^Tb,$$

where $\| E \|_2 \approx \mathbf{u}\| A^T \|_2 \| A \|_2 \approx \mathbf{u}\| A^TA \|_2$ and thus we can expect

$$\frac{\| \hat{x}_{LS} - x_{LS} \|_2}{\| x_{LS} \|_2} \;\approx\; \mathbf{u}\kappa_2(A^TA) \;=\; \mathbf{u}\kappa_2(A)^2 . \qquad (5.3.2)$$

In other words, the accuracy of the computed normal equations solution depends on the square of the condition. This seems to be consistent with Example 5.3.1 but more refined comments follow in §5.3.9.

Example 5.3.2 It should be noted that the formation of A^TA can result in a severe loss of information.

$$A \;=\; \begin{bmatrix} 1 & 1 \\ 10^{-3} & 0 \\ 0 & 10^{-3} \end{bmatrix} \text{ and } b \;=\; \begin{bmatrix} 2 \\ 10^{-3} \\ 10^{-3} \end{bmatrix}$$

then $\kappa_2(A) \approx 1.4 \cdot 10^3$, $x_{LS} = [1\ 1]^T$, and $\rho_{LS} = 0$. If the normal equations method is executed with base 10, $t = 6$ arithmetic, then a divide-by-zero occurs during the solution process, since

$$fl(A^TA) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

is exactly singular. On the other hand, if 7-digit arithmetic is used, then $\hat{x}_{LS} = [\,2.000001\ ,\ 0\,]^T$ and $\| \hat{x}_{LS} - x_{LS} \|_2 / \| x_{LS} \|_2 \approx \mathbf{u}\kappa_2(A)^2$.

5.3.3 LS Solution Via QR Factorization

Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and $b \in \mathbb{R}^m$ be given and suppose that an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ has been computed such that

$$Q^TA = R = \begin{array}{c} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \end{array} \begin{array}{l} n \\ m-n \end{array} \qquad (5.3.3)$$

is upper triangular. If

$$Q^Tb = \begin{bmatrix} c \\ d \end{bmatrix} \begin{array}{l} n \\ m-n \end{array}$$

then

$$\| Ax - b \|_2^2 \;= \| Q^TAx - Q^Tb \|_2^2 \;= \| R_1x - c \|_2^2 \;+ \| d \|_2^2$$

for any $x \in \mathbb{R}^n$. Clearly, if $\text{rank}(A) = \text{rank}(R_1) = n$, then x_{LS} is defined by the upper triangular system $R_1 x_{LS} = c$. Note that

$$\rho_{LS} = \| d \|_2.$$

We conclude that the full rank LS problem can be readily solved once we have computed the QR factorization of A. Details depend on the exact QR procedure. If Householder matrices are used and Q^T is applied in factored form to b, then we obtain

Algorithm 5.3.2 (Householder LS Solution) If $A \in \mathbb{R}^{m \times n}$ has full column rank and $b \in \mathbb{R}^m$, then the following algorithm computes a vector $x_{LS} \in \mathbb{R}^n$ such that $\| Ax_{LS} - b \|_2$ is minimum.

```
Use Algorithm 5.2.1 to overwrite $A$ with its QR factorization.
for $j = 1{:}n$
    $v(j) = 1;\ v(j+1{:}m) = A(j+1{:}m, j)$
    $b(j{:}m) = (I_{m-j+1} - \beta_j vv^T) b(j{:}m)$
end
Solve $R(1{:}n, 1{:}n) x_{LS} = b(1{:}n)$ using back substitution.
```

This method for solving the full rank LS problem requires $2n^2(m - n/3)$ flops. The $O(mn)$ flops associated with the updating of b and the $O(n^2)$ flops associated with the back substitution are not significant compared to the work required to factor A.

It can be shown that the computed $\hat{x}_{LS}$ solves

$$\min \| (A + \delta A)x - (b + \delta b) \|_2 \tag{5.3.4}$$

where

$$\| \delta A \|_F \le (6m - 3n + 41)n\mathbf{u} \| A \|_F + O(\mathbf{u}^2) \tag{5.3.5}$$

and

$$\| \delta b \|_2 \le (6m - 3n + 40)n\mathbf{u} \| b \|_2 + O(\mathbf{u}^2). \tag{5.3.6}$$

These inequalities are established in Lawson and Hanson (1974, p.90ff) and show that $\hat{x}_{LS}$ satisfies a "nearby" LS problem. (We cannot address the relative error in $\hat{x}_{LS}$ without an LS perturbation theory, to be discussed shortly.) We mention that similar results hold if Givens QR is used.

5.3.4 Breakdown in Near-Rank Deficient Case

Like the method of normal equations, the Householder method for solving the LS problem breaks down in the back substitution phase if $\text{rank}(A) < n$. Numerically, trouble can be expected whenever $\kappa_2(A) = \kappa_2(R) \approx 1/\mathbf{u}$. This is in contrast to the normal equations approach, where completion of the Cholesky factorization becomes problematical once $\kappa_2(A)$ is in the neighborhood of $1/\sqrt{\mathbf{u}}$. (See Example 5.3.2.) Hence the claim in Lawson and Hanson (1974, 126–127) that for a fixed machine precision, a wider class of LS problems can be solved using Householder orthogonalization.

5.3.5 A Note on the MGS Approach

In principle, MGS computes the thin QR factorization $A = Q_1R_1$. This is enough to solve the full rank LS problem because it transforms the normal equations $(A^TA)x = A^Tb$ to the upper triangular system $R_1x = Q_1^Tb$. But an analysis of this approach when Q_1^Tb is explicitly formed introduces a $\kappa_2(A)^2$ term. This is because the computed factor $\hat{Q}_1$ satisfies $\| \hat{Q}_1^T\hat{Q}_1 - I_n \|_2 \approx \mathbf{u}\kappa_2(A)$ as we mentioned in §5.2.9.

However, if MGS is applied to the augmented matrix

$$A_+ = [\, A \; b \,] = [\, Q_1 \; q_{n+1} \,] \begin{bmatrix} R_1 & z \\ 0 & \rho \end{bmatrix},$$

then $z = Q_1^Tb$. Computing Q_1^Tb in this fashion and solving $R_1x_{LS} = z$ produces an LS solution $\hat{x}_{LS}$ that is "just as good" as the Householder QR method. That is to say, a result of the form (5.3.4)-(5.3.6) applies. See Björck and Paige (1992).

It should be noted that the MGS method is slightly more expensive than Householder QR because it always manipulates m-vectors whereas the latter procedure deals with ever shorter vectors.

5.3.6 Fast Givens LS Solver

The LS problem can also be solved using fast Givens transformations. Suppose $M^TM = D$ is diagonal and

$$M^TA = \begin{array}{c} \begin{bmatrix} S_1 \\ 0 \end{bmatrix} \end{array} \begin{array}{c} n \\ m-n \end{array}$$

is upper triangular. If

$$M^Tb = \begin{array}{c} \begin{bmatrix} c \\ d \end{bmatrix} \end{array} \begin{array}{c} n \\ m-n \end{array}$$

then

$$\| Ax - b \|_2^2 = \| D^{-1/2}M^T(Ax - b) \|_2^2 = \left\| D^{-1/2} \left(\begin{bmatrix} S_1 \\ 0 \end{bmatrix} x - \begin{bmatrix} c \\ d \end{bmatrix} \right) \right\|_2^2$$

for any $x \in \mathbb{R}^n$. Clearly, x_{LS} is obtained by solving the nonsingular upper triangular system $S_1x = c$.

The computed solution $\hat{x}_{LS}$ obtained in this fashion can be shown to solve a nearby LS problem in the sense of (5.3.4)-(5.3.6). This may seem surprising since large numbers can arise during the calculation. An entry in the scaling matrix D can double in magnitude after a single fast Givens update. However, largeness in D must be exactly compensated for by largeness in M, since $D^{-1/2}M$ is orthogonal at all stages of the computation. It is this phenomenon that enables one to push through a favorable error analysis.

5.3.7 The Sensitivity of the LS Problem

We now develop a perturbation theory that assists in the comparison of the normal equations and QR approaches to the LS problem. The theorem below examines how the LS solution and its residual are affected by changes in A and b. In so doing, the condition of the LS problem is identified.

Two easily established facts are required in the analysis:

$$\begin{aligned} \| A \|_2 \, \| (A^TA)^{-1}A^T \|_2 &= \kappa_2(A) \\ \| A \|_2^2 \;\; \| (A^TA)^{-1} \|_2 &= \kappa_2(A)^2 \end{aligned} \tag{5.3.7}$$

These equations can be verified using the SVD.

Theorem 5.3.1 *Suppose x, r, $\hat{x}$, and $\hat{r}$ satisfy*

$$\begin{aligned} \| Ax - b \|_2 &= \min & r &= b - Ax \\ \| (A + \delta A)\hat{x} - (b + \delta b) \|_2 &= \min & \hat{r} &= (b + \delta b) - (A + \delta A)\hat{x} \end{aligned}$$

where A and δA are in $\mathbb{R}^{m\times n}$ with $m \geq n$ and $0 \neq b$ and δb are in $\mathbb{R}^m$. If

$$\epsilon = \max\left\{ \frac{\| \delta A \|_2}{\| A \|_2}, \frac{\| \delta b \|_2}{\| b \|_2} \right\} < \frac{\sigma_n(A)}{\sigma_1(A)}$$

and

$$\sin(\theta) = \frac{\rho_{LS}}{\| b \|_2} \neq 1$$

where $\rho_{LS} = \| Ax_{LS} - b \|_2$, then

$$\frac{\| \hat{x} - x \|_2}{\| x \|_2} \leq \epsilon \left\{ \frac{2\kappa_2(A)}{\cos(\theta)} + \tan(\theta)\kappa_2(A)^2 \right\} + O(\epsilon^2) \tag{5.3.8}$$

$$\frac{\| \hat{r} - r \|_2}{\| b \|_2} \leq \epsilon\,(1 + 2\kappa_2(A)) \min(1, m - n) + O(\epsilon^2)\,. \tag{5.3.9}$$

Proof. Let E and f be defined by $E = \delta A/\epsilon$ and $f = \delta b/\epsilon$. By hypothesis $\| \delta A \|_2 < \sigma_n(A)$ and so by Theorem 2.5.2 we have $\operatorname{rank}(A + tE) = n$ for all $t \in [0, \epsilon]$. It follows that the solution $x(t)$ to

$$(A + tE)^T(A + tE)x(t) \;=\; (A + tE)^T(b + tf) \tag{5.3.10}$$

is continuously differentiable for all $t \in [0, \epsilon]$. Since $x = x(0)$ and $\hat{x} = x(\epsilon)$, we have

$$\hat{x} = x \;+\; \epsilon\dot{x}(0) \;+\; O(\epsilon^2).$$

The assumptions $b \neq 0$ and $\sin(\theta) \neq 1$ ensure that x is nonzero and so

$$\frac{\| \hat{x} - x \|_2}{\| x \|_2} \;=\; \epsilon\,\frac{\| \dot{x}(0) \|_2}{\| x \|_2} \;+\; O(\epsilon^2). \tag{5.3.11}$$

In order to bound $\| \dot{x}(0) \|_2$, we differentiate (5.3.10) and set $t = 0$ in the result. This gives

$$E^T A x + A^T E x + A^T A \dot{x}(0) \;=\; A^T f + E^T b$$

i.e.,

$$\dot{x}(0) \;=\; (A^T A)^{-1} A^T (f - Ex) + (A^T A)^{-1} E^T r. \tag{5.3.12}$$

By substituting this result into (5.3.11), taking norms, and using the easily verified inequalities $\| f \|_2 \le \| b \|_2$ and $\| E \|_2 \le \| A \|_2$ we obtain

$$\begin{aligned}\frac{\| \hat{x} - x \|_2}{\| x \|_2} \;\le\;& \epsilon\left\{\| A \|_2 \| (A^T A)^{-1} A^T \|_2 \left(\frac{\| b \|_2}{\| A \|_2 \| x \|_2} \;+\; 1\right)\right.\\ &\left. +\; \frac{\rho_{LS}}{\| A \|_2 \| x \|_2}\| A \|_2^2 \, \| (A^T A)^{-1} \|_2\right\} \;+\; O(\epsilon^2).\end{aligned}$$

Since $A^T(Ax - b) = 0$, Ax is orthogonal to $Ax - b$ and so

$$\| b - Ax \|_2^2 \;+ \| Ax \|_2^2 \;= \| b \|_2^2 \,.$$

Thus,

$$\| A \|_2^2 \, \| x \|_2^2 \;\ge \| b \|_2^2 \;- \rho_{LS}^2$$

and so by using (5.3.7)

$$\frac{\| \hat{x} - x \|_2}{\| x \|_2} \;\le\; \epsilon\left\{\kappa_2(A)\left(\frac{1}{\cos(\theta)} \;+\; 1\right) \;+\; \kappa_2(A)^2\frac{\sin(\theta)}{\cos(\theta)}\right\} \;+\; O(\epsilon^2)$$

thereby establishing (5.3.8).

To prove (5.3.9), we define the differentiable vector function $r(t)$ by

$$r(t) = (b + tf) - (A + tE)x(t)$$

and observe that $r = r(0)$ and $\hat{r} = r(\epsilon)$. Using (5.3.12) it can be shown that

$$\dot{r}(0) = \left(I - A(A^TA)^{-1}A^T\right)(f - Ex) - A(A^TA)^{-1}E^Tr .$$

Since $\| \hat{r} - r \|_2 = \epsilon \| \dot{r}(0) \|_2 + O(\epsilon^2)$ we have

$$\begin{aligned}\frac{\| \hat{r} - r \|_2}{\| b \|_2} &= \epsilon\frac{\| \dot{r}(0) \|_2}{\| b \|_2} + O(\epsilon^2) \\ &\leq \epsilon\left\{\| I - A(A^TA)^{-1}A^T \|_2 \left(1 + \frac{\| A \|_2 \| x \|_2}{\| b \|_2}\right)\right. \\ &\quad \left. + \| A(A^TA)^{-1} \|_2 \| A \|_2 \frac{\rho_{LS}}{\| b \|_2}\right\} + O(\epsilon^2) .\end{aligned}$$

Inequality (5.3.9) now follows because

$$\| A \|_2 \| x \|_2 = \| A \|_2 \| A^+b \|_2 \leq \kappa_2(A) \| b \|_2 ,$$

$$\rho_{LS} = \| (I - A(A^TA)^{-1}A^T)b \|_2 \leq \| I - A(A^TA)^{-1}A^T \|_2 \| b \|_2 ,$$

and

$$\| (I - A(A^TA)^{-1}A^T \|_2 = \min(m - n, 1). \ \square$$

An interesting feature of the upper bound in (5.3.8) is the factor

$$\tan(\theta)\kappa_2(A)^2 = \frac{\rho_{LS}}{\sqrt{\| b \|_2^2 - \rho_{LS}^2}}\kappa_2(A)^2 .$$

Thus, in nonzero residual problems it is the square of the condition that measures the sensitivity of x_{LS}. In contrast, residual sensitivity depends just linearly on $\kappa_2(A)$. These dependencies are confirmed by Example 5.3.1.

5.3.8 Normal Equations Versus QR

It is instructive to compare the normal equation and QR approaches to the LS problem. Recall the following main points from our discussion:

- The sensitivity of the LS solution is roughly proportional to the quantity $\kappa_2(A) + \rho_{LS}\kappa_2(A)^2$.

- The method of normal equations produces an $\hat{x}_{LS}$ whose relative error depends on the square of the condition.

- The QR approach (Householder, Givens, careful MGS) solves a nearby LS problem and therefore produces a solution that has a relative error approximately given by $\mathbf{u}(\kappa_2(A) + \rho_{LS}\kappa_2(A)^2)$.

Thus, we may conclude that if ρ_{LS} is small and $\kappa_2(A)$ is large, then the method of normal equations does not solve a nearby problem and will usually render an LS solution that is less accurate than a stable QR approach. Conversely, the two methods produce comparably inaccurate results when applied to large residual, ill-conditioned problems.

Finally, we mention two other factors that figure in the debate about QR versus normal equations:

- The normal equations approach involves about half of the arithmetic when $m \gg n$ and does not require as much storage.
- QR approaches are applicable to a wider class of matrices because the Cholesky process applied to A^TA breaks down "before" the back substitution process on $Q^TA = R$.

At the very minimum, this discussion should convince you how difficult it can be to choose the "right" algorithm!

Problems

P5.3.1 Assume $A^TAx = A^Tb$, $(A^TA + F)\hat{x} = A^Tb$, and $2\| F \|_2 \le \sigma_n(A)^2$. Show that if $r = b - Ax$ and $\hat{r} = b - A\hat{x}$, then $\hat{r} - r = A(A^TA + F)^{-1}Fx$ and

$$\| \hat{r} - r \|_2 \le 2\kappa_2(A)\frac{\| F \|_2}{\| A \|_2}\| x \|_2.$$

P5.3.2 Assume that $A^TAx = A^Tb$ and that $A^TA\hat{x} = A^Tb + f$ where $\| f \|_2 \le cu\| A^T \|_2\| b \|_2$ and A has full column rank. Show that

$$\frac{\| x - \hat{x} \|_2}{\| x \|_2} \le cu\kappa_2(A)^2\frac{\| A^T \|_2\| b \|_2}{\| A^Tb \|}.$$

P5.3.3 Let $A \in \mathbb{R}^{m\times n}$ with $m > n$ and $y \in \mathbb{R}^m$ and define $\bar{A} = [A\ y] \in \mathbb{R}^{m\times(n+1)}$. Show that $\sigma_1(\bar{A}) \ge \sigma_1(A)$ and $\sigma_{n+1}(\bar{A}) \le \sigma_n(A)$. Thus, the condition grows if a column is added to a matrix.

P5.3.4 Let $A \in \mathbb{R}^{m\times n}$ $(m \ge n)$, $w \in \mathbb{R}^n$, and define

$$B = \begin{bmatrix} A \\ w^T \end{bmatrix}.$$

Show that $\sigma_n(B) \ge \sigma_n(A)$ and $\sigma_1(B) \le \sqrt{\| A \|_2^2 + \| w \|_2^2}$. Thus, the condition of a matrix may increase or decrease if a row is added.

P5.3.5 (Cline 1973) Suppose that $A \in \mathbb{R}^{m\times n}$ has rank n and that Gaussian elimination with partial pivoting is used to compute the factorization $PA = LU$, where $L \in \mathbb{R}^{m\times n}$ is unit lower triangular, $U \in \mathbb{R}^{n\times n}$ is upper triangular, and $P \in \mathbb{R}^{m\times m}$ is a permutation. Explain how the decomposition in P5.2.5 can be used to find a vector $z \in \mathbb{R}^n$ such that $\| Lz - Pb \|_2$ is minimized. Show that if $Ux = z$, then $\| Ax - b \|_2$ is minimum. Show that this method of solving the LS problem is more efficient than Householder QR from the flop point of view whenever $m \le 5n/3$.

P5.3.6 The matrix $C = (A^TA)^{-1}$, where rank$(A) = n$, arises in many statistical applications and is known as the *variance-covariance matrix*. Assume that the factorization

$A = QR$ is available. (a) Show $C = (R^TR)^{-1}$. (b) Give an algorithm for computing the diagonal of C that requires $n^3/3$ flops. (c) Show that

$$R = \begin{bmatrix} \alpha & v^T \\ 0 & S \end{bmatrix} \quad \Rightarrow \quad C = (R^TR)^{-1} = \begin{bmatrix} (1 + v^TC_1v)/\alpha^2 & -v^TC_1/\alpha \\ -C_1v/\alpha & C_1 \end{bmatrix}$$

where $C_1 = (S^TS)^{-1}$. (d) Using (c), give an algorithm that overwrites the upper triangular portion of R with the upper triangular portion of C. Your algorithm should require $2n^3/3$ flops.

P5.3.7 Suppose $A \in \mathbb{R}^{n\times n}$ is symmetric and that $r = b - Ax$ where $r, b, x \in \mathbb{R}^n$ and x is nonzero. Show how to compute a symmetric $E \in \mathbb{R}^{n\times n}$ with minimal Frobenius norm so that $(A + E)x = b$. Hint. Use the QR factorization of $[\,x\,,\ r\,]$ and note that $Ex = r \Rightarrow (Q^TEQ)(Q^Tx) = Q^Tr$.

P5.3.8 Show how to compute the nearest circulant matrix to a given Toeplitz matrix. Measure distance with the Frobenius norm.

Notes and References for Sec. 5.3

Our restriction to least squares approximation is not a vote against minimization in other norms. There are occasions when it is advisable to minimize $\| Ax - b \|_p$ for $p = 1$ and ∞. Some algorithms for doing this are described in

A.K. Cline (1976a). "A Descent Method for the Uniform Solution to Overdetermined Systems of Equations," *SIAM J. Num. Anal. 13*, 293–309.

R.H. Bartels, A.R. Conn, and C. Charalambous (1978). "On Cline's Direct Method for Solving Overdetermined Linear Systems in the L_∞ Sense," *SIAM J. Num. Anal. 15*, 255–70.

T.F. Coleman and Y. Li (1992). "A Globally and Quadratically Convergent Affine Scaling Method for Linear L_1 Problems," *Mathematical Programming, 56, Series A*, 189–222.

Y. Li (1993). "A Globally Convergent Method for L_p Problems," *SIAM J. Optimization 3*, 609–629.

Y. Zhang (1993). "A Primal-Dual Interior Point Approach for Computing the L_1 and L_∞ Solutions of Overdetermined Linear Systems," *J. Optimization Theory and Applications* 77, 323–341.

The use of Gauss transformations to solve the LS problem has attracted some attention because they are cheaper to use than Householder or Givens matrices. See

G. Peters and J.H. Wilkinson (1970). "The Least Squares Problem and Pseudo-Inverses," *Comp. J. 13*, 309–16.

A.K. Cline (1973). "An Elimination Method for the Solution of Linear Least Squares Problems," *SIAM J. Num. Anal. 10*, 283–89.

R.J. Plemmons (1974). "Linear Least Squares by Elimination and MGS," *J. Assoc. Comp. Mach. 21*, 581–85.

Important analyses of the LS problem and various solution approaches include

G.H. Golub and J.H. Wilkinson (1966). "Note on the Iterative Refinement of Least Squares Solution," *Numer. Math. 9*, 139–48.

A. van der Sluis (1975). "Stability of the Solutions of Linear Least Squares Problem," *Numer. Math. 23*, 241–54.

Y. Saad (1986). "On the Condition Number of Some Gram Matrices Arising from Least Squares Approximation in the Complex Plane," *Numer. Math. 48*, 337–348.

Å. Björck (1987). "Stability Analysis of the Method of Seminormal Equations," *Lin. Alg. and Its Applic. 88/89*, 31–48.

J. Gluchowska and A. Smoktunowicz (1990). "Solving the Linear Least Squares Problem with Very High Relative Accuracy," *Computing 45*, 345–354.

Å. Björck (1991). "Component-wise Perturbation Analysis and Error Bounds for Linear Least Squares Solutions," *BIT 31*, 238–244.

Å. Björck and C.C. Paige (1992). "Loss and Recapture of Orthogonality in the Modified Gram-Schmidt Algorithm," *SIAM J. Matrix Anal. Appl. 13*, 176–190.

B. Waldén, R. Karlson, J. Sun (1995). "Optimal Backward Perturbation Bounds for the Linear Least Squares Problem," *Numerical Lin. Alg. with Applic. 2*, 271–286.

The "seminormal" equations are given by $R^TRx = A^Tb$ where $A = QR$. In the above paper it is shown that by solving the seminormal equations an acceptable LS solution is obtained if one step of fixed precision iterative improvement is performed.

An Algol implementation of the MGS method for solving the LS problem appears in

F.L. Bauer (1965). "Elimination with Weighted Row Combinations for Solving Linear Equations and Least Squares Problems," *Numer. Math.* 7, 338–52. See also Wilkinson and Reinsch (1971, 119–33).

Least squares problems often have special structure which, of course, should be exploited.

M.G. Cox (1981). "The Least Squares Solution of Overdetermined Linear Equations having Band or Augmented Band Structure," *IMA J. Num. Anal. 1*, 3–22.

G. Cybenko (1984). "The Numerical Stability of the Lattice Algorithm for Least Squares Linear Prediction Problems," *BIT 24*, 441–455.

P.C. Hansen and H. Gesmar (1993). "Fast Orthogonal Decomposition of Rank-Deficient Toeplitz Matrices," *Numerical Algorithms 4*, 151–166.

The use of Householder matrices to solve sparse LS problems requires careful attention to avoid excessive fill-in.

J.K. Reid (1967). "A Note on the Least Squares Solution of a Band System of Linear Equations by Householder Reductions," *Comp. J. 10*, 188–89.

I.S. Duff and J.K. Reid (1976). "A Comparison of Some Methods for the Solution of Sparse Over-Determined Systems of Linear Equations," *J. Inst. Math. Applic. 17*, 267–80.

P.E. Gill and W. Murray (1976). "The Orthogonal Factorization of a Large Sparse Matrix," in *Sparse Matrix Computations*, ed. J.R. Bunch and D.J. Rose, Academic Press, New York, pp. 177-200.

L. Kaufman (1979). "Application of Dense Householder Transformations to a Sparse Matrix," *ACM Trans. Math. Soft. 5*, 442–51.

Although the computation of the QR factorization is more efficient with Householder reflections, there are some settings where the Givens approach is advantageous. For example, if A is sparse, then the careful application of Givens rotations can minimize fill-in.

I.S. Duff (1974). "Pivot Selection and Row Ordering in Givens Reduction on Sparse Matrices," *Computing 13*, 239–48.

J.A. George and M.T. Heath (1980). "Solution of Sparse Linear Least Squares Problems Using Givens Rotations," *Lin. Alg. and Its Applic. 34*, 69–83.

5.4 Other Orthogonal Factorizations

If A is rank deficient, then the QR factorization need not give a basis for ran(A). This problem can be corrected by computing the QR factorization of a column-permuted version of A, i.e., $A\Pi = QR$ where Π is a permutation.

The "data" in A can be compressed further if we permit right multiplication by a general orthogonal matrix Z:

$$Q^T AZ = T.$$

There are interesting choices for Q and Z and these, together with the column pivoted QR factorization, are discussed in this section.

5.4.1 Rank Deficiency: QR with Column Pivoting

If $A \in \mathbb{R}^{m\times n}$ and rank(A) $< n$, then the QR factorization does not necessarily produce an orthonormal basis for ran(A). For example, if A has three columns and

$$A = [\,a_1,\, a_2,\, a_3\,] = [\,q_1,\, q_2,\, q_3\,]\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

is its QR factorization, then rank(A) = 2 but ran(A) does not equal any of the subspaces span$\{q_1, q_2\}$, span$\{q_1, q_3\}$, or span$\{q_2, q_3\}$.

Fortunately, the Householder QR factorization procedure (Algorithm 5.2.1) can be modified in a simple way to produce an orthonormal basis for ran(A). The modified algorithm computes the factorization

$$Q^T A\Pi = \begin{array}{cc} \left[\begin{array}{cc} R_{11} & R_{12} \\ 0 & 0 \end{array}\right] & \begin{array}{c} r \\ m-r \end{array} \\ \begin{array}{cc} \;r\; & \;n-r \end{array} & \end{array} \qquad (5.4.1)$$

where r = rank(A), Q is orthogonal, R_{11} is upper triangular and nonsingular, and Π is a permutation. If we have the column partitionings $A\Pi = [\,a_{c_1}, \ldots, a_{c_n}\,]$ and $Q = [\,q_1, \ldots, q_m\,]$, then for $k = 1{:}n$ we have

$$a_{c_k} = \sum_{i=1}^{\min\{r,k\}} r_{ik} q_i \in \text{span}\{q_1, \ldots, q_r\}$$

implying

$$\text{ran}(A) = \text{span}\{q_1, \ldots, q_r\}.$$

The matrices Q and Π are products of Householder matrices and interchange matrices respectively. Assume for some k that we have computed

Householder matrices $H_1, \ldots, H_{k-1}$ and permutations $\Pi_1, \ldots, \Pi_{k-1}$ such that

$$(H_{k-1} \cdots H_1) A (\Pi_1 \cdots \Pi_{k-1}) = \tag{5.4.2}$$

$$R^{(k-1)} = \begin{array}{cc} \left[\begin{array}{cc} R_{11}^{(k-1)} & R_{12}^{(k-1)} \\ 0 & R_{22}^{(k-1)} \end{array}\right] & \begin{array}{c} k-1 \\ m-k+1 \end{array} \\ \begin{array}{cc} k-1 & n-k+1 \end{array} & \end{array}$$

where $R_{11}^{(k-1)}$ is a nonsingular and upper triangular matrix. Now suppose that

$$R_{22}^{(k-1)} = \left[z_k^{(k-1)}, \ldots, z_n^{(k-1)} \right]$$

is a column partitioning and let $p \geq k$ be the smallest index such that

$$\| z_p^{(k-1)} \|_2 = \max \left\{ \| z_k^{(k-1)} \|_2, \cdots, \| z_n^{(k-1)} \|_2 \right\} . \tag{5.4.3}$$

Note that if $k-1 = \text{rank}(A)$, then this maximum is zero and we are finished. Otherwise, let Π_k be the n-by-n identity with columns p and k interchanged and determine a Householder matrix H_k such that if $R^{(k)} = H_k R^{(k-1)} \Pi_k$, then $R^{(k)}(k+1{:}m, k) = 0$. In other words, Π_k moves the largest column in $R_{22}^{(k-1)}$ to the lead position and $\tilde{H}_k$ zeroes all of its subdiagonal components.

The column norms do not have to be recomputed at each stage if we exploit the property

$$Q^T z = \begin{array}{cc} \left[\begin{array}{c} \alpha \\ w \end{array}\right] & \begin{array}{c} 1 \\ s-1 \end{array} \end{array} \qquad \Longrightarrow \qquad \| w \|_2^2 = \| z \|_2^2 - \alpha^2,$$

which holds for any orthogonal matrix $Q \in \mathbb{R}^{s \times s}$. This reduces the overhead associated with column pivoting from $O(mn^2)$ flops to $O(mn)$ flops because we can get the new column norms by updating the old column norms, e.g.,

$$\| z^{(j)} \|_2^2 = \| z^{(j-1)} \|_2^2 - r_{kj}^2 .$$

Combining all of the above we obtain the following algorithm established by Businger and Golub (1965):

Algorithm 5.4.1 (Householder QR With Column Pivoting) Given $A \in \mathbb{R}^{m \times n}$ with $m \geq n$, the following algorithm computes $r = \text{rank}(A)$ and the factorization (5.4.1) with $Q = H_1 \cdots H_r$ and $\Pi = \Pi_1 \cdots \Pi_r$. The upper triangular part of A is overwritten by the upper triangular part of R and components $j+1{:}m$ of the jth Householder vector are stored in $A(j+1{:}m, j)$. The permutation Π is encoded in an integer vector *piv*. In particular, Π_j is the identity with rows j and $piv(j)$ interchanged.

```
for $j = 1{:}n$
    $c(j) = A(1{:}m,j)^T A(1{:}m,j)$
end
$r = 0$; $\tau = \max\{c(1),\ldots,c(n)\}$
Find smallest $k$ with $1 \le k \le n$ so $c(k) = \tau$
while $\tau > 0$
    $r = r + 1$
    $piv(r) = k$; $A(1{:}m,r) \leftrightarrow A(1{:}m,k)$; $c(r) \leftrightarrow c(k)$
    $[v,\beta] = \text{house}(A(r{:}m,r))$
    $A(r{:}m,r{:}n) = (I_{m-r+1} - \beta vv^T)A(r{:}m,r{:}n)$
    $A(r+1{:}m,r) = v(2{:}m-r+1)$
    for $i = r+1{:}n$
        $c(i) = c(i) - A(r,i)^2$
    end
    if $r < n$
        $\tau = \max\{c(r+1),\ldots,c(n)\}$
        Find smallest $k$ with $r+1 \le k \le n$ so $c(k) = \tau$.
    else
        $\tau = 0$
    end
end
```

This algorithm requires $4mnr - 2r^2(m+n) + 4r^3/3$ flops where $r = \text{rank}(A)$. As with the nonpivoting procedure, Algorithm 5.2.1, the orthogonal matrix Q is stored in factored form in the subdiagonal portion of A.

Example 5.4.1 If Algorithm 5.4.1 is applied to

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 5 & 6 \\ 1 & 8 & 9 \\ 1 & 11 & 12 \end{bmatrix},$$

then $\Pi = [e_3\ e_2\ e_1]$ and to three significant digits we obtain

$$A\Pi = QR = \begin{bmatrix} -.182 & -.816 & .514 & .191 \\ -.365 & .408 & -.827 & .129 \\ .548 & .000 & .113 & -.829 \\ -.730 & .408 & .200 & .510 \end{bmatrix} \begin{bmatrix} -16.4 & -14.600 & -1.820 \\ 0.0 & .816 & -.816 \\ 0.0 & .000 & 0.000 \end{bmatrix}.$$

5.4.2 Complete Orthogonal Decompositions

The matrix R produced by Algorithm 5.4.1 can be further reduced if it is post-multiplied by an appropriate sequence of Householder matrices. In particular, we can use Algorithm 5.2.1 to compute

$$Z_r \cdots Z_1 \begin{bmatrix} R_{11}^T \\ R_{12}^T \end{bmatrix} = \begin{bmatrix} T_{11}^T \\ 0 \end{bmatrix} \begin{array}{l} r \\ n-r \end{array} \qquad (5.4.4)$$

where the Z_i are Householder transformations and T_{11}^T is upper triangular. It then follows that

$$Q^TAZ = T = \begin{array}{cc|c} \multicolumn{2}{c}{\left[\begin{array}{cc} T_{11} & 0 \\ 0 & 0 \end{array}\right]} & \begin{array}{c} r \\ m-r \end{array} \\ r & n-r & \end{array} . \qquad (5.4.5)$$

where $Z = \Pi Z_1 \cdots Z_r$. We refer to any decomposition of this form as a *complete orthogonal decomposition*. Note that null(A) = ran$(Z(1{:}n, r+1{:}n))$. See P5.2.5 for details about the exploitation of structure in (5.4.4).

5.4.3 Bidiagonalization

Suppose $A \in \mathbb{R}^{m\times n}$ and $m \geq n$. We next show how to compute orthogonal U_B (m-by-m) and V_B (n-by-n) such that

$$U_B^TAV_B = \left[\begin{array}{ccccc} d_1 & f_1 & 0 & \cdots & 0 \\ 0 & d_2 & f_2 & & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & & d_{n-1} & f_{n-1} \\ 0 & \cdots & & 0 & d_n \\ \hline \multicolumn{5}{c}{0} \end{array}\right] . \qquad (5.4.6)$$

$U_B = U_1 \cdots U_n$ and $V_B = V_1 \cdots V_{n-2}$ can each be determined as a product of Householder matrices:

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \xrightarrow{U_1} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow{V_1}$$

$$\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow{U_2} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{V_2}$$

$$\begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{U_3} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \xrightarrow{U_4} \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & 0 \end{bmatrix} .$$

In general, U_k introduces zeros into the kth column, while V_k zeros the appropriate entries in row k. Overall we have:

Algorithm 5.4.2 (Householder Bidiagonalization) Given $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, the following algorithm overwrites A with $U_B^TAV_B = B$ where B is upper bidiagonal and $U_B = U_1 \cdots U_n$ and $V_B = V_1 \cdots V_{n-2}$. The essential part of U_j's Householder vector is stored in $A(j+1{:}m, j)$ and the essential part of V_j's Householder vector is stored in $A(j, j+2{:}n)$.

```
for $j = 1{:}n$
    $[v, \beta] = \text{house}(A(j{:}m, j))$
    $A(j{:}m, j{:}n) = (I_{m-j+1} - \beta vv^T)A(j{:}m, j{:}n)$
    $A(j+1{:}m, j) = v(2{:}m-j+1)$
    if $j \leq n-2$
        $[v, \beta] = \text{house}(A(j, j+1{:}n)^T)$
        $A(j{:}m, j+1{:}n) = A(j{:}m, j+1{:}n)(I_{n-j} - \beta vv^T)$
        $A(j, j+2{:}n) = v(2{:}n-j)^T$
    end
end
```

This algorithm requires $4mn^2 - 4n^3/3$ flops. Such a technique is used in Golub and Kahan (1965), where bidiagonalization is first described. If the matrices U_B and V_B are explicitly desired, then they can be accumulated in $4m^2n - 4n^3/3$ and $4n^3/3$ flops, respectively. The bidiagonalization of A is related to the tridiagonalization of A^TA. See §8.2.1.

Example 5.4.2 If Algorithm 5.4.2 is applied to

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix},$$

then to three significant digits we obtain

$$\hat{B} = \begin{bmatrix} 12.8 & 21.8 & 0 \\ 0 & 2.24 & -.613 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \hat{V}_B = \begin{bmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & -.667 & -.745 \\ 0.00 & -.745 & .667 \end{bmatrix}$$

$$\hat{U}_B = \begin{bmatrix} -.0776 & -.833 & .392 & -.383 \\ -.3110 & -.451 & -.238 & .802 \\ -.5430 & -.069 & .701 & -.457 \\ -.7760 & .312 & .547 & .037 \end{bmatrix}.$$

5.4.4 R-Bidiagonalization

A faster method of bidiagonalizing when $m \gg n$ results if we upper triangularize A first before applying Algorithm 5.4.2. In particular, suppose we compute an orthogonal $Q \in \mathbb{R}^{m\times m}$ such that

$$Q^TA = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

is upper triangular. We then bidiagonalize the square matrix R_1,

$$U_R^TR_1V_B = B_1 .$$

Here U_R and V_B are n-by-n orthogonal and B_1 is n-by-n upper bidiagonal. If $U_B = Q\,\text{diag}\,(U_R, I_{m-n})$ then

$$U^TAV = \begin{bmatrix} B_1 \\ 0 \end{bmatrix} \equiv B$$

is a bidiagonalization of A.

The idea of computing the bidiagonalization in this manner is mentioned in Lawson and Hanson (1974, p.119) and more fully analyzed in Chan (1982a). We refer to this method as R-bidiagonalization. By comparing its flop count ($2mn^2+2n^3$) with that for Algorithm 5.4.2 ($4mn^2-4n^3/3$) we see that it involves fewer computations (approximately) whenever $m \geq 5n/3$.

5.4.5 The SVD and its Computation

Once the bidiagonalization of A has been achieved, the next step in the Golub-Reinsch SVD algorithm is to zero the superdiagonal elements in B. This is an iterative process and is accomplished by an algorithm due to Golub and Kahan (1965). Unfortunately, we must defer our discussion of this iteration until §8.6 as it requires an understanding of the symmetric eigenvalue problem. Suffice it to say here that it computes orthogonal matrices U_Σ and V_Σ such that

$$U_\Sigma^TBV_\Sigma = \Sigma = \text{diag}(\sigma_1,\ldots,\sigma_n) \in \mathbb{R}^{m\times n} .$$

By defining $U = U_BU_\Sigma$ and $V = V_BV_\Sigma$ we see that $U^TAV = \Sigma$ is the SVD of A. The flop counts associated with this portion of the algorithm depend upon "how much" of the SVD is required. For example, when solving the LS problem, U^T need never be explicitly formed but merely applied to b as it is developed. In other applications, only the matrix $U_1 = U(:,1{:}n)$ is required. Altogether there are six possibilities and the total amount of work required by the SVD algorithm in each case is summarized in the table below. Because of the two possible bidiagonalization schemes, there are two columns of flop counts. If the bidiagonalization is achieved via Algorithm 5.4.2, the Golub-Reinsch (1970) SVD algorithm results, while if R-bidiagonalization is invoked we obtain the R-SVD algorithm detailed in Chan (1982a). By comparing the entries in this table (which are meant only as approximate estimates of work), we conclude that the R-SVD approach is more efficient unless $m \approx n$.

Required	Golub-Reinsch SVD	R-SVD
Σ	$4mn^2 - 4n^3/3$	$2mn^2 + 2n^3$
Σ, V	$4mn^2 + 8n^3$	$2mn^2 + 11n^3$
Σ, U	$4m^2n - 8mn^2$	$4m^2n + 13n^3$
Σ, U_1	$14mn^2 - 2n^3$	$6mn^2 + 11n^3$
Σ, U, V	$4m^2n + 8mn^2 + 9n^3$	$4m^2n + 22n^3$
Σ, U_1, V	$14mn^2 + 8n^3$	$6mn^2 + 20n^3$

Problems

P5.4.1 Suppose $A \in \mathbb{R}^{m\times n}$ with $m < n$. Give an algorithm for computing the factorization

$$U^TAV = [\,B\ O\,]$$

where B is an m-by-m upper bidiagonal matrix. (Hint: Obtain the form

$$\begin{bmatrix} \times & \times & 0 & 0 & 0 & 0 \\ 0 & \times & \times & 0 & 0 & 0 \\ 0 & 0 & \times & \times & 0 & 0 \\ 0 & 0 & 0 & \times & \times & 0 \end{bmatrix}.$$

using Householder matrices and then "chase" the $(m, m+1)$ entry up the $(m+1)$st column by applying Givens rotations from the right.)

P5.4.2 Show how to efficiently bidiagonalize an n-by-n upper triangular matrix using Givens rotations.

P5.4.3 Show how to upper bidiagonalize a tridiagonal matrix $T \in \mathbb{R}^{n\times n}$ using Givens rotations.

P5.4.4 Let $A \in \mathbb{R}^{m\times n}$ and assume that $0 \neq v$ satisfies $\| Av \|_2 = \sigma_n(A)\| v \|_2$ Let Π be a permutation such that if $\Pi^T v = w$, then $|w_n| = \| w \|_\infty$. Show that if $A\Pi = QR$ is the QR factorization of $A\Pi$, then $|r_{nn}| \leq \sqrt{n}\sigma_n(A)$. Thus, there always exists a permutation Π such that the QR factorization of $A\Pi$ "displays" near rank deficiency.

P5.4.5 Let $x, y \in \mathbb{R}^m$ and $Q \in \mathbb{R}^{m\times m}$ be given with Q orthogonal. Show that if

$$Q^Tx = \begin{array}{c} \begin{bmatrix} \alpha \\ u \end{bmatrix} \end{array}\begin{array}{c} 1 \\ m-1 \end{array} \qquad Q^Ty = \begin{bmatrix} \beta \\ v \end{bmatrix}\begin{array}{c} 1 \\ m-1 \end{array}$$

then $u^Tv = x^Ty - \alpha\beta$.

P5.4.6 Let $A = [\,a_1, \ldots, a_n\,] \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ be given. For any subset of A's columns $\{a_{c_1}, \ldots, a_{c_k}\}$ define

$$\text{res}\,[\,a_{c_1}, \ldots, a_{c_k}\,] = \min_{x \in \mathbb{R}^k} \| [\,a_{c_1}, \ldots, a_{c_k}\,]x - b \|_2$$

Describe an alternative pivot selection procedure for Algorithm 5.4.1 such that if $QR = A\Pi = [\,a_{c_1}, \ldots, a_{c_n}\,]$ in the final factorization, then for $k = 1{:}n$:

$$\text{res}\,[\,a_{c_1}, \ldots, a_{c_k}\,] = \min_{i \geq k} \text{res}[a_{c_1}, \ldots, a_{c_{k-1}}, a_{c_i}]$$

Notes and References for Sec. 5.4

Aspects of the complete orthogonal decomposition are discussed in

R.J. Hanson and C.L. Lawson (1969). "Extensions and Applications of the Householder Algorithm for Solving Linear Least Square Problems," *Math. Comp.* *23*, 787–812.

P.A. Wedin (1973). "On the Almost Rank-Deficient Case of the Least Squares Problem," *BIT 13*, 344–54.

G.H. Golub and V. Pereyra (1976). "Differentiation of Pseudo-Inverses, Separable Nonlinear Least Squares Problems and Other Tales," in *Generalized Inverses and Applications* , ed. M.Z. Nashed, Academic Press, New York, pp. 303–24.

The computation of the SVD is detailed in §8.6. But here are some of the standard references concerned with its calculation:

G.H. Golub and W. Kahan (1965). "Calculating the Singular Values and Pseudo-Inverse of a Matrix," *SIAM J. Num. Anal.* *2*, 205–24.

P.A. Businger and G.H. Golub (1969). "Algorithm 358: Singular Value Decomposition of the Complex Matrix," *Comm. ACM 12*, 564–65.

G.H. Golub and C. Reinsch (1970). "Singular Value Decomposition and Least Squares Solutions," *Numer. Math. 14*, 403–20. See also Wilkinson and Reinsch(1971, pp. 1334–51).

T.F. Chan (1982). "An Improved Algorithm for Computing the Singular Value Decomposition," *ACM Trans. Math. Soft. 8*, 72–83.

QR with column pivoting was first discussed in

P.A. Businger and G.H. Golub (1965). "Linear Least Squares Solutions by Householder Transformations," *Numer. Math.* *7*, 269–76. See also Wilkinson and Reinsch (1971, pp. 11–18).

Knowing when to stop in the algorithm is difficult. In questions of rank deficiency, it is helpful to obtain information about the smallest singular value of the upper triangular matrix R. This can be done using the techniques of §3.5.4 or those that are discussed in

I. Karasalo (1974). "A Criterion for Truncation of the QR Decomposition Algorithm for the Singular Linear Least Squares Problem," *BIT 14*, 156–66.

N. Anderson and I. Karasalo (1975). "On Computing Bounds for the Least Singular Value of a Triangular Matrix," *BIT 15*, 1–4.

Other aspects of rank estimation with QR are discussed in

L.V. Foster (1986). "Rank and Null Space Calculations Using Matrix Decomposition without Column Interchanges," *Lin. Alg. and Its Applic.* *74*, 47–71.

T.F. Chan (1987). "Rank Revealing QR Factorizations," *Lin. Alg. and Its Applic. 88/89*, 67–82.

T.F. Chan and P. Hansen (1992). "Some Applications of the Rank Revealing QR Factorization," *SIAM J. Sci. and Stat. Comp. 13*, 727–741.

J.L. Barlow and U.B. Vemulapati (1992). "Rank Detection Methods for Sparse Matrices," *SIAM J. Matrix Anal. Appl. 13*, 1279–1297.

T-M. Hwang, W-W. Lin, and E.K. Yang (1992). "Rank-Revealing LU Factorizations," *Lin. Alg. and Its Applic. 175*, 115–141.

C.H. Bischof and P.C. Hansen (1992). "A Block Algorithm for Computing Rank-Revealing QR Factorizations," *Numerical Algorithms 2*, 371-392.

S. Chandrasekaren and I.C.F. Ipsen (1994). "On Rank-Revealing Factorizations," *SIAM J. Matrix Anal. Appl. 15*, 592–622.

R.D. Fierro and P.C. Hansen (1995). "Accuracy of TSVD Solutions Computed from Rank-Revealing Decompositions," *Numer. Math. 70*, 453–472.

5.5 The Rank Deficient LS Problem

If A is rank deficient, then there are an infinite number of solutions to the LS problem and we must resort to special techniques. These techniques must address the difficult problem of numerical rank determination.

After some SVD preliminaries, we show how QR with column pivoting can be used to determine a minimizer x_B with the property that Ax_B is a linear combination of $r = \text{rank}(A)$ columns. We then discuss the minimum 2-norm solution that can be obtained from the SVD.

5.5.1 The Minimum Norm Solution

Suppose $A \in \mathbb{R}^{m\times n}$ and $\text{rank}(A) = r < n$. The rank deficient LS problem has an infinite number of solutions, for if x is a minimizer and $z \in \text{null}(A)$ then $x + z$ is also a minimizer. The set of all minimizers

$$\mathcal{X} = \{x \in \mathbb{R}^n : \| Ax - b \|_2 = \min \}$$

is convex, for if $x_1, x_2 \in \mathcal{X}$ and $\lambda \in [0, 1]$, then

$$\begin{aligned} \| A(\lambda x_1 + (1-\lambda)x_2) - b \|_2 &\le \lambda \| Ax_1 - b \|_2 + (1-\lambda) \| Ax_2 - b \|_2 \\ &= \min \| Ax - b \|_2 . \end{aligned}$$

Thus, $\lambda x_1 + (1-\lambda)x_2 \in \mathcal{X}$. It follows that $\mathcal{X}$ has a unique element having minimum 2-norm and we denote this solution by x_{LS}. (Note that in the full rank case, there is only one LS solution and so it must have minimal 2-norm. Thus, we are consistent with the notation in §5.3.)

5.5.2 Complete Orthogonal Factorization and x_{LS}

Any complete orthogonal factorization can be used to compute x_{LS}. In particular, if Q and Z are orthogonal matrices such that

$$Q^TAZ = T = \begin{array}{cc} \left[\begin{array}{cc} T_{11} & 0 \\ 0 & 0 \end{array}\right] & \begin{array}{c} r \\ m-r \end{array} \\ \begin{array}{cc} r & n-r \end{array} & \end{array} \qquad r = \text{rank}(A)$$

then

$$\| Ax - b \|_2^2 = \| (Q^TAZ)Z^Tx - Q^Tb \|_2^2 = \| T_{11}w - c \|_2^2 + \| d \|_2^2$$

where

$$Z^Tx = \left[\begin{array}{c} w \\ y \end{array}\right] \begin{array}{c} r \\ n-r \end{array} \qquad Q^Tb = \left[\begin{array}{c} c \\ d \end{array}\right] \begin{array}{c} r \\ m-r \end{array} .$$

Clearly, if x is to minimize the sum of squares, then we must have $w = T_{11}^{-1}c$. For x to have minimal 2-norm, y must be zero, and thus,

$$x_{LS} = Z \left[\begin{array}{c} T_{11}^{-1}c \\ 0 \end{array}\right] .$$

5.5.3 The SVD and the LS Problem

Of course, the SVD is a particularly revealing complete orthogonal decomposition. It provides a neat expression for x_{LS} and the norm of the minimum residual $\rho_{LS} = \| Ax_{LS} - b \|_2$.

Theorem 5.5.1 *Suppose $U^TAV = \Sigma$ is the SVD of $A \in \mathbb{R}^{m\times n}$ with $r =$* rank(A). *If $U = [\, u_1, \ldots, u_m \,]$ and $V = [\, v_1, \ldots, v_n \,]$ are column partitionings and $b \in \mathbb{R}^m$, then*

$$x_{LS} = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i \tag{5.5.1}$$

minimizes $\| Ax - b \|_2$ and has the smallest 2-norm of all minimizers. Moreover

$$\rho_{LS}^2 = \| Ax_{LS} - b \|_2^2 = \sum_{i=r+1}^{m} (u_i^T b)^2. \tag{5.5.2}$$

Proof. For any $x \in \mathbb{R}^n$ we have:

$$\begin{aligned} \| Ax - b \|_2^2 &= \| (U^TAV)(V^Tx) - U^Tb \|_2^2 = \| \Sigma\alpha - U^Tb \|_2^2 \\ &= \sum_{i=1}^{r} (\sigma_i \alpha_i - u_i^T b)^2 + \sum_{i=r+1}^{m} (u_i^T b)^2 \end{aligned}$$

where $\alpha = V^Tx$. Clearly, if x solves the LS problem, then $\alpha_i = (u_i^T b/\sigma_i)$ for $i = 1{:}r$. If we set $\alpha(r+1{:}n) = 0$, then the resulting x clearly has minimal 2-norm. □

5.5.4 The Pseudo-Inverse

Note that if we define the matrix $A^+ \in \mathbb{R}^{n\times m}$ by $A^+ = V\Sigma^+U^T$ where

$$\Sigma^+ = \text{diag}\left(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_r}, 0, \ldots, 0\right) \in \mathbb{R}^{n\times m} \qquad r = \text{rank}(A)$$

then $x_{LS} = A^+b$ and $\rho_{LS} = \| (I - AA^+)b \|_2$. A^+ is referred to as the *pseudo-inverse* of A. It is the unique minimal Frobenius norm solution to the problem

$$\min_{X \in \mathbb{R}^{n\times m}} \| AX - I_m \|_F . \tag{5.5.3}$$

If rank$(A) = n$, then $A^+ = (A^TA)^{-1}A^T$, while if $m = n =$ rank(A), then $A^+ = A^{-1}$. Typically, A^+ is defined to be the unique matrix $X \in \mathbb{R}^{n\times m}$ that satisfies the four *Moore-Penrose conditions:*

$$\begin{array}{llll} \text{(i)} & AXA = A & \text{(iii)} & (AX)^T = AX \\ \text{(ii)} & XAX = X & \text{(iv)} & (XA)^T = XA. \end{array}$$

These conditions amount to the requirement that AA^+ and A^+A be orthogonal projections onto $\text{ran}(A)$ and $\text{ran}(A^T)$, respectively. Indeed, $AA^+ = U_1U_1^T$ where $U_1 = U(1{:}m, 1{:}r)$ and $A^+A = V_1V_1^T$ where $V_1 = V(1{:}n, 1{:}r)$.

5.5.5 Some Sensitivity Issues

In §5.3 we examined the sensitivity of the full rank LS problem. The behavior of x_{LS} in this situation is summarized in Theorem 5.3.1. If we drop the full rank assumptions then x_{LS} is not even a continuous function of the data and small changes in A and b can induce arbitrarily large changes in $x_{LS} = A^+b$. The easiest way to see this is to consider the behavior of the pseudo inverse. If A and δA are in $\mathbb{R}^{m\times n}$, then Wedin (1973) and Stewart (1975) show that

$$\| (A+\delta A)^+ - A^+ \|_F \le 2\| \delta A \|_F \max\left\{ \| A^+ \|_2^2 \;,\; \| (A+\delta A)^+ \|_2^2 \right\}.$$

This inequality is a generalization of Theorem 2.3.4 in which perturbations in the matrix inverse are bounded. However, unlike the square nonsingular case, the upper bound does not necessarily tend to zero as δA tends to zero. If

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \text{and} \qquad \delta A = \begin{bmatrix} 0 & 0 \\ 0 & \epsilon \\ 0 & 0 \end{bmatrix}$$

then

$$A^+ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \text{and} \qquad (A+\delta A)^+ = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1/\epsilon & 0 \end{bmatrix}$$

and $\| A^+ - (A+\delta A)^+ \|_2 = 1/\epsilon$. The numerical determination of an LS minimizer in the presence of such discontinuities is a major challenge.

5.5.6 QR with Column Pivoting and Basic Solutions

Suppose $A \in \mathbb{R}^{m\times n}$ has rank r. QR with column pivoting (Algorithm 5.4.1) produces the factorization $A\Pi = QR$ where

$$R = \begin{array}{cc} \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} & \begin{array}{c} r \\ m-r \end{array} \\ \begin{array}{cc} \;\; r & \; n-r \end{array} & \end{array} .$$

Given this reduction, the LS problem can be readily solved. Indeed, for any $x \in \mathbb{R}^n$ we have

$$\begin{aligned} \| Ax - b \|_2^2 &= \| (Q^TA\Pi)(\Pi^Tx) - (Q^Tb) \|_2^2 \\ &= \| R_{11}y - (c - R_{12}z) \|_2^2 + \| d \|_2^2 , \end{aligned}$$

where

$$\Pi^T x = \begin{array}{c|c} \left[\begin{array}{c} y \\ z \end{array}\right] & \begin{array}{c} r \\ n-r \end{array} \end{array} \quad \text{and} \quad Q^T b = \begin{array}{c|c} \left[\begin{array}{c} c \\ d \end{array}\right] & \begin{array}{c} r \\ m-r \end{array} \end{array} .$$

Thus, if x is an LS minimizer, then we must have

$$x = \Pi \left[\begin{array}{c} R_{11}^{-1}(c - R_{12}z) \\ z \end{array} \right] .$$

If z is set to zero in this expression, then we obtain the *basic solution*

$$x_B = \Pi \left[\begin{array}{c} R_{11}^{-1}c \\ 0 \end{array} \right] .$$

Notice that x_B has at most r nonzero components and so Ax_B involves a subset of A's columns.

The basic solution is not the minimal 2-norm solution unless the submatrix R_{12} is zero since

$$\| x_{LS} \|_2 = \min_{z \in \mathbb{R}^{n-r}} \left\| x_B - \Pi \left[\begin{array}{c} R_{11}^{-1}R_{12} \\ -I_{n-r} \end{array} \right] z \right\|_2 . \qquad (5.5.4)$$

Indeed, this characterization of $\| x_{LS} \|_2$ can be used to show

$$1 \le \frac{\| x_B \|_2}{\| x_{LS} \|_2} \le \sqrt{1 + \| R_{11}^{-1}R_{12} \|_2^2} . \qquad (5.5.5)$$

See Golub and Pereyra (1976) for details.

5.5.7 Numerical Rank Determination with $A\Pi = QR$

If Algorithm 5.4.1 is used to compute x_B, then care must be exercised in the determination of rank(A). In order to appreciate the difficulty of this, suppose

$$fl(H_k \cdots H_1 A \Pi_1 \cdots \Pi_k) = \hat{R}^{(k)} = \begin{array}{cc} \left[\begin{array}{cc} \hat{R}_{11}^{(k)} & \hat{R}_{12}^{(k)} \\ 0 & \hat{R}_{22}^{(k)} \end{array}\right] & \begin{array}{c} k \\ m-k \end{array} \\ \begin{array}{cc} k & n-k \end{array} & \end{array}$$

is the matrix computed after k steps of the algorithm have been executed in floating point. Suppose rank$(A) = k$. Because of roundoff error, $\hat{R}_{22}^{(k)}$ will not be exactly zero. However, if $\hat{R}_{22}^{(k)}$ is suitably small in norm then it is reasonable to terminate the reduction and declare A to have rank k. A typical termination criteria might be

$$\| \hat{R}_{22}^{(k)} \|_2 \le \epsilon_1 \| A \|_2 \qquad (5.5.6)$$

for some small machine-dependent parameter ϵ_1. In view of the roundoff properties associated with Householder matrix computation (cf. §5.1.12), we know that $\hat{R}^{(k)}$ is the exact R factor of a matrix $A + E_k$, where

$$\| E_k \|_2 \leq \epsilon_2 \| A \|_2 \qquad \epsilon_2 = O(\mathbf{u}).$$

Using Theorem 2.5.2 we have

$$\sigma_{k+1}(A + E_k) = \sigma_{k+1}(\hat{R}^{(k)}) \;\leq\; \| \hat{R}_{22}^{(k)} \|_2 .$$

Since $\sigma_{k+1}(A) \leq \sigma_{k+1}(A + E_k) + \| E_k \|_2$, it follows that

$$\sigma_{k+1}(A) \leq (\epsilon_1 + \epsilon_2) \| A \|_2.$$

In other words, a relative perturbation of $O(\epsilon_1 + \epsilon_2)$ in A can yield a rank-k matrix. With this termination criterion, we conclude that QR with column pivoting "discovers" rank degeneracy if in the course of the reduction $\hat{R}_{22}^{(k)}$ is small for some $k < n$.

Unfortunately, this is not always the case. A matrix can be nearly rank deficient without a single $\hat{R}_{22}^{(k)}$ being particularly small. Thus, QR with column pivoting *by itself* is not entirely reliable as a method for detecting near rank deficiency. However, if a good condition estimator is applied to R it is practically impossible for near rank deficiency to go unnoticed.

Example 5.5.1 Let $T_n(c)$ be the matrix

$$T_n(c) \;=\; \text{diag}(1, s, \ldots, s^{n-1}) \begin{bmatrix} 1 & -c & -c & \cdots & -c \\ 0 & 1 & -c & \cdots & -c \\ & & \ddots & & \vdots & \vdots \\ \vdots & & & 1 & -c \\ 0 & & \cdots & & 1 \end{bmatrix}$$

with $c^2 + s^2 = 1$ with $c, s > 0$ (See Lawson and Hanson (1974, p.31).) These matrices are unaltered by Algorithm 5.4.1 and thus $\| R_{22}^{(k)} \|_2 \geq s^{n-1}$ for $k = 1{:}n-1$. This inequality implies (for example) that the matrix $T_{100}(.2)$ has no particularly small trailing principal submatrix since $s^{99} \approx .13$. However, it can be shown that $\sigma_n = O(10^{-8})$.

5.5.8 Numerical Rank and the SVD

We now focus our attention on the ability of the SVD to handle rank-deficiency in the presence of roundoff. Recall that if $A = U\Sigma V^T$ is the SVD of A, then

$$x_{LS} = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i \tag{5.5.7}$$

where $r = \text{rank}(A)$. Denote the computed versions of U, V, and $\Sigma = \text{diag}(\sigma_i)$ by $\hat{U}$, $\hat{V}$, and $\hat{\Sigma} = \text{diag}(\hat{\sigma}_i)$. Assume that both sequences of singular values range from largest to smallest. For a reasonably implemented SVD algorithm it can be shown that

$$\hat{U} = W + \Delta U \qquad W^TW = I_m \qquad \| \Delta U \|_2 \le \epsilon \tag{5.5.8}$$

$$\hat{V} = Z + \Delta V \qquad Z^TZ = I_n \qquad \| \Delta V \|_2 \le \epsilon \tag{5.5.9}$$

$$\hat{\Sigma} = W^T(A + \Delta A)Z \qquad \| \Delta A \|_2 \le \epsilon \| A \|_2 \tag{5.5.10}$$

where ϵ is a small multiple of u, the machine precision. In plain English, the SVD algorithm computes the singular values of a "nearby" matrix $A + \Delta A$.

Note that $\hat{U}$ and $\hat{V}$ are not necessarily close to their exact counterparts. However, we can show that $\hat{\sigma}_k$ is close to σ_k. Using (5.5.10) and Theorem 2.5.2 we have

$$\begin{aligned} \sigma_k &= \min_{\text{rank}(B)=k-1} \| A - B \|_2 \\ &= \min_{\text{rank}(B)=k-1} \| (\hat{\Sigma} - B) - W^T(\Delta A)Z \|_2 . \end{aligned}$$

Since $\| W^T(\Delta A)Z \|_2 \le \epsilon \| A \|_2 = \epsilon\sigma_1$ and

$$\min_{\text{rank}(B)=k-1} \| \hat{\Sigma}_k - B \|_2 = \hat{\sigma}_k$$

it follows that $|\sigma_k - \hat{\sigma}_k| \le \epsilon\sigma_1$ for $k = 1{:}n$. Thus, if A has rank r then we can expect $n - r$ of the computed singular values to be small. Near rank deficiency in A cannot escape detection when the SVD of A is computed.

Example 5.5.2 For the matrix $T_{100}(.2)$ in Example 5.5.1, $\sigma_n \approx .367 \cdot 10^{-8}$.

One approach to estimating $r = \text{rank}(A)$ from the computed singular values is to have a tolerance $\delta > 0$ and a convention that A has "numerical rank" $\hat{r}$ if the $\hat{\sigma}_i$ satisfy

$$\hat{\sigma}_1 \ge \cdots \ge \hat{\sigma}_{\hat{r}} > \delta \ge \hat{\sigma}_{\hat{r}+1} \ge \cdots \ge \hat{\sigma}_n .$$

The tolerance δ should be consistent with the machine precision, e.g. $\delta = \text{u}\| A \|_\infty$. However, if the general level of relative error in the data is larger than u, then δ should be correspondingly bigger, e.g., $\delta = 10^{-2}\| A \|_\infty$ if the entries in A are correct to two digits.

If $\hat{r}$ is accepted as the numerical rank then we can regard

$$x_{\hat{r}} = \sum_{i=1}^{\hat{r}} \frac{\hat{u}_i^T b}{\hat{\sigma}_i} \hat{v}_i$$

as an approximation to x_{LS}. Since $\| x_{\hat{r}} \|_2 \approx 1/\sigma_{\hat{r}} \leq 1/\delta$ then δ may also be chosen with the intention of producing an approximate LS solution with suitably small norm. In §12.1, we discuss more sophisticated methods for doing this.

If $\hat{\sigma}_{\hat{r}} \gg \delta$, then we have reason to be comfortable with $x_{\hat{r}}$ because A can then be unambiguously regarded as a rank($A_{\hat{r}}$) matrix (modulo δ).

On the other hand, $\{\hat{\sigma}_1, \ldots, \hat{\sigma}_n\}$ might not clearly split into subsets of small and large singular values, making the determination of $\hat{r}$ by this means somewhat arbitrary. This leads to more complicated methods for estimating rank which we now discuss in the context of the LS problem.

For example, suppose $r = n$, and assume for the moment that $\Delta A = 0$ in (5.5.10). Thus $\sigma_i = \hat{\sigma}_i$ for $i = 1{:}n$. Denote the ith columns of the matrices $\hat{U}$, $\hat{W}$, $\hat{V}$, and Z by u_i, w_i, v_i, and z_i, respectively. Subtracting $x_{\hat{r}}$ from x_{LS} and taking norms we obtain

$$\| x_{\hat{r}} - x_{LS} \|_2 \leq \sum_{i=1}^{\hat{r}} \frac{\| (w_i^T b) z_i - (u_i^T b) v_i \|_2}{\sigma_i} + \sqrt{\sum_{i=\hat{r}+1}^{n} \left(\frac{w_i^T b}{\sigma_i} \right)^2} .$$

From (5.5.8) and (5.5.9) it is easy to verify that

$$\| (w_i^T b) z_i - (u_i^T b) v_i \|_2 \leq 2(1+\epsilon)\epsilon \| b \|_2 \qquad (5.5.11)$$

and therefore

$$\| x_{\hat{r}} - x_{LS} \|_2 \leq \frac{\hat{r}}{\sigma_{\hat{r}}} 2(1+\epsilon)\epsilon \| b \|_2 + \sqrt{\sum_{i=\hat{r}+1}^{n} \left(\frac{w_i^T b}{\sigma_i} \right)^2} .$$

The parameter $\hat{r}$ can be determined as that integer which minimizes the upper bound. Notice that the first term in the bound increases with $\hat{r}$, while the second decreases.

On occasions when minimizing the residual is more important than accuracy in the solution, we can determine $\hat{r}$ on the basis of how close we surmise $\| b - Ax_{\hat{r}} \|_2$ is to the true minimum. Paralleling the above analysis, it can be shown that

$$\| b - Ax_{\hat{r}} \|_2 - \| b - Ax_{LS} \|_2 \leq (n - \hat{r}) \| b \|_2 + \epsilon \| b \|_2 \left(\hat{r} + \frac{\hat{\sigma}_1}{\hat{\sigma}_{\hat{r}}}(1+\epsilon) \right) .$$

Again $\hat{r}$ could be chosen to minimize the upper bound. See Varah (1973) for practical details and also the LAPACK manual.

5.5.9 Some Comparisons

As we mentioned, when solving the LS problem via the SVD, only Σ and V have to be computed. The following table compares the efficiency of this approach with the other algorithms that we have presented.

LS Algorithm	Flop Count
Normal Equations	$mn^2 + n^3/3$
Householder Orthogonalization	$2mn^2 - 2n^3/3$
Modified Gram Schmidt	$2mn^2$
Givens Orthogonalization	$3mn^2 - n^3$
Householder Bidiagonalization	$4mn^2 - 4n^3/2$
R-Bidiagonalization	$2mn^2 + 2n^3$
Golub-Reinsch SVD	$4mn^2 + 8n^3$
R-SVD	$2mn^2 + 11n^3$

Problems

P5.5.1 Show that if

$$A = \begin{array}{cc} \left[\begin{array}{cc} T & S \\ 0 & 0 \end{array}\right] & \begin{array}{c} r \\ m-r \end{array} \\ \begin{array}{cc} r & n-r \end{array} & \end{array}$$

where $r = \text{rank}(A)$ and T is nonsingular, then

$$X = \begin{array}{cc} \left[\begin{array}{cc} T^{-1} & 0 \\ 0 & 0 \end{array}\right] & \begin{array}{c} r \\ n-r \end{array} \\ \begin{array}{cc} r & m-r \end{array} & \end{array}$$

satisfies $AXA = A$ and $(AX)^T = (AX)$. In this case, we say that X is a (1,3) *pseudo-inverse* of A. Show that for general A, $x_B = Xb$ where X is a (1,3) pseudo-inverse of A.

P5.5.2 Define $B(\lambda) \in \mathbb{R}^{n\times m}$ by $B(\lambda) = (A^TA + \lambda I)^{-1}A^T$, where $\lambda > 0$. Show

$$\| B(\lambda) - A^+ \|_2 = \frac{\lambda}{\sigma_r(A)[\sigma_r(A)^2 + \lambda]} \qquad r = \text{rank}(A)$$

and therefore that $B(\lambda) \to A^+$ as $\lambda \to 0$.

P5.5.3 Consider the rank deficient LS problem

$$\min_{\substack{y\in\mathbb{R}^r \\ z\in\mathbb{R}^{n-r}}} \left\| \left[\begin{array}{cc} R & S \\ 0 & 0 \end{array}\right] \left[\begin{array}{c} y \\ z \end{array}\right] - \left[\begin{array}{c} c \\ d \end{array}\right] \right\|_2$$

where $R \in \mathbb{R}^{r\times r}$, $S \in \mathbb{R}^{r\times n-r}$, $y \in \mathbb{R}^r$, and $z \in \mathbb{R}^{n-r}$. Assume that R is upper triangular and nonsingular. Show how to obtain the minimum norm solution to this problem by computing an appropriate QR factorization without pivoting and then solving for the appropriate y and z.

P5.5.4 Show that if $A_k \to A$ and $A_k^+ \to A^+$, then there exists an integer k_0 such that $\text{rank}(A_k)$ is constant for all $k \geq k_0$.

P5.5.5 Show that if $A \in \mathbb{R}^{m\times n}$ has rank n, then so does $A + E$ if we have the inequality $\| E \|_2 \| A^+ \|_2 < 1$.

Notes and References for Sec. 5.5

The pseudo-inverse literature is vast, as evidenced by the 1,775 references in

M.Z. Nashed (1976). *Generalized Inverses and Applications*, Academic Press, New York.

The differentiation of the pseudo-inverse is further discussed in

C.L. Lawson and R.J. Hanson (1969). "Extensions and Applications of the Householder Algorithm for Solving Linear Least Squares Problems," *Math. Comp. 23*, 787–812.
G.H. Golub and V. Pereyra (1973). "The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate," *SIAM J. Num. Anal. 10*, 413–32.

Survey treatments of LS perturbation theory may be found in Lawson and Hanson (1974), Stewart and Sun (1991), Björck (1996), and

P.A. Wedin (1973). "Perturbation Theory for Pseudo-Inverses," *BIT 13*, 217–32.
G.W. Stewart (1977). "On the Perturbation of Pseudo-Inverses, Projections, and Linear Least Squares," *SIAM Review 19*, 634–62.

Even for full rank problems, column pivoting seems to produce more accurate solutions. The error analysis in the following paper attempts to explain why.

L.S. Jennings and M.R. Osborne (1974). "A Direct Error Analysis for Least Squares," *Numer. Math. 22*, 322–32.

Various other aspects rank deficiency are discussed in

J.M. Varah (1973). "On the Numerical Solution of Ill-Conditioned Linear Systems with Applications to Ill-Posed Problems," *SIAM J. Num. Anal. 10*, 257–67.
G.W. Stewart (1984). "Rank Degeneracy," *SIAM J. Sci. and Stat. Comp. 5*, 403–413.
P.C. Hansen (1987). "The Truncated SVD as a Method for Regularization," *BIT 27*, 534–553.
G.W. Stewart (1987). "Collinearity and Least Squares Regression," *Statistical Science 2*, 68–100.

We have more to say on the subject in §12.1 and §12.2.

5.6 Weighting and Iterative Improvement

The concepts of scaling and iterative improvement were introduced in the Chapter 3 context of square linear systems. Generalizations of these ideas that are applicable to the least squares problem are now offered.

5.6.1 Column Weighting

Suppose $G \in \mathbb{R}^{n\times n}$ is nonsingular. A solution to the LS problem

$$\min \| Ax - b \|_2 \qquad A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m \tag{5.6.1}$$

can be obtained by finding the minimum 2-norm solution y_{LS} to

$$\min \| (AG)y - b \|_2 \tag{5.6.2}$$

and then setting $x_G = Gy_{LS}$. If rank$(A) = n$, then $x_G = x_{LS}$. Otherwise, x_G is the minimum G-norm solution to (5.6.1), where the G-norm is defined by $\| z \|_G = \| G^{-1}z \|_2$.

The choice of G is important. Sometimes its selection can be based on à priori knowledge of the uncertainties in A. On other occasions, it may be desirable to normalize the columns of A by setting

$$G = G_0 \equiv \mathrm{diag}(1/\| A(:,1) \|_2, \ldots, 1/\| A(:,n) \|_2).$$

Van der Sluis (1969) has shown that with this choice, $\kappa_2(AG)$ is approximately minimized. Since the computed accuracy of y_{LS} depends on $\kappa_2(AG)$, a case can be made for setting $G = G_0$.

We remark that column weighting affects singular values. Consequently, a scheme for determining numerical rank may not return the same estimates when applied to A and AG. See Stewart (1984b).

5.6.2 Row Weighting

Let $D = \mathrm{diag}(d_1, \ldots, d_m)$ be nonsingular and consider the *weighted least squares problem*

$$\text{minimize } \| D(Ax - b) \|_2 \qquad A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m. \tag{5.6.3}$$

Assume $\mathrm{rank}(A) = n$ and that x_D solves (5.6.3). It follows that the solution x_{LS} to (5.6.1) satisfies

$$x_D - x_{LS} = (A^T D^2 A)^{-1} A^T (D^2 - I)(b - Ax_{LS}). \tag{5.6.4}$$

This shows that row weighting in the LS problem affects the solution. (An important exception occurs when $b \in \mathrm{ran}(A)$ for then $x_D = x_{LS}$.)

One way of determining D is to let d_k be some measure of the uncertainty in b_k, e.g., the reciprocal of the standard deviation in b_k. The tendency is for $r_k = e_k^T(b - Ax_D)$ to be small whenever d_k is large. The precise effect of d_k on r_k can be clarified as follows. Define

$$D(\delta) = \mathrm{diag}(d_1, \ldots, d_{k-1}, d_k\sqrt{1+\delta}, d_{k+1}, \ldots, d_m)$$

where $\delta > -1$. If $x(\delta)$ minimizes $\| D(\delta)(Ax - b) \|_2$ and $r_k(\delta)$ is the k-th component of $b - Ax(\delta)$, then it can be shown that

$$r_k(\delta) = \frac{r_k}{1 + \delta d_k^2 e_k^T A(A^T D^2 A)^{-1} A^T e_k}. \tag{5.6.5}$$

This explicit expression shows that $r_k(\delta)$ is a monotone decreasing function of δ. Of course, how r_k changes when all the weights are varied is much more complicated.

Example 5.6.1 Suppose

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

If $D = I_4$ then $x_D = [\,-1,\ .85\,]^T$ and $r = b - Ax_D = [\,.3,\ -.4,\ -.1,\ .2\,]^T$. On the other hand, if $D = \text{diag}(\,1000,\ 1,\ 1,\ 1\,)$ then we have $x_D \approx [\,-1.43,\ 1.21\,]^T$ and $r = b - Ax_D = [\,.000428\ \ -.571428\ \ -.142853\ .285714\,]^T$.

5.6.3 Generalized Least Squares

In many estimation problems, the vector of observations b is related to x through the equation

$$b \;=\; Ax + w \tag{5.6.6}$$

where the *noise vector* w has zero mean and a symmetric positive definite *variance-covariance* matrix $\sigma^2 W$. Assume that W is known and that $W = BB^T$ for some $B \in \mathbb{R}^{m\times m}$. The matrix B might be given or it might be W's Cholesky triangle. In order that all the equations in (5.6.6) contribute equally to the determination of x, statisticians frequently solve the LS problem

$$\min \| B^{-1}(Ax - b) \|_2 \,. \tag{5.6.7}$$

An obvious computational approach to this problem is to form $\tilde{A} = B^{-1}A$ and $\tilde{b} = B^{-1}b$ and then apply any of our previous techniques to minimize $\| \tilde{A}x - \tilde{b} \|_2$. Unfortunately, x will be poorly determined by such a procedure if B is ill-conditioned.

A much more stable way of solving (5.6.7) using orthogonal transformations has been suggested by Paige (1979a, 1979b). It is based on the idea that (5.6.7) is equivalent to the *generalized least squares* problem,

$$\min_{b=Ax+Bv} \quad v^T v \,. \tag{5.6.8}$$

Notice that this problem is defined even if A and B are rank deficient. Although Paige's technique can be applied when this is the case, we shall describe it under the assumption that both these matrices have full rank.

The first step is to compute the QR factorization of A:

$$Q^T A \;=\; \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \qquad Q = \begin{array}{cc} [\; Q_1 & Q_2 \;] \\ n & m-n \end{array} \,.$$

An orthogonal matrix $Z \in \mathbb{R}^{m\times m}$ is then determined so that

$$Q_2^T B Z = \begin{array}{cc} [\; 0 & S \;] \\ n & m-n \end{array} \qquad Z = \begin{array}{cc} [\; Z_1 & Z_2 \;] \\ n & m-n \end{array}$$

where S is upper triangular. With the use of these orthogonal matrices the constraint in (5.6.8) transforms to

$$\begin{bmatrix} Q_1^T b \\ Q_2^T b \end{bmatrix} = \begin{bmatrix} R_1 \\ 0 \end{bmatrix} x + \begin{bmatrix} Q_1^T B Z_1 & Q_1^T B Z_2 \\ 0 & S \end{bmatrix} \begin{bmatrix} Z_1^T v \\ Z_2^T v \end{bmatrix} .$$

Notice that the "bottom half" of this equation determines v,

$$Su = Q_2^T b \qquad v = Z_2 u, \tag{5.6.9}$$

while the "top half" prescribes x:

$$R_1 x = Q_1^T b - (Q_1^T B Z_1 Z_1^T + Q_1^T B Z_2 Z_2^T)v = Q_1^T b - Q_1^T B Z_2 u. \tag{5.6.10}$$

The attractiveness of this method is that all potential ill-conditioning is concentrated in triangular systems (5.6.9) and (5.6.10). Moreover, Paige (1979b) has shown that the above procedure is numerically stable, something that is not true of any method that explicitly forms $B^{-1}A$.

5.6.4 Iterative Improvement

A technique for refining an approximate LS solution has been analyzed by Björck (1967, 1968). It is based on the idea that if

$$\begin{bmatrix} I_m & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} \qquad A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m \tag{5.6.11}$$

then $\| b - Ax \|_2 = \min$. This follows because $r + Ax = b$ and $A^T r = 0$ imply $A^T A x = A^T b$. The above augmented system is nonsingular if $\text{rank}(A) = n$, which we hereafter assume.

By casting the LS problem in the form of a square linear system, the iterative improvement scheme (3.5.5) can be applied:

```
$r^{(0)} = 0;\ x^{(0)} = 0$
for $k = 0, 1,$
    $\begin{bmatrix} f^{(k)} \\ g^{(k)} \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r^{(k)} \\ x^{(k)} \end{bmatrix}$
    $\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} p^{(k)} \\ z^{(k)} \end{bmatrix} = \begin{bmatrix} f^{(k)} \\ g^{(k)} \end{bmatrix}$
    $\begin{bmatrix} r^{(k+1)} \\ x^{(k+1)} \end{bmatrix} = \begin{bmatrix} r^{(k)} \\ x^{(k)} \end{bmatrix} + \begin{bmatrix} p^{(k)} \\ z^{(k)} \end{bmatrix}$
end
```

The residuals $f^{(k)}$ and $g^{(k)}$ must be computed in higher precision and an original copy of A must be around for this purpose.

If the QR factorization of A is available, then the solution of the augmented system is readily obtained. In particular, if $A = QR$ and $R_1 = R(1{:}n, 1{:}n)$, then a system of the form

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} p \\ z \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

transforms to

$$\begin{bmatrix} I_n & 0 & R_1 \\ 0 & I_{m-n} & 0 \\ R_1^T & 0 & 0 \end{bmatrix} \begin{bmatrix} h \\ f_2 \\ z \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ g \end{bmatrix}$$

where

$$Q^T f = \begin{array}{c} \\ \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \end{array} \begin{array}{c} n \\ m-n \end{array} \qquad Q^T p = \begin{array}{c} \\ \begin{bmatrix} h \\ f_2 \end{bmatrix} \end{array} \begin{array}{c} n \\ m-n \end{array} .$$

Thus, p and z can be determined by solving the triangular systems $R_1^T h = g$ and $R_1 z = f_1 - h$ and setting $p = Q \begin{bmatrix} h \\ f_2 \end{bmatrix}$. Assuming that Q is stored in factored form, each iteration requires $8mn - 2n^2$ flops.

The key to the iteration's success is that both the LS residual and solution are updated—not just the solution. Björck (1968) shows that if $\kappa_2(A) \approx \beta^q$ and t-digit, β-base arithmetic is used, then $x^{(k)}$ has approximately $k(t-q)$ correct base β digits, provided the residuals are computed in double precision. Notice that it is $\kappa_2(A)$, not $\kappa_2(A)^2$, that appears in this heuristic.

Problems

P5.6.1 Verify (5.6.4).

P5.6.2 Let $A \in \mathbb{R}^{m \times n}$ have full rank and define the diagonal matrix

$$\Delta = \text{diag}(\underbrace{1, \ldots, 1}_{k-1}, (1+\delta), \underbrace{1, \ldots, 1}_{m-k})$$

for $\delta > -1$. Denote the LS solution to min $\| \Delta(Ax - b) \|_2$ by $x(\delta)$ and its residual by $r(\delta) = b - Ax(\delta)$. (a) Show

$$r(\delta) = \left(I - \delta \frac{A(A^TA)^{-1}A^T e_k e_k^T}{1 + \delta e_k^T A(A^TA)^{-1}A^T e_k} \right) r(0) .$$

(b) Letting $r_k(\delta)$ stand for the kth component of $r(\delta)$, show

$$r_k(\delta) = \frac{r_k(0)}{1 + \delta e_k^T A(A^TA)^{-1}A^T e_k} .$$

(c) Use (b) to verify (5.6.5).

P5.6.3 Show how the SVD can be used to solve the generalized LS problem when the matrices A and B in (5.6.8) are rank deficient.

P5.6.4 Let $A \in \mathbb{R}^{m \times n}$ have rank n and for $\alpha \geq 0$ define

$$M(\alpha) = \begin{bmatrix} \alpha I_m & A \\ A^T & 0 \end{bmatrix}.$$

Show that

$$\sigma_{m+n}(M(\alpha)) = \min\left\{\alpha\,,\ -\frac{\alpha}{2} + \sqrt{\sigma_n(A)^2 + \left(\frac{\alpha}{2}\right)^2}\right\}$$

and determine the value of α that minimizes $\kappa_2(M(\alpha))$.

P5.6.5 Another iterative improvement method for LS problems is the following:

```
$x^{(0)} = 0$
for $k = 0, 1, \ldots$
    $r^{(k)} = b - Ax^{(k)}$    (double precision)
    $\| Az^{(k)} - r^{(k)} \|_2 = \min$
    $x^{(k+1)} = x^{(k)} + z^{(k)}$
end
```

(a) Assuming that the QR factorization of A is available, how many flops per iteration are required? (b) Show that the above iteration results by setting $g^{(k)} = 0$ in the iterative improvement scheme given in §5.6.4.

Notes and References for Sec. 5.6

Row and column weighting in the LS problem is discussed in Lawson and Hanson (SLS, pp. 180-88). The various effects of scaling are discussed in

A. van der Sluis (1969). "Condition Numbers and Equilibration of Matrices," *Numer. Math. 14*, 14–23.

G.W. Stewart (1984b). "On the Asymptotic Behavior of Scaled Singular Value and QR Decompositions," *Math. Comp. 43*, 483–490.

The theoretical and computational aspects of the generalized least squares problem appear in

S. Kourouklis and C.C. Paige (1981). "A Constrained Least Squares Approach to the General Gauss-Markov Linear Model," *J. Amer. Stat. Assoc.* 76, 620–25.

C.C. Paige (1979a). "Computer Solution and Perturbation Analysis of Generalized Least Squares Problems," *Math. Comp. 33*, 171–84.

C.C. Paige (1979b). "Fast Numerically Stable Computations for Generalized Linear Least Squares Problems," *SIAM J. Num. Anal. 16*, 165–71.

C.C. Paige (1985). "The General Limit Model and the Generalized Singular Value Decomposition," *Lin. Alg. and Its Applic. 70*, 269–284.

Iterative improvement in the least squares context is discussed in

G.H. Golub and J.H. Wilkinson (1966). "Note on Iterative Refinement of Least Squares Solutions," *Numer. Math. 9*, 139–48.

Å. Björck and G.H. Golub (1967). "Iterative Refinement of Linear Least Squares Solutions by Householder Transformation," *BIT* 7, 322–37.

Å. Björck (1967). "Iterative Refinement of Linear Least Squares Solutions I," *BIT 7*, 257–78.

Å. Björck (1968). "Iterative Refinement of Linear Least Squares Solutions II," *BIT 8*, 8–30.

Å. Björck (1987). "Stability Analysis of the Method of Seminormal Equations for Linear Least Squares Problems," *Linear Alg. and Its Applic. 88/89*, 31–48.

5.7 Square and Underdetermined Systems

The orthogonalization methods developed in this chapter can be applied to square systems and also to systems in which there are fewer equations than unknowns. In this brief section we discuss some of the various possibilities.

5.7.1 Using QR and SVD to Solve Square Systems

The least squares solvers based on the QR factorization and the SVD can be used to solve square linear systems: just set $m = n$. However, from the flop point of view, Gaussian elimination is the cheapest way to solve a square linear system as shown in the following table which assumes that the right hand side is available at the time of factorization:

Method	Flops
Gaussian Elimination	$2n^3/3$
Householder Orthogonalization	$4n^3/3$
Modified Gram-Schmidt	$2n^3$
Bidiagonalization	$8n^3/3$
Singular Value Decomposition	$12n^3$

Nevertheless, there are three reasons why orthogonalization methods might be considered:

- The flop counts tend to exaggerate the Gaussian elimination advantage. When memory traffic and vectorization overheads are considered, the QR approach is comparable in efficiency.

- The orthogonalization methods have guaranteed stability; there is no "growth factor" to worry about as in Gaussian elimination.

- In cases of ill-conditioning, the orthogonal methods give an added measure of reliability. QR with condition estimation is very dependable and, of course, SVD is unsurpassed when it comes to producing a meaningful solution to a nearly singular system.

We are not expressing a strong preference for orthogonalization methods but merely suggesting viable alternatives to Gaussian elimination.

We also mention that the SVD entry in Table 5.7.1 assumes the availability of b at the time of decomposition. Otherwise, $20n^3$ flops are required because it then becomes necessary to accumulate the U matrix.

If the QR factorization is used to solve $Ax = b$, then we ordinarily have to carry out a back substitution: $Rx = Q^Tb$. However, this can be avoided by "preprocessing" b. Suppose H is a Householder matrix such that $Hb = \beta e_n$ where e_n is the last column of I_n. If we compute the QR factorization of $(HA)^T$, then $A = H^T R^T Q^T$ and the system transforms to

$$R^T y = \beta e_n$$

where $y = Q^T x$. Since R^T is lower triangular, $y = (\beta/r_{nn})e_n$ and so

$$x = \frac{\beta}{r_{nn}} Q(:,n).$$

5.7.2 Underdetermined Systems

We say that a linear system

$$Ax = b \qquad A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m \tag{5.7.1}$$

is *underdetermined* whenever $m < n$. Notice that such a system either has no solution or has an infinity of solutions. In the second case, it is important to distinguish between algorithms that find the minimum 2-norm solution and those that do not necessarily do so. The first algorithm we present is in the latter category. Assume that A has full row rank and that we apply QR with column pivoting to obtain:

$$Q^T A \Pi = [\, R_1 \ R_2 \,]$$

where $R_1 \in \mathbb{R}^{m\times m}$ is upper triangular and $R_2 \in \mathbb{R}^{m\times(n-m)}$. Thus, $Ax = b$ transforms to

$$(Q^T A\Pi)(\Pi^T x) = [\, R_1 \ R_2 \,]\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = Q^T b$$

where

$$\Pi^T x = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

with $z_1 \in \mathbb{R}^m$ and $z_2 \in \mathbb{R}^{(n-m)}$. By virtue of the column pivoting, R_1 is nonsingular because we are assuming that A has full row rank. One solution to the problem is therefore obtained by setting $z_1 = R_1^{-1} Q^T b$ and $z_2 = 0$.

Algorithm 5.7.1 Given $A \in \mathbb{R}^{m\times n}$ with rank$(A) = m$ and $b \in \mathbb{R}^m$, the following algorithm finds an $x \in \mathbb{R}^n$ such that $Ax = b$.

```
$Q^T A\Pi = R$        (QR with column pivoting.)
Solve $R(1{:}m, 1{:}m)z_1 = Q^T b$.
Set $x = \Pi \begin{bmatrix} z_1 \\ 0 \end{bmatrix}$.
```

This algorithm requires $2m^2n - m^3/3$ flops. The minimum norm solution is not guaranteed. (A different Π would render a smaller z_1.) However, if we compute the QR factorization

$$A^T = QR = Q\begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

with $R_1 \in \mathbb{R}^{m\times m}$, then $Ax = b$ becomes

$$(QR)^Tx = \begin{bmatrix} R_1^T & 0 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = b$$

where

$$Q^Tx = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \qquad z_1 \in \mathbb{R}^m,\ z_2 \in \mathbb{R}^{n-m}.$$

Now the minimum norm solution *does* follow by setting $z_2 = 0$.

Algorithm 5.7.2 Given $A \in \mathbb{R}^{m\times n}$ with $\text{rank}(A) = m$ and $b \in \mathbb{R}^m$, the following algorithm finds the minimal 2-norm solution to $Ax = b$.

```
$A^T = QR$        (QR factorization)
Solve $R(1{:}m, 1{:}m)^T z = b$.
$x = Q(:, 1{:}m)z$
```

This algorithm requires at most $2m^2n - 2m^3/3$

The SVD can also be used to compute the minimal norm solution of an underdetermined $Ax = b$ problem. If

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T \qquad r = \text{rank}(A)$$

is A's singular value expansion, then

$$x = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i .$$

As in the least squares problem, the SVD approach is desirable whenever A is nearly rank deficient.

5.7.3 Perturbed Underdetermined Systems

We conclude this section with a perturbation result for full-rank underdetermined systems.

Theorem 5.7.1 *Suppose* rank$(A) = m \le n$ *and that* $A \in \mathbb{R}^{m\times n}$, $\delta A \in \mathbb{R}^{m\times n}$, $0 \ne b \in \mathbb{R}^m$, *and* $\delta b \in \mathbb{R}^m$ *satisfy*

$$\epsilon \;=\; \max\{\epsilon_A, \epsilon_b\} \;<\; \sigma_m(A),$$

where $\epsilon_A = \|\,\delta A\,\|_2/\|\,A\,\|_2$ *and* $\epsilon_b = \|\,\delta b\,\|_2/\|\,b\,\|_2$. *If* x *and* $\hat{x}$ *are minimum norm solutions that satisfy*

$$Ax = b \qquad\qquad (A+\delta A)\hat{x} = b + \delta b$$

then

$$\frac{\|\,\hat{x}-x\,\|_2}{\|\,x\,\|_2} \quad\le\quad \kappa_2(A)\left(\epsilon_A \min\{2, n-m+1\} + \epsilon_b\right) + O(\epsilon^2).$$

Proof. Let E and f be defined by $\delta A/\epsilon$ and $\delta b/\epsilon$. Note that rank$(A+tE) = m$ for all $0 < t < \epsilon$ and that

$$x(t) \;=\; (A+tE)^T\left((A+tE)(A+tE)^T\right)^{-1}(b+tf)$$

satisfies $(A+tE)x(t) = b+tf$. By differentiating this expression with respect to t and setting $t=0$ in the result we obtain

$$\dot{x}(0) \;=\; \left(I - A^T(AA^T)^{-1}A\right)E^T(AA^T)^{-1}b + A^T(AA^T)^{-1}(f - Ex).$$

Since

$$\|\,x\,\|_2 \;=\; \|\,A^T(AA^T)^{-1}b\,\|_2 \ge \sigma_m(A)\|\,(AA^T)^{-1}b\,\|_2,$$
$$\|\,I - A^T(AA^T)^{-1}A\,\|_2 \;=\; \min(1, n-m),$$

and

$$\frac{\|\,f\,\|_2}{\|\,x\,\|_2} \;\le\; \frac{\|\,f\,\|_2\|\,A\,\|_2}{\|\,b\,\|_2},$$

we have

$$\frac{\|\,\hat{x}-x\,\|_2}{\|\,x\,\|_2} \;=\; \frac{x(\epsilon)-x(0)}{\|\,x(0)\,\|_2} \;=\; \epsilon\frac{\|\,\dot{x}(0)\,\|_2}{\|\,x\,\|_2} \;+\; O(\epsilon^2)$$
$$\le\; \epsilon\,\min(1, n-m)\left\{\frac{\|\,E\,\|_2}{\|\,A\,\|_2} + \frac{\|\,f\,\|_2}{\|\,b\,\|_2} + \frac{\|\,E\,\|_2}{\|\,A\,\|_2}\right\}\kappa_2(A) \;+\; O(\epsilon^2)$$

from which the theorem follows. □

Note that there is no $\kappa_2(A)^2$ factor as in the case of overdetermined systems.

Problems

P5.7.1 Derive the above expression for $\dot{x}(0)$.

P5.7.2 Find the minimal norm solution to the system $Ax = b$ where $A = [\,1\;2\;3\,]$ and $b = 1$.

P5.7.3 Show how triangular system solving can be avoided when using the QR factorization to solve an underdetermined system.

P5.7.4 Suppose $b, x \in \mathbb{R}^n$ are given. Consider the following problems:

(a) Find an unsymmetric Toeplitz matrix T so $Tx = b$.

(b) Find a symmetric Toeplitz matrix T so $Tx = b$.

(c) Find a circulant matrix C so $Cx = b$.

Pose each problem in the form $Ap = b$ where A is a matrix made up of entries from x and p is the vector of sought-after parameters.

Notes and References for Sec. 5.7

Interesting aspects concerning singular systems are discussed in

T.F. Chan (1984). "Deflated Decomposition Solutions of Nearly Singular Systems," *SIAM J. Num. Anal.* *21*, 738–754.

G.H. Golub and C.D. Meyer (1986). "Using the QR Factorization and Group Inversion to Compute, Differentiate, and estimate the Sensitivity of Stationary Probabilities for Markov Chains," *SIAM J. Alg. and Dis. Methods*, 7, 273–281.

Papers on underdetermined systems include

R.E. Cline and R.J. Plemmons (1976). "L_2-Solutions to Underdetermined Linear Systems," *SIAM Review 18*, 92-106.

M. Arioli and A. Laratta (1985). "Error Analysis of an Algorithm for Solving an Underdetermined System," *Numer. Math. 46*, 255–268.

J.W. Demmel and N.J. Higham (1993). "Improved Error Bounds for Underdetermined System Solvers," *SIAM J. Matrix Anal. Appl. 14*, 1–14.

The QR factorization can of course be used to solve linear systems. See

N.J. Higham (1991). "Iterative Refinement Enhances the Stability of QR Factorization Methods for Solving Linear Equations," *BIT 31*, 447–468.