Chapter 12

Linear System Applications

You should be familiar with

- Trigonometric functions and infinite series (Section 12.1)
- Evaluating an integral (Section 12.1)
- Derivatives of a function of one variable
- Partial derivatives (Sections 12.2 and 12.3)
- Acquaintance with finite difference equations (Section 12.2)
- Computing a local minima or maxima (Section 12.3)

We have discussed bases, dimensions, inner products, norms, and the *LU* decomposition with and without partial pivoting for the solution of $n \times n$ linear algebraic systems of equations. Linear algebra is used in almost all applied engineering and science applications. In many cases, its use is hidden from direct view, but is a critical component of an algorithm. Until this point, we have presented problems involving the solution to a truss, an electrical circuit, data encryption, as well as other applications. In this chapter, we discuss four additional applications that make use of linear algebra, Fourier series, finite difference techniques for solving partial differential equations, an introduction to least squares, and cubic spline interpolation.

12.1 FOURIER SERIES

To this point, we have dealt with finite-dimensional vector spaces, but there are many applications for vector spaces with infinitely many dimensions. A particularly important example is *Fourier series*. When we introduced the inner product of vectors in Chapter 6, we also presented the L^2 inner product of functions. If f and g are functions over the interval $a \le x \le b$, then

$$\langle f, g\rangle = \int_a^b f(x)\, g(x)\, dx, \quad \text{and} \quad \|f\|_{L^2}^2 = \int_a^b f^2(x)\, dx.$$

Consider the infinite sequence of trigonometric functions

$$\left\{\frac{1}{\sqrt{2\pi}}, \frac{\cos x}{\sqrt{\pi}}, \frac{\sin x}{\sqrt{\pi}}, \frac{\cos 2x}{\sqrt{\pi}}, \frac{\sin 2x}{\sqrt{\pi}}, \dots, \frac{\cos nx}{\sqrt{\pi}}, \frac{\sin nx}{\sqrt{\pi}}, \dots\right\}$$

A straightforward computation shows that this is an orthonormal sequence over the interval $-\pi \le x \le \pi$.

$$\left\langle \frac{\cos ix}{\sqrt{\pi}}, \frac{\cos jx}{\sqrt{\pi}}\right\rangle = \frac{1}{\pi}\int_{-\pi}^{\pi} \cos ix\, \cos jx\, dx = 0, \quad i \ne j,$$
$$\left\langle \frac{\sin ix}{\sqrt{\pi}}, \frac{\sin jx}{\sqrt{\pi}}\right\rangle = \frac{1}{\pi}\int_{-\pi}^{\pi} \sin ix\, \sin jx\, dx = 0, \quad i \ne j,$$
$$\left\langle \frac{\cos ix}{\sqrt{\pi}}, \frac{\sin jx}{\sqrt{\pi}}\right\rangle = \frac{1}{\pi}\int_{-\pi}^{\pi} \cos ix\, \sin jx\, dx = 0, \quad i \ne j,$$

and

$$\left\langle \frac{\cos ix}{\sqrt{\pi}}, \frac{\cos ix}{\sqrt{\pi}}\right\rangle = \frac{1}{\pi}\int_{-\pi}^{\pi} \cos ix \cos ix\, dx = \frac{1}{\pi}\int_{-\pi}^{\pi} \cos^2(ix)\, dx = 1,$$
$$\left\langle \frac{\sin ix}{\sqrt{\pi}}, \frac{\sin ix}{\sqrt{\pi}}\right\rangle = \frac{1}{\pi}\int_{-\pi}^{\pi} \sin^2(ix)\, dx = 1.$$

Numerical Linear Algebra with Applications. http://dx.doi.org/10.1016/B978-0-12-394435-1.00012-0
Copyright © 2015 Elsevier Inc. All rights reserved.

We now consider this infinite sequence to be an orthonormal basis for what we term a *function space*, a set of linear combinations of the basis functions.

$$f(x) = a_0\left(\frac{1}{\sqrt{2\pi}}\right) + a_1\left(\frac{\cos x}{\sqrt{\pi}}\right) + b_1\left(\frac{\sin x}{\sqrt{\pi}}\right) + a_2\left(\frac{\cos 2x}{\sqrt{\pi}}\right) + b_2\left(\frac{\sin 2x}{\sqrt{\pi}}\right) + \cdots =$$
$$a_0\left(\frac{1}{\sqrt{2\pi}}\right) + \frac{1}{\sqrt{\pi}}\sum_{i=1}^{\infty}(a_i \cos \mathrm{i}x + b_i \sin \mathrm{i}x) \tag{12.1}$$

In general, such a linear combination is an infinite series. Not all functions belong to this function space; for instance, let $a_i = 1, b_i = 0, i \geq 0$. In this case, the series is

$$f(x) = \frac{1}{\sqrt{2\pi}} + \frac{1}{\sqrt{\pi}}(\cos x + \cos 2x + \cos 3x + \cdots).$$

If $x = 0$, then $f(0) = \frac{1}{\sqrt{2\pi}} + \frac{1}{\sqrt{\pi}}(1 + 1 + 1 + \cdots)$ is infinite. For f to belong to the function space, we require that $\|f\|_{L^2}^2$ be finite. Since

$$\left\{\frac{1}{\sqrt{2\pi}}, \frac{\cos x}{\sqrt{\pi}}, \frac{\sin x}{\sqrt{\pi}}, \frac{\cos 2x}{\sqrt{\pi}}, \frac{\sin 2x}{\sqrt{\pi}}, \ldots, \frac{\cos nx}{\sqrt{\pi}}, \frac{\sin nx}{\sqrt{\pi}}, \ldots\right\}$$

is an orthonormal sequence

$$\langle f, f\rangle = \int_{-\pi}^{\pi} f^2(x)\,dx = a_0^2 + a_1^2 + b_1^2 + a_2^2 + b_2^2 + \cdots + a_n^2 + b_n^2 + \cdots. \tag{12.2}$$

Thus, a function is in this function space if the series 12.2 converges, and we call it the *Fourier series* for f. Since the function is composed of periodic functions, f is periodic. It can be shown that this function space obeys the rules for a vector space, and the Cauchy-Schwarz inequality, $\langle f, g\rangle \leq \|f\|_{L^2}\|g\|_{L^2}$ holds. This function space is an example of a *Hilbert space*, a mathematical concept that has many applications in such areas as electrical engineering, vibration analysis, optics, acoustics, signal and image processing, econometrics, and quantum mechanics.

The a_i, b_i in Equation 12.1 are called the *Fourier coefficients* for f. So far, we have said that if the series 12.2 converges, the function it defines is in the function space. Assume for the moment that the function f has a Fourier series. How do we compute the Fourier coefficients? The answer lies in the fact that the basis is orthonormal relative to the inner product. We'll compute the coefficients of the terms $\cos \mathrm{i}x$, $0 \leq i < \infty$. Start with

$$f(x) = a_0\left(\frac{1}{\sqrt{2\pi}}\right) + \frac{1}{\sqrt{\pi}}\sum_{i=1}^{\infty}(a_i \cos \mathrm{i}x + b_i \sin \mathrm{i}x). \tag{12.3}$$

Multiply both sides of Equation 12.3 by $\cos kx$ for any $k \geq 1$ and integrate from $-\pi$ to π.

$$\begin{aligned}\int_{-\pi}^{\pi} f(x)\cos kx &= a_0\left(\frac{1}{\sqrt{2\pi}}\right)\int_{-\pi}^{\pi}\cos kx\,\mathrm{d}x + \frac{1}{\sqrt{\pi}}\int_{-\pi}^{\pi}\left(\sum_{i=1}^{\infty}(a_i\cos kx\cos \mathrm{i}x + b_i \cos kx \sin \mathrm{i}x)\right)\mathrm{d}x \\ &= 0 + \frac{1}{\sqrt{\pi}}\sum_{i=1}^{\infty}\left(\int_{-\pi}^{\pi}(a_i\cos kx\cos \mathrm{i}x)\,\mathrm{d}x + \int_{-\pi}^{\pi}(b_i \sin kx \sin \mathrm{i}x)\,\mathrm{d}x\right) = a_k\frac{1}{\sqrt{\pi}}\int_{-\pi}^{\pi}\cos^2 kx = \sqrt{\pi}a_k,\end{aligned}$$

and so

$$a_k = \frac{1}{\sqrt{\pi}}\int_{-\pi}^{\pi} f(x)\cos kx, \quad k \geq 1.$$

In a similar fashion,

$$b_k = \frac{1}{\sqrt{\pi}}\int_{-\pi}^{\pi} f(x)\sin kx, \quad k \geq 1.$$

It only remains to compute a_0:

$$\int_{-\pi}^{\pi} f(x)\,\mathrm{d}x = a_0\left(\frac{1}{\sqrt{2\pi}}\right)\int_{-\pi}^{\pi}\mathrm{d}x + \frac{1}{\sqrt{\pi}}\sum_{i=1}^{\infty}\left(\int_{-\pi}^{\pi}(a_i\cos \mathrm{i}x)\,\mathrm{d}x + \int_{-\pi}^{\pi}(b_i\sin \mathrm{i}x)\,\mathrm{d}x\right) = \sqrt{2\pi}a_0 + 0 = \sqrt{2\pi}a_0.$$

Thus,

$$a_0 = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} f(x)\,\mathrm{d}x.$$

Summary: The Fourier coefficients for the function f, $-\pi \le x \le \pi$, are

$$a_0 = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} f(x)\,dx.$$

$$a_k = \frac{1}{\sqrt{\pi}} \int_{-\pi}^{\pi} f(x)\cos kx\,dx,\ k \ge 1.$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\sin kx\,dx,\ k \ge 1.$$

Now, if we have a periodic function f, is there a Fourier series that converges to f? The following theorem can be found in the literature on Fourier series.

Theorem 12.1. *Assume that f has period 2π and is piecewise continuously differentiable on $-\pi \le x \le \pi$. Then the Fourier series*

$$a_0\left(\frac{1}{\sqrt{2\pi}}\right) + \frac{1}{\sqrt{\pi}} \sum_{i=1}^{\infty} (a_i \cos \mathrm{i}x + b_i \sin \mathrm{i}x)$$

converges at every point at which f is continuous and otherwise to

$$\frac{f(x+) + f(x-)}{2},$$

where

$$f(l+) = \lim_{x \to l+} f(x)$$

and

$$f(l-) = \lim_{x \to l-} f(x)$$

are the right and left-side limits of f at x.

Remark 12.1. Our discussion of Fourier series deals with functions having period 2π. If the function is periodic over another interval, $-L \le x \le L$, let $t = \pi x/L$ and

$$g(t) = f(x) = f\left(\frac{Lt}{\pi}\right).$$

g has period 2π, and $x = -L, x = L$ correspond to $t = -\pi$, $t = \pi$, respectively. Find the Fourier series for $g(t)$ and, using substitution, find the Fourier series for $f(x)$.

12.1.1 The Square Wave

The *square wave* is a very useful function in many engineering applications. It is periodic and, in this example, has period 2π, and is $+1$ in one-half the period and -1 in the other half (Figure 12.1). The Fourier series for the square wave is not difficult to compute. All the cosine terms in its Fourier series are 0, since the function is odd ($f(-x) = -f(x)$). The series is

$$\frac{4}{\pi}\left[\frac{\sin x}{1} + \frac{\sin 3x}{3} + \frac{\sin 5x}{5} + \cdots\right].$$

It is interesting to see how the series converges (Figure 12.2). As $n \to \infty$, the partial sums of sine terms "wiggle less and less" and converge to the line segments comprising the wave. The book software contains the sound file square.mp3. Play it to hear this waveform.

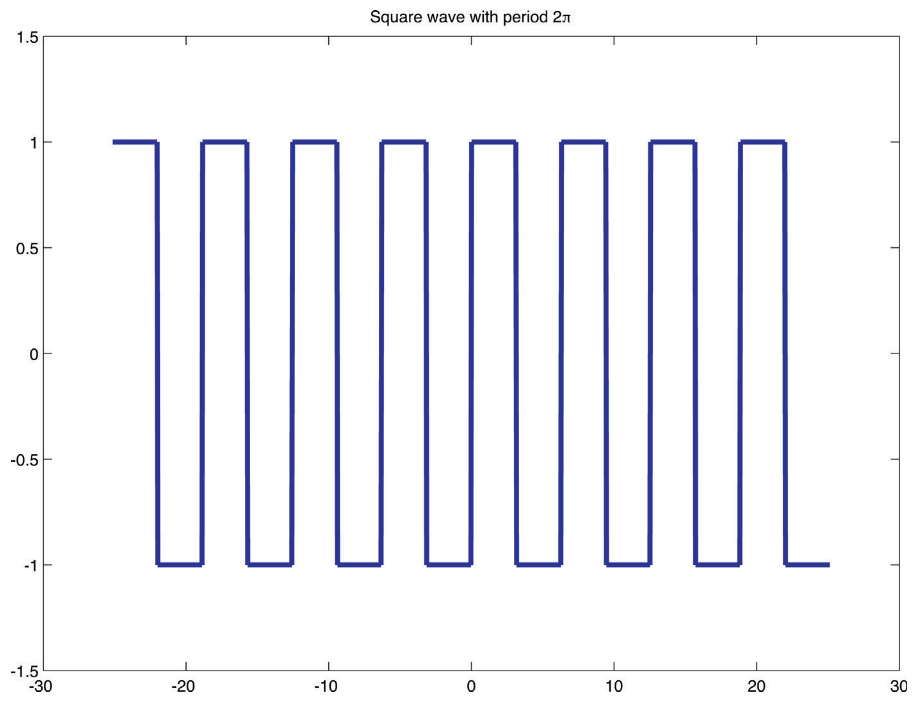


FIGURE 12.1 Square wave with period 2π.

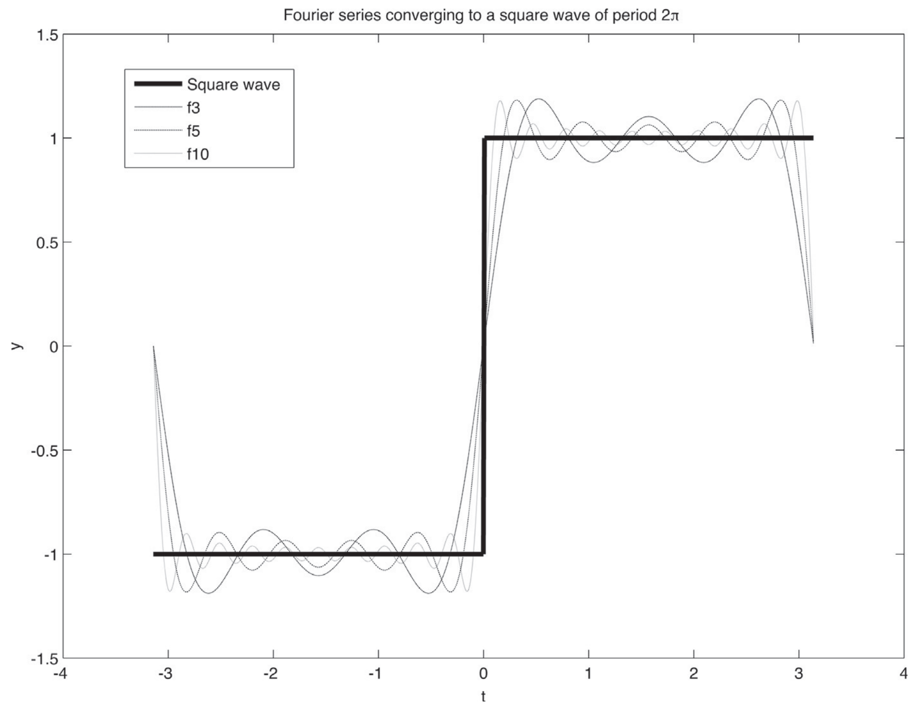


FIGURE 12.2 Fourier series converging to a square wave.

12.2 FINITE DIFFERENCE APPROXIMATIONS

Differential equations are at the heart of many engineering problems. They appear in fluid mechanics, thermodynamics, vibration analysis, and many other areas. For most of the problems that appear in practice, we cannot find an analytical solution and must rely on numerical techniques. Applications of linear algebra appear particularly in what are termed *initial-boundary value problems*. In problems like these, an initial condition is known at $t = 0$, and the value of the solution is known on the boundary of an object. We must use these known values to approximate the solution in the interior of the object.

12.2.1 Steady-State Heat and Diffusion

Suppose a thin rod is given an initial temperature distribution, then insulated on the sides. The ends of the rod are kept at the same fixed temperature; e.g., suppose at the start of the experiment, both ends are immediately plunged into ice water. We are interested in how the temperature along the rod varies with time. Suppose that the rod has a length L (in meters), and we establish a coordinate system along the rod as illustrated in Figure 12.3. Let $u(x, t)$ represent the temperature at the point x meters along the rod at time t (in seconds). We start with an initial temperature distribution $u(x, 0) = f(x)$. This problem is modeled using a partial differential equation called the *heat equation*:

$$\begin{aligned} \frac{\partial u}{\partial t} &= c\frac{\partial^2 u}{\partial x^2}, \quad 0 \le x \le L, \quad 0 \le t \le T, \\ u(0,t) &= u(L,t) = 0, \\ u(x,0) &= f(x). \end{aligned}$$

The constant c is the *thermal diffusivity*. For most functions $f(x)$, we cannot obtain an exact solution to the problem, so we must resort to numerical methods. So where does linear algebra come in? We divide the space interval $0 \le x \le L$ into small subintervals of length $h = L/m$, and the time interval into small subintervals of length $k = T/n$, where m and n are integers (Figure 12.4). We wish to approximate $u(x, t)$ at the grid points (x_i, t_j), where $(x_1, x_2, x_3, \ldots, x_m, x_{m+1}) = (0, h, 2h, \ldots, L - h, L)$ and $(t_1, t_2, t_3, \ldots, t_n, t_{n+1}) = (0, k, 2k, \ldots, T - k, T)$. We denote this approximate solution by

$$u_{ij} \approx u(x_i, t_j).$$

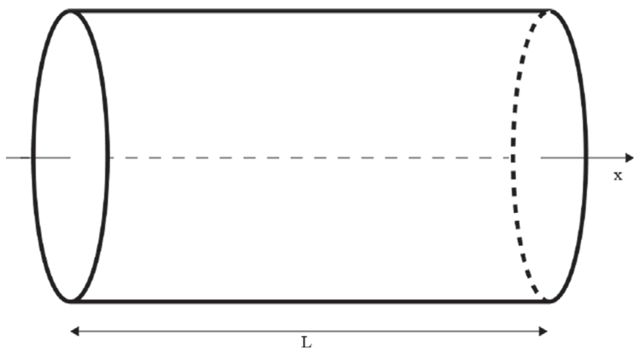


FIGURE 12.3 The heat equation: a thin rod insulated on its sides.

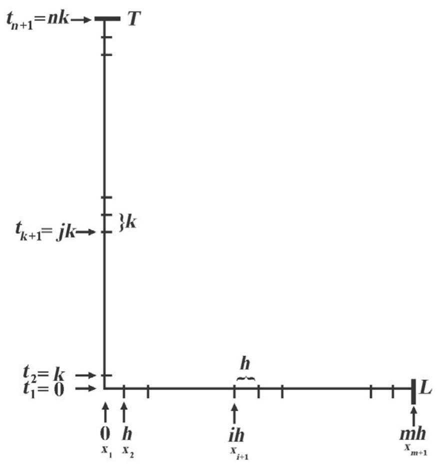


FIGURE 12.4 Numerical solution of the heat equation: subdivisions of the x and t axes.

If m and n are large, the values of h and k are small, and we can approximate each derivative at a point (x_i, t_j) using *finite difference equations* as follows:

$$\frac{\partial u}{\partial t}(x_i, t_{j+1}) \approx \frac{u_{i,j+1} - u_{ij}}{k}, \quad 1 \le j \le n, \tag{12.4}$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_{j+1}) \approx \frac{1}{h^2}\left(u_{i-1,j+1} - 2u_{i,j+1} + u_{i+1,j+1}\right), \quad 2 \le i \le m. \tag{12.5}$$

These approximations are developed using Taylor series for a function of two variables, and the interested reader can refer to Ref. [33, Chapter 6], for an explanation. By equating these approximations, we have

$$\frac{u_{i,j+1} - u_{ij}}{k} = \frac{c}{h^2}\left(u_{i-1,j+1} - 2u_{i,j+1} + u_{i+1,j+1}\right).$$

All but one term in the finite difference formula involves $j+1$, so isolate it to obtain

$$u_{i,j} = -ru_{i-1,j+1} + (1+2r)\,u_{i,j+1} - ru_{i+1,j+1}, \; 2 \le i \le m, \; 1 \le j \le n, \tag{12.6}$$

where $r = ck/h^2$. Notice that Equations 12.4 and 12.5 specify relationships between points in the grid pattern of Figure 12.5. Figure 12.6 provides a view of the entire grid. In Figure 12.6, the black circles represent boundary and initial values and the open circles represent a general pattern of the four points used in each difference equation.

Using matrix notation, Equation 12.6 becomes

$$u_j = Bu_{j+1} - rb_{j+1}, \tag{12.7}$$

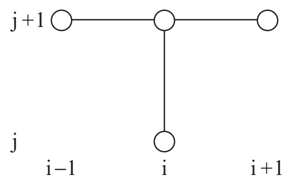


FIGURE 12.5 Numerical solution of the heat equation:locally related points in the grid.

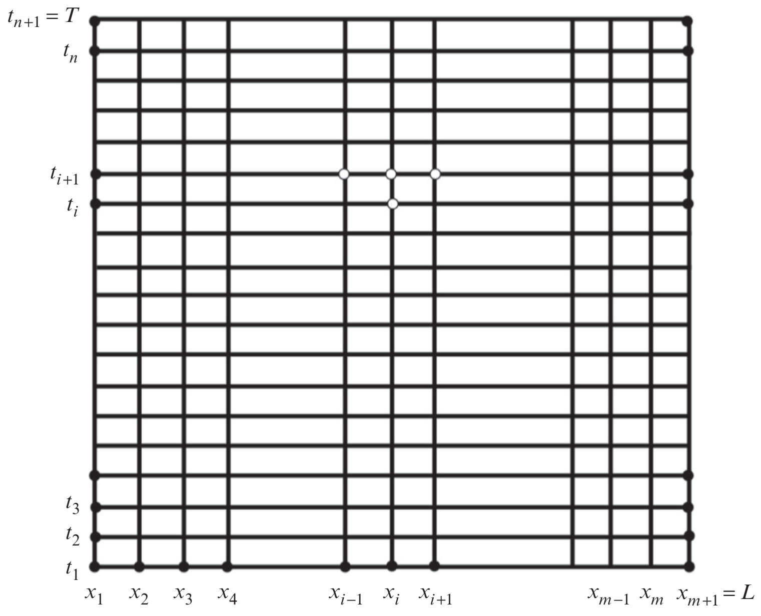


FIGURE 12.6 Grid for the numerical solution of the heat equation.

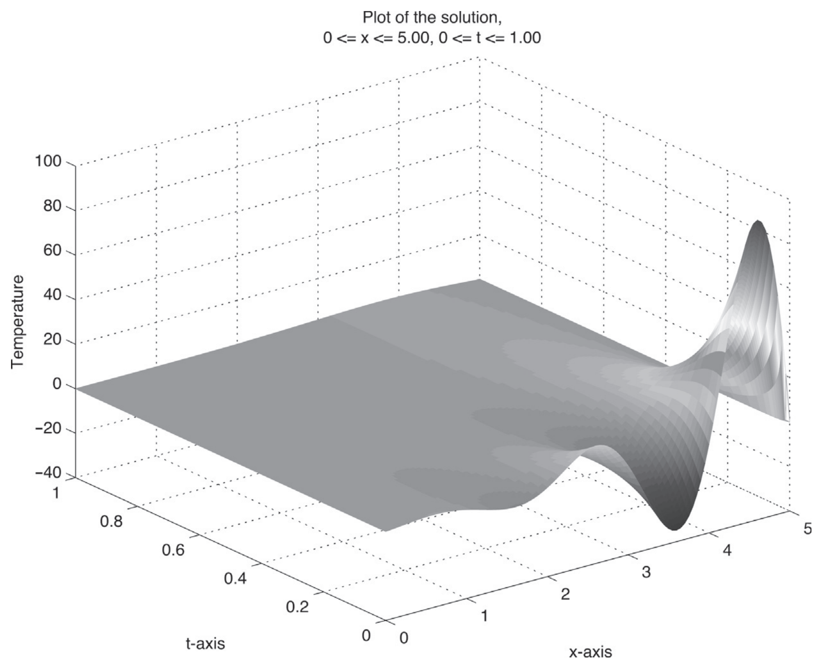


FIGURE 12.7 Graph of the solution for the heat equation problem.

where

$$B = \begin{bmatrix} 1+2r & -r & 0 & \dots & 0 \\ -r & 1+2r & -r & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & -r & 1+2r & -r \\ 0 & 0 & 0 & -r & 1+2r \end{bmatrix}$$

and b_{j+1} accounts for the boundary or initial conditions. Now, we can write Equation 12.7 as

$$Bu_{j+1} = u_j + rb_{j+1}. \tag{12.8}$$

This is a linear algebraic system to solve for u_{j+1}. Most of the elements of B are 0, so it is a *sparse matrix*; furthermore, it is *tridiagonal*. The Thomas algorithm presented in Section 9.4 takes advantage of the tridiagonal structure and solves system 12.7 in $O(n)$ flops as opposed to $O(n^3)$. Think of Equation 12.8 this way. Values of u_j and rb_{j+1} give us values for u_{j+1}, and we work our way up the grid. This is not a "chicken and an egg" problem, since the right-hand side of Equation 12.8 is known at time $t = 0$. We now give an example of our finite difference equations in action by approximating and graphing the solution to the heat flow problem

$$\begin{aligned} \frac{\partial u}{\partial t} &= 0.875\frac{\partial^2 u}{\partial x^2}, \quad 0 \le x \le 5.0, \quad 0 \le t \le 1.0, \\ u(0,t) &= u(5,t) = 0, \\ u(x,0) &= e^x \sin(\pi x). \end{aligned} \tag{12.9}$$

The required MATLAB code, `heateq.m`, is located in the software distribution. We present a graph of the approximate solution obtained in Figure 12.7.

We will deal with more problems like this in Chapters 20 and 21, where we discuss the Poisson and biharmonic equations.

12.3 LEAST-SQUARES POLYNOMIAL FITTING

We now consider a problem in data analysis. Assume that during an experiment, we collect m measurements of a quantity y that depends on a parameter t; in other words, we have the set of experimental points $(t_1, y_1), (t_2, y_2), \dots, (t_m, y_m)$. We want to use those m measurements to approximate other y values inside and outside the range t_1 to t_m, processes called *interpolation* and *extrapolation*, respectively. One approach to the problem is to fit a polynomial to the data in the "least-squares" sense. Assume the polynomial is

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2x^2 + a_1x + a_0.$$

Fit the polynomial by minimizing the sum of the squares of the deviation of each data point from the corresponding polynomial value; in other words, find the polynomial that will minimize the *residual*

$$E = \sum_{i=1}^{m} \left(y_i - \left(a_0 + a_1t_i + a_2t_i^2 + \cdots + a_nt_i^n\right)\right)^2.$$

We know from multivariable calculus that we do this by requiring the partial derivatives of E with respect to $a_0, a_1, \ldots, a_n$ be 0, or

$$\frac{\partial E}{\partial a_i} = 0, \quad 0 \le i \le n.$$

The partial derivatives are

$$\left.\begin{array}{rcl}
\frac{\partial E}{\partial a_0} & = & -2\sum_{i=1}^{m}\left(y_i - a_0 - a_1t_i - a_2t_i^2 - \cdots - a_nt_i^n\right) \\
\frac{\partial E}{\partial a_1} & = & -2\sum_{i=1}^{m} t_i\left(y_i - a_0 - a_1t_i - a_2t_i^2 - \cdots - a_nt_i^n\right) \\
\frac{\partial E}{\partial a_2} & = & -2\sum_{i=1}^{m} t_i^2\left(y_i - a_0 - a_1t_i - a_2t_i^2 - \cdots - a_nt_i^n\right) \\
\vdots & & \\
\frac{\partial E}{\partial a_m} & = & -2\sum_{i=1}^{m} t_i^n\left(y_i - a_0 - a_1t_i - a_2t_i^2 - \cdots - a_nt_i^n\right)
\end{array}\right\}.$$

Setting these equations to zero, we have

$$\begin{array}{rcl}
a_0m + a_1\sum_{i=1}^{m} t_i + a_2\sum_{i=1}^{m} t_i^2 + \cdots + a_n\sum_{i=1}^{m} t_i^n & = & \sum_{i=1}^{m} y_i \\
a_0\sum_{i=1}^{m} t_i + a_1\sum_{i=1}^{m} t_i^2 + \cdots + a_n\sum_{i=1}^{m} t_i^{n+1} & = & \sum_{i=1}^{m} t_iy_i \\
a_0\sum_{i=1}^{m} t_i^2 + a_1\sum_{i=1}^{m} t_i^3 + \cdots + a_n\sum_{i=1}^{m} t_i^{n+2} & = & \sum_{i=1}^{m} t_i^2y_i \\
 & \vdots & \\
a_0\sum_{i=1}^{m} t_i^n + a_1\sum_{i=1}^{m} t_i^{n+1} + \cdots + a_n\sum_{i=1}^{m} t_i^{2n} & = & \sum_{i=1}^{m} t_i^ny_i
\end{array} \tag{12.10}$$

Let $S_k = \sum_{i=1}^{m} t_i^k,\ k = 0, 1, \ldots, 2n$, $b_k = \sum_{i=1}^{m} t_i^k y_i, k = 0, 1, \ldots, n$, and write Equation 12.10 as a matrix equation.

$$\begin{bmatrix} S_0 & S_1 & \ldots & S_n \\ S_1 & S_2 & \ldots & S_{n+1} \\ S_2 & S_3 & \ldots & S_{n+2} \\ \vdots & \vdots & \ddots & \vdots \\ S_n & S_{n+1} & \ldots & S_{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}. \tag{12.11}$$

(Note that $S_0 = m$.) This is a system of $(n + 1)$ equations in $n + 1$ unknowns $a_0, a_1, \ldots, a_n$.

We can write system 12.11 in a different form. Define the $m \times (n + 1)$ *Vandermonde matrix and the vector* y

$$V = \begin{bmatrix} 1 & t_1 & \ldots & t_1^n \\ 1 & t_2 & \ldots & t_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & \ldots & t_m^n \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

Using Equation 12.11, it follows that $V^{\mathrm{T}}y = b$:

$$V^{\mathrm{T}}y = \begin{bmatrix} 1 & 1 & \dots & 1 \\ t_1 & t_2 & \dots & t_m \\ \vdots & \vdots & \ddots & \vdots \\ t_1^n & t_2^n & \dots & t_m^n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^m y_i \\ \sum_{i=1}^m t_i y_i \\ \vdots \\ \sum_{i=1}^m t_i^n y_i \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = b.$$

Now note that $V^{\mathrm{T}}Va = b$:

$$\begin{aligned} V^{\mathrm{T}}Va &= \begin{bmatrix} 1 & 1 & \dots & 1 \\ t_1 & t_2 & \dots & t_m \\ \vdots & \vdots & \ddots & \vdots \\ t_1^n & t_2^n & \dots & t_m^n \end{bmatrix} \begin{bmatrix} 1 & t_1 & \dots & t_1^n \\ 1 & t_2 & \dots & t_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & \dots & t_m^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} \\ &= \begin{bmatrix} S_0 & S_1 & \dots & S_n \\ S_1 & S_2 & \dots & S_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_n & S_{n+1} & \dots & S_{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} = b \end{aligned}$$

by referencing Equation 12.11. Thus,

$$V^{\mathrm{T}}Va = V^{\mathrm{T}}y = b.$$

A system of equations of this form is called the normal equations.

12.3.1 Normal Equations

Definition 12.1. Let $A \in R^{m\times n}$. The system of n equations and n unknowns

$$A^{\mathrm{T}}Ax = A^{\mathrm{T}}y$$

is called the *normal equations*.

Normal equations will become very important when we discuss linear least-squares problems in Chapter 16. Usually in these types of problems, either $m > n$ (*overdetermined system*) or $m < n$ (*underdetermined system*). If we have 25 data points and want to fit a straight line in the least-squares sense, we have $m = 25$ and $n = 2$.
To solve our least-squares problem, compute $V^{\mathrm{T}}y$ to obtain b. Then compute $V^{\mathrm{T}}V$ and solve the square system $\left(V^{\mathrm{T}}V\right) a = b$.

Example 12.1. The following eight data points show the relationship between the number of fishermen and the amount of fish (in thousand pounds) they can catch a day.

Number of Fishermen	Fish Caught
4	7
5	8
9	9
10	12
12	15
14	20
18	26
22	35

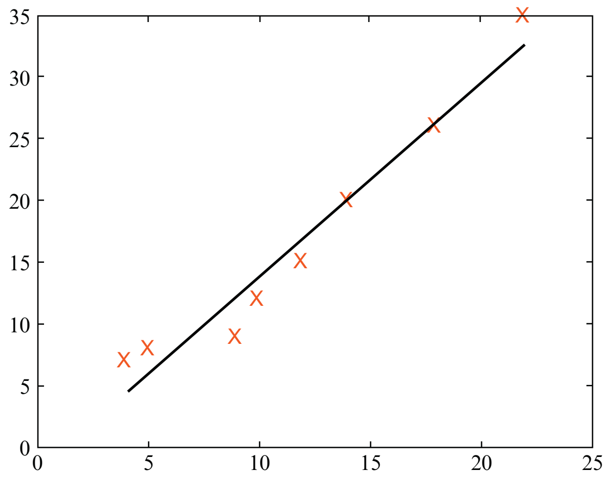


FIGURE 12.8 Linear least-squares approximation.

Case 1: First, we will fit a straight line ($n = 1$) to the data. The Vandermonde matrix V is of dimension 8×2, and its transpose has dimension 2×8, so $V^{\mathrm{T}}V$ is a 2×2 matrix. Compute

$$V^{\mathrm{T}}y = b = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 4 & 5 & 9 & 10 & 12 & 14 & 18 & 22 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \\ 9 \\ 12 \\ 15 \\ 20 \\ 26 \\ 35 \end{bmatrix} = \begin{bmatrix} 132 \\ 1967 \end{bmatrix}.$$

Now solve $V^{\mathrm{T}}Va = \begin{bmatrix} 132 \\ 1967 \end{bmatrix}$ to obtain the linear least-squares solution

$$a_0 = -1.9105, \quad a_1 = 1.5669.$$

Thus, the line that fits the data in the least-squares sense is $y = 1.5669x - 1.9105$. Figure 12.8 shows the data and the least-squares line (often called the *regression line*).

To estimate the value for 16 fishermen, compute $1.5669\,(16) - 1.9105 = 23.1591$.

Case 2: We will do a quadratic fit ($n = 2$).

$$V^{\mathrm{T}}y = \begin{bmatrix} 132 \\ 1967 \\ 33,685 \end{bmatrix} = b$$

Now solve $V^{\mathrm{T}}Va = \begin{bmatrix} 132 \\ 1967 \\ 33,685 \end{bmatrix}$ to obtain $a = \begin{bmatrix} 5.2060 \\ 0.1539 \\ 0.0554 \end{bmatrix}$. The best quadratic polynomial in the least-squares sense is $5.2060 + 0.1539x + 0.0554x^2$. Figure 12.9 shows the data and the least-squares quadratic polynomial.

To estimate the value for 16 fishermen, compute $5.2060 + 0.1539\,(16) + 0.0554(16)^2 = 21.8485$. ■

Remark 12.2. In Example 12.1, the quadratic polynomial appears to be more accurate; however, using a higher degree polynomial does not guarantee better results. As the dimensions of the Vandermonde matrices get larger, they become ill-conditioned.

Example 12.2. Estimating absolute zero.

Charles's Law for ideal gas states that at constant volume a linear relationship exists between the pressure p and the temperature t. An experiment takes gas in a sealed container that is initially submerged in ice water ($t = 0\,°\mathrm{C}$). The temperature is increased in 10° increments with the pressure measured each 10°.

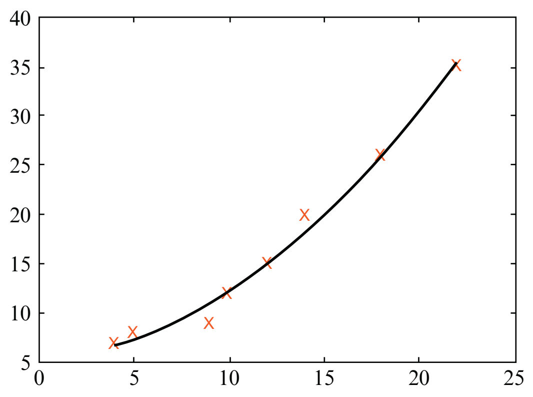


FIGURE 12.9 Quadratic least-squares approximation.

t	0	10	20	30	40	50	60	70	80	90	100
p	0.94	0.96	1.00	1.05	1.07	1.09	1.14	1.17	1.21	1.24	1.28

After finding a linear least-squares estimate, extrapolate the function to determine absolute zero, i.e., the temperature where the pressure is 0.

$$V^{\mathrm{T}}y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 10 & 20 & 30 & 40 & 50 & 60 & 70 & 80 & 90 & 100 \end{bmatrix} \begin{bmatrix} 0.94 \\ 0.96 \\ 1.00 \\ 1.05 \\ 1.07 \\ 1.09 \\ 1.14 \\ 1.17 \\ 1.21 \\ 1.24 \\ 1.28 \end{bmatrix} = \begin{bmatrix} 12.1500 \\ 645.1000 \end{bmatrix}.$$

Solve $V^{\mathrm{T}}Va = \begin{bmatrix} 12.1500 \\ 645.100 \end{bmatrix}$ to obtain the regression line $p = 0.0034t + 0.9336$. The temperature when $p = 0$ is $t_{\text{abs zero}} = -\frac{0.9336}{0.0034} = -273.1383$. The true value of absolute zero in Celsius is $-273.15\,^{\circ}\mathrm{C}$.

The function `vandermonde(t,m)` in the software distribution constructs the Vandermonde matrix. The following MATLAB program finds the least-squares approximation to absolute zero and draws a graph (Figure 12.10) showing the regression line and the data points.

```
>> t = 0:10:100;
>> V = vandermonde(t,1);
>> p = [.94 .96 1 1.05 1.07 1.09 1.14 1.17 1.21 1.24 1.28]';
>> b = V'*p;
>> a = (V'*V)\b

a =

    0.9336
    0.0034

>> abszero_approx = -a(1)/a(2)

abszero_approx =

 -273.1383

>> temp = [-300 100];
>> plot(t, p, 'o', temp, a(2)*temp + a(1));
```

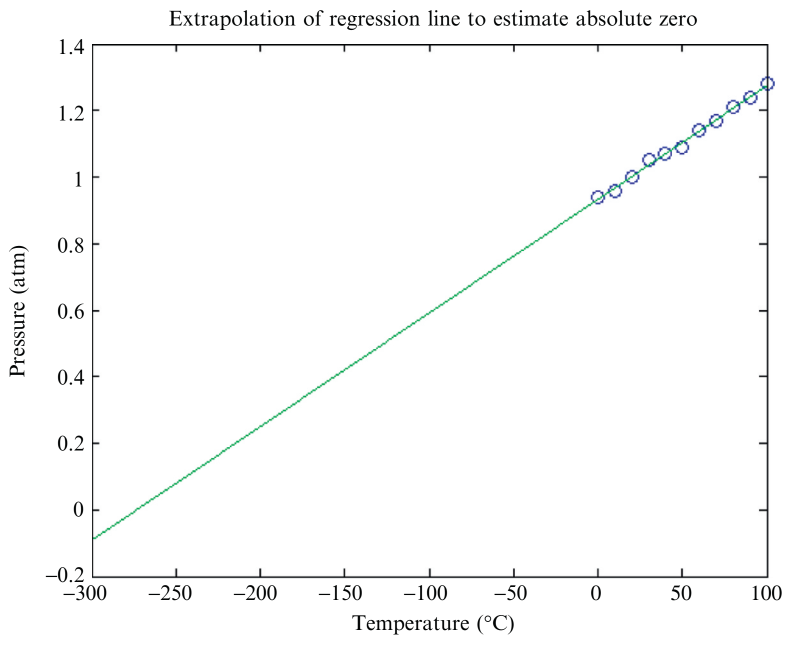


FIGURE 12.10 Estimating absolute zero.

```
>> xlabel('temperature (C)');
>> ylabel('pressure (atm)');
>> title('Extrapolation of Regression Line to Estimate Absolute Zero');
```
■

An alternative to using least-squares is *Lagrange interpolation*. This process takes n distinct data points and finds the unique polynomial of degree $n-1$ that passes through the points. Lagrange interpolation is discussed in the problems.

Both least-squares and Lagrange interpolation fit one polynomial to the data. An alternative is to fit a piecewise polynomial, and a premier method that uses this approach is cubic splines.

12.4 CUBIC SPLINE INTERPOLATION

Least squares can be used for either interpolation or extrapolation. Example 12.2 used extrapolation. When applying interpolation in a range $a \le t \le b$, estimates can be computed only for values in that range. Most engineers and scientists are familiar with *linear interpolation* (also called *linear splines*), in which the data points are joined by line segments, and a value between two data points is estimated using a point on the line segment.

Example 12.3. An experiment yields measurements

$$\{(1.3,\ 2.8),\ (1.7,\ 3.2),\ (1.9,\ 3.1),\ (2.3,\ 3.5),\ (2.7,\ 4.8),\ (3.1,\ 4.2),\ (3.6,\ 5.3),\ (4.0,\ 4.8)\}.$$

Given any value t in the range $t_i \le t \le t_{i+1}$

$$f(t) = y_i + \frac{(t-t_i)}{(t_{i+1}-t_i)}\left(y_{i+1}-y_i\right) \tag{12.12}$$

is a piecewise linear function defined for all points in the data set. For instance, approximate the measurement at $t = 2.5$. The value of t lies between $t_4 = 2.3$ and $t_5 = 2.7$. Using Equation 12.12, the approximate value is $f(2.5) = 3.5 + \frac{(2.5-2.3)}{(2.7-2.3)}(4.8-3.5) = 4.15$ (Figure 12.11) ■

Cubic splines use the same idea, but in a more sophisticated fashion. Instead of using a line segment between two points, the algorithm uses a cubic polynomial (Figure 12.12) to form a piecewise cubic function. The data points t_{i-1}, t_i where two polynomials from adjacent intervals meet are called *knots*. Following the development of cubic splines in Ref. [34] with assistance from Ref. [35], assume the data points are

$$\{(t_1, y_1), (t_2, y_2), (t_3, y_3), \ldots, (t_n, y_n), (t_{n+1}, y_{n+1})\},$$

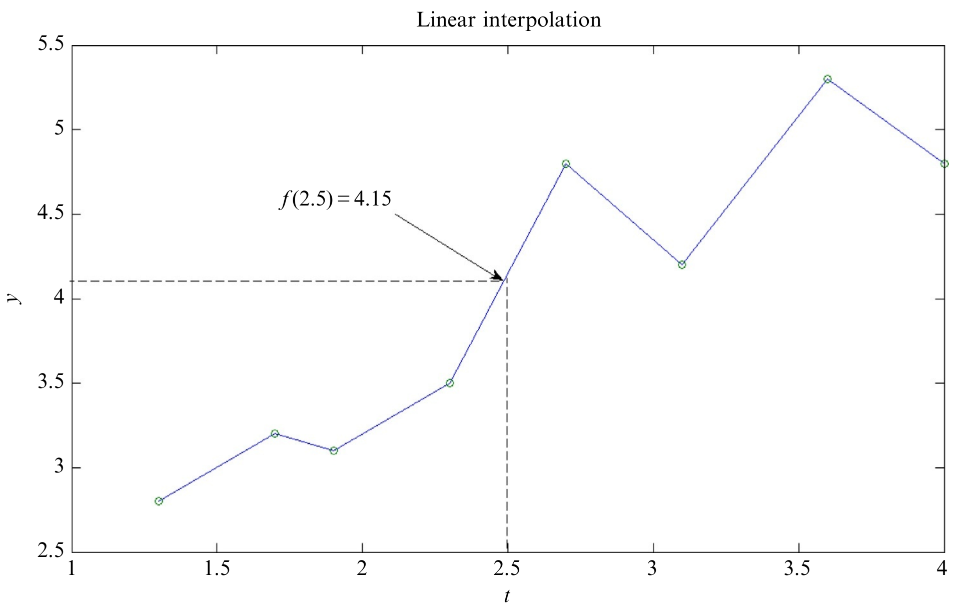
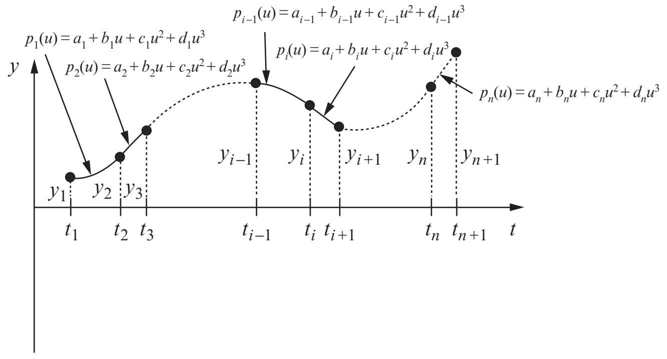


FIGURE 12.11 Linear interpolation.



$p_{i-1}(u) = a_{i-1} + b_{i-1}u + c_{i-1}u^2 + d_{i-1}u^3$
$p_1(u) = a_1 + b_1u + c_1u^2 + d_1u^3$
$p_2(u) = a_2 + b_2u + c_2u^2 + d_2u^3$
$p_i(u) = a_i + b_iu + c_iu^2 + d_iu^3$
$p_n(u) = a_n + b_nu + c_nu^2 + d_nu^3$
y
y_1 y_2 y_3 y_{i-1} y_i y_{i+1} y_n y_{n+1}
t_1 t_2 t_3 t_{i-1} t_i t_{i+1} t_n t_{n+1} t

FIGURE 12.12 Cubic splines.

and that the cubic polynomial p_i, $1 \le i \le n$ between t_i and t_{i+1} is parameterized by u

$$p_i\,(u) = a_i + b_iu + c_iu^2 + d_iu^3, \quad 0 \le u \le 1. \tag{12.13}$$

As it turns out, we will never need to deal directly with the t_i. The algorithm deals with them implicitly. The polynomials must agree at the points t_i, $2 \le i \le n$, which leads to

$$\begin{aligned} p_i\,(0) &= a_i = y_i, && (12.14)\\ p_i\,(1) &= a_i + b_i + c_i + d_i = y_{i+1}. && (12.15)\end{aligned}$$

These conditions make the piecewise polynomial continuous. Let D_i, $1 \le i \le n+1$, be the value of the first derivative of the p_i at the knots, and require that at the interior points the first derivatives must agree. This assures that the function defined by the piecewise polynomials is continuous and differentiable. Thus, for $i = 2, \ldots, n$

$$\begin{aligned} p_i'\,(0) &= b_i = D_i, && (12.16)\\ p_i'\,(1) &= b_i + 2c_i + 3d_i = D_{i+1}. && (12.17)\end{aligned}$$

Solve Equations 12.14–12.17 for a_i, b_i, c_i, and d_i to obtain

$$\begin{aligned} a_i &= y_i, && (12.18)\\ b_i &= D_i, && (12.19)\\ c_i &= 3\,(y_{i+1} - y_i) - 2D_i - D_{i+1}, && (12.20)\\ d_i &= 2\,(y_i - y_{i+1}) + D_i + D_{i+1}. && (12.21)\end{aligned}$$

Using Equations 12.18–12.21, the a_i, b_i, c_i, and d_i follow from the D_i and y_i.

Now require that the second derivatives match at the $n-1$ interior points $\{t_2, t_3, \ldots, t_n\}$, so

$$p''_{i-1}(1) = p''_i(0),$$

and

$$2c_{i-1} + 6d_{i-1} = 2c_i, \quad 2 \le i \le n. \tag{12.22}$$

Also require that $p_1(0) = y_1$ and $p_n(1) = y_{n+1}$, which gives

$$a_1 = y_1, \tag{12.23}$$
$$a_n + b_n + c_n + d_n = y_{n+1}. \tag{12.24}$$

Each cubic polynomial p_i, $1 \le i \le n$ has four unknown coefficients $\{a_i, b_i, c_i, d_i\}$, so there are a total of $4n$ unknowns. Here are the equations we have:

- Equating the polynomial values at the interior points: $2(n-1)$ equations
- Equating the first derivatives at the interior points: $(n-1)$ equations (substitute 12.14 into 12.15 to obtain only one equation)
- Equating the second derivatives at the interior points: $(n-1)$ equations
- Require that $p_1(0) = y_1$ and $p_n(1) = y_{n+1}$: 2 equations

TOTAL = $4n - 2$ *equations.*

We still require two more equations. The choice of the two additional equations determines the type of cubic spline. Require

$$\begin{aligned} p''_1(0) &= 0, \\ p''_n(1) &= 0, \end{aligned} \tag{12.25}$$

which implies

$$c_1 = 0, \tag{12.26}$$
$$2c_n + 6d_n = 0. \tag{12.27}$$

With some manipulation, we can reduce the problem to the solution of an $(n+1) \times (n+1)$ system of equations that we can solve much faster. Using Equations 12.20 and 12.22, we have

$$2c_{i-1} + 6d_{i-1} = 2\left[3(y_{i+1} - y_i) - 2D_i - D_{i+1}\right].$$

From Equations 12.20 and 12.21, it follows that

$$\begin{aligned} c_{i-1} &= 3(y_i - y_{i-1}) - 2D_{i-1} - D_i \\ d_{i-1} &= 2(y_{i-1} - y_i) + D_{i-1} + D_i, \end{aligned}$$

so

$$2\left[3(y_i - y_{i-1}) - 2D_{i-1} - D_i\right] + 6\left[2(y_{i-1} - y_i) + D_{i-1} + D_i\right] = 2\left[3(y_{i+1} - y_i) - 2D_i - D_{i+1}\right] \tag{12.28}$$

Simplify Equation 12.28 and move the unknowns to the left-hand side, and we have the $n-1$ equations in $n+1$ unknowns.

$$D_{i+1} + 4D_i + D_{i-1} = 3(y_{i+1} - y_{i-1}), \quad 2 \le i \le n. \tag{12.29}$$

We must add two more equations to Equation 12.29. From Equation 12.20, $c_1 = 3(y_2 - y_1) - 2D_1 - D_2$, and by using Equation 12.26

$$2D_1 + D_2 = 3(y_2 - y_1). \tag{12.30}$$

From Equation 12.27 $c_n = -3d_n$, and by substituting it into Equation 12.24, there results

$$a_n + b_n - 2d_n = y_{n+1}. \tag{12.31}$$

Apply Equations 12.18, 12.19, and 12.21 with $i = n$ in Equation 12.31, and perform some algebra, to obtain

$$D_n + 2D_{n+1} = 3(y_{n+1} - y_n). \tag{12.32}$$

After adding Equations 12.30 and 12.32 to Equation 12.29, we have the symmetric tridiagonal system

$$\begin{bmatrix} 2 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 2 \end{bmatrix} \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ \vdots \\ D_n \\ D_{n+1} \end{bmatrix} = \begin{bmatrix} 3\left(y_2 - y_1\right) \\ 3\left(y_3 - y_1\right) \\ 3\left(y_4 - y_2\right) \\ \vdots \\ 3\left(y_n - y_{n-2}\right) \\ 3\left(y_{n+1} - y_{n-1}\right) \\ 3\left(y_{n+1} - y_n\right) \end{bmatrix}. \tag{12.33}$$

Solve this $(n+1) \times (n+1)$ tridiagonal system using the linear $[O(n)]$ Thomas algorithm presented in Section 9.4, and apply Equations 12.18–12.21 to determine the $\{a_i, b_i, c_i, d_i\}$.

We have defined *natural cubic splines*. There are other ways to obtain the two additional equations. The *not-a-knot* condition requires that the third derivatives are continuous at t_2, and t_n. In a *clamped cubic spline*, the first derivative of the spline is specified at the end points, so that $p_1'(0) = f'(0)$ and $p_n'(1) = f'(x_{n+1})$. This requires that the derivatives are known or can be estimated at the endpoints.

Algorithm 12.1 builds natural cubic splines. It also graphs the data and the spline approximation, but it does not perform interpolation at a specified point or points. That is left to the exercises.

Algorithm 12.1 Cubic Spline Approximation

```
function CUBICSPLINEB(t,y)
   % CUBSPLINEB Natural cubic spline interpolation
   % CUBSPLINEB(t,y) computes a natural cubic spline approximation
   % to the data t, y. It then plots the data and the cubic spline.
   % Input: $n+1$ data points $(t_1 y_i)$.
   % Output: $n \times 4$ matrix. Row 1 is the cubic polynomial fit to
   % $t_1 \le t \le t_2, \ldots,$ Row $n$ is the cubic polynomial fit to
   % $t_n \le t \le t_{n+1}$. The function makes a plot of the data
   % and the cubic spline approximation.
   $b_1 = 2$
   $b_{n+1} = 2$
   % build the right-hand side of the system.
   $rhs_1 = 3\left(y_2 - y_1\right)$
   $rhs_{n+1} = 3\left(y_{n+1} - y_n\right)$

   for i = 2:n do
      $rhs_i = 3\left(y_{i+1} - y_{i-1}\right)$
   end for
   % solve the system using the linear Thomas algorithm.
   $D = thomas(a, b, c, rhs)$
   % construct the cubic polynomials in the rows of S.
   for i = 1:n do
      $S_{i1} = 2\left(y_i - y_{i+1}\right) + D_i + D_{i+1}$
      $S_{i2} = 3\left(y_{+1} - y_i\right) - 2D_i - D_{i+1}$
      $S_{i3} = D_i$
      $S_{i4} = y_i$
   end for
   % plot the data and the cubic spline approximation.
   for i = 1:n do
      $p_i$ = spline $S_i$
      plot $p_i$ over the interval $t_i \le t \le t_{i+1}$
   end for
   plot $(t, y)$, marking each point with '*'
end function
```

NLALIB: The function `cubicsplineb` implements Algorithm 12.1.

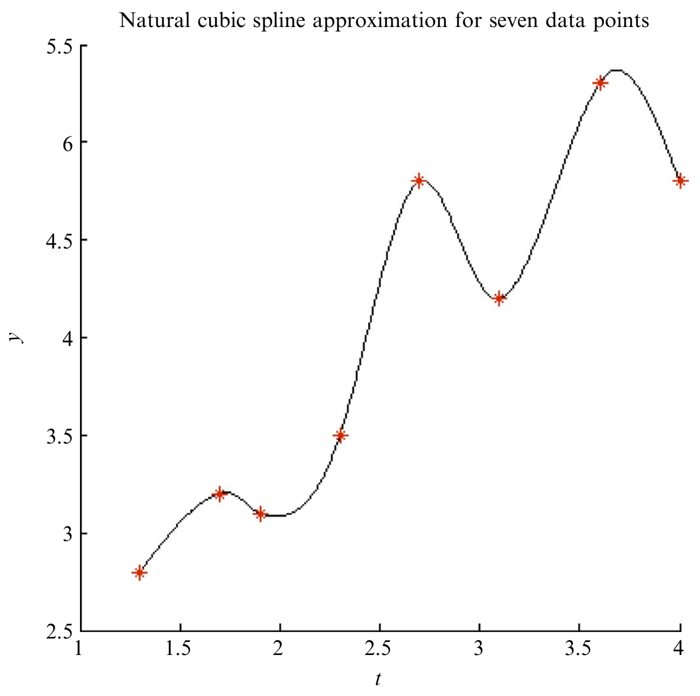


FIGURE 12.13 Cubic spline approximation.

Example 12.4. Using the data from Example 12.3, fit the data using cubic spline interpolation and graph the results (Figure 12.13). Approximate the value at *tval* = 2.5.

```
>> S = cubicsplineb(t,y);
>> p = S(4,:);
>> tval = 2.5;
>> % compute value of parameter u by interpolation;
>> u = (tval - t(4))/(t(5) - t(4));
>> polyval(p, u)

ans =

    4.2868
```
■

12.5 CHAPTER SUMMARY

Fourier Series

A Fourier series is an infinite series of trigonometric functions that, under the correct conditions, converges to a periodic function. Fourier series expose frequencies in waves and can serve as an exact solution to some partial differential equations. Computing Fourier series for square, sawtooth, and triangle waves is useful in the analysis of musical sounds.

Finite Difference Approximations

A finite difference approximation is an expression involving the function at various points that approximates an ordinary or a partial derivative. To approximate the solution to an ordinary or partial differential equation, approximate the derivatives at a series of grid points, normally close together. Using boundary and initial conditions that determine the right-hand side, create a linear system of equations. The solution provides approximations to the actual solution at the grid points. In this chapter, we use finite differences to approximate the solution to the one-dimensional heat equation with initial and boundary conditions. This requires solving a large tridiagonal system, for which we use the Thomas algorithm from Section 9.4.

Least-Squares Polynomial Fitting

Given experimental data points $(t_1, y_1), (t_2, y_2), \ldots (t_m, y_m)$, find a unique polynomial $p(t) = a_n t^n + a_{n-1} t^{n-1} + \cdots + a_2 t^2 + a_1 t + a_0$ such that

$$E = \sum_{i=1}^{m} \left(y_i - \left(a_0 + a_1 t_i + a_2 t_i^2 \cdots + a_n t_i^n \right) \right)^2$$

is a minimum. The minimum occurs when

$$\frac{\partial E}{\partial a_0} = 0, \frac{\partial E}{\partial a_1} = 0, \ldots, \frac{\partial E}{\partial a_n} = 0.$$

After some manipulation, we determine that this results in an $(n+1) \times (n+1)$ system of equations. If we define the $m \times (n+1)$ Vandermonde matrix as

$$V = \begin{bmatrix} 1 & t_1 & \ldots & t_1^n \\ 1 & t_2 & \ldots & t_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & \ldots & t_m^n \end{bmatrix},$$

then it follows that the system of equations we have to solve is

$$V^{\mathrm{T}} V \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = V^{\mathrm{T}} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

This is an example of the normal equations, the primary equations for linear least-squares problems that we will study in Chapter 16.

Cubic Spline Interpolation

A polynomial least-squares approximation can be used for interpolation or extrapolation. If high-accuracy interpolation is required, cubic splines are a superior choice. Given the experimental data points

$$\{(t_1, y_1), (t_2, y_2), (t_3, y_3), \ldots, (t_n, y_n), (t_{n+1}, y_{n+1})\},$$

fit a cubic polynomial p_i, $1 \le i \le n$ between t_i and t_{i+1} parameterized by u

$$p_i(u) = a_i + b_i u + c_i u^2 + d_i u^3, \quad 0 \le u \le 1.$$

Require that the cubics agree with the y-data at $t_1, t_2, \ldots, t_{n+1}$; in addition, require that the first and second derivatives of the splines match at the interior points

$$(t_2, y_2), (t_3, y_3), \ldots, (t_n, y_n),$$

and that the second derivative of p_1 is zero at t_1, and the second derivative of p_n is zero at t_{n+1}. After some manipulation, there results an $(n+1) \times (n+1)$ tridiagonal system of equations that must be solved. This is quickly done using the Thomas algorithm.

Cubic splines have many applications, including computer graphics, image interpolation and digital filtering, and modeling airplane drag as a function of mach number, the speed of the airplane with respect to the free stream airflow [36].

12.6 PROBLEMS

12.1 The following function S is called a *sawtooth wave* and has period 2π:

$$S(t) = \begin{cases} t & -\pi < t < \pi \\ S(t + 2\pi k) = S(t) & -\infty < t < \infty, \quad k \in \mathbb{Z} \end{cases}$$

The file sawtooth.mp3 in the book software distribution plays the sawtooth wave sound. Figure 12.14 is a graph of the function over the interval $-4\pi \le t \le 4\pi$. Find its Fourier series. Hint: The sawtooth wave is an odd function.

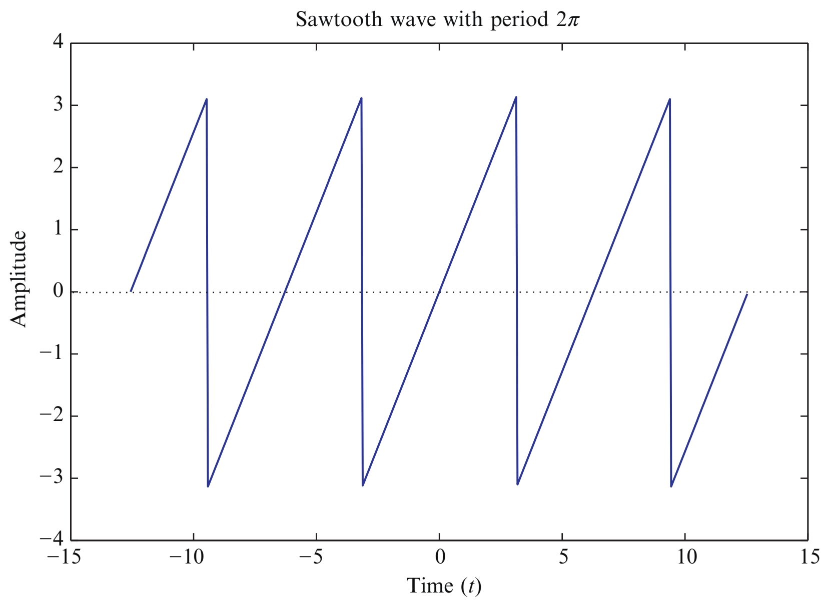


FIGURE 12.14 Sawtooth wave with period 2π.

12.2 Find the Fourier series for the triangle wave defined by

$$f(t) = \begin{cases} \frac{\pi}{2} + t, & -\pi \le t \le 0 \\ \frac{\pi}{2} - t, & 0 < t \le \pi \end{cases}.$$

12.3 Find the Fourier series for the function

$$f(t) = \begin{cases} -1, & -\pi \le t \le -\frac{\pi}{2} \\ 0, & -\frac{\pi}{2} < t \le \frac{\pi}{2} \\ 1, & \frac{\pi}{2} < t \le \pi \end{cases}.$$

12.4 Orthogonal functions play an important role in mathematics, science, and engineering, and the Fourier series is an example. Another example is the *Chebyshev polynomials* that play an important role in proving the convergence of the conjugate gradient method for solving large, sparse, linear systems, among other uses. We will discuss this method in Chapter 21. One approach to define the Chebyshev polynomials is through the use of trigonometric functions.

a. By using the trigonometric identities

$$\cos(\alpha + \beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta,$$
$$\sin(\alpha + \beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta),$$
$$\cos^2(\theta) + \sin^2(\theta) = 1,$$

show that

$$\cos((n+1)\theta) = 2\cos\theta\cos n\theta - \cos((n-1)\theta). \tag{12.34}$$

b. The Chebyshev polynomial, $T_n(x)$ of degree n is a polynomial with integer coefficients such that

$$\cos n\theta = T_n(\cos\theta);$$

in other words there is a polynomial

$$T_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

such that

$$\cos n\theta = a_n\cos^n\theta + a_{n-1}\cos^{n-1}\theta + \cdots + a_2\cos^2\theta + a_1\cos\theta + a_0. \tag{12.35}$$

i. Show that Equation 12.35 is true for $n = 0$ and $n = 1$.
ii. By applying Equation 12.34, use mathematical induction to prove Equation 12.35 for all n.
iii. Let $x = \cos\theta$ so that θ

$$T_n(x) = \cos\left(n\cos^{-1}(x)\right), \quad -1 \le x \le 1.$$

iv. Show that

$$\max_{-1\le x\le 1} T_n(x) = 1.$$

v. Prove that the n roots of $T_n(x)$ are

$$x_i = \cos\left(\frac{(2i-1)\pi}{2n}\right).$$

vi. Prove that the Chebyshev polynomials satisfy the recurrence relation

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x). \tag{12.36}$$

vii. Using Equation 12.36, find the first five Chebyshev polynomials.

c. If f and g are continuous functions over the interval $a \le x \le b$, then f and g are orthogonal with respect to the weight function w if

$$\int_a^b f(x)\,g(x)\,w(x)\,\mathrm{d}x = 0.$$

Show that the Chebyshev polynomials are orthogonal over $-1 \le x \le 1$ with respect to the weight function

$$\frac{1}{\sqrt{1-x^2}}.$$

by proceeding as follows.
i. Investigate the possible values of

$$\int_0^\pi \cos m\theta\,\cos n\theta\,\mathrm{d}\theta.$$

ii. Make a change of variable $x = \cos\theta$ and show that

$$\int_{-1}^1 T_m(x)\,T_n(x)\,\frac{\mathrm{d}x}{\sqrt{1-x^2}} = 0, \quad m \ne n$$

12.5 Finite difference methods are also used to approximate the solution to ordinary differential equations. Consider the boundary value problem for the general second-order equation with constant coefficients

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2} + p\frac{\mathrm{d}y}{\mathrm{d}x} + qy = r(x), \quad a \le x \le b,$$
$$y(a) = YA, \quad y(b) = YB.$$

Let the interval $a \le x \le b$ be divided into n subintervals of width $h = (b-a)/n$. Using the central difference approximations

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

find the linear system that must be solved to approximate $y_2, y_3, \ldots, y_n$.

12.6 Although not used as often as cubic splines, *quadratic splines* use a quadratic equation $p_i(x) = a_ix^2 + b_ix + c_i$ between knots. Assume that the data points are

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1})$$

and that the piecewise quadratic function is continuous and differentiable. Determine the set of linear equations that determine the coefficients a_i, b_i, and c_i. You will need one additional equation so that the system is square. Do this by assuming that

$$p_1''(x_1) = 0.$$

12.6.1 MATLAB Problems

12.7 Recall Figure 12.2 in which partial sums of the Fourier series for the square wave were graphed on the same set of axes, demonstrating convergence. Write a MATLAB program that graphs the sawtooth wave defined in Problem 12.1 over $-2\pi \le t \le 2\pi$, along with the partial sums for $n = 3, 7, 10, 12$.

12.8 Repeat Problem 12.7 for the triangle wave defined in Problem 12.2.

12.9 Repeat Problem 12.7 for the periodic function defined in Problem 12.3.

12.10 The *Gibbs phenomenon* is the odd way in which the Fourier series of a piecewise continuously differentiable periodic function behaves at a jump discontinuity, such as that in a square or triangle wave [37]. The nth partial sum of the Fourier series exhibits large oscillations near the jump, which might cause the partial sum's value to rise above the function value. The overshoot does not die out as the number of terms in the partial sum increases, but approaches a finite limit. Experiment with the sawtooth wave defined in Problem 12.1 and present evidence of the phenomenon.

12.11 Use `heateq.m` to approximate and graph the solution of the following heat conduction problem:

$$\begin{aligned}\frac{\partial u}{\partial t} &= 1.25\frac{\partial^2 u}{\partial x^2}, \quad 0 \le x \le 5.0, \quad 0 \le t \le 1.0,\\ u(0,t) &= u(L,t) = 0,\\ u(x,0) &= x\sin x.\end{aligned}$$

12.12

a. Write a function

```
[x,y] = ode2ndconst(a,b,n,YA,YB,p,q,r)
% The function solves a general second order linear ODE of the form:
%   d2y/dx2 + p(dy/dx) + qy = r(x)
% using the finite difference method.
% The boundary conditions are assumed to be of the form:
%   y(a) = YA and y(b) = YB
% Second order central differences are used.
%Input arguments:
% a First value of x (first point in the domain).
% b Last value of x (last point in the domain).
% n Number of subintervals.
% YA Boundary condition at x=a.
% YB Boundary condition at x=b.
% p The constant p.
% q The constant q.
% r Right-hand side function r(x).
%Output arguments:
% x A vector with the x coordinate of the solution points.
% y A vector with the y coordinate of the solution points.
```

that uses the result of Problem 12.5 to solve a boundary value problem for a general linear ordinary differential equation with constant coefficients.

b. Graph the solution to the problem

$$\begin{aligned}\frac{d^2y}{dx^2} + 8\frac{dy}{dx} + 5y &= \sin(x), \quad 0 \le x \le 2,\\ y(0) &= 1, \quad y(2) = 2.\end{aligned}$$

12.13 **a.** Modify your function in Problem 12.12 to approximate the solution of a boundary value problem for a linear ordinary differential equation of the form

$$\frac{dy^2}{dx^2} + p(x)\frac{dy}{dx} + q(x)y = r(x), \quad a \le x \le b$$
$$y(a) = YA, \quad y(b) = YB$$

Name your function `ode2nd`.

b. Plot the solution to each boundary value problem.

i. $\frac{d^2y}{dx^2}+\left(\frac{1}{1+x^2}\right)\frac{dy}{dx}+\left(x^2-1\right)y=1,\ y(1)=1,\ y(5)=2$

ii. $\frac{d^2y}{dx^2}+x\sin(x)\frac{dy}{dx}+x^2y=\coth(x),\ y(-1)=-1,\ y(6)=-2$

12.14

a. Plot the data, the linear regression line, and the quadratic least-squares curve on the same set of axes.

t	y
1.3	−12.3
1.8	−8.5
2.3	−5.6
3.6	−2.3
4.9	−0.5
5.3	1.3
6.5	1.5

b. Compute both linear and quadratic least-squares estimates for $t = 2.0$ and $t = 6.8$.

12.15 The data in the table give the approximate population of the United States for selected years from 1845 until 2000.

Year	1845	1871	1915	1954	2000
Populations (millions)	20	30	100	160	280

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 linear least squares to determine the constants b and m for which the function best fits the data. Use the equation to estimate the population in the year 1970. Hint: Take the logarithm of the function.

12.16

a. Develop a MATLAB function

```
function p = lq(t, y, n)
```

that finds the least-squares fit for the data $(t,\ y)$. The return value is the polynomial in MATLAB format; in other words, if the polynomial is

$$a_nt^n+a_{n-1}t^{n-1}+\cdots+a_2t^2+a_1t+a_0,$$

then $p=\begin{bmatrix} a_n & a_{n-1} & \dots & a_2 & a_1 & a_0\end{bmatrix}^{\mathrm{T}}$.

b.

Use your function to verify your calculations in Problem 12.14. Also, fit a cubic polynomial and compare the results with the lower-order approximations.

12.17 Find out how MATLAB performs least-squares fitting, and solve Problem 12.14.

12.18 Lagrange interpolation is an alternative to least squares. Given n points

$$\{(t_1,y_1),(t_2,y_2),\dots,(t_n,y_n)\},$$

there is a unique polynomial of degree $n-1$ that passes through all n points. One approach is to find the polynomial

$$p(t)=a_{n-1}t^{n-1}+a_{n-2}t^{n-2}+\cdots+a_2t^2+a_1t+a_0$$

by solving the system of equations

$$\begin{aligned}
a_{n-1}t_1^{n-1}+a_{n-2}t_1^{n-2}+\cdots+a_2t_1^2+a_1t_1+a_0 &= y_1\\
a_{n-1}t_2^{n-1}+a_{n-2}t_2^{n-2}+\cdots+a_2t_2^2+a_1t_2+a_0 &= y_2\\
&\vdots\\
a_{n-1}t_n^{n-1}+a_{n-2}t_n^{n-2}+\cdots+a_2t_n^2+a_1t_n+a_0 &= y_n
\end{aligned}$$

To approximate a value $\bar{y}$ at $\bar{t}$ between t_1 and t_n, compute $\bar{y}=p\left(\bar{t}\right)$.

a. Write a function `p = lagrangel(t,y)` that returns the interpolating polynomial for the data $(t,\ y)$.

b. Find the fourth-order interpolating polynomial for the points $\{(1, 2), (1.5, 2.5), (2.3, 3.5), (2.9, 4.5), (3.3, 5.0)\}$ and graph the points and the polynomial on the same set of axes.

c. Graph the points and the seventh-order polynomial that passes through
`t = (0:7)', y = [0 2 0 2 0 2 0 2]'.`

d. Graph the points and the cubic spline approximation.

e. Comment on the results.

12.19 Sometimes it is not necessary to employ linear algebra to solve a problem. It may be more efficient to approach the problem a different way. This problem presents an alternate approach to Lagrange interpolation as presented in Problem 12.18. Given n points

$$\{(t_1, y_1), (t_2, y_2), \dots, (t_n, y_n)\},$$

there is a unique polynomial $p(t)$ of degree $n-1$ that passes through all n points.

a. Show that

$$p(t) = \frac{(t-t_2)(t-t_3)\dots(t-t_n)}{(t_1-t_2)(t_1-t_3)\dots(t_1-t_n)}y_1 + \frac{(t-t_1)(t-t_3)\dots(t-t_n)}{(t_2-t_1)(t_2-t_3)\dots(t_2-t_n)}y_2 + \dots + \frac{(t-t_1)(t-t_2)\dots(t-t_{n-1})}{(t_n-t_1)(t_n-t_2)\dots(t_n-t_{n-1})}y_n.$$

b. Repeat Problem 12.18, part (b), by developing a function `p = lagrange2(t,y)` that uses the polynomial of part (a). You will find the MATLAB function `conv` useful in constructing the product of polynomial factors.

12.20 There is a famous example, known as Runge's phenomenon, showing that a Lagrange polynomial (discussed in Problems 12.18 and 12.19) may oscillate at the edges of an interval when using Lagrange interpolation with polynomials of high degree. Let

$$f(x) = \frac{1}{1+25x^2},$$

and consider interpolation at the points $(x_i, f(x_i))$, where

$$x_i = -1 + \frac{2(i-1)}{n}, \quad 1 \le i \le n+1.$$

Plot $f(x)$ and the Lagrange interpolate for $n = \{5, 10, 15, 20\}$ on the same axes. Comment on the results.

12.21 Write a function `lininterp(t,y,tval)` that performs linear interpolation to approximate the data value at tval. Test it with the data from Example 12.3.

12.22

a. Remove the plotting code from `cubsplineb` and name the function `cubspline`. It returns only the matrix of cubic polynomials.

b. Write a function `ival = cubsplineinterp(S, t, tval)`.
`S` is the matrix of cubic polynomials that defines a cubic spline.
`t` is the vector of independent variables.
`tval` is the value at which to interpolate.
`ival` is the interpolated value at `tval`.

c. Using the data from Example 12.3, approximate the values at $t = 1.5, 1.85, 2.55, 3.0, 3.8$.

12.23 Determine how MATLAB performs not-a-knot cubic spline interpolation and use MATLAB to solve Problem 12.22(c).