6 Discrete Transform Methods

Transform methods can be viewed as semi-analytical alternatives to finite differences for spatial differentiation in applications where *high degree of accuracy* is required. This chapter is an introduction to transform methods, also referred to as spectral methods, for solution of partial differential equations. We shall begin with the discrete Fourier transform, which is applied to numerical differentiation of periodic data and for solving elliptic PDEs in rectangular geometries. Discrete Fourier transform is also used extensively in signal processing, but this important application of transform methods will not be discussed here. For non-periodic data we will use transform methods based on Chebyshev polynomial expansions. Once the basic machinery for numerical differentiation with transform methods is developed, we shall see that their use for solving partial differential equations is straightforward.

6.1 Fourier Series

Consider the representation of a continuous *periodic* function f as a combination of pure harmonics

$$f(x) = \sum_{k=-\infty}^{\infty} \hat{f}_k e^{ikx}, \tag{6.1}$$

where $\hat{f}_k$ is the Fourier coefficient corresponding to the wavenumber k. Here the k values are integers because the period is taken to be 2π. In Fourier analysis one is interested in knowing what harmonics contribute to f and by how much. This information is provided by $\hat{f}_k$. The Fourier series for the derivative of $f(x)$ is obtained by simply differentiating (6.1)

$$f'(x) = \sum_{k=-\infty}^{\infty} ik\hat{f}_k e^{ikx}. \tag{6.2}$$

By analogy with the Fourier transform of f in (6.1), the Fourier coefficients of f' are $ik\hat{f}_k$. In this section the machinery for calculating $\hat{f}_k$ will be developed for *discrete data*. Once $\hat{f}_k$ is obtained, it is simply multiplied by ik to obtain the Fourier coefficients of f'. The result is then substituted in the discrete version of (6.2) to compute f'.

6.1.1 Discrete Fourier Series

If the periodic function f is defined only on a discrete set of N grid points, $x_0, x_1, x_2, \dots, x_{N-1}$, then f can be represented by a discrete Fourier transform. Discrete Fourier transform of a sequence of N numbers, $f_0, f_1, f_2, \dots, f_{N-1}$ is defined by

$$f_j = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{f}_k e^{ikx_j} \qquad j = 0, 1, 2, \dots, N-1 \tag{6.3}$$

where

$$\hat{f}_{-\frac{N}{2}}, \hat{f}_{-\frac{N}{2}+1}, \dots, 0, \dots, \hat{f}_{\frac{N}{2}-1}$$

are the discrete Fourier coefficients of f. Here, we take N to be even and the period of f to be 2π. A consequence of 2π periodicity is having integer wavenumbers. The sequence f_j consists of the values of f evaluated at equidistant points along the axis $x_j = jh$ with the grid spacing $h = 2\pi/N$. Note that f is assumed to be a periodic function with $f_0 = f_N$, and thus, the sequence $f_0, f_1, \dots, f_{N-1}$ does not involve any redundancy. In the more general case of period of length L the wavenumbers appearing in the argument of the exponential would be $(2\pi/L)k$ instead of k, and the grid spacing becomes $h = L/N$, which results in an identical expression for the arguments of the exponentials as in the 2π periodic case. Thus, the actual period does not appear in the expression for the discrete Fourier transform of f, but it does appear in the expression for its derivative (see (6.2)).

Equation (6.3) constitutes N algebraic equations for the unknown (complex) Fourier coefficients $\hat{f}_k$. However, instead of using Gauss elimination, or some other solution technique from linear algebra to solve this system, it is much easier and more efficient, to use the *discrete* orthogonality property of the Fourier series to get the Fourier coefficients. Therefore, we will first establish the discrete orthogonality of Fourier series. Consider the summation

$$I = \sum_{j=0}^{N-1} e^{ikx_j} e^{-ik'x_j} = \sum_{j=0}^{N-1} e^{ih(k-k')j}.$$

If $h(k-k')$ is not a multiple of 2π, then I is a geometric series with the multiplier

$e^{ih(k-k')}$. Thus, for $k - k' \neq mN$ (m is an integer),

$$I = \frac{1 - e^{ih(k-k')N}}{1 - e^{ih(k-k')}}.$$

Since $h = 2\pi/N$, the numerator is zero and we have the following statement of discrete orthogonality:

$$\sum_{j=0}^{N-1} e^{ikx_j} e^{-ik'x_j} = \begin{cases} N, & \text{if } k = k' + mN, m = 0, \pm 1, \pm 2, \ldots \\ 0, & \text{otherwise.} \end{cases} \tag{6.4}$$

Now, we will use this important result to obtain the Fourier coefficients $\hat{f}_k$. Multiplying both sides of (6.3) by $e^{-ik'x_j}$ and summing from $j = 0$ to $N - 1$ results in

$$\sum_{j=0}^{N-1} f_j e^{-ik'x_j} = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \sum_{j=0}^{N-1} \hat{f}_k e^{ix_j(k-k')}.$$

Using the orthogonality property (6.4), we have

$$\hat{f}_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-ikx_j} \quad k = -\frac{N}{2}, \frac{N}{2} + 1, \ldots, \frac{N}{2} - 1. \tag{6.5}$$

Equations (6.3) and (6.5) constitute the discrete Fourier transform pair for the discrete data, f_j. Equation (6.5) is sometimes referred to as the forward transform (from the physical space x to the Fourier space k) and (6.3) is referred to as the inverse transform (for recovering the function from its Fourier coefficients).

6.1.2 Fast Fourier Transform

For complex data, straightforward summations for each transform ((6.3) or (6.5)) requires about $4N^2$ arithmetic operations (multiplications and additions), assuming that the values of the trigonometric functions are tabulated. An ingenious algorithm, developed in the 1960s and called the fast Fourier transform (FFT), reduces this operations count to $O(N \log_2 N)$. This is a dramatic reduction for large values of N. The original algorithm was developed for $N = 2^m$, but algorithms that allow more general values of N have since been developed. The fast Fourier transform algorithm has been the subject of many articles and books and therefore will not be presented here. Very efficient FFT computer programs are also available for virtually all computer platforms used for scientific computing. For example, *Numerical Recipes* has a set of programs for the general FFT algorithm and several of its useful variants for real functions and for sine and cosine transforms, which are mentioned later in this chapter.

6.1.3 Fourier Transform of a Real Function

Whether *f* is real or complex the Fourier coefficients of *f* are generally complex. However, when *f* is real, there is a useful relationship relating its Fourier coefficients corresponding to negative and positive wavenumbers. This property reduces the storage requirements; the original N real data points f_j are equivalently represented by $N/2$ complex Fourier coefficients. We can easily derive this relationship by revisiting (6.5). Changing k to $-k$ in (6.5) produces

$$\hat{f}_{-k} = \frac{1}{N}\sum_{j=0}^{N-1} f_j e^{ikx_j}. \tag{6.6}$$

Taking the complex conjugate of this expression and noting that since f is real it is equal to its own complex conjugate, we obtain

$$\hat{f}^*_{-k} = \frac{1}{N}\sum_{j=0}^{N-1} f_j e^{-ikx_j}. \tag{6.7}$$

Comparison with (6.5) leads to this important result for real functions

$$\hat{f}_{-k} = \hat{f}^*_k. \tag{6.8}$$

As mentioned in the previous section, there are fast transform programs for real functions that take advantage of this property to reduce the required memory and execution time.

EXAMPLE 6.1 Calculation of Discrete Fourier Transform

(a) Consider the periodic function $f(x) = \cos 3x$ with period 2π, defined on the discrete set of points $x_j = (2\pi/N)j$, where $j = 0, \ldots, N-1$. Since

$$f_j = \cos 3x_j = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{f}_k e^{ikx_j} = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{f}_k(\cos kx_j + i\sin kx_j),$$

calculation of the Fourier coefficients is straightforward and obtained by inspection. They are given by

$$\hat{f}_k = \begin{cases} 1/2 & \text{if } k = \pm 3, \\ 0 & \text{otherwise.} \end{cases}$$

The result is independent of the number of discrete points N as long as $N \geq 8$.

(b) Consider now the periodic square function (Figure 6.1), which is given by

$$f(x) = \begin{cases} 1 & \text{if } 0 \leq x < \pi \\ -1 & \text{if } \pi \leq x < 2\pi, \end{cases}$$

and defined on the same discrete set of points. Let $N = 16$. Instead of directly using (6.5) to calculate the Fourier coefficients, we use *Numerical*

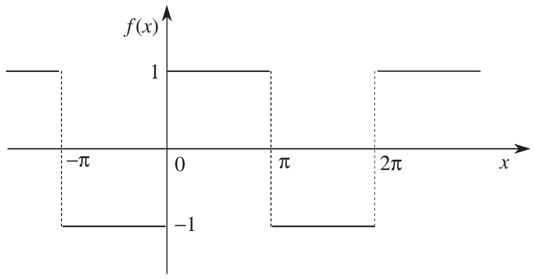

Figure 6.1 Periodic square function in Example 6.1(b).

Recipes' `realft` fast Fourier transform subroutine for real functions. The magnitudes of the Fourier coefficients are shown in Figure 6.2 and the coefficients corresponding to the positive wavenumbers are tabulated below. Fourier coefficients for negative wavenumbers are given by $\hat{f}_{-|k|} = \hat{f}^*_{|k|}$ because $f(x)$ is real.

k	$\mathbf{Re}(\hat{f}_k)$	$\mathbf{Im}(\hat{f}_k)$	$\lvert\hat{f}_k\rvert$
0	0	0	0
1	0.125	−0.628	0.641
2	0	0	0
3	0.125	−0.187	0.225
4	0	0	0
5	0.125	−0.084	0.150
6	0	0	0
7	0.125	−0.025	0.127
8	0	0	0

Using (6.5), it can be shown that if f_j is an odd function then its discrete

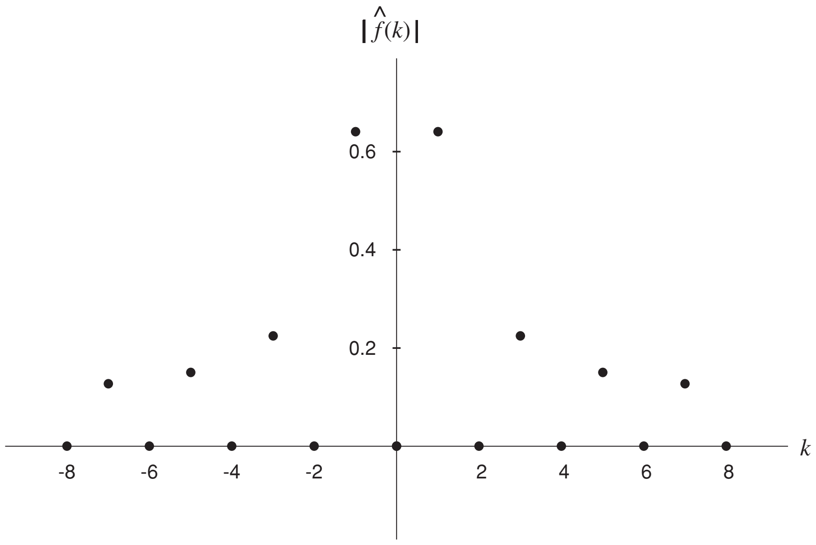

Figure 6.2 The magnitudes of the Fourier coefficients of the square function in Example 6.1(b).

Fourier transform $\hat{f}_k$ is imaginary and odd. The square function in this example can be made odd by redefining its values at 0 and π to be zeros instead of 1 and -1. In this case, the real part of the Fourier coefficients would be zero and the imaginary part would be unaltered compared to the original case.

6.1.4 Discrete Fourier Series in Higher Dimensions

The results and methodology of discrete Fourier transform can be extended to multiple dimensions in a straightforward manner. Consider the function $f(x, y)$ which is doubly periodic in the x and y directions and discretized using N_1 grid points in x and N_2 grid points in y. The two-dimensional Fourier series representation of f is given by

$$f(x_m, y_l) = \sum_{k_1=-\frac{N_1}{2}}^{\frac{N_1}{2}-1} \sum_{k_2=-\frac{N_2}{2}}^{\frac{N_2}{2}-1} \hat{f}_{k_1,k_2} e^{ik_1x_m} e^{ik_2y_l}$$
$$m = 0, 1, 2, \ldots, N_1 - 1 \quad l = 0, 1, 2, \ldots, N_2 - 1, \tag{6.9}$$

where $\hat{f}$ is the (complex) Fourier coefficient of f corresponding to wavenumbers k_1 and k_2 in the x and y directions respectively. Using the orthogonality result (6.4) for each direction, we obtain

$$\hat{f}_{k_1,k_2} = \frac{1}{N_1}\frac{1}{N_2} \sum_{m=0}^{N_1} \sum_{l=0}^{N_2} f_{m,l} e^{-ik_1x_m} e^{-ik_2y_l} \tag{6.10}$$
$$k_1 = -\frac{N_1}{2}, -\frac{N_1}{2} + 1, \ldots, \frac{N_1}{2} - 1 \quad \text{and} \quad k_2 = -\frac{N_2}{2}, -\frac{N_2}{2} + 1, \ldots, \frac{N_2}{2} - 1.$$

If f is real, it can be easily shown as in the previous section that

$$\hat{f}^*_{-k_1,-k_2} = \hat{f}_{k_1,k_2}.$$

Thus, Fourier coefficients in one half (*not one quarter*) of the (k_1, k_2) space are sufficient to determine all the Fourier coefficients in the entire (k_1, k_2) plane. All these results can be generalized to higher dimensions. For example, in three dimensions

$$\hat{f}^*_{-\boldsymbol{k}} = \hat{f}_{\boldsymbol{k}}$$

where $\boldsymbol{k} = (k_1, k_2, k_3)$ is the wavenumber vector.

6.1.5 Discrete Fourier Transform of a Product of Two Functions

The following is an important result that will be used later for the solution of non-linear equations by transform methods. Let

$$H(x) = f(x)g(x).$$

Our objective is to express the Fourier transform of H in terms of the Fourier transforms of f and g. The discrete Fourier transform of H is

$$\hat{H}_m = (\widehat{fg})_m = \frac{1}{N}\sum_{j=0}^{N-1} f_j g_j e^{-imx_j}.$$

Substituting for f_j and g_j their respective Fourier representations, we obtain

$$\hat{H}_m = \frac{1}{N}\sum_{j=0}^{N-1}\sum_{k}\sum_{k'} \hat{f}_k \hat{g}_{k'} e^{ikx_j} e^{ik'x_j} e^{-imx_j}. \tag{6.11}$$

The sum over j is non-zero only if $k + k' = m$ or $m \pm N$ (recall that, $x_j = (2\pi N)j$). The part of the summation corresponding to $k + k' = m \pm N$ is known as the *aliasing error* and should be discarded because the Fourier exponentials corresponding to these wavenumbers cannot be resolved on the grid of size N. Thus, using the definition (6.5) the Fourier transform of the product is

$$\hat{H}_m = \sum_{k=-N/2}^{N/2-1} \hat{f}_k \hat{g}_{m-k}. \tag{6.12}$$

This is the convolution sum of the Fourier coefficients of f and g. The inverse transform of $\hat{H}_m$ is sometimes used as the means to calculate the product of f and g. If we simply multiplied f and g at each grid point, the resulting discrete function would be "contaminated" by the aliasing errors and would not be equal to the inverse transform of $\hat{H}_m$ in (6.12). Aliasing errors are simply ignored in many calculations, in part because the alternative, alias-free method of calculation of the product via (6.12) is expensive, requiring $O(N^2)$ operations, and aliasing errors are usually small if sufficient number of grid points are used. However, in some large-scale computations aliasing errors have led to very inaccurate solutions. We will illustrate the effect of aliasing error in the following example.

EXAMPLE 6.2 Discrete Fourier Transform of a Product–Aliasing

Consider the functions $f(x) = \sin 2x$ and $g(x) = \sin 3x$ defined on the grid points $x_j = (2\pi/N)j$, where $j = 0, \dots, N-1$. For $N \geq 8$, their discrete Fourier transforms are

$$\hat{f}_k = \begin{cases} \mp i/2 & \text{if } k = \pm 2 \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad \hat{g}_k = \begin{cases} \mp i/2 & \text{if } k = \pm 3 \\ 0 & \text{otherwise.} \end{cases}$$

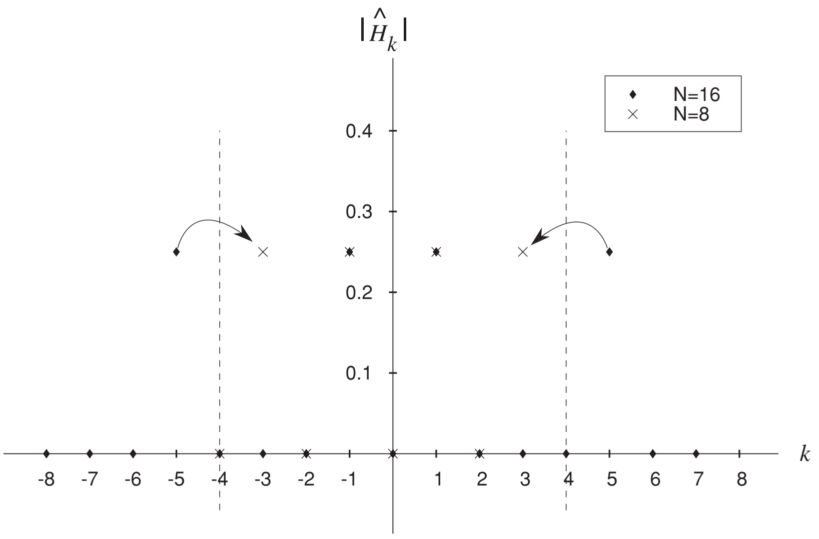

Figure 6.3 The magnitude of the Fourier coefficient $\hat{H}_k$ for $N = 8$ and $N = 6$ in Example 6.2.

Using trigonometric identities, their product $H(x) = f(x)g(x)$ is equal to $0.5(\cos x - \cos 5x)$.

We want to calculate the discrete Fourier transform of $H(x)$ using discrete values of f and g. For $N = 16$, using (6.12) or simply multiplying f and g at each grid point and inverse transforming, we obtain

$$\hat{H}_k = \begin{cases} 1/4 & \text{if } k = \pm 1 \\ -1/4 & \text{if } k = \pm 5 \\ 0 & \text{otherwise,} \end{cases}$$

which is the Fourier transform of the discrete function $0.5(\cos x_j - \cos 5x_j)$. Thus the exact Fourier coefficients of $H(x)$ are recovered.

We use now a smaller number of points ($N = 8$) to calculate the discrete Fourier coefficients of $H(x)$. Equation (6.12) gives

$$\hat{H}_k = \begin{cases} 1/4 & \text{if } k = \pm 1 \\ 0 & \text{otherwise,} \end{cases}$$

which corresponds to the discrete function $0.5 \cos x_j$. The 8-point grid is able to resolve Fourier modes up to the wavenumber $k = N/2 = 4$. Therefore, the part of $H(x)$ corresponding to $k = 5$ is lost when representing $H(x)$ discretely. The error involved is the truncation error since it results from truncating the Fourier series.

If we multiply f and g at each grid point and Fourier transform the result, we obtain

$$\hat{H}_k = \begin{cases} 1/4 & \text{if } k = \pm 1 \\ -1/4 & \text{if } k = \pm 3 \\ 0 & \text{otherwise,} \end{cases}$$

which is the Fourier transform of the discrete function $0.5(\cos x_j - \cos 3x_j)$! We notice the appearance of a new mode: $\cos 3x_j$. This is the aliasing error that has contaminated the results. It is the alias or misrepresentation of the $\cos 5x$ mode that appears (uses the alias) as $\cos 3x$. This is illustrated in Figure 6.3.

6.1.6 Discrete Sine and Cosine Transforms

If the function f is not periodic, transforms based on other than harmonic functions are usually more suitable representations of f. For example, if f is an even function (i.e., $f(x) = f(-x)$), expansion based on cosines would be a more suitable representation for f.

Consider the function f defined on an equidistant set of $N + 1$ points on the interval $0 \le x \le \pi$ on the real axis. Discrete cosine transform of f is defined by the following pair of relations

$$f_j = \sum_{k=0}^{N} a_k \cos kx_j \quad j = 0, 1, 2, \ldots, N \tag{6.13}$$

$$a_k = \frac{2}{c_k N} \sum_{j=0}^{N} \frac{1}{c_j} f_j \quad \cos kx_j \quad k = 0, 1, 2, \ldots, N, \tag{6.14}$$

where

$$c_l = \begin{cases} 2 & \text{if } l = 0, N \\ 1 & \text{otherwise,} \end{cases}$$

and $x_j = jh$ with $h = \pi/N$. Note that in contrast to the periodic Fourier transform, the values of f at both ends of the interval, f_0 and f_N, are included. Relation (6.13) is the definition of cosine transform for f. As in Fourier transforms, (6.14) is derived using the discrete orthogonality property of the cosines:

$$\sum_{j=0}^{N} \frac{1}{c_j} \cos kx_j \cos k'x_j = \begin{cases} 0 & \text{if } k \ne k' \\ \frac{1}{2} c_k N & \text{if } k = k'. \end{cases} \tag{6.15}$$

Discrete orthogonality of cosines given in (6.15) can be easily derived by substituting complex exponential representations for cosines in (6.15) and using geometric series, as was done in the Fourier case. Derivation of both equations (6.14) and (6.15) are left as exercises at the end of this chapter. Similarly, if f is an odd function (i.e., $f(x) = -f(-x)$), then it is best represented based on sine series. The sine transform pair is given by

$$f_j = \sum_{k=0}^{N} b_k \sin kx_j \quad j = 0, 1, 2, \ldots, N \tag{6.16}$$

$$b_k = \frac{2}{N} \sum_{j=0}^{N} f_j \sin kx_j \quad k = 0, 1, 2, \ldots, N. \tag{6.17}$$

Note that the $\sin kx_j$ term is zero at both ends of the summation index; they are included here to maintain similarity with the cosine transform relations.

EXAMPLE 6.3 Calculation of the Discrete Sine and Cosine Transforms

Consider the function $f(x) = x^2/\pi^2$, defined on the discrete points $x_j = (\pi/N)j$, where $j = 0, \ldots, N$. Let $N = 16$. We use *Numerical Recipes*' `cosft1` and `sinft` which are fast cosine and sine transform routines. The magnitudes of the coefficients are plotted in Figure 6.4. It is clear that the coefficients of the cosine expansion decay faster than those of the sine expansion. The sine expansion needs more terms to approximate the function on the whole interval as accurately as the cosine approximation because $f(\pi) \neq 0$. The odd periodic continuation of $f(x)$ is discontinuous at $x = \pi \pm 2n\pi$, n integer; the even continuation is not discontinuous (its slope is). The discontinuity slows the convergence of the expansion.

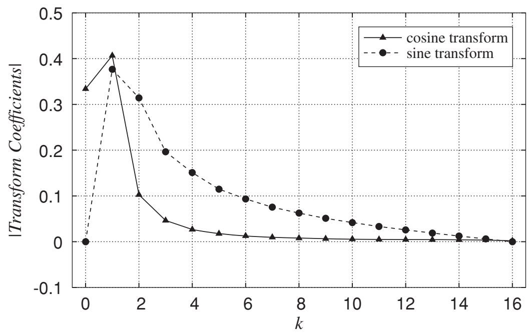

Figure 6.4 Magnitude of the cosine and sine transform coefficients for $f(x) = x^2/\pi^2$ in Example 6.3.

6.2 Applications of Discrete Fourier Series

6.2.1 Direct Solution of Finite Differenced Elliptic Equations

In this section we will give an example of a novel application of transform methods for solving elliptic partial differential equations. Consider the Poisson equation

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = Q(x, y)$$

with $\phi = 0$ on the boundaries of a rectangular domain. Suppose we seek a finite difference solution of this equation using a second-order finite difference scheme with $M + 1$ points in the x direction (including the boundaries) and $N + 1$ points in the y direction. Let the uniform mesh spacing in the x direction be denoted by Δ_1 and the mesh spacing in the y direction by Δ_2. The finite difference equations are

$$\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j} + \frac{\Delta_1^2}{\Delta_2^2}(\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}) = \Delta_1^2 Q_{i,j}, \qquad (6.18)$$

where

$$i = 1, 2, \ldots, M-1 \quad \text{and} \quad j = 1, 2, \ldots, N-1$$

are the mesh points inside the domain. This is a system of linear algebraic equations for the $(N-1) \times (M-1)$ unknowns. As pointed out in Section 5.10, for typical values of M and N, this system of equations is usually too large for a straightforward application of Gauss elimination. Here, we shall use sine series and the fast sine transform algorithm to obtain the solution of this *system of algebraic equations*.

Assume a solution of the form

$$\phi_{i,j} = \sum_{k=1}^{M-1} \hat{\phi}_{k,j} \sin\left[\frac{\pi k i}{M}\right] \quad i = 1, 2, \ldots, M-1, j = 1, 2, \ldots, N-1. \qquad (6.19)$$

Whether this assumed solution would work will be determined after substitution into (6.18). Note that the assumed solution does not include the boundaries, but it is consistent with the homogeneous boundary conditions. The sine transform of the right-hand side is similarly expressed as

$$Q_{i,j} = \sum_{k=1}^{M-1} \hat{Q}_{k,j} \sin\left[\frac{\pi k i}{M}\right] \quad i = 1, 2, \ldots, M-1, \quad j = 1, 2, \ldots, N-1.$$

Substituting these representations in the finite differenced equation (6.18), we obtain

$$\begin{aligned}
&\sum_{k=1}^{M-1} \hat{\phi}_{k,j} \left\{\sin\left[\frac{\pi k(i-1)}{M}\right] - 2\sin\left[\frac{\pi k i}{M}\right] + \sin\left[\frac{\pi k(i-1)}{M}\right]\right\} \\
&\quad + \sum_{k=1}^{M-1} \left(\frac{\Delta_1^2}{\Delta_2^2}\right) \left\{\hat{\phi}_{k,j+1} - 2\hat{\phi}_{k,j} + \hat{\phi}_{k,j-1}\right\} \sin\left[\frac{\pi k i}{M}\right] \\
&\quad = \Delta_1^2 \sum_{k=1}^{M-1} \hat{Q}_{k,j} \sin\left[\frac{\pi k i}{M}\right]. \qquad (6.20)
\end{aligned}$$

Using trigonometric identities, we have

$$\sin\left[\frac{\pi k(i+1)}{M}\right]-2\sin\left[\frac{\pi ki}{M}\right]+\sin\left[\frac{\pi k(i-1)}{M}\right]$$
$$=\sin\left[\frac{\pi ki}{M}\right]\left[2\cos\frac{\pi k}{M}-2\right].$$

By equating the coefficients of $\sin\pi ki/M$ in (6.20) (which amounts to using the discrete orthogonality property of the sines), we will obtain the following equation for the coefficients of the sine series:

$$\hat{\phi}_{k,j+1}+\left[\frac{\Delta_2^2}{\Delta_1^2}\left(2\cos\frac{\pi k}{M}-2\right)-2\right]\hat{\phi}_{k,j}+\hat{\phi}_{k,j-1}=\Delta_2^2\hat{Q}_{k,j}. \qquad (6.21)$$

For each k, this is a tridiagonal system of equations that can be easily solved.

Thus, the procedure for solving the Poisson equation can be summarized as follows. First, for each $j = 1, 2, \ldots, (N-1)$ the right-hand side function, $Q_{i,j}$ in (6.18), is sine transformed to obtain $\hat{Q}_{k,j}$:

$$\hat{Q}_{k,j}=\frac{2}{M}\sum_{i=1}^{M-1}Q_{k,j}\sin\left[\frac{\pi ki}{M}\right] \qquad k=1,2,\ldots,M-1, \quad j=1,2,\ldots,N-1.$$

Then, the tridiagonal system of equations (6.21) is solved for each $k = 1, 2, \ldots,$ $(M-1)$. Finally, $\phi_{i,j}$ is obtained from (6.19) using discrete fast sine transform.

Thus, the two-dimensional problem has been separated into $(M-1)$ one-dimensional problems. Since each sine transform requires $O(M\log_2 M)$ operations and each tridiagonal system $O(N)$ operations, overall, the method requires $O(NM\log_2 M)$ operations. It is a direct and a low-cost method for elliptic equations. However, the class of problems for which it works is limited. One must have a uniform mesh in the direction of transform (in this case, the x direction) and the coefficients in the PDE may not be a function of the transform direction. Non-uniform meshes and non-constant coefficients may be used in the other direction(s).

It should be emphasized that this solution procedure is simply a method for solving the system of linear equations (6.18). It is not a spectral numerical solution of the Poisson equation. Spectral methods is the subject of the remaining sections of this chapter. Furthermore, the sine series only involves the interior points. However, the fact that the representation for ϕ is also consistent with the boundary conditions is a key to the success of the method. For non-homogeneous boundary conditions, a change of variables must be introduced which would transform the inhomogeneity to the right-hand side term. For problems with Neumann boundary conditions, cosine series can be used instead of sine series.

EXAMPLE 6.4 Poisson Equation With Non-homogeneous Boundary Conditions

Consider the Poisson equation

$$\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}=30(x^2-x)+30(y^2-y)\quad 0\le x\le 1,\quad 0\le y\le 1,$$

with $\psi(0, y) = \sin 2\pi y$ and $\psi = 0$ on the other boundaries of the square domain. The exact solution is

$$\psi(x, y) = 15(x^2-x)(y^2-y)-\sin 2\pi y\frac{\sinh\, 2\pi(x-1)}{\sinh\, 2\pi}.$$

Let us solve the equation numerically using sine transform in the x direction. The dependent variable should have homogeneous boundary conditions at $x = 0$ and $x = 1$. Introducing a new variable $\phi(x,\ y)$ given by

$$\phi(x, y) = \psi(x, y) + (x-1)\sin\, 2\pi y$$

results in a new Poisson equation

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}=30(x^2-x)+30(y^2-y)\ -4\pi^2(x-1)\sin 2\pi y,$$

with $\phi(0,\ y) = \phi(1,\ y) = \phi(x,\ 0) = \phi(x,\ 1) = 0$. We now solve this equation for $M = N = 32$ ($\Delta_1 = \Delta_2 = 1/M$). For each j in (6.18), we use *Numerical Recipes*' `sinft` to obtain $\hat{Q}_{k,j}$, where $k = 1, 2, \ldots, (M-1)$. For each k, we solve the tridiagonal system of equations (6.21). Finally, $\hat{\phi}_{k,j}$ is transformed to $\phi_{i,j}$ using `sinft` again. The solution of the original equation is then given by

$$\psi_{i,j} = \phi_{i,j} - (x_i - 1)\sin 2\pi y_j.$$

Both numerical and exact solutions are plotted in Figure 6.5. The two plots are indistinguishable; the maximum error is 0.001.

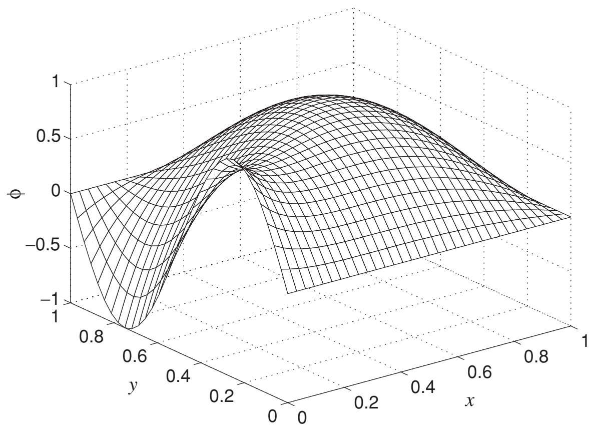

Figure 6.5 Numerical and exact solutions of the Poisson equation in Example 6.4.

6.2.2 Differentiation of a Periodic Function Using Fourier Spectral Method

The modified wavenumber approach discussed in Chapter 2 naturally points to the development of a highly accurate alternative to finite difference techniques: spectral numerical differentiation. Consider a periodic function $f(x)$ defined on N equally spaced grid points, $x_j = j\Delta$, and $j = 0, 1, 2, \ldots, N-1$. The spectral derivative of f is computed as follows. First, the discrete Fourier transform of f is computed as in (6.5)

$$\hat{f}_k = \frac{1}{N}\sum_{j=0}^{N-1} f_j e^{-ikx_j}$$

where,

$$k = \frac{2\pi}{L}n \quad n = -N/2,\ -N/2+2, \ldots, N/2-1.$$

Then, the Fourier transform of the derivative approximation is computed by multiplying the Fourier transform of f by ik

$$\widehat{Df_k} = ik\hat{f}_k \quad n = -N/2, -N/2+2,\ \ldots,\ N/2-1.$$

In practice, the Fourier coefficient of the derivative corresponding to the oddball wavenumber is set to zero, i.e., $\widehat{Df}_{-N/2} = 0$. This ensures that the derivative remains real in physical space (see Section 6.1.3), and it is only an issue when N is even.

Finally, the numerical derivative at a typical point j is obtained from inverse transformation

$$\left.\frac{\partial f}{\partial x}\right|_j = \sum_{k=-N/2}^{N/2-1} \widehat{Df}_k e^{ikx_j}.$$

It is easy to see that this procedure yields the *exact* derivative of the harmonic function $f(x) = e^{ikx}$ at the grid points if $|k| \le N/2-1$. In fact, the spectral derivative is more accurate than any finite difference scheme for periodic functions. The major cost involved is that of using the fast Fourier transform.

EXAMPLE 6.5 Differentiation Using the Fourier Spectral Method and Second-Order Central Difference Formula

(a) Consider the harmonic function $f(x) = \cos 3x$ defined on the discrete points $x_j = (2\pi/N)j$, where $j = 0, \ldots, N-1$. Its Fourier coefficients were calculated in Example 1(a). The Fourier coefficients of the derivative are given by $\widehat{Df}_k = ik\hat{f}_k$. They are therefore

$$\widehat{Df}_k = \begin{cases} -(3/2)i & \text{if } k = -3 \\ (3/2)i & \text{if } k = 3 \\ 0 & \text{otherwise.} \end{cases}$$

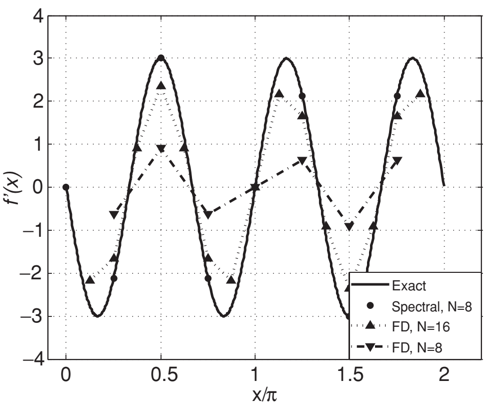

Figure 6.6 Numerical derivative of $\cos 3x$ in Example 6.5(a) using Fourier spectral method and second-order central finite difference formula (FD).

The corresponding inverse transform is $Df_j = -3 \sin 3x_j$, which is the exact derivative of $f(x) = \cos\, 3x$ at the grid points. This exact answer is obtained as long as $N \geq 8$ (because $N/2 - 1 \geq 3$). For comparison, the second-order central difference formula (2.7) is also used to compute the derivative. Results are plotted in Figure 6.6 for $N = 8$ and 16 points. It is clear that the finite difference method requires many more points to give a result as accurate as the spectral method.

(b) Consider now the function $f(x) = 2\pi x - x^2$ defined on the same discrete set of points. We compute the Fourier coefficients of f_j using *Numerical Recipes*' `realft`, multiply $\hat{f}_k$ by ik, set the Fourier coefficient corresponding to $-N/2$ to zero, and finally inverse transform using `realft` to obtain the numerical derivative of f_j. Results are plotted in Figure 6.7 for $N = 16$. The finite difference derivative (computed at the interior points) is exact since its truncation error for a quadratic is zero (see (2.7)). The spectral derivative is less accurate especially near the boundaries where the periodic continuation of $f(x)$ is discontinuous.

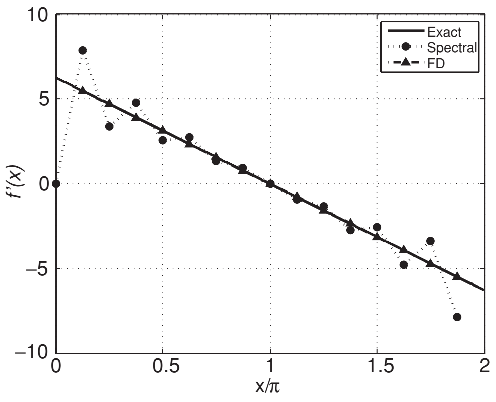

Figure 6.7 Numerical derivative of $2\pi x - x^2$ in Example 6.5(b) using Fourier spectral method and second-order finite differences (FD), with $N = 16$.

6.2.3 Numerical Solution of Linear, Constant Coefficient Differential Equations with Periodic Boundary Conditions

The Fourier differentiation technique is easily applied to the numerical solution of partial differential equations with periodic boundary conditions. Below we will present two examples, one for an elliptic equation, and another for an unsteady initial boundary value problem.

EXAMPLE 6.6 Poisson Equation

Consider the Poisson equation

$$\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} = Q(x,\, y) \tag{6.22}$$

in a periodic rectangle of length L_1 along the x axis and width L_2 along the y direction. Let us discretize the space with M uniformly spaced grid points in x and N grid points in y. The solution at each grid point is represented as

$$P_{l,j} = \sum_{n_1=-\frac{M}{2}}^{\frac{M}{2}-1} \sum_{n_2=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{P}_{k_1,k_2} e^{ik_1 x_l} e^{ik_2 y_j}$$

$$l = 0,\ 1,\ 2,\ \ldots,\ M-1 \quad j = 0,\ 1,\ 2,\ \ldots,\ N-1, \tag{6.23}$$

where

$$x_l = lh_1, \quad h_1 = \frac{L_1}{M}, \quad y_j = jh_2, \quad h_2 = \frac{L_2}{N}, \quad k_1 = \frac{2\pi}{L_1}n_1, \quad k_2 = \frac{2\pi}{L_2}n_2.$$

Substituting (6.23) and the corresponding Fourier series representation for $Q_{l,j}$ into (6.22) and using the orthogonality of the Fourier exponentials, we obtain

$$-k_1^2 \hat{P}_{k_1,k_2} - k_2^2 \hat{P}_{k_1,k_2} = \hat{Q}_{k_1,k_2}, \tag{6.24}$$

which can be solved for $\hat{P}_{k_1,k_2}$ to yield

$$\hat{P}_{k_1,k_2} = -\frac{\hat{Q}_{k_1,k_2}}{k_1^2 + k_2^2}. \tag{6.25}$$

This is valid when k_1 and k_2 are not both equal to zero. The solution of the Poisson equation (6.22) with periodic boundary conditions is indeterminant to within an arbitrary constant. We can therefore set

$$\hat{P}_{0,0} = c,$$

where c is an arbitrary constant. Recall that $\hat{P}_{0,0}$ is simply the average of P over the domain (see 6.10). The inverse transform of $\hat{P}_{k_1,k_2}$ yields the desired solution $P_{l,j}$. Note that if we sum both sides of the Poisson equation with periodic boundary conditions over the domain, we get

$$\sum_{x_l} \sum_{y_j} Q(x_l,\, y_j) = 0.$$

Thus, the prescribed Q should satisfy this condition for the well posedness of the equation. An equivalent presentation of this condition is $\hat{Q}_{0,0} = 0$ (see (6.10)). This consistency condition can also be deduced from (6.24) by setting both wavenumbers equal to zero.

EXAMPLE 6.7 Initial Boundary Value Problem

(a) Consider the convection–diffusion equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} + f(x,\, t) \tag{6.26}$$

in the domain $0 \le x \le L$, with periodic boundary conditions in x, and with initial condition $u(x,\,0) = u_0(x)$. Since u is periodic in space, we will expand it in discrete Fourier series

$$u(x_j,\, t) = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{u}_k(t) e^{ikx_j}.$$

Substitution into (6.26) and using the orthogonality of the Fourier exponentials yields

$$\frac{d\hat{u}_k}{dt} = -(ik + \nu k^2)\hat{u}_k + \hat{f}_k(t).$$

This is an ordinary differential equation that can be solved for each $k = (2\pi/L)n$, with $n = 0,\, 1,\, 2,\, \ldots,\, N/2 - 1$, using a time advancement scheme. Here, we are assuming that u is real and therefore we need to carry only half the wavenumbers. The solution at any time t is obtained by inverse Fourier transformation of $\hat{u}_k(t)$.

(b) As a numerical example, we solve

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0.05 \frac{\partial^2 u}{\partial x^2},$$

on $0 \le x \le 1$ with

$$u(x,\,0) = \begin{cases} 1 - 25(x - 0.2)^2 & \text{if } 0 \le x < 0.4 \\ 0 & \text{otherwise.} \end{cases}$$

Let $N = 32$. We first use *Numerical Recipes*' `realft` to inverse transform $u(x_j, 0)$ and obtain $\hat{u}_k(0)$, $k = 2\pi n$, $n = 0,\, 1,\, 2, \ldots, N/2 - 1$. Next we advance in time the differential equation

$$\frac{d\hat{u}_k}{dt} = -(ik + 0.05k^2)\hat{u}_k$$

for each k using a fourth-order Runge–Kutta scheme. This equation is exactly the model equation we studied in Chapter 4, i.e., $y' = \lambda y$.

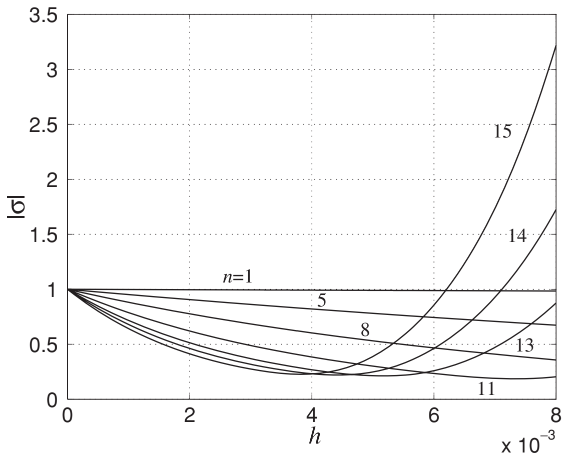

Figure 6.8 $|\sigma|$ versus h for $k = 2\pi n$, $n = 1, 5, 8, 11, 13, 14, 15$, in Example 6.7(b).

For stability, the time step h is chosen such that $\lambda h = -(ik + 0.05k^2)h$ falls inside the stability diagram of Figure 4.8. For fourth-order Runge–Kutta this means that

$$|\sigma| = \left|1 + \lambda h + \frac{\lambda^2 h^2}{2} + \frac{\lambda^3 h^3}{6} + \frac{\lambda^4 h^4}{24}\right| \le 1.$$

If we plot $|\sigma|$ versus h for each k (see Figure 6.8), we find that as h increases, $|\sigma|$ becomes greater than 1 for the largest k value first, $k = 2\pi\,(N/2 - 1)$. From the plot, the maximum value of h that can be used is 0.00620. In our calculation we used $h = 0.006$.

The solution is plotted in Figure 6.9 for $t = 0.25$, 0.5, and 0.75. The solution propagates and diffuses in time, in accordance with the properties of the convective–diffusion equation.

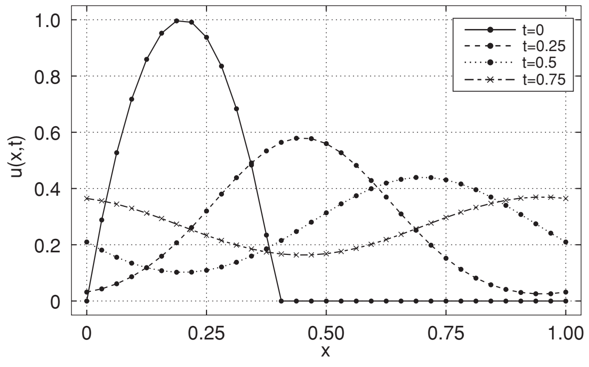

Figure 6.9 Numerical solution of the convective–diffusion equation in Example 6.7(b).

6.3 Matrix Operator for Fourier Spectral Numerical Differentiation

Up to this point we have described Fourier spectral numerical differentiation in terms of several steps: FFT of the function f, setting the oddball Fourier coefficient to zero, multiplying by ik, and inverse transforming back to the *physical space*. In some applications it is convenient or even necessary to have a compact representation of the spectral Fourier derivative operator in the physical space rather than the wave space. In this section we shall develop a physical space operator in the form of a matrix for numerical differentiation of a periodic discrete function and give an example of its application. This operator is, of course, completely equivalent to the familiar wave-space procedure.

Let u be a function defined on the grid

$$x_j = \frac{2\pi j}{N} \quad j = 0,\ 1, 2, \ldots, N-1.$$

Discrete Fourier transform of u is given by the following pair of equations:

$$\hat{u}_k = \frac{1}{N} \sum_{j=0}^{N-1} u(x_j) e^{-ikx_j} \tag{6.27}$$

and

$$u(x_j) = \sum_{k=-N/2}^{N/2-1} \hat{u}_k e^{ikx_j}.$$

Recall that the spectral derivative of u at the grid points is given by

$$(Du)_j = \sum_{k=-N/2+1}^{N/2-1} ik\hat{u}_k e^{ikx_j},$$

where the Fourier coefficient corresponds to the oddball wavenumber equal to zero (see Section 6.2.2). Substituting for $\hat{u}_k$ from (6.27) yields

$$(Du)_l = \frac{1}{N} \sum_{k=-N/2+1}^{N/2-1} \sum_{j=0}^{N-1} iku(x_j) e^{-ikx_j} e^{ikx_l} = \frac{1}{N} \sum_k \sum_j iku_j e^{\frac{2\pi ik}{N}(l-j)}$$
$$l = 0,\ 1,\ 2,\ \ldots, N-1.$$

Let

$$d_{lj} = \frac{1}{N} \sum_{k=-N/2+1}^{N/2-1} ike^{\frac{2\pi ik}{N}(l-j)} \quad l,\ j = 0, 1, 2, \ldots, N-1. \tag{6.28}$$

Then the derivative of u at each grid point is given by

$$(Du)_l = \sum_{j=0}^{N-1} d_{lj} u_j \quad l = 0, 1, 2, \ldots, N-1. \tag{6.29}$$

The right-hand side of this expression is in the form of multiplication of an $N \times N$ matrix D with elements d_{lj}, and the vector $\boldsymbol{u}$ with elements u_j. The matrix D is the physical space differentiation operator that we were after. We can simplify the expression for d_{lj} into a compact trigonometric expression without a summation. To evaluate the sum in (6.28), we first consider the geometric series

$$\begin{aligned}
S = \sum_{k=-N/2+1}^{N/2-1} e^{ikx} &= e^{i(-N/2+1)x} + e^{i(-N/2+2)x} + \cdots + e^{i(N/2-1)x} \\
&= e^{i(-N/2+1)x}\left[1 + e^{ix} + e^{2ix} + \cdots + e^{i(N-2)x}\right] \\
&= e^{i(-N/2+1)x}\frac{1-e^{i(N-1)x}}{1-e^{ix}} \\
&= \frac{e^{i(-N/2+1)x} - e^{i(N/2)x}}{1-e^{ix}} \\
&= \frac{e^{i(-N/2+1/2)x} - e^{i(N/2-1/2)x}}{e^{-ix/2} - e^{ix/2}} \\
&= \frac{\sin\left(\frac{N-1}{2}x\right)}{\sin\frac{x}{2}}.
\end{aligned}$$

This expression can be differentiated to yield the desired sum

$$\frac{dS}{dx} = \sum_{k=-N/2+1}^{N/2-1} ike^{ikx} = \frac{\left(\frac{N-1}{2}\right)\cos\left(\frac{N-1}{2}x\right)\sin\frac{x}{2} - \frac{1}{2}\cos\frac{x}{2}\sin\left(\frac{N-1}{2}x\right)}{\left(\sin\frac{x}{2}\right)^2}.$$

The result can be further simplified by using the trigonometric identities

$$\begin{aligned}
\sin\left(\frac{Nx}{2} - \frac{x}{2}\right) &= \sin\frac{Nx}{2}\cos\frac{x}{2} - \cos\frac{Nx}{2}\sin\frac{x}{2} \\
\cos\left(\frac{Nx}{2} - \frac{x}{2}\right) &= \cos\frac{Nx}{2}\cos\frac{x}{2} + \sin\frac{Nx}{2}\sin\frac{x}{2},
\end{aligned}$$

and noting that in (6.28) we could make the following substitution:

$$x = \frac{2\pi}{N}(l-j).$$

After these substitutions and simplifications, we finally arrive at

$$\frac{dS}{dx} = \frac{N}{2}(-1)^{l-j}\cot\left[\frac{\pi(l-j)}{N}\right].$$

Thus, the matrix elements for Fourier spectral differentiation are

$$d_{lj} = \begin{cases} \frac{1}{2}(-1)^{l-j}\cot\left[\frac{\pi(l-j)}{N}\right] & \text{if } l \neq j \\ 0 & \text{if } l = j. \end{cases} \tag{6.30}$$

This result for the diagonal elements of the matrix is obtained directly from (6.28).

The problem of Fourier spectral differentiation has thus been converted to a matrix multiplication in physical space as in (6.29), and transformation to the wave space is not necessary. Recall from linear algebra (see Appendix) that multiplication of a full matrix and a vector requires $O(N^2)$ operations, which is more expensive than the $O(N \log_2 N)$ operations for the Fourier transform method. However, in some applications such as the numerical solution of differential equations with non-constant coefficients, having a derivative operator in the physical space is especially useful. Finite difference operators can also be written in matrix form, but they always lead to banded matrices. The fact that the Fourier spectral derivative operator is essentially a full matrix reflects the global or fully coupled nature of spectral differentiation: the derivative of a function at any grid point is dependent on the functional values at *all* the grid points.

EXAMPLE 6.8 Burgers Equation

We illustrate the use of the derivative matrix operator by solving the non-linear Burgers equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \frac{\partial^2 u}{\partial x^2},$$

on $0 \le x \le 2\pi,\ 0 < t \le 0.6$ with $u(x, 0) = 10 \sin(x)$ and periodic boundary conditions. Using explicit Euler for time advancement yields the discretized form of the equation as

$$\boldsymbol{u}^{(n+1)} = \boldsymbol{u}^{(n)} + h(D^2\boldsymbol{u}^{(n)} - UD\boldsymbol{u}^{(n)}),$$

where $\boldsymbol{u}^{(n)}$ is a column vector with elements $u_j^{(n)}$ and $j = 0, \ldots, N-1$, D is a matrix whose elements are d_{lj} from (6.30), and U is a diagonal matrix formed from the elements of $\boldsymbol{u}^{(n)}$.

We estimate the time step h by performing stability analysis on the following linearized form of the Burgers equation:

$$\frac{\partial u}{\partial t} + u_{\max}\frac{\partial u}{\partial x} = \frac{\partial^2 u}{\partial x^2},$$

where $u_{\max}$ is the maximum absolute value of $u(x, t)$ over the given domain. We assume that the maximum value of $u(x, t)$ occurs at $t = 0$; that is, $u_{\max} = 10$. This assumption will be verified later by checking that the numerical solution of u does not exceed 10. Substituting the mode $\hat{u}_k(t)e^{ikx}$ for u, we have

$$\frac{d\hat{u}_k}{dt} = \lambda\hat{u}_k, \qquad \text{where} \quad \lambda = -k(k + iu_{\max}).$$

For stability of the explicit Euler method, the condition $|1 + \lambda h| \le 1$ must be satisfied. This is equivalent to $(1 + h\lambda_R)^2 + (h\lambda_I)^2 \le 1$ or

$$h \le -2\frac{\lambda_R}{|\lambda|^2}.$$

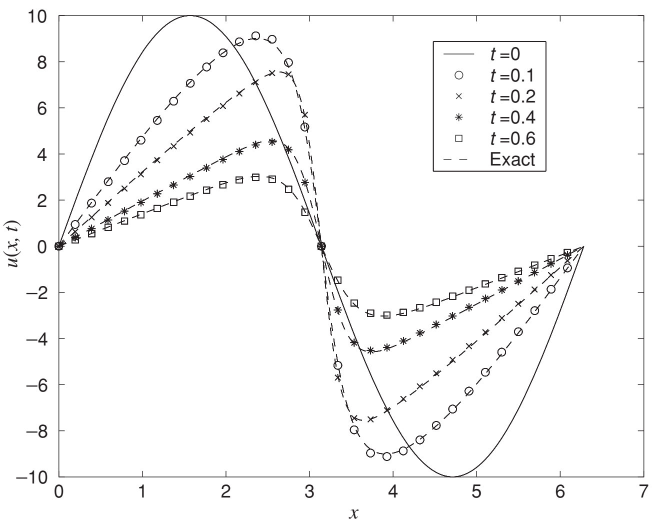

Figure 6.10 Numerical solution of the Burgers equation in Example 6.8.

Substituting $-k(k + iu_{\max})$ for λ gives

$$h \leq \frac{2}{k^2 + u_{\max}^2}.$$

The worst case scenario corresponds to the maximum value of $|k|$, i.e., $N/2$. For $N = 32$ and $u_{\max} = 10$, we obtain $h \leq 0.0056$. We use $h = 0.005$ in the present calculations. Solutions at $t = 0.1$, 0.2, 0.4, and 0.6 are shown in Figure 6.10.

The exact solution can be obtained from E. R. Benton and G. W. Platzman, "A table of solutions of the one dimensional Burgers equation," *Q. Appl. Math.* 30 (1972), p. 195–212, case 5. It is plotted in Figure 6.10 with dashed lines. The agreement is very good. In fact a similar agreement with the exact solution can be obtained with only $N = 16$.

The solution illustrates the main feature of the Burgers equation, which consists of a competition between convection and diffusion phenomena. The former causes the solution to steepen progressively with time, whereas the latter damps out high gradients. As a result, the solution first steepens and then slowly decays, as shown in Figure 6.10.

6.4 Discrete Chebyshev Transform and Applications

Discrete Fourier series are not appropriate for representation of non-periodic functions. When Fourier series are used for non-periodic functions, the convergence of the series with increasing number of terms is rather slow. In the remaining sections of this chapter we will develop the discrete calculus tools for non-periodic functions, using transform methods.

An arbitrary but smooth function can be represented efficiently in terms of a series of a class of *orthogonal polynomials* which are the eigenfunctions of the so-called *singular* Sturm–Liouville differential equations. Sines and cosines are examples of eigenfunctions of *non–singular* Sturm–Liouville problems. One of the advantages of using these polynomial expansions to approximate arbitrary functions is their superior resolution capabilities near boundaries. A rich body of theoretical work has established the reasons for excellent convergence properties of these series, which is outside the scope of this book. We will only use one class of these polynomials called Chebyshev polynomials.

An arbitrary smooth function, $u(x)$ defined in the domain $-1 \leq x \leq 1$ is approximated by a finite series of Chebyshev polynomials:

$$u(x) = \sum_{n=0}^{N} a_n T_n(x). \tag{6.31}$$

Chebyshev polynomials are solutions (eigenfunctions) of the differential equation

$$\frac{d}{dx}\left[\sqrt{1-x^2}\frac{dT_n}{dx}\right] + \frac{\lambda_n}{\sqrt{1-x^2}}T_n = 0,$$

where the eigenvalues $\lambda_n = n^2$. The first few Chebyshev polynomials are

$$T_0 = 1, \quad T_1 = x, \quad T_2 = 2x^2 - 1, \quad T_3 = 4x^3 - 3x, \ldots. \tag{6.32}$$

A key property of the Chebyshev polynomials is that they become simple cosines with the transformation of the independent variable $x = \cos\theta$, which maps $-1 \leq x \leq 1$ into $0 \leq \theta \leq \pi$. The transformation is

$$T_n(\cos\theta) = \cos n\theta. \tag{6.33}$$

This is the most attractive feature of Chebyshev polynomial expansions because the representation is reverted to cosine transforms, and in the discrete case one can take advantage of the FFT algorithm. Using a trigonometric identity, the following recursive relation for generating Chebyshev polynomials can be easily derived:

$$T_{n+1}(x) + T_{n-1}(x) = 2xT_n(x) \quad n \geq 1. \tag{6.34}$$

Other important properties of Chebyshev polynomials are

$$\begin{aligned} |T_n(x)| &\leq 1 \qquad \text{in } -1 \leq x \leq 1, \quad \text{and} \\ T_n(\pm 1) &= (\pm 1)^n. \end{aligned}$$

To use Chebyshev polynomials for numerical analysis, the domain $-1 \leq x \leq 1$ is discretized using the "cosine" mesh:

$$x_j = \cos\frac{\pi j}{N} \quad j = N,\ N-1,\ \ldots,\ 1,\ 0. \tag{6.35}$$

It turns out that these grid points are the same as a particular set of Gauss quadrature points discussed in Chapter 2. If the problem is defined on a different domain than $-1 \leq x \leq 1$, the independent variable should be transformed to $-1 \leq x \leq 1$. For example, the domain $0 \leq x < \infty$ can be mapped into $-1 \leq \psi < 1$ by the transformation:

$$x = \alpha \frac{1+\psi}{1-\psi}, \qquad \psi = \frac{x-\alpha}{x+\alpha},$$

where α is a constant parameter of the transformation.

As a direct consequence of the *discrete* orthogonality of cosine expansions, Chebyshev polynomials are discretely orthogonal under summation over $x_n = \cos(\pi n/N)$. That is

$$\sum_{n=0}^{N} \frac{1}{c_n} T_m(x_n) T_p(x_n) = \begin{cases} N & \text{if } m = p = 0, N \\ N/2 & \text{if } m = p \neq 0, N \\ 0 & \text{if } m \neq p, \end{cases}$$

where

$$c_n = \begin{cases} 2 & \text{if } n = 0, N \\ 1 & \text{otherwise.} \end{cases}$$

The discrete Chebyshev transform representation of a function u defined on a discrete set of points given by the cosine distribution in (6.35) is defined as

$$u_j = \sum_{n=0}^{N} a_n T_n(x_j) = \sum_{n=0}^{N} a_n \cos \frac{n\pi j}{N} \quad j = 0,\ 1,\ 2,\ \ldots,\ N \tag{6.36}$$

where the coefficients are obtained using the orthogonality property by multiplying both sides of (6.36) by $(1/c_j)T_p(x_j)$ and summing over all j:

$$a_n = \frac{2}{c_n N} \sum_{j=0}^{N} \frac{1}{c_j} u_j T_n(x_j) = \frac{2}{c_n N} \sum_{j=0}^{N} \frac{1}{c_j} u_j \cos \frac{n\pi j}{N}$$
$$n = 0,\ 1,\ 2,\ \ldots,\ N. \tag{6.37}$$

Comparing (6.36) to (6.13), the Chebyshev coefficients for any function u in the domain $-1 \leq x \leq 1$ are exactly the coefficients of the cosine transform obtained using the values of u at the cosine mesh (6.35); i.e., $u_j = u[\cos(\pi j/N)]$.

EXAMPLE 6.9 Calculation of the Discrete Chebyshev Coefficients

We calculate the Chebyshev coefficients of x^4 and $4(x^2 - x^4)e^{-x/2}$ on $-1 \leq x \leq 1$ using *Numerical Recipes'* `cosft1`. As long as $N \geq 4$, the coefficients for x^4 are

$$\begin{cases} a_0 = 0.375, & a_2 = 0.5, \quad a_4 = 0.125 \\ a_n = 0 & \text{otherwise.} \end{cases}$$

We can validate this result as follows. Using (6.32) and (6.34), T_4 is given by

$$T_4 = 2x\,T_3 - T_2 = 2x(4x^3 - 3x) - T_2 = 8x^4 - 6x^2 - T_2.$$

Substituting $T_2 + T_0$ for $2x^2$ gives

$$x^4 = 0.375T_0 + 0.5T_2 + 0.125T_4,$$

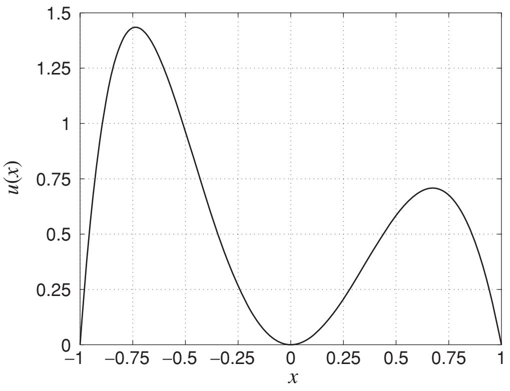

Figure 6.11 The function $u(x) = 4(x^2 - x^4)e^{-x/2}$ in Example 6.9.

which is in accordance with the coefficients obtained using `cosft1`. The function $u(x) = 4(x^2 - x^4)e^{-x/2}$ is plotted in Figure 6.11 and the magnitude of its Chebyshev coefficients for $N = 8$ are plotted in Figure 6.12. Strictly,

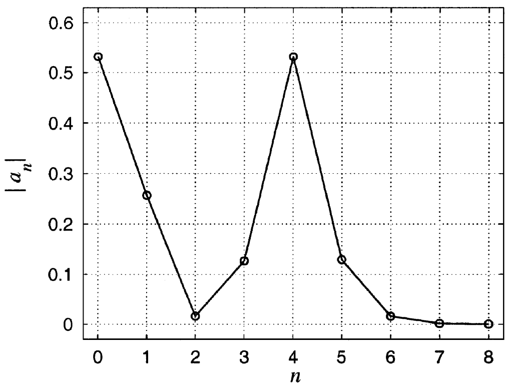

Figure 6.12 The magnitudes of the Chebyshev coefficients of $4(x^2 - x^4)e^{-x/2}$ in Example 6.9.

since u is not a polynomial it would have an infinite number of non-zero Chebyshev coefficients. However, the coefficients a_n are negligible for $n \geq 7$; i.e., only seven Chebyshev polynomials are needed to accurately represent $4(x^2 - x^4)e^{-x/2}$.

6.4.1 Numerical Differentiation Using Chebyshev Polynomials

The next step in the development of Chebyshev calculus is to derive a procedure for numerical differentiation of a function defined on the grid (6.35). Our objective is to obtain a recursive relationship between the coefficients of the Chebyshev transforms of a function and its derivative. In the case of Fourier expansion for a periodic function, this procedure was simply to multiply the Fourier transform of the function by ik. This is a bit more involved for Chebyshev representation, but not too difficult. Having the coefficients of the Chebyshev transform of the derivative, we obtain the derivative in the physical space on the grid (6.35) by inverse transformation.

We will first derive a useful identity relating the Chebyshev polynomials and their first derivatives. Recall from the definition of Chebyshev polynomials (6.35):

$$T_n(x) = \cos n\theta \qquad x = \cos\theta.$$

Differentiating this expression

$$\frac{dT_n}{dx} = \frac{d\cos n\theta}{d\theta}\frac{d\theta}{dx} = \frac{n\sin n\theta}{\sin\theta},$$

and using the trigonometric identity

$$2\sin\theta\cos n\theta = \sin(n+1)\theta - \sin(n-1)\theta,$$

we obtain the desired identity relating Chebyshev polynomials and their derivatives

$$2T_n(x) = \frac{1}{n+1}T'_{n+1} - \frac{1}{n-1}T'_{n-1} \quad n > 1. \tag{6.38}$$

Now consider the Chebyshev expansions of the function u and its derivative:

$$u(x) = \sum_{n=0}^{N} a_n T_n \tag{6.31}$$

$$u'(x) = \sum_{n=0}^{N-1} b_n T_n, \tag{6.39}$$

where a_n are the coefficients of u and b_n are the coefficients of its derivative. Note that since u is represented as a polynomial of degree N, its derivative can be a polynomial of degree at most $N-1$. Differentiating (6.31) and equating the result to (6.39) gives

$$\sum_{n=0}^{N-1} b_n T_n = \sum_{n=0}^{N} a_n T_n'.$$

Substituting for T_n using (6.38), we have

$$b_0 T_0 + b_1 T_1 + \sum_{n=2}^{N-1} b_n \frac{1}{2}\left[\frac{T_{n+1}'}{n+1} - \frac{T_{n-1}'}{n-1}\right] = \sum_{n=0}^{N} a_n T_n'. \tag{6.40}$$

Equating the coefficients of T_n', we finally obtain

$$\frac{b_{n-1}}{2n} - \frac{b_{n+1}}{2n} = a_n$$

or

$$b_{n-1} - b_{n+1} = 2na_n \quad n = 2, 3, \ldots, N-1, \tag{6.41}$$

where it is understood that $b_N = 0$ (see (6.39)). So far, we have $N-2$ equations for N unknowns. Equating the coefficients of T_N' on both sides of (6.40) yields

$$b_{N-1} = 2Na_N,$$

which is the same as we would obtain from (6.41), if we were to extend its range to N noting that $b_{N+1} = 0$. We still need one more equation. Noting that $T_1' = T_0$ and $T_2' = 4T_1$ from (6.40), we have

$$b_0 T_1' + \frac{1}{4} b_1 T_2' - \frac{b_2}{2} T_1' - \frac{1}{4} b_3 T_2' + \cdots = \sum_{n=0}^{N} a_n T_n'.$$

Equating the coefficients of T_1' from both sides gives

$$b_0 - \frac{1}{2} b_2 = a_1.$$

Hence, equation (6.41) can be generalized to yield all b_n as follows:

$$c_{n-1} b_{n-1} - b_{n+1} = 2na_n \quad n = 1, 2, \ldots, N \tag{6.42}$$

with $b_N = b_{N+1} = 0$.

In summary, to compute the derivative of a function u defined on the grid (6.35), one first computes its Chebyshev transform using (6.37), then the coefficients of its derivative are obtained from (6.42) by a straightforward marching from the highest coefficient to the lowest, and finally, the inverse transformation (6.36) is used to obtain u' at the grid points given by the cosine distribution.

A formal solution for the coefficients b_n in (6.42) can be written as

$$b_m = \frac{2}{c_m} \sum_{\substack{p=m+1 \\ p+m \text{ odd}}}^{N} p a_p. \tag{6.43}$$

The derivation of this equation is left as an exercise at the end of this chapter.

EXAMPLE 6.10 Calculation of Derivatives Using Discrete Chebyshev Transform

We want to calculate the derivatives of x^4 and $4(x^2 - x^4)e^{-x/2}$ defined on the cosine mesh inside the interval $-1 \le x \le 1$. We first calculate the coefficients b_n using (6.42) and the Chebyshev transform coefficients a_n already computed in Example 6.9. We then inverse transform b_n using `cosft1`, which is equivalent to (6.36), to obtain the derivative at the cosine mesh. For x^4 we obtain:

$$\begin{cases} b_1 = 3 & b_3 = 1 \\ b_n = 0 & \text{otherwise.} \end{cases}$$

This means that the derivative at the grid points is $3T_1(x_j) + T_3(x_j)$. From (6.32), this is equal to $4x_j^3$ which is the exact derivative of x^4 at the grid points.

The coefficients of the derivative of $4(x^2 - x^4)e^{-x/2}$ are computed and used to calculate the derivative, which is plotted in Figure 6.13 for $N = 5$. The results show good agreement with the exact derivative. For comparison, the derivative using second-order finite differences are also shown in Figure 6.13. In calculating the finite difference derivative, we use (2.7) for the interior grid points, (2.12) for the left boundary point, and

$$u'_j = \frac{3u_j - 4u_{j-1} + u_{j-2}}{2h}$$

at the right boundary point.

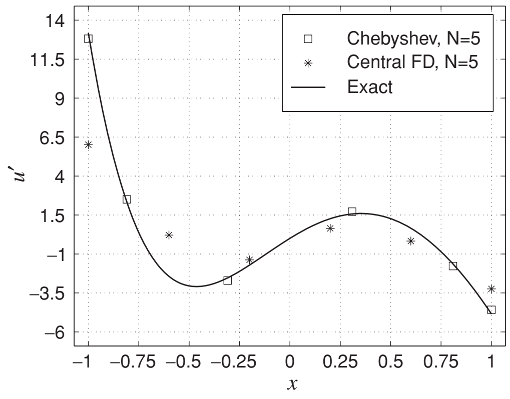

Figure 6.13 The derivative of $4(x^2 - x^4)e^{-x/2}$ using Chebyshev transform and central finite differences with $N = 5$ in Example 6.10.

6.4.2 Quadrature Using Chebyshev Polynomials

Equation (6.38) can also be used to derive a quadrature formula in a manner analogous to numerical differentiation. Integrating both sides of (6.38) leads to

$$\int T_n(x)dx = \begin{cases} T_1 + \alpha_0 & \text{if } n = 0 \\ \frac{1}{4}(T_0 + T_2) + \alpha_1 & \text{if } n = 1 \\ \frac{1}{2}\left[\frac{1}{n+1}T_{n+1} - \frac{1}{n-1}T_{n-1}\right] + \alpha_n, & \text{otherwise,} \end{cases}$$

where α_i are the integration constants. If u is represented by (6.31) and its definite integral $g(x) = \int_{-1}^{x} u(\xi)\,d\xi$ is represented by another Chebyshev expansion with coefficients d_n, then

$$\begin{aligned} g(x) &= \int_{-1}^{x} u(\xi)\,d\xi = \sum_{n=0}^{N+1} d_n T_n = \sum_{n=0}^{N} a_n \int T_n(x)\,dx \\ &= a_0 T_1 + \frac{a_1}{4}(T_0 + T_2) \\ &\quad + \sum_{n=2}^{N} \left\{ \frac{a_n}{2}\left[\frac{1}{n+1}T_{n+1} - \frac{1}{n-1}T_{n-1}\right] + \alpha_n \right\} + \alpha_0 + \alpha_1. \end{aligned}$$

Equating the coefficients of the same Chebyshev polynomials from both sides leads to the following recursive equation for the coefficients of the integral

$$d_n = \frac{1}{2n}(c_{n-1}a_{n-1} - a_{n+1}) \quad n = 1, 2, \ldots, N+1, \tag{6.44}$$

where it is understood that $a_{N+1} = a_{N+2} = 0$. All the integration constants and the coefficient of T_0 on the right-hand side can be combined into one integration constant that is equal to d_0. To obtain d_0, we note that $g(-1) = 0$, which leads to

$$\sum_{n=0}^{N+1} d_n(-1)^n = 0,$$

which can be solved for d_0 to yield

$$d_0 = d_1 - d_2 + d_3 - \cdots + (-1)^{N+2} d_{N+1}. \tag{6.45}$$

EXAMPLE 6.11 Calculation of Integrals Using Discrete Chebyshev Transform

We calculate the integrals

$$I_1 = \int_1^{\pi} \frac{\sin x}{2x^3}dx \quad \text{and} \quad I_2 = \int_1^{8} \frac{\log x}{x}dx$$

of Examples 3.1 and 3.4, respectively. The intervals of both integrals are transformed to $[-1, 1]$ by using the transformations or change of variables:

$$y = \frac{2x - (\pi + 1)}{\pi - 1} \quad \text{in } I_1, \quad \text{and} \quad y = \frac{2x - 9}{7} \quad \text{in } I_2.$$

The integrals then become

$$I_1 = \int_{-1}^{1} \frac{\pi - 1}{4} \frac{\sin[0.5(\pi - 1)y + 0.5(\pi + 1)]}{[0.5(\pi - 1)y + 0.5(\pi + 1)]^3} dy$$

and

$$I_2 = \int_{-1}^{1} \frac{7}{2} \frac{\log(3.5y + 4.5)}{3.5y + 4.5} dy.$$

These integrals are of the form $g(x) = \int_{-1}^{x} u(\xi)\, d\xi$. We first calculate a_n, the Chebyshev transform of the integrand $u(\xi)$, using `cosft1`. We then calculate d_n, the coefficients of its integral $g(x)$, from (6.44) and (6.45). Finally, we inverse transform d_n using `cosft1` to obtain $g(x)$ which can be evaluated at $x = 1$. In this case, we do not even need to inverse transform d_n to get $g(x)$ and then $g(1)$; $g(1)$ is simply equal to $\sum_{n=0}^{N+1} d_n$. The resulting % error in I_1 is 6.07×10^{-3} for $N = 8$ and 4.56×10^{-7} for $N = 16$, which is much lower than the error of any method in Example 3.1. The error ϵ in I_2 is 1.04×10^{-3} for $N = 8$ and 2.43×10^{-6} for $N = 16$. Comparing to Example 3.4, the Chebyshev quadrature performance is better than the performance of Simpson's rule but not as good as that of Gauss–Legendre quadrature.

6.4.3 Matrix Form of Chebyshev Collocation Derivative

As with Fourier spectral differentiation discussed in Section 6.3, it is sometimes desirable to have a physical space operator for numerical differentiation using Chebyshev polynomials. Consider the function $f(x)$ in the interval $-1 \le x \le 1$. We wish to compute the derivative of f on the set of collocation points $x_n = \cos \pi n/N$, with $n = 0, 1, 2, \ldots, N$. The discrete Chebyshev representation of f is given by

$$f(x) = \sum_{p=0}^{N} a_p T_p(x)$$

and

$$a_p = \frac{2}{N c_p} \sum_{n=0}^{N} \frac{1}{c_n} T_p(x_n) f_n \qquad c_n = \begin{cases} 2 & n = 0,\ N \\ 1 & \text{otherwise} \end{cases}$$
$$p = 0,\ 1,\ 2,\ 3, \ldots,\ N.$$

This expression can be written in matrix form for the vector of Chebyshev coefficients:

$$\boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \hat{T}\boldsymbol{f},$$

where

$$\hat{T} = \frac{2}{N}\begin{bmatrix} \frac{T_0(x_0)}{4} & \frac{T_0(x_1)}{2} & \cdots & \frac{T_0(x_N)}{4} \\ \frac{T_1(x_0)}{2} & T_1(x_1) & \cdots & \frac{T_1(x_N)}{2} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{T_N(x_0)}{4} & \frac{T_N(x_1)}{2} & \cdots & \frac{T_N(x_N)}{4} \end{bmatrix}.$$

Similarly the derivative of f is given by

$$f'(x_n) = \sum_{p=0}^{N} b_p T_p(x_n)$$

or

$$\boldsymbol{f}' = T\boldsymbol{b},$$

where

$$T = \begin{bmatrix} T_0(x_0) & T_1(x_0) & \cdots & T_N(x_0) \\ T_0(x_1) & T_1(x_1) & \cdots & T_N(x_1) \\ \vdots & \vdots & \vdots & \vdots \\ T_0(x_N) & T_1(x_N) & \cdots & T_N(x_N) \end{bmatrix}.$$

Recall that using (6.43), we can explicitly express the Chebyshev coefficients of f in terms of the Chebyshev coefficients of f':

$$b_p = \frac{2}{c_p} \sum_{\substack{n=p+1 \\ n+p \text{ odd}}}^{N} n a_n.$$

Again, in vector form this expression can be written as

$$\boldsymbol{b} = G\boldsymbol{a},$$

where

$$G_{pn} = \begin{cases} 0 & \text{if } p \geq n \text{ or } p+n \text{ even}, \\ \frac{2n}{c_p} & \text{otherwise.} \end{cases}$$

Thus, we have the following expression for f' at the collocation points:

$$\boldsymbol{f}' = TG\boldsymbol{a} = TG\hat{T}\boldsymbol{f} = D\boldsymbol{f},$$

where

$$D = TG\hat{T}. \tag{6.46}$$

The $(N+1) \times (N+1)$ matrix D is the desired physical space operator for Chebyshev spectral numerical differentiation. Multiplication of D by the vector consisting of the values of f on the grid results in an accurate representation of f' at the grid points. However, expression (6.46) for D is not very convenient because it is given formally in terms of the product of three matrices. It turns out that one can derive an explicit and compact expression for the elements of D using Lagrange polynomials as discussed in Chapter 1. This derivation is algebraically very tedious and is left as exercises for the motivated reader at the end of this chapter (Exercises 18 and 19); we simply state the result here. The elements of the $(N+1) \times (N+1)$ Chebyshev collocation derivative matrix D are

$$d_{jk} = \begin{cases} \frac{c_j(-1)^{j+k}}{c_k(x_j - x_k)} & j \neq k \\ \frac{-x_j}{2(1-x_j^2)} & j = k, \quad j \neq 0,\ N \\ \frac{2N^2+1}{6} & j = k = 0 \\ -\frac{2N^2+1}{6} & j = k = N, \end{cases} \tag{6.47}$$

where x_j are the locations of the grid points given by (6.35) and

$$c_j = \begin{cases} 2 & \text{if } j = 0,\ N \\ 1 & \text{otherwise.} \end{cases}$$

EXAMPLE 6.12 Calculation of Derivatives Using Chebyshev Derivative Matrix Operator

We use the Chebyshev derivative matrix operator to differentiate $u(x) = 4(x^2 - x^4)e^{-x/2}$ of Example 6.10. Let the vectors $\boldsymbol{x}$ and $\boldsymbol{u}$ represent the collocation points $x_n = \cos(\pi n/N)$, $n = 0,\ 1,\ 2, \ldots, N$, and the values of u at these points, respectively. For $N = 5$, $\boldsymbol{x}$ and $\boldsymbol{u}$ are

$$\boldsymbol{x} = \begin{bmatrix} 1.000 \\ 0.809 \\ 0.309 \\ -0.309 \\ -0.809 \\ -1.000 \end{bmatrix}, \qquad \boldsymbol{u} = \begin{bmatrix} 0 \\ 0.604 \\ 0.296 \\ 0.403 \\ 1.355 \\ 0 \end{bmatrix}.$$

The matrix operator D, whose elements are obtained from (6.47), is

$$D = \begin{bmatrix} 8.500 & -10.472 & 2.894 & -1.528 & 1.106 & -0.500 \\ 2.618 & -1.171 & -2.000 & 0.894 & -0.618 & 0.276 \\ -0.724 & 2.000 & -0.171 & -1.618 & 0.894 & -0.382 \\ 0.382 & -0.894 & 1.618 & 0.171 & -2.000 & 0.724 \\ -0.276 & 0.618 & -0.894 & 2.000 & 1.171 & -2.618 \\ 0.500 & -1.106 & 1.528 & -2.894 & 10.472 & -8.500 \end{bmatrix}.$$

We multiply D by $\boldsymbol{u}$ to obtain the derivative of $\boldsymbol{u}$ at the collocation points:

$$\boldsymbol{u}' = D\boldsymbol{u} = \begin{bmatrix} -4.581 \\ -1.776 \\ 1.717 \\ -2.703 \\ 2.502 \\ 12.813 \end{bmatrix}.$$

These values are exactly the ones obtained in Example 6.10 (see Figure 6.13).

EXAMPLE 6.13 Convection Equation with Non-constant Coefficients

We solve the equation

$$u_t + 2xu_x = 0 \qquad u(x, 0) = \sin 2\pi x,$$

on the domain $-1 \leq x \leq 1$ using the matrix form of the Chebyshev collocation derivative to calculate the spatial derivatives. This is a one-dimensional wave equation with characteristics going out of the domain at both ends and thus there is no need for boundary conditions. Using the explicit Euler scheme for time advancement, the discretized form of the equation is

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + h(-2XD\boldsymbol{u}^n),$$

where $\boldsymbol{u}^n$ is a column vector with elements u_j^n, $j = 0, \ldots, N-1$; D is a matrix whose elements are d_{lj} from (6.47); and X is a diagonal matrix with x_j, $j = 0, \ldots, N-1$, on its diagonal.

For $N = 16$ and $h = 0.001$, solutions at $t = 0.3$ and 0.6 are shown in Figure 6.14. The agreement with the exact solution $(\sin(\pi x e^{-2t}))$ is very good. Similar agreement can also be obtained with $N = 8$. From Figure 6.14, we see that the solution at the origin does not move. This is expected since the wave speed, x, is zero at $x = 0$. Also, the parts of the wave to the right and left of the origin propagated to the right and left, respectively. The wave shape is distorted since the speed of propagation is different from point to point.

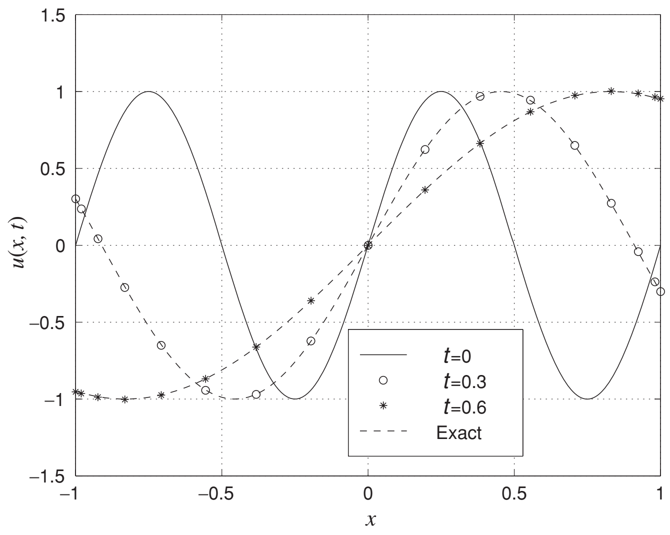

Figure 6.14 Numerical solution of the convection equation in Example 6.13.

6.5 Method of Weighted Residuals

The method of weighted residuals provides a framework for solving partial differential equations with transform methods. It is as a generalization of the methods discussed earlier where the numerical solution is expressed as a linear combination of a set of basis functions. The task at hand is to solve for the expansion coefficients by enforcing the differential equation in a weighted global or integral sense rather than by enforcing it at each spatial grid point.

A general statement of the problem we desire to solve is typically

$$\mathcal{L}(u) = f(x,t) \quad \text{for } x \in \mathcal{D} \tag{6.48}$$

with the general boundary conditions

$$\mathcal{B}(u) = g(x,t) \text{ on } \partial\mathcal{D}. \tag{6.49}$$

Here, the operator $\mathcal{L}(u)$ contains some spatial derivatives, such as a simple differential operator $\mathcal{L}(u) = \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} + u$, or a convective–diffusive operator $\mathcal{L}(u) = \frac{\partial u}{\partial t} + V\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2}$, and may either be linear or nonlinear in u.

The solution $u(x,t)$ is approximated by the function $\tilde{u}(x,t)$, which is assumed to be expressible as a combination of basis functions, ϕ_n:

$$\tilde{u} = \sum_{n=0}^{N} c_n(t)\phi_n(x). \tag{6.50}$$

The choice of basis functions used in the expansion depends on the application and the type of equation one wishes to solve. Frequently used choices for $\phi_n(x)$ include complex exponentials $e^{ik_n x}$, polynomials x^n, eigenfunctions of singular

Sturm–Louiville problems discussed in previous sections, or some variation thereof.

In general, the approximated solution $\tilde{u}$ does not satisfy the original equation (6.48) exactly. Instead, the method of weighted residuals aims to find the solution $\tilde{u}$ which minimizes the *residual* $R = \mathcal{L}(\tilde{u}) - f$ of (6.48) in the weighted integral sense:

$$\int_{\mathcal{D}} w_i R \, dx = 0 \quad i = 0, 1, \ldots, N,$$

for some weight functions $w_i(x)$. Inserting the expansion of the approximated solution (6.50) into the residual gives

$$\int_{\mathcal{D}} w_i(x) \left[\mathcal{L}\left(\sum_{n=0}^{N} c_n \phi_n \right) - f \right] dx = 0. \tag{6.51}$$

For operators $\mathcal{L}(u)$ that contain spatial differential operators, and for sufficiently differentiable weight functions $w_i(x)$, integration by parts turns equation (6.51) into the *weak form* of the original equation (6.48), which is the form ultimately used in the finite element method. A variety of weight functions $w_i(x)$ can be selected to solve equation (6.51). For weight functions (also called test functions) which are taken from the same space of functions as $\tilde{u}$, the method of weighted residuals is also known as the *Galerkin method.* Inserting $w_i(x) = \phi_i(x)$ into (6.51) gives the following system of equations for the unknown coefficients, c_n:

$$\int_{\mathcal{D}} \phi_i \left[\mathcal{L}\left(\sum_{n=0}^{N} c_n \phi_n \right) - f \right] dx = 0 \quad i = 0, 1, \ldots, N.$$

The Fourier spectral method used to solve equation (6.26) is an example of the Galerkin method with test functions $\phi_k(x) = (e^{ikx})^* = e^{-ikx}$.

In mathematical terms, the objective of the Galerkin method is to minimize the L_2 error by making the error orthogonal to the approximation subspace spanned by ϕ_i. This is the approach commonly used in deriving the finite element method.

6.6 The Finite Element Method

Although the finite element method can be developed from several different approaches, including variational or Rayleigh–Ritz procedure; only the method of weighted residuals is introduced below owing to its close connection to spectral methods described earlier. We first consider one-dimensional linear problems to simplify the analysis and to obtain a better understanding of the finite element method. However, the main advantage of the finite element method is in solving multi-dimensional problems in geometrically *complex* domains. Two-dimensional formulations will be discussed in Section 6.7.

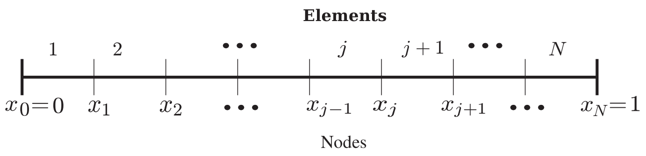

Figure 6.15 A schematic of the discretized domain, showing the placement of nodes x_j and elements $1, 2, \ldots, N$.

6.6.1 Application of the Finite Element Method to a Boundary Value Problem

For a simple illustration of the finite element method, we first consider the one-dimensional boundary value problem

$$\frac{d^2}{dx^2}u(x) + u(x) = f(x) \tag{6.52}$$

inside the domain $0 \le x \le 1$. We consider the case of general, or *natural*, boundary conditions at $x = 0$ and $x = 1$ expressed in the form

$$\begin{aligned} \alpha u(0) + \left.\frac{du}{dx}\right|_{x=0} &= A \\ \beta u(1) + \left.\frac{du}{dx}\right|_{x=1} &= B. \end{aligned} \tag{6.53}$$

Discretization of the domain in x is accomplished by placing $N - 1$ nodes in the interior, with node j located at x_j, as shown in Figure 6.15. The nodes also subdivide the domain into N *elements*, where the *jth* element occupies the region $x_{j-1} \le x \le x_j$ and has the width $\Delta_j = x_j - x_{j-1}$. In general, the nodes can be nonuniformly spaced throughout the domain, so each element may be of a different size. Although many choices of basis functions, $\phi_j(x)$, are possible, the simplest choice is piecewise linear functions defined by

$$\phi_j(x) = \begin{cases} 0 & x < x_{j-1} \\ \frac{x-x_{j-1}}{x_j-x_{j-1}} & x_{j-1} \le x < x_j \\ \frac{x-x_{j+1}}{x_j-x_{j+1}} & x_j \le x < x_{j+1} \\ 0 & x \ge x_{j+1} \end{cases} \qquad j = 1, 2, \ldots, N-1. \tag{6.54a}$$

with the functions ϕ_0 and ϕ_N given by

$$\phi_0(x) = \begin{cases} \frac{x-x_1}{x_0-x_1} & x_0 \le x < x_1 \\ 0 & x_1 \le x \end{cases} \tag{6.54b}$$

$$\phi_N(x) = \begin{cases} 0 & x < x_{N-1} \\ \frac{x-x_{N-1}}{x_N-x_{N-1}} & x_{N-1} \le x \le x_N. \end{cases} \tag{6.54c}$$

Higher-order polynomial versions of ϕ_j can also be constructed. However, the definition given by (6.54a)–(6.54c) satisfies the critical requirements for approximation functions: that they are continuous and differentiable within each

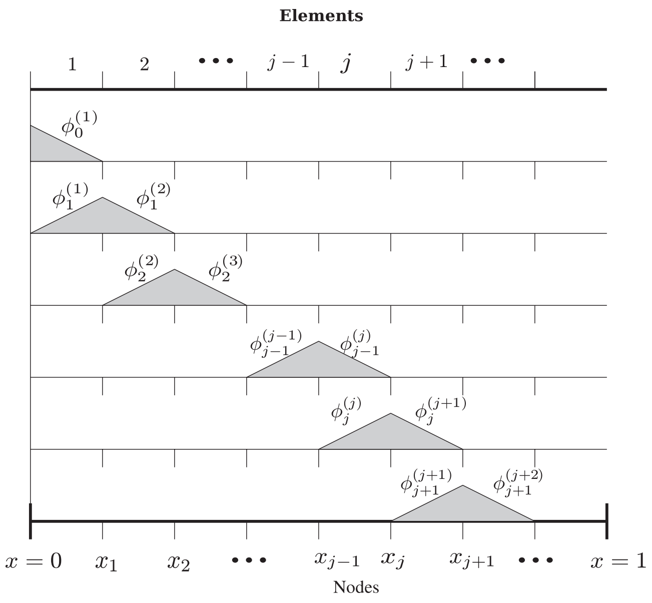
Elements
1
2
j − 1
j
j + 1
$\phi_0^{(1)}$
$\phi_1^{(1)}$ $\phi_1^{(2)}$
$\phi_2^{(2)}$ $\phi_2^{(3)}$
$\phi_{j-1}^{(j-1)}$ $\phi_{j-1}^{(j)}$
$\phi_j^{(j)}$ $\phi_j^{(j+1)}$
$\phi_{j+1}^{(j+1)}$ $\phi_{j+1}^{(j+2)}$
$x = 0$ x_1 x_2 x_{j-1} x_j x_{j+1} $x = 1$
Nodes

Figure 6.16 The series of approximating functions $\phi_j^{(m)}(x)$.

element. Polynomial approximation functions can be considered as *Lagrange polynomials*, and can be derived in a similar manner.

The fact that each basis function is nonzero only in two elements makes the subsequent computational procedures much simpler. The portion of $\phi_j(x)$ that resides on element m is denoted by $\phi_j^{(m)}(x)$, so definition (6.54a) can be re-expressed as

$$\phi_j^{(j)}(x) = \frac{x - x_{j-1}}{x_j - x_{j-1}}, \quad \phi_j^{(j+1)}(x) = \frac{x - x_{j+1}}{x_j - x_{j+1}}, \quad \phi_j^{(m)}(x) = 0 \ \text{for}\, m \neq j, j+1. \tag{6.55}$$

Thus, in a given element m, only two nonzero functions exist: $\phi_{m-1}^{(m)}(x)$ and $\phi_m^{(m)}(x)$. A diagram of the sequence of $\phi_j(x)$ functions is shown in Figure 6.16.

With this choice of basis functions, the numerical solution for $u(x)$ is expressed as:

$$u(x) \approx \tilde{u} = \sum_{j=0}^{N} u_j \phi_j(x), \tag{6.56}$$

where u_j are the values of $\tilde{u}(x)$ at the nodes x_j since $\phi_j(x_j) = 1$. For general basis functions, however, the coefficients, u_j, are not necessarily the same as the nodal values of the solution. The solution to (6.52) can be found in terms of the method of weighted residuals

$$\int_0^1 \left(\frac{d^2\tilde{u}}{dx^2} + \tilde{u} - f \right) w_i \, dx = 0 \quad i = 0, 1, \ldots, N.$$

Since first derivative of $\tilde{u}$ is discontinuous, but integrable (due to piecewise linearity of the basis functions), integration by parts can be used to avoid singularities in the weak form of the equations

$$\left[\frac{d\tilde{u}}{dx}w_i\right]_0^1 - \int_0^1 \frac{d\tilde{u}}{dx}\frac{dw_i}{dx}\,dx + \int_0^1 \tilde{u}w_i\,dx - \int_0^1 fw_i\,dx = 0. \tag{6.57}$$

Following the method of weighted residuals, the approximated form of the solution given by (6.56) is substituted into the integrals of (6.57), yielding

$$\left[\frac{d\tilde{u}}{dx}w_i\right]_0^1 - \int_0^1 \left(\frac{d}{dx}\sum_{j=0}^{N} u_j\phi_j\right)\frac{dw_i}{dx}\,dx$$
$$+ \int_0^1 \left(\sum_{j=0}^{N} u_j\phi_j\right) w_i\,dx - \int_0^1 fw_i\,dx = 0.$$

With the Galerkin method, the choice of weight function w_i is selected from the same set of interpolating polynomials listed above, so $w_i(x) = \phi_i(x)$. This produces a set of $N + 1$ equations for the unknown coefficients, u_j, and $\frac{du}{dx}$ at the boundaries.

$$\left[\frac{d\tilde{u}}{dx}\phi_i\right]_0^1 - \sum_{j=0}^{N} u_j \int_0^1 \frac{d\phi_j}{dx}\frac{d\phi_i}{dx}\,dx + \sum_{j=0}^{N} u_j \int_0^1 \phi_j\phi_i\,dx = \int_0^1 f\phi_i\,dx$$
$$i = 0, 1, \ldots, N. \tag{6.58}$$

With the boundary conditions given by (6.53) the system can now be closed with $N + 3$ equations for $N + 3$ unknowns. Incorporating the boundary conditions (6.53), equation (6.58) is re-written as:

$$\alpha u_0\delta_{i0} - \beta u_N\delta_{iN} - \sum_{j=0}^{N} u_j \int_0^1 \frac{d\phi_j}{dx}\frac{d\phi_i}{dx}\,dx + \sum_{j=0}^{N} u_j \int_0^1 \phi_j\phi_i\,dx$$
$$= \int_0^1 f\phi_i\,dx - B\delta_{iN} + A\delta_{i0} \quad i = 0, 1, \ldots, N, \tag{6.59}$$

where we have used the Kronecker delta symbol δ_{ij} to represent $\phi_i(0) = \delta_{i0}$ and $\phi_i(1) = \delta_{iN}$.

In the case of Dirichlet boundary conditions, where the values at the endpoints are specified as $u(0) = u_0 = a$ and $u(1) = u_N = b$, equation (6.58) produces a set of $N + 1$ equations. However, for these boundary conditions, the unknowns are the $N - 1$ nodal values u_j in the interior of the domain, plus the values of the derivatives at the boundaries: $\frac{du}{dx}|_{x=0}$ and $\frac{du}{dx}|_{x=1}$.

The systematic procedure of solving equation (6.59) follows by computing the integral quantities in terms of known parameters. Noting that this procedure requires different treatments for different boundary conditions, here, for simplicity, we only describe it for the case of homogeneous Dirichlet boundary conditions, $u_0 = u_N = 0$. In this case, the internal u_j values could be obtained by directly solving equation (6.58) for $i = 1, 2, \ldots, N-1$:

$$-\sum_{j=1}^{N-1} u_j \int_0^1 \frac{d\phi_j}{dx}\frac{d\phi_i}{dx}\,dx + \sum_{j=1}^{N-1} u_j \int_0^1 \phi_j\phi_i\,dx = \int_0^1 f\phi_i\,dx$$
$$i = 1, 2, \ldots, N-1, \tag{6.60}$$

which is a set of $N-1$ equations for the $N-1$ interior u_j coefficients. Note that the boundary term $[\frac{d\tilde{u}}{dx}\phi_i]_0^1$ vanishes for these values of i.

In general, the function $f(x)$ is supplied either analytically or given discretely at the nodes, x_j. If the analytical form of $f(x)$ is supplied, then the integral on the right-hand side of equation (6.60) can be computed directly. However, if the function is given only at the points x_j, we may use the following representation of f:

$$f(x) \approx \sum_{j=0}^{N} f_j\phi_j(x), \tag{6.61}$$

where $f_j = f(x_j)$. This allows equation (6.60) to be expressed in the more compact form

$$\sum_{j=1}^{N-1}(-D_{ij} + C_{ij})u_j = \sum_{j=0}^{N} C_{ij} f_j \quad i = 1, 2, \ldots, N-1, \tag{6.62}$$

where the symmetric matrices D_{ij} and C_{ij} are defined by the integrals

$$D_{ij} = \int_0^1 \frac{d\phi_i}{dx}\frac{d\phi_j}{dx}\,dx, \tag{6.63a}$$

and

$$C_{ij} = \int_0^1 \phi_i\phi_j\,dx. \tag{6.63b}$$

The task of computing these matrices is now straightforward, given the functional form of $\phi_i(x)$ in (6.54a). For instance, to compute D_{ij}, we first note that ϕ_i vanishes outside the region $x_{i-1} \le x \le x_{i+1}$, allowing us to restrict the integration to the elements i and $i+1$.

$$\begin{aligned} D_{ij} &= \int_0^1 \frac{d\phi_i}{dx}\frac{d\phi_j}{dx}\,dx \\ &= \int_{x_{i-1}}^{x_{i+1}} \frac{d\phi_i}{dx}\frac{d\phi_j}{dx}\,dx \\ &= \underbrace{\int_{x_{i-1}}^{x_i} \frac{d\phi_i^{(i)}}{dx}\frac{d\phi_j^{(i)}}{dx}\,dx}_{\text{element } i} + \underbrace{\int_{x_i}^{x_{i+1}} \frac{d\phi_i^{(i+1)}}{dx}\frac{d\phi_j^{(i+1)}}{dx}\,dx}_{\text{element } i+1}. \end{aligned}$$

Since the basis functions are linear inside each element, the integrands are constant for each integral. From (6.55) these constants can be computed yielding:

$$D_{ij} = \frac{d\phi_j^{(i)}}{dx} - \frac{d\phi_j^{(i+1)}}{dx}.$$

The nonzero elements of D_{ij} are

$$\begin{aligned} D_{i-1,i} &= -\frac{1}{\Delta_i} \\ D_{i,i} &= \frac{1}{\Delta_i} + \frac{1}{\Delta_{i+1}} \\ D_{i+1,i} &= -\frac{1}{\Delta_{i+1}}. \end{aligned}$$

The calculation of the matrix C_{ij} proceeds in an analogous manner and is deferred to the exercises at the end of this chapter. The nonzero matrix elements are:

$$\begin{aligned} C_{i-1,i} &= \frac{\Delta_i}{6} \\ C_{i,i} &= \frac{\Delta_i}{3} + \frac{\Delta_{i+1}}{3} \\ C_{i+1,i} &= \frac{\Delta_{i+1}}{6}. \end{aligned} \tag{6.64}$$

Combining both D_{ij} and C_{ij} into a single tridiagonal matrix $A_{ij} = C_{ij} - D_{ij}$ allows us to express equation (6.62) in the canonical form

$$\mathbf{Au} = \mathbf{b}. \tag{6.65}$$

The entries of the banded matrix A_{ij} are then given by

$$A_{j,j-1} = \frac{1}{\Delta_j} + \frac{\Delta_j}{6} \tag{6.66a}$$

$$A_{j,j} = -\left(\frac{1}{\Delta_j} + \frac{1}{\Delta_{j+1}}\right) + \frac{\Delta_j}{3} + \frac{\Delta_{j+1}}{3} \tag{6.66b}$$

$$A_{j,j+1} = \frac{1}{\Delta_{j+1}} + \frac{\Delta_{j+1}}{6} \tag{6.66c}$$

and the right-hand side is given by

$$b_j = \frac{\Delta_j}{6} f_{j-1} + \left(\frac{\Delta_j}{3} + \frac{\Delta_{j+1}}{3}\right) f_j + \frac{\Delta_{j+1}}{6} f_{j+1}. \tag{6.66d}$$

The solution of the $(N-1) \times (N-1)$ tridiagonal system (6.65) results in the values of u_j at the internal node points.

EXAMPLE 6.14 One-Dimensional Boundary Value Problem

Consider the solution to the differential equation

$$\frac{d^2}{dx^2}u(x) + u(x) = x^3 \tag{6.67}$$

over the domain $0 \le x \le 1$ and boundary conditions $u(0) = 0$ and $u(1) = 0$. The exact solution is

$$u(x) = -6x + x^3 + 5\csc(1)\sin(x).$$

Using a uniform mesh with N elements gives a mesh spacing of $\Delta = 1/N$, and grid points located at $x_j = j\Delta$, resulting in $f_j = (j\Delta)^3$. The solution u_j to the tridiagonal system (6.65) is plotted in Figure 6.17 for $N = 4$, $N = 8$, and $N = 16$ along with the exact solution. With eight elements the agreement with the exact solution is already very good.

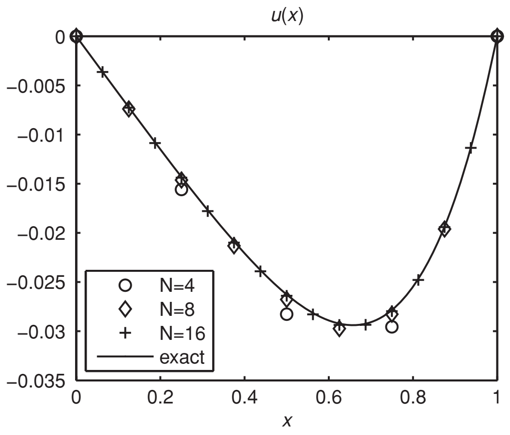

Figure 6.17 The solution u_j to equation (6.67) for $N = 4$ $N = 8$, and $N = 16$, compared with the exact solution.

6.6.2 Comparison with Finite Difference Method

If the mesh spacing is uniform, such that $\Delta_j = \Delta$, then equations (6.65) and (6.66a)–(6.66d) can be condensed into

$$\left(\frac{1}{\Delta} + \frac{\Delta}{6}\right)u_{j-1} + \left(-\frac{2}{\Delta} + \frac{2\Delta}{3}\right)u_j + \left(\frac{1}{\Delta} + \frac{\Delta}{6}\right)u_{j+1}$$
$$= \frac{\Delta}{6}f_{j-1} + \frac{2\Delta}{3}f_j + \frac{\Delta}{6}f_{j+1} \quad j = 1, 2, \ldots, N-1. \tag{6.68}$$

We can rearrange (6.68) into the following form:

$$\frac{u_{j+1}-2u_j+u_{j-1}}{\Delta^2}+\left[\frac{1}{6}u_{j+1}+\frac{2}{3}u_j+\frac{1}{6}u_{j-1}\right]=\frac{1}{6}f_{j-1}+\frac{2}{3}f_j+\frac{1}{6}f_{j+1}. \tag{6.69}$$

This discrete version of the original differential equation (6.52) contains three major terms: a second-order difference of $\frac{d^2u}{dx^2}|_j$, plus the weighted averages of u_j and f_j. In operator notation, this could be expressed as

$$D^2[u_j]+W[u_j]=W[f_j], \tag{6.70}$$

where the second-order central finite difference operator, D^2, is the product of the forward difference operator, $D_+[a_j]=\frac{a_{j+1}-a_j}{\Delta}$, and the backward difference operator, $D_-[a_j]=\frac{a_j-a_{j-1}}{\Delta}$, and is given by,

$$D^2[a_j]=\frac{a_{j+1}-2a_j+a_{j-1}}{\Delta^2},$$

and the weighted averaging operator is denoted by

$$W[a_j]=\frac{a_{j+1}+4a_j+a_{j-1}}{6}.$$

The order of accuracy of (6.69) can be established by obtaining its associated modified equation, similar to what we described in Section 5.5. Taylor series expansion of f_{j-1} and f_{j+1} result in:

$$W[f_j]=\frac{1}{6}f_{j-1}+\frac{2}{3}f_j+\frac{1}{6}f_{j+1}=f_j+\frac{\Delta^2}{6}f_j''+O(\Delta^4),$$

and the second-order finite difference of $\frac{d^2u}{dx^2}|_j$ is expanded as

$$D^2[u_j]=\frac{u_{j+1}-2u_j+u_{j-1}}{\Delta^2}=u_j''+\frac{\Delta^2}{12}u_j^{(iv)}+O(\Delta^4).$$

Collecting all the terms gives

$$\begin{aligned}&D^2[u_j]+W[u_j]-W[f_j]-(u_j''+u_j-f_j)\\&\quad=\Delta^2\left(\frac{1}{12}u_j^{(iv)}+\frac{1}{6}u_j''-\frac{1}{6}f_j''\right)+O(\Delta^4)\end{aligned} \tag{6.71}$$

If u_j satisfies the discretized equation in (6.70), it will satisfy the exact differential equation with an error term proportional to Δ^2,

$$u_j''+u_j-f_j=-\Delta^2\left(\frac{1}{12}u_j^{(iv)}+\frac{1}{6}u_j''-\frac{1}{6}f_j''\right)+O(\Delta^4), \tag{6.72}$$

showing that the finite element formulation is second-order accurate. The right-hand side of (6.71) can be further simplified. Taking the second derivative of (6.72) results in $u_j''-f_j''=-u_j^{(iv)}+O(\Delta^2)$ which can be substituted to simplify the right-hand side of (6.71).

$$D^2[u_j] + W[u_j] - W[f_j] - (u_j'' + u_j - f_j) = -\Delta^2 \left[\frac{1}{12} u_j^{(iv)}\right] + \cdots. \tag{6.73}$$

For comparison, a similar analysis for the standard finite difference discretization of (6.52) (without the weighted averaging) would give

$$D^2[u_j] + u_j - f_j - (u_j'' + u_j - f_j) = \Delta^2 \left[\frac{1}{12} u_j^{(iv)}\right] + \cdots, \tag{6.74}$$

showing that the two methods are equivalent with respect to order of accuracy; even the magnitudes of the leading order error terms are the same. The finite element method uses the weighted average of u and f instead of their local values. It is interesting that the method obtained from averaging (6.73) and (6.74) would be fourth-order accurate without any additional effort.

6.6.3 Comparison with a Padé Scheme

A similar comparison can be made between finite element and Padé schemes. Using the difference operator $D^2/(1 + \frac{1}{12}\Delta^2 D^2)$ to represent Padé differentiation (see Exercise 7 in Chapter 2), equation (6.52) can be discretized as

$$\frac{D^2[u_j]}{1 + \frac{1}{12}\Delta^2 D^2} + u_j = f_j.$$

Multiplying both sides by the operator $(1 + \frac{\Delta^2}{12} D^2)$ gives

$$D^2[u_j] + \left[1 + \frac{\Delta^2}{12} D^2\right] u_j = \left[1 + \frac{\Delta^2}{12} D^2\right] f_j,$$

which can be expanded in terms of a tridiagonal system at every point j

$$\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta^2} + \left[\frac{1}{12} u_{j+1} + \frac{5}{6} u_j + \frac{1}{12} u_{j-1}\right] = \frac{1}{12} f_{j-1} + \frac{5}{6} f_j + \frac{1}{12} f_{j+1}. \tag{6.75}$$

Notice that equation (6.75), which used the fourth-order Padé scheme for the second derivative in (6.52), also involves a second-order difference operator D^2 and a weighted averaging operator W_P:

$$D^2[u_j] + W_P[u_j] = W_P[f_j].$$

While the D^2 operator is identical to the one used by the finite element method, the weighted averaging operator W_P involves different coefficients. Also note that the result for Padé is the same as the average of (6.71) and (6.74), confirming its fourth-order accuracy.

6.6.4 A Time-Dependent Problem

Consider the constant coefficient convection equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{6.76}$$

over the domain $0 \le x \le 1$, with $N + 1$ grid points including the two boundaries. The finite element solution to (6.76) can be constructed by using the method of weighted residuals and integrating by parts with test functions $w_i(x)$ to obtain

$$\int_0^1 \frac{\partial \tilde{u}}{\partial t} w_i \, dx + c\tilde{u}w_i|_0^1 - c\int_0^1 \tilde{u}\frac{\partial w_i}{\partial x}\, dx = 0 \quad i = 0, 1, \ldots, N. \tag{6.77}$$

The function $\tilde{u}(x, t)$ can be represented in terms of linear interpolating functions ϕ_j:

$$\tilde{u}(x, t) = \sum_{j=0}^{N} u_j(t)\phi_j(x).$$

Substituting into (6.77) and using the Galerkin formulation, the system of equations becomes

$$\sum_{j=0}^{N} \frac{du_j}{dt}\int_0^1 \phi_j\phi_i \, dx - c\sum_{j=0}^{N} u_j \int_0^1 \phi_j \frac{d\phi_i}{dx}\, dx - cu_0\delta_{i0} + cu_N\delta_{iN} = 0$$
$$i = 0, 1, \ldots, N.$$

Consolidating all of the interpolation integrals into the matrices C_{ij} and D'_{ij} gives

$$\sum_j \left(\frac{du_j}{dt}C_{ij} - cu_jD'_{ij}\right) - cu_0\delta_{i0} + cu_N\delta_{iN} = 0, \tag{6.78}$$

where D'_{ij} is a tridiagonal matrix with nonzero entries given by

$$D'_{i,i-1} = \frac{1}{2}, \qquad D'_{i,i+1} = -\frac{1}{2},$$

and C_{ij}'s were given in Section 6.6.1. This leads to $N + 1$ equations for $N + 1$ unknown nodal values, u_j. However, to obtain a well-posed system one of the boundary equations should be replaced by a boundary condition. Assuming $c > 0$, the nodal value of u at the left boundary should be prescribed. For the interior nodes with uniform mesh spacing, Δ, the finite element formulation of (6.76) leads to the following tridiagonal system

$$\left[\frac{1}{6}\left.\frac{du}{dt}\right|_{j+1} + \frac{2}{3}\left.\frac{du}{dt}\right|_{j} + \frac{1}{6}\left.\frac{du}{dt}\right|_{j-1}\right] + \frac{c}{2\Delta}\left(u_{j+1} - u_{j-1}\right) = 0$$
$$j = 1, 2, \ldots, N - 1. \tag{6.79}$$

Compared to a straightforward application of central difference scheme, apparently, the finite element method leads to a weighted average of the time derivative scheme.

It is interesting to compare this result with the application of the Padé scheme to equation (6.76). Using the fourth-order Padé scheme for the spatial derivative in (6.76) leads to

$$\frac{du_j}{dt} + c\frac{D_0\left[u_j\right]}{1 + \frac{\Delta^2}{6}D^2} = 0,$$

where the central difference operator $D_0\left[u_j\right] = (u_{j+1} - u_{j-1})/2\Delta$. Multiplying both sides by $[1 + \frac{\Delta^2}{6}D^2]$

$$\left[1 + \frac{\Delta^2}{6}D^2\right]\frac{du_j}{dt} + cD_0\left[u_j\right] = 0,$$

and expanding the operators D^2 and D_0 and collecting terms, leads to the same system as the finite element method:

$$\left[\frac{1}{6}\frac{du}{dt}\bigg|_{j+1} + \frac{2}{3}\frac{du}{dt}\bigg|_{j} + \frac{1}{6}\frac{du}{dt}\bigg|_{j-1}\right] + \frac{c}{2\Delta}\left(u_{j+1} - u_{j-1}\right) = 0.$$

Thus, the finite element formulation with linear elements appears to be fourth-order accurate for this problem. This remarkable result appears to be coincidental.

The One-Dimensional Heat Equation

As another example of the application of the one-dimensional finite element method consider the time-dependent heat equation

$$\frac{\partial u}{\partial t} - \alpha\frac{\partial^2 u}{\partial x^2} = 0, \tag{6.80}$$

on a uniform grid with elements of width Δx. By strict analogy with the formulation of the boundary value problem in Sections 6.6.1 and 6.6.2, we can readily write the resulting discrete equations of the finite element method

$$W\left[\frac{du_j}{dt}\right] - \alpha D^2\left[u_j\right] = 0,$$

or

$$\frac{1}{6}\frac{du}{dt}\bigg|_{j-1} + \frac{2}{3}\frac{du}{dt}\bigg|_{j} + \frac{1}{6}\frac{du}{dt}\bigg|_{j+1} = \alpha\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2}.$$

For time advancement, the Crank–Nicolson scheme leads to

$$\begin{aligned}\frac{1}{6}\left(u_{j-1}^{n+1}-u_{j-1}^{n}\right)+\frac{2}{3}\left(u_{j}^{n+1}-u_{j}^{n}\right)+\frac{1}{6}\left(u_{j+1}^{n+1}-u_{j+1}^{n}\right)\\ =\beta\left(u_{j+1}^{n+1}-2u_{j}^{n+1}+u_{j-1}^{n+1}\right)+\beta\left(u_{j+1}^{n}-2u_{j}^{n}+u_{j-1}^{n}\right)\end{aligned}\tag{6.81}$$

where $\beta=\alpha\Delta t/2\Delta x^2$, and the subscript on u refers to the spatial grid, and the superscript refers to the time step. Equation (6.81) can be rearranged to yield a tridiagonal system for the solution at the next time step $n+1$.

$$\begin{aligned}\left(\frac{1}{6}-\beta\right)u_{j-1}^{n+1}+\left(\frac{2}{3}+2\beta\right)u_{j}^{n+1}+\left(\frac{1}{6}-\beta\right)u_{j+1}^{n+1}\\ =\left(\frac{1}{6}+\beta\right)u_{j-1}^{n}+\left(\frac{2}{3}-2\beta\right)u_{j}^{n}+\left(\frac{1}{6}+\beta\right)u_{j+1}^{n}\end{aligned}\tag{6.82}$$

EXAMPLE 6.15 Unsteady Heat Equation

Consider the one-dimensional heat equation

$$\frac{\partial u}{\partial t}-\alpha\frac{\partial^2 u}{\partial x^2}=0$$

on the domain $0\le x\le 1$ with Dirichlet boundary conditions, $u(x=0,t)=u(x=1,t)=0$, and the initial condition $u(x,t=0)=\sin(\pi x)$. The exact solution is $u(x,t)=\sin(\pi x)e^{-\alpha\pi^2 t}$. For the finite element solution to the problem, we employ a grid with N uniform elements of size $\Delta x=1/N$. The tridiagonal system (6.82) can be used to solve for u_i^n. For $N=8$, $\alpha=0.1$, $\Delta t=0.1$, the solution is plotted in Figure 6.18 along with the exact solution.

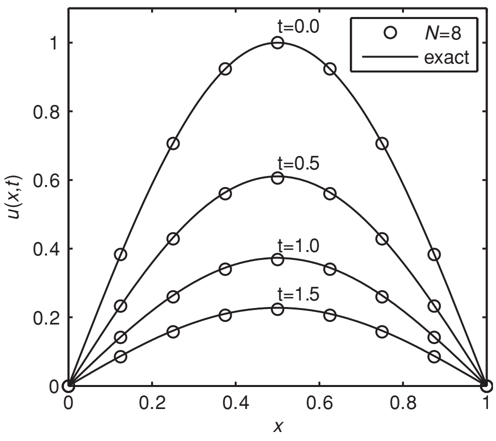

Figure 6.18 The solution to the one-dimensional heat equation for $N=8$ at times $t=0$, 0.5, 1.0, and 1.5.

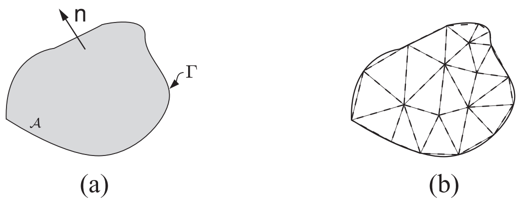

Figure 6.19 (a) A schematic of the two-dimensional domain $\mathcal{A}$ with boundary Γ, and (b) a possible mesh used to discretize the domain.

6.7 Application to Complex Domains

The procedures outlined for the one-dimensional problems can be extended naturally to two dimensions. However, while the formulation still remains manageable, much of the simplicity in one dimension disappears when the details of the geometry and basis functions are taken into account.

Consider the Poisson equation

$$\nabla^2 u = q(\mathbf{x}) \tag{6.83}$$

in the two-dimensional domain shown in Figure 6.19a with homogeneous Neumann boundary conditions, such that $\frac{\partial u}{\partial n} = 0$ on the boundary Γ. The domain is discretized into a series of nodal points and two-dimensional elements, such as triangles or quadrilaterals, connecting them. For simplicity, we consider only triangular elements in this discussion, with node points located at the vertices of the triangles (see Figure 6.19b). On this discretized mesh, we aim to find the value of the approximated solution $\tilde{u}(\mathbf{x})$ at each nodal point.

Following the method of weighted residuals, the residual $R = \nabla^2\tilde{u} - q$ is first integrated over the domain with a weighting function $w_i(x, y)$.

$$\int_{\mathcal{A}} w_i \left[\nabla^2\tilde{u} - q\right] d\mathcal{A} = 0. \tag{6.84}$$

The term in the integrand involving the Laplacian can be replaced with the following identity

$$\nabla \cdot (w_i \nabla\tilde{u}) - (\nabla w_i) \cdot (\nabla\tilde{u}) = w_i \nabla^2\tilde{u}. \tag{6.85}$$

In addition, the divergence theorem acting on the first term of (6.85) yields the following boundary term

$$\int_{\mathcal{A}} \nabla \cdot (w_i \nabla\tilde{u})\, d\mathcal{A} = \int_{\Gamma} w_i \frac{\partial\tilde{u}}{\partial n}\, d\Gamma, \tag{6.86}$$

where $\frac{\partial}{\partial n}$ is the derivative in the direction normal to the boundary and pointing outward. Note that equation (6.86) is equivalent to applying integration by parts to equation (6.84). Inserting both (6.85) and (6.86) into (6.84), yields the weak formulation of the problem:

$$-\int_{\mathcal{A}} (\nabla \tilde{u}) \cdot (\nabla w_i)\, d\mathcal{A} + \int_{\Gamma} \frac{\partial \tilde{u}}{\partial n} w_i \, d\Gamma = \int_{\mathcal{A}} w_i q \, d\mathcal{A}. \tag{6.87}$$

Returning to the homogeneous Neumann boundary conditions, $\frac{\partial \tilde{u}}{\partial n} = 0$ on the boundary Γ, the second term in (6.87) vanishes. For inhomogeneous Neumann boundary conditions, a finite flux $\frac{\partial \tilde{u}}{\partial n} = f$ is specified on the boundary of the domain, which can be absorbed into the inhomogeneous term $\int_{\mathcal{A}} w_i q \, d\mathcal{A}$.

To express equation (6.87) in two-dimensional Cartesian coordinates, the gradients of $\tilde{u}$ and w_i are written explicitly as

$$(\nabla \tilde{u}) \cdot (\nabla w_i) = \frac{\partial \tilde{u}}{\partial x}\frac{\partial w_i}{\partial x} + \frac{\partial \tilde{u}}{\partial y}\frac{\partial w_i}{\partial y}.$$

Similar to the one-dimensional problems (i.e., 6.56), the approximate solution can be expressed as a linear combination of basis functions $\phi_j(x, y)$

$$\tilde{u}(x, y) = \sum_{j=1}^{N} u_j \phi_j(x, y), \tag{6.88}$$

where the coefficients u_j are the values of the solution at the nodal points (x_j, y_j) and N is the number of basis functions (same as number of nodes). Note that N is typically smaller than the number of elements for triangular mesh. As in the Galerkin method, the weighting function is also selected from the same space of basis functions

$$w_i(x, y) = \phi_i(x, y) \quad i = 1, 2, \ldots, N. \tag{6.89}$$

In cases where the inhomogeneous term, $q(x, y)$, is given discretely at the nodal points, the right-hand side can also be expressed as

$$q(x, y) = \sum_{j=1}^{N} q_j \phi_j(x, y). \tag{6.90}$$

Substituting equations (6.88)–(6.90) into (6.87), we arrive at the finite element formulation for the Poisson equation:

$$-\sum_{j=1}^{N} u_j \int_{\mathcal{A}} \left(\frac{\partial \phi_i}{\partial x}\frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial y}\frac{\partial \phi_j}{\partial y} \right) dx\, dy = \sum_{j=1}^{N} q_j \int_{\mathcal{A}} \phi_i(x, y)\phi_j(x, y)\, dx\, dy$$
$$i = 1, 2, \ldots, N \tag{6.91}$$

where the summation is over all basis functions. As we shall see in Section 6.7.1, the basis functions are constructed such that they are nonzero only in the neighborhood of their corresponding node. This can be used to simplify equation (6.91) in a systematical routine. For example, for each i the domain of integration can be limited to the neighborhood of node i. Furthermore, the summation index j can be limited to those nodes whose basis functions overlap with that of node i. The integral on the left-hand side of (6.91) is termed the *Stiffness matrix*

$$K_{ij} = \int_{\mathcal{A}} \left(\frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial y} \frac{\partial \phi_j}{\partial y} \right) dx\, dy, \tag{6.92a}$$

while the integral on the right-hand side is termed the *Mass matrix*

$$M_{ij} = \int_{\mathcal{A}} \phi_i(x, y) \phi_j(x, y)\, dx\, dy. \tag{6.92b}$$

K_{ij} and M_{ij} are analogous to matrices D_{ij} and C_{ij} in one-dimensional case discussed earlier. These matrices allow equation (6.91) to be expressed compactly as

$$-\sum_{j=1}^{N} K_{ij} u_j = \sum_{j=1}^{N} M_{ij} q_j \quad i = 1, 2, \ldots, N. \tag{6.93}$$

This amounts to solving an $N \times N$ system for N unknown nodal values of u_j. Once a particular mesh geometry is specified and the basis functions ϕ_i defined, then both K_{ij} and M_{ij} can be calculated and equation (6.93) be solved for the values u_j.

6.7.1 Constructing the Basis Functions

In constructing the basis functions $\phi_i(x, y)$, the simplest and most convenient choice is to select piecewise linear functions on triangular elements. Following the same idea as in one-dimensional cases, each basis function is equal to one at a single node and is nonzero only on elements sharing that node. These properties uniquely determine N continuous basis functions corresponding to N nodes. Figure 6.20 shows a schematic of these functions. Separate linear relations are used to define basis functions on each triangular element. The coordinates of the nodes of each element can be employed to define these linear functions.

$$\phi_i(x, y) = \begin{cases} \frac{(x-x_j)(y_k-y_j)-(y-y_j)(x_k-x_j)}{(x_i-x_j)(y_k-y_j)-(y_i-y_j)(x_k-x_j)} & \text{if } (x, y) \text{ is in the element defined by nodes } i, j, k \\ 0 & \text{otherwise.} \end{cases} \tag{6.94}$$

Note in Figure 6.20 if (x, y) is in any of the five triangles with common vertex 1, then $\phi_1(x, y)$ would be nonzero. To use equation (6.94) to evaluate ϕ_1 in each one of these triangles, (x_i, y_i) should be replaced by the coordinates of node 1 and (x_j, y_j) and (x_k, y_k) should be replaced by coordinates of the two other nodes of the triangle.

Similar to the notation given in the one-dimensional case, ϕ_i^m is used to denote the ith basis function evaluated in the element m. ϕ_i^m is nonzero only

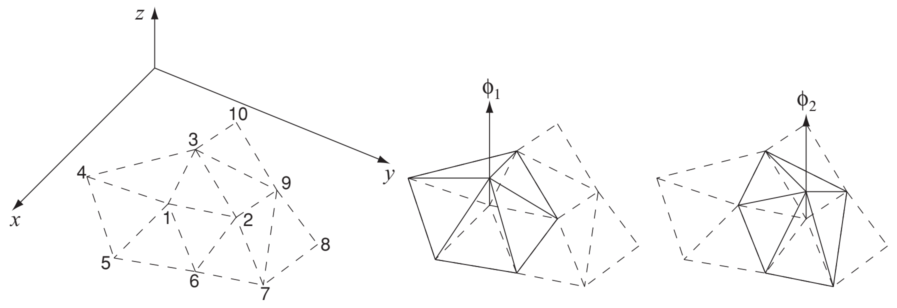

Figure 6.20 A schematic discretization of domain using triangular elements. Shown are 10 nodes and 9 elements and basis functions corresponding to nodes 1 and 2.

if node i is at the boundary of element m and can be written in the following form:

$$\phi_i^m(x, y) = a_i^m x + b_i^m y + c_i^m, \tag{6.95}$$

where coefficients a_i^m, b_i^m, and c_i^m are obtained from the coordinates of the element m nodes according to equation (6.94).

The next natural step would be to compute the matrices K_{ij} and M_{ij} by evaluating the integrals of equations (6.92a) and (6.92b). These integrals can be evaluated separately in each element and then summed over elements with nonzero contribution. Using equation (6.95) in the expression of the Stiffness matrix yields

$$K_{ij} = \sum_m A_m \left(a_i^m a_j^m + b_i^m b_j^m\right), \tag{6.96}$$

where A_m is the area of element m.

Some benefits can be gained by using local coordinates to evaluate these integrals. For example, the following integral which contributes to M_{ij}

$$\int_m \phi_i^m \phi_j^m \, dx \, dy, \tag{6.97}$$

can be evaluated using the coordinates ξ and η instead of x and y. Assuming $i \neq j$, the following choices for ξ and η simplifies the integration domain in (6.97)

$$\xi(x, y) = a_i^m x + b_i^m y + c_i^m = \phi_i^m(x, y),$$
$$\eta(x, y) = a_j^m x + b_j^m y + c_j^m = \phi_j^m(x, y).$$

Under this transformation, the integration domain maps to a triangle defined by coordinates (0,0), (1,0), and (0,1) as shown in Figure 6.21. The new expression of the integral is

$$\int_0^1 \int_0^{1-\eta} \xi\eta \frac{\mathrm{d}\xi \, \mathrm{d}\eta}{|a_i^m b_j^m - b_i^m a_j^m|} \qquad i \neq j.$$

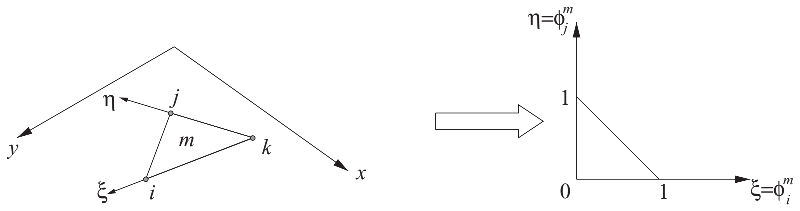

Figure 6.21 Element m with its nodes i, j, and k can be transformed to a simpler triangle using $\xi = \phi_i^m$ and $\eta = \phi_j^m$ as the new coordinates.

The constant coefficient in the denominator is the determinant of the Jacobian matrix. Using this expression which can be evaluated analytically, M_{ij} can be easily computed.

$$M_{ij} = \frac{1}{12} \sum_{i,j \in m} A_m \quad i \neq j. \tag{6.98a}$$

Similarly one can show that

$$M_{ii} = \frac{1}{6} \sum_{i \in m} A_m. \tag{6.98b}$$

In general, depending on the original partial differential equation, different integrals need to be evaluated to reduce the problem to a system of algebraic equations and transformation typically makes this task simpler.

EXAMPLE 6.16 Two-Dimensional Poisson Equation

Consider the Poisson equation

$$\nabla^2 u = -2\pi^2 \sin(\pi x) \sin(\pi y), \tag{6.99}$$

over the domain shown in Figure 6.22 with nonhomogeneous Dirichlet boundary conditions given in the figure. The exact solution to this equation is $u = \sin(\pi x) \sin(\pi y)$.

To obtain a finite element solution, first we need to decompose the domain into triangular elements. For this purpose simple meshing software such as Matlab's PDE toolbox routines, which are widely available can be used. A typical meshing routine outputs all the necessary information required for computing the mass and stiffness matrices. This information includes the coordinates of each node and the nodes of each element. For instance, the mesh shown in Figure 6.22 is linked with the following output for nodal coordinates:

$$\begin{gathered}
(x_1, y_1) = (0.3836, 0.3766) \\
(x_2, y_2) = (0.3736, 0.6364) \\
(x_3, y_3) = (0.6264, 0.3966) \\
\vdots \\
(x_{24}, y_{24}) = (1.0, 0.5).
\end{gathered}$$

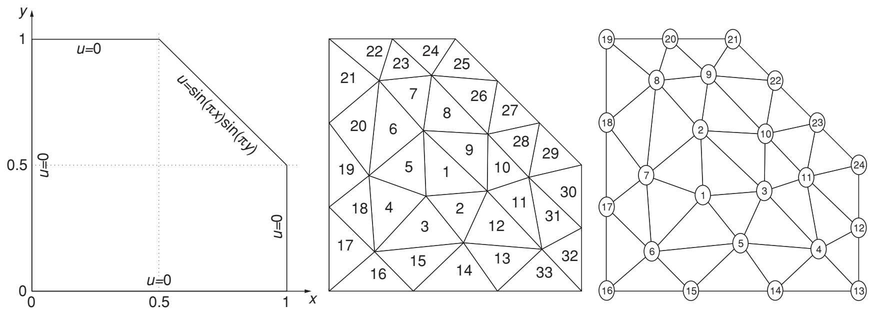

Figure 6.22 A schematic of the geometry and boundary conditions used in Example 6.3. The element and nodal indices are shown on the right. Thirty-three elements are defined by 11 interior nodes and 13 nodes at the boundary.

The nodal indices of elements are typically given in a matrix format. In this example, a 33×3 integer matrix is generated by the meshing routine corresponding to nodal indices of the 33 triangular elements. The first few lines of this matrix are:

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 5 \\ 1 & 5 & 6 \\ 1 & 6 & 7 \\ & \dots & \end{bmatrix}.$$

In other words, the first element involves nodes 1,2, and 3; the second element involves nodes 1, 3, 5, etc.

The area of each element can be computed from this information. For the element m with nodes i, j, k the area is:

$$A_m = |(x_j - x_i)(y_k - y_i) - (y_j - y_i)(x_k - x_i)|/2.$$

For example, the area of the first element is 0.03164. Next, equation (6.94) is used to compute the basis functions in each element.

$$\phi_1^1 = -3.7897x - 3.9950y + 3.9582$$
$$\phi_2^1 = -0.3161x + 3.8369y - 1.3238.$$
$$\dots$$

In other words, $(a_1^1, b_1^1) = (-3.7897, -3.9950)$, $(a_2^1, b_2^1) = (-0.3161, 3.8369)$, etc. In a typical computer program, by looping through all triangular elements the necessary information can be computed and stored for subsequent use.

Each triangular element contributes into nine different elements of matrices K_{ij} and M_{ij}. For example, element 5 contributes into K_{11}, K_{12}, K_{17}, K_{21}, K_{22}, K_{27}, K_{71}, K_{72}, and K_{77}. By looping through all triangles these contributions can be summed to obtain elements of matrices K_{ij} and M_{ij}. For example, from equation (6.96) the contribution of element 1 to K_{12} is

$$A_1(a_1^1 a_2^1 + b_1^1 b_2^1) = 0.03164 \times (3.7897 \times 0.3161 - 3.9950 \times 3.8369) = -0.4471.$$

$$K_{ij} = \begin{bmatrix} \times & \times & \times & 0 & \times & \times & \times & 0 & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 & 0 & \times & \times & \times & \times & 0 \\ \times & \times & \times & \times & \times & 0 & 0 & 0 & 0 & \times & \times \\ 0 & 0 & \times & \times & \times & 0 & 0 & 0 & 0 & 0 & \times \\ \times & 0 & \times & \times & \times & \times & 0 & 0 & 0 & 0 & 0 \\ \times & 0 & 0 & 0 & \times & \times & \times & 0 & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 & \times & \times & \times & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 & \times & \times & \times & 0 & 0 \\ 0 & \times & 0 & 0 & 0 & 0 & 0 & \times & \times & \times & 0 \\ 0 & \times & \times & 0 & 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & 0 & 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}$$

Figure 6.23 Nonzero elements of K_{ij} indicated by "$\times$" for $1 \le i, j \le 11$.

Following this procedure, the complete 24×24 matrices K_{ij} and M_{ij} can be computed. Then equation (6.93) can be used to solve for the 11 interior nodal values

$$-\sum_{j=1}^{24} K_{ij} u_j = \sum_{j=1}^{24} M_{ij} q_j \quad i = 1, 2, \ldots, 11.$$

The boundary values, $u_{12}, u_{13}, \ldots, u_{24}$, are already known from the boundary condition given in Figure 6.21. This leads to the following 11×11 system for the unknown coefficients $u_1, u_2, \ldots, u_{11}$.

$$-\sum_{j=1}^{11} K_{ij} u_j = \sum_{j=12}^{24} K_{ij} u_j + \sum_{j=1}^{24} F_{ij} q_j \quad i = 1, 2, \ldots, 11. \tag{6.100}$$

The nonzero elements of the left-hand side matrix are shown in Figure 6.23. One can see that equation (6.100) does not lead to a banded matrix system as in one-dimensional cases. For large number of elements, however, most of the matrix elements are zero and sparsity of the system could be leveraged to speed up the solution algorithm.

The solution field and its comparison with the exact solution are shown in Figure 6.24. By using only 11 interior points in a two-dimensional domain the finite element method has reasonably well predicted the solution to the Poisson equation. The grid convergence of the solution is established by repeating this procedure using 448 elements.

Simple partial differential equations, such as the one described in this example, can be solved conveniently using widely available packages such as MATLAB's PDE toolbox without the requirement of programing a code to compute finite element matrices. For example, MATLAB's **pdetool** command provides a graphical interface through which a user can define a two-dimensional geometry using a combination of drawing tools and inputing the coordinates of boundary nodes. After the geometry is defined, the boundary condition for each edge can be selected from a menu. The user can specify inhomogeneous Neumann, Dirichlet or mixed boundary conditions. In another menu, the user can select the partial differential equation

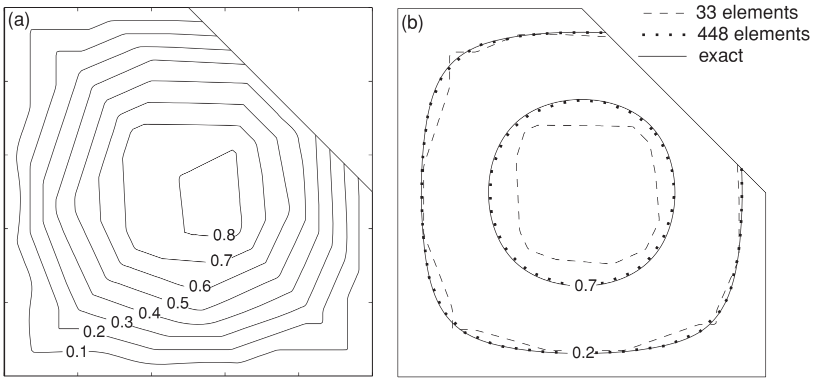

Figure 6.24 (a) Finite element solution to equation (6.99) using 33 elements. (b) Two contours of the solutions using 33 and 448 elements in comparison with the exact solution.

to be solved from a list of canonical elliptic PDE's. The mesh generation is done automatically; however, the user can specify parameters such as the maximum mesh size and growth rate to control the mesh. After these inputs are provided to Matlab, the PDE toolbox will use its own routines to form the stiffness and mass matrices and the solution will be computed automatically.

EXAMPLE 6.17

In this example, we use Matlab's *pdetool* to solve the heat equation in a complex geometry. Consider the steady heat equation

$$\nabla^2 u = 0,$$

in the domain shown in Figure 6.25 with an interior and an exterior boundary. The interior boundary is specified in polar coordinates (r, θ)[*]

$$r = 0.3 + 0.1\sin(\theta) + 0.15\sin(5\theta),$$

with the Dirichlet boundary condition $u = 1$, and the exterior boundary

$$r = 1 + 0.2\cos(\theta) + 0.15\sin(4\theta),$$

has the homogeneous Dirichlet condition, $u = 0$. Both boundaries are discretized using 100 edge elements as shown in Figure 6.25. A small Matlab program was written to compute these coordinates and then this program was read as a macro using Matlab's *pdetool*. The default-generated mesh, with 1676 triangular elements, is shown in the figure together with the contour plots of the solution.

[*] Orszag, S. A. 1980 J. Comp. phys. 37, 70–92.

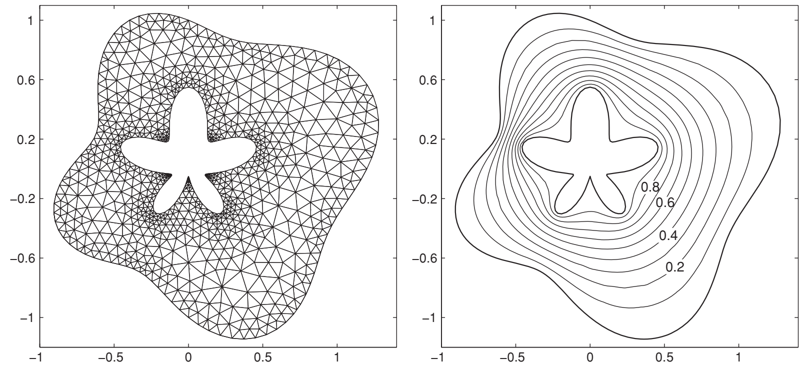

Figure 6.25 A MATLAB-generated mesh for problem of Example 6.4 and contours of the finite element solution.

EXERCISES

1. Show that the Fourier coefficients of the discrete convolution sum

$$c_j = \sum_{n=0}^{N-1} f_n g_{j-n} = (f * g)_j$$

are given by

$$\hat{c}_k = N \hat{f}_k \hat{g}_k.$$

2. Consider the triple product defined by

$$B_{mn} = \sum_{j=0}^{N-1} u_j u_{j+m} u_{j+n}.$$

Show that the bi-spectrum, $\hat{B}_{k_1 k_2}$, the two-dimensional Fourier coefficients of B_{mn} are given by

$$\hat{B}_{k_1 k_2} = N \hat{u}_{k_1} \hat{u}_{k_2} \hat{u}^*_{(k_1+k_2)}.$$

3. *Aliasing.*
   (a) Compute the Fourier transform of the product $y_1 y_2$ using 32 grid points in the interval $(0, 2\pi)$ and discuss any resulting aliasing error.

$$y_1(x) = \sin(2x) + 0.1 \sin(15x)$$
$$y_2(x) = \sin(2x) + 0.1 \cos(15x)$$

   (b) Compute the Fourier transform of
      (i) $y(x)\frac{dy(x)}{dx}$
      (ii) $\frac{d}{dx}\left(\frac{y^2(x)}{2}\right)$

   where

$$y(x) = \sin(2x) + 0.01 \sin(15x)$$

   and show that the difference is due to aliasing. Note that analytically they are equal.

4. The discrete cosine series is defined by

$$f_j = \sum_{k=0}^{N} a_k \cos(kx_j) \quad j = 0, 1, 2, \ldots, N,$$

where $x_j = \pi j/N$. Prove that the coefficients of the series are given by

$$a_k = \frac{2}{N}\frac{1}{c_k}\sum_{j=0}^{N}\frac{1}{c_j} f_j \cos(kx_j) \quad k = 0, 1, 2, \ldots, N,$$

where

$$c_j = \begin{cases} 2 & j = 0, N \\ 1 & \text{otherwise.} \end{cases}$$

5. Given $H(x) = f(x)g(x)$, express the discrete cosine transform of H in terms of the discrete cosine transforms of f and g.

6. Use an FFT routine to compute the Fourier coefficients of

$$f(x) = \cos\frac{n\pi x}{L} \quad 0 \le x < L,$$

with $N = 8$, $L = 7$, and $n = 2, 3$. Use an FFT routine to compute the inverse transform of the coefficients to verify that the original data are recovered.

7. Compute the Fourier coefficients using FFT of

$$f(x) = \cos(2x) + \frac{1}{2}\cos(4x) + \frac{1}{6}\cos(12x) \quad 0 \le x < 2\pi,$$

for $N = 8$, 16, 32, and 64.

8. Consider the function $f(x)$ defined as follows:

$$f(x) = \begin{cases} e^{-x} & \text{for } 0 \le x < L \\ 0 & \text{otherwise.} \end{cases}$$

Obtain the Fourier coefficients using FFT. Discuss the importance of L and N. In addition, compare the computational time of using the fast Fourier transform to the computational time of the brute-force ($O(N^2)$) Fourier transform. (Graph the computational time on a log–log plot.) To get good timing data, you may have to call the FFT routine several times for each value of N.

9. Differentiate the following functions using FFT and second-order finite differences. Show your results, including errors, graphically. Use $N = 16$, 32.
   (a) $f(x) = \sin 3x + 3\cos 6x \quad 0 \le x < 2\pi$.
   (b) $f(x) = 6x - x^2 \quad 0 \le x < 2\pi$.

10. Consider the ODE

$$f'' - f' - 2f = 2 + 6\sin 6x - 38\cos 6x,$$

defined on $0 \le x \le 2\pi$ with periodic boundary conditions. Solve it using FFT and a second-order central finite difference scheme with $N = 16$, 64. Compare the results. For the finite difference calculations you may use $f(0) = f(2\pi) = 0$.

11. Discuss how to solve the following equation using the Fourier spectral method:

$$u_{xx} + (\sin x)u_x = -(\sin x + \sin 2x)e^{\cos x},$$

on $0 \le x \le 2\pi$ with periodic boundary conditions. Derive a set of algebraic equations for the Fourier coefficients. Be sure to carefully consider the boundary conditions and verify that the resulting matrix equation is non-singular.

12. Write a program that computes the Chebyshev transform of an arbitrary function, and test your program by transforming $1, x^3$, and x^6. Use your program to compute and plot the Chebyshev expansion coefficients for

(a) $f(x) = xe^{-x/2}$.

(b) $f(x) = \begin{cases} +1 & -1 \le x \le 0 \\ -1 & 0 < x \le 1. \end{cases}$

Use $N = 4$, 8, and 16.

13. Write a program to calculate the derivative of an arbitrary function using the Chebyshev transform. Test your program by differentiating polynomials and use it to differentiate the functions in Exercise 11. Take $N = 4$, 8, 16, 32 and compare to the exact answers.

14. Use mathematical induction to show that

$$b_m = \frac{2}{c_m} \sum_{\substack{p=m+1 \\ p+m \text{ odd}}}^{N} p a_p,$$

where a_p are the Chebyshev coefficients of some function $f(x)$ and b_m are the Chebyshev coefficients of $f'(x)$.

15. Use the Chebyshev transform program in Exercise 11 to calculate the integral of an arbitrary function. Test your program by integrating polynomials and use it to integrate the functions in Exercise 11. Take $N = 4$, 8,16, 32 and compare to the exact values.

16. Use the matrix form of the Chebyshev collocation derivative to differentiate $f(x) = x^5$ for $-1 \le x \le 1$. Compare to the exact answer.

17. Solve the convection equation

$$u_t + 2u_x = 0,$$

for $u(x, t)$ on the domain $-1 \le x \le 1$ subject to the boundary and initial conditions

$$u(-1, t) = \sin \pi t \quad u(x, 0) = 0.$$

The exact solution is

$$u = \begin{cases} 0 & x \ge -1 + 2t \\ \sin \pi (t - \frac{x+1}{2}) & -1 \le x \le -1 + 2t. \end{cases}$$

Use the discrete Chebyshev transform and second-order finite difference methods. Plot the solution at several t. Plot the rms of the error at $t = 7/8$ versus N. Compare the accuracy of the two methods.

18. Show that the interior $N - 1$ Chebyshev grid points given by (6.35) are the zeros of T'_N which is a polynomial of degree $N - 1$.
19. In this exercise we will go through the key steps leading to expression (6.47) for the elements of the Chebyshev derivative matrix. We will begin by using the results from Exercise 10 of Chapter 1. Let $\phi_{N+1}(x)$ be a polynomial of degree $N + 1$:

$$\phi_{N+1}(x) = \prod_{l=0}^{N} (x - x_l).$$

Show that the matrix elements obtained in Exercise 10 of Chapter 1 can be recast in the following form:

$$d_{jk} = \frac{\phi'_{N+1}(x_j)}{\phi'_{N+1}(x_k)(x_j - x_k)} \quad j \neq k. \tag{1}$$

If $x_0 = 1, x_N = -1$, and the remaining x_j are the zeros of the polynomial $Q_{N-1}(x)$, then

$$\phi_{N+1}(x) = (1 - x^2)\, Q_{N-1}. \tag{2}$$

Show that

$$d_{jk} = \frac{\left(1 - x_j^2\right) Q'_{N-1}(x_j)}{\left(1 - x_k^2\right) Q'_{N-1}\,(x_k)\,(x_j - x_k)} \quad j \neq k \text{ and } j, k \neq 0, N \tag{3}$$

For $j = k$, again referring to Exercise 10 of Chapter 1, we want to evaluate

$$d_{jj} = \sum_{\substack{l=0 \\ l \neq j}}^{N} \frac{1}{x_j - x_l}.$$

Let $\phi_{N+1}(x) = (x - x_j)g(x)$, and let $x_k(k = 0, 1, 2, \ldots, N$ except for $k = j)$ be the zeros of g. Show that

$$\frac{g'(x_j)}{g(x_j)} = \frac{\phi''_{N+1}(x_j)}{2\phi'_{N+1}(x_j)}, \tag{4}$$

and hence

$$d_{jj} = \frac{\phi''_{N+1}(x_j)}{2\phi'_{N+1}(x_j)}.$$

For Chebyshev polynomials, $x_0 = -1, x_N = 1$, and the remaining x_j are the zeros of T'_N (see Exercise 17). Using the fact that Q_{N-1} in (2) is simply equal to T'_N, you should now be able to derive the matrix elements given in (6.47), i.e.,

$$d_{jk} = \begin{cases} \frac{c_j(-1)^{j+k}}{c_k(x_j - x_k)} & j \neq k \\ \frac{-x_j}{2(1-x_j^2)} & j = k, \quad j \neq 0, N \\ \frac{2N^2+1}{6} & j = k = 0 \\ -\frac{2N^2+1}{6} & j = k = N. \end{cases}$$

20. From the definition $C_{ij} = \int_0^1 \phi_i \phi_j \, dx$ obtain the C_{ij} matrix for linear basis functions and verify your results by comparing with (6.64).

21. Use an appropriate discretization in time for (6.79) and derive a full discretized scheme for constant coefficient convection equation. How would you use Runge–Kutta-type schemes for time integration?

22. (a) Derivation of finite element formulation for the convection equation (6.76) presented in Section 6.6.4 involves integration by parts (see (6.77)). Show that derivation without integration by parts results in the same finite element formulation. (b) For the case $c = 1$ use 16 elements to discretize the domain and obtain the finite element formulation. Show that if no boundary condition is used, this solution can become unbounded in time.

23. Compare the finite element formulation of the heat equation (6.80) with the fourth-order Padé formulation. What is the spatial accuracy of the finite element formulation with linear elements for this problem and how does it compare with that for the convection equation (6.76)?

24. (a) In an axisymmetric configuration the heat equation is

$$\frac{\partial u}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)$$

defined in the domain $0.5 \leq r \leq 1$ with the boundary conditions $\frac{\partial u}{\partial r} = 0$ at $r = 0.5$ and $u = 0$ at $r = 1$. Develop a finite element formulation to solve this problem. For the initial condition $u(r, t = 0) = 1$ use your formulation and obtain a numerical solution to the system.

(b) Use Matlab's PDE toolbox to solve this problem in the two-dimensional domain with triangular mesh. Compare your result with that of part (a) at time $t = 0.1$.

25. Consider the Poisson equation

$$\nabla^2 u + f = 0$$

on the triangular domain shown in Figure 6.26. The source term f is taken to be constant over the entire domain. Homogeneous Neumann boundary conditions are imposed on two sides of the triangle, while a Dirichlet boundary condition is imposed on the third. The domain is discretized into six nodes and four equal elements, each one being an isosceles right triangle with height 1/2 and length 1/2. Use the finite element method to formulate the problem and obtain the solutions for the six nodal values.

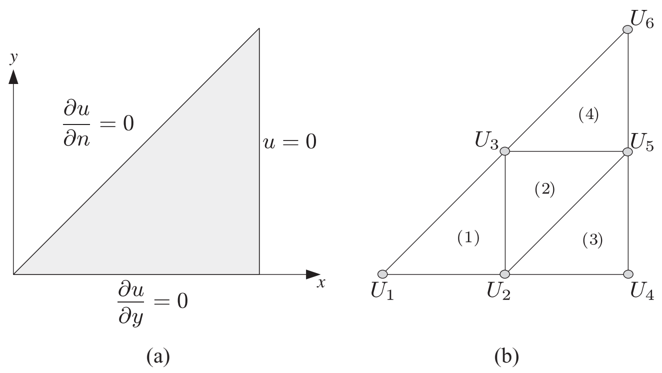
y
$\frac{\partial u}{\partial n} = 0$
$u = 0$
x
$\frac{\partial u}{\partial y} = 0$
(a)
U_6
(4)
U_3
U_5
(2)
(1)
(3)
U_1
U_2
U_4
(b)

Figure 6.26 (a) A schematic of the geometry and boundary conditions used in Exercise 25, and (b) triangular elements used to discretize the geometry.

FURTHER READING

Bracewell, R. N. *The Fourier Transform and Its Applications*, Second Edition. McGraw-Hill, 1986.

Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A. *Spectral Methods in Fluid Dynamics*. Springer-Verlag, 1988.

Dahlquist, G., and Björck, Å. *Numerical Methods*. Prentice-Hall, 1974, Chapter 9.

Gottlieb, D., and Orszag, Steven A. *Numerical Analysis of Spectral Methods: Theory and Applications*. Society for Industrial and Applied Mathematics (SIAM), 1977.

Hirsch, C. *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics*. Elsevier Butterworth-Heinemann, 2007.

Hockney, R. W., and Eastwood, J. W. *Computer Simulation Using Particles*: IOP (Inst. of Physics) Publishing Ltd. 1988, reprinted 1994.

Orszag, S. A. Spectral Methods for Problems in Complex Geometries. *J. Comp. Phys. 37*, 70–92.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes: The Art of Scientific Computing*, Third Edition. Cambridge University Press, 2007, Chapters 12 and 13.

Snyder, M. A. *Chebyshev Methods in Numerical Approximation*. Prentice-Hall, 1966, Chapters 1, 2, and 3.

Trefethen, L. N. *Spectral Methods in Matlab*. Society for Industrial and Applied Mathematics, 2005.

Zienkiewicz, O. C., Taylor, R. L., and Zhu, J. Z. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier Butterworth-Heinemann, 2005.