Chapter 2
A First Example

In Chap. 1, we explained the fundamental philosophy that underpins the finite element method—that is, the region on which a differential equation is defined is partitioned into smaller regions known as elements, and the solution on each of these elements is approximated using a low-order polynomial function. In this chapter, with the aid of a simple example, we illustrate how this may be done. This overview will require the definition of some terms that the reader may not be familiar with, as well as a few technical details. We will, however, undertake to keep these definitions to a minimum and will focus on the underlying strategy of applying the finite element method without getting bogged down by these technical details. As a consequence, we will inevitably skate over some mathematical rigour, but will make a note to return to these points in later chapters.

2.1 Some Brief Mathematical Preliminaries

We make a few comments on notation before we begin our overview of the finite element method. Throughout this book, we use the convention that a solution of a differential equation that is represented by a lowercase letter, for example $u(x)$, is the exact, or analytic, solution of the differential equation. The corresponding uppercase letter, $U(x)$ in this case, will represent the finite element approximation to the exact solution.

We will frequently make use of vectors and matrices, and adopt the following conventions.

1. Vectors will be assumed to be column vectors, i.e. a vector with many rows but only one column, and will be denoted by bold font. Entry i of the vector $\mathbf{x}$ will be denoted by x_i.

J. Whiteley, *Finite Element Methods*, Mathematical Engineering,
DOI 10.1007/978-3-319-49971-0_2

2. Matrices will be represented by upper case letters typeset in italic font. The entry in row i and column j of the matrix A will be denoted by $A_{i,j}$.
3. The indexing of both vectors and matrices will start from 1. A vector $\mathbf{b}$ with N entries, for example, will have entries $b_1, b_2, \ldots, b_N$. This allows the finite element algorithms to be written in a way that facilitates the writing of software in programming languages or environments such as MATLAB, Octave or Fortran, where array indexing starts from 1 (known as "one-based indexing"). Readers using programming languages such as C, C++ and Python, where the indexing of arrays starts from 0 (known as "zero-based indexing") will need to adapt the algorithms to take account of this feature of the programming language being used.

2.2 A Model Differential Equation

We will use a simple boundary value problem to demonstrate the application of the finite element method. A boundary value problem comprises both a differential equation and boundary conditions. In this chapter, we will consider only *Dirichlet boundary conditions*, where the value of the unknown function $u(x)$ is specified on the boundary. More complex boundary conditions, such as those that include the derivative of $u(x)$ on the boundary, will be discussed in Chap. 3. Our model boundary problem comprises the differential equation

$$-\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 2, \qquad 0 < x < 1, \tag{2.1}$$

and the Dirichlet boundary conditions given by

$$u(0) = u(1) = 0. \tag{2.2}$$

The region $0 < x < 1$ on which the equation is defined is known as the *domain*. This simple example, with solution $u(x) = x(1 - x)$, will now be used to exhibit the underlying principles of calculating the finite element solution of a differential equation.

2.3 The Weak Formulation

Let $u(x)$ be the solution of the model differential equation, Eq. (2.1), subject to the boundary conditions given by Eq. (2.2). The function $u(x)$ is known as the *classical solution* of this differential equation, and specification of $u(x)$ through Eqs. (2.1) and (2.2) is known as the *classical formulation*.

We now introduce the *weak formulation* of the model problem, a concept that may be unfamiliar to some readers. In this chapter, we limit ourselves to illustrating the weak formulation of a differential equation by example, rather than giving a watertight definition through precise mathematical statements. A more complete and rigorous treatment of this material will be presented in Sect. 3.2.

To derive the weak formulation of our model problem, we first define $v(x)$ to be a continuous function that satisfies

$$v(0) = v(1) = 0, \tag{2.3}$$

so that $v(x) = 0$ at the values of x where we apply Dirichlet boundary conditions. We now multiply the differential equation, given by Eq. (2.1), by $v(x)$, and integrate the resulting product over the interval $0 < x < 1$ on which the differential equation is defined, giving

$$\int_0^1 -\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} v(x)\, \mathrm{d}x = \int_0^1 2v(x)\, \mathrm{d}x.$$

We now apply integration by parts to the left-hand side of the equation above to obtain

$$\left[-\frac{\mathrm{d}u}{\mathrm{d}x} v(x)\right]_0^1 + \int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\, \mathrm{d}x = \int_0^1 2v(x)\, \mathrm{d}x,$$

which may be written as

$$-\left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x=1} v(1) + \left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x=0} v(0) + \int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\, \mathrm{d}x = \int_0^1 2v(x)\, \mathrm{d}x, \tag{2.4}$$

where

$$\left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x=1} \quad \text{and} \quad \left.\frac{\mathrm{d}u}{\mathrm{d}x}\right|_{x=0}$$

denote the value taken by $\frac{\mathrm{d}u}{\mathrm{d}x}$ at $x = 1$ and $x = 0$, respectively. Remembering our condition on $v(x)$, given by Eq. (2.3), we see that the first two terms in the sum on the left-hand side of Eq. (2.4) are zero, and we may write this equation as

$$\int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\, \mathrm{d}x = \int_0^1 2v(x)\, \mathrm{d}x.$$

Assuming that all derivatives and integrals we have used exist, we may now specify the *weak formulation* of our model differential equation by:

find $u(x)$ that satisfies the Dirichlet boundary conditions given by Eq. (2.2) and is such that

$$\int_0^1 \frac{\mathrm{d}u}{\mathrm{d}x}\frac{\mathrm{d}v}{\mathrm{d}x}\,\mathrm{d}x = \int_0^1 2v(x)\,\mathrm{d}x, \tag{2.5}$$

for all continuous functions $v(x)$ that satisfy $v(0) = v(1) = 0$.

The solution $u(x)$ given by Eq. (2.5) is known as the *weak solution*. The functions $v(x)$ that appear in Eq. (2.5) are known as *test functions*.

The concept of weak solutions will be revisited in the next chapter, providing some mathematical details that have been omitted above. For now, the reader need only know what has been derived above, namely that a classical solution of the differential equation and boundary conditions given by Eqs. (2.1) and (2.2) is also a weak solution that satisfies Eq. (2.5).

2.4 Elements and Nodes

We now begin to lay out some of the ingredients required by the finite element method. We have already explained that we need to partition the domain on which the differential equation is specified into smaller regions. Our model differential equation, Eq. (2.1), is defined on the domain $0 < x < 1$, and we now partition this domain into N intervals of equal length $h = \frac{1}{N}$. If we define

$$x_i = \frac{i-1}{N}, \qquad i = 1, 2, \ldots, N+1, \tag{2.6}$$

then these intervals are the regions $x_k \leq x \leq x_{k+1}$, for $k = 1, 2, \ldots, N$. We refer to these intervals as *elements*. The points x_i that define the element boundaries are known as *nodes*. We also define element k to be the element that occupies $x_k \leq x \leq x_{k+1}$, for $k = 1, 2, \ldots, N$. The elements and nodes are collectively known as the *computational mesh*, the *finite element mesh* or, more simply, the *mesh*. The nodes and elements for the case where $N = 5$ are shown in Fig. 2.1.

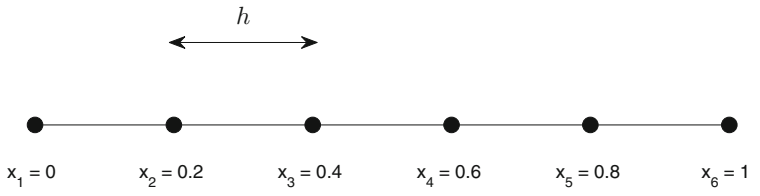

Fig. 2.1 The nodes and elements in a finite element mesh when the interval $0 < x < 1$ is partitioned into five equally sized elements

The mesh we have described above contains elements that are equally sized. We will see in the next chapter that it is straightforward to generalise the definition of the mesh so that it contains elements that are of different sizes.

2.5 Basis Functions

In this chapter, we will use the finite element method to calculate a linear approximation to the solution on each element, that is continuous across element boundaries. In the previous section, we partitioned the domain $0 < x < 1$ into N equally sized elements, each of length $h = \frac{1}{N}$. Note that this implies that

$$x_{j+1} - x_j = h, \qquad j = 1, 2, \ldots, N. \tag{2.7}$$

We now specify some functions that will be required when calculating the finite element solution. Using the property of h given by Eq. (2.7), the functions $\phi_j(x)$, for $j = 1, 2, \ldots, N+1$, are defined by

$$\phi_1(x) = \begin{cases} (x_2 - x)/h, & x_1 \le x \le x_2, \\ 0, & \text{otherwise}, \end{cases} \tag{2.8}$$

$$\phi_j(x) = \begin{cases} (x - x_{j-1})/h, & x_{j-1} \le x \le x_j, \\ (x_{j+1} - x)/h, & x_j \le x \le x_{j+1}, \\ 0, & \text{otherwise}, \end{cases} \qquad j = 2, 3, \ldots, N, \tag{2.9}$$

$$\phi_{N+1}(x) = \begin{cases} (x - x_N)/h, & x_N \le x \le x_{N+1}, \\ 0, & \text{otherwise}. \end{cases} \tag{2.10}$$

These functions are illustrated in Fig. 2.2. For reasons that will soon become clear, we refer to these functions as *basis functions*.

The functions $\phi_j(x), j = 1, 2, \ldots, N+1$, defined by Eqs. (2.8)–(2.10) have three properties that are useful when calculating the finite element solution: (i) they are linear functions on each element; (ii) they are continuous functions; and (iii) they satisfy, for $i = 1, 2, \ldots, N+1$, the condition

$$\phi_j(x_i) = \begin{cases} 1, & i = j, \\ 0, & i \ne j. \end{cases} \tag{2.11}$$

These properties may easily be verified using the definition of basis functions given by Eqs. (2.8)–(2.10), and the plots of these functions in Fig. 2.2. As a consequence, the function $\phi_j(x)$ takes the value 1 at the node where $x = x_j$, and the value 0 at all other nodes. This is a very important property of basis functions, as we will see in later chapters when using more general basis functions to calculate the finite element approximation of more complex differential equations.

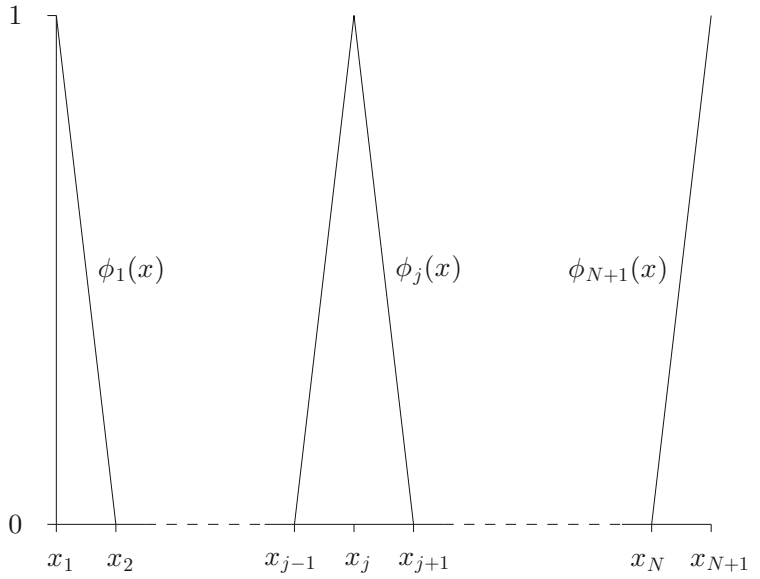

Fig. 2.2 The basis functions defined by Eqs. (2.8)–(2.10)

2.6 The Finite Element Solution

We will now use the basis functions $\phi_j(x), j = 1, 2, \ldots, N+1$, defined by Eqs. (2.8)–(2.10), to generate a finite element solution of our model differential equation that is a linear approximation to the solution on each element. We claim that we can write the finite element solution, $U(x)$, as a linear sum of these basis functions:

$$U(x) = \sum_{j=1}^{N+1} U_j \phi_j(x), \tag{2.12}$$

where the as yet unknown values $U_j, j = 1, 2, \ldots, N+1$, are to be determined. Our claim that the finite element solution, $U(x)$, is of the form given by Eq. (2.12) is straightforward to verify. As the basis functions $\phi_j(x)$ are linear functions on each element and continuous, the sum given by Eq. (2.12) also satisfies these properties and is therefore a suitable candidate for the finite element solution. Finally we note that, on using the property of basis functions given by Eq. (2.11), we may write

$$\begin{aligned} U(x_i) &= \sum_{j=1}^{N+1} U_j \phi_j(x_i) \\ &= U_i, \end{aligned} \tag{2.13}$$

and so U_i is the finite element approximation to the solution at the node where $x = x_i$.

We now move on to describe how to determine the values U_i, $i = 1, 2, \ldots, N+1$, that complete the definition of the finite element solution given by Eq. (2.12).

2.7 Algebraic Equations Satisfied by the Finite Element Solution

Having prescribed the functional form for the finite element solution by Eq. (2.12), we now generate a system of algebraic equations satisfied by the coefficients U_j, $j = 1, 2, \ldots, N+1$. This is achieved by modifying the weak formulation of the model differential equation, given by Eq. (2.5), to specify the finite element solution $U(x)$ by:

> find $U(x)$ that satisfies the Dirichlet boundary conditions given by Eq. (2.2) and is such that
>
> $$\int_0^1 \frac{\mathrm{d}U}{\mathrm{d}x}\frac{\mathrm{d}\phi_i}{\mathrm{d}x}\,\mathrm{d}x = \int_0^1 2\phi_i(x)\,\mathrm{d}x, \tag{2.14}$$
>
> for all basis functions $\phi_i(x)$, $i = 1, 2, \ldots, N+1$, that satisfy $\phi_i(0) = \phi_i(1) = 0$.

The statement above is nothing more than the weak formulation, Eq. (2.5), restated with two minor modifications. First, we replace the weak solution $u(x)$ by the finite element solution $U(x)$. Second we restrict the test functions to be the finite set of basis functions $\phi_i(x)$ that satisfy

$$\phi_i(0) = \phi_i(1) = 0.$$

We see, from the definitions of the basis functions given by Eqs. (2.8)–(2.10) and their illustration in Fig. 2.2, that this condition on $\phi_i(x)$ is satisfied by all basis functions except $\phi_1(x)$ and $\phi_{N+1}(x)$. We therefore have $N-1$ functions $\phi_2(x), \phi_3(x), \ldots, \phi_N(x)$ that satisfy this condition.

We now use Eq. (2.14) to derive a system of algebraic equations, satisfied by the $N+1$ unknown values $U_1, U_2, \ldots, U_{N+1}$, that specify the finite element solution given by Eq. (2.12). As we have $N+1$ unknown values, we require a system of $N+1$ algebraic equations to determine these values.

The first condition in the specification of the finite element solution by Eq. (2.14) is that $U(x)$ satisfies the Dirichlet boundary conditions given by Eq. (2.2). These boundary conditions are applied at the node where $x = x_1$ and the node where $x = x_{N+1}$ and may be written as

$$U(x_1) = 0, \qquad U(x_{N+1}) = 0.$$

Using Eq. (2.13), we see that these Dirichlet boundary conditions are satisfied provided that

$$U_1 = 0, \tag{2.15}$$

$$U_{N+1} = 0, \tag{2.16}$$

and these are our first two algebraic equations.

We have already noted that the $N-1$ basis functions ϕ_i, where $i = 2, 3, \ldots, N$, satisfy the condition on test functions given in Eq. (2.14), i.e. that $\phi_i(0) = \phi_i(1) = 0$. The remaining $N-1$ algebraic equations result from substituting these functions into the integral equation given in Eq. (2.14). Noting that the definition of the finite element solution given by Eq. (2.12) may be differentiated to give

$$\frac{\mathrm{d}U}{\mathrm{d}x} = \sum_{j=1}^{N+1} U_j \frac{\mathrm{d}\phi_j}{\mathrm{d}x},$$

substitution of $\frac{\mathrm{d}U}{\mathrm{d}x}$ into the integral equation given by Eq. (2.14) yields

$$\int_0^1 \left(\sum_{j=1}^{N+1} U_j \frac{\mathrm{d}\phi_j}{\mathrm{d}x} \right) \frac{\mathrm{d}\phi_i}{\mathrm{d}x} \, \mathrm{d}x = \int_0^1 2\phi_i(x) \, \mathrm{d}x, \qquad i = 2, 3, \ldots, N,$$

which, after a little manipulation, becomes

$$\sum_{j=1}^{N+1} \left(\int_0^1 \frac{\mathrm{d}\phi_i}{\mathrm{d}x} \frac{\mathrm{d}\phi_j}{\mathrm{d}x} \, \mathrm{d}x \right) U_j = \int_0^1 2\phi_i(x) \, \mathrm{d}x, \qquad i = 2, 3, \ldots, N. \tag{2.17}$$

This may be written as

$$\sum_{j=1}^{N+1} A_{i,j} U_j = b_i, \qquad i = 2, 3, \ldots, N, \tag{2.18}$$

where for $i = 2, 3, \ldots, N$,

$$A_{i,j} = \int_0^1 \frac{\mathrm{d}\phi_i}{\mathrm{d}x} \frac{\mathrm{d}\phi_j}{\mathrm{d}x} \, \mathrm{d}x, \qquad j = 1, 2, 3, \ldots, N+1, \tag{2.19}$$

$$b_i = \int_0^1 2\phi_i(x) \, \mathrm{d}x. \tag{2.20}$$

We may combine Eqs. (2.15), (2.16) and (2.18) into the $(N+1) \times (N+1)$ linear system

$$A\mathbf{U} = \mathbf{b}. \tag{2.21}$$

The entries of A and $\mathbf{b}$ in row i, where $i = 2, 3, \ldots, N$, are given by Eqs. (2.19) and (2.20). Using Eqs. (2.15) and (2.16), we see that the entries of A and $\mathbf{b}$ in rows 1 and $N+1$ are given by

$$A_{1,j} = \begin{cases} 1, & j = 1, \\ 0, & j \neq 1, \end{cases} \tag{2.22}$$

$$A_{(N+1),j} = \begin{cases} 1, & j = N+1, \\ 0, & j \neq N+1, \end{cases} \tag{2.23}$$

$$b_1 = 0, \tag{2.24}$$

$$b_{N+1} = 0. \tag{2.25}$$

The algebraic equations that comprise the linear system given by Eq. (2.21) fall into two categories. The first category is equations such as Eqs. (2.15) and (2.16) that arise from demanding that the finite element solution satisfies all Dirichlet boundary conditions. The second category is equations such as Eq. (2.18) that arise from using a suitable basis function as the test function in the integral condition given by Eq. (2.14). This categorisation of equations will be a feature of all the systems of algebraic equations that we derive in later chapters.

Having derived the linear system of algebraic equations that determines the values $U_1, U_2, \ldots, U_{N+1}$, we now introduce a flexible technique for the practical computation of the entries of the matrix A and the vector $\mathbf{b}$.

2.8 Assembling the Algebraic Equations

Most entries of the matrix A and vector $\mathbf{b}$ that appear in the linear system given by Eq. (2.21) are of the form specified by Eqs. (2.19) and (2.20) and are defined by integrals over the whole domain on which the differential equation is specified. We will now demonstrate that, from a computational implementation viewpoint, it is convenient to compute these entries by summing the contributions from individual elements. That is, for $i = 2, 3, \ldots, N$, we write Eqs. (2.19) and (2.20) as the sum of integrals over individual elements:

$$A_{i,j} = \sum_{k=1}^{N} \int_{x_k}^{x_{k+1}} \frac{\mathrm{d}\phi_i}{\mathrm{d}x} \frac{\mathrm{d}\phi_j}{\mathrm{d}x} \, \mathrm{d}x, \qquad j = 1, 2, \ldots, N+1, \tag{2.26}$$

$$b_i = \sum_{k=1}^{N} \int_{x_k}^{x_{k+1}} 2\phi_i(x) \, \mathrm{d}x. \tag{2.27}$$

Using Eq. (2.26) we see that the contribution to entry $A_{i,j}$ from integrating over the element that occupies $x_k \leq x \leq x_{k+1}$ is given by

$$\int_{x_k}^{x_{k+1}} \frac{\mathrm{d}\phi_i}{\mathrm{d}x} \frac{\mathrm{d}\phi_j}{\mathrm{d}x} \, \mathrm{d}x. \tag{2.28}$$

From the definition of the basis functions, given by Eqs. (2.8)–(2.10) and illustrated in Fig. 2.2, we see that $\phi_k(x)$ and $\phi_{k+1}(x)$ are the only basis functions that are nonzero on the element occupying $x_k \leq x \leq x_{k+1}$: all other basis functions are identically zero on this element. Hence, the integral above, over $x_k \leq x \leq x_{k+1}$, yields nonzero contributions only to the entries $A_{k,k}, A_{k,k+1}, A_{k+1,k}, A_{k+1,k+1}$ of A. Motivated by this observation, we calculate only these nonzero contributions when integrating over this element and store these contributions in a 2×2 matrix called $A^{(k)}_{\text{local}}$. The relationship between the location of the entries of $A^{(k)}_{\text{local}}$ and the location of the entries of A that they contribute to is given in Table 2.1.

The entries of $A^{(k)}_{\text{local}}$ are known as the *local contributions* to the matrix A from element k, and A is often referred to as the *global matrix*. We may compute A efficiently by looping over all elements, calculating only the nonzero local contributions from each element, before adding the local contributions to the appropriate entries of the global matrix A. Having calculated the entries of A that are given by Eq. (2.19), we complete the specification of the matrix by setting the remaining entries, i.e. those defined by Eqs. (2.22) and (2.23).

The entries of $\mathbf{b}$ that are given by Eq. (2.27) may be calculated in a similar manner to that used for the entries of A that are given by Eq. (2.26). The contribution to b_i, $i = 2, 3, \ldots, N$, from integrating over the element that occupies $x_k \leq x \leq x_{k+1}$ is given by

Table 2.1 The relationship between the location of the entries of $A^{(k)}_{\text{local}}$ and the location of the entries of A that they contribute to

Entry in $A^{(k)}_{\text{local}}$		Entry in A	
Row	Column	Row	Column
1	1	k	k
1	2	k	$k+1$
2	1	$k+1$	k
2	2	$k+1$	$k+1$

Table 2.2 The relationship between the location of the entries of $\mathbf{b}^{(k)}_{\text{local}}$ and the location of the entries of $\mathbf{b}$ that they contribute to

Entry in $\mathbf{b}^{(k)}_{\text{local}}$	Entry in $\mathbf{b}$
1	k
2	$k+1$

$$\int_{x_k}^{x_{k+1}} 2\phi_i(x)\,\mathrm{d}x. \tag{2.29}$$

As $\phi_k(x)$ and $\phi_{k+1}(x)$ are the only basis functions that are nonzero inside the element that occupies $x_k \leq x \leq x_{k+1}$, the only nonzero contributions to $\mathbf{b}$ from this element are to the entries b_k and b_{k+1}. We again calculate only these nonzero contributions and store them in a vector of length 2 called $\mathbf{b}^{(k)}_{\text{local}}$, known as the *local contribution* to the global vector $\mathbf{b}$. The relationship between the location of the entries of $\mathbf{b}^{(k)}_{\text{local}}$ and the location of the entries of $\mathbf{b}$ that they contribute to is given in Table 2.2. We may then calculate the global vector $\mathbf{b}$ by looping over all elements, calculating only the nonzero contributions to Eq. (2.27) from each element, and adding these contributions into $\mathbf{b}$. We then use Eqs. (2.24) and (2.25) to set the other entries.

2.8.1 Calculating the Local Contributions

We explained above that the entries of the global matrix A and global vector $\mathbf{b}$ that are given by Eqs. (2.19) and (2.20) should be assembled by summing the local contributions to these entries from each element. We now explain how these local contributions may be calculated.

Using the mapping between the entries of $A^{(k)}_{\text{local}}$ and the entries of A, given by Table 2.1, and the definition of the contribution to entries of A from element k, given by Eq. (2.28), we see that the entries of $A^{(k)}_{\text{local}}$ are given by

$$A^{(k)}_{\text{local}} = \begin{pmatrix} \int_{x_k}^{x_{k+1}} \frac{\mathrm{d}\phi_k}{\mathrm{d}x}\frac{\mathrm{d}\phi_k}{\mathrm{d}x}\,\mathrm{d}x & \int_{x_k}^{x_{k+1}} \frac{\mathrm{d}\phi_k}{\mathrm{d}x}\frac{\mathrm{d}\phi_{k+1}}{\mathrm{d}x}\,\mathrm{d}x \\ \int_{x_k}^{x_{k+1}} \frac{\mathrm{d}\phi_{k+1}}{\mathrm{d}x}\frac{\mathrm{d}\phi_k}{\mathrm{d}x}\,\mathrm{d}x & \int_{x_k}^{x_{k+1}} \frac{\mathrm{d}\phi_{k+1}}{\mathrm{d}x}\frac{\mathrm{d}\phi_{k+1}}{\mathrm{d}x}\,\mathrm{d}x \end{pmatrix}. \tag{2.30}$$

Similarly, using Table 2.2 and Eq. (2.29), the entries of $\mathbf{b}^{(k)}_{\text{local}}$ may be written as

$$\mathbf{b}^{(k)}_{\text{local}} = \begin{pmatrix} \int_{x_k}^{x_{k+1}} 2\phi_k(x)\,\mathrm{d}x \\ \int_{x_k}^{x_{k+1}} 2\phi_{k+1}(x)\,\mathrm{d}x \end{pmatrix}. \tag{2.31}$$

The integrals required to calculate the entries of $A^{(k)}_{\text{local}}$ and $\mathbf{b}^{(k)}_{\text{local}}$ are sufficiently simple in this case that we may evaluate them analytically. Using Eqs. (2.8)–(2.10), we see that, for $x_k \leq x \leq x_{k+1}$,

$$\phi_k(x) = \frac{x_{k+1} - x}{h},$$
$$\phi_{k+1}(x) = \frac{x - x_k}{h},$$

and so the derivatives of these functions take the constant values given by

$$\frac{\mathrm{d}\phi_k}{\mathrm{d}x} = -\frac{1}{h},$$
$$\frac{\mathrm{d}\phi_{k+1}}{\mathrm{d}x} = \frac{1}{h}.$$

Elementary integration allows us to deduce that the entries of $A^{(k)}_{\text{local}}$ and $\mathbf{b}^{(k)}_{\text{local}}$ are given by

$$A^{(k)}_{\text{local}} = \begin{pmatrix} \frac{1}{h} & -\frac{1}{h} \\ -\frac{1}{h} & \frac{1}{h} \end{pmatrix},$$
$$\mathbf{b}^{(k)}_{\text{local}} = \begin{pmatrix} h \\ h \end{pmatrix}.$$

2.8.2 *Assembling the Global Matrix*

We have proposed calculating the entries of the global matrix A and global vector $\mathbf{b}$ that are of the form given by Eqs. (2.19) and (2.20), by looping over all elements in the mesh, calculating the nonzero local contributions $A^{(k)}_{\text{local}}$ and $\mathbf{b}^{(k)}_{\text{local}}$ from each element, and using these contributions to increment the appropriate entries of A and $\mathbf{b}$, given by Tables 2.1 and 2.2.

Readers may have spotted a potential flaw with this approach for calculating A and $\mathbf{b}$. The expressions for $A_{i,j}$ and b_i given by Eqs. (2.19) and (2.20) are only valid for $i = 2, 3, \ldots, N$, with the entries for rows 1 and $N+1$ given by Eqs. (2.22)–(2.25). When calculating the local contributions from the element that occupies $x_1 \leq x \leq x_2$, we evaluate contributions to both row 1 and row 2 of A and $\mathbf{b}$. The entries of row 1 of A and $\mathbf{b}$ do not, however, fall into the pattern given by Eqs. (2.19) and (2.20), although the contribution to row 2 of both A and $\mathbf{b}$ from this element is correct. Similarly, when calculating the local contribution from the element occupying $x_N \leq x \leq x_{N+1}$ we calculate contributions to row $N + 1$ of A and $\mathbf{b}$ that are not of the correct form. We could avoid adding these incorrect entries into rows 1 and $N + 1$ by performing a check, before adding any local contributions into the global matrix and vector, to ensure we do not add any entries into rows 1 and $N + 1$. There is nothing wrong with this approach. However, to promote clear, modular code—especially for the more complex problems encountered later in this book—it is easier to add the local contributions into rows 1 and $N + 1$ and then to overwrite these erroneous entries afterwards with the correct values, given by Eqs. (2.22)–(2.25).

Having assembled the matrix A and the vector $\mathbf{b}$, most of the work required to calculate the finite element solution given by Eq. (2.12) has been done. All that remains is to solve the linear system given by Eq. (2.21) to calculate $\mathbf{U}$, before substituting the entries of $\mathbf{U}$ into Eq. (2.12). We now summarise the steps we have taken when applying the finite element method and then present an exemplar computational implementation.

2.9 A Summary of the Steps Required

The aim of this chapter has been to give an overview of the finite element method by illustrating its application to an example boundary value problem. We now give a summary of the steps required when applying the finite element method. We will see in later chapters that this summary may be used as a guide for the application of the finite element method to more general differential equations.

1. Derive the weak formulation of the boundary value problem from the specified differential equation and boundary conditions.
2. Define the finite element mesh by partitioning the domain into elements, and specifying the nodes defined by these elements.
3. Use the nodes and elements in the mesh to define suitable basis functions.
4. Write the finite element solution as a linear sum of the basis functions. Modify the weak formulation of the problem to determine the system of algebraic equations that the finite element solution satisfies. These algebraic equations will fall into one of two categories. The first category is equations that ensure that all Dirichlet boundary conditions are satisfied. The second category is equations that arise from substituting suitable test functions into an integral condition on the finite element solution.
5. Assemble the algebraic equations. Begin with the second category of equations given in step #4 above. Calculate the nonzero contributions from each element, and use these local contributions to increment the correct entries of the global matrix and vector. After this has been done, modify the equations to take account of the boundary conditions.
6. Solve the system of algebraic equations.

2.10 Computational Implementation

We will now develop a practical computational implementation of the material described in this chapter. This implementation will assemble and solve the system of algebraic equations.

The entries of the system of algebraic equations depend on the basis functions used, which in turn depend on the finite element mesh. Before we can assemble the algebraic equations, we must therefore specify the finite element mesh. The mesh we

defined in Sect. 2.4 requires only specification of the number of elements, N. This is sufficient to define the location of the nodes $x_1, x_2, \ldots, x_{N+1}$, and the element length h. Further, these nodes allow us to specify the basis functions through the material given in Sect. 2.5. We are then in a position to assemble the algebraic equations. We will therefore implement the material described in this chapter by writing a function that accepts a positive integer, N, that represents the number of elements, as an input, and returns the vector $\mathbf{x}$ (containing the nodes in the mesh) and the vector $\mathbf{U}$ (containing the finite element solution at each node).

A MATLAB script for the implementation described above is given in Listing 2.1. This function may be called by saving the file as `Chap2_CalculateModelFemSolution.m` and typing, in a MATLAB session,

```
[x, U] = Chap2_CalculateModelFemSolution(40);
```

to run with $N = 40$ elements. Variable names in this script have been chosen to correspond to the mathematical notation used in this chapter. This function begins by initialising entries of the global matrix A and global vector $\mathbf{b}$ to zero in lines 4 and 5, before generating a vector containing the equally spaced nodes in the mesh in line 8. We then loop over all elements in lines 12–23, calculating the nonzero local contributions to A and $\mathbf{b}$ given by Eqs. (2.30) and (2.31), before incrementing the appropriate entries of the global linear system, given by Tables 2.1 and 2.2. MATLAB users will be aware that the expression `A(k:k+1, k:k+1)` that appears in line 21 is shorthand for the submatrix given by

$$\begin{pmatrix} A_{k,k} & A_{k,k+1} \\ A_{k+1,k} & A_{k+1,k+1} \end{pmatrix},$$

and that the expression `b(k:k+1)` that appears in line 22 represents the subvector given by

$$\begin{pmatrix} b_k \\ b_{k+1} \end{pmatrix}.$$

Note that the loop over elements introduces incorrect entries into rows 1 and $N+1$ of A and $\mathbf{b}$, for the reasons explained in Sect. 2.8.2. These entries are overwritten with the correct values in lines 26–31. Note that line 26 uses MATLAB notation to set all entries of row 1 to zero. We then solve the linear system using MATLAB's backslash operator in line 34. This is a suitable method for solving linear systems of the modest size generated here. However, iterative techniques may be more suitable for solving the larger systems of algebraic equations that arise from the application of the finite element method to partial differential equations in later chapters. A summary of these linear algebra techniques is given in Appendix A. Finally, in lines 37–39, we plot the finite element solution that has been calculated, which is a linear approximation to the solution on each element. This finite element solution may be plotted by drawing a straight line between the finite element solution at adjacent nodes: this is exactly what is plotted by line 37 of the listing.

Listing 2.1 `Chap2_CalculateModelFemSolution.m`, a MATLAB function for calculating the finite element solution of the model problem.

```
function [x, U] = Chap2_CalculateModelFemSolution(N)

% Initialise A and b to zero
A = zeros(N+1, N+1);
b = zeros(N+1, 1);

% Generate N+1 nodes, equally spaced between 0 and 1
x = linspace(0, 1, N+1);

% Loop over elements calculating local contributions
% and incrementing the global linear system
for k=1:N
    % Calculate element length
    h = 1/N;

    % Calculate local contributions
    A_local = [1/h, -1/h; -1/h, 1/h];
    b_local = [h; h];

    % Increment global system
    A(k:k+1, k:k+1) = A(k:k+1, k:k+1) + A_local;
    b(k:k+1) = b(k:k+1) + b_local;
end

% Set Dirichlet boundary conditions
A(1,:) = 0;
A(1,1) = 1;
b(1) = 0;
A(N+1,:) = 0;
A(N+1,N+1) = 1;
b(N+1) = 0;

% Solve linear system and plot FE solution
U = A\b;

% Plot finite element solution
plot(x, U, '-o')
xlabel('x')
ylabel('U')
```

2.11 Evaluating the Finite Element Solution at a Given Point

We have now developed a computational implementation of the finite element method that computes the quantities $U_1, U_2, \ldots, U_{N+1}$ that appear in the finite element solution given by Eq. (2.12). We will often need to evaluate the solution at some given

point x, that is, evaluate $U(x)$. If x is a node in the mesh, i.e. $x_i = x$ for some i, we may then use Eq. (2.13) to write

$$U(x) = U_i.$$

If x is not a node in the mesh, then it will lie in element k for some k, where $x_k < x < x_{k+1}$. We first locate the element k that x lies in. Noting, as we did earlier in this chapter, that only the basis functions $\phi_k(x)$ and $\phi_{k+1}(x)$ are nonzero on element k, the finite element solution given by Eq. (2.12) may be written, for $x_k < x < x_{k+1}$, as

$$U(x) = U_k\phi_k(x) + U_{k+1}\phi_{k+1}(x).$$

We then use the definition of the basis functions $\phi_k(x)$ and $\phi_{k+1}(x)$, given by Eqs. (2.8)–(2.10), to deduce that

$$U(x) = \frac{x_{k+1} - x}{h}U_k + \frac{x - x_k}{h}U_{k+1}.$$

2.12 Is the Finite Element Solution Correct?

It is always worth checking the solution to any mathematical problem, whether the solution has been obtained by analytic "pen and paper" techniques or as the output from a computer program. Easily made typographical errors, such as a missed factor of 2, will at best lead to inaccuracies in the solution. There is, however, the potential for completely nonsensical output, for example a physical quantity being assigned a value that is calculated from taking the square root of a negative number. It is, therefore, useful to perform some verification of the computation of a finite element solution. An exhaustive verification, such as comparing the finite element solution of a differential equation with the true solution of the equation, is rarely possible. After all, if we knew the true solution of a differential equation, we would be unlikely to go to the effort of computing the finite element solution. Nevertheless, some validation is possible. Below we list a few simple tests that can usually be carried out.

- The simplest test that can be carried out is to plot the finite element solution to check that the solution looks realistic and takes values that are reasonable—a computed chemical concentration that takes a negative value, for example, should set alarm bells ringing.
- Another simple check that should be carried out is to compare finite element solutions that have been calculated using different numbers of elements. The finite element solution should become more accurate, and approach a limit as the number of elements is successively increased.
- Finally, analytic solutions may exist for some choices of parameter values that appear in the differential equation, or for simplifications of the differential equa-

tion. The finite element solution should be computed in these cases and compared to the analytic solution.

2.13 Exercises

For these exercises, you will need to download the MATLAB script given in Listing 2.1. Remember that the true solution to the differential equation on which this script is based is $u(x) = x(1 - x)$.

2.1 In this exercise, we will modify the MATLAB script given in Listing 2.1 so that it returns the finite element solution at the point $x = 0.45$.

(a) Modify the MATLAB script so that it returns the finite element solution at $x = 0.45$ when this point is a node of the mesh. Use the true solution, and a mesh with $N = 100$ elements, to verify that your answer is correct.
(b) Use the material given in Sect. 2.11 to modify the MATLAB script to evaluate the finite element solution at $x = 0.45$ when this point is not a node of the mesh. Use the true solution, and a mesh with $N = 101$ elements, to verify that your answer is correct.
(c) Combine your answers to parts (a) and (b) so that the MATLAB script first determines whether or not $x = 0.45$ is a node of the mesh, before returning the finite element solution at that point.

2.2 We denote the error between the finite element solution and the true solution by $E(x) = U(x) - u(x)$. The L^2 norm of the error of the finite element solution, $\|E\|_{L^2}$, is defined to be

$$\|E\|_{L^2} = \sqrt{\int_0^1 (U(x) - u(x))^2 \, \mathrm{d}x},$$

which may be written as the sum of the contribution from individual elements:

$$\|E\|_{L^2} = \sqrt{\sum_{k=1}^{N} \int_{x_k}^{x_{k+1}} (U(x) - u(x))^2 \, \mathrm{d}x}.$$

(a) In Sect. 2.11, we saw that, for $x_k < x < x_{k+1}$, the finite element solution is given by

$$U(x) = \frac{x_{k+1} - x}{h} U_k + \frac{x - x_k}{h} U_{k+1}.$$

Use this expression for $U(x)$ to explain why, for the model problem used in this chapter, we may write

$$\|E\|_{L^2}^2 = \sum_{k=1}^{N} \int_{x_k}^{x_{k+1}} \left(\frac{x_{k+1} - x}{h} U_k + \frac{x - x_k}{h} U_{k+1} - x(1-x) \right)^2 \mathrm{d}x.$$

(b) Evaluate, by direct integration, the quantity

$$\int_{x_k}^{x_{k+1}} \left(\frac{x_{k+1} - x}{h} U_k + \frac{x - x_k}{h} U_{k+1} - x(1-x) \right)^2 \mathrm{d}x,$$

and write your answer as a function of U_k and U_{k+1}.

(c) Using your answers to parts (a) and (b), modify the MATLAB script given in Listing 2.1 so that it outputs the value of $\|E\|_{L^2}$ for a given number of elements, N.

(d) Define the element length by $h = \frac{1}{N}$ where N is the number of elements. You may assume that, as $h \to 0$, $\|E\|_{L^2}$ is related to the element length h by

$$\|E\|_{L^2} = Ch^p,$$

for some constants C and p. Taking the logarithm of both sides of this expression gives

$$\log\left(\|E\|_{L^2}\right) = \log C + p \log h,$$

and so plotting $\log\left(\|E\|_{L^2}\right)$ against $\log h$ will give a straight line of gradient p as $h \to 0$. By plotting $\log\left(\|E\|_{L^2}\right)$ against $\log h$ for several values of h, estimate the constant p.

2.3 In Sect. 2.11, we saw that, for $x_k < x < x_{k+1}$, the finite element solution is given by

$$U(x) = \frac{x_{k+1} - x}{h} U_k + \frac{x - x_k}{h} U_{k+1}.$$

(a) Write down the derivative of the finite element solution, $\frac{\mathrm{d}U}{\mathrm{d}x}$, on element k.

(b) Modify the MATLAB script given in Listing 2.1 so that it plots the derivative of the solution. Note that this derivative is a function that may be discontinuous across element boundaries.

2.4 A boundary value problem is defined by

$$-3\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = 2, \qquad 0 < x < 1,$$

together with boundary conditions

$$u(0) = 1, \qquad u(1) = 2.$$

(a) Write down the analytic solution to this boundary value problem.
(b) By working through the steps required to calculate the finite element solution, summarised in Sect. 2.9, modify the MATLAB script given in Listing 2.1 to calculate the finite element approximation of the boundary value problem above. Compare your finite element solution to the true solution.

http://www.springer.com/978-3-319-49970-3

Finite Element Methods
A Practical Guide
Whiteley, J.
2017, XI, 232 p. 28 illus., Hardcover
ISBN: 978-3-319-49970-3