An Introduction to Finite Element Methods

Mehrdad Negahban
Department of Engineering Mechanics
University of Nebraska
Lincoln, NE 68588

November 22, 2000

Contents

Chapter 1

A 1-D Example

1.1 Introduction

The finite element method (FEM) is a numerical method to solve differential equations. There are several basic steps to solving a problem by the FEM method. These steps are:

1. Formulation of the problem into a differential or integral equation.
2. Development of the *weak formulation* of the problem.
3. Finite element approximation of the domain and variables.
4. Assembling the stiffness matrix and load vector and solving the problem.
5. Post processing (analysis and presentation of the results).

1.2 Steady State Heat Conduction in a Bar

We will proceed to describe the steps in the finite element method by the aid of an example. Consider the problem of heat conduction through a bar.

1.2.1 Step 1: Formulation of the problem

The problem under consideration is that of the steady state flow of heat through a bar. Figure 1.1 shows how heat flows through an element of this bar. Balancing the heat entering and exiting this element gives

$$q(x) + f(x)\triangle x = q(x + \triangle x), \tag{1.1}$$

where q is the heat flux along the bar and $f(x)$ is the heat flow through the lateral surfaces per unit length of the bar. Reorganizing and taking the limit as $\triangle x$ goes to zero yields

$$\frac{dq}{dx} = f(x). \tag{1.2}$$

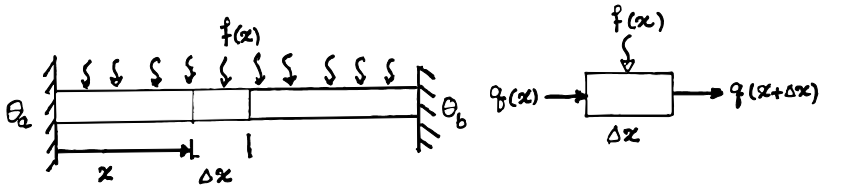

Figure 1.1: Steady state thermal conduction in a bar.

Introducing Fourier's law of heat conduction ($q = -\kappa\frac{d\theta}{dx}$) into this equation results in

$$-\kappa\frac{d^2\theta}{dx^2} = f(x), \tag{1.3}$$

where θ is the temperature and κ is the coefficient of thermal conduction.

Since the problem is a second order ordinary differential equation one can prescribe two conditions. We will look at solving the above differential equation subject to any one of the following boundary conditions:

- $\theta = \theta_a$ at $x = a$ and $\theta = \theta_b$ at $x = b$,
- $\theta = \theta_a$ at $x = a$ and $q = -\kappa\frac{d\theta}{dx} = q_b$ at $x = b$,
- $q = -\kappa\frac{d\theta}{dx} = q_a$ at $x = a$ and $\theta = \theta_b$ at $x = b$, or
- $q = -\kappa\frac{d\theta}{dx} = q_a$ at $x = a$ and $q = -\kappa\frac{d\theta}{dx} = q_b$ at $x = b$.

1.2.2 Step 2: Formulation of the weak form

The solution to this problem is a temperature field which satisfies the differential equation

$$-\kappa\frac{d^2\theta}{dx^2} = f(x) \tag{1.4}$$

at every point in the bar. We will relax this requirement by replacing this equation by

$$-\int_{\mathcal{R}} \kappa\overline{\theta}\frac{d^2\theta}{dx^2}dx = \int_{\mathcal{R}} \overline{\theta}f(x)dx, \tag{1.5}$$

where $\mathcal{R}$ is any region of the bar and $\overline{\theta}$ is a test function. If one requires that this equation hold for all possible test functions $\overline{\theta}$, it will be possible to get the original differential equation from this integral equation. Any point in the bar can be isolated by selecting a test function which is only positive and is nonzero only in a small region around that point. The aim is not to require the integral equation to hold for all possible $\overline{\theta}$ functions, but to select a set of test functions and require the integral equality to hold for this set of functions. Hence, we do not require the differential equation to be exactly satisfied at all points, but require the temperature field to only satisfy the integral equation for a select number of test functions (we have weakened the original equation).

We will now use integration by parts to get the weak form of the problem as

$$-[\kappa\overline{\theta}\frac{d\theta}{dx}]_{x=a}^{b} + \int_a^b \kappa\frac{d\overline{\theta}}{dx}\frac{d\theta}{dx}dx = \int_a^b \overline{\theta}f(x)dx, \tag{1.6}$$

subject to the appropriate boundary conditions. The application of integration by parts also introduces a weakening of the problem since we are no longer required to have temperature fields which have a second derivative. It is now only required that the temperature field have a first derivative as can be seen from equation (1.6).

1.2.3 Step 3: Finite element approximation

We will use two node line elements to approximate both the temperature field and the test functions. The length of the bar will be partitioned into *ne* elements as shown in Figure 1.2. Each element has an element number and one node at each end. Nodes are given two numbers. The global node number is one which is unique for each node and which distinguishes it from all other nodes. For each element there is also a local numbering system. For a two node element the local nodes are distinguished in the element by designating a local node number of 1 to one node and 2 to the other node.

The temperature and the test function will be approximated over each element by

$$\theta = \theta_1^e N_1(x) + \theta_2^e N_2(x) = \left[\begin{array}{cc} N_1(x) & N_2(x) \end{array}\right] \left[\begin{array}{c} \theta_1^e \\ \theta_2^e \end{array}\right], \tag{1.7}$$

and

$$\overline{\theta} = \overline{\theta}_1^e N_1(x) + \overline{\theta}_2^e N_2(x) = \left[\begin{array}{cc} \overline{\theta}_1^e & \overline{\theta}_2^e \end{array}\right] \left[\begin{array}{c} N_1(x) \\ N_2(x) \end{array}\right], \tag{1.8}$$

where θ_i^e is the value of θ at local node i, $\overline{\theta}_i^e$ is the value of $\overline{\theta}$ at local node i, and $N_i(x)$ are known functions of x. The N_i are known as *shape functions.* The shape functions are selected such that $N_1 = 1$ at local node 1, $N_1 = 0$ at local node 2, $N_2 = 0$ at local node 1, $N_2 = 1$ at local node 2. For a two node line element of length l_e and local coordinate s as shown in Figure 1.3, the shape functions will be

$$N_1 = 1 - \frac{s}{l_e}, \tag{1.9}$$

and

$$N_2 = \frac{s}{l_e}. \tag{1.10}$$

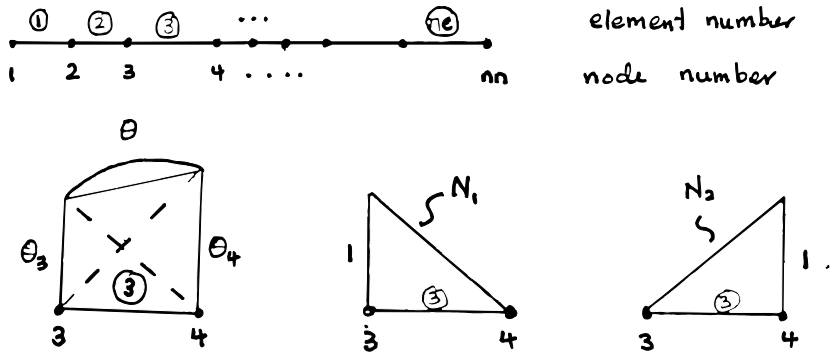

Figure 1.2: Approximation of θ and $\overline{\theta}$ by the use of shape functions.

The shape functions are selected such that the sum of the shape functions at each point will add up to unity. It must be noted that the shape functions are different for different elements, even though this is not made explicit in the notation.

The two integrals in the weak formulation of the problem, given in equation (1.6), can each be written as a sum of integrals, each integral being over one element. This can be written as

$$\int_a^b \kappa \frac{d\overline{\theta}}{dx}\frac{d\theta}{dx}dx = \sum_{e=1}^{ne}\int_{\Omega_e} \kappa \frac{d\overline{\theta}}{dx}\frac{d\theta}{dx}dx, \tag{1.11}$$

and

$$\int_a^b \overline{\theta} f(x)dx = \sum_{e=1}^{ne}\int_{\Omega_e} \overline{\theta} f(x)dx, \tag{1.12}$$

where Ω_e is the domain of element number e.

Over each element one can use the above approximations for the temperature and test function to get

$$\int_{\Omega_e} \kappa \frac{d\overline{\theta}}{dx}\frac{d\theta}{dx}dx = \int_0^{l_e} \kappa \frac{d[\overline{\theta}_1^e N_1 + \overline{\theta}_2^e N_2]}{ds}\frac{d[\theta_1^e N_1 + \theta_2 N_2]}{ds}ds, \tag{1.13}$$

and

$$\int_{\Omega_e} \overline{\theta} f(x)dx = \int_0^{l_e} [\overline{\theta}_1^e N_1 + \overline{\theta}_2^e N_2] f(x)ds, \tag{1.14}$$

since $x = x_1 + s$, where x_1 is the location of the first node in the element. Since θ_1^e, θ_2^e, $\overline{\theta}_1^e$, and $\overline{\theta}_2^e$ are constants, one can write

$$\frac{d\overline{\theta}}{dx} = \frac{d[\overline{\theta}_1^e N_1 + \overline{\theta}_2^e N_2]}{ds} = \left[\begin{array}{cc} \overline{\theta}_1^e & \overline{\theta}_2^e \end{array}\right] \left[\begin{array}{c} \frac{dN_1}{ds} \\ \frac{dN_2}{ds} \end{array}\right], \tag{1.15}$$

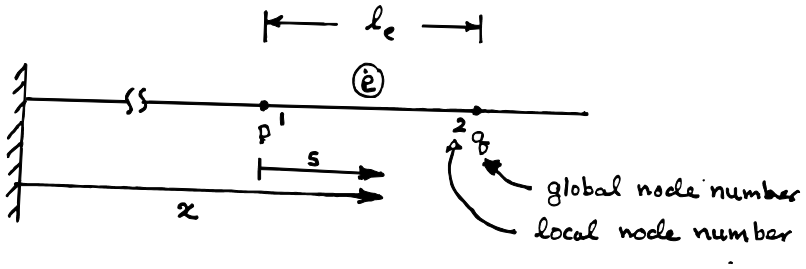

Figure 1.3: Relation between local and global coordinates.

and

$$\frac{d\theta}{dx} = \frac{d[\theta_1^e N_1 + \theta_2^e N_2]}{ds} = \left[\begin{array}{cc} \frac{dN_1}{ds} & \frac{dN_2}{ds} \end{array} \right] \left[\begin{array}{c} \theta_1^e \\ \theta_2^e \end{array} \right]. \tag{1.16}$$

Substitution of these relations into the integrals of equations (1.13) and (1.14) gives

$$\int_{\Omega_e} \kappa \frac{d\overline{\theta}}{dx} \frac{d\theta}{dx} dx = \kappa \left[\begin{array}{cc} \overline{\theta}_1^e & \overline{\theta}_2^e \end{array} \right] \int_0^{l_e} \left[\begin{array}{c} \frac{dN_1}{ds} \\ \frac{dN_2}{ds} \end{array} \right] \left[\begin{array}{cc} \frac{dN_1}{ds} & \frac{dN_2}{ds} \end{array} \right] ds \left[\begin{array}{c} \theta_1^e \\ \theta_2^e \end{array} \right], \tag{1.17}$$

and

$$\int_{\Omega_e} \overline{\theta} f(x) dx = \left[\begin{array}{cc} \overline{\theta}_1^e & \overline{\theta}_2^e \end{array} \right] \int_0^{l_e} \left[\begin{array}{c} N_1 \\ N_2 \end{array} \right] f(x) ds. \tag{1.18}$$

These equations can be written as

$$\int_{\Omega_e} \kappa \frac{d\overline{\theta}}{dx} \frac{d\theta}{dx} dx = [\overline{\theta}^e]^T [K^e] [\theta^e], \tag{1.19}$$

and

$$\int_{\Omega_e} \overline{\theta} f(x) dx = [\overline{\theta}^e]^T [f^e], \tag{1.20}$$

where

$$[\overline{\theta}^e] = \left[\begin{array}{c} \overline{\theta}_1^e \\ \overline{\theta}_2^e \end{array} \right], \tag{1.21}$$

$$[\theta^e] = \left[\begin{array}{c} \theta_1^e \\ \theta_2^e \end{array} \right], \tag{1.22}$$

$$[K^e] = \begin{bmatrix} K^e_{11} & K^e_{12} \\ K^e_{21} & K^e_{22} \end{bmatrix} = \kappa \int_0^{l_e} \begin{bmatrix} (\frac{dN_1}{ds})^2 & \frac{dN_1}{ds}\frac{dN_2}{ds} \\ \frac{dN_2}{ds}\frac{dN_1}{ds} & (\frac{dN_2}{ds})^2 \end{bmatrix} ds, \tag{1.23}$$

$$[f^e] = \begin{bmatrix} f^e_1 \\ f^e_2 \end{bmatrix} = \int_0^{l_e} \begin{bmatrix} N_1 \\ N_2 \end{bmatrix} f(x) ds. \tag{1.24}$$

The matrix $[K^e]$ is known as the *element stiffness matrix* and the matrix $[f^e]$ is known as the *element load vector.* For the shape functions in (1.9) and (1.10) we have $\frac{dN_1}{ds} = -\frac{1}{l_e}$ and $\frac{dN_2}{ds} = \frac{1}{l_e}$. Therefore, the element stiffness matrix can be written as

$$[K^e] = \kappa \int_0^{l_e} \begin{bmatrix} \frac{1}{l_e^2} & -\frac{1}{l_e^2} \\ -\frac{1}{l_e^2} & \frac{1}{l_e^2} \end{bmatrix} ds = \begin{bmatrix} \frac{\kappa}{l_e} & -\frac{\kappa}{l_e} \\ -\frac{\kappa}{l_e} & \frac{\kappa}{l_e} \end{bmatrix}. \tag{1.25}$$

To evaluate the element load vector one needs to know $f(x)$. We will later calculate this for a particular example.

The weak form of the problem can now be written as

$$[\overline{\theta} q]_{x=a}^b + \sum_{e=1}^{ne} [\overline{\theta}^e]^T [K^e][\theta^e] = \sum_{e=1}^{ne} [\overline{\theta}^e]^T [f^e], \tag{1.26}$$

where Fourier's law is used to replace the derivative of temperature in the first term.

1.2.4 Step 4: Assembling the stiffness matrix and load vector

The next step is to organize the problem in the form of a set of algebraic equations. Consider the example of a problem with two elements and three nodes as shown in Figure 1.4. The two summations in equation (1.26) can be written as

$$\sum_{e=1}^{2} [\overline{\theta}^e]^T [K^e][\overline{\theta}^e] = \begin{bmatrix} \overline{\theta}^1_1 & \overline{\theta}^1_2 \end{bmatrix} \begin{bmatrix} K^1_{11} & K^1_{12} \\ K^1_{21} & K^1_{22} \end{bmatrix} \begin{bmatrix} \theta^1_1 \\ \theta^1_2 \end{bmatrix} + \begin{bmatrix} \overline{\theta}^2_1 & \overline{\theta}^2_2 \end{bmatrix} \begin{bmatrix} K^2_{11} & K^2_{12} \\ K^2_{21} & K^2_{22} \end{bmatrix} \begin{bmatrix} \theta^2_1 \\ \theta^2_2 \end{bmatrix}$$

$$= \begin{bmatrix} \overline{\theta}_1 & \overline{\theta}_2 & \overline{\theta}_3 \end{bmatrix} \begin{bmatrix} K^1_{11} & K^1_{12} & 0 \\ K^1_{21} & K^1_{22} + K^2_{11} & K^2_{12} \\ 0 & K^2_{21} & K^2_{22} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}, \tag{1.27}$$

and

$$\sum_{e=1}^{2} [\overline{\theta}^e][f^e] = \begin{bmatrix} \overline{\theta}^1_1 & \overline{\theta}^1_2 \end{bmatrix} \begin{bmatrix} f^1_1 \\ f^1_2 \end{bmatrix} + \begin{bmatrix} \overline{\theta}^2_1 & \overline{\theta}^2_2 \end{bmatrix} \begin{bmatrix} f^2_1 \\ f^2_2 \end{bmatrix}$$

$$= \begin{bmatrix} \overline{\theta}_1 & \overline{\theta}_2 & \overline{\theta}_3 \end{bmatrix} \begin{bmatrix} f^1_1 \\ f^1_2 + f^2_1 \\ f^2_2 \end{bmatrix} \tag{1.28}$$

Therefore, for this problem the weak formulation given in equation (1.29) can be put in the form

$$[\overline{\theta}_3 q_3 - \overline{\theta}_1 q_1] + [\overline{\theta}]^T [K][\theta] = [\overline{\theta}][f], \tag{1.29}$$

where $[K]$ is known as the *global stiffness matrix,* [f] is known as the *global load vector,* and

$$[\overline{\theta}] = \begin{bmatrix} \overline{\theta}_1 \\ \overline{\theta}_2 \\ \overline{\theta}_3 \end{bmatrix}, \tag{1.30}$$

$$[\theta] = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}, \tag{1.31}$$

$$[K] = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} = \begin{bmatrix} K_{11}^1 & K_{12}^1 & 0 \\ K_{21}^1 & K_{22}^1 + K_{11}^2 & K_{12}^2 \\ 0 & K_{21}^2 & K_{22}^2 \end{bmatrix}, \tag{1.32}$$

$$[f] = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} f_1^1 \\ f_2^1 + f_1^2 \\ f_2^2 \end{bmatrix}. \tag{1.33}$$

To illustrate the way in which one arrives at the final set of equations we will write the expanded form of equation (1.29) in the form

$$[\overline{\theta}_3 q_3 - \overline{\theta}_1 q_1] + \begin{bmatrix} \overline{\theta}_1 & \overline{\theta}_2 & \overline{\theta}_3 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \overline{\theta}_1 & \overline{\theta}_2 & \overline{\theta}_3 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}. \tag{1.34}$$

The test functions $\overline{\theta}$ are arbitrary functions which we will select in the manner shown in Figure 1.4. Three test functions are selected since we will have three unknowns in this problem (θ_1, θ_2, θ_3; q_1, θ_2, θ_3; θ_1, θ_2, q_2; or q_1, θ_2, q_2). Each test function provides one algebraic equation for the unknowns. As can be seen from Figure 1.4, the first test function has $\overline{\theta}_1 = 1$, $\overline{\theta}_2 = 0$, and $\overline{\theta}_3 = 0$. The second test function has $\overline{\theta}_1 = 0$, $\overline{\theta}_2 = 1$, and $\overline{\theta}_3 = 0$. The third test function has $\overline{\theta}_1 = 0$, $\overline{\theta}_2 = 0$, and $\overline{\theta}_3 = 1$. The following three equations are obtained from the substitution of the selected test functions into equation (1.34).

$$\begin{aligned} -q_1 + K_{11}\theta_1 + K_{12}\theta_2 + K_{13}\theta_3 &= f_1, \\ K_{21}\theta_1 + K_{22}\theta_2 + K_{23}\theta_3 &= f_2, \\ q_3 + K_{31}\theta_1 + K_{32}\theta_2 + K_{33}\theta_3 &= f_3. \end{aligned} \tag{1.35}$$

These equations can now be solved for the three unknowns. If q_1 and q_3 are specified, then this system can be written as

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} f_1 + q_1 \\ f_2 \\ f_3 - q_3 \end{bmatrix}. \tag{1.36}$$

This would be the problem which one would solve if this was a well posed problem. It turns out that this particular set of boundary values are not proper. Since the temperature is not fixed at any point in the bar, the temperature profile can slide up and down. An examination of the stiffness matrix shows that its determinant is zero (i.e., the system is singular).

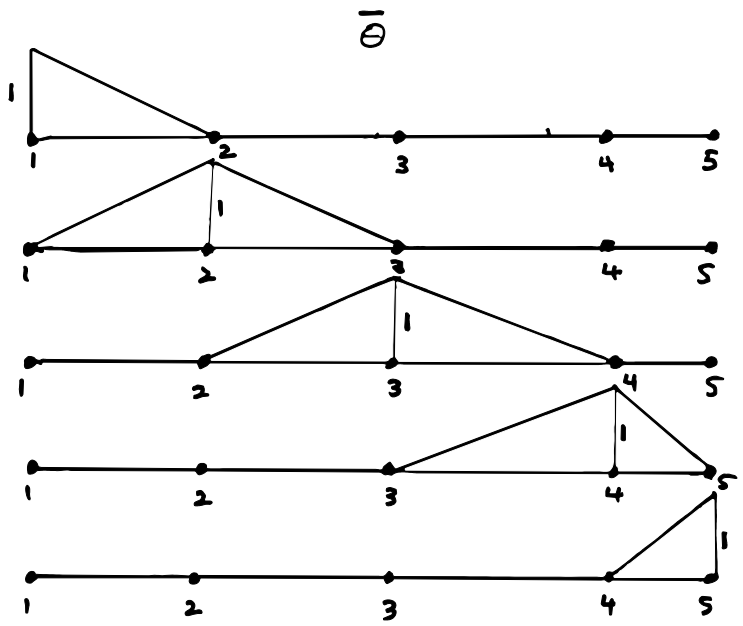

Figure 1.4: Test functions used to get the final form of the equation.

If θ_1 and θ_3 are specified, then the middle equation can be used to find θ_2, and the first and last equation are used to find the heat flux at the two ends. The, mixed type boundary conditions can also be solved in a similar manner.

Before we proceed it is important to state that in practice the boundary conditions are not imposed as was presented above. Theoretically, if we have known values of temperature at the boundary we should eliminated these known values from the set of unknowns and introduce the unknown heat fluxes in the boundary term into the list of unknowns. To avoid this complex procedure, the *penalty method* is used for imposing the temperature boundary. This method modifies the stiffness matrix and load vector to fix the value of temperature to a given value at a boundary node. The method is based on the fact that for any boundary node i there will be an equation

$$K_{ii}\theta_i + \quad other \;\; terms \quad = f_i. \tag{1.37}$$

To impose the condition $\theta_i = \hat{\theta}$ one can add a term $pK_{ii}\theta_i$ to the left hand side of the equation and the term $pK_{ii}\hat{\theta}$ to the right hand side of the equation to get

$$K_{ii}\theta_i + pK_{ii}\theta_i + \quad other \;\; terms \quad = f_i + pK_{ii}\hat{\theta} \tag{1.38}$$

where p is a large number. After dividing by pK_{ii} one gets

$$\frac{\theta_i}{p} + \theta_i + \frac{other \;\; terms}{pK_{ii}} = \frac{f_i}{pK_{ii}} + \hat{\theta} \tag{1.39}$$

All term in this equation become small accept the two newly added ones. Therefore, the equation will become dominated by the equation $\theta_i = \hat{\theta}$ and the other terms become unimportant. Since the method eliminates the importance of any other terms entering this equation, it will not be necessary to introduce the unknown heat flux into the list of unknowns.

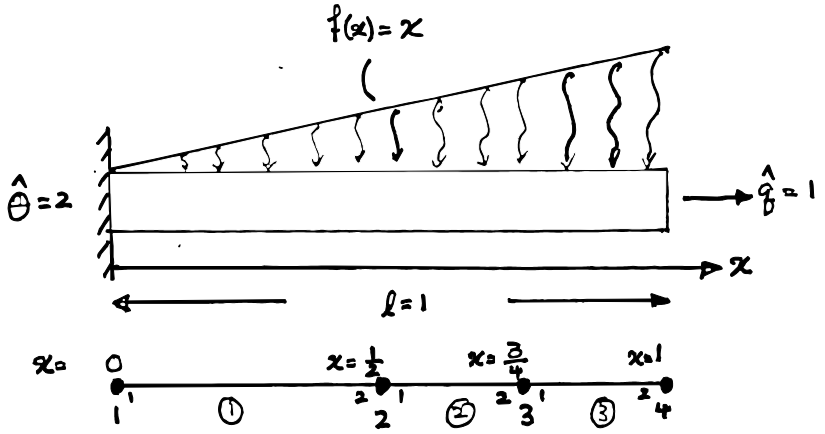

Figure 1.5: Example problem.

In short, to impose a temperature boundary condition at global node i one can multiply K_{ii} in the stiffness matrix by $(1+p)$ (i.e., add pK_{ii} to K_{ii}), and add $pK_{ii}\hat{\theta}$ to f_i. To impose a heat flux boundary condition at global node j one adds q to f_j, where q is a heat flux directed into the bar.

1.3 An example

As an example consider the problem of the bar shown in Figure 1.5. The bar is of length $l = 1$, is connected to a constant temperature thermal bath of temperature $\hat{\theta} = 2$ from the left and heat is being drawn from it at a constant rate $\hat{q} = 1$ from the right. The heat flow into the bar per unit length from the lateral surfaces is given by $f(x) = x$. The coefficient of thermal conduction $\kappa = 1$.

Three two node line elements are selected as shown in Figure 1.5. The local stiffness matrices can be calculated using equation (1.25) and will be

$$[K^1] = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix},$$

$$[K^2] = \begin{bmatrix} 4 & -4 \\ -4 & 4 \end{bmatrix},$$

$$[K^3] = \begin{bmatrix} 4 & -4 \\ -4 & 4 \end{bmatrix}.$$

The global stiffness matrix will, therefore, be

$$[K] = \begin{bmatrix} 2 & -2 & 0 & 0 \\ -2 & (2+4) & -4 & 0 \\ 0 & -4 & (4+4) & -4 \\ 0 & 0 & -4 & 4 \end{bmatrix}.$$

The load vectors can be calculated using equation (1.24). Using equation (1.9) and (1.10), the shape functions for the first element are

$$N_1 = 1 - 2x, \quad N_2 = 2x.$$

Therefore,

$$f_1^1 = \int_0^{\frac{1}{2}} (1-2x)x dx = \frac{1}{24},$$

$$f_2^1 = \int_0^{\frac{1}{2}} 2x^2 dx = \frac{1}{12}.$$

The shape functions for the second element are

$$N_1 = 1 - 4(x - \frac{1}{2}), \quad N_2 = 4(x - \frac{1}{2}).$$

Therefore,

$$f_1^2 = \int_{\frac{1}{2}}^{\frac{3}{4}} [1 - 4(x - \frac{1}{2})]x dx = \frac{7}{96},$$

$$f_2^2 = \int_{\frac{1}{2}}^{\frac{3}{4}} 4(x - \frac{1}{2})x dx = \frac{1}{12}.$$

The shape functions for the third element are

$$N_1 = 1 - 4(x - \frac{3}{4}), \quad N_2 = 4(x - \frac{3}{4}).$$

Therefore,

$$f_1^3 = \int_{\frac{3}{4}}^{1} [1 - 4(x - \frac{3}{4})]x dx = \frac{5}{48},$$

$$f_2^3 = \int_{\frac{3}{4}}^{1} 4(x - \frac{3}{4})x dx = \frac{11}{96}.$$

The global load vector will become

$$[f] = \begin{bmatrix} \frac{1}{24} \\ \frac{1}{12} + \frac{7}{96} \\ \frac{1}{12} + \frac{5}{48} \\ \frac{11}{96} \end{bmatrix}.$$

The problem that we must solve is

$$\begin{bmatrix} 2(1+p) & -2 & 0 & 0 \\ -2 & 6 & -4 & 0 \\ 0 & -4 & 8 & -4 \\ 0 & 0 & -4 & 4 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{24} + 4p \\ \frac{5}{32} \\ \frac{3}{16} \\ \frac{11}{96} - 1 \end{bmatrix},$$

where p is a large number and the stiffness matrix and load vectors have been modified to impose the boundary conditions.

For $p = 1$ the solution will be

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \frac{7}{4} \\ \frac{71}{48} \\ \frac{167}{128} \\ \frac{13}{12} \end{bmatrix} = \begin{bmatrix} 1.75 \\ 1.47916 \\ 1.30468 \\ 1.08333 \end{bmatrix}.$$

For $p = 10$ the solution is

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \frac{79}{40} \\ \frac{409}{240} \\ \frac{979}{640} \\ \frac{157}{120} \end{bmatrix} = \begin{bmatrix} 1.975 \\ 1.70416 \\ 1.52968 \\ 1.30833 \end{bmatrix}.$$

For $p = 100$ the solution is

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \frac{799}{400} \\ \frac{259}{150} \\ \frac{4967}{3200} \\ \frac{1597}{1200} \end{bmatrix} = \begin{bmatrix} 1.9975 \\ 1.72666 \\ 1.55218 \\ 1.33083 \end{bmatrix}.$$

The exact solution is

$$\theta = 2 - \frac{x^3}{6} - \frac{x}{2},$$

which yields

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 2 \\ \frac{83}{48} \\ \frac{199}{128} \\ \frac{4}{3} \end{bmatrix} = \begin{bmatrix} 2 \\ 1.72916 \\ 1.55468 \\ 1.33333 \end{bmatrix}.$$

Comparison of the exact solution and the finite element method solution shows that for $p = 100$ there is less than 1% error in the finite element solution.

Chapter 2

A 2-D Example

The following is an example of how to solve a two dimensional problem using the finite element method. The problem under consideration is the torsion of a noncircular member.

2.1 Step 1: The torsion problem

The differential equation for the torsion problem is given by

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = -2G\theta, \tag{2.1}$$

where ϕ is a function of x and y and is called the potential function, G is the shear modulus, and θ is the amount of twist per unit length of the bar.[1] To solve the torsion problem one must solve for the potential function ϕ which satisfies this differential equation at every point in the cross section Ω and which also satisfies the boundary conditions

$$\phi = 0 \tag{2.2}$$

on the boundary curve $\partial\Omega$.

If one defines the gradient operator in the x-y plain as

$$\nabla = \hat{i}\frac{\partial}{\partial x} + \hat{j}\frac{\partial}{\partial y}, \tag{2.3}$$

then one will have

$$\nabla \circ \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \tag{2.4}$$

Therefore, one can write the problem as

$$\nabla \circ \nabla \phi = -2G\theta \quad in \quad \Omega \tag{2.5}$$

[1] A similar equation represents the displacement of a thin elastic membrane under pressure. This equation is

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\frac{P}{T},$$

where u is the displacement of the membrane, P is pressure, and T is the constant tension per unit length.

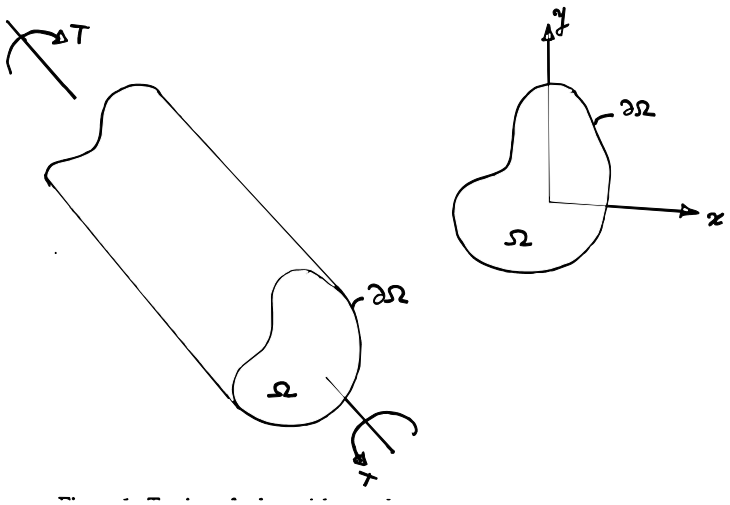

Figure 2.1: Torsion of a bar with an arbitrary cross section.

and

$$\phi = 0 \quad on \quad \partial\Omega \tag{2.6}$$

In this problem the shear stress at any point of the cross section can be found from ϕ using

$$\tau_{xy} = \frac{\partial \phi}{\partial z} \quad and \quad \tau_{xz} = -\frac{\partial \phi}{\partial y}. \tag{2.7}$$

The torque T required to crate a twist of θ is given by

$$T = 2\int_{\Omega} \phi dA, \tag{2.8}$$

where dA is the differential element of area.

2.2 Step 2: The weak form of the torsion problem

In a simple way one can describe the generation of the weak form of an equation as follows. If for every c one can show $ca = cb$, then one can conclude that $a = b$. The weak form of equation $a = b$ in this case is $ca = cb$.

To formulate the weak form of the torsion problem we will multiply both sides of (2.5) by a arbitrary potential function $\bar{\phi}(x, y)$. This will give the relation

$$\bar{\phi}\nabla \circ \nabla\phi = -2G\theta\bar{\phi} \quad in \quad \Omega. \tag{2.9}$$

Since this equation must hold for every point in the cross section Ω, we can integrate this relation over any region $\mathcal{R}$ of the cross section to get

$$\int_{\mathcal{R}} \bar{\phi}\nabla \circ \nabla\phi dA = -2\int_{\mathcal{R}} G\theta\bar{\phi}dA. \tag{2.10}$$

This equation is regarded as the weak form of the differential equation (2.5). For this equation to be equal to equation (2.5), this relation must hold for every function $\bar{\phi}$.

We will rewrite equation (2.10) in a more convenient form using first the relation

$$\nabla \circ (\bar{\phi}\nabla\phi) = \bar{\phi}\nabla \circ \nabla\phi + \nabla\bar{\phi} \circ \nabla\phi. \tag{2.11}$$

Substitution of $\bar{\phi}\nabla \circ \nabla\phi$ from this equation into (2.10) yields

$$\int_{\mathcal{R}} \nabla \circ (\bar{\phi}\nabla\phi)dA - \int_{\mathcal{R}} \nabla\bar{\phi} \circ \nabla\phi dA = -2\int_{\mathcal{R}} G\theta\bar{\phi}dA. \tag{2.12}$$

Next we use the divergence theorem which states

$$\int_{\mathcal{R}} \nabla \circ \mathbf{v}dA = \int_{\mathcal{C}} \mathbf{v} \circ \mathbf{n}ds, \tag{2.13}$$

for every vector $\mathbf{v}$ and where $\mathbf{n}$ is the outward normal to $\mathcal{C}$ and ds is the differential element of length along $\mathcal{C}$. Selecting $\mathbf{v} = \bar{\phi}\nabla\phi$ one can convert the first integral over $\mathcal{R}$ into an integral over $\mathcal{C}$. This results into the final weak form of the equation as

$$\int_{\mathcal{C}} \bar{\phi}\nabla\phi \circ \mathbf{n}dA - \int_{\mathcal{R}} \nabla\bar{\phi} \circ \nabla\phi dA = -2\int_{\mathcal{R}} G\theta\bar{\phi}dA. \tag{2.14}$$

2.3 Step 3: Discretization of the domain

The third step is to discretize the domain. Here we will use the three node triangular element which is the simplest two dimensional element to discretize the cross section Ω. Figure 2.2 shows this element. This three node triangular element has one node at each of its corners. Figure 2.2 also shows an example of how the domain of the cross section can be modeled using this element. In this example the domain is modeled by six triangular elements. The element number is denoted by a number contained in a circle. This model of the domain is called a finite element mesh. This particular mesh has seven nodes.

The nodes in each element have both a "local node number" and a "global node number". The global node number refers to the node numbers given when discretize the domain. Each element also has a set of local node numbers. In this case the local node numbers for each element are 1, 2, and 3. For example, Figure 2.3 shows both the local and global node numbers for element 2. As can be seen from the figure the local node 1 refers to the global node 2, the local node 2 refers to the global node 8 and the local node 3 refers to the global node 9.

There is no order in how one numbers the elements and gives the global node numbers. To avoid keeping track of sign changes, in this problem we will always select local node numbers as counter-clock-wise.

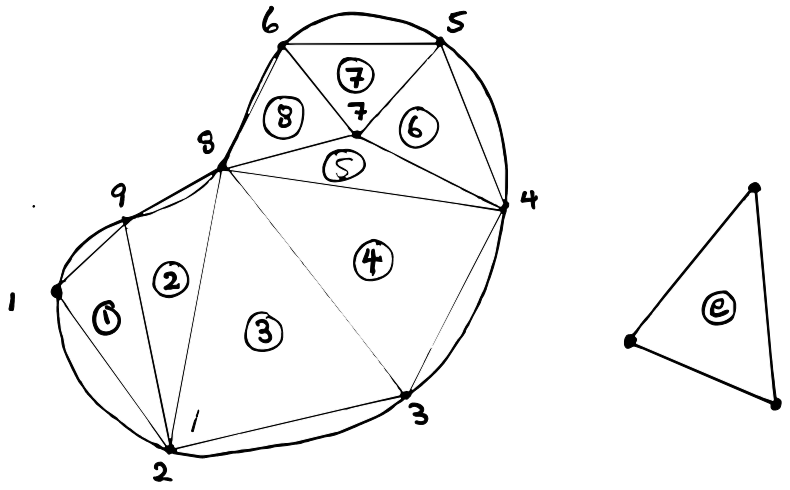

Figure 2.2: Three node triangular elements used to discretize the cross section Ω.

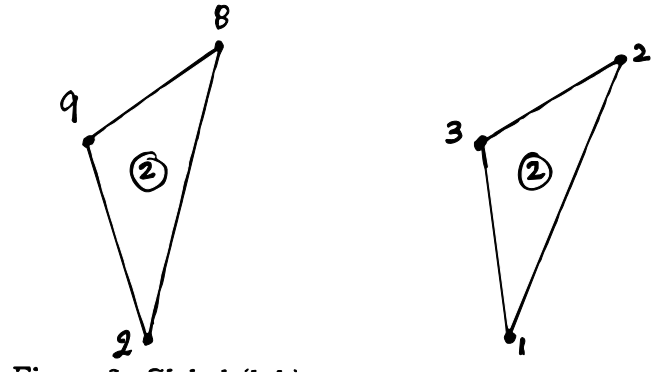

Figure 2.3: Global (left) and local (right) node numbers for element 2.

It is essential that we have a method for going from the local node numbers to the global node number. To do this we will introduce *node connectivity* which is a matrix denoted by IJK(I,N), where IJK is the global node number for a node with a local node number I in element N. As an example consider the local and global node numbers given in Figure 2.3. The connectivity for element 2 will be given by

$$IJK(1,2) = 2, \quad IJK(2,2) = 8, \quad IJK(3,2) = 9. \tag{2.15}$$

The function ϕ is approximated on each element by

$$\phi = \phi_1 N_1(x,y) + \phi_2 N_2(x,y) + \phi_3 N_3(x,y), \tag{2.16}$$

where ϕ_1, ϕ_2, and ϕ_3 are the values of ϕ at local node number 1, 2, and 3, respectively, and N_1, N_2 and N_3 are known functions of x and y. N_i are known as *shape functions.* Let (x_1, y_1), (x_2, y_2) and (x_3, y_3) denote the coordinates of the three nodes of an element. The three conditions $\phi(x_1, y_1) = \phi_1$, $\phi(x_2, y_2) = \phi_2$ and $\phi(x_3, y_3) = \phi_3$ can be satisfied by nine restrictions on the three N_i. At the first node

$$N_1(x_1, y_1) = 1, \quad N_2(x_1, y_1) = 0, \quad N_3(x_1, y_1) = 0. \tag{2.17}$$

At the second node

$$N_1(x_2, y_2) = 0, \quad N_2(x_2, y_2) = 1, \quad N_3(x_2, y_2) = 0. \tag{2.18}$$

At the third node

$$N_1(x_3, y_3) = 0, \quad N_2(x_3, y_3) = 0, \quad N_3(x_3, y_3) = 1. \tag{2.19}$$

Since there are three restrictions on each shape function, one can represent each shape function by a linear function of x an y and solve for the unknown constants. Let us select for each element e the shape functions

$$N_1^e = a_1^e + b_1^e x + c_1^e y \tag{2.20}$$
$$N_2^e = a_2^e + b_2^e x + c_2^e y \tag{2.21}$$
$$N_3^e = a_3^e + b_3^e x + c_3^e y \tag{2.22}$$

where the value of the constants for each element can be evaluated from the above restrictions on N_i. To simplify the notation, we will leave out the superscript e from the notation.

Since in addition to the area integrals we have line integrals on the boundary, we need to discretize the boundary $\partial\Omega$. For the boundary we will use two node line elements. In this case each element has one node at each end. Figure 2.4 shows the discretization of the boundary. As in the discretization of the area, we have both a local and a global numbering system for the nodes. The connectivity of these two numbering systems is given by a matrix IJ(I,N), where IJ is the global node number for local node I of element N.

The function ϕ in the boundary will be approximated by a function of the form

$$\phi = \phi_1 \hat{N}_1(x,y) + \phi_2 \hat{N}_2(x,y), \tag{2.23}$$

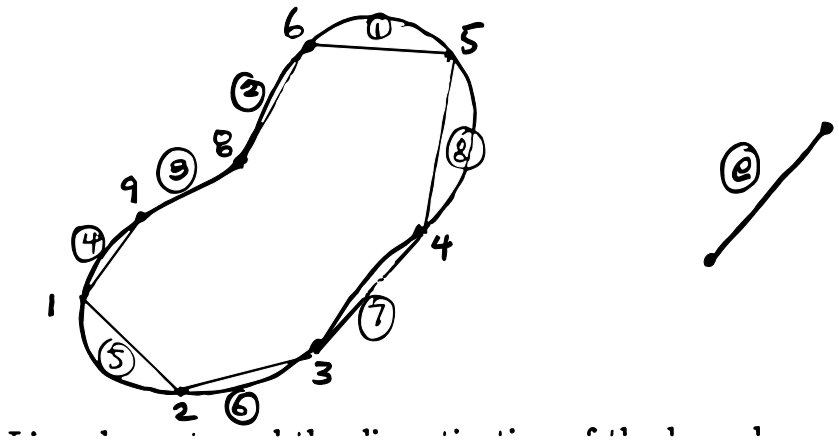

Figure 2.4: Line elements and the discretization of the boundary

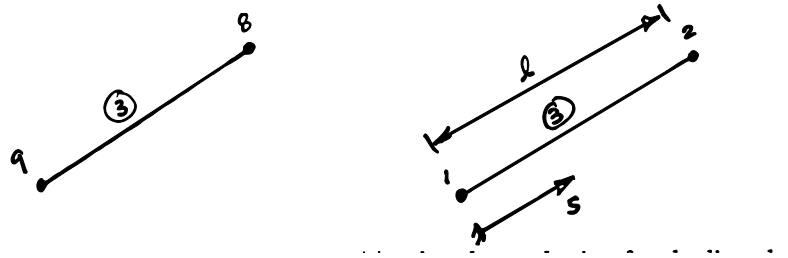

Figure 2.5: Local coordinate s and local node numbering for the line element.

where ϕ_1 and ϕ_2 are the values of ϕ at the two nodes and $\hat{N}_1$ and $\hat{N}_2$ are the shape functions. Using a local coordinate system as shown in figure 2.5, one can write these shape functions as

$$\hat{N}_1 = 1 - \frac{s}{l}, \quad and \quad \hat{N}_2 = \frac{s}{l} \tag{2.24}$$

It must be mentioned that the sum of the shape functions at each point of the element must equal unity. This can be used as a check to see if one has selected the right shape functions.

2.4 Step 4: Finite element approximation of the weak form

For a problem with NN number of nodes the goal is to set the problem in the form

$$[K]\{\phi\} = \{f\}, \tag{2.25}$$

where $[K]$ is an (NN,NN) matrix of constants known as the *global stiffness matrix,* $\{\phi\}$ is the (NN,1) matrix of unknown values of ϕ at the nodes, and $\{f\}$ is known as the *global load vector* and is an (NN,1) matrix of known constants. Once the problem is set into this form one can use a linear equation solver to get the unknown ϕ.

Selecting the region $\mathcal{R}$ in equation (2.14) as the region of one element Ω_e, one can get one equation for each element. Adding these equations over all the elements one gets

$$\sum_{e=1}^{NBE} \int_{\partial\Omega_e} \bar{\phi} \nabla\phi \circ \mathbf{n} ds - \sum_{e=1}^{NE} \int_{\Omega_e} \nabla\bar{\phi} \circ \nabla\phi dA = -2 \sum_{e=1}^{NE} \int_{\Omega_e} G\theta\bar{\phi} dA, \tag{2.26}$$

where NE is the number of elements and NBE is the number of boundary elements. Note that the boundary integrals on the internal boundary between any two adjacent elements cancels out of the summation leaving only the boundary integral over $\partial\Omega$.

On each element we will use the approximations for ϕ and $\bar{\phi}$ as

$$\phi = \phi_1 N_1 + \phi_2 N_2 + \phi_3 N_3, \tag{2.27}$$
$$\bar{\phi} = \bar{\phi}_1 N_1 + \bar{\phi}_2 N_2 + \bar{\phi}_3 N_3, \tag{2.28}$$

where it is understood that the numbers refer to local node numbers for element e, and that the N_i are the shape functions for element e. This equation can be written as

$$\phi = \begin{bmatrix} N_1 & N_2 & N_3 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix} \tag{2.29}$$

and

$$\bar{\phi} = \begin{bmatrix} \bar{\phi}_1 & \bar{\phi}_2 & \bar{\phi}_3 \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} \tag{2.30}$$

The derivatives of ϕ and $\bar{\phi}$ can be written as

$$\frac{\partial \phi}{\partial x} = \left[\begin{array}{ccc} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial x} \end{array} \right] \left[\begin{array}{c} \phi_1 \\ \phi_2 \\ \phi_3 \end{array} \right], \tag{2.31}$$

$$\frac{\partial \phi}{\partial y} = \left[\begin{array}{ccc} \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \frac{\partial N_3}{\partial y} \end{array} \right] \left[\begin{array}{c} \phi_1 \\ \phi_2 \\ \phi_3 \end{array} \right], \tag{2.32}$$

$$\frac{\partial \bar{\phi}}{\partial x} = \left[\begin{array}{ccc} \bar{\phi}_1 & \bar{\phi}_2 & \bar{\phi}_3 \end{array} \right] \left[\begin{array}{c} \frac{\partial N_1}{\partial x} \\ \frac{\partial N_2}{\partial x} \\ \frac{\partial N_3}{\partial x} \end{array} \right], \tag{2.33}$$

$$\frac{\partial \bar{\phi}}{\partial y} = \left[\begin{array}{ccc} \bar{\phi}_1 & \bar{\phi}_2 & \bar{\phi}_3 \end{array} \right] \left[\begin{array}{c} \frac{\partial N_1}{\partial y} \\ \frac{\partial N_2}{\partial y} \\ \frac{\partial N_3}{\partial y} \end{array} \right]. \tag{2.34}$$

The first equation for the area integral over Ω_e can be written as

$$\int_{\Omega_e} \nabla \bar{\phi} \circ \nabla \phi dA = \int_{\Omega_e} \frac{\partial \bar{\phi}}{\partial x} \frac{\partial \phi}{\partial x} + \frac{\partial \bar{\phi}}{\partial y} \frac{\partial \phi}{\partial y} dA. \tag{2.35}$$

The first term of this equation can be written as

$$\int_{\Omega_e} \frac{\partial \bar{\phi}}{\partial x} \frac{\partial \phi}{\partial x} dA = \left[\begin{array}{ccc} \bar{\phi}_1 & \bar{\phi}_2 & \bar{\phi}_3 \end{array} \right] \int_{\Omega_e} \left[\begin{array}{c} \frac{\partial N_1}{\partial x} \\ \frac{\partial N_2}{\partial x} \\ \frac{\partial N_3}{\partial x} \end{array} \right] \left[\begin{array}{ccc} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial x} \end{array} \right] dA \left[\begin{array}{c} \phi_1 \\ \phi_2 \\ \phi_3 \end{array} \right]. \tag{2.36}$$

A similar expression can be obtained for the second term of this integral. Therefore, this integral can be represented as

$$\int_{\Omega_e} \nabla \bar{\phi} \circ \nabla \phi dA = [\bar{\phi}]^T [K^e] [\phi], \tag{2.37}$$

where $[K^e]$ is the element stiffness matrix given by

$$[K^e] = \int_{\Omega_e} \left[\begin{array}{ccc} (\frac{\partial N_1}{\partial x})^2 + (\frac{\partial N_1}{\partial y})^2 & \frac{\partial N_1}{\partial x}\frac{\partial N_2}{\partial x} + \frac{\partial N_1}{\partial y}\frac{\partial N_2}{\partial y} & \frac{\partial N_1}{\partial x}\frac{\partial N_3}{\partial x} + \frac{\partial N_1}{\partial y}\frac{\partial N_3}{\partial y} \\ \frac{\partial N_1}{\partial x}\frac{\partial N_2}{\partial x} + \frac{\partial N_1}{\partial y}\frac{\partial N_2}{\partial y} & (\frac{\partial N_2}{\partial x})^2 + (\frac{\partial N_2}{\partial y})^2 & \frac{\partial N_2}{\partial x}\frac{\partial N_3}{\partial x} + \frac{\partial N_2}{\partial y}\frac{\partial N_3}{\partial y} \\ \frac{\partial N_1}{\partial x}\frac{\partial N_3}{\partial x} + \frac{\partial N_1}{\partial y}\frac{\partial N_3}{\partial y} & \frac{\partial N_2}{\partial x}\frac{\partial N_3}{\partial x} + \frac{\partial N_2}{\partial y}\frac{\partial N_3}{\partial y} & (\frac{\partial N_3}{\partial x})^2 + (\frac{\partial N_3}{\partial y})^2 \end{array} \right] dA. \tag{2.38}$$

Using the assumed relations for the shape functions one has

$$\frac{\partial N_1}{\partial x} = b_1 \quad \frac{\partial N_1}{\partial y} = c_1 \tag{2.39}$$

$$\frac{\partial N_2}{\partial x} = b_2 \quad \frac{\partial N_2}{\partial y} = c_2 \tag{2.40}$$

$$\frac{\partial N_3}{\partial x} = b_3 \quad \frac{\partial N_3}{\partial y} = c_3 \tag{2.41}$$

Therefore $[K^e]$ will be given by

$$[K^e] = (area \ \ of \ \ \Omega_e) \begin{bmatrix} b_1^2 + c_1^2 & b_1c_2 + c_1c_2 & b_1b_3 + c_1c_3 \\ b_1b_2 + c_1c_2 & b_2^2 + c_2^2 & b_2b_3 + c_2c_3 \\ b_1b_3 + c_1c_3 & b_1b_3 + c_2c_3 & b_2^2 + c_2^2 \end{bmatrix}. \tag{2.42}$$

In a similar way the second area integral over Ω_e can be written as

$$2\int_{\Omega_e} G\theta\bar{\phi}dA = \begin{bmatrix} \bar{\phi}_1 & \bar{\phi}_2 & \bar{\phi}_3 \end{bmatrix} \begin{bmatrix} f_1^e \\ f_2^e \\ f_3^e \end{bmatrix}, \tag{2.43}$$

where

$$\begin{bmatrix} f_1^e \\ f_2^e \\ f_3^e \end{bmatrix} = 2G\theta \int_{\Omega_e} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} dA. \tag{2.44}$$

Substitution into equation (2.26) will give the equation

$$\sum_{e=1}^{NBE} \int_{\partial\Omega_e} \bar{\phi}\nabla\phi \circ \mathbf{n}ds - \{\bar{\phi}\}^T[K]\{\phi\} = -\{\bar{\phi}\}^T\{f\}, \tag{2.45}$$

where $[K]$ is the global stiffness matrix, $\{\bar{\phi}\}$ is the column vector of the values of the test function at the nodes, and $\{\phi\}$ is the column vector of the unknown values of ϕ at the nodes. Systematically setting the value of the test function at one node equal to unity and all the other nodes equal to zero will give the equation

$$[K]\{\phi\} = \{f\}, \tag{2.46}$$

where the elimination of the boundary terms will be discussed later.

The combination of the element stiffness matrix to obtain the global stiffness matrix is known as assembling the global stiffness matrix. Using the notation

- NN: number of nodes
- NE: number of elements
- X(I), Y(I): x and y coordinate of global node I
- XE(I), YE(I): x and y coordinates of local node number I
- IJK(I,J): connectivity of local node I of element J
- SKE(I,J): the (3,3) element stiffness matrix $[K^e]$
- SK(I,J): the (NN,NN) global stiffness matrix $[K]$
- FE(I): the (3,1) element load vector
- F(I): the (NN,1) global load vector

The program would be as follows

```
C
C     Do for each element
C
      DO N=1,NE
C
C        Extract the local node coordinates
C
         DO I=1,3
            XE(I)=X(IJK(I,N))
            YE(I)=Y(IJK(I,N))
         ENDDO
C
C        Calculate element stiffness matrix
C
         CALL STIFF(XE,YE,AREA,SKE)
C
C        Assemble SKE into SK
C
         DO I=1,3
            I1=IJK(I,N)
            DO J=1,3
               J1=IJK(J,N)
               SK(I1,J1)=SK(I1,J1)+SKE(I1,J1)
            ENDDO
         ENDDO
C
C        Calculate element load vector
C
         CALL FLOAD(XE,YE,AREA,FE)
C
C        Assemble FE into F
C
         DO I=1,3
            I1=IJK(I,N)
            F(I1)=F(I1)+FE(I1)
         ENDDO
      ENDDO
```

Before we proceed to solve the equations we need to impose the boundary conditions. Theoretically, we should eliminated the known values of $\phi = 0$ on the boundary nodes from the list of unknowns $\{\phi\}$ and introduce as new unknowns the values of $\nabla\phi \circ \mathbf{n}$ on the boundary nodes. To avoid this complex procedure, the *penalty method* will be used for imposing the

boundary condition $\phi = 0$ on $\partial\Omega$. This method modifies the stiffness matrix and vector to fix the value of ϕ at the boundary nodes to a given value $\hat{\phi}$. The method is based on the fact that for any boundary node i there will be an equation

$$K_{ii}\phi_i + \quad other \;\; terms \quad = f_i. \tag{2.47}$$

To impose the equation $\phi_i = \hat{\phi}$ one can add a term $pK_{ii}\phi_i$ to the left hand side of the equation and the term $pK_{ii}\hat{\phi}$ to the right hand side of the equation to get

$$K_{ii}\phi_i + pK_{ii}\phi_i \;\; other \;\; terms \quad = f_i + pK_{ii}\hat{\phi} \tag{2.48}$$

where p is a large number. When one divides this equation by pK_{ii} all term will be small accept the two added terms. Therefore, the equation will become dominated by the equation $\phi_i = \hat{\phi}$ and the other terms become unimportant. Since the method eliminates the importance of any other terms entering this equation, it will not be necessary to introduce the boundary integrals whenever one knows ϕ on the boundary. In short, to fix the value of ϕ at any node i one can add $P * SK(I,I)$ to $SK(I,I)$ and $P * SK(I,I) * BPHI$ to $F(I)$, where $BPHI$ is the known value of ϕ at the node, $\hat{\phi}$.

In this problem once the stiffness and load vector are modified one can solve the equation

$$[K]\{\phi\} = \{f\} \tag{2.49}$$

to get the values of ϕ at the nodes. Knowing these values one can also solve for the torque using the equation

$$T = 2\int_{\Omega} \phi dA = 2\sum_{e=1}^{NE} \int_{\Omega_e} \phi dA. \tag{2.50}$$

In problems which boundary conditions in the form of $\nabla\phi \circ \mathbf{n}$ are provided one must add the terms resulting from the boundary integrals into the load vector. In problems with mixed boundary conditions one makes the appropriate modification to the stiffness and load vector depending on the type of boundary condition.

Chapter 3

Programing Considerations in FEM

3.1 Organization of finite element method programs

Finite element method programs are normally organized in the following manner. This allows each part of the program to be a separate module which can be replaced from one application to another.

1. Division of the domain into elements, numbering the elements, numbering the nodes, providing the connection between local node numbers and global node numbers, providing the coordinate for each node.

2. Numbering boundary elements and connection between local node numbers and global node numbers.

3. Assembling global stiffness matrix, $[K]$, and load vector, $[f]$, element by element.

   (a) Calculate the element stiffness matrix, $[K^e]$, for each element and assemble it into the global stiffness matrix.

   (b) Calculate the element load vector, $[f^e]$, for each element and assemble it into the global load vector.

4. Modifying global stiffness matrix and global load vector to enforce boundary conditions.

5. Solving the resulting system of equations (i.e., $[K][\theta] = [f]$) for the value of the unknown function at each nodal point, $[\theta]$.

6. Presenting the results in a suitable format (post-processing).

This layout of the program provides a certain degree of flexibility. For example, the same problem can be solved using several different equation solvers, or the same program can solve different problems just by changing the method for calculating the element stiffness matrix, load vector, and boundary conditions.

3.2 Elements, nodes, connectivity, etc.

In the following sections we will develop different parts of the finite element program. To do this we will need to identify several variables and provide some conventions.

The following is a list of information normally provided to the program in the form of input data:

- Number of Elements (NE): The total number of elements in the problem
- Number of Nodes (NN): The total number of nodes in the problem
- Number of Nodes Per Element (NNPE): The number of nodes in each element
- Number of Boundary Elements (NBE): The total number of boundary elements
- Number of Nodes Per Boundary Element (NNPBE): The number of nodes in each boundary element
- Number of Coordinates (NC): The number of coordinates describing the problem (1 for 1-D problems, 2 for 2-D problems, etc.)
- Coordinates of Each node (X(NC,NN)): X(1,I) is the x-coordinate of node I, X(2,I) is the y-coordinate of node I, etc.
- Connectivity (IJK(NNPE,NE)): IJK(I,J) is the global node number of local node I of element J
- Boundary Element Connectivity (IJ(NNPBE,NBE)): IJ(I,J) is the global node number of local node I of boundary element J

The connectivity are used to allow calculation of the global node number using the element number and local node number. This will allow the evaluation of terms element by element and then assembling them into the global stiffness matrix and load vector, which are organized in terms of global node numbers.

The following are a list of variables which are used by the program, but are not provided in the input.

- Coordinates for local nodes (XE(NC,NNPE)): XE(1,1) is the x-coordinate of the first local node, XE(2,1) is the y-coordinate of the first local node, etc.
- Global Stiffness Matrix (SK(NN,NN)): The NN by NN global stiffness matrix
- Element Stiffness Matrix (SKE(NNPE,NNPE)): The NNPE by NNPE element stiffness matrix
- Global Load Vector (F(NN)): The global load vector
- Element Load Vector (FE(NNPE)): The element load vector

3.3 Assembling the global stiffness matrix and load vector

The following program can be used to assemble the global stiffness matrix and load vector. The program calculates the element stiffness matrix and load vector and assembles them into the global stiffness matrix and load vector one element at a time.

```
C
C     Assemble [K] and [f] element by element
C
      DO N=1,NE
C
C        Extract the local node coordinates
C
         DO I=1,NNPE
            DO J=1,NC
               XE(J,I)=X(J,IJK(I,N))
            ENDDO
         ENDDO
C
C        Calculate element stiffness matrix
C
         CALL STIFF(XE,SKE)
C
C        Assemble SKE into SK
C
         DO I=1,NNPE
            I1=IJK(I,N)
            DO J=1,NNPE
               J1=IJK(J,N)
               SK(I1,J1)=SK(I1,J1)+SKE(I,J)
            ENDDO
         ENDDO
C
C        Calculate element load vector
C
         CALL FLOAD(XE,FE)
C
C        Assemble FE into F
C
         DO I=1,NNPE
            I1=IJK(I,N)
            F(I1)=F(I1)+FE(I)
         ENDDO
```

```
      ENDDO
```

Note that in this program subroutines STIFF and FLOAD are used to calculate the element stiffness matrix and load vector. This provides the flexibility to use the same program but to change the problem by only changing these two subroutines, and appropriate terms for the boundary conditions.

For example, in the heat conduction problem of Chapter 1 the element stiffness matrix can be introduced using the following subroutine.

```
C
C     Subroutine to evaluate stiffness matrix
C
      SUBROUTINE STIFF(XE,SKE)
C
      REAL XE(1,2), SKE(2,2)
C
C     EPS = SMALL NUMBER
C
      EPS= 0.0000001
C
C     CTC = Coefficient of thermal conduction
C
      CTC=
C
      RL = ABS(XE(1,2)-XE(1,1))
      IF(RL.LE.EPS) THEN
         WRITE(*,*) 'ELEMENT LENGTH IS TOO SMALL '
         STOP
      ELSE
         TEM=CTC/RL
         SKE(1,1)=TEM
         SKE(1,2)=-TEM
         SKE(2,1)=-TEM
         SKE(2,2)=TEM
      ENDIF
C
      RETURN
      END
```

Note that in this program the coefficient of thermal conduction was included in the subroutine. Sometimes it is more convenient to read this information in the main program and transfer it to the subroutine through the arguments or a common block. This would allow changing of the coefficient without compiling the program.

In anticipation of heat input functions, $f(x)$, in the form of polynomials, Gaussian quadrature can be used for calculating the load vectors. The following might be used as a subroutine for calculating the load vector.

```
C
C     Subroutine to evaluate load vector
C
      SUBROUTINE FLOAD(XE,FE)
C
      REAL XE(1,2), FE(2)
C
C     Calculation of points for 5 point Gaussian Quadrature
C
      XAVE = (XE(1,2)+XE(1,1))/2.0
      RL = ABS(XE(1,2)-XE(1,1))/2.0
      XMIN = XAVE -RL
      X1 = XAVE - 0.906179846*RL
      X2 = XAVE - 0.538469310*RL
      X2 = XAVE
      X4 = XAVE + 0.538469310*RL
      X5 = XAVE + 0.906179846*RL
      FE(1) = 0.236926885*RN1(X1,XMIN,RL)*FUN(X1)
      FE(1) = FE(1) + 0.478628670*RN1(X2,XMIN,RL)*FUN(X2)
      FE(1) = FE(1) + 0.568888889*RN1(X3,XMIN,RL)*FUN(X3)
      FE(1) = FE(1) + 0.478628670*RN1(X4,XMIN,RL)*FUN(X4)
      FE(1) = FE(1) + 0.236926885*RN1(X5,XMIN,RL)*FUN(X5)
      FE(1) = FE(1)*RL
      FE(2) = 0.236926885*RN2(X1,XMIN,RL)*FUN(X1)
      FE(2) = FE(2) + 0.478628670*RN2(X2,XMIN,RL)*FUN(X2)
      FE(2) = FE(2) + 0.568888889*RN2(X3,XMIN,RL)*FUN(X3)
      FE(2) = FE(2) + 0.478628670*RN2(X4,XMIN,RL)*FUN(X4)
      FE(2) = FE(2) + 0.236926885*RN2(X5,XMIN,RL)*FUN(X5)
      FE(2) = FE(2)*RL
C
      RETURN
      END
C
C     *********************************
C
C     The heat input function
C
      FUNCTION FUN(X)
C
      FUN =
```

```
C
      RETURN
      END
C
C     ********************************
C
C     Shape function N1
C
      FUNCTION RN1(X,XMIN,RL)
C
      RN1 = 1.0 - (X-XMIN)/(2.0*RL)
C
      RETURN
      END
C
C     ********************************
C
C     Shape function N2
C
      FUNCTION RN2(X,XMIN,RL)
C
      RN2 = (X-XMIN)/(2.0*RL)
C
      RETURN
      END
```

Note that in this setup FUN is the heat input function, $f(x)$, which must be provided by the user.

3.4 Boundary conditions

There are many different boundary conditions which can be imposed. The form of these conditions are determined by the specifics of each problem. A common boundary condition is to prescribe the value of variable at a particular node. As described in the previous chapters, this can be done by altering both the stiffness matrix and load vector to make the desired constraint dominate all other terms.

The following program can be included in the main program to fix the value of θ at TBC(I) at node TBCN(I), where I=1,2,3, ..., NTBC, and NTBC is the number of nodes with temperature boundary conditions.

```
C
C     Modify SK and F for temperature boundary conditions
C
```

```
      P=1.0E20
      DO I=1,NTBC
         F(TBCN(I)) = F(TBCN(I))+P*SK(TBCN(I),TBCN(I))*TBC(I)
         SK(TBCN(I),TBCN(I)) = SK(TBCN(I),TBCN(I))*(1.0+P)
      ENDDO
C
```

We are now in the position to solve the heat conduction problem in chapter 2 with temperature boundary conditions at both ends of the bar. If there are heat flux boundary conditions, we must also modify the load vector F to impose these conditions. Consider the case when the value of the heat flux, q, is specified as HFBC(I) at node HFBCN(I), where I=1,2,3, ..., NHFBC, and NHFBC is the number of nodes with heat flux boundary conditions. The following program can be included in the main program to enforce the heat flux boundary conditions.

```
C
C     Modify F for heat flux boundary conditions
C
C     HFBC is positive for heat flowing out of the bar
C
      DO I=1,NHFBC
         F(HFBCN(I)) = F(HFBCN(I))-HFBC(I)
      ENDDO
C
```

3.5 Solving the system of equations

After modifying the global stiffness matrix and load vector to enforce the boundary conditions, one can solve for the unknowns by using a subroutine for solving a system of linear algebraic equations. Note that in most cases the stiffness matrix is sparse and it might be possible to find an efficient solver to handle this step.

3.6 General comments

As can be seen from the example presented in this section, each problem defines a different stiffness matrix, load vector, and boundary conditions. In spite of these differences, one can provide a systematic way of organizing most problems. This organization makes understanding programs simpler and can accelerate the process of developing programs based on the finite element method.

We have not yet explored the use of different element types. One can introduce higher order elements into the program as simply as two node elements were introduced.