Find the coefficients $\{a_j\}_{j=0}^m$ and $\{b_j\}_{j=1}^m$ that minimize the square error

$$\int_0^L [f(x) - f_m(x)]^2 \, dx. \tag{11.2}$$

Strictly speaking, this problem can be formulated for the more general class of *square integrable* functions, but since this involves subtle definitions from the theory of Lesbegue integration, we focus on functions with jump discontinuities at a finite number of points, which covers most common practical applications.

Trigonometric approximation

The following solution of Problem 11.1 is typically presented in courses in *Partial Differential Equations*.

Theorem 11.1 *There exists a unique solution of Problem 11.1 with*

$$a_j = \frac{2}{L} \int_0^L f(x) \cos\left(\frac{2\pi j x}{L}\right) dx, \qquad j = 0, 1, ..., m, \tag{11.3}$$

$$b_j = \frac{2}{L} \int_0^L f(x) \sin\left(\frac{2\pi j x}{L}\right) dx, \qquad j = 1, 2, ..., m. \tag{11.4}$$

For any small $\epsilon > 0$, there exists $m \geq 1$ such that

$$\int_0^L [f(x) - f_m(x)]^2 \, dx \leq \epsilon^2. \tag{11.5}$$

A typical example of the trigonometric approximation is the representation of the sign function on the interval $[-1, 1]$

$$f(x) = \text{sign}(x) = \begin{cases} +1, & 0 < x < 1 \\ -1, & -1 < x < 0 \end{cases} \tag{11.6}$$

by the trigonometric sum of sinusoidal functions

$$f_m(x) = \sum_{j=1}^m \frac{4}{\pi(2j-1)} \sin \pi(2j-1)x, \qquad -1 \leq x \leq 1. \tag{11.7}$$

The partial sum $f_m(x)$ is an odd periodic function with the period $L = 2$, which satisfies the Dirichlet boundary conditions $f_m(0) = f_m(1) = 0$. Notice that the original function $f(x)$ has two jump discontinuities at the points

$x = 0$ and $x = 1$. Theorem 11.1 guarantees that the distance (11.5) can be reduced with larger values of m for any given small $\epsilon^2 > 0$. Indeed, the error

$$\int_{-1}^1 [f(x) - f_m(x)]^2 \, dx = 2 - \frac{16}{\pi^2} \sum_{j=1}^m \frac{1}{(2j-1)^2}$$

can be reduced below any small number ϵ^2, since the sum $\sum_{j=1}^m \frac{1}{(2j-1)^2}$ converges absolutely to $\frac{\pi^2}{8}$ as m goes to infinity.

Convergence of trigonometric approximation

If the function $f(x)$ is sufficiently smooth, the partial sum $f_m(x)$ converges *pointwise* to $f(x)$ at each point $x \in [0, L]$ as $m \to \infty$. Moreover, for sufficiently smooth $f(x)$ there exists a uniform bound on the absolute distance $|f(x) - f_m(x)|$ for all $x \in [0, L]$, which converges to zero as $m \to \infty$. In this case, we say that $f_m(x)$ converges to $f(x)$ *uniformly* on $[0, L]$. In general, the absolute distance between $f(x)$ and $f_m(x)$ can be estimated in terms of powers of m depending on the smoothness of $f(x)$.

Theorem 11.2 *Let $f(x)$ be k-times continuously differentiable on $[0, L]$ for some $k \geq 1$ and let the periodic boundary conditions $f(0) = f(L)$, $f'(0) = f'(L)$, ..., $f^{(k)}(0) = f^{(k)}(L)$ be satisfied. Then, there exists constant $C_k > 0$ such that*

$$\sup_{0 \leq x \leq L} |f(x) - f_m(x)| \leq \frac{C_k}{m^{k+1}}, \qquad m \geq 1. \tag{11.8}$$

If the function $f(x)$ has jump discontinuities either in the interior points $0 < x < L$ or at the endpoints $x = 0$ and $x = L$ (when $f(0) \neq f(L)$), the partial sum $f_m(x)$ converges pointwise to the mean value of $f(x)$ at the jump discontinuity. In this case, the convergence is nonuniform and *Gibbs oscillations* near the discontinuity points arise resulting in the nonvanishing local error as $m \to \infty$. For the sign function (11.6) with the jump discontinuity at $x = 0$, the partial sum $f_m(x)$ overshoots the value $f(x) = 1$ for small positive x for any $m \geq 1$. As $m \to \infty$, the local error shifts toward the point $x = 0$, but it does not vanish [9].

Trigonometric interpolation

Following the trigonometric approximation, the trigonometric interpolation can be introduced in two different ways. In the first method, the partial sum (11.1) is discretized on the uniform grid with constant step size to match the given set of y-values and the solution of the resulting linear system represents discretizations of integrals in the coefficients (11.3)–(11.4). In the second method, the same discretizations of the integrals (11.3)–(11.4) are recovered from analysis of eigenvalues and eigenvectors of the difference eigenvalue problems. Whereas the first method extends the linear systems of polynomial interpolation in Section 5.2, the second method is closely related to calculus of differences for numerical derivatives in Section 6.2.

Problem 11.2 (Trigonometric Interpolation) Let a discrete set of data points (x_1, y_1), (x_2, y_2), ..., (x_{n+1}, y_{n+1}) be defined on the uniform grid of equally spaced x-values

$$x_k = \frac{(k-1)L}{n}, \qquad k = 1, 2, \ldots, n+1, \tag{11.9}$$

subject to the periodic boundary condition $y_{n+1} = y_1$. When $n = 2m$, find the interpolant $F_m(x)$ in the form,

$$F_m(x) = \frac{1}{2}a_0 + \sum_{j=1}^{m-1} a_j \cos\left(\frac{2\pi jx}{L}\right) + b_j \sin\left(\frac{2\pi jx}{L}\right) + \frac{1}{2}a_m \cos\left(\frac{2\pi mx}{L}\right), \tag{11.10}$$

such that

$$F_m(x_k) = y_k, \qquad k = 1, 2, \ldots, n. \tag{11.11}$$

We have adopted several conventions in Problem 11.2. We enumerate the coefficients a_j from $j = 0$ to $j = m$ to be consistent with the trigonometric sum (11.1) in Problem 11.1. On the other hand, we enumerate the discrete grid from $k = 1$ to $k = n+1$ to keep consistent with Problem 5.1. The reason why we have defined the coefficient a_m with the factor $\frac{1}{2}$ will be clear from the solution of the system (11.11). Furthermore, we have dropped the coefficient b_m since it is not defined on the discrete grid (11.9) where $\sin(\pi(k-1)) = 0$ for any $k \in \mathbb{N}$. The set of equations (11.11) is truncated at $k = n$ since the equation for $k = n+1$ is redundant as a result of periodic boundary conditions $y_{n+1} = y_1$. Finally, we shall consider only the case of even values of n. Problem 11.2 can be extended to the case of odd values of n, but all formulas need to be rewritten with simple modifications.

The system of equations (11.10)–(11.11) is equivalent to the linear algebraic system for $(m+1)$ coefficients $\{a_j\}_{j=0}^{m}$ and $(m-1)$ coefficients $\{b_j\}_{j=1}^{m-1}$:

$$y_k = \frac{1}{2}a_0 + \sum_{j=1}^{m-1} a_j \cos\left(\frac{\pi j(k-1)}{m}\right) + b_j \sin\left(\frac{\pi j(k-1)}{m}\right) + \frac{1}{2}a_m \cos(\pi(k-1)), \tag{11.12}$$

where $k = 1, 2, \ldots, 2m$.

The MATLAB solver \ (backslash) gives a unique solution to the system (11.12), assuming that the linear system is neither singular nor ill-conditioned. The next example presents a numerical solution for the sign function (11.6), extended from the symmetric interval $[-1, 1]$ to the fundamental interval $[0, 2]$ by periodic continuation. The MATLAB script `trigonometric_sums` performs computations of the coefficients $\{a_j\}_{j=0}^{m}$ and $\{b_j\}_{j=1}^{m-1}$ of Problem 11.2. These coefficients are compared with the exact values (11.7) for the coefficients $\{b_j\}_{j=1}^{m}$ of Problem 11.1.


```
% trigonometric_sums
x = -1 : 0.25 : 1; y = sign(x);
n = length(x) - 1; m = n/2; L = 2;
y(1) = 0; y(n+1) = 0;  % continuation of data to x in [0,L]
xx = [x(m+1:n),x(1:m)]'; yy=[y(m+1:n),y(1:m)]';
A = 0.5*ones(n,1);  % building the coefficient matrix
for j = 1 : (m-1)
        A = [ A, cos(2*pi*j*xx/L), sin(2*pi*j*xx/L) ];
end
A = [ A, 0.5*cos(2*pi*m*xx/L) ];
c = A\yy;  % Fourier coefficients of interpolation
a = [c(1);c(2:2:n)]', b = [0;c(3:2:n);0]'
bb = 0; jj = 1;  % Fourier coefficients of approximation
for j = 1 : m-1
    if (jj == 1)
        bb = [bb,4/(pi*j)]; jj = 0;
    else
        bb = [bb,0]; jj = 1;
    end
end
xInt = -1: 0.002 : 1;
yInt = 0.5*a(1)*ones(1,length(xInt));
for j = 1 : (m-1)    % trigonometric interpolation
    yInt = yInt + a(1+j)*cos(2*pi*j*xInt/L) + b(1+j)*sin(2*pi*j*xInt/L);
end
yInt = yInt + 0.5*a(m+1)*cos(2*pi*m*xInt/L);
yyInt = zeros(1,length(xInt)); % trigonometric approximation
for j = 1 : (m-1)
    yyInt = yyInt + bb(1+j)*sin(2*pi*j*xInt/L);
end
plot(x,y,'.g',xInt,yInt,'b',xInt,yyInt,':r');
```

When the MATLAB script `trigonometric_sums` is executed, its output shows that all coefficients $\{a_j\}_{j=0}^{m}$ are of the order of machine precision. This is explained by the fact that the function $f(x)$ is odd in x. On the other hand, the coefficients $\{b_j\}_{j=1}^{m-1}$ of the trigonometric interpolation are different from the coefficients $\{b_j\}_{j=1}^{m}$ of the trigonometric approximation. Therefore, the functions $F_m(x)$ and $f_m(x)$ give generally different representations of the original function $f(x)$. Figure 11.1 plots functions $F_m(x)$ (solid curve) and

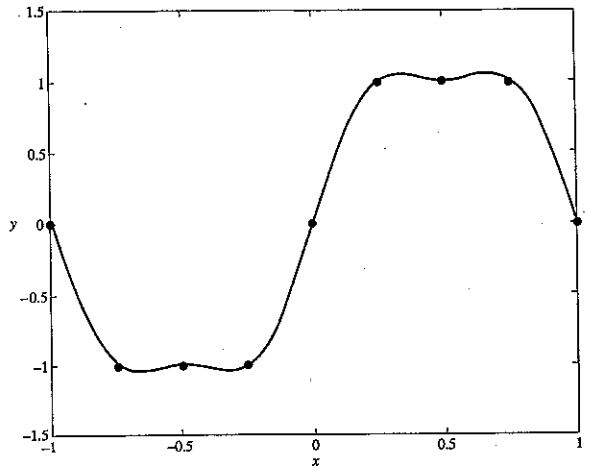

Figure 11.1 Trigonometric interpolation $F_m(x)$ (solid curve), trigonometric approximation $f_m(x)$ (dashed curve), and the original function $f(x)$ (dots).

$f_m(x)$ (dashed curve) with respect to the original function $f(x)$ (dots). The trigonometric approximation displays Gibbs oscillations at the jump discontinuities $x = 0$ and $x = \pm 1$, while trigonometric interpolation fits much closer to the given data points.

```
>> trigonometric_sums

a =
  1.0e-015 *
    0.1248   -0.0236    0.0565   -0.0883   -0.0139

b =
         0    1.2071    0.0000    0.2071         0

bb =
         0    1.2732         0    0.4244
```


Exercise 11.1 Compute the trigonometric approximation $f_m(x)$ for the sign function (11.6) with different values of m and prove numerically that the local truncation error for $f_m(x)$ does not reduce with larger values of m but shifts toward the jump point $x = 0$. Plot the maximum error for the trigonometric approximation versus m.

Exercise 11.2 Repeat Exercise 11.1 for the trigonometric interpolation $F_m(x)$.

To ensure that the linear system (11.12) is neither singular nor ill-conditioned, compute the product of the coefficient matrix A and its transposed matrix A′ in the previous example. The result is surprising: the product matrix is diagonal, implying that A can be made into an orthogonal matrix afterward by appropriate scaling of its columns.

```
>> P = A'*A

P =
    Columns 1 to 6
    2.0000    0.0000         0   -0.0000    0.0000         0
    0.0000    4.0000    0.0000    0.0000   -0.0000   -0.0000
         0    0.0000    4.0000   -0.0000         0    0.0000
   -0.0000    0.0000   -0.0000    4.0000    0.0000    0.0000
    0.0000   -0.0000         0    0.0000    4.0000   -0.0000
         0   -0.0000    0.0000    0.0000   -0.0000    4.0000
   -0.0000    0.0000   -0.0000   -0.0000    0.0000    0.0000
         0         0   -0.0000    0.0000    0.0000   -0.0000

    Columns 7 to 8
   -0.0000         0
    0.0000         0
   -0.0000   -0.0000
   -0.0000    0.0000
    0.0000    0.0000
    0.0000   -0.0000
    4.0000   -0.0000
   -0.0000    2.0000
```

The diagonal entries of `A'*A` are found to match $m = 4$, except for the first and last entries that match $\frac{m}{2} = 2$. By using these normalization factors and the orthogonality of columns of A, we obtain the exact solution of Problem 11.2.

Theorem 11.3 *There exists a unique solution $F_m(x)$ of Problem 11.2 for any $m \geq 1$:*

$$a_j = \frac{1}{m}\sum_{k=1}^{n} y_k \cos\left(\frac{\pi j(k-1)}{m}\right), \qquad j = 0, 1, \ldots, m, \tag{11.13}$$

$$b_j = \frac{1}{m}\sum_{k=1}^{n} y_k \sin\left(\frac{\pi j(k-1)}{m}\right), \qquad j = 1, \ldots, m-1. \tag{11.14}$$

Because the first and last terms in the interpolating function $F_m(x)$ in (11.10) are defined with the factor $\frac{1}{2}$, the summation formulas (11.13)–(11.14) are defined uniformly for any j. By using the vector dot products for fast vector-matrix computations of summation formulas, you can replace the MATLAB solver for the linear system (11.12) in the previous example by the summation formulas (11.13)–(11.14). The result is, of course, the same.

```
>> for j = 0 : m
      a(j+1) = yy'*cos(2*pi*j*xx/L)/m;
      b(j+1) = yy'*sin(2*pi*j*xx/L)/m;
   end
>> a

ans =
  1.0e-016 *
         0         0    0.4020         0         0

>> b

ans =
         0    1.2071    0.0000    0.2071    0.0000
```

Discrete Fourier transform

The fast Fourier transform is a popular algorithm for fast computations of the discrete Fourier transforms (11.13)–(11.14). It is implemented in most software packages and computational libraries. MATLAB operates with two basic functions: `fft` and `ifft` for the discrete Fourier transforms and their various modifications. The fast Fourier transform operates with the complex form of the trigonometric interpolation:

$$y_k = \frac{1}{n}\sum_{j=0}^{n-1} c_j e^{i\pi j(k-1)/m}, \qquad k = 1, 2, \ldots, n, \tag{11.15}$$

$$c_j = \sum_{k=1}^{n} y_k e^{-i\pi j(k-1)/m}, \qquad j = 0, 1, \ldots, n-1, \tag{11.16}$$

where for $j = 1, 2, \ldots, m-1$

$$c_j = m(a_j - ib_j), \qquad c_{n-j} = m(a_j + ib_j) = c_{-j},$$

and for $j = 0$ and $j = m$

$$c_0 = ma_0, \qquad c_m = ma_m.$$

The coefficients $\{a_j\}_{j=0}^{m}$ and $\{b_j\}_{j=1}^{m-1}$ can be found from the coefficients $\{c_j\}_{j=0}^{m}$ by

$$a_j = \frac{\mathrm{Re}(c_j)}{m}, \qquad b_j = -\frac{\mathrm{Im}(c_j)}{m},$$

where c_0 and c_m are real. We note that the relation $c_{n-j} = c_{-j}$ can be derived from the simple identity

$$e^{i\pi j(k-1)/m} = e^{i\pi(j-n)(k-1)/m}. \tag{11.17}$$

The latter modification allows us to move coefficients with negative indices $\{c_j\}_{j=-m}^{-1}$ to coefficients with positive indices $\{c_j\}_{j=m}^{n-1}$. The fast Fourier transform is based on the complex discrete Fourier transform with $n = 2^N$, where $N \in \mathbb{N}$. See [6] for details of the computational algorithm. The fast Fourier transform requires $n \log n$ computational operations versus n^2 operations of the discrete Fourier transform.

Exercise 11.5 Apply the MATLAB function `fft` to the function $f(x) = e^{-x^2}$ on $[-10, 10]$ with $n = 2^N$ and $n = 2^N - 2$ data points and compute the CPU time of each application. Show that there is no difference between the CPU time of the two operations even when the values of N becomes larger.

Finishing the previous example, we give yet another equivalent method for computing coefficients of the trigonometric interpolation:

```
>> c = fft(yy);
>> a = (real(c(1:m+1))/m)'

aF =
     0     0     0     0     0

>> b = (-imag(c(1:m+1))/m)'

bF =
         0    1.2071         0    0.2071         0
```

The formula (11.15) is referred to as the *inverse discrete Fourier transform*, whereas the formula (11.16) is referred to as the *discrete Fourier transform*. You can confirm that the two operations are inverse to each other in the sense that the successive use of (11.16) and (11.15) restores the same set of data values y_k, $k = 1, 2, \ldots, n$.

```
>> yy = ifft(c)'

yy =
        Columns 1 to 6
        0    1.0000    1.0000    1.0000         0   -1.0000
        Column 7 to 8
        -1.0000   -1.0000
```

The pair of direct and inverse transforms is used in the *pseudo-spectral method*, where the solution of a time-dependent problem is computed by means of iterations and each iteration involves the discrete Fourier transform of the unknown function, its derivatives, and its multiplications.

11.2 Errors of Trigonometric Interpolation

The summation formulas (11.13)–(11.14) can be viewed as an application of the trapezoidal integration rule to computations of the continuous Fourier integrals (11.3)–(11.4) over the equally spaced discrete grid of x-values. (The trapezoidal rule is discussed in Section 6.5.) It is surprising that the approximate trapezoidal rule recovers the exact summation formula found from the orthogonality of the coefficient matrix of the linear system (11.12). This miraculous property can be explained by the method of spectral decompositions from *Linear Algebra* (eigenvalues are covered in Section 4.1). Moreover, the same method can be extended to construction of other spectral interpolations, such as the polynomial interpolation with orthogonal (for example, Chebyshev and Legendre) polynomials [24].

Let us discuss the principal difference between the trigonometric approximation in Problem 11.1 and the trigonometric interpolation in Problem 11.2. Problem 11.1 operates on the infinite-dimensional space of continuous functions $f(x)$ defined on a finite interval $[0, L]$ with the periodic boundary conditions. On the other hand, Problem 11.2 operates on the finite-dimensional space of data points $y_1, y_2, \ldots, y_{n+1}$ defined on the grid of equally spaced values $x_1, x_2, \ldots, x_{n+1}$ with the boundary conditions $y_{n+1} = y_1$. As a result, we may view the linear system (11.12) as a unique decomposition of the vector $\mathbf{y} = [y_1, y_2, \ldots, y_n] \in \mathbb{R}^n$ over the set of basis vectors in $\mathbb{R}^n$, where the unknown values of $a_0, a_1, \ldots, a_m$ and $b_1, b_2, \ldots, b_{m-1}$ for $n = 2m$ are coordinates of the decomposition. The discrete trigonometric functions in the system (11.12) represent an orthogonal set of eigenvectors of a linear eigenvalue problem that builds a particular basis in $\mathbb{R}^n$.

To understand the linear eigenvalue problem for discrete trigonometric functions, recall the continuous theory of the Sturm–Liouville eigenvalue problem.

Eigenvalue problems

Theorem 11.2 *Consider the boundary-value problem on the finite interval,*

$$u''(x) + \lambda u(x) = 0, \qquad 0 < x < L, \tag{11.18}$$

subject to the periodic boundary conditions $u(L) = u(0)$ and $u'(L) = u'(0)$. The problem (11.18) has a complete orthogonal set of eigenfunctions $u_j(x) = e^{i2\pi jx/L}$, $j \in \mathbb{Z}$, which corresponds to the set of eigenvalues $\lambda_j = \left(\frac{2\pi j}{L}\right)^2$ and satisfies the orthogonality relations

$$\int_0^L u_j(x)\bar{u}_l(x)dx = L\delta_{j,l}, \tag{11.19}$$

where $\delta_{j,l}$ is the Kroneker symbol. Any square integrable function $f(x)$ can be represented as the complex Fourier series

$$f(x) = \sum_{j\in\mathbb{Z}} c_j u_j(x), \qquad c_j = \frac{1}{L}\int_0^L f(x)\bar{u}_j(x)dx. \tag{11.20}$$

Replacing the continuous second derivative in the differential equation (11.18) with the second-order central difference (see Section 6.2), we obtain the difference eigenvalue problem:

$$u_{k+1} - 2u_k + u_{k-1} + h^2\lambda u_k = 0, \qquad 1 \le k \le n, \tag{11.21}$$

subject to the periodic boundary conditions $u_{n+1} = u_1$ and $u_0 = u_N$. Here, h is the step size of the spatial discretization, such that $h = L/n$. The difference eigenvalue problem (11.21) has a complete set of exact solutions for eigenvectors $\mathbf{u} = [u_1, u_2, \ldots, u_n]$ and eigenvalues λ:

$$u_k = e^{i2\pi j(k-1)/n}, \qquad \lambda = \frac{4}{h^2}\sin^2\left(\frac{\pi j}{n}\right), \qquad 0 \le j \le n-1. \tag{11.22}$$

The eigenvalue for $j = 0$ is simple, while all other eigenvalues are double. The validity of the solution (11.22) can be verified from (11.21) by the direct substitution and the use of the elementary trigonometric identity:

$$e^{i\theta} - 2 + e^{-i\theta} = 2\cos\theta - 2 = -4\sin^2\frac{\theta}{2}.$$

If the difference equation (11.21) is viewed as the matrix eigenvalue problem for a three-banded coefficient matrix, the set of eigenvectors and eigenvalues can be found with the MATLAB eigenvalue solver `eig`. These computations are illustrated in the MATLAB script `eigenvalues_eigenvectors`.

```
% eigenvalues_eigenvectors
n = 101;
A=2*diag(ones(1,n))-diag(ones(1,n-1),1)-diag(ones(1,n-1),-1);
A(1,n) = -i; A(n,1) = -1;
[V,D] = eig(A); lambda = diag(D);
lambda=[lambda(1:2:n);lambda(n-1:-2:2)]; % reorganize eigenvalues
kappa = pi*(0:1:n)/n; lambdaTheory = 4*sin(kappa).^2;
plot(kappa,lambdaTheory,'r:',kappa(1:n),lambda','b.')
x = (0:1:n-1)'/(n-1); v1 = V(:,1)/max(V(:,1)); % first five eigenvectors
v2=V(:,2)/max(V(:,2)); v3 = V(:,3)/max(V(:,3));
v4=V(:,4)/max(V(:,4)); v5 = V(:,5)/max(V(:,5));
plot(x,v1,'b',x,v2,'g',x,v3,'g:',x,v4,'m',x,v5,'m:');
```

When the MATLAB script `eigenvalues_eigenvectors` is executed, it computes eigenvalues of the difference eigenvalue problem (11.21) and plots them together with the exact solution (11.22) in Figure 11.2 (*left*). Figure 11.2 (*right*) shows graphs of eigenvectors for the smallest five eigenvalues starting with the ground state $u_k = 1$ and alternating through even $u_k = \cos(2\pi jk/n)$ and odd $u_k = \sin(2\pi jk/n)$ eigenfunctions (11.22).

The three-banded coefficient matrix in the difference eigenvalue problem (11.21) is symmetric. The eigenvalues λ are all real, while the eigenvectors are

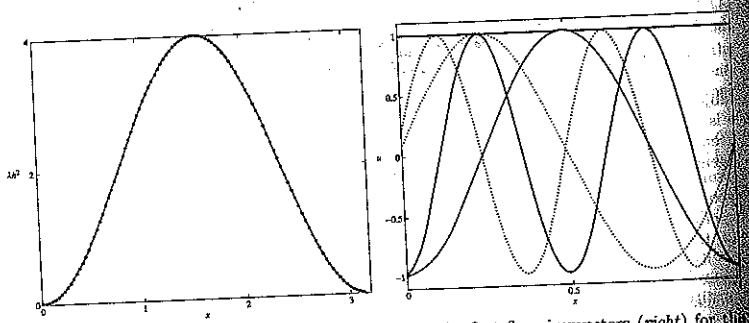

Figure 11.2 Eigenvalues (*left*) and the first five eigenvectors (*right*) for the difference eigenvalue problem (11.21).

the same as the complex exponentials of the inverse discrete Fourier transform (11.15). By adding complex numbers on the unit circle,

$$\sum_{k=1}^{n} e^{2\pi i(j-l)(k-1)/n} = n\delta_{j,l},$$

we recover the discretization of the continuous inner products (11.19). All these facts follow from general results in *Linear Algebra*. In particular, Theorem 4.6 implies that any vector $\mathbf{x} \in \mathbb{R}^n$ is uniquely represented over the orthogonal basis of eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$ of an n-by-n symmetric (self-adjoint) matrix, such that

$$\mathbf{x} = x_1\mathbf{u}_1 + x_2\mathbf{u}_2 + \ldots + x_n\mathbf{u}_n, \tag{11.23}$$

where the coordinates $x_1, x_2, \ldots, x_n$ are found from the projection formulas

$$x_j = \frac{\langle \mathbf{x}, \mathbf{u}_j \rangle}{\langle \mathbf{u}_j, \mathbf{u}_j \rangle}, \qquad 1 \le j \le n. \tag{11.24}$$

The pair of decomposition formulas (11.23)–(11.24) becomes the pair of discrete Fourier transforms (11.15)–(11.16). Whereas the discrete Fourier transform (11.16) corresponds to the second-order trapezoidal rule to the continuous Fourier integrals (11.3)–(11.4), the difference eigenvalue problem (11.21) corresponds to the second-order central-difference approximation of the differential eigenvalue problem (11.18). These facts explain the "miracle" of orthogonality of the coefficient matrix of the linear system (11.12).

Convergence of trigonometric interpolation

Suppose now that the data points (x_1, y_1), (x_2, y_2),..., (x_{n+1}, y_{n+1}) correspond to the continuous function $f(x)$ evaluated on the discrete grid (11.9). The function $F_m(x)$ is a trigonometric interpolant for the function $f(x)$ on the interval $[0, L]$. The distance between these two functions defines the *truncation error* of the trigonometric interpolation. Given that we investigate in detail the truncation error of the polynomial interpolation in Section 5.5, we can now ask if the trigonometric interpolation performs a better job for a representation of the function $f(x)$. It is proved in *Numerical Analysis* that the trigonometric interpolant has an exponentially small truncation error in terms of the number of data points provided that the function $f(x)$ is extended into an analytic function off the real interval $[0, L]$.

Theorem 11.5 *Let $f(x)$ be a periodic function on $[0, L]$, which can be extended to an analytic function in the complex strip $|\mathrm{Im}(z)| < a$ with $|u(x + iy)| \le c$ uniformly in the rectangle $[0, L] \times [-a, a]$, where $a > 0$ and $c > 0$. Then, for any sufficiently large m there exists $C > 0$ such that*

$$\max_{0 \le x \le L} |f(x) - F_m(x)| \le Ce^{-am}. \tag{11.25}$$

This result is referred to as *spectral accuracy* of spectral methods, which we now illustrate with several examples. In the first example, we consider the Runge function $f(x) = 1/(1+25x^2)$ on the interval $[-1,1]$ when the polynomial interpolation features the polynomial wiggle (see Section 5.5). The MATLAB script `Runge_trig_interpolation` computes the trigonometric interpolation of the Runge function for $n = 4, 8, 12$ subintervals.

```
% Runge_trig_interpolation
for n = 4 : 4 : 12
    x = linspace(-1,1,n+1);
    y = 1./(1+25*x.^2);
    m = n/2; L = 2;
    xx = [x(m+1:n),x(1:m)]';
    yy = [y(m+1:n),y(1:m)]';
    for j = 0 : m   % coefficients of interpolation
        a(j+1) = 2*yy'*cos(2*pi*j*xx/L)/n;
        b(j+1) = 2*yy'*sin(2*pi*j*xx/L)/n;
    end
    xInt = -1 : 0.001 : 1;
    yInt = 0.5*a(1)*ones(1,length(xInt));
    for j = 1 : (m-1) % trigonometric interpolant
        yInt = yInt + a(1+j)*cos(2*pi*j*xInt/L) + b(1+j)*sin(2*pi*j*xInt/L);
    end
    yInt = yInt + 0.5*a(m+1)*cos(2*pi*m*xInt/L);
    plot(xInt,yInt,x,y,'.');
end
yExact = 1./(1+25*xInt.^2);
plot(xInt,yExact,':r');
```

The output of the MATLAB script `Runge_trig_interpolation` is shown in Figure 11.3. No polynomial wiggle for the Runge function occurs in the trigonometric interpolation for large values of m.

The next example computes the truncation error of the trigonometric interpolation for two functions $f(x) = 1/(1+25x^2)$ and $f(x) = 1/(1+x^2)$ on the interval $[-5,5]$ versus the number m, where $n = 2m$. The MATLAB script `error_trig_interpolation` contains code for computations of the errors for the two functions.

```
% error_trig_interpolation
for m = 1 : 300
    x = linspace(-5,5,2*m+1);
    y1 = 1./(1 + 25*x.^2);
    y2 = 1./(1 + x.^2);
    n= 2*m; L = 10;
    xx = [x(m+1:n),x(1:m)]';
```


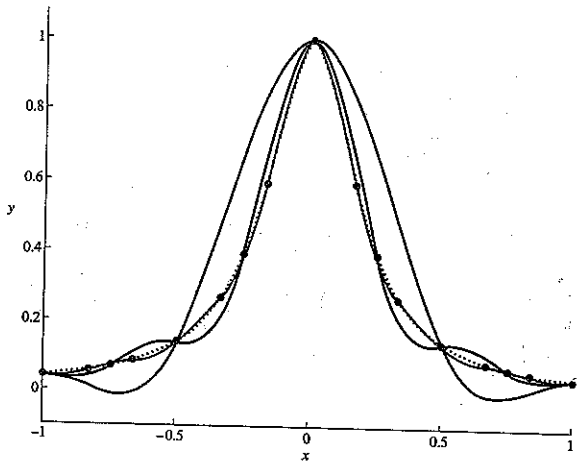

Figure 11.3 Trigonometric interpolation of the Runge function.

```
    yy1 = [y1(m+1:n),y1(1:m)]';
    yy2 = [y2(m+1:n),y2(1:m)]';
    for j = 0 : m
        a1(j+1) = 2*yy1'*cos(2*pi*j*xx/L)/n;
        b1(j+1) = 2*yy1'*sin(2*pi*j*xx/L)/n;
        a2(j+1) = 2*yy2'*cos(2*pi*j*xx/L)/n;
        b2(j+1) = 2*yy2'*sin(2*pi*j*xx/L)/n;
    end
    xInt = linspace(-5,5,1001);
    yInt1 = 0.5*a1(1)*ones(1,length(xInt));
    for j = 1 : (m-1)
        yInt1 = yInt1 + a1(1+j)*cos(2*pi*j*xInt/L);
        yInt1 = yInt1 + b1(1+j)*sin(2*pi*j*xInt/L);
    end
    yInt1 = yInt1 + 0.5*a1(m+1)*cos(2*pi*m*xInt/L);
    yInt2 = 0.5*a2(1)*ones(1,length(xInt));
    for j = 1 : (m-1)
        yInt2 = yInt2 + a2(1+j)*cos(2*pi*j*xInt/L);
```

```
        yInt2 = yInt2 + b2(1+j)*sin(2*pi*j*xInt/L);
    end
    yInt2 = yInt2 + 0.5*a2(m+1)*cos(2*pi*m*xInt/L);
    yExact1 = 1./(1 + 25*xInt.^2);
    yExact2 = 1./(1 + xInt.^2);
    Error1(m) = max(abs(yExact1-yInt1));
    Error2(m) = max(abs(yExact2-yInt2));
end
hold on; semilogy(Error1,'b.'); semilogy(Error2,'m.');
```

The output of the MATLAB script `error_trig_interpolation` is shown in Figure 11.4. For smaller values of m, the logarithm of the total error reduces linearly in m for both functions. Because the function $f(x) = 1/(1+x^2)$ has a pole at $a = 1$ and the function $f(x) = 1/(1+25x^2)$ has a pole at $a = \frac{1}{5}$, the logarithm of the total error drops faster for the first function in agreement with Theorem 11.5. For larger values of m, the logarithmic errors for both functions decay much slower than the linear decay predicted

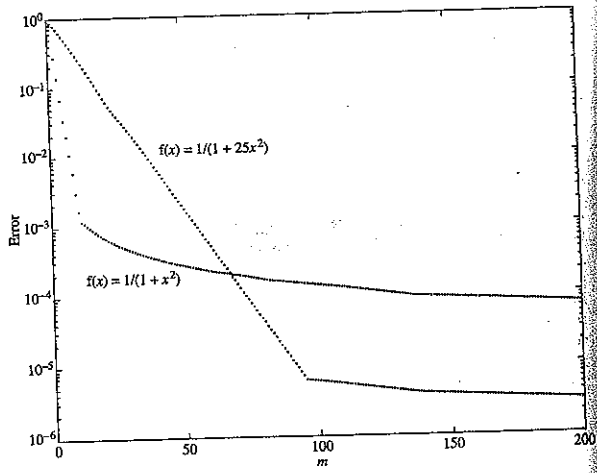

Figure 11.4 Maximum error versus the number m of the trigonometric interpolation.

by the theorem. The slow decay of the total error is caused by the influence of the round-off error, which increases with larger values of m. Compared to the slow decay of the truncation error and the growth of round-off error in the polynomial interpolation for large m (see Section 5.5), the trigonometric interpolation gives a much more robust algorithm for numerical representation of the function $f(x)$.

Similar to Theorem 11.2, the error of the trigonometric interpolation is $O(m^{-k-1})$ if the function $f(x)$ has k continuous derivatives (see convergence theorems in [28]). In this case, the efficiency of the trigonometric interpolation becomes comparable to the efficiency of the piecewise polynomial interpolation (see Section 12.1).

Exercise 11.4 Compute the maximum truncation error for the trigonometric interpolation $F_m(x)$ of the function $f(x) = \sqrt{1-x^2}$ on the interval $[-1, 1]$ versus the number m. Show that the maximum truncation error does not satisfy the bound (11.25) because $f(x)$ is not a real analytic function in the end points $x = \pm 1$. Because $f(x)$ is not differentiable at $x = \pm 1$, show that the maximum error is $O(m^{-1})$ as m increases.

Since we have assumed periodic boundary conditions at the endpoints of the interpolation interval $[0, L]$, improvements in the trigonometric interpolation are possible by reducing the step size h and simultaneously increasing the number of data points n. Unlike the case of polynomial interpolation, it is impossible to reduce the step size h and the interpolating interval $[0, L]$ simultaneously because it results in the violation of the boundary conditions on the function $f(x)$. In addition, you must be careful when no specific boundary conditions for the function $f(x)$ are implied by the interpolation problem. Interpolations based on other orthogonal functions (such as Chebyshev and Legendre interpolations) could be more appropriate for representation of functions without specific boundary conditions [24].

11.3 Trigonometric Methods for Differential Equations

In the problems where spectral accuracy of Theorem 11.5 can be reached, spectral methods give a rapidly convergent approximation. In these problems, the Galerkin method becomes accurate with very few terms in the trigonometric sum (11.1), while the collocation method can be applied with very few grid points in the uniform grid (11.9). These methods would work for various problems, including numerical solutions of boundary-value problems for differential equations. We shall develop trigonometric approximation and interpolation for the simplest boundary-value problems, leaving the general theory of spectral methods and their applications for more specialized texts [9, 24, 28].

Consider the heat equation (10.42) as the basic example of Problem 10.2. In particular, consider the boundary-value problem

$$u_t = u_{xx} + f(x), \qquad 0 < x < 1, \quad t > 0, \tag{11.26}$$

subject to the Dirichlet boundary conditions $u(0,t) = u(1,t) = 0$ and the initial condition $u(x,0) = g(x)$, where $f(x)$ and $g(x)$ are given functions on $[0,1]$. For instance, you can take a particular example of these functions as

$$f(x) = 10x(1-x), \qquad g(x) = 0.2\sin(3\pi x).$$

Using trigonometric approximation for odd functions on the symmetric interval $[-1,1]$ (see Problem 11.1 and Theorem 11.1 with $L = 2$), the function $f(x)$ is expanded into the trigonometric sum,

$$f_m(x) = \sum_{j=1}^{m} b_j \sin(\pi j x), \qquad 0 \le x \le 1, \tag{11.27}$$

where

$$b_j = 2\int_0^1 f(x)\sin(\pi j x)dx, \qquad 1 \le j \le m. \tag{11.28}$$

If $f(x) = 10x(1-x)$, the explicit expression for b_j is

$$b_j = \frac{40(1-(-1)^j)}{\pi^3 j^3}.$$

Because the function $f(x)$ is continuously differentiable on $[0,1]$, Theorem 11.2 states that the partial sum $f_m(x)$ converges to $f(x)$ pointwise and uniformly on $[0,1]$ as $m \to \infty$, while the difference between $f_m(x)$ and $f(x)$ is $O(m^{-2})$. The same order follows also from the explicit expression for b_j since the sum $\sum_{j=m+1}^{\infty} \frac{1}{j^3}$ is $O(m^{-2})$. Because the truncation error does not decay exponentially, the trigonometric approximation is not spectrally accurate. This inaccuracy is explained by the fact that if the function $f(x)$ is extended as an odd function from $[0,1]$ to $[-1,1]$, and then it is continued periodically from $[-1,1]$ to $\mathbb{R}$ with period $L = 2$, then the second derivative $f''(x)$ has jump discontinuities at the points $x = 0$ and $x = 1$. As a result, the function $f(x)$ cannot be analytically continued in the complex plane.

Galerkin method

The Galerkin method is based on the approximation of the solution $u(x,t)$ of the time evolution problem (11.26) by the trigonometric sum:

$$u(x,t) = \sum_{j=1}^{m} u_j(t)\sin(\pi j x), \qquad 0 \le x \le 1, \tag{11.29}$$

which satisfies the boundary conditions $u(0,t) = u(1,t) = 0$. The initial values for $u_j(0)$ follow from the initial condition $u(x,0) = g(x)$ expanded into the same trigonometric sum with

$$u_j(0) = 2\int_0^1 g(x)\sin(\pi j x)dx, \qquad 1 \le j \le m. \tag{11.30}$$

When $g(x) = 0.2\sin(3\pi x)$, the explicit expression for $u_j(0)$ is $u_j(0) = 0.2\delta_{3,j}$, where $\delta_{i,j}$ is the Kronecker symbol. By substituting (11.27) and (11.29) into the heat equation (11.26), we find the uncoupled systems of ordinary differential equations,

$$\frac{du_j}{dt} = -\pi^2 j^2 u_j(t) + b_j, \qquad 1 \le j \le m. \tag{11.31}$$

The exact solution of the ODE system (11.31) is available in analytic form:

$$u_j(t) = u_j(0)e^{-\pi^2 j^2 t} + \frac{b_j}{\pi^2 j^2}\left(1 - e^{-\pi^2 j^2 t}\right), \qquad 1 \le j \le m, \tag{11.32}$$

where $u_j(0)$ are given. Alternatively, initial-value ODE solvers can be applied to the numerical solution of the ODE system (11.31). In either case, the trigonometric sum (11.29) becomes the numerical approximation of the solution surface $u(x,t)$. The MATLAB script `Galerkin_method` shows details of the Galerkin method supplemented by the fourth-order Runge–Kutta method for solutions of the ODE system (11.31) (see Section 9.2).

```
% Galerkin_method
m = 5; j = 1 : m;
b = 40*(1 - (-1).^j)./(pi^3*j.^3);  % inhomogeneous term
u = zeros(1,m); u(3) = 0.2;         % initial condition
T = 0.5; t = linspace(0,T,101); dt = t(2)-t(1); % time grid
uSpectrum(1,:) = u;
for k = 1 : length(t)-1          % fourth-order Runge-Kutta method
    k1 = -pi^2*j.^2.*u+b; u1 = u + 0.5*dt*k1;
    k2 = -pi^2*j.^2.*u1+b; u2 = u + 0.5*dt*k2;
    k3 = -pi^2*j.^2.*u2+b; u3 = u + dt*k3;
    k4 = -pi^2*j.^2.*u3+b;
    u = u + dt*(k1+2*k2+2*k3+k4)/6;
    uSpectrum(k+1,:) = u;
end
x = linspace(0,1,101);
U = zeros(length(t),length(x));
```

```
for k = 1 : length(t)              % solution surface
   for jj = 1 : m
      U(k,:) = U(k,:) + uSpectrum(k,jj)*sin(pi*jj*x);
   end
end
[X,T] = meshgrid(x,t); mesh(X,T,U);
```

When the MATLAB script `Galerkin_method` is executed, it computes the numerical approximation of the solution surface $u(x,t)$ by the trigonometric sum (11.29) with $m = 5$ for $0 \le t \le 0.5$. Figure 11.5 shows the solution surface $u(x,t)$, which displays a transformation of the initial condition $u(x,0) = g(x)$ to the time-independent solution of the boundary-value problem (11.26):

$$u_\infty(x) = \lim_{t\to\infty} u(x,t) = \frac{5}{6}x(x^3 - 2x^2 + 1). \tag{11.33}$$

Except for the initial time $t = 0$, there exists an error between the exact solution $u(x,t)$ of the heat equation (11.26) and the trigonometric sum (11.29). The numerical error is caused by two main sources. The first source of the numerical error is a discretization of the numerical solution of the ODE system

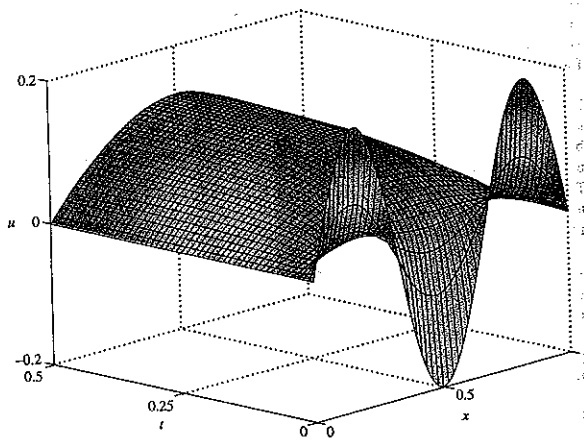

Figure 11.5 Solution of the heat equation (11.26) by the Galerkin method.

(11.31). Although the exact solution (11.32) is available in this particular case, the numerical ODE solver is often the only available tool for computations of solutions of the ODE systems. Unfortunately, the ODE system (11.31) becomes stiff for large values of m because the decay rate $\lambda_j = \pi^2 j^2$ becomes larger with larger values of j. As a result, all explicit ODE solvers develop instabilities for larger values of j, while implicit ODE solvers may produce a stable numerical solution of the ODE systems (see Section 9.5). Even if the ODE solver is implicit and stable, it leads to a numerical error that depends on the time step τ and converges to zero as a power function of τ. For instance, the global truncation error of the fourth-order Runge–Kutta method is $O(\tau^4)$ (see Section 9.2).

The other source of the numerical error is a truncation of the trigonometric sum (11.29). This source is controlled by Theorem 11.2 on the trigonometric approximation. In the particular problem (11.26) with $f(x) = 10x(1-x)$, the limiting stationary solution $u_\infty(x)$ is the polynomial of degree four (11.33). If $u_\infty(x)$ is extended to the entire axis as an odd periodic function with period $L = 2$, it has jump discontinuities in the fourth derivative, implying that the corresponding trigonometric sum (11.29) has an $O(m^{-4})$ error as $m \to \infty$. Since $u(x,0) = g(x) = \sin(3\pi x)$ is represented by the trigonometric sum exactly, the truncation error of the representation for $u(x,t)$ coincides with that for $u_\infty(x)$ for any fixed value of $t > 0$. If $f(x)$ were represented by a trigonometric sum (11.27) with spectral accuracy, then the solution $u(x,t)$ would be represented by the sum (11.29) with spectral accuracy, too.

Exercise 11.5 Use the exact solution (11.32) for the ODE system (11.31) and show that the error between the exact solution $u(x,t)$ in the Fourier series form and the trigonometric sum (11.29) for a fixed value of $t > 0$ is $O(m^{-4})$ for $f(x) = 10x(1-x)$ and is exponentially small in m for $f(x) = \sin(\pi x)$.

Collocation method

By using trigonometric interpolation, we develop the collocation method, which is based on the numerical approximation of the solution $u(x,t)$ of the PDE (11.26) at the uniform grid

$$x_k = \frac{(k-1)}{(m+1)}, \qquad 1 \le k \le m+2. \tag{11.34}$$

Compared to the formalism in Problem 11.2, the trigonometric sum (11.29) extends the sum (11.10) by increasing the index m by one. In addition, the cosine terms in the sum (11.10) are identically zero as the function $u(x,t)$ is extended into a periodic odd function on the x-axis with the period $L = 2$. The initial values for $u_j(0)$ in the trigonometric sum (11.29) follow from the trigonometric interpolation of the initial condition $u(x,0) = g(x)$ by

$$g(x_k) = \sum_{j=1}^{m} u_j(0)\sin(\pi j x_k), \qquad 2 \le k \le m+1,$$

with the inversion formula

$$u_j(0) = \frac{2}{m+1} \sum_{k=2}^{m+1} g(x_k) \sin(\pi j x_k), \qquad 1 \le j \le m. \tag{11.35}$$

Compared to the sum (11.10), the summation over $m+3 \le k \le n-1$ is not performed, since the function $g(x)$ is extended into a periodic odd function on values of x_k beyond the range $1 \le k \le m+2$. Similarly, the source term in the heat equation (11.26) is approximated at the grid points (11.34) by

$$f(x_k) = \sum_{j=1}^{m} b_j \sin(\pi j x_k), \qquad 2 \le k \le m+1,$$

with the inversion formula

$$b_j = \frac{2}{m+1} \sum_{k=2}^{m+1} f(x_k) \sin(\pi j x_k), \qquad 1 \le j \le m. \tag{11.36}$$

When the trigonometric sums for $u(x,t)$ and $f(x)$ are substituted into the heat equation (11.26), the same system of uncoupled ordinary differential equations (11.31) arises. The ODE system can be solved with an initial-value ODE solver such as the fourth-order Runge–Kutta method. The MATLAB script `collocation_method` shows details of the collocation method for the functions $f(x)$ and $g(x)$, while the ODE solver is coded similarly to the MATLAB script `Galerkin_method`.

```
% collocation_method
m = 5; x = linspace(0,1,m+2);
f = 10*x.*(1-x); % source term
for j = 1 : m
    b(j) = 2*f*sin(pi*j*x')/(m+1);
end
g = 0.2*sin(3*pi*x); % initial condition
for j = 1 : m
    u(j) = 2*g*sin(pi*j*x')/(m+1);
end
j = 1:m; T = 0.5;
t = linspace(0,T,101); dt = t(2)-t(1);
uSpectrum(1,:) = u;
for k = 1 : length(t)-1  % time iterations
     k1 = -pi^2*j.^2.*u+b; u1 = u + 0.5*dt*k1;
     k2 = -pi^2*j.^2.*u1+b; u2 = u + 0.5*dt*k2;
     k3 = -pi^2*j.^2.*u2+b; u3 = u + dt*k3;
     k4 = -pi^2*j.^2.*u3+b;
     u = u + dt*(k1+2*k2+2*k3+k4)/6;
     uSpectrum(k+1,:) = u;
end
x = linspace(0,1,101);
U = zeros(length(t),length(x));
for k = 1 : length(t)   % solution surface
   for jj = 1 : m
       U(k,:) = U(k,:) + uSpectrum(k,jj)*sin(pi*jj*x);
   end
end
[X,T] = meshgrid(x,t); mesh(X,T,U);
```


When the MATLAB script `collocation_method` is executed, it computes the numerical approximation of the solution surface $u(x,t)$, which is shown in Figure 11.6. Both the transient process and the limiting solution look similar to ones modelled by the Galerkin method (see Figure 11.5).

Let us now compare the truncation errors of the Galerkin and collocation methods for numerical approximation of the limiting solutions $u_\infty(x)$ of the

Errors of spectral methods

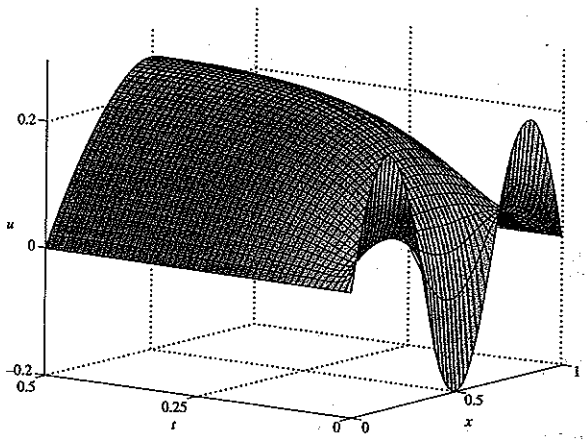

Figure 11.6 Solution of the heat equation (11.26) by the collocation method.

heat equation (11.26). The limiting solution $u_\infty(x)$ solves the boundary-value ODE problem

$$u_{xx} + f(x) = 0, \qquad 0 < x < 1, \tag{11.37}$$

subject to the Dirichlet boundary conditions $u(0) = u(1) = 0$. In both methods, the solution $u(x)$ is approximated by the trigonometric sum (11.29) with the time-independent coefficients $u_j = b_j/(\pi^2 j^2)$, $j = 1, \ldots, m$. The only difference occurs in the computations of the values of b_j. These values are computed from continuous integrals (11.28) in the Galerkin method, while they are computed from the discrete sum (11.36) in the collocation method. Numerical solutions of the ODE problem (11.37) with $f(x) = 10x(1-x)$ are computed in the MATLAB script `errors_trig_methods` by both methods and the truncation errors are found from the exact solution (11.33).

```
% errors_trig_methods
M = 100; x = linspace(0,1,1001);
for m = 1 : 2 : M
    j = 1 : m; b1 = 40*(1 - (-1).^j)./(pi^3*j.^3);
    xx = linspace(0,1,m+2); f = 10*xx.*(1-xx);
    for jj = 1 : m
        b2(jj) = 2*f*sin(pi*jj*xx')/(m+1);
    end
    u1 = b1./(pi^2*j.^2); u2 = b2./(pi^2*j.^2);
    U1 = zeros(size(x)); U2 = zeros(size(x));
    for jj = 1 : m
        U1 = U1 + u1(jj)*sin(pi*jj*x);
        U2 = U2 + u2(jj)*sin(pi*jj*x);
    end
    Uexact = 5*x.*(x.^3-2*x.^2+1)/6;
    Error1((m+1)/2) = sqrt(sum((U1-Uexact).^2));
    Error2((m+1)/2) = sqrt(sum((U2-Uexact).^2));
end
m = 1 : 2 : M; % power fit for the error dependence
a1 = polyfit(log(m),log(Error1),1); power1 = a1(1)
ErrorApr1=exp(a1(2))*exp(power1*log(m));
a2=polyfit(log(m),log(Error2),1); power2 = a2(1)
ErrorApr2=exp(a2(2))*exp(power2*log(m));
plot(m,log(Error1),'b.',m,log(ErrorApr1),':g'); hold on;
plot(m,log(Error2),'r.',m,log(ErrorApr2),':y')
```

When the MATLAB script `errors_trig_methods` is executed, the errors of the Galerkin and collocation methods are computed and shown as dots in Figure 11.7. The best power fits for the two data sets are also computed and plotted by the dotted curves. You can see from the power fits that the error

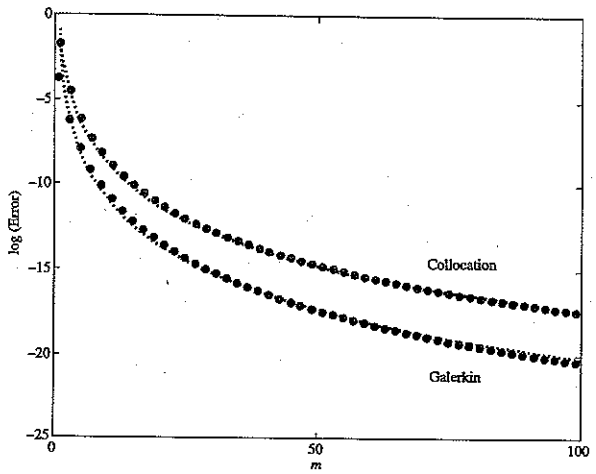

Figure 11.7 Truncation errors for the numerical solutions of the ODE problem (11.37) with the Galerkin and collocation methods.

for either numerical solution is $O(m^{-4})$ because the solution $u_\infty(x)$ in (11.32) has the jump discontinuities in the fourth derivative. You can also see from Figure 11.7 that the Galerkin method has a smaller truncation error compared to the collocation method with the same rate of convergence.

```
>> errors_trigonometric_methods

power1 =
    -3.9924

power2 =
    -3.6631
```

The heat equation (11.26) is one of the simplest problems for applications of spectral methods. Other linear and nonlinear differential equations offer

more challenges for the Galerkin and collocation methods. We review these challenges when attempting to solve the boundary-value ODE problem

$$-u'' + f(x)u = g(x), \qquad 0 < x < 2\pi, \tag{11.38}$$

where $f(x+2\pi) = f(x)$ and $g(x+2\pi) = g(x)$ subject to the periodic boundary conditions $u(2\pi) = u(0)$ and $u'(2\pi) = u'(0)$. In particular, we consider $f(x) = \cos x$ and $g(x) = \sin x$. (Eigenvalues of the linear eigenvalue problem $-u'' + \cos x u = \lambda u$ are approximated with the finite-difference method in Section 10.1, where the Neumann boundary conditions $u'(0) = u'(2\pi) = 0$ are used.) If no periodic solution of the homogeneous ODE $-u'' + f(x)u = 0$ exists, the inhomogeneous boundary-value problem (11.38) admits a unique periodic solution $u(x)$, which can be approximated numerically by using trigonometric sums. Both trigonometric approximation and interpolation lead to a linear system with a full coefficient matrix, compared to the diagonal system that follows from the ODE (11.37).

Advanced collocation method

When the trigonometric interpolation is used in the numerical solution of the ODE problem (11.38), the interval $[0, 2\pi]$ is represented by the uniform grid

$$x_k = \frac{2\pi(k-1)}{n}, \qquad k = 1, 2, \ldots, n+1, \tag{11.39}$$

where n is even. The periodic function $u(x)$ is then represented by the complex trigonometric sum

$$u(x) = \frac{1}{n} \sum_{j=-m+1}^{m-1} c_j e^{ijx} + \frac{1}{2n} \left(c_{-m} e^{-imx} + c_m e^{imx} \right), \tag{11.40}$$

where the set of coefficients $\{c_j\}_{j=-m}^{m}$ is computed from the set of function values $\{u_k\}_{k=1}^{n}$ with $u_k = u(x_k)$ by the inversion formula

$$c_j = \sum_{k=1}^{n} u_k e^{-ijx_k}, \qquad -m \le j \le m. \tag{11.41}$$

The complex trigonometric sum (11.40) is obtained from the trigonometric sum (11.10) in Problem 11.2. The coefficients $\{c_j\}_{j=-m}^{m}$ are defined by the same formulas as those given later in (11.15)–(11.16) but no reflection to positive indices $c_{n-j} = c_{-j}$ is used. Proceeding with the collocation method for a numerical solution of the ODE problem (11.38), we define the functions $f(x)$ and $g(x)$ at the collocation points (11.39) and meet the obstacle that the linear problem for the set of coefficients $\{u_k\}_{k=1}^{n}$ is not closed because the derivative terms $u''(x_k)$ are not defined in terms of the values $\{u_k\}_{k=1}^{n}$. A solution to this obstacle is constructed in [28]. According to this solution, we define the interpolation function $S(x)$ that passes through points of the discrete delta function $S(x_k) = \delta_{k,1}$. Using (11.41), we obtain that $c_j = 1$ for all $-m \le j \le m$, leading to

$$\begin{aligned} S(x) &= \frac{1}{n} \left(1 + e^{ix} + e^{-ix} + \ldots + e^{i(m-1)x} + e^{-i(m-1)x} + \cos(mx) \right) \\ &= \frac{1}{n} \left(\frac{1 - e^{imx}}{1 - e^{ix}} + \frac{1 - e^{-imx}}{1 - e^{-ix}} + \cos(mx) - 1 \right) \\ &= \frac{\sin(mx)\sin x}{n(1 - \cos x)} = \frac{\sin(mx)\cos(x/2)}{2m\sin(x/2)}. \end{aligned}$$

Using the function $S(x)$, the trigonometric interpolation is written in the form

$$u(x) = \sum_{k=1}^{n} u_k S(x - x_k), \tag{11.42}$$

so that the first and second derivatives of $u(x)$ at the grid point $x = x_k$ are expressed in the matrix form

$$u'(x_k) = \sum_{l=1}^{n} (\mathbf{D}_1)_{k,l} u_l, \qquad u''(x_k) = \sum_{l=1}^{n} (\mathbf{D}_2)_{k,l} u_l,$$

where $\mathbf{D}_1$ and $\mathbf{D}_2$ are symmetric matrices obtained from the first and second derivatives of $S(x)$ in the form [28]:

$$(\mathbf{D}_1)_{i,j} = \begin{cases} 0, & i = j, \\ \frac{1}{2}(-1)^{i-j} \cot(\pi(i-j)/n), & i \ne j \end{cases}$$

and

$$(\mathbf{D}_2)_{i,j} = \begin{cases} -\frac{n^2}{12} - \frac{1}{6}, & i = j, \\ -\frac{(-1)^{i-j}}{2\sin^2(\pi(i-j)/n)}, & i \ne j \end{cases}$$

By eliminating $u''(x_k)$ from the ODE (11.38) at the grid point $x = x_k$, we can close the system of linear equations for the set $\{u_k\}_{k=1}^{n}$, solve it with MATLAB linear algebra, and display the solution $u(x)$.

Advanced Galerkin method

When the trigonometric approximation is used in the numerical solution of the ODE problem (11.38), the periodic function $u(x)$ is represented by the complex trigonometric sum

$$u(x) = \sum_{j=-m}^{m} c_j e^{ijx}, \tag{11.43}$$

where the set of coefficients $\{c_j\}_{j=-m}^{m}$ is computed from the continuous function $u(x)$ by the inversion formula

$$c_j = \frac{1}{2\pi} \int_0^{2\pi} u(x) e^{-ijx}, \qquad -m \le j \le m. \tag{11.44}$$

The functions $f(x)$ and $g(x)$ can be represented exactly by

$$\cos x = \frac{e^{ix} + e^{-ix}}{2}, \qquad \sin x = \frac{e^{ix} - e^{-ix}}{2i}.$$

Proceeding with the Galerkin method for a numerical solution of the ODE problem (11.38), we substitute the trigonometric sum (11.43) into the ODE (11.38) and meet the obstacle that the set of coefficients $\{c_j\}_{j=-m}^{m}$ is not closed because the product term $f(x)u(x)$ generates Fourier terms $e^{i(m+1)x}$ and $e^{-i(m+1)x}$ beyond the truncation order of the sum (11.43). If the explicit form for $f(x)$ and $g(x)$ is used, the linear system for the set $\{c_j\}_{j=-m}^{m}$ takes the explicit form of the difference equation

$$j^2 c_j + \frac{1}{2}(c_{j+1} + c_{j-1}) = \frac{1}{2i}(\delta_{j,1} - \delta_{j,-1}), \qquad -m \le j \le m. \tag{11.45}$$

Assuming that the coefficients c_j become smaller with larger values of j, we can truncate the linear system (11.45) beyond the terms of $-m \le j \le m$. Because $c_j \neq 0$ for $|j| \ge m+1$, the truncation of the linear system (11.45) introduces an additional truncation error to the numerical approximation of $u(x)$ by the trigonometric sum (11.43). Because of the truncation procedure, the linear system (11.45) becomes closed and can be solved with MATLAB linear algebra.

The corresponding computations of the collocation and Galerkin methods are coded in the MATLAB script bvp_trig_methods. Two numerical approximations of the solution $u(x)$ obtained by the collocation and Galerkin methods are shown graphically in Figure 11.8.

```
% bvp_trig_methods
n = 100; x = linspace(0,2*pi,n+1);
f = diag(cos(x(1:n))); % collocation method
g = sin(x(1:n))';
D = -diag(ones(1,n))*(n^2/12+1/6); % matrix for second derivatives
for j = 1 : n-1
    for k = j+1 : n
        D(j,k) = 0.5*(-1)^(j-k-1)/(sin(pi*(j-k)/n))^2;
        D(k,j) = D(j,k);
    end
end
A = -D + f; u = A\g; u(n+1) = u(1);
m = n/2; j = -m : m;    % Galerkin method
A = diag(j.^2) + 0.5*(diag(ones(n,1),1) + diag(ones(n,1),-1));
gg = zeros(n+1,1); % inhomogeneous term
gg(m) = -1/(2*i); gg(m+2) = 1/(2*i);
c = A\gg; % solution in Fourier space
xx = linspace(0,2*pi,1001); uu = zeros(1,1001);
```


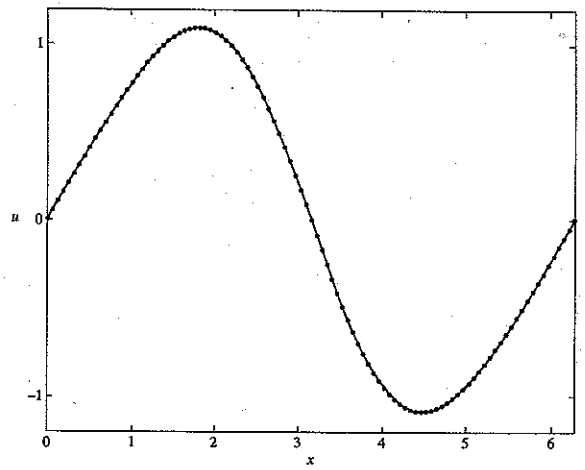

Figure 11.8 Numerical approximations of the solution $u(x)$ of the boundary-value problem (11.38) by the collocation method (dots) and by the Galerkin method (solid).

```
for k = 1 : n+1   % solution in physical space
    uu = uu + c(k)*exp(i*j(k)*xx);
end
plot(x,u,'.b',xx,real(uu),'r');
```

Exercise 11.8 Show numerically that the co-norm of the difference between the numerical approximations of the collocation and Galerkin methods reduces with larger values of m. Plot the co-norm of the computational error versus $n = 2m$ and fit the dependence with a power law.

Pseudospectral method

The boundary-value ODE problem (11.38) leads to the full coefficient matrix in either the collocation or Galerkin method. Another popular spectral method called *the pseudospectral method* overcomes this obstacle and produces a diagonal linear system for the price of an iterative method. The pseudospectral method iterates numerical approximations of the boundary-value problem

(11.38) similar to how elliptic boundary-value PDE problems are iterated by embedding the elliptic problem in the parabolic problem (see Section 10.5). Consider the boundary-value PDE problem

$$u_t = u_{xx} - f(x)u + g(x), \qquad 0 < x < 2\pi, \quad t > 0, \tag{11.46}$$

subject to the periodic boundary conditions $u(0,t) = u(2\pi,t)$ and $u_x(0,t) = u_x(2\pi,t)$ and the initial condition $u(x,0) = u_0(x)$. The stationary (time-independent) solutions of the time-evolution PDE problem (11.46) coincide with the solutions of the boundary-value ODE problem (11.38). After the time discretization is performed with the explicit Euler method, we obtain the iterative rule that defines a sequence of functions $\{u_k(x)\}_{k=0}^{\infty}$:

$$u_{k+1}(x) = u_k(x) + \tau\left(u_k''(x) - f(x)u_k(x) + g(x)\right). \tag{11.47}$$

When time iterations are performed, the given functions $u_k(x)$, $f(x)$, and $g(x)$ can be computed on the grid points (11.39) and all terms of the iterative scheme (11.47) can be diagonalized using the trigonometric sums (11.40) with the inversion formula (11.41). Let $c_j^{(k)}$ denote Fourier coefficients of the approximation $u_k(x)$, $b_j^{(k)}$ denote Fourier coefficients of the product term $f(x)u_k(x)$, and a_j denote the Fourier coefficients of the function $g(x)$. The iterative scheme (11.47) becomes diagonal in terms of the coefficients $c_j^{(k)}$:

$$c_j^{(k+1)} = c_j^{(k)}(1 - \tau j^2) + \tau(a_j - b_j^{(k)}), \qquad -m \le j \le m. \tag{11.48}$$

Since the problem (11.46) is linear, the iterative procedure (11.48) converges to a solution from any starting approximation $u_0(x)$, if it converges at all. Therefore, we can start the iterative procedure with $u_0(x) = 0$. As in all other iterative methods, iterations can be stopped when the distance between two successive iterations becomes smaller than a given tolerance. The MATLAB script `pseudospectral_method` performs computations of the pseudospectral method for the time-evolution PDE problem (11.46).

```
% pseudospectral_method
n = 50; m = n/2; x = linspace(0,2*pi,n+1);
f = cos(x(1:n)); g = sin(x(1:n)); % inhomogeneous terms
j = -m : m;  % discrete Fourier transform for g(x)
for jj = 1 : length(j)
    a(jj) = g*exp(-i*j(jj)*x(1:n)');
end
c = zeros(size(a)); % initial approximation
for k = 1 : n
    u(k) = real(c(2:n)*exp(i*(-m+1:m-1)'*x(k)));
    u(k) = (u(k)+(c(1)*exp(-i*m*x(k))+c(n+1)*exp(i*m*x(k)))/2)/n;
end
```



```
tau = 0.001; du = 1; toler = 10^(-9);
count = 0; term = 100000; % pseudospectral method
while ((du > toler) & (count < term))
      ff = f.*u;  % discrete transform for f(x) u_k(x)
      for jj = 1 : length(j)
          b(jj) = ff*exp(-i*j(jj)*x(1:n)');
      end
      cc = c + tau*(-j.^2.*c-b+a);
      for k = 1 : n % inverse transform for u_{k+1}(x)
          uu(k) = real(cc(2:n)*exp(i*(-m+1:m-1)'*x(k)));
          uu(k) = (uu(k)+(cc(1)*exp(-i*m*x(k))+cc(n+1)*exp(i*m*x(k)))/2)/n;
      end
      du = max(abs(uu-u)); c = cc;
      u = uu; count = count + 1;
end
fprintf('The algorithm converges after %d iterations\n',count);
u(n+1) = u(1); plot(x,u,'.');
```

When the MATLAB script `pseudospectral_method` is executed, it computes the sequence of numerical approximations $\{u_k(x)\}_{k=0}^{k_{\text{term}}}$ that starts with $u_0(x) = 0$ and terminates at $k = k_{\text{term}}$ when the distance between two successive iterations becomes smaller than the tolerance 10^{-9}. The number of iterations is displayed. The solution $u(x)$ at the grid points (11.39) is shown graphically in Figure 11.9. It looks similar to the solution obtained by the collocation and Galerkin methods in Figure 11.8.

```
>> pseudospectral_method

The algorithm converges after 15058 iterations
```

The use of explicit single-step methods such as the Euler method in pseudospectral methods can limit their applicability as a result of instabilities of explicit ODE solvers. When the iterations (11.48) are considered with $f(x) = g(x) = 0$, it is clear that the Euler method is stable only if

$$1 - j^2\tau > -1, \qquad -m \le j \le m,$$

such that $\tau < 2/m^2$. When the number of terms m in the trigonometric sum (11.40) grows, the time step τ becomes smaller and the number of iterations it takes to reach the same level of tolerance grows. This property results in slow convergence and large round-off error of the pseudospectral method.

There are several ways to improve the convergence and stability of the pseudospectral methods. For stability, implicit methods such as the implicit Euler method can be used for a numerical approximation of the solution $u(x)$.

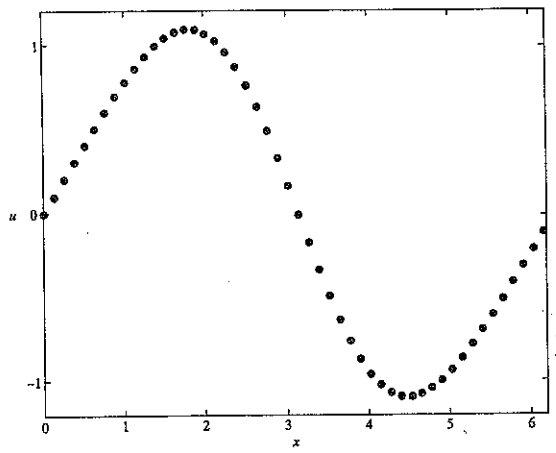

Figure 11.9 Numerical approximation of the solution $u(x)$ of the boundary-value problem (11.38) by the pseudospectral method.

Then, the number m can be arbitrarily large with no effects on stability of iterations. For convergence, we can avoid diagonalizing each term in the iterative scheme (11.47), which requires computations of the direct and inverse discrete Fourier transforms at each iteration of the time-evolution scheme. When the second derivatives of the solution $u_k(x)$ are approximated at the grid points (11.39) with the matrix $\mathbf{D}_2$ obtained from the representation (11.42) (as in the collocation method), the iterative scheme can be closed for the vector of the approximation $u_k(x)$ evaluated at the grid points (11.39). When the convolution sum is truncated beyond the terms $|j| \geq m+1$ (as in the Galerkin method), the iterative scheme can be closed for the vector of the Fourier coefficients $c_j^{(k)}$ computed for the approximation $u_k(x)$.

When the nonlinear differential equations are considered, pseudospectral methods are more useful compared to the direct collocation and Galerkin methods. For instance, the power terms u^2 and u^3 as well as their derivatives can be computed from the kth iteration $u_k(x)$ directly, by using the pair of direct and inverse discrete Fourier transforms. Iterative methods become explicit methods suitable for numerical approximations of solutions of the time-independent nonlinear boundary-value PDE problem.

11.4 Summary and Notes

In this chapter, we studied properties of trigonometric interpolation and approximation and their applications to numerical solutions of ordinary and partial differential equations.

- **Section 11.1:** Trigonometric approximation is formulated in Problem 11.1. The solution to this problem is given in Theorem 11.1, while convergence of the approximation for smooth functions is described in Theorem 11.2. Trigonometric interpolation is formulated in Problem 11.2. The solution to this problem is given in Theorem 11.3 and is implemented in the pair of discrete Fourier transforms.

- **Section 11.2:** Discrete eigenvalue problems for difference equations are analyzed in connection to continuous eigenvalue problems for differential operators (Theorem 11.4). The error of trigonometric interpolation for real analytic functions is described in Theorem 11.5. Analysis of convergence of trigonometric interpolation shows that no Runge phenomenon can occur for trigonometric sums.

- **Section 11.3:** Two spectral methods originate from applications of the trigonometric approximation and interpolation, namely, the Galerkin and collocation methods. The simplest (diagonal) application of these methods is described in the example of the inhomogeneous heat equation. Errors of spectral methods and details of their numerical implementations are discussed for two examples of the boundary-value problems for second-order ordinary differential equations. An additional pseudospectral method is described in the context of the time-evolution PDE problem that embeds the boundary-value ODE problem.

Trigonometric approximations and Sturm–Liouville eigenvalue problems are presented in [27]. Trigonometric interpolations and spectral accuracy are covered in [28]. Applications of spectral methods to numerical approximations of solutions of ordinary and partial differential equations are treated in [9].

11.5 Exercises

1. Compute the Fourier series, the trigonometric approximant $f_m(x)$, and the trigonometric interpolant $F_m(x)$ for the function

$$f(x) = 1 - |x|, \qquad -1 < x < 1,$$

which is extended periodically with the period $L = 2$. Plot the functions $f_m(x)$, $F_m(x)$, and $f(x)$ on $[-1, 1]$ for $m = 10$. Plot the L^2-norm of the error E of the trigonometric approximation and interpolation versus

the number of terms m in log-log scale and find the power fits in the dependencies of E versus m. Compare the power fits with Theorem 11.2.

2. Repeat the previous exercise for the function $f(x)$ on $(0,1)$ reflected antisymmetrically on $(-1,0)$ and extended periodically with the period $L=2$. Observe the Gibbs phenomenon by plotting the local error of the trigonometric interpolation and approximation on $[-1,1]$ for different values of m.

3. Compute the trigonometric interpolation $F_m(x)$ of the function

$$f(x) = \frac{1+2\sin x}{3-2\cos x}, \qquad 0 \le x \le 2\pi$$

and plot it on $[0,2\pi]$ for $m=5$. Find the power fit for the 2-norm of the error E versus the number of terms m. Compare the power fit with Theorem 11.5.

4. Write the MATLAB function `[a] = LSsine(x,y,n)`, where `x` and `y` are the column vectors of m elements, $0 < n \le m$, and `a` is the column vector of n elements in the sine interpolation

$$F_{\text{sine}}(x) = a_1 \sin(\pi x) + a_2 \sin(2\pi x) + \ldots + a_n \sin(n\pi x).$$

Apply this function for the data points related to the function $f(x) = x(1-x)e^{-x}$ on $[0,1]$ with $n=m$ and plot the 2-norm of the error versus n.

5. Use the MATLAB function `fft` and compute the discrete Fourier transform of the function $f(x) = x(1-x^2)$ on $[0,1]$. Plot the trigonometric interpolant $F_m(x)$ on a dense grid of data points on $[-1,1]$. Explain why the resulting function is even in x. Modify the function $f(x)$ so that the same computation produces an odd function $F_m(x)$ on $[-1,1]$.

6. Find eigenvalues and eigenvectors for the discrete eigenvalue problem related to the set of orthogonal Hermite polynomials:

$$u_{k+1} - 2u_k + u_{k-1} - hx_k(u_{k+1} - u_{k-1}) + h^2\lambda u_k = 0, \qquad 1 \le k \le n,$$

subject to the Dirichlet boundary conditions $u_0 = u_{n+1} = 0$, where the points $\{x_k\}_{k=1}^{n+1}$ represent a uniform grid on $[-L,L]$ and L is sufficiently large, say, $L=10$. Plot the distribution of eigenvalues and the first five eigenvectors of the Hermite difference eigenvalue problem.

7. Consider the integral representation of the Bessel function

$$J_0(x) = \frac{1}{\pi}\int_0^\pi e^{ix\cos t}dt.$$

For a set of equally spaced grid points on the x-interval $[0,5]$, replace $e^{ix\cos t}$ for $t \in [0,\pi]$ with the trigonometric interpolant $F_m(t)$, integrate the function $F_m(t)$ analytically on $[0,\pi]$, and compute the Bessel function $J_0(x)$.

8. Consider the boundary-value ODE problem

$$y'' + x^2 y = 1, \qquad 0 < x < 1,$$

subject to the Dirichlet boundary conditions $y(0) = y(1) = 0$. Construct the numerical approximations of the solution based on the Galerkin and collocation methods. Find the power fits of the total square error of the numerical approximations versus the step size h on $[0,1]$.

9. Consider the linear eigenvalue problem

$$y'' + x^2 y = \lambda y, \qquad 0 < x < 1,$$

subject to the Dirichlet boundary conditions $y(0) = y(1) = 0$. Approximate the spectrum of eigenvalues by truncating the trigonometric approximation for $m = 100$ terms. Show that the value $\lambda = 0$ is not the eigenvalue in the Galerkin method. Repeat the exercise by replacing derivatives with matrices in the collocation method on the $n = 100$ equally spaced grid points.

10. Consider the Hill equation

$$-y'' + \cos xy = \lambda y, \qquad 0 < x < 2\pi$$

subject to the periodic boundary conditions $y(x+2\pi) = y(x)$. Expand the solution in the trigonometric series and truncate the system at $m = 100$ terms in the Galerkin method. Plot the spectrum of eigenvalues and the first five eigenfunctions for the smallest eigenvalues. Illustrate that the number of zeros of $y(x)$ on $(0,2\pi)$ increases in the same ascending order as the eigenvalues are sorted.

11. Repeat the previous exercise with the antiperiodic boundary conditions $y(x+2\pi) = -y(x)$. Show that each pair of eigenvalues with the antiperiodic eigenfunctions is located between each pair of eigenvalues with the periodic eigenfunctions and vice versa.

12. Consider the linear Schrödinger equation

$$iu_t = u_{xx}, \qquad 0 < x < 1, \ t > 0$$

for complex-valued function $u(x,t)$, subject to the Dirichlet boundary conditions $u(0,t) = u(1,t) = 0$ and the initial condition

$u(x,0) = e^{-x^2+ix^2}$. Construct the numerical approximation of the solution surface $u(x,t)$ for $x \in [0,1]$ and $t \in [0,10]$ based on the Galerkin and collocation methods supplemented with the exact ODE integration. Study how the 2-norm of the error converges as $h \to 0$ for different fixed values of t.

13. Consider the wave equation

$$u_{tt} = u_{xx} + \cos(2x), \qquad 0 < x < \pi, \quad t > 0,$$

subject to the Neumann boundary conditions $u_x(0,t) = u_x(\pi,t) = 0$ and the initial conditions $u(x,0) = u_t(x,0) = 0$. Construct the numerical approximations of the solution surface $u(x,t)$ for $x \in [0,\pi]$ and $t \in [0,3]$ based on the Galerkin and collocation methods and the Heun method. Show the two solution surfaces and the exact solution of the wave equation.

14. Consider the *Burgers* equation

$$u_t = uu_x + u_{xx}, \qquad 0 < x < 2\pi, \quad t > 0,$$

subject to the periodic boundary conditions $u(0,t) = u(2\pi,t)$ and $u_x(0,t) = u_x(2\pi,t)$ and the initial condition $u(x,0) = e^{-x^2}$. Compute the numerical approximation for $t \in [0,3]$ by using the pseudospectral method based on the direct and inverse discrete Fourier transforms. Study how convergence and stability of the pseudospectral method depend on step size h and time step τ.

15. Repeat the previous exercise with the pseudospectral method based on the collocation method supplemented by the matrix representation of the first and second derivatives of $u(x,t)$ in x. Compare convergence and stability between the two pseudospectral methods.

CHAPTER 12

Splines and Finite Elements

ALTHOUGH TRIGONOMETRIC INTERPOLATION offers an accurate numerical solution to the interpolation problem, it also may have some shortcomings. The discrete grid of x-values has to be equally spaced to enforce all nice properties of trigonometric interpolation, such as the orthogonality and convergence of discrete trigonometric functions. On the other hand, although polynomial interpolation is valid on nonequally spaced discrete grids, it may develop a polynomial wiggle. There exists an alternative method to overcome the limitations of both trigonometric and polynomial interpolations. If the entire interpolation interval is decomposed into smaller intervals connected at the given data points, the degree of interpolating polynomials can be reduced to avoid the polynomial wiggle. This idea leads to the *spline interpolation*, which is a basis for the *finite-element method*, a useful tool in numerical approximations of solutions of boundary-value problems for differential equations.

This chapter covers the construction, properties, and errors of spline and Hermite interpolations and applications of finite elements to numerical approximations of solutions of ordinary differential equations.

12.1 Spline Interpolation

We refer to the polynomial of a low degree between two adjacent data points as *a spline* and to the grid point that connects two adjacent splines as *a breaking point*. The simplest *linear* splines between each two subsequent breaking points coincide with the piecewise linear interpolation for the given set of data points. Starting with *quadratic* and *cubic* polynomials, you will see a difference between spline interpolation and piecewise polynomial interpolation. The former match the first-order, second-order, and higher-order derivatives of the resulting curve at the breaking points, whereas the latter prepares independent polynomials between each two subsequent breaking points. Similar to the uniform polynomial interpolation, spline and piecewise polynomial interpolations with higher-order (fourth, fifth, etc.) polynomials may develop polynomial wiggles. As a result, such higher-order interpolation is used less often.

Let the set of $(n+1)$ data points (x_1,y_1), (x_2,y_2), ..., (x_{n+1},y_{n+1}) be ordered in the ascending order of distinct breaking points

$$x_1 < x_2 < \ldots < x_{n+1} \tag{12.1}$$