Chapter 16

Least-Squares Problems

You should be familiar with

- Rank
- Proof by contradiction
- Cholesky decomposition
- *QR* decomposition
- SVD
- Residual
- Block matrix notation

We introduced the concept of least squares in Section 12.3 by developing the least-squares algorithm for fitting a polynomial to data. Even in that case, we found the unique polynomial by solving a square system of equations $Ax = b$, where A was nonsingular. In many areas such as curve fitting, statistics, and geodetic modeling, A is either singular or has dimension $m \times n$, $m \neq n$. If $m > n$, there are more equations than unknowns, and the system is said to be *overdetermined*. In most cases, overdetermined systems have no solution. In the case $m < n$, there are more unknowns than equations, and we say such systems are *underdetermined.* In this situation, there are usually an infinite number of solutions. It is clear that the Gaussian elimination techniques we have studied will not be useful in these cases.

Since singular, over- and underdetermined systems do not give us a solution in the exact sense, the solution is to find a vector x such that Ax is as close as possible to b. A way of doing this is to find a vector x such that the residual $r(x) = \|Ax - b\|_2$ is a minimum. Recall that the Euclidean norm, $\|\cdot\|_2$, of a vector in $\mathbb{R}^n$ is $\sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$, so if we want to minimize $\|Ax - b\|_2$, we call x a *least-squares* solution. Finding a least-squares solution to $Ax = b$ is known as the *linear least-squares problem*. We need to formally define the problem so we can develop rigorous techniques for solving it.

Definition 16.1 (The least-squares problem). Given a real $m \times n$ matrix A and a real vector b, find a real vector $x \in \mathbb{R}^n$ such that the function $r(x) = \|Ax - b\|_2$ is minimized. It is possible that the solution x will not be unique.

Assume that $m > n$. Since $x \in \mathbb{R}^n$, and A is an $m \times n$ matrix, Ax is a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$, and the range of the transformation, $R(A)$, is a subspace of $\mathbb{R}^m$. Given any $y \in R(A)$, there is an $x \in R^n$ such that $Ax = y$. If $b \in \mathbb{R}^m$ is in $R(A)$, we have a solution. If b is not in $R(A)$, consider the vector $Ax - b$ that joins the endpoints of the vectors Ax and b. Since b is not in $R(A)$, project b onto the plane $R(A)$ to obtain a vector $u \in R(A)$. There must be a vector $x \in \mathbb{R}^n$ such that $Ax = u$. The distance between the two points is $\|Ax - b\|_2$ is as small as possible, so x is the solution we want (Figure 16.1).

The vector $b - Ax$ is orthogonal to $R(A)$, and since every vector in $R(A)$ is a linear combination of the columns of A (vectors in R^m), it must be the case that $b - Ax$ is orthogonal to the every column of A. Mathematically this says that the inner product of $b - Ax$ with each column of A must be zero. If

$$a_i = \begin{bmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{m-1,i} \\ a_{mi} \end{bmatrix}$$

is column i, then $\langle a_i, b - Ax \rangle = a_i^{\mathrm{T}}(b - Ax) = 0$, $1 \leq i \leq n$, and

$$A^{\mathrm{T}}(b - Ax) = 0,$$

Numerical Linear Algebra with Applications. http://dx.doi.org/10.1016/B978-0-12-394435-1.00016-8

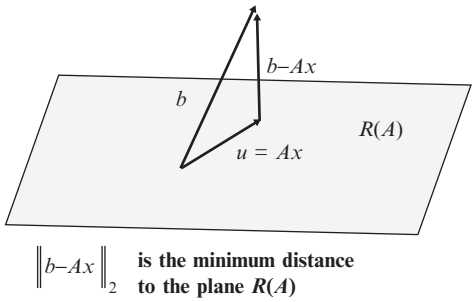

FIGURE 16.1 Geometric interpretation of the least-squares solution.

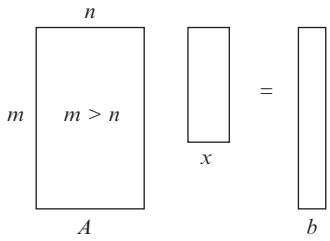

FIGURE 16.2 An overdetermined system.

and

$$A^{\mathrm{T}}Ax = A^{\mathrm{T}}b.$$

These are the normal equations specified in Definition 12.1.

16.1 EXISTENCE AND UNIQUENESS OF LEAST-SQUARES SOLUTIONS

The geometric argument we presented is not an mathematical proof. We should give some mathematical justification that if $m > n$, a solution exists and satisfies the normal equations. Figure 16.2 graphically represents an overdetermined system.

16.1.1 Existence and Uniqueness Theorem

In order to prove the existence and uniqueness to the solution of the least-squares problem, we first consider the case $m \geq n$.

Let A be an $m \times n$ matrix. Then, each of the n columns has m components and is a member of $\mathbb{R}^m$, and each of m rows has n components and is a member of $\mathbb{R}^n$. The columns of A span a subspace of $\mathbb{R}^m$, and the rows of A span a subspace of $\mathbb{R}^n$. The column rank of A is the number of linearly independent columns of A, and the row rank of A is the number of linear independent rows of A. Theorem 15.3 proves that the column rank and row rank of A are equal.

Definition 16.2. An $m \times n$ matrix A has full rank if rank $(A) = \min(m, n)$.

If $m \geq n$, and A has full rank, then rank$(A) = n$, and the columns of A are linearly independent.

Lemma 16.1. *Let A be an $m \times n$ matrix, $m \geq n$. A has full rank if and only if the $n \times n$ matrix $A^{\mathrm{T}}A$ is nonsingular.*

Proof. Use proof by contradiction. Assume A has full rank, but $A^{\mathrm{T}}A$ is singular. In this case, the $n \times n$ homogeneous system $A^{\mathrm{T}}Ax = 0$ has a nonzero solution x. As a result, $x^{\mathrm{T}}A^{\mathrm{T}}Ax = 0$, which says that $\langle Ax, Ax\rangle = \|Ax\|_2^2 = 0$, so $Ax = 0$, and A cannot have full rank.

Again, use proof by contradiction. Assume $A^{\mathrm{T}}A$ is nonsingular, but A does not have full rank. Since A is rank deficient, there is a nonzero vector x such that $Ax = 0$. As a result, $A^{\mathrm{T}}Ax = 0$, $x \neq 0$, so $A^{\mathrm{T}}A$ is singular. □

Theorem 16.1 establishes the link between a least-squares solution and the normal equations and tells us the solution is unique if the matrix has full rank. The proof is somewhat technical and can be omitted if desired, but the results are very important to remember.

Theorem 16.1. **a.** *Given an $m \times n$ matrix A with $m \geq n$ and an $m \times 1$ column vector b, an $n \times 1$ column vector x exists such that x is a least-squares solution to $Ax = b$ if and only if x satisfies the normal equations*

$$A^{\mathrm{T}}Ax = A^{\mathrm{T}}b.$$

b. *The least-squares solution x is unique if and only if A has full rank.*

Proof. To prove part (1), assume that $A^{\mathrm{T}}A\bar{x} = A^{\mathrm{T}}b$, so that $\bar{x}$ is a solution of the normal equations. Now, if x is any vector in $\mathbb{R}^n$,

$$\begin{aligned}
\|Ax - b\|_2^2 &= \|Ax - A\bar{x} + A\bar{x} - b\|_2^2 = \langle [(A\bar{x} - b) + A(x - \bar{x})], [(A\bar{x} - b) + A(x - \bar{x})] \rangle \\
&= \|A\bar{x} - b\|_2^2 + 2\langle A(x - \bar{x}), (A\bar{x} - b) \rangle + \|A(x - \bar{x})\|_2^2 \\
&= \|A\bar{x} - b\|_2^2 + 2(A(x - \bar{x}))^{\mathrm{T}}(A\bar{x} - b) + \|A(x - \bar{x})\|_2^2 \\
&= \|A\bar{x} - b\|_2^2 + 2(x - \bar{x})^{\mathrm{T}}\left(A^{\mathrm{T}}A\bar{x} - A^{\mathrm{T}}b\right) + \|A(x - \bar{x})\|_2^2 \\
&= \|A\bar{x} - b\|_2^2 + \|A(x - \bar{x})\|_2^2 \\
&\geq \|A\bar{x} - b\|_2^2,
\end{aligned}$$

and $\bar{x}$ is a solution to the least-squares problem.

Now assume that $\bar{x} \in \mathbb{R}^n$ is a solution to the least-squares problem, so that $\|A\bar{x} - b\|_2$ is minimum. Thus, $\|b - A\bar{x}\|_2^2 \leq \|b - Ay\|_2^2$ for any $y \in \mathbb{R}^n$. Given any vector $z \in \mathbb{R}^n$, let $y = \bar{x} + tz$, where t is a scalar. Then,

$$\begin{aligned}
\|b - A\bar{x}\|_2^2 \leq \|b - A(\bar{x} + tz)\|_2^2 &= ([b - A\bar{x}] - tAz)^{\mathrm{T}}([b - A\bar{x}] - tAz) \\
&= \|b - A\bar{x}\|_2^2 - 2t(b - A\bar{x})^{\mathrm{T}}Az + t^2\|Az\|_2^2.
\end{aligned}$$

Thus,

$$0 \leq -2t(b - A\bar{x})^{\mathrm{T}}Az + t^2\|Az\|_2^2.$$

If $t > 0$,

$$0 \leq -2(b - A\bar{x})^{\mathrm{T}}Az + t\|Az\|_2^2,$$

and

$$2(b - A\bar{x})^{\mathrm{T}}Az \leq t\|Az\|_2^2.$$

If $t < 0$,

$$0 \leq 2(b - A\bar{x})^{\mathrm{T}}Az + |t|\|Az\|_2^2.$$

As $t \to 0^+$ or $t \to 0^-$, we have $2(b - A\bar{x})^{\mathrm{T}}Az \leq 0$ and $0 \leq 2(b - A\bar{x})^{\mathrm{T}}Az$, so

$$(b - A\bar{x})^{\mathrm{T}}Az = 0$$

for all $z \in \mathbb{R}^n$. Thus,

$$(b - A\bar{x})^T Az = (Az)^T(b - A\bar{x}) = z^T A^T(b - A\bar{x}) = z^T\left(A^Tb - A^TA\bar{x}\right) = 0$$

for all $z \in \mathbb{R}^n$. Choose $z = \left(A^Tb - A^TA\bar{x}\right)$, so $\left\|A^Tb - A^TA\bar{x}\right\|_2^2 = 0$ and $A^{\mathrm{T}}Ax = A^{\mathrm{T}}b$.

For part (2), if x is the unique solution to $A^{\mathrm{T}}Ax = A^{\mathrm{T}}b$, then $A^{\mathrm{T}}A$ is nonsingular so A must have full rank according to Lemma 16.1. If A has full rank, $A^{\mathrm{T}}A$ is nonsingular by Lemma 16.1, and $A^{\mathrm{T}}Ax = A^{\mathrm{T}}b$ has a unique solution. □

16.1.2 Normal Equations and Least-Squares Solutions

The least-squares residual equation $r = b - Ax$ is what we defined as the residual for the solution of an ordinary $n \times n$ linear system. If we multiply the residual equation by A^{T}, the result is

$$A^{\mathrm{T}} r = A^{\mathrm{T}} b - A^{\mathrm{T}} Ax,$$

which says that we do not expect r to be 0. We want $A^{\mathrm{T}} r$ to be zero.

Theorem 16.2 (Least-squares residual equation). *Let $r = b - Ax$. Then $A^{\mathrm{T}} r = 0$ if and only if x is a least-squares solution.*

Proof. If x is a least-squares solution, then it satisfies the normal equations by Theorem 16.1. Then $A^{\mathrm{T}} Ax = A^{\mathrm{T}} b$ and $A^{\mathrm{T}} b - A^{\mathrm{T}} Ax = A^{\mathrm{T}} (b - Ax) = A^{\mathrm{T}} r = 0$.

When $A^{\mathrm{T}} r = 0$, $A^{\mathrm{T}} (b - Ax) = 0$, and $A^{\mathrm{T}} Ax = A^{\mathrm{T}} b$, so by Theorem 16.1, x is a least-squares solution. □

Example 16.1. Let $A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 3 & 8 \\ 2 & 9 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 8 \end{bmatrix}$, which is a system of four equations in two unknowns. The rank of A is 2, so we know there is a unique least-squares solution. Formulate and solve the normal equations.

$A^{\mathrm{T}} A = \begin{bmatrix} 18 & 53 \\ 53 & 170 \end{bmatrix}$ and $A^{\mathrm{T}} b = \begin{bmatrix} 38 \\ 127 \end{bmatrix}$. Solve $\begin{bmatrix} 18 & 53 \\ 53 & 170 \end{bmatrix} x = \begin{bmatrix} 38 \\ 127 \end{bmatrix}$ to obtain the solution $x = \begin{bmatrix} -1.0797 \\ 1.0837 \end{bmatrix}$. ■

16.1.3 The Pseudoinverse, $m \geq n$

Assume that A is a full rank $m \times n$ matrix, $m \geq n$. By Lemma 16.1, $A^{\mathrm{T}} A$ is invertible, so the matrix product $\left(A^{\mathrm{T}} A\right)^{-1} A^{\mathrm{T}}$ is defined. From Theorem 16.1, if A has full rank and x is a solution to the least-squares problem, then $A^{\mathrm{T}} Ax = A^{\mathrm{T}} b$, and so

$$x = \left(A^{\mathrm{T}} A\right)^{-1} A^{\mathrm{T}} b.$$

Definition 16.3. The matrix

$$A^{\dagger} = \left(A^{\mathrm{T}} A\right)^{-1} A^{\mathrm{T}}$$

is called the *pseudoinverse* or the *Moore-Penrose generalized inverse*. The solution to the full-rank overdetermined least-squares problem $Ax = b$ is $x = A^{\dagger} b$.

Remark 16.1. There is an equivalent definition for the pseudoinverse that uses components from the singular value decomposition (SVD). This form can be used for rank-deficient matrices, and we will present the definition in Section 16.4.

Example 16.2. $A = \begin{bmatrix} 1 & 6 & -1 \\ 4 & 2 & 1 \\ 0 & 3 & 5 \\ 2 & 6 & 9 \\ -1 & 5 & -8 \end{bmatrix}$, $b = \begin{bmatrix} 2 \\ 12 \\ 5 \end{bmatrix}$. The rank of A is 3, so A has full rank. Solve the system $Ax = b$.

$$x = A^{\dagger} b = \left[\left(A^{\mathrm{T}} A\right)^{-1} A^{\mathrm{T}}\right] b = \begin{bmatrix} 0.0691 & -0.0109 & -0.0101 \\ -0.0109 & 0.0111 & 0.0002 \\ -0.0101 & 0.0002 & 0.0075 \end{bmatrix} \begin{bmatrix} 1 & 4 & 0 & 2 & -1 \\ 6 & 2 & 3 & 6 & 5 \\ -1 & 1 & 5 & 9 & -8 \end{bmatrix} b$$

$$= \begin{bmatrix} 0.0138 & 0.2447 & -0.0831 & -0.0178 & -0.0431 \\ 0.0556 & -0.0212 & 0.0345 & 0.0469 & 0.0647 \\ -0.0162 & 0.0324 & 0.038 & 0.0485 & -0.0487 \end{bmatrix} \begin{bmatrix} 2 \\ 12 \\ 5 \end{bmatrix} = \begin{bmatrix} 2.5662 \\ -0.0174 \\ -0.2790 \end{bmatrix}$$

The pseudoinverse, $A^{\dagger}$ generalizes the definition of the inverse of a square matrix. In particular, when A is a nonsingular $n \times n$ matrix

$$A^{\ddagger} = \left(A^{\mathrm{T}}A\right)^{-1} A^{\mathrm{T}} = A^{-1}\left(A^{\mathrm{T}}\right)^{-1} A^{\mathrm{T}} = A^{-1}I = A^{-1}$$ ■

Recall that if B is a nonsingular matrix, $\kappa(B) = \|B\|_2 \left\|B^{-1}\right\|_2$. Given that $A^{\dagger} = A^{-1}$ for a nonsingular matrix, the following definition makes good sense for a full-rank matrix.

Definition 16.4. If A is a full rank $m \times n$ matrix, then the condition number of A is

$$\kappa(A) = \left\|A^{\dagger}\right\| \|A\|. \tag{16.1}$$

Definition 14.3 specifies the condition number of a general matrix. For a full rank matrix, $m \geq n$, Definition 16.4 is equivalent (Problem 16.4).

Example 16.3. If A is the 4×2 matrix in Example 16.1, then

$$\begin{aligned}\|A\| \left\|A^{\dagger}\right\| = 13.6622 \left\|\left(A^{\mathrm{T}}A\right)^{-1} A^{\mathrm{T}}\right\| &= 13.662184 \left\| \begin{bmatrix} 18 & 53 \\ 53 & 170 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 2 & 3 & 2 \\ 3 & 4 & 8 & 9 \end{bmatrix} \right\| \\ &= 13.6621841\,(0.8623494) = 11.7815762\end{aligned}$$

A is a well-conditioned matrix. On the other hand, let V be the 6×4 Vandermonde matrix formed from $x = \begin{bmatrix} 1.1 & 1.8 & 2.3 & 2.7 & 3.3 & 3.5 \end{bmatrix}^{\mathrm{T}}$, $m = 4$. The function `cond` in MATLAB computes the condition number of a nonsquare matrix, as you can see.

```
>> V = vandermonde(x,4);
>> norm(inv(V'*V)*V')*norm(V)

ans =
  3.0888e+004

>> cond(V)

ans =
  3.0888e+004
```

Clearly, V is an ill-conditioned matrix. ■

16.1.4 The Pseudoinverse, *m<n*

If $m < n$ and rank $(A) = m$, then A^{T} is of dimension $n \times m$, $n > m$, and the pseudoinverse is defined for A^{T}. We have

$$\left(A^{\mathrm{T}}\right)^{\dagger} = \left(\left(A^{\mathrm{T}}\right)^{\mathrm{T}} A^{\mathrm{T}}\right)^{-1} \left(A^{\mathrm{T}}\right)^{\mathrm{T}} = \left(AA^{\mathrm{T}}\right)^{-1} A$$

Now take the transpose to obtain

$$A^{\dagger} = A^{\mathrm{T}}\left[\left(AA^{\mathrm{T}}\right)^{-1}\right]^{\mathrm{T}} = A^{\mathrm{T}}\left(AA^{\mathrm{T}}\right)^{-1}.$$

Note that AA^{T} is an $m \times m$ matrix, and is nonsingular because A has full rank. This leads us to the definition.

Definition 16.5. Assume that A is an $m \times n$ matrix, $m < n$ and has full rank. The pseudoinverse is the well-defined product

$$A^{\dagger} = A^{\mathrm{T}}\left(AA^{\mathrm{T}}\right)^{-1}$$

16.2 SOLVING OVERDETERMINED LEAST-SQUARES PROBLEMS

There are three basic methods for solving overdetermined least-squares problems. The first of these, using the normal equations, was the standard method for many years. However, using the *QR* decomposition or the SVD gives much better results in most cases.

16.2.1 Using the Normal Equations

$A^{\mathrm{T}}A$ is a symmetric matrix. Consider the product $x^{\mathrm{T}}A^{\mathrm{T}}Ax = (Ax)^{\mathrm{T}}(Ax) = \|Ax\|_2^2$. If we assume that A has full rank, $Ax \neq 0$ for any $x \neq 0$, and so $A^{\mathrm{T}}A$ is positive definite. As a result, the Cholesky decomposition applies (Section 13.3). Recall that the Cholesky decomposition of a positive definite matrix M is of the form $M = R^{\mathrm{T}}R$, where R is an upper triangular matrix. To solve the normal equations, proceed as follows:

Solve the Normal Equations Using the Cholesky Decomposition

a. Find the Cholesky decomposition $A^{\mathrm{T}}A = R^{\mathrm{T}}R$.
b. Solve the system $R^{\mathrm{T}}y = A^{\mathrm{T}}b$ using forward substitution.
c. Solve the system $Rx = y$ using back substitution.

Example 16.4. There are three mountains m_1, m_2, m_3 that from one site have been measured as 2474 ft., 3882 ft., and 4834 ft. But from m_1, m_2 looks 1422 ft. taller and m_3 looks 2354 ft. taller, and from m_2, m_3 looks 950 ft. taller. This data gives rise to an overdetermined set of linear equations for the height of each mountain.

$$\begin{array}{rrrcl} m_1 & & & = & 2474 \\ & m_2 & & = & 3882 \\ & & m_3 & = & 4834 \\ -m_1 & +m_2 & & = & 1422 \\ -m_1 & & +m_3 & = & 2354 \\ & -m_2 & +m_3 & = & 950 \end{array}$$

In matrix form, the least-squares problem is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 2474 \\ 3882 \\ 4834 \\ 1422 \\ 2354 \\ 950 \end{bmatrix}.$$

$A^{\mathrm{T}}A = \begin{bmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{bmatrix}$ has the Cholesky decomposition

$$\begin{bmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{bmatrix} = \begin{bmatrix} 1.7321 & 0 & 0 \\ -0.5774 & 1.6330 & 0 \\ -0.5774 & -0.8165 & 1.4142 \end{bmatrix} \begin{bmatrix} 1.7321 & -0.5774 & -0.5774 \\ 0 & 1.6330 & -0.8165 \\ 0 & 0 & 1.4142 \end{bmatrix}$$

Solve

$$\begin{bmatrix} 1.7321 & 0 & 0 \\ -0.5774 & 1.6330 & 0 \\ -0.5774 & -0.8165 & 1.4142 \end{bmatrix} y = A^{\mathrm{T}} \begin{bmatrix} 2474 \\ 3882 \\ 4834 \\ 1422 \\ 2354 \\ 950 \end{bmatrix} = \begin{bmatrix} -1302 \\ 4354 \\ 8138 \end{bmatrix}, \quad y = \begin{bmatrix} -751.7 \\ 2400.5 \\ 6833.5 \end{bmatrix}$$

Solve

$$\begin{bmatrix} 1.7321 & -0.5774 & -0.5774 \\ 0 & 1.6330 & -0.8165 \\ 0 & 0 & 1.4142 \end{bmatrix} x = y$$

$$m = \begin{bmatrix} 2472.0 \\ 3886.0 \\ 4832.0 \end{bmatrix}$$

■

Algorithm 16.1 uses the normal equations to solve the least-squares problem.

Algorithm 16.1 Least-Squares Solution Using the Normal Equations

```
function NORMALSOLVE(A,b)
   % Solve the overdetermined least-squares problem using the normal equations.
   % Input: m × n full-rank matrix A, m ≥ n and an m × 1 column vector b.
   % Output: the unique least-squares solution to Ax = b and the residual
   c = A^T b
   Use the Cholesky decomposition to obtain A^T A = R^T R
   Solve the lower triangular system R^T y = c
   Solve the upper triangular system Rx = y
   return[ x  ||b − Ax||_2 ]
end function
```

NLALIB: The function `normalsolve` implements Algorithm 16.1 by using the functions `cholesky` and `cholsolve`.

Efficiency

The efficiency analysis is straightforward.

- $c = A^{\mathrm{T}}b$: $2nm$ flops
- Form $A^{\mathrm{T}}A$: $2mn^2$ flops
- Cholesky decomposition of $A^{\mathrm{T}}A$: $\frac{n^3}{3} + n^2 + \frac{5n}{3}$ flops
- Forward and back substitution: $2\left(n^2 + n - 1\right)$ flops

The total is $n^2\left(2m + \frac{n}{3}\right) + 2mn + 3n^2 + \frac{11}{3}n - 2 \simeq n^2\left(2m + \frac{n}{3}\right)$ flops.

Computational Note

It must be noted that, although relatively easy to understand and implement, it has serious problems in some cases.

- There may be some loss of significant digits when computing $A^{\mathrm{T}}A$. It may even be singular to working precision.
- We will see in Section 16.3 that the accuracy of the solution using the normal equations depends on the square of condition number of the matrix. If $\kappa\,(A)$ is large, the results can be seriously in error.

16.2.2 Using the *QR* Decomposition

The *QR* decomposition can be used very effectively to solve the least-squares problem $Ax = b$, $m \geq n$, when A has full rank. Our implementation of the *QR* decomposition using Gram-Schmidt required A to have full rank, but the algorithm is not as stable as others. In Chapter 17, we will develop two better algorithms for computing the *QR* decomposition. In this chapter, we will use the MATLAB function `qr` to compute a unique solution to the full-rank least-squares problem.

Using the reduced *QR* decomposition, $A = Q^{m\times n}R^{n\times n}$, we have

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

so the normal equations become

$$R^{\mathrm{T}}Rx = A^{\mathrm{T}}b = (QR)^{\mathrm{T}}\,b = R^{\mathrm{T}}Q^{\mathrm{T}}b. \tag{16.2}$$

Note that in reduced *QR* decomposition R is an $n \times n$ matrix. We can simplify Equation 16.2 by using an important property of the reduced *QR* decomposition.

Lemma 16.2. *If $A^{\mathrm{T}}A = QR$ is the reduced decomposition of $A^{\mathrm{T}}A$, then $r_{ii} > 0$, $1 \leq i \leq n$, where the r_{ii} are the diagonal entries of R.*

Proof. $A^{\mathrm{T}}A = (QR)^{\mathrm{T}}(QR) = R^{\mathrm{T}}Q^{\mathrm{T}}QR = R^{\mathrm{T}}R$. $A^{\mathrm{T}}A$ is positive definite, and by Theorem 13.3, R^{T} has positive diagonal entries. □

By Lemma 16.2, the diagonal entries of R^{T} are positive, so R^{T} is invertible, and from Equation 16.2 we have

$$Rx = Q^{\mathrm{T}}b \tag{16.3}$$

Since R is an upper triangular, system 16.3 can be solved simply by back substitution. This leads to Algorithm 16.2.

Algorithm 16.2 Solving the Least-Squares Problem Using the QR Decomposition

```
function QRLSTSQ(A,b)
   % Solve the least-squares problem using QR decomposition.
   % Input: m × n matrix A of full rank and m × 1 column vector b.
   % Output: The solution to the least-squares problem Ax=b and the residual
   Compute the reduced QR decomposition of A: A=QR
   c = Q^T b
   Solve the upper triangular system Rx=c
   return[ x  ‖b − Ax‖_2 ]
end function
```

NLALIB: The function `qrlstsq` implements Algorithm 16.2.

Example 16.5. The table shows the length-weight relation for a species of salmon. Find a power function $W = \alpha L^{\beta}$ which best fits the data.

L	0.5	1.0	2.0
W	1.77	10	56.6

Take the natural logarithm of the power function to obtain $\ln(W) = \beta \ln(L) + \ln(\alpha)$. In problems of this type, it is often necessary or convenient to perform a change of variable. Let $y = \ln(W)$, $x = \ln(L)$, and $b = \ln(\alpha)$, and $m = \beta$. Create a table containing the logarithms of L and W.

x	−0.6931	0.0000	0.6931
y	0.5710	2.3026	4.0360

We want the line $y = mx+b$ which best fits the data, so we must find b and m such that $\begin{array}{rcrcl} -0.6931m & + & b & = & 0.5710 \\ 0m & + & b & = & 2.3026 \\ 0.6931m & + & b & = & 4.0360 \end{array}$, an overdetermined system.

We must solve the least-squares problem $\begin{bmatrix} -0.6931 & 1 \\ 0 & 1 \\ 0.6931 & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} 0.5710 \\ 2.3026 \\ 4.0360 \end{bmatrix}$ for b and m. The following MATLAB code does this for us.

```
>> A = [-0.6931 1;0 1;0.6931 1];
>> b = [0.5710 2.3026 4.0360]';
>> x = qrlstsq(A,b)

x =
    2.4996
    2.3032
```

The least-squares line is $y = 2.4996x + 2.3032$. Now we need to find the corresponding power function.

$W = \mathrm{e}^{y} = \mathrm{e}^{2.4996x+2.3032} = \mathrm{e}^{2.3032}(\mathrm{e}^{x})^{2.4996} = \mathrm{e}^{2.3032}\left(\mathrm{e}^{\ln(L)}\right)^{2.4996} = \mathrm{e}^{2.3032}L^{2.4996}$. The power function that best fits this data is (Figure 16.3)

$$W = 10.0062L^{2.4996}$$

■

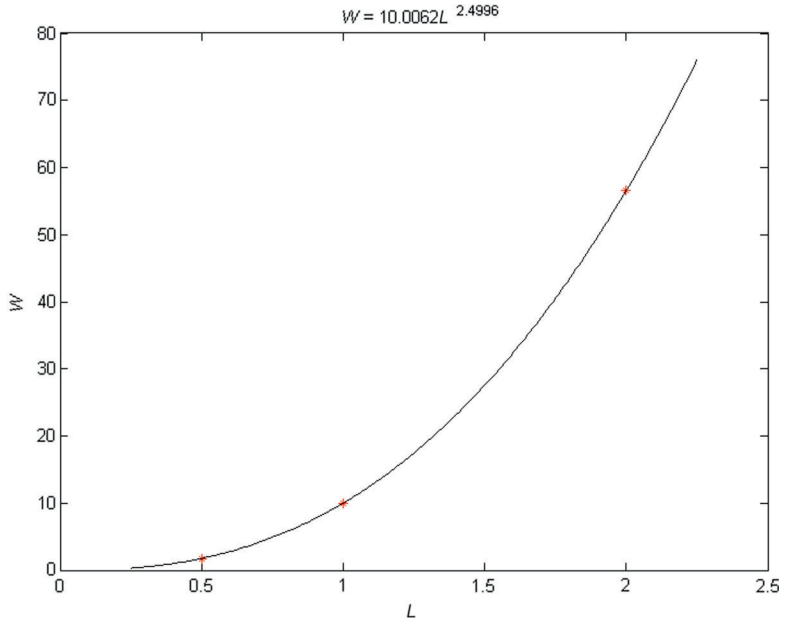

FIGURE 16.3 Least-squares estimate for the power function.

Efficiency

The decomposition is normally computed by the use of Householder reflections (Sections 17.8 and 17.9), and the cost of this algorithm is approximately $2n^2\left(m-\frac{n}{3}\right)$ flops. The product $Q^{\mathrm{T}}b$ requires $2mn$, and back substitution costs n^2+n-1 flops, so the total cost of Algorithm 16.2 is

$$2mn^2-\frac{2n^3}{3}+2mn+n^2+n+1.$$

The cost of 16.2 is dominated by computing the reduced QR factorization, so we ignore $2mn+n^2+n+1$ and say that the QR solution to the full-rank least-squares problem, $m\geq n$ costs approximately

$$2mn^2-\frac{2n^3}{3}$$

flops.

16.2.3 Using the SVD

Just like the QR decomposition, there is a *reduced SVD*, and it has the form (Figure 16.4)

$$A^{m\times n}=U^{m\times n}\Sigma^{n\times n}\left(V^{n\times n}\right)^{\mathrm{T}}.$$

In MATLAB, use the command `[U S V] = svd(A,0)`. The reduced SVD is also a powerful tool for computing full-rank least-squares solutions, $m\geq n$. The following manipulations show how to use it.

Apply the reduced SVD to A and obtain $A=U\Sigma V^{\mathrm{T}}$, where $U\in\mathbb{R}^{m\times n}$ has orthonormal columns, and $V\in\mathbb{R}^{n\times n}$ is orthogonal, and $\Sigma=\mathrm{diag}\left(\sigma_1,\sigma_2,\ldots,\sigma_n\right)\in\mathbb{R}^{n\times n}$, $\sigma_i>0$, $1\leq i\leq n$. Form the normal equations.

$$A^{\mathrm{T}}A=\left(U\Sigma V^{\mathrm{T}}\right)^{\mathrm{T}}U\Sigma V^{\mathrm{T}}$$

and so

$$A^{\mathrm{T}}A=(V\Sigma)\,\Sigma V^{\mathrm{T}}.$$

Now, $A^{\mathrm{T}}b=(V\Sigma)\,U^{\mathrm{T}}b$, and so the normal equations become

$$(V\Sigma)\,\Sigma V^{\mathrm{T}}x=(V\Sigma)\,U^{\mathrm{T}}b$$

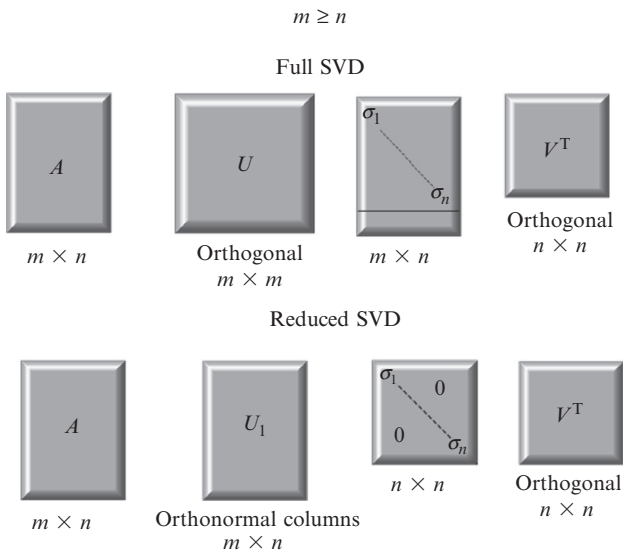

FIGURE 16.4 The reduced SVD for a full rank matrix.

Since V is orthogonal and Σ is a diagonal matrix with positive entries, $V\Sigma$ is invertible, and after multiplying the previous equation by $(V\Sigma)^{-1}$ we have

$$\Sigma V^{\mathrm{T}} x = U^{\mathrm{T}} b.$$

First solve $\Sigma y = U^{\mathrm{T}} b$, followed by $V^{\mathrm{T}} x = y$. Since Σ is a diagonal matrix, the solution to $\Sigma y = U^{\mathrm{T}} b$ is simple. Let $U^{\mathrm{T}} b = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix}$. Then,

$$\begin{bmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & 0 & \\ & 0 & \ddots & \\ 0 & & & \sigma_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix},$$

and $y_i = \frac{c_i}{\sigma_i}$, $1 \leq i \leq n$. The solution to $V^{\mathrm{T}} x = y$ is $x = Vy$. We summarize these steps in Algorithm 16.3.

Algorithm 16.3 Solving the Least-Squares Problem Using the SVD

```
function SVDLSTSQ(A,b)
   % Use the SVD to solve the least-squares problem.
   % Input: An m × n full-rank matrix A, m ≥ n, and an m × 1 vector b.
   % Output: The solution to the least-squares problem Ax=b and the residual
   Compute the reduced SVD of A: A = UΣV^T
   c = U^T b
   % Solve the system Σy = c
   for i=1:n do
      y_i = c_i/σ_i
   end for
   x = Vy
   return[ x  ||b − Ax||_2 ]
end function
```

NLALIB: The function `svdlstsq` implements Algorithm 16.3. The implementation uses `svd(A,0)` to compute the reduced SVD for A, and computes y using

```
y = c./diag(S),
```

where S is the diagonal matrix.

Example 16.6. The velocity of an enzymatic reaction with Michaelis-Menton kinetics is given by

$$v(s) = \frac{\alpha s}{1 + \beta s} \tag{16.4}$$

Find the Michaelis-Menton equation which best fits the data:

s	1	4	6	16
v	4	10	12	16

Inverting Equation 16.4 gives the Lineweaver-Burke equation:

$$\frac{1}{v} = \frac{1}{\alpha}\frac{1}{s} + \frac{\beta}{\alpha} \tag{16.5}$$

Perform the following change of variable: $y = \frac{1}{v}$ and $x = \frac{1}{s}$. Let $m = \frac{1}{\alpha}$ and $b = \frac{\beta}{\alpha}$. Equation 16.5 then becomes

$$y = mx + b$$

Recompute the table to reflect the change of variables.

x	1.0000	0.2500	0.1667	0.0625
y	0.2500	0.1000	0.0833	0.0625

Find the least-squares fit for $y = mx + b$ by solving the following 4×2 set of equations

$$\begin{aligned} 1.0000m &+ b = 0.2500 \\ 0.2500m &+ b = 0.1000 \\ 0.1667m &+ b = 0.0833 \\ 0.0625m &+ b = 0.0625 \end{aligned}$$

that correspond to the matrix equation

$$\begin{bmatrix} 1.0000 & 1 \\ 0.2500 & 1 \\ 0.1667 & 1 \\ 0.0625 & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} 0.2500 \\ 0.1000 \\ 0.0833 \\ 0.0625 \end{bmatrix}.$$

The following MATLAB code solves for m and b and then computes α, β.

```
>> A = [1.0000 1.0000;0.2500 1.0000;0.1667 1.0000;0.0625 1.0000];
>> b = [0.2500 0.1000 0.0833 0.0625]';
>> x = svdlstsq(A,b);
>> m = x(1);
>> b = x(2);
>> alpha = 1/m

alpha =

   4.9996

>> beta = alpha*b

beta =

   0.2499
```

The least-squares approximation is (Figure 16.5)

$$v(s) = \frac{4.9996s}{1 + 0.2499s}$$

■

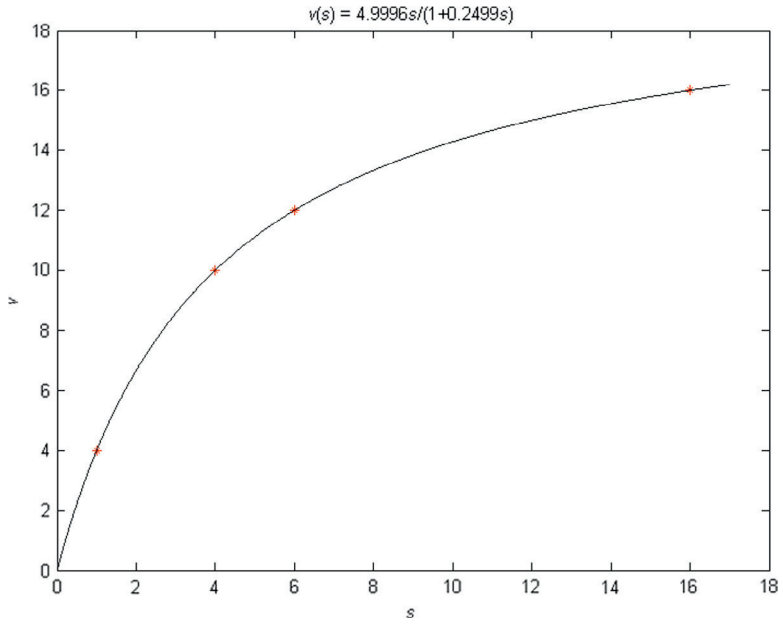

FIGURE 16.5 Velocity of an enzymatic reaction.

Efficiency

Algorithms for the computation of the SVD are complex, and we will develop two in Chapter 21. We will use the estimate given in Ref. [2, p. 493] of approximately $14mn^2 + 8n^3$ flops. Generally, using *QR* is faster for full rank least-squares problems. As we will see in Section 16.4, the SVD should be used for rank deficient problems.

16.2.4 Remark on Curve Fitting

To fit a polynomial of degree n to a set of points(x_i, y_i), you can avoid using the normal equations by computing the vandermonde matrix $V(x, n)$ and solving the least-squares problem $Vx = y$ using the *QR* or SVD approach.

16.3 CONDITIONING OF LEAST-SQUARES PROBLEMS

A perturbation analysis of least-squares problems is quite complex. This section will present a theorem without proof and provide some examples to illustrate what the theorem tells us. As you might expect, the perturbation result depends of the condition number for an $m \times n$ matrix, $m \geq n$. We will see that least-squares problems are more sensitive to perturbations than the solution of linear systems using Gaussian elimination. A sketch of the proof and a discussion of its consequences can be found in Ref. [1, pp. 117-118].

Theorem 16.3. *Assume that δA and δb are perturbations of the full-rank matrix $A \in \mathbb{R}^{m\times n}(m \geq n)$ and the vector $b \in \mathbb{R}^m$, respectively. Assume that x is the unique solution to the least-squares problem $Ax = b$ and that $\hat{x}$ is a solution to the least-squares problem $(A + \Delta A)\,\hat{x} = b + \Delta b$. Let r be the residual $r = b - Ax$, and assume that*

$$\epsilon = \max\left(\frac{\|\delta A\|_2}{\|A\|_2}, \frac{\|\delta b\|_2}{\|b\|_2}\right) < \frac{1}{\kappa\,(A)}.$$

Then,

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

where $\sin\theta = \frac{\|r\|_2}{\|b\|_2}$. *In other words,* θ, $0 < \theta < \frac{\pi}{2}$, *is the angle between the vectors b and Ax (Figure 16.1).*

There are a number of important consequences of Theorem 16.3.

a. If θ is small, the residual is small (Figure 16.1), and the sensitivity to perturbations depends on $\kappa(A)$.
b. If θ is not close to $\frac{\pi}{2}$, but $\kappa(A)$ is large, then the sensitivity depends on$(\kappa(A))^2$.
c. If θ is near $\frac{\pi}{2}$, the solution is nearly zero, but the solution to the least-squares problem, $\hat{x}$, produced by perturbations in A and b will not be zero, and $\frac{\|\hat{x}-x\|_2}{\|x\|_2}$ will be very large.

Example 16.7. In Ref. [49], Golub refers to the matrix

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

Choose $\epsilon = 0.01$. The following MATLAB script solves the least-squares problem $Ax = b$ with $b = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}}$ using the function `svdlstsq`. Then, the nonzero elements of A are each perturbed by the same random value in the range $0 < \delta a_{ij} < 0.00001$, the elements of b perturbed by random values in the range $0 < \delta b_i < 0.001$, and `svdlstsq` finds the solution to the perturbed problem. Then, the code computes the norm of the difference between the two solutions. The result is interesting.

```
>> B = .00001*rand*A + A;
>> c = b + .001*rand(6,1);
>> x = svdlstsq(A,b);
>> y = svdlstsq(B,c);
>> norm(x-y)

ans =

   0.038215669993397
```

There is a very good reason for this. The condition number $\kappa(A) = \|A\|_2 \left\|A^{\ddagger}\right\| = 223.609$. That does not seem too bad, but remember Theorem 16.3 says that the relative error can be bounded by the square of the condition number, and $(223.609)^2 = 50001$. ■

16.3.1 Sensitivity when using the Normal Equations

[19], p. 118, states the following result that is an alternative to the bound in Theorem 16.3:

$$\frac{\|\tilde{x} - x\|_2}{\|x\|_2} \leq \frac{\epsilon\kappa_2(A)}{1 - \epsilon\kappa_2(A)} \left(2 + (\kappa_2(A) + 1)\frac{\|r\|_2}{\|A\|_2 \|x\|_2}\right),$$

where r is the residual $r = \|b - Ax\|_2$. Using the normal equations requires solving $A^T Ax = A^T b$, and so the accuracy depends on the condition number $\kappa_2\left(A^T A\right)$. From Theorem 10.5, part (4)

$$\kappa_2\left(A^T A\right) = \kappa_2^2(A).$$

Thus the relative error, $\frac{\|\tilde{x}-x\|_2}{\|x\|_2}$, is always bounded by $\kappa_2^2(A)$ rather than $\kappa_2(A)$. As a result, use of the normal equations can lose twice as many digits of accuracy compared to the use of the *QR* or SVD method. Using the normal equations is only acceptable when the condition number of A is small, and in this case it is faster than either the *QR* or SVD approaches.

16.4 RANK-DEFICIENT LEAST-SQUARES PROBLEMS

To this point, we have dealt with least-squares problems $Ax = b$, $m \geq n$ and A having full rank. What happens if $m \geq n$ and rank $(A) < n$? Such problems are termed *rank deficient*, and arise in areas of science and engineering such as acoustics,

tomography, electromagnetic scattering, image restoration, remote sensing, and the study of atmospheres, so there has to be a means of handling them.

Recall that a solution using the reduced QR decomposition requires solving $Rx = Q^{\mathrm{T}}b$. In a rank-deficient problem, the reduced QR decomposition gives an upper triangular matrix R that is singular, and there will be infinitely many solutions to the least-squares problem; in other words there are infinitely many vectors x that minimize $\|b - Ax\|_2$ for $x \in \mathbb{R}^n$.

It is also possible that a problem is "almost rank deficient."

Example 16.8. Let $A = \begin{bmatrix} 1.0000 & -0.3499 \\ -2.0000 & 0.6998 \\ 8.0000 & -2.8001 \end{bmatrix}$. In the reduced QR decomposition,

$$R = \begin{bmatrix} 8.3066 & -2.9072 \\ 0 & 0.0002 \end{bmatrix},$$

and R is very close to singular and is singular to 3-digit accuracy. Using the QR method for computing the unique least-squares solution in situations like this will be highly sensitive to perturbations or will be impossible if R is singular. ■

The solution to rank-deficient or nearly rank-deficient problems is to make a wise choice among the solutions, and the choice is to pick the solution with the smallest 2-norm, the *minimum norm solution*. There are two approaches to solving such a problem, using the QR method with column pivoting [2, pp. 288-298], or using the SVD. We will choose the latter.

Let $A \in \mathbb{R}^{m\times n}$ have rank r. The full SVD has the following form:

$$A = U \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} V^{\mathrm{T}}$$

$$\begin{matrix} r & m-r & & r & n-r \end{matrix}$$

$$\mathrm{U} = [U_r, \quad U_{m-r}], \quad V = [V_r \quad V_{n-r}]$$

where U_r is the submatrix consisting of the first r columns, and so forth. The SVD is a very revealing orthogonal decomposition, as we saw in Section 15.2.1 when we discussed the four fundamental subspaces of a matrix. The SVD can be used to determine a formula for the minimum norm least-squares solution x_{LS} as well as the norm of the minimum residual r_{LS}. Theorem 16.4 shows how to compute x_{LS} and r_{LS}:

$$x_{\mathrm{LS}} = \sum_{i=1}^{r} \left(\frac{u_i^{\mathrm{T}} b}{\sigma_i} \right) v_i,$$

and the corresponding residual

$$r_{\mathrm{LS}} = \sqrt{\sum_{i=r+1}^{m} \left(u_i^{\mathrm{T}} b\right)^2}.$$

Computing the rank is a nontrivial problem. After computing the SVD, MATLAB determines the rank as follows:

```
tol=max(size(A))*eps(max(sigma))
r=sum(sigma > tol)
```

In the computation of `tol`, `max(size(A))` is the largest dimension of A, and `eps(max(sigma))` is the distance between σ_1 and the next largest floating point number in double precision arithmetic. To estimate the rank, r, find the number of singular values larger than tol. Theorem 16.4 provides a proof that x_{LS} and r_{LS} are correct, and can be omitted if desired.

Theorem 16.4. *Let $A = U\widetilde{\Sigma}V^{\mathrm{T}}$ be the SVD of $A^{m\times n}$, with $r = \mathrm{rank}\,(A)$. If $U = \begin{bmatrix} u_1 & \dots & u_m \end{bmatrix}$ and $V = \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}$ are the columns of U and V, and $b \in \mathbb{R}^m$, then*

$$x_{\mathrm{LS}} = \sum_{i=1}^{r} \left(\frac{u_i^{\mathrm{T}} b}{\sigma_i} \right) v_i$$

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

$$r_{\mathrm{LS}}^2 = \|b - Ax_{\mathrm{LS}}\|_2^2 = \sum_{i=r+1}^{m} \left(u_i^{\mathrm{T}} b\right)^2.$$

Proof. Multiplication by an orthogonal matrix does not change the 2-norm, so

$$\|b - Ax\|_2^2 = \left\|U^{\mathrm{T}}(b - Ax)\right\|_2^2. \tag{16.6}$$

Using the fact that $U^{\mathrm{T}}AV = \widetilde{\Sigma}$, Equation 16.6 gives

$$\|b - Ax\|_2^2 = \left\|U^{\mathrm{T}}b - \left(U^{\mathrm{T}}Ax\right)\right\|_2^2 = \left\|U^{\mathrm{T}}b - \left(U^{\mathrm{T}}AV\right)\left(V^{\mathrm{T}}x\right)\right\|_2^2 = \left\|U^{\mathrm{T}}b - \widetilde{\Sigma}\alpha\right\|_2^2, \tag{16.7}$$

where $\alpha = V^{\mathrm{T}}x = \begin{bmatrix} \alpha_1 & \dots & \alpha_n \end{bmatrix}^{\mathrm{T}}$. Now,

$$\widetilde{\Sigma}\alpha = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix}\alpha = \begin{bmatrix} \overset{r}{\sigma_1\alpha_1} & \dots & \sigma_r\alpha_r & \overset{m-r}{0} & \dots & 0 \end{bmatrix}^{\mathrm{T}}.$$

Write U^{T} as $\begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix}$, where u_i are the rows of U^{T}, and then note that row i of the product $U^{\mathrm{T}}b$ is the inner product of u_i and b, which is $u_i^{\mathrm{T}}b$, so we obtain

$$U^{\mathrm{T}}b = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix} b = \begin{bmatrix} u_1^{\mathrm{T}}b \\ \vdots \\ u_m^{\mathrm{T}}b \end{bmatrix}. \tag{16.8}$$

Use Equations 16.8 and 16.9 in Equation 16.7, to obtain

$$\|b - Ax\|_2^2 = \left\| \begin{bmatrix} u_1^{\mathrm{T}}b - \sigma_1\alpha_1 \\ \vdots \\ u_r^{\mathrm{T}}b - \sigma_r\alpha_r \\ u_{r+1}^{\mathrm{T}}b \\ \vdots \\ u_m^{\mathrm{T}}b \end{bmatrix} \right\|_2^2,$$

from which we get

$$\|b - Ax\|_2^2 = \sum_{i=1}^{r}\left(u_i^{\mathrm{T}}b - \sigma_i\alpha_i\right)^2 + \sum_{i=r+1}^{m}\left(u_i^{\mathrm{T}}b\right)^2. \tag{16.9}$$

Notice that the term $\sum_{i=r+1}^{m}\left(u_i^{\mathrm{T}}b\right)^2$ in Equation 16.9 is independent of x, so the minimum value of $\|Ax - b\|_2^2$ occurs when

$$\alpha_i = \frac{u_i^{\mathrm{T}}b}{\sigma_i}, \quad 1 \le i \le r. \tag{16.10}$$

Since $\alpha = V^{\mathrm{T}}x$, $x = V\alpha$, from Equation 16.10, we have

$$x = \sum_{i=1}^{r}\left(\frac{u_i^{\mathrm{T}}b}{\sigma_i}\right)v_i + \sum_{i=r+1}^{n}\alpha_i v_i \tag{16.11}$$

If we choose x such that $\alpha_i = 0$, $r+1 \le i \le n$, and then x will have the smallest possible 2-norm, and the minimum residual is

$$r_{\mathrm{LS}} = \sqrt{\sum_{i=r+1}^{m}\left(u_i^{\mathrm{T}}b\right)^2}. \tag{16.12}$$

□

Remark 16.2. If $m \ge n$ and rank $(A) = n$, then the term $\sum_{i=r+1}^{n}\alpha_i v_i$ is not present in Equation 16.11, and the solution is unique. If A is rank deficient, then $r < n$, and there are infinitely many ways to choose α_i, $r+1 \le i \le n$, so the rank-deficient

least-squares problem has infinitely many solutions. However, there is only one solution x_{LS} with minimum norm; that is, when we choose $\alpha_i = 0$, $r+1 \le i \le n$ so that

$$x_{LS} = \sum_{i=1}^{r} \left(\frac{u_i^T b}{\sigma_i} \right) v_i. \tag{16.13}$$

Example 16.9. Let $A = \begin{bmatrix} 2 & 1 & 1 & 2 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 \\ 2 & 2 & 2 & 3 \end{bmatrix}$, $b = \begin{bmatrix} -1 \\ 5 \\ 3 \\ 2 \end{bmatrix}$. The rank of A is 3, so A is rank deficient and we will apply the results of Theorem 16.4. First, compute the SVD:

$$A = \begin{bmatrix} -0.4364 & 0.8165 & 0 & -0.3780 \\ -0.4364 & -0.4082 & 0.7071 & -0.3780 \\ -0.4364 & -0.4082 & 0.7071 & -0.3780 \\ -0.6547 & 0 & 0 & 0.7559 \end{bmatrix} \begin{bmatrix} 7 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -0.4364 & -0.4364 & -0.4364 & -0.6547 \\ 0.8165 & -0.4082 & -0.4082 & 0 \\ 0 & 0.7071 & -0.7071 & 0 \\ -0.3780 & -0.3780 & -0.3780 & 0.7559 \end{bmatrix}$$

Equation 16.13 gives the minimum norm solution.

$$\begin{aligned} x_{LS} &= \sum_{i=1}^{3} \left(\frac{u_i^T b}{\sigma_i} \right) v_i \\ &= \frac{\begin{bmatrix} -0.4364 & -0.4364 & -0.4364 & -0.6547 \end{bmatrix} \begin{bmatrix} -1 & 5 & 3 & 2 \end{bmatrix}^T}{7} \begin{bmatrix} -0.4364 \\ -0.4364 \\ -0.4364 \\ -0.6547 \end{bmatrix} + \cdots = \begin{bmatrix} -3.0612 \\ 2.9388 \\ 0.9388 \\ 0.4082 \end{bmatrix} \end{aligned}$$

■

Algorithm 16.4 finds the minimum norm solution to the rank-deficient least-squares problem, but it will work for a full-rank problem; however, with full-rank problems, use either svdlstsq or qrlstsq.

Algorithm 16.4 Minimum Norm Solution to the Least-Squares Problem

```
function RDLSTSQ(A,b)
   % Compute the minimum norm solution to the
   % linear least-squares problem using the SVD.
   % Input: m × n matrix A and m × 1 vector b.
   % Output: Solution x and the residual
   [ U  Σ̃  V ] = svd(A)
   σ = diag(Σ̃)
   % Compute the rank of A.
   tol = max(size(A)) eps(max(σ))
   r = Σ(elements of σ > tol)
   x = 0
   for i=1:r do
      x = x + ( (U(:, i)^T b) / σ(i) ) V(:, i)
   end for
   residual = 0
   for i=r+1:m do
      residual = residual + (b^T U(:, i))^2
   end for
   return[ x  √residual ]
end function
```

NLALIB: The function `rdlstsq` implements Algorithm 16.4.

Example 16.10. Let

$$A = \begin{bmatrix} 7 & 1 & 8 \\ -1 & 5 & 0 \\ -1 & 6 & 9 \\ 0 & 2 & 9 \\ 1 & 2 & 3 \\ 3 & 5 & 7 \end{bmatrix}, \quad b1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

and

$$B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad b2 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}.$$

The rank of A is 3, so it is not rank deficient, and `qrlstsq` applies. However, the rank of B is 1 and we must use `rdlstsq`.

```
>> x1 = qrlstsq(A,b1)
x1 =
    0.0618
    0.1548
    0.0454
>> x = rdlstsq(B,b2)
x =
    0.8333
    0.8333
    0.8333
```

■

It is now appropriate to introduce an equivalent definition for the pseudoinverse that is a helpful theoretical tool.

Definition 16.6. Let A be an $m \times n$ matrix, $m \geq n$, having rank $r \leq n$, with SVD $A = U\tilde{\Sigma}V^{\mathrm{T}}$. The pseudoinverse is

$$A^{\dagger} = V\Sigma^{+}U^{\mathrm{T}},$$

where

$$\Sigma^{+} = \operatorname{diag}\left(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \ldots, \frac{1}{\sigma_r}, 0, 0, \ldots, 0\right)$$

is an $n \times m$ diagonal matrix. If A has full rank, this definition is equivalent to Definition 16.3 (Problem 16.11).

This definition enables us to reformulate the results of Theorem 16.4 using the pseudoinverse, namely, that $x_{\mathrm{LS}} = A^{\dagger}b$. Let u_i, $1 \leq i \leq m$, be the columns of U, and v_i, $1 \leq i \leq n$, be the columns of V. Begin by computing $U^{\mathrm{T}}b$, which we can write as

$$U^{\mathrm{T}}b = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_r \\ u_{r+1} \\ \vdots \\ u_m \end{bmatrix} b = \begin{bmatrix} \langle u_1, b\rangle \\ \langle u_2, b\rangle \\ \vdots \\ \langle u_r, b\rangle \\ \langle u_{r+1}, b\rangle \\ \vdots \\ \langle u_m, b\rangle \end{bmatrix},$$

and then

$$\Sigma^{+}U^{\mathrm{T}}b = \begin{bmatrix} \frac{1}{\sigma_1}\langle u_1,\, b\rangle \\ \frac{1}{\sigma_2}\langle u_2,\, b\rangle \\ \vdots \\ \frac{1}{\sigma_r}\langle u_r,\, b\rangle \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

To finish up,

$$\begin{aligned} V\Sigma^{+}U^{\mathrm{T}}b &= \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_1}\langle u_1,\, b\rangle \\ \frac{1}{\sigma_2}\langle u_2,\, b\rangle \\ \vdots \\ \frac{1}{\sigma_r}\langle u_r,\, b\rangle \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{\sigma_1}\langle u_1,\, b\rangle v_{11} + \frac{1}{\sigma_2}\langle u_2,\, b\rangle v_{12} + \cdots + \frac{1}{\sigma_r}\langle u_r,\, b\rangle v_{1r} \\ \frac{1}{\sigma_1}\langle u_1,\, b\rangle v_{21} + \frac{1}{\sigma_2}\langle u_2,\, b\rangle v_{22} + \cdots + \frac{1}{\sigma_r}\langle u_r,\, b\rangle v_{2r} \\ \vdots \\ \frac{1}{\sigma_1}\langle u_1,\, b\rangle v_{n1} + \frac{1}{\sigma_2}\langle u_2,\, b\rangle v_{n2} + \cdots + \frac{1}{\sigma_r}\langle u_r,\, b\rangle v_{nr} \end{bmatrix} \\ &= \frac{1}{\sigma_1}\langle u_1,\, b\rangle v_1 + \frac{1}{\sigma_2}\langle u_1,\, b\rangle v_2 + \cdots \frac{1}{\sigma_r}\langle u_1,\, b\rangle v_r \\ &= \sum_{i=1}^{r}\left(\frac{1}{\sigma_i}u_i^{\mathrm{T}}b\right)v_i = x_{\mathrm{LS}} \end{aligned}$$

It is also the case that

$$\|b - Ax_{LS}\|_2 = \left\|\left(I^{m\times m} - AA^{\dagger}\right)b\right\|_2 \tag{16.14}$$

(Problem 16.12). The algorithm rdlstsq is more efficiently stated as is, rather than using the pseudoinverse. However, the pseudoinverse plays an important role in developing theorems dealing with least squares.

16.4.1 Efficiency

The full SVD computation requires approximately $4m^2n + 8mn^2 + 9n^3$ flops [2, p. 493]. Once the SVD computation is complete, the rank computation requires negligible effort. The for loop executes r times. Each execution requires 1 division, 1 addition, $2m$ flops for computing $U(:, i)^{\mathrm{T}}\, b$, and n multiplications, so the cost of the for loop is $r\,(2m + n + 2)$. This is negligible, so the cost of `rdlstsq` is $\approx 4m^2n + 8mn^2 + 9n^3$.

Remark 16.3. If a matrix is almost rank deficient (has one or more small singular values), an SVD approach should be used. The algorithm rdlstsq is a good choice, but svdlstsq can also be used.

16.5 UNDERDETERMINED LINEAR SYSTEMS

Consider the least-squares problem $Ax = b$, where A is an $m \times n$ matrix, $m < n$, and b is an $m \times 1$ vector, and the solution x is has dimension $n \times 1$. Such a system is called *underdetermined* (Figure 16.6).

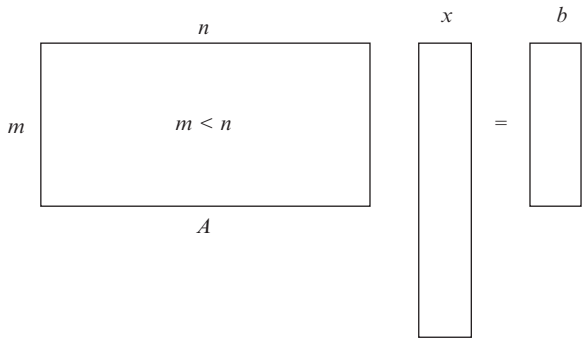

FIGURE 16.6 Underdetermined system.

If A has full rank (rank $(A) = m$), then the matrix A^{T} with dimension $n \times m$, $n > m$ has full rank. The QR decomposition can be used to obtain a solution to a full-rank underdetermined system by applying it to A^{T}. Compute the full QR decomposition $A^{\mathrm{T}} = QR$, where Q is $n \times n$ and R is $n \times m$. Then, using block matrix notation

$$Q^{\mathrm{T}}A^{\mathrm{T}} = R = \begin{matrix} \\ m \\ n-m \end{matrix} \overset{m}{\begin{bmatrix} R_1 \\ 0 \end{bmatrix}}$$

where R_1 is an $m \times m$ upper triangular matrix. Now, $A = R^{\mathrm{T}}Q^{\mathrm{T}}$, so $Ax = b$ can be written as $R^{\mathrm{T}}\left(Q^{\mathrm{T}}x\right) = b$. Again using block matrix notation the equation becomes

$$\begin{matrix} \\ m \end{matrix} \overset{\begin{matrix} m & n-m \end{matrix}}{\begin{bmatrix} R_1^{\mathrm{T}} & 0 \end{bmatrix}} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix},$$

where $y = Q^{\mathrm{T}}x$. Solve $R_1^{\mathrm{T}} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = b$, and let $y = \begin{bmatrix} y_1 & \dots & y_m & 0 & \dots & 0 \end{bmatrix}^{\mathrm{T}}$. We must now compute $x = Qy$ to obtain the solution. There is no need to perform the entire $(n \times n)$ $(n \times 1)$ product due to the presence of the $n - m$ zeros in y. We have

$$Qy = \begin{bmatrix} q_{11} & \cdots & q_{1m} & q_{1,m+1} & \cdots & q_{1n} \\ q_{21} & \cdots & q_{2m} & q_{2,m+1} & \cdots & q_{2n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q_{n1} & \cdots & q_{nm} & q_{n,m+1} & \cdots & q_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} & q_{1,m+1} & \cdots & q_{1n} \\ & q_{2,m+1} & \cdots & q_{2n} \\ Q_1 & \vdots & \vdots & \vdots \\ & \vdots & \vdots & \vdots \\ & q_{n,m+1} & \cdots & q_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$= Q_1^{n \times m} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}$$

Algorithm 16.5 implements this process.

Algorithm 16.5 Solution of Full-Rank Underdetermined System Using QR Decomposition

```
function UQRLSTSQ(A,b)
  % Solve the m × n full rank underdetermined system Ax=b
  % using the QR decomposition
  % Input: m × n matrix A, m < n, and m × 1 vector b.
  % Output: minimum norm solution n × 1 vector x and the residual
  [ Q  R ] = qr(A^T)
  R_1 = R(1 : m, 1 : m)
  Solve the m × m lower triangular system R_1^T y = b
  Q_1 = Q(1 : n, 1 : m)
  x = Q_1 y
  return[ x  ‖b − Ax‖_2 ]
end function
```

NLALIB: The function `uqrlstsq` implements Algorithm 16.5.

Example 16.11. Find the minimum norm solution to the full-rank system $Ax = b$, where

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

and

$$b = \begin{bmatrix} 1 & 5 & 8 \end{bmatrix}^{\mathrm{T}}.$$

```
>> rank(A)

ans =
     3

>> uqrlstsq(A,b)

ans =
  -18.4286
   13.6000
   -7.5143
   -2.0571
    3.4000
```

■

The results of Theorem 16.4 apply to the underdetermined rank deficient problems as well. By changing a_{32} from 12 to 7, the matrix A has rank 2, and we must use the algorithm implemented by the function `rdlstsq`.

Example 16.12. $A = \begin{bmatrix} 1 & 3 & 5 & 7 & 9 \\ -1 & -2 & -3 & -4 & -5 \\ 6 & 7 & 8 & 9 & 10 \end{bmatrix}$ and $b = \begin{bmatrix} 1 & 5 & 8 \end{bmatrix}^{\mathrm{T}}$.

```
>> x = rdlstsq(A,b)

x =
    1.1741
    0.7361
    0.2980
   -0.1401
   -0.5782
```

■

Remark 16.4. The MATLAB function `pinv` computes the pseudoinverse and can be used for overdetermined, underdetermined, or rank-deficient problems in the following way:

```
x = pinv(A)*b
```

16.5.1 Efficiency

The QR decomposition requires approximately $4\left(n^2m - nm^2 + \frac{m^3}{3}\right)$ flops when Q is required. The solution of the $m \times m$ lower triangular system requires $m^2 + m - 1$ flops, and forming the product $Q_1 y$ costs $2mn$ flops. The QR decomposition dominates, so the cost is approximately $4\left(n^2m - nm^2 + \frac{m^3}{3}\right)$ flops.

16.6 CHAPTER SUMMARY

The Least-Squares Problem

If A is an $m \times n$ matrix, a least-squares solution to the problem $Ax = b$, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$ is a value of x for which $\|b - Ax\|_2$ is minimum. Figure 16.1 shows that the residual vector, $b - Ax$ must be orthogonal to the range of A, a subset of $\mathbb{R}^m$. This leads to the fact that the normal equations $A^{\mathrm{T}}Ax = A^{\mathrm{T}}b$ must be satisfied by any least-squares solution.

Existence, Uniqueness, and the Normal Equations

If A is an $m \times n$ matrix, $m \geq n$, a vector $x \in \mathbb{R}^n$ is a least-squares solution if and only if x satisfies the normal equations; furthermore, x is unique if and only if A has full rank.

The Pseudoinverse

The pseudoinverse, $A^{\dagger} = \left(A^{\mathrm{T}}A\right)^{-1}A^{\mathrm{T}}$, is a generalization of the square matrix inverse; in fact, if A is square and nonsingular, $A^{\dagger} = A^{-1}$. The pseudoinverse allows the definition of the condition number for a general $m \times n$ matrix as $\kappa(A) = \left\|A^{\dagger}\right\|_2 \|A\|_2$.

Overdetermined Problems

A least-squares problem is overdetermined if A has dimension $m \times n$, $m > n$; in other words, there are more equations than unknowns. This is the most common type of least-squares problem.

Solving Overdetermined Problems Using the Normal Equations

If A has full rank, solving the normal equations $A^{\mathrm{T}}Ax = A^{\mathrm{T}}b$ by applying the Cholesky decomposition to the symmetric positive definite matrix $A^{\mathrm{T}}A$ yields a unique least-squares solution. Despite its simplicity, this approach is almost never used, since

- There may be some loss of significant digits when computing $A^{\mathrm{T}}A$. It may even be singular to working precision.
- The accuracy of the solution using the normal equations depends on the square of condition number of the matrix. If $\kappa(A)$ is large, the results can be seriously in error.

Solving Overdetermined Problems Using the *QR* Decomposition

This is a time-honored technique for solving full-rank overdetermined least-squares problems. Find the reduced QR decomposition of A, and the solution is $Rx = Q^{\mathrm{T}}b$, easily obtained using back substitution. The primary cost is the computation of the QR decomposition.

Solving overdetermined Problems Using the SVD

By first computing the reduced SVD of A, $A = U\Sigma V^{\mathrm{T}}$, the solution to the full-rank overdetermined least-squares problem is found by first solving $\Sigma y = U^{\mathrm{T}}b$, followed by computing $x = Vy$.

Conditioning of Least-Squares Problems

Perturbation analysis for least-squares problems is complex. If θ is the angle between b and the range of Ax, then

a. if θ is small, the residual is small (Figure 16.1), and the sensitivity to perturbations depends on $\kappa(A)$.
b. if θ is not close to $\frac{\pi}{2}$, but $\kappa(A)$ is large, then the sensitivity depends on$(\kappa(A))^2$.
c. if θ is near $\frac{\pi}{2}$, the solution is nearly zero, but the solution to the least-squares problem, $\hat{x}$, produced by perturbations in A and b will not be zero, and $\frac{\|\hat{x}-x\|_2}{\|x\|_2}$ will be very large.

Thus, assuming that b is not almost orthogonal to $R(A)$, relative errors depend at best on $\kappa(A)$ and at worst on $(\kappa(A))^2$.

Rank Deficient Problems

An overdetermined problem is rank deficient if rank $(A) < n$. Such systems have infinitely many solutions, so the strategy is to determine the unique solution $\hat{x}$ with minimum norm among the solutions. After computing the SVD of A, there is a simple formula for the minimum norm solution. The section also develops an alternative but equivalent definition of the pseudoinverse. This formulation is primarily a theoretical tool.

Underdetermined Problems

A problem is underdetermined when A has dimension $m \times n$, $m < n$. The matrix A^{T} is of full rank, and a solution can be obtained using the QR decomposition and submatrix operations that avoid computations dealing with blocks of zeros.

16.7 PROBLEMS

16.1 Three variables x, y, and z are required to satisfy the equations

$$\begin{aligned} 3x - y + 7z &= 0 \\ 2x - y + 4z &= 1/2 \\ x - y + z &= 1 \\ 6x - 4y + 10z &= 3 \end{aligned}$$

Is there a solution in the normal sense, not a least-squares solution? If so, is it unique, or are there infinitely many solutions?

16.2 If A is an invertible matrix and assuming exact arithmetic, are the solutions using Gaussian elimination and least squares identical? Prove your assertion.

16.3 If A is an $m \times n$ matrix, $m \geq n$, and b is an $m \times 1$ vector, show that if $A^{\mathrm{T}}b = 0$, b is orthogonal to the range of A.

16.4 If A is an $m \times n$ full-rank matrix, $m \geq n$, show that

$$\kappa(A) = \left\|A^{\ddagger}\right\|_2 \|A\|_2 = \frac{\sigma_1}{\sigma_n},$$

where σ_1 and σ_n are the largest and smallest singular values of A, respectively.

16.5

a. What is the Moore-Penrose inverse of a nonzero column vector?
b. Answer the same question for a row vector.

16.6

a. If A is a full rank $m \times n$ matrix, with $m < n$, show that

$$x = A^{\mathrm{T}}\left(AA^{\mathrm{T}}\right)^{-1} b + \left(I - A^{\mathrm{T}}\left(AA^{\mathrm{T}}\right)^{-1} A\right) y$$

satisfies the normal equations, where y is any $n \times 1$ vector. If $y = 0$, $x = A^{\mathrm{T}}\left(AA^{\mathrm{T}}\right)^{-1} b$ is the minimum norm solution.
b. Show how to evaluate x without computing $\left(AA^{\mathrm{T}}\right)^{-1}$.
c. Is using this formula a reliable way to compute x? Explain your answer.

16.7 There are many identities involving the pseudoinverse. Let $A \in \mathbb{R}^{m \times n}$. Show that
a. $A^{\dagger}AA^{\dagger} = A^{\dagger}$

b. $\left(A^{\dagger}A\right)^{\mathrm{T}} = I$
c. $AA^{\dagger}A = A$
d. $\left(AA^{\dagger}\right)^{\mathrm{T}} = AA^{\dagger}$

16.8 Show that if $A \in \mathbb{R}^{m\times n}$,
a. $\left(I - AA^{\dagger}\right)A = 0$
b. $A\left(I - A^{\dagger}A\right) = 0$

16.9 Show that if A has orthonormal columns, then $A^{\dagger} = A^{\mathrm{T}}$.

16.10 If $A \in \mathbb{R}^{m\times n}$, then $B^{n\times m}$ is a *right inverse* of A if $AB = I^{m\times m}$. $C^{n\times m}$ is a *left inverse* of A if $CA = I^{n\times n}$.
a. Prove that if rank $(A) = n$, $A^{\dagger}$ is a left inverse of A.
b. Prove that if rank $(A) = m$, $A^{\dagger}$ is a right inverse of A.

16.11 If $A^{m\times n}$, $m \geq n$, is of full rank, show that Definitions 16.3 and 16.6 for $A^{\dagger}$ are equivalent.

16.12 Show that $\|b - Ax_{LS}\|_2 = \left\|\left(I - AA^{\dagger}\right)b\right\|_2$.

16.13 Prove that all solutions $\hat{x}$ of $\min_{x\in\mathbb{R}^n} \|Ax - b\|_2$ are of the form $\hat{x} = A^{\dagger}b + \eta$, where $\eta \in \text{null}\,(A)$.

16.14 If some components of $Ax - b$ are more important than others, a weight $w_i > 0$ can be attached to each component. By forming the diagonal matrix $W = \begin{bmatrix} w_1 & & 0 \\ & \ddots & \\ 0 & & w_m \end{bmatrix}$, the problem becomes one of minimizing the norm of $W(Ax - b)$. This called a *weighted least-squares* problem. The approach is to define an inner product and minimize the norm of $Ax - b$ relative to this inner product. Of course, this inner product must use W in some way.
a. If S is a symmetric positive definite matrix, define $\langle x, y\rangle_S = x^{\mathrm{T}}Sy$, and verify it is an inner product by showing that
i. $\langle x, y + z\rangle_S = \langle x, y\rangle_S + \langle x, z\rangle_S$
ii. $\langle cx, y\rangle_S = \langle x, cy\rangle_S = c\langle x, y\rangle_S$, where c is a scalar
iii. $\langle x, y\rangle_S = \langle y, x\rangle_S$
iv. $\langle x, 0\rangle_S = 0$
v. If $\langle x, x\rangle_S = 0$, then $x = 0$.
b. Show that the weight matrix W is positive definite.
c. Using an argument similar to the one used to derive the normal equations $A^{\mathrm{T}}Ax = A^{\mathrm{T}}b$, derive, $A^{\mathrm{T}}WAx = A^{\mathrm{T}}Wb$, the normal equations for the weighted least-squares problem.
d. Let $W^{\frac{1}{2}} = \text{diag}\left(\sqrt{w_1}, \sqrt{w_2}, \ldots, \sqrt{w_m}\right)$, so that $\left(W^{\frac{1}{2}}\right)^2 = W$. Rewrite the normal equations in part (c) so that the problem becomes an ordinary least-squares problem with the rescaled matrix $W^{\frac{1}{2}}A$.
e. Develop an algorithm to solve the weighted least-squares problem, $m \geq n$, using the QR decomposition.

16.15 In Ref. [2, pp. 288-291], there is a discussion of rank deficient least-squares problem that includes some sensitivity issues, which are more complex than those for full-rank problems. It is stated that small changes in A and b can cause large changes to the solution, $x_{\text{LS}} = A^{\dagger}b$. The sensitivity is related to the behavior of the pseudoinverse. Using the example in Ref. [2], let $A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$ and $\delta A = \begin{bmatrix} 0 & 0 \\ 0 & \epsilon \\ 0 & 0 \end{bmatrix}$.
a. Show that

$$\left\|A^{\dagger} - (A + \delta A)^{\dagger}\right\|_2 = \frac{1}{\epsilon}.$$

b. What does this say about

$$\lim_{\delta A\to 0} \left\|A^{\dagger} - (A + \delta A)^{\dagger}\right\|_F?$$

16.7.1 MATLAB Problems

16.16 To facilitate printing vectors, implement a function

```
printvec(msg,x)
```

that outputs the string `msg`, followed by the elements of `x`, eight per line. Place the file `printvec.m` in a directory of your choice, and add the directory to the MATLAB search path.

16.17 Using least squares, fit a line and a parabola to the data. Plot the two curves on the same set of axes.

x	0	2	3	5	8	11	12	15
y	50	56	60	72	85	100	110	125

Compute the condition number of the associated Vandermonde matrix in each case.

16.18 Let

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 3 \\ 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 5 \\ 1 \end{bmatrix}.$$

In parts (a)–(d), find the unique least-squares solution x using

a. $x = A^{\dagger}b$

b. The normal equations

c. The QR method

d. The SVD method

e. Find $\kappa\,(A)$.

16.19 The data in the table give the approximate population of the United States every decade from 1900 to 1990.

Year	1900	1910	1920	1930	1940	1950	1960	1970	1980	1990
Populations (millions)	76.1	92.4	106.5	123.1	132.1	152.3	180.7	205.1	227.2	249.4

Assume that the population growth can be modeled with an exponential function $p = be^{mx}$, where x is the year and p is the population in millions. Use least squares to determine the constants b and m for which the function best fits the data, and graph the data and the exponential curve on the same set of axes. Use the equation to estimate the population in the years 1998, 2010, and 2030.

16.20 Consider the following data:

x	1.0	1.4	1.9	2.2	2.8	3.0
y	0.26667	0.23529	0.20513	0.19048	0.16667	0.16

Determine the coefficients α and β in the function $y = \frac{1}{\alpha x+\beta}$ that best fits the data using least squares, and graph the data and the curve $\frac{1}{\alpha x+\beta}$ on the same set of axes.

16.21 The following are values of a function of the form $y = \frac{x}{\alpha+\beta x}$. Using least squares, estimate α and β and approximate y at $x = \begin{Bmatrix} 25.0 & 33.8 & 36.0 \end{Bmatrix}$. Graph the data and the curve $\frac{x}{\alpha x+\beta}$ on the same set of axes.

x	20	21.3	21.9	30.6	32.0	33.3
y	1.0152	1.027	1.032	1.0859	1.0922	1.0976

16.22 Radium-226 decays exponentially. The data shows the amount, y, of radium-226 in grams after t years. Find an exponential function $y = Ae^{rt}$ which best fits the data. Estimate the half-life of radium-226. Graph the data and the curve $y = Ae^{rt}$ on the same set of axes.

t	0	100	350	500	1000	1800	2000
y	10	9.5734	8.5847	8.0413	6.4662	4.5621	4.1811

16.23 The table shows the length-weight relation for Pacific halibut. Find a power function $W = \alpha L^{\beta}$ which best fits the data. Graph the data and the curve $W = \alpha L^{\beta}$ on the same set of axes.

L	0.5	1.0	1.5	2.0	2.5
W	1.3	10.4	35	82	163

16.24 The equation for a stretched beam is

$$y = l + Fs,$$

where l is the original length, F is the force applied, and s is the inverse coefficient of stiffness. The following measurements were taken:

F	20	25	27	30	33
y	22.3	22.8	23.2	23.5	25.5

Estimate the initial length and the inverse coefficient of stiffness.

16.25 A resistor in an electric circuit satisfies the equation $V = RI$, where V is the voltage and I is the current. The following measurements of voltage and current were taken:

I	2.00	1.90	1.85	2.06	1.95
V	4.95	5.05	5.10	4.92	5.02

Estimate the resistance.

16.26 The equation $z = Ax + By + D$ represents a plane, and the points in the table are on or close to a plane. Use least squares to determine the plane, and graph the data points and the plane on the same set of xyz-axes.

x	−1.5	−1.0	−1.9	0.0	0.5	1.0	1.3	1.95
y	1.0	1.6	1.5	−1.4	−1.3	−1.6	−1.8	1.9
z	1.0	1.4	−0.3	6.4	7.3	8.6	9.4	7.0

The following MATLAB statements plot the data points:

```
% plot the data points in black circles
H=plot3(x,y,z,'ko');
% double the circle size
set(H,'Markersize',2*get(H,'Markersize'));
% fill the circles with black
set(H,'Markerfacecolor','k');
hold on;
```

To plot the plane, use the functions `meshgrid` and `surf`. After calling `surf`, the statement `alpha(0.5)` will assure that the plane will not obscure portions of the circles.

16.27 Graph the following data, and then fit and graph polynomials of degree 1, 2, and 3. All graphs must be on the same set of axes. Which fit seems best?

x	−0.25	0.00	0.25	0.50	0.75	1.00	1.25	1.50	1.75	2.00
y	1.9047	1.3	0.63281	0.1375	0.048437	0.6	2.0266	4.5625	8.4422	13.9

16.28 The MATLAB command `pinv(A)` computes $A^{\dagger}$, the pseudoinverse. The MATLAB statement `pinv(A)*b` computes the solution to a least-squares problem. For each part, you are given a least-squares problem. Solve each problem two ways, using one of the functions developed in this chapter and by using `pinv`.

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

b.
$$\begin{bmatrix} 2 & 3 & 1 \\ -1 & 2 & 3 \\ 3 & 1 & -2 \\ 6 & 9 & 3 \end{bmatrix} x = \begin{bmatrix} -1 \\ 3 \\ 5 \\ 0 \end{bmatrix}$$

c. $\begin{bmatrix} 2 & 5 & 6 & 8 \\ -1 & 3 & 7 & 1 \\ 5 & 1 & 6 & 2 \end{bmatrix} x = \begin{bmatrix} 1 \\ -1 \\ 10 \end{bmatrix}$

d. $\begin{bmatrix} 1 & -6 & 2 & -4 \\ 3 & -18 & 6 & -12 \\ 9 & 0 & 12 & 0 \end{bmatrix} x = \begin{bmatrix} -2 \\ 8 \\ 12 \end{bmatrix}$

16.29 Let $A = \begin{bmatrix} 1 & 3 & -1 \\ 2 & 1 & 8 \\ 7 & 6 & 23 \end{bmatrix}$.

a. If $b = \begin{bmatrix} 1 & 1 & 3 \end{bmatrix}^{\mathrm{T}}$, use Gaussian elimination to determine if there is a unique solution, infinitely many solutions, or no solution to $Ax = b$.
b. If your answer to part (a) is infinitely many or no solutions, can you find a minimum norm solution?
c. For $b = \begin{bmatrix} 3 & 11 & 36 \end{bmatrix}^{\mathrm{T}}$, find the infinitely many solutions using Gaussian elimination.
d. Find the least-squares minimum norm solution to the rank deficient problem (c), and verify that the minimum norm solution is one of the infinitely many solutions obtained in part (c).

16.30 Let

$$A = \begin{bmatrix} 1 & -1 & 3 \\ 8 & 8 & 1 \\ 4 & 6 & -12 \\ 6 & -9 & 0 \\ 3 & 4 & 4 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 3 \\ -1 \\ 6 \\ 15 \end{bmatrix}.$$

a. Show that there is a unique solution.
b. Find the solution in the following ways:
i. Using the normal equations
ii. Using the *QR* method
iii. Using the SVD method

16.31 Let $A = \begin{bmatrix} 1 & 4 & 10 \\ -2 & 6 & -6 \\ 5 & -7 & 23 \\ 3 & 3 & 21 \end{bmatrix}$ and $b = \begin{bmatrix} 1 \\ 12 \\ -1 \\ 2 \end{bmatrix}$.

a. Try to solve the least-squares problem $Ax = b$ using the *QR* decomposition.
b. Try to solve the least-squares problem $Ax = b$ using the SVD.
c. Explain the results of parts (a) and (b).
d. Try to find a solution using `rdlstsq`. Explain your result.

16.32 The 50×40 matrix `RD.mat` in the software distribution is a rank deficient. Let `b = rand(50,1)`.
a. Find the minimum norm solution to `RD*x = b`.
b. Find two different solutions that produce the same residual, and show that their norms are greater than that of the minimum norm solution.

16.33 Build the matrix

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

and vectors

$$b_1 = \begin{bmatrix} -5 & 2 & 9 & 15 \end{bmatrix}^{\mathrm{T}}, \quad b_2 = \begin{bmatrix} 7 & 1 & 3 & 6 \end{bmatrix}^{\mathrm{T}}.$$

The rank of A is 2, so A is rank deficient. Find the minimal norm solution to the problems

$$\min_{x\in\mathbb{R}^3} \|Ax - b_1\| \quad \text{and} \quad \min_{x\in\mathbb{R}^3} \|Ax - b_2\|,$$

and compute $\|Ax_i - b_i\|$, $i = 1, 2$. Explain the results.

16.34

a. Show that the 50×35 matrix, ARD, in the software distribution is "almost rank deficient."

b. Let $b = \begin{bmatrix} 1 & 1 & \dots & 1 & 1 \end{bmatrix}^{\mathrm{T}}$ and solve the least-squares problem $ARDx = b$ using `normalsolve`, `qrlstsq`, `svdlstsq`, and `rdlstsq`. In each case, also compute the residual. Compare the results by filling-in the table.

Method	Residual
normalsolve	
qrlstsq	
svdlstsq	
rdlstsq	

16.35 Build the matrix A and vector b using the following statements:

```
A = rosser; A(9:10,:) = ones(2,8); b = ones(10,1).
```

a. Compute `x = pinv(A)*b`, and accept x as the correct solution.

b. Solve the least-squares problem $Ax = b$ using the normal equations to obtain solution x_1.

c. Do part (a), except use the QR decomposition algorithm to obtain solution x_2.

d. Do part (a), except use the SVD decomposition algorithm to obtain solution x_3.

e. Compute $\frac{\|x-x_i\|_2}{\|x\|_2}$, $i = 1, 2, 3$.

f. Explain the results, taking Remark 16.2 into account.

16.36 Build the Lauchli matrix `A = gallery('lauchli',50)`, and let `b = (1:51)'`.

a. Does A have full rank?

b. Compute the condition number of A.

c. Compute the least-squares solution, x_1, using `qrlstsq`.

d. Obtain solution x_2 by using the normal equations.

e. Comment on the results.

16.37 Solve the underdetermined system and give the residual.

$$\begin{bmatrix} 2 & -4 & 4 & 0.077 \\ 0 & -2 & 2 & -0.056 \\ 2 & -2 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 3.86 \\ -3.47 \\ 0 \end{bmatrix}$$

16.38 Build a 9×25 system using the following MATLAB code, find the minimum norm solution and the residual.

```
>> x = 1:25;
>> V = vandermonde(x,8);
>> V = V';
>> b = (1:9)';
```

16.39 Find the rank of the matrix.

$$\begin{bmatrix} 1 & 3 & 6 & 2 & 1 \\ 3 & -7 & 9 & 2 & 1 \\ 5 & -9 & 51 & 14 & 7 \\ -16 & 36 & -66 & -16 & -8 \end{bmatrix} x = \begin{bmatrix} 1 \\ -1 \\ 3 \\ 5 \end{bmatrix}.$$

Will `uqrlstsq` work? If it does, use it, and if not find the minimum norm solution using `rdlstsq`.

16.40 A Toeplitz matrix is a matrix in which each diagonal from left to right is constant. For instance,

$$T = \begin{bmatrix} 5 & 4 & 3.001 & 2 & 1 & 1 & 2 & 3 \\ 6 & 5 & 4 & 3.001 & 2 & 1 & 1 & 2 \\ 7 & 6 & 5 & 4 & 3.001 & 2 & 1 & 1 \\ 8 & 7 & 6 & 5 & 4 & 3.001 & 2 & 1 \\ 9 & 8 & 7 & 6 & 5 & 4 & 3.001 & 2 \\ 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3.001 \\ 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 \\ 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 \\ 13 & 12 & 11 & 10 & 9 & 8 & 7 & 6 \\ 14 & 13 & 12 & 11 & 10 & 9 & 8 & 7 \\ 15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 \end{bmatrix}$$ is a Toeplitz matrix. This interesting matrix was found in Ref. [50]. We will use this matrix to investigate the conditioning of $m \times n$ least-squares problems, $m > n$.

a. Build T and vectors b and bp as follows:

```
>> r = [5 4 3.001 2 1 1 2 3];
>> c = [5 6 7 8 9 10 11 12 13 14 15];
>> T = toeplitz(c,r);
>> b = [21.001 24.001 29.001 36.001 44.001 52.001 60.0 68.0...
76.0 84.0 92.0]';
>> bp = [21.001 24.0002 29.0001 36.0003 44.00 52.0003 59.999...
68.0001 75.9999 84.0001 91.9999]';
```

b. Compute $\frac{\|b-bp\|_2}{\|b\|_2}$.

c. Use the function `svdlstsq` to solve the two problems

$$Tx = b, \quad T\,(xp) = bp,$$

d. Compute

$$\frac{\|x - xp\|_2}{\|x\|_2}.$$

e. Discuss your results.

16.41

a. Using the results of Problem 16.14, write a function

```
function [x, r] = wlstsq(A,b,w)
```

that solves the weighted least-squares problem. The argument `w` is a vector of weights, not a diagonal matrix.

b. Given weights $\begin{bmatrix} 2 & 4 & 5 & 1 & 6 \end{bmatrix}^{\mathrm{T}}$, solve the weighted least-squares problem using `wlstsq`.

$$\begin{aligned} x_1 + 2x_2 + x_3 - x_4 &= 1 \\ 2x_1 + 5x_2 - x_3 + x_4 &= 2 \\ 4x_1 + x_2 - 3x_3 - x_4 &= -1 \\ -x_1 + x_2 + 3x_3 + 7x_4 &= 0 \\ 5x_1 - x_2 + x_3 - 8x_4 &= 3 \end{aligned}$$

16.42 Section 11.9 discussed iterative improvement for the solution of a linear system. Iterative improvement can also be done for least-squares solutions. Consider the following algorithm outline for the full-rank overdetermined problem:

```
function LSTSQIMP(A,b,x,numsteps)
  % Iterative refinement for least squares.
  % Execute numsteps of improvement.
  % Normally numsteps is small.
  for i=1:numsteps do
     r = b − Ax
     correction=qrlstsq(A,r)
     x=x+correction
```

```
    end for
    return [x,b-Ax]
end function
```

a. Implement the function lstsqimp in MATLAB.
b. Add iterative refinement to the following code, and see if you can improve the result.

```
A = gallery('lauchli',50);
b = (1:51)';

[x,r] = normalsolve(A,b);
fprintf('The residual using the normal equations = %g\n',r);
```

Remark 16.5. This method is helpful only when the initial residual is small. The algorithm actually used in practice is described in Ref. [2, pp. 268-269].