A Report

On

Finite Element Method and its applications to Boundary Value Problems

Prepared in partial fulfillment

of

Study Project Course

MATH F266

By

Balaje K

Birla Institute of Technology and Science
K.K.Birla Goa Campus

Acknowledgements

Firstly I would like to thank ***Dr. Anil Kumar***, *Assistant Professor, Department of Mathematics*, Birla Institute of Technology and Science Pilani, K.K.Birla Goa Campus for providing me with an excellent guidance from the start to the end of this project.

I would like to thank the *Institution* for providing us with an excellent background and infrastructure to help me complete this project.

THANK YOU !!

Contents

Chapter 1

Introduction

1.1 Introduction

The finite element method is a numerical method like the finite difference method but is more general and powerful in its application to real-world problems that involve complicated physics, geometry, and boundary conditions.

In the finite element method, a given domain is viewed as a collection of subdomains, and over each subdomain the governing equation is approximated by any of the traditional variational methods. The main reason behind seeking approximate solution on a collection of subdomains, is the fact that it is easier to represent a complicated function as a collection of simple polynomials.

1.2 A Simple Problem

Consider the problem of finding the perimeter of a circle of radius R. A well known formula to find the perimeter or the circumference of the circle is $2\pi R$. The same can be done by using **Finite Element Analysis**. We take this example to illustrate the basic idea behind **Finite Element Analysis**.

One can readily see that an obvious way to do that is to construct straight lines that approximate the circle, measure the length of each straight lines and sum them up. Let us formally describe the steps involved in the procedure, and a few terms that will be used frequently in this report.

1. **Discretization of the domain**:
   Our domain i.e., the perimeter of the circle is divided into a finite number of say, n subdomains. These subdomains(*elements*) are the line segments that approximate the circle. This process is known as *Discretization of the domain*. This collection of subdomains in known as the *finite element mesh*. The elements are connected to each other in points known as *nodes*. If the size of each element is the same, then the mesh is known as *uniform mesh*, otherwise it is known as *non-uniform mesh*.

2. **Element Equations:**
   In this step we isolate one of the elements and construct the equations that describes the solution to our problem in that element. Let us denote each of our subdomain in the example

by Ω_e.
The length of each subinterval (i.e., subdomain) is given by

$$h_e = 2R\ sin\left(\frac{1}{2}\theta_e\right) \tag{1.2.1}$$

where R is the radius of the circle, and θ_e is the angle subtended by the line segment on the center of the circle.

3. **Assembly of element equations and the solution:**
This step is the most crucial step to obtain the solution to our problem. The equations obtained from the above step must be assembled in a meaningful way to get the solution. Let us now illustrate this step for our example.
Since we have assumed uniform meshing, we have $\theta_e = \frac{2\,\pi}{n}$, where n is the number of elements. Using equation (3.1.1), we have,

$$h_e = 2R\ sin\left(\frac{\pi}{n}\right) \tag{1.2.2}$$

and the **approximate perimeter** P_n,

$$P_n = \sum_{e=1}^{n} h_e = n\left(2R\ sin\frac{\pi}{n}\right) \tag{1.2.3}$$

We now observe that, as the number of elements n increases, our expression (1.2.3) tend to the original formula for the perimeter of the circle. Since, this is an approximation to the actual value, there is an error in the computation, for which we have the last step.

4. **Convergence and error estimate:** The local error in approximating the length of the segment, is given by the expression,

$$E_e = |S_e - h_e| = R\left(\frac{2\pi}{n} - 2\ sin\frac{\pi}{n}\right) \tag{1.2.4}$$

where $S_e = R\,\theta_e$. From (1.2.4), we have our global error,

$$E = n\ E_e = 2\ R\left(\pi - n\ sin\frac{\pi}{n}\right) \tag{1.2.5}$$

As $n \to \infty$ and taking $x = \frac{1}{n}$, we have,

$$P_n = 2R\ n\ sin\frac{\pi}{n} = 2R\ \frac{sin\ \pi x}{x}$$

and

$$\lim_{n\to\infty} P_n = \lim_{x\to 0}\left(2R\ \frac{sin\ \pi x}{x}\right) = \lim_{x\to 0}\left(2\pi R\ \frac{cos\ \pi x}{1}\right) = 2\pi R \tag{1.2.6}$$

Thus we have shown that P_n converges to the exact value of the perimeter.

The procedure illustrated above is generally the same for all finite element problems. This can be convieniently implemented in computers to find approximate solutions. Before starting the analysis of boundary value problems, let us first describe some basic concepts that will be used to build up further discussion.

1.3 Preliminaries

1.3.1 Weighted Integral Statements

Consider the following second order differential equation ,

$$-\frac{d}{dx}\left[a(x)\frac{du}{dx}\right] + c(x)u = f(x), \quad 0 < x < L \tag{1.3.1}$$

subject to the following boundary conditions

$$u(0) = u_0, \quad \left[a(x)\frac{du}{dx}\right]_{x=L} = Q_0 \tag{1.3.2}$$

This particular form of the differential equation is of great importance in engineering applications viz. Heat transfer problems, deflections of a bar or cable etc. The functions $a(x)$, $c(x)$ and $f(x)$ can be taken accordingly.

We are now interested to find an approximate solution of the form,

$$U_N = \sum_{j=1}^{N} c_j \phi_j(x) + \phi_0(x) \tag{1.3.3}$$

The functions $\phi_j(x) : j = 0, 1, 2 \ldots N$ are known functions. The choice of these functions are entirely dependent on the boundary conditions of the problem. The function $\phi_0(x)$ is known as the non-homogeneous part of the approximate solution and $\sum c_j \phi_j(x)$ is known as the homogeneous part of the solution. The $\phi_j(x) : j = 1, 2 \ldots N$ must satisfy the homogeneous boundary condition while the function $\phi_0(x)$ must satisfy the specified boundary conditions.

Thus to obtain the complete solution, we must determine c_j so that the approximate solution satisfies the boundary value problem. The most obvious way is to take the approximate solution and substitute it into the differential equation and obtain the values of c_j. But this does not guarantee a solution as the system of equations (in c_j) that we get may be inconsistent. Thus we need a new way to obtain a consistent system.

Consider the approximate solution to satisfy the differential equation in a weighted integral sense i.e.,

$$\int_0^L w(x)\, R\, dx = 0 \tag{1.3.4}$$

$$R \equiv -\frac{d}{dx}\left[a(x)\frac{dU_N}{dx}\right] + c(x)U_N - f(x) \tag{1.3.5}$$

where R is known as the residual and $w(x)$ is known as the **weight function**. This guarantees a set of linearly independent system of equations as long as the choices of $w(x)$ is linearly independent, eg. $w(x) = 1, x, x^2$. The number of choices of $w(x)$ must be restricted to N so as to get equal number of equations and unknowns.

1.3.2 Linear and Bilinear functionals

A function $B(u)$ is called a linear function if and only if it satisfies the following condition.

$$B(\alpha u + \beta v) = \alpha B(u) + \beta B(v). \tag{1.3.6}$$

A functional is a mapping that maps functions to real numbers. Some examples of a functional are given below.

$$\begin{aligned} I(u,v) &= \int_a^b (u^2 + v + 1)dx \\ I(\phi,\psi) &= \int_a^b \left(\frac{d\phi}{dx}\frac{d\psi}{dx} + c(x)\ \phi\ \psi\right) dx \end{aligned} \tag{1.3.7}$$

Also equation (1.3.7) is a **symmetric functional**, as $I(\phi,\psi) = I(\psi,\phi)$

A functional $B(u,v)$ is bilinear if it is linear in both the variables, i.e., if

$$\begin{aligned} B(\alpha u_1 + \beta u_2, v) &= \alpha B(u_1, v) + \beta B(u_2, v) \\ B(u, \alpha v_1 + \beta v_2) &= \alpha B(u, v_1) + \beta B(u, v_2) \end{aligned} \tag{1.3.8}$$

An example of a bilinear functional is

$$B(u,v) = \int_a^b \left(p(x)uv + q(x)\frac{du}{dx}\frac{dv}{dx}\right) dx$$

1.3.3 Construction of weak form of a Boundary Value Problem

Consider the same boundary value problem that we considered in (1.3.1). Consider the differential equation (1.3.1) to vanish in the weighted integral sense, i.e.,

$$0 = \int_a^b w(x) \left(-\frac{d}{dx}\left[a(x)\frac{du}{dx}\right] + c(x)u - f(x)\right) \tag{1.3.9}$$

Applying integration by parts on the first term of 1.3.9, we get

$$0 = \int_a^b \left(a\frac{dw}{dx}\frac{du}{dx} + w\ u\ c(x) - wf(x)\right) dx - \left[w\ a\frac{du}{dx}\right]_0^L \tag{1.3.10}$$

The next step is to identify the type of boundary conditions. There are two types of boundary condition *essential* and *natural*. First examine all the boundary terms in the integral statement 1.3.9. It contains both the weight function w and the dependent variable u. The coefficient of the weight function is termed as the **secondary variable** and the dependent variable expressed in the same form as the weight function in the boundary term is known as the **primary variable.** Specification of the primary variables in the boundary conditions is known as the *essential boundary condition* and the specification of the secondary valriables in the boundary condition is termed as the *natural boundary condition.*

Define the direction cosine (the cosine of the angle between the positive x-axis and the normal to the boundary) n_x where $n_x = -1$ on the left side of the domain and $n_x = 1$ on the right hand side of the domain. Now (1.3.9) can be rewritten as,

$$\begin{aligned} 0 &= \int_a^b \left(a\frac{dw}{dx}\frac{du}{dx} + w\ u\ c(x) - wf(x) \right) dx - \left[w\ a\frac{du}{dx}n_x \right]_{x=0} - \left[w\ a\frac{du}{dx}n_x \right]_{x=L} \\ 0 &= \int_a^b \left(a\frac{dw}{dx}\frac{du}{dx} + w\ u\ c(x) - wf(x) \right) dx - (wQ)_0 - (wQ)_L \end{aligned} \tag{1.3.11}$$

The last step is to apply the boundary conditions. In weak formulations, the meaning of weight function is that the virtual *change* in the primary variable. So if the primary variable is specified at a point, the the weight function vanishes at that point. So for the boundary condtions 1.3.2, the weight function vanishes at $x = 0$, and 1.3.11 becomes,

$$0 = \int_a^b \left(a\frac{dw}{dx}\frac{du}{dx} + w\ u\ c(x) - wf(x) \right) dx - (wQ)_L \tag{1.3.12}$$

The equation (1.3.12) is the **weak form** of the boundary value problem (1.3.1), (1.3.2). The word "weak" refers to the weakened continuity of u, which was required to be twice-differentiable in (1.3.1) but once differentiable in (1.3.12). Every differential equation of order two or more has a weak form.

Chapter 2

Variational Methods

2.1 Introduction

The main goal of this article is to illustrate how finite element method can be used to solve a boundary value problem. The variational methods provide us with a strong foundation with which our discussion on Finite Element Analysis can be built on. Moreover, elementwise application of the variational methods is nothing but the finite element method.
There are a number of variational methods that is commonly used in solving a Boundary value problem. Some of them are **Ritz method**, **weighted residual methods** like the Galerkin Method, Petrov-Galerkin method, Least Squares method etc. In this chapter we will be discussing the working of each method.

2.2 Ritz method

Consider the following boundary value problem,

$$-\frac{d}{dx}\left(a(x)\frac{du}{dx}\right) = f(x) \quad 0 < x < L \tag{2.2.1}$$

subject to the boundary conditions.

From the weak formulation of the above problem, let

$$B(w,u) = \int_0^L \left(\frac{dw}{dx}\frac{du}{dx}\right) dx \tag{2.2.2}$$

$$l(w) = \int_0^L w\ f\ \ dx + w(L)Q_L \tag{2.2.3}$$

Thus weak form of 2.2.1 can be written as $B(w,u) = l(w)$.

Suppose let

$$U_N = \sum_{j=1}^{N} c_j\phi_j(x) + \phi_0(x) \tag{2.2.4}$$

where ϕ_0 satisfies the specified boundary conditions, and ϕ_j satisfies the homogeneous form of the boundary conditions. Taking $w = \phi_j$ and substituting U_N in $B(w,u) = l(w)$, we have

$$B(\phi_i, \sum c_j\phi_j + \phi_0) = l(\phi_i) \qquad i = 1, 2 \ldots, N \tag{2.2.5}$$

Since $B(.,.)$ is bilinear we have

$$\sum_{j=1}^{N} B(\phi_i, \phi_j)c_j + \phi_0 = l(\phi_i) - B(\phi_i, \phi_0) \quad i = 1, 2 \ldots, N \tag{2.2.6}$$

$$\sum_{j=1}^{N} K_{ij}c_j = F_i \quad i = 1, 2 \ldots, N \tag{2.2.7}$$

where $K_{ij} = B(\phi_i, \phi_j)$ and $F_i = l(\phi_i) - B(\phi_i, \phi_0)$.

The equation 2.2.7 can be expressed as a system of linear equations,

$$[K][c] = \{f\} \tag{2.2.8}$$

The choice of ϕ_i's are not arbitrary and must satisfy a set of conditions. The main drawback of the Ritz method is that the choice of ϕ_i's are not always easy to find.

1. Each ϕ_i's must be defined so that $B(\phi_i, \phi_j)$ is nonzero and defined. ϕ_i must satisfy the homogeneous form of the essential boundary condtion specified.
2. For any N the set $\{\phi_i\}_{i=1}^{N}$ along with the columns (and rows) of $B(\phi_i, \phi_j)$ must be *linearly independent.*
3. The set ϕ_i must be complete. For example, when ϕ_i are algebraic polynomials, completeness requires that the set ϕ_i contains all the terms of the lowest order admissible, up to the highest order desired.
4. The only requirement on ϕ_0 is that it must satisfy the specified essential boundary conditions. Also ϕ_0 must be the lowest order function that satisfies the specified essential boundary condtions.

2.3 Weighted Residual Methods

As the name suggests, weighted residual methods involve the use of weighted integral form which was discussed in **Section 1.2.1** in the previous chapter. Let us recall the weighted integral form of a differential equation.

$$\int_a^b w(x) \;\; R \;\; dx = 0 \tag{2.3.1}$$

where R is known as the residual. This method is very similar to the Ritz method. Like in the Ritz method, we seek an approximate solution to the problem which is of the same form as discussed in 2.2.4.

The conditions on the choice of weight functions is more stringent than the Ritz Method. In the weighted residual method, the functions ϕ_j must have differentiability as dictated by the order of the differential equation (because it does not use weak form). The functions ϕ_j must satisfy the homogeneous boundary conditions and must be linearly independent. Similarly the function ϕ_0 must satisfy the specified essential boundary condition.

Weighted residual methods are classified based on the choice of weight functions $w(x)$. Some popular weighted residual methods are summarized in the table below.

S.No	Method	Choice of weight function
1.	Petrov-Galerkin	$w(x) = \psi_i(x) \neq \phi_i(x)$
2.	Galerkin	$w(x) = \phi_i(x)$
3.	Least Squares	$w(x) = \frac{\partial R}{\partial c_i}$
4.	Collocation	$w(x) = \delta(x - x_i)$, δ is the **Dirac Delta** function

The main disadvantage in all these variational methods is the difficulty in constructing the approximation functions. A lot of conditions are imposed on the selection of the approximation function viz., continuity, differentiability, completeness etc. The process becomes incresingly complex as we when we encounter geometrically complex domain. Finite Element Analysis overcomes all these shortcomings by providing us with a systematic way of constructing approximation functions.

Chapter 3

Finite element model of second order differential equations in one dimension

3.1 Introduction

So far, we have seen how traditional variational methods like Ritz, weighted residual methods like Galerkin Method can be used to solve a Boundary Value Problem. In this chapter, we discuss how a finite element model is constructed and used to solve a BVP.

In this chapter, we consider a general form a second order ordinary differential equation of the form,

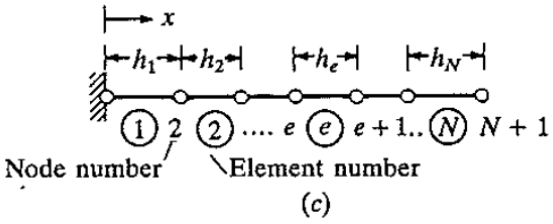
$$p(x)\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} + r(x)y = f(x) \qquad a < x < b \tag{3.1.1}$$

where y is a function of x, along with the boundary condtions. Let us attempt to derive the finite element model of this second order ODE.

3.2 Finite Element Model

3.2.1 Discretization of the domain

Figure 3.1: Discretization of an interval



The domain in the figure is divided into N subintervals of sizes $h_1, h_2, \ldots, h_N$ respectively. We have $N + 1$ node points. This is an example of how a linear domain is divided into subdomains. Each subdomain may or may not have equal size. Usually during analysis, throughout the chapter, we consider *uniform meshing*, i.e., subintervals of equal sizes. Before proceeding to the next section consider the following equations.

The size of each subdomain h_e,

$$h_e = x_{e+1} - x_e \tag{3.2.1}$$

In case of uniform length of subintervals,

$$h_e = \frac{b-a}{N} = \frac{x_{N+1} - x_1}{N} \tag{3.2.2}$$

The words subdomain and subintervals are used interchangably in this chapter.

3.2.2 Constructing weak form

Next, we construct the element equation for each element.

Dividing (3.1.1) by $p(x)$ throughout, we have,

$$\frac{d^2u}{dx^2} + \frac{q(x)}{p(x)}\frac{du}{dx} + \frac{r(x)}{p(x)}u = \frac{f(x)}{p(x)} \tag{3.2.3}$$

Writing the weighted integral form of the differential equation 3.2.3

$$\int_a^b w(x) \quad \left(\frac{d^2u}{dx^2} + \frac{q(x)}{p(x)}\frac{du}{dx} + \frac{r(x)}{p(x)}u - \frac{f(x)}{p(x)}\right) dx = 0 \tag{3.2.4}$$

Using integration by parts for the first term alone,

$$\int_a^b \left(-\frac{dw}{dx}\frac{du}{dx} + w\frac{q(x)}{p(x)}\frac{du}{dx} + w\frac{q(x)}{p(x)}u - w\frac{f(x)}{p(x)}\right) dx + \left[w\frac{du}{dx}\right]_a^b = 0 \tag{3.2.5}$$

(3.2.5) is the weak form corresponding to the differential equation (3.1.1) Let

$$F(w,u) = \int_a^b \left(-\frac{dw}{dx}\frac{du}{dx} + w\frac{q(x)}{p(x)}\frac{du}{dx} + w\frac{q(x)}{p(x)}u\right) dx \tag{3.2.6}$$

$$G(w) = \int_a^b \left(w\frac{f(x)}{p(x)}\right) dx + \left[w\frac{du}{dx}\right]_a^b \tag{3.2.7}$$

3.2.3 Approximation Functions

The main idea behind finite element method is to construct functions or curves that approximate the original curve on the **subdomain**. These approximate functions then are used to compute numerical solutions to the problem. To construct these **interpolation functions**, we use the **Lagrange interpolating polynomials**.

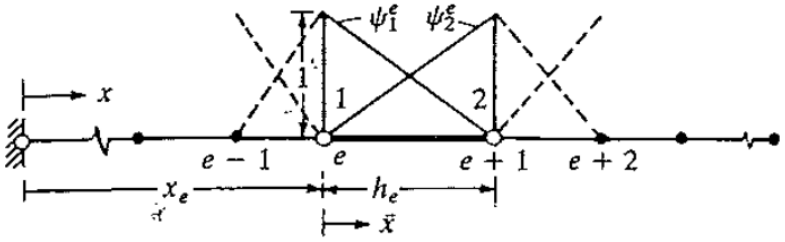
Linear Interpolation functions: For any subdomain, say (x_e, x_{e+1}) in the domain, **linear approximation functions** ψ_I^e and ψ_2^e are constructed as shown in the figure.
If u_1^e and u_2^e are the values of the functions at node 1 and 2 respectively, the approximation of the original curve on the subdomain is given by $u_h^e = u_1^e\psi_1^e + u_2^e\psi_2^e$. Repeat this over all the subdomains and we get the entire curve. Our aim, here is to find the values of u_1^e and u_2^e.

Setting the local coordinate system $\bar{x}$ at node 1 as shown, we have our Linear approximation functions ψ_1^e and ψ_2^e as

$$\psi_1^e = 1 - \frac{\bar{x}}{h_e} \tag{3.2.8}$$

$$\psi_2^e = \frac{\bar{x}}{h_e} \tag{3.2.9}$$

Figure 3.2: Constructing linear interpolating functions



Quadratic Interpolation Functions: As we have seen that in case of Linear Interpolation functions, we require two local nodes to describe the polynomial. In case of Quadratic Interpolation Functions, we require three local nodes and hence we have three interpolation functions. The approximation functions are given by:

$$\psi_1^e = \left(1 - \frac{\bar{x}}{h}\right)\left(1 - \frac{2\bar{x}}{h}\right) \tag{3.2.10}$$

$$\psi_2^e = 4\left(\frac{\bar{x}}{h}\right)\left(1 - \frac{\bar{x}}{h}\right) \tag{3.2.11}$$

$$\psi_3^e = -\left(\frac{\bar{x}}{h}\right)\left(1 - \frac{2\bar{x}}{h}\right) \tag{3.2.12}$$

3.2.4 Element equations

Our next task is to derive the equations or rather a system of equations, that describe each element or subinterval in the model. In this article, we use the Ritz procedure to develop the finite element model. Consider the weak form of the differential equation (3.2.5). We take

$$u = \sum_{j=1}^{n} u_j^e \psi_j^e \tag{3.2.13}$$

$$w = \psi_i^e \tag{3.2.14}$$

Using this in (3.2.5) we get,

$$\sum_{j=1}^{n} K_{ij}^e u_j^e = f_i^e + Q_i^e \qquad i = 1, 2 \ldots n \tag{3.2.15}$$

where,

$$K_{ij}^e = \int_a^b \left(-\frac{d\psi_i^e}{dx}\frac{d\psi_j^e}{dx} + \frac{p(x)}{q(x)}\psi_i^e\frac{d\psi_j^e}{dx} + \frac{r(x)}{p(x)}\psi_i^e\psi_j^e\right) dx \tag{3.2.16}$$

$$f_i^e = \int_a^b \psi_i^e \frac{f(x)}{p(x)}\, dx \tag{3.2.17}$$

and Q_i^e denote the secondary variables. This concludes the construction of element equations. The matrix K_{ij}^e is usually called the stiffness matrix in structural analysis and f_i^e is called the source/force vector. The value n denotes the *local degrees of freedom* of the element. To solve this system, we need an extra $2n$ conditions. So we assemble these element matrices to find the solution at each nodes.

3.3 Assembling the element equations

In this section we assemble each element matrices to obtain the solutions at global nodes. If e is the number of elements in the domain, n, the local degrees of freedom, then the **global degrees of freedom** N is given by

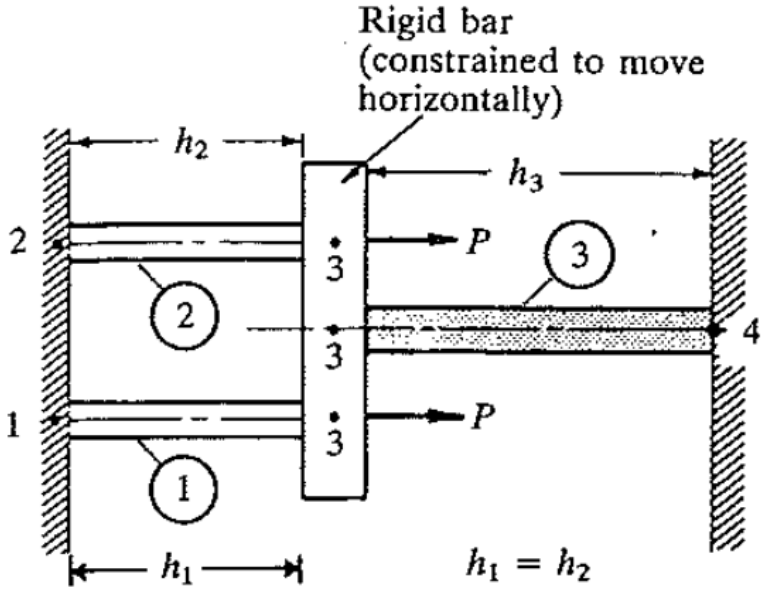
$$N = (n-1)e + 1 \tag{3.3.1}$$

Connectivity Matrix: The connectivity matrix is a very powerful tool, used to assemble the element equations. Denoted by

$$B = [b_{ij}]_{m \times n} \tag{3.3.2}$$

where b_{ij} is the global node number corresponding to the local node number j of element i. Thus, m is the number of elements present in the domain and n is the number of local nodes per element.

Figure 3.3: Connectivity of elements



Example The connectivity matrix for the system shown above is given by

$$\begin{bmatrix} 1 & 3 \\ 2 & 3 \\ 3 & 4 \end{bmatrix}$$

Let the assembled matrix be denoted by $K = K_{ij}$. Each element of K_{ij} is obtained from the connectivity matrix as follows

$$K_{11} = K_{11}^1 \quad K_{33} = K_{22}^1 + K_{22}^2 + K_{11}^3 \quad K_{23} = K_{12}^2 \quad K_{34} = K_{12}^3 \quad K_{12} = 0 \quad K_{24} = 0 \tag{3.3.3}$$

and so on. ■

Similarly the entire assembled system for the problem that we are discussing can be deduced. The connectivity matrix for the problem is given by,

$$\begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ \vdots & \vdots \end{bmatrix}$$

The final assembled matrices for the model using linear interpolation functions is given by

$$\begin{bmatrix} K_{11}^1 & K_{12}^1 & 0 & \dots & 0 \\ 0 & K_{22}^1 + K_{11}^2 & K_{12}^2 & \dots & 0 \\ 0 & K_{21}^2 & K_{22}^2 + K_{11}^3 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \dots & K_{22}^N \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_N \end{Bmatrix} = \begin{Bmatrix} f_1^1 \\ f_2^1 + f_1^2 \\ f_2^2 + f_1^3 \\ \vdots \\ f_2^N \end{Bmatrix} + \begin{Bmatrix} Q_1^1 \\ Q_2^1 + Q_1^2 \\ Q_2^2 + Q_1^3 \\ \vdots \\ Q_2^N \end{Bmatrix} \tag{3.3.4}$$

The connectivity matrix is used mainly for computer implementations of the finite element problems. A sample program is run for the following problems and results are computed and plotted against the exact solution.

3.4 Imposition of boundary condtions

Boundary conditons are classified into three types, namely

1. Dirichlet Boundary Condition
2. Neumann Boundary Condition
3. Mixed or Robin Boundary Conditions.

In case of Dirichlet boundary condition, the primary variables will be specified at the boundaries, while in case of Neumann boundary conditions, secondary variables will be specified in either one or both of the boundaries. In mixed boundary conditions, a "mixture" of primary and secondary boundary condtions are prescribed at the boundary.

In Finite element method to apply boundary conditions of the *Dirichlet* type, the assembled matrix system should be adjusted to include the values, specified at the boundary. For example if the boundary conditions are, $u(a) = u_0$ and $u(b) = 0$. Then the last row and column of the assembled system must be removed, whereas the first nodal value is to be set as u_0. Thus the system (3.3.4) becomes,

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & K_{22}^1 + K_{11}^2 & K_{12}^2 & \dots & \\ 0 & K_{21}^2 & K_{22}^2 + K_{11}^3 & \dots & \\ \vdots & \vdots & \vdots & & \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ \vdots \\ U_{N-1} \end{Bmatrix} = \begin{Bmatrix} u_0 \\ f_2^1 + f_1^2 \\ \vdots \\ f_2^{N-1} \end{Bmatrix} + \begin{Bmatrix} 0 \\ Q_2^1 + Q_1^2 \\ \vdots \\ Q_2^{N-1} \end{Bmatrix} \tag{3.4.1}$$

In Finite element method to apply boundary conditions of the *Neumann* type, the assembled matrix system should be adjusted to include the values, specified at the boundary. For example if the boundary conditions are, $u(a) = 0$ and $u'(b) = 1$. Then the first row and column of the assembled system must be removed, whereas the last value in the $\{Q\}$ matrix is to be set as 1. Thus the system (3.3.4) becomes,

$$\begin{bmatrix} K_{12}^1 & 0 & \dots & 0 \\ K_{22}^1 + K_{11}^2 & K_{12}^2 & \dots & 0 \\ K_{21}^2 & K_{22}^2 + K_{11}^3 & \dots & 0 \\ \vdots & \vdots & \vdots & \\ 0 & 0 & \dots & K_{22}^N \end{bmatrix} \begin{Bmatrix} U_2 \\ U_3 \\ \vdots \\ U_N \end{Bmatrix} = \begin{Bmatrix} f_2^1 + f_1^2 \\ f_2^2 + f_1^3 \\ \vdots \\ f_2^N \end{Bmatrix} + \begin{Bmatrix} Q_2^1 + Q_1^2 \\ Q_2^2 + Q_1^3 \\ \vdots \\ 1 \end{Bmatrix} \tag{3.4.2}$$

To apply boundary conditions of the *Mixed* type, the assembled matrix system should be adjusted to include the values, specified at the boundary. For example if the boundary conditions are, $u(a) = 0$ and $u'(b) + 2 \;\; u(b) = 1$. Then the first row and column of the assembled system must be removed, whereas the coefficient of $u(b)$ should be added to the last element of the K matrix. Thus the system (3.3.4) becomes,

$$\begin{bmatrix} K_{12}^1 & 0 & \dots & 0 \\ K_{22}^1 + K_{11}^2 & K_{12}^2 & \dots & 0 \\ K_{21}^2 & K_{22}^2 + K_{11}^3 & \dots & 0 \\ \vdots & \vdots & \vdots & \\ 0 & 0 & \dots & K_{22}^N + 2 \end{bmatrix} \begin{Bmatrix} U_2 \\ U_3 \\ \vdots \\ U_N \end{Bmatrix} = \begin{Bmatrix} f_2^1 + f_1^2 \\ f_2^2 + f_1^3 \\ \vdots \\ f_2^N \end{Bmatrix} + \begin{Bmatrix} Q_2^1 + Q_1^2 \\ Q_2^2 + Q_1^3 \\ \vdots \\ 1 \end{Bmatrix} \tag{3.4.3}$$

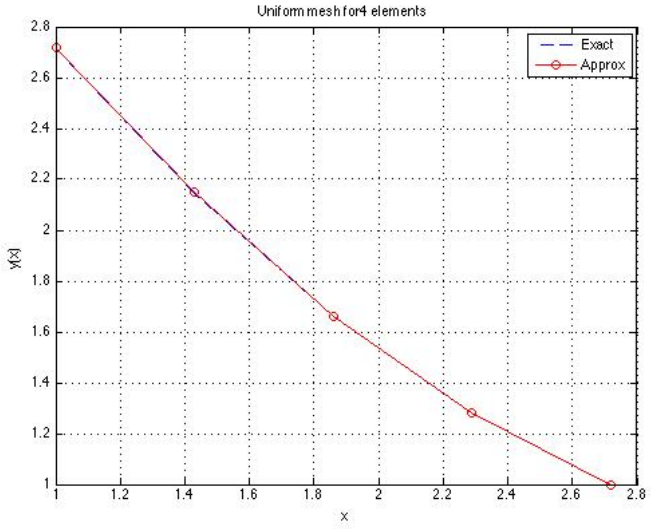
3.5 Examples

Example 1:

$$x^2y'' + xy' + x = 1 \tag{3.5.1}$$

subject to the conditions,

$$y(1) = e \quad y(e) = 1 \tag{3.5.2}$$



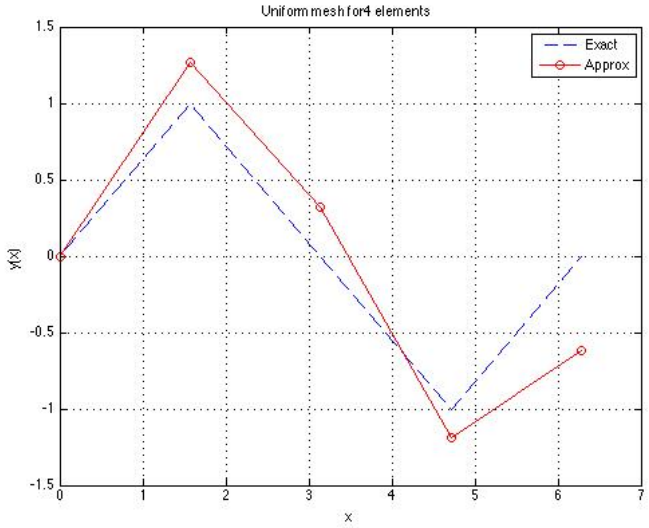
■

Example 2:

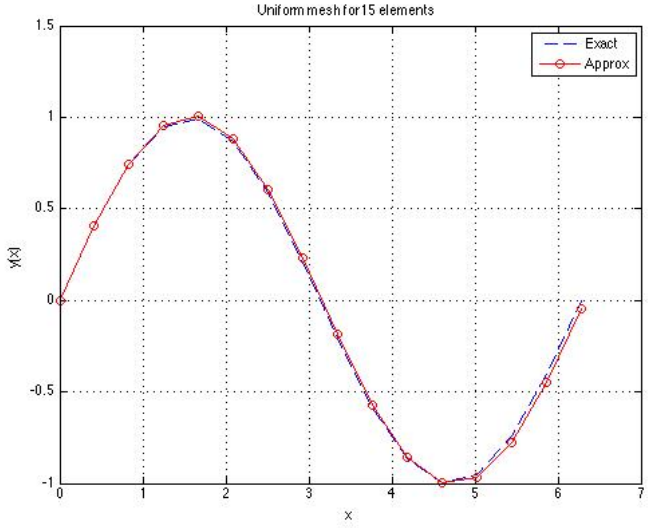
$$y'' + y = 0 \tag{3.5.3}$$

subject to the condtions,

$$y(0) = 0 \quad y'(2\pi) = 1 \tag{3.5.4}$$



The exact solution for this BVP is $y(x) = sin(x)$. The result shown above is when an uniform mesh of four elements are used to compute the solution. From the graph, it is clear that the solution is not very accurate. The solution can be improved by increasing the number of elements used which is evident from the next plot.

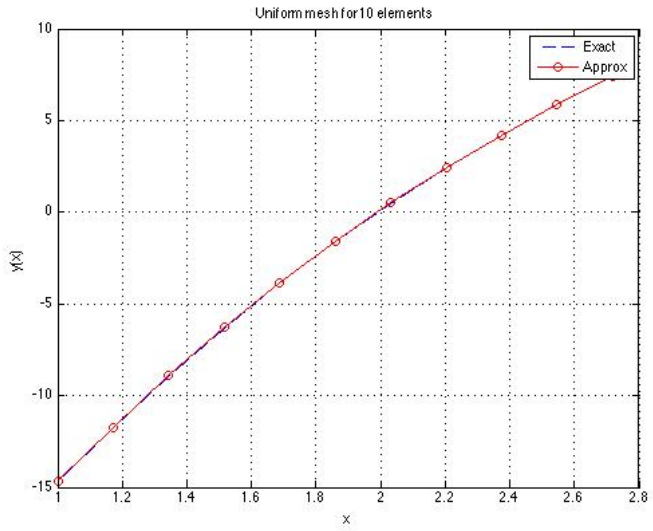


Example 3:

$$x^2y'' + xy + y = x \tag{3.5.5}$$

subject to the condtions,

$$y(1) + y'(1) = e \quad y(e) = e^2 \tag{3.5.6}$$



This chapter marks the end of our discussion about the finite element analysis if one dimensional boundary value problems. The next chapter deals with FEA of time depedent IBVPs. ■

Chapter 4

Time dependent problems

4.1 Introduction

The finite element model of time dependent problems are mainly done in two ways. One way is to consider time variable as a coordinate and choosing approximate functions of the form

$$u(x,t) \approx u_h(x,t) = \sum_{j=1}^{n} u_j^e \psi_j^e(x,t) \tag{4.1.1}$$

Other way is to separate the solution into two variables, one dependent on space and the other on time. This would lead us to an approximation of the form,

$$u(x,t) \approx u_h(x,t) = \sum_{j=1}^{n} u_j^e(t) \psi_j^e(x) \tag{4.1.2}$$

Equation (4.1.1) is known as the **coupled formulation** and (4.1.2) is called the **decoupled formulation.** The values $u_j^e(t)$ are computed at the nodes and are time dependent functions. So to find the values of u at different times, we use iterative methods such as **Galerkin Method, Crank Nicolson Method** etc. In this chapter, we discuss the finite element analysis of the decoupled form and to introduce the methods that are widely used to solve time dependent problems.

4.2 Semidiscretized form

For simplicity, let us take a basic initial boundary value problem,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad 0 \le x \le 1, \quad t \ge 0 \tag{4.2.1}$$

subject to the initial conditions

$$u(x,0) = 1$$

and the boundary conditions

$$u(0,t) = 0 \quad \frac{\partial}{\partial t} u(1,t) = 0$$

Writing the weighted integral form of the above problem we have,

$$0 = \int_0^1 w \left(\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} \right) dx \tag{4.2.2}$$

Using integration by parts on the second term, and simplifying we have the weak form,

$$0 = \int_0^1 \left(w\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\frac{\partial w}{\partial x} \right) dx - \left[w\frac{\partial u}{\partial x} \right]_0^1 \tag{4.2.3}$$

With the same notations and using the Galerkin finite element model that was discussed in the previous chapter, we have $w = \psi_i^e$ and using (4.1.2) $u = \sum u^e(t)\psi^e(x)$, we get,

$$0 = \sum_{j=0}^{n} \int_0^1 \left(\psi_i\psi_j \frac{\partial u_j(t)}{\partial t} + \frac{d\psi_i}{dx}\frac{d\psi_j}{dx}u_j(t) \right) dx - Q_i^e \tag{4.2.4}$$

Writing in matrix form we get,

$$[M^e]\{\dot{u}\} + [K^e]\{u\} = \{Q\} \tag{4.2.5}$$

$$M^e = \int_0^1 (\psi_i\psi_j)\, dx \tag{4.2.6}$$

$$K^e = \int_0^1 \left(\frac{d\psi_i}{dx}\frac{d\psi_j}{dx} \right) dx \tag{4.2.7}$$

Equation (4.2.7) is called the **semidiscrete form** of the model. With choice of appropriate interpolation functions, element equations are obtained. The next step is to obtain a iterative procedure to find the values of u at different values of t at the nodes.

4.3 Time Marching Scheme

The α *family of approximation* is a method of approximating derivatives that are obtained by taking a weighted mean of time derivatives at two consecutive time steps.

$$(1-\alpha)\dot{u}_s + (\alpha)\dot{u}_{s+1} = \frac{u_{s+1} - u_s}{\Delta t_{s+1}} \tag{4.3.1}$$

The same idea can be extended to matrix system. The final **time-marching** scheme obtained by appropriate substitution is given by,

$$\begin{aligned}([M^e] + \Delta t\ \ \alpha[K^e])\{u^e\}_{s+1} = ([M^e] - \Delta t\ \ (1-\alpha)[K^e])\{u^e\}_s + \\ \Delta t\ \ (\alpha\{Q^e\}_{s+1} + (1-\alpha)\{Q^e\}_s) \quad 0 \le \alpha \le 1\end{aligned} \tag{4.3.2}$$

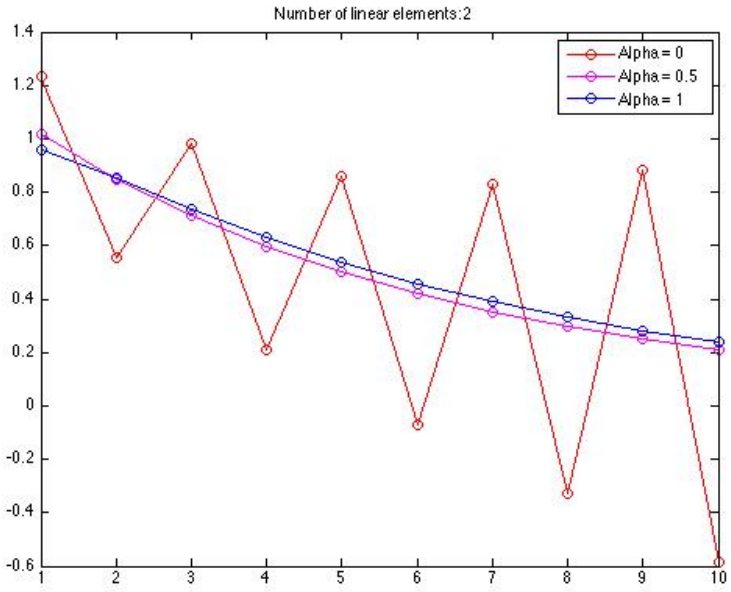
The process of assembly and the imposition of boundary condtions remains the same. This gives us the **fully discretized** system. The values of α plays a very important role in the stability of schemes. The stability can be classified into two categories, *condtionally stable* and *stable* systems. The **Crank Nicolson system** is when $\alpha = \frac{1}{2}$. For $\alpha < \frac{1}{2}$, the systems, then become conditionally stable and the solution starts to diverge as the time step Δt goes beyond a certain maximum value. The condition is given by,

$$\Delta t < \Delta t_{critical} \equiv \frac{2}{(1-2\alpha)\lambda} \tag{4.3.3}$$

where λ is the largest eigenvalue of the problem,

$$([M] - \lambda\,[K])\{u\} = Q \tag{4.3.4}$$

Figure 4.1: Plots illustrating instability

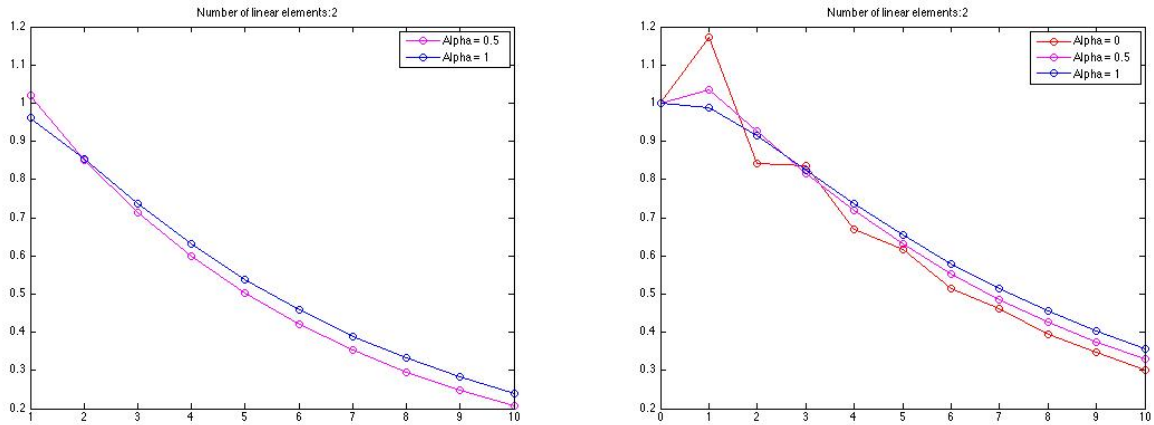


The problem was solved using MATLAB (with 2 elements and Linear interpolation functions) and the results were plotted for different values of α and Δt.

The above figure compares the solution of our problem for different values of α and $\Delta t = 0.0675$. Note that when $\alpha = 0.5$ **(Crank Nicolson)** and $\alpha = 1$ **(Backward Difference)**, we get a nice smooth plot, indicating that the schemes are **stable**. But when $\alpha = 0$ **(Forward difference)**, the scheme diverges for $\Delta t = 0.0675$. On using (4.3.3) and (4.3.4), we have $\Delta t_{critcal} = 0.0631$. Since $\Delta t \to \Delta t_{critical}$, we get an **unstable** scheme.

The stable solutions are computed and plotted for $t = 0 - 10s$ and $\Delta t = 0.0675$, $\Delta t = 0.05$ respectively.

Figure 4.2: Plots for different values of Δt



Example : Consider the following problem to solve this non-homogeneous heat equation subject to the following initial and boundary conditions.

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} + tsin(x) \\ y(0,t) &= 0 \\ y(\pi,t) &= 0 \\ y(0,x) &= sin(x) \end{aligned} \tag{4.3.5}$$

In this case, we introduce a source term that is dependent on both space (x) and time (t). The exact solution for the problem is,

$$u(x,t) = (2e^{-t} + t - 1)sin(x) \tag{4.3.6}$$

Following the same procedure, we write the weak form of (4.3.6)

$$0 = \int_0^\pi \left(w\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\frac{\partial w}{\partial x} - w\ tsin(x) \right) dx - \left[w\frac{\partial u}{\partial x} \right]_0^\pi \tag{4.3.7}$$

Using the Galerkin Finite element model,

$$0 = \sum_{j=0}^{n} \int_0^\pi \left(\psi_i\psi_j \frac{\partial u_j(t)}{\partial t} + \frac{d\psi_i}{dx}\frac{d\psi_j}{dx}u_j(t) - \psi_i\ tsin(x) \right) dx \tag{4.3.8}$$

We choose to neglect the boundary term as it is equal to 0 (from the boundary conditions). Writing in matrix form we get the **semidiscretized form**,

$$[M^e]\{\dot{u}\} + [K^e]\{u\} = [F^e] \tag{4.3.9}$$

$$M^e = \int_0^\pi (\psi_i\psi_j)\, dx \tag{4.3.10}$$

$$K^e = \int_0^\pi \left(\frac{d\psi_i}{dx}\frac{d\psi_j}{dx} \right) dx$$

$$F^e = t \int_0^\pi \psi_i(x) sin(x) dx$$

The next step is to **fully discretize** our Finite element model. For that we use the α family of approximation that was discussed before. Assuming uniform time steps and writing it for matrices, we have,

$$\Delta t\, [(1-\alpha)\{\dot{u}\}_s + \alpha\{\dot{u}\}_{s+1}] = \{u\}_{s+1} - \{u\}_s \tag{4.3.11}$$

where $\{u\}_s$ denotes the vector of nodal values of u at time $t = t_s$.

Applying (4.3.9) for $\{\dot{u}\}_s$ and $\{\dot{u}\}_{s+1}$ and substiting it in (4.3.11) and making appropriate substitutions we get the following **time marching scheme**.

$$\hat{K}_{s+1}\{u\}_{s+1} = \bar{K}_s\{u\}_s + \bar{F}_{s,s+1} \tag{4.3.12}$$

where,

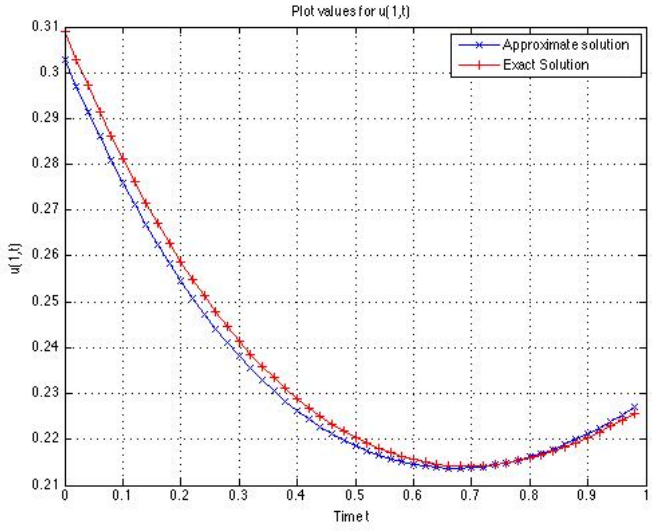
$$\hat{K_{s+1}} = M + \alpha\Delta t K_{s+1}$$

$$\bar{K}_s = M - (1-\alpha)\,\Delta t K_s$$

$$\bar{F}_{s,s+1} = \Delta t\,[\alpha F_{s+1} + (1-\alpha)F_s]$$

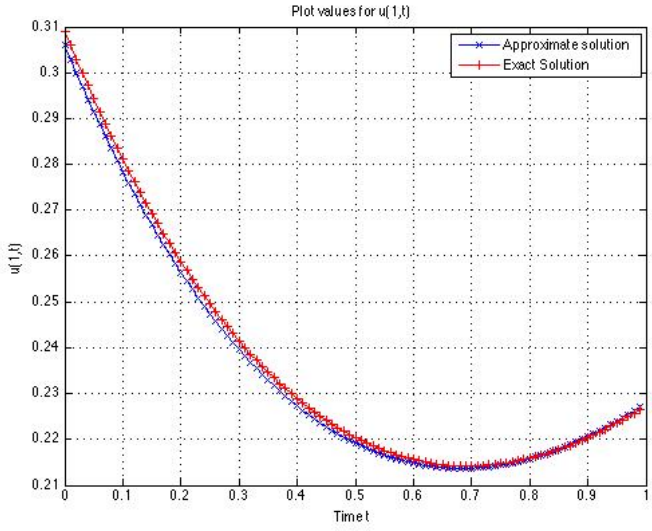
The next step is to assemble the system obtained in (4.3.12) . The procedure for assembling matrices is the same as discussed for Ordinary Differential Equations. Our matrix $F_{s,s+1}$ is time dependent in this case and the values are substitued recursively in the assembled system. For $\{u\}_s$ at $t = 0$, we use the initial condtion. This is done by substituing the node points x_i in the initial condition function $(sin(x_i))$. This entire procedure can be implemented in MATLAB and the results for $u(1, t)$ is presented as a graph below. $(\Delta t = 0.02)$ The results can be improved by

Figure 4.3: Solution for $\Delta t = 0.02$



further decreasing the time step as shown. $(\Delta t = 0.01)$

Figure 4.4: Solution for $\Delta t = 0.01$



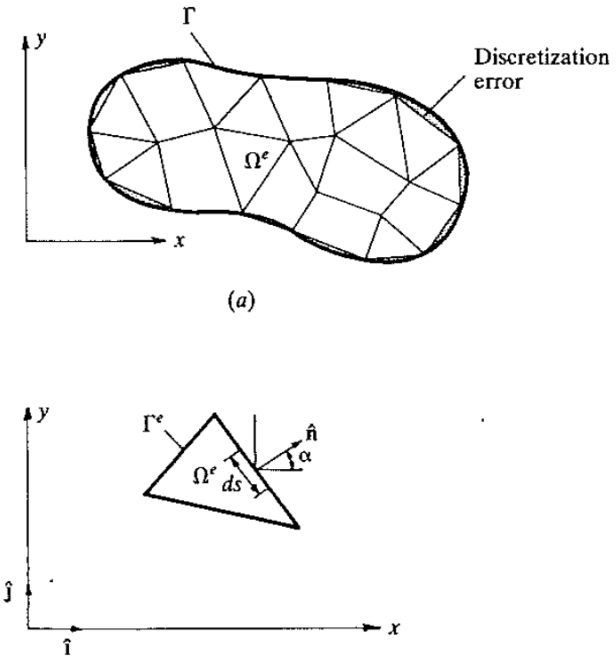
Chapter 5

Introduction to Single Variable Problems in Two Dimensions

5.1 Introduction

Finite element analysis of a single variable problem in two dimensions is the same as that of one dimensional problems, but are often complicated than one dimension problems. Usually Discretization is performed over the entire domain as shown in the figure. As a result we have two

Figure 5.1: Discretization



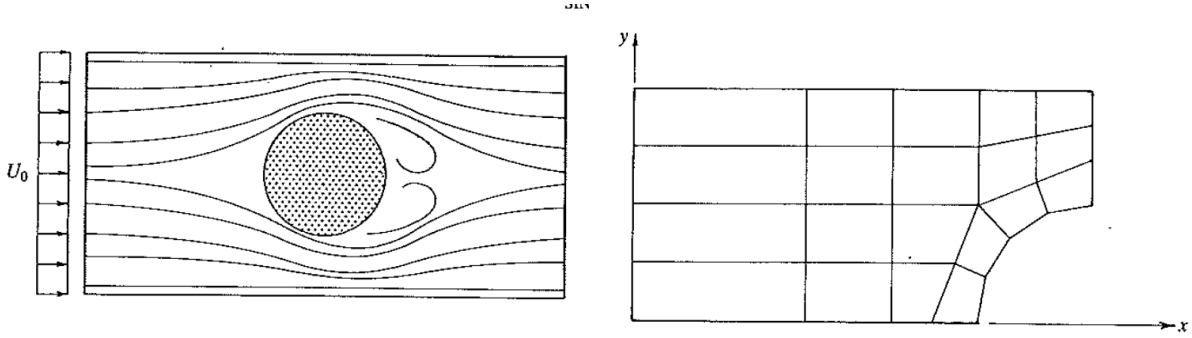
types of errors. One arising from using approximation functions known as **approximation errors** and the other from the process of discretization itself, known as **discretization errors**. A typical dicretization involves using triangular and quadrilateral elements to approximate the domain.

5.2 Rules for mesh generation in two dimensional problems

The choice of element type, density of the elements depends on the geometry of the domain, the nature of the problem to be analyzed, and the degree of accuracy desired. Keeping all these in mind, a general procedure for generating the finite element mesh includes

1. The elements that are selected should characterize the governing equations of the problem.
2. The number, shape & type (i.e., linear or quadratic) of elements should be such that the geometry of the domain is represented as accurately as desired.
3. The density of the elements should be such that regions of large gradients of the solution are adequately modeled.

Figure 5.2: Mesh generation for inviscid flow around a cylinder



4. Mesh refinements (as shown in Figure 5.2) should vary gradually from high density regions to low-density regions. If transition elements are used, they should be used away from critical regions. Transition elements are those that connect lower-order elements to higher order elements.

5.3 Analysis of a simple problem

Consider the problem of finding the solution to $u(x, y)$ to the following second order partial differential equation,

$$-\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=f \quad in\ \Omega \tag{5.3.1}$$

where Ω a square region given by

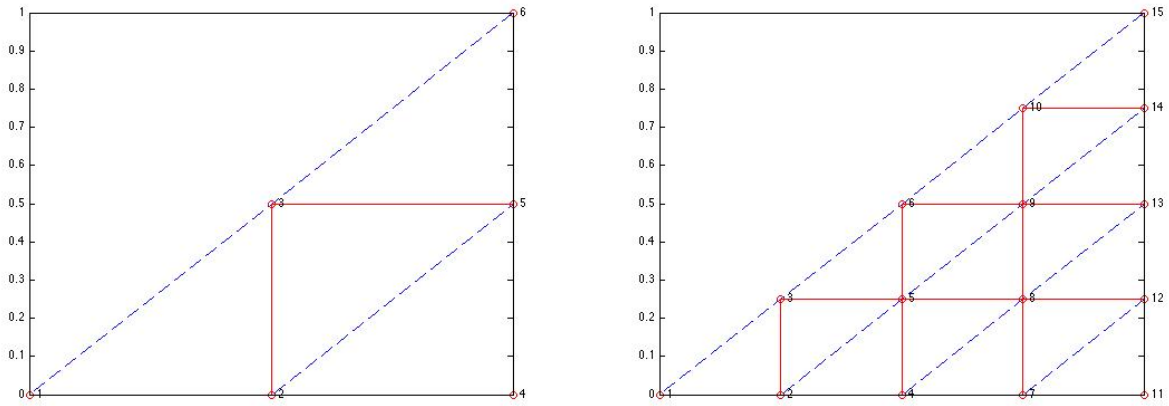
$$\Omega=\{(x,y)/-A<x<A,\ -A<y<A\} \tag{5.3.2}$$

and a boundary condition,

$$u=0 \text{ in the boundary } \Gamma$$

Owing to the symmetry of the domain along the diagonal $x = y$, we model the triangular domain as shown. The weak form of the problem is obtained by using the same procedure that

Figure 5.3: Discretization using 4 and 16 elements respectively



was illustrated for one dimensional problems. Ignoring the boundary integral term that arises from integration (Flux inside the domain = 0)., we have the weak form,

$$\int_{\Omega_e} \left(\frac{\partial w}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial w}{\partial y}\frac{\partial u}{\partial y} - wf \right) dxdy = 0 \tag{5.3.3}$$

For a typical element in the domain, we have the interpolation functions,

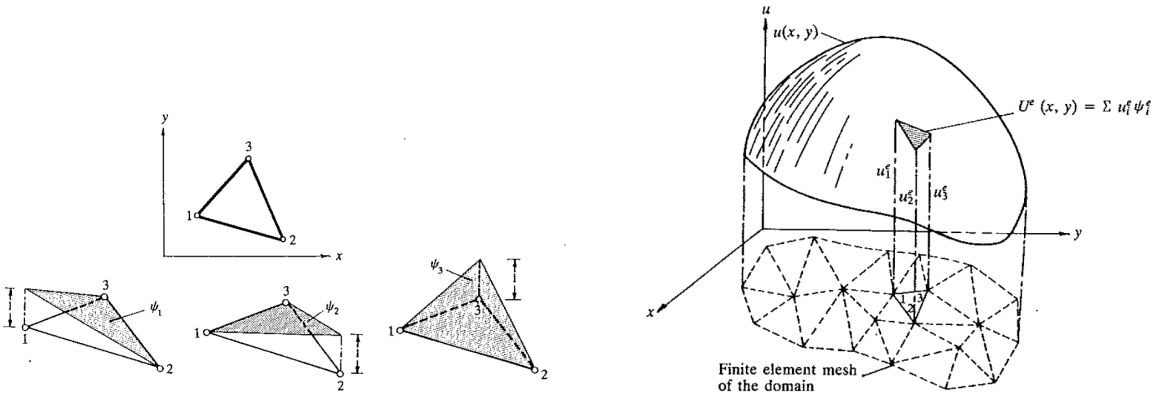
$$\psi_1(x,y) = 1 - 2x - 2y \tag{5.3.4}$$
$$\psi_2(x,y) = 2x \tag{5.3.5}$$
$$\psi_3(x,y) = 2y \tag{5.3.6}$$

with the local coordinate system (x, y). This assumption can be comfortably made as each element in the domain is exactly identical. Using the same procedure (*Galerkin finite element model*), the problem can be solved at the nodes.

5.4 Constructing triangular interpolation functions

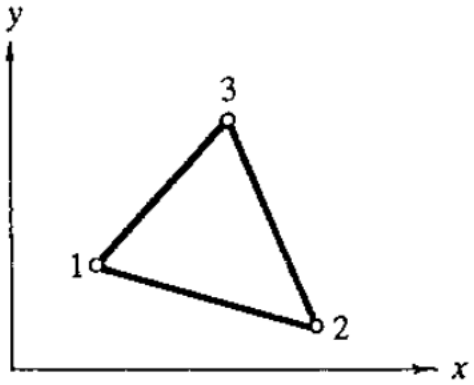
In the previous section, saw how a domain can be discretized and how the weak form of a problem can be constructed. How does these interpolation functions actually work?

Figure 5.4: Triangular Interpolation functions



The interpolation functions generated by the element is actually a plane. These small planes actually approximate solution surface, much like the case of one dimensional problems. This is clearly illustrated in Figure 5.4. Let us see now, how the interpolation functions for a triangular element is derived. If (x_1, y_1), (x_2, y_2), (x_3, y_3) are the coordinates of nodes 1, 2 & 3 respectively,

Figure 5.5: Triangular element



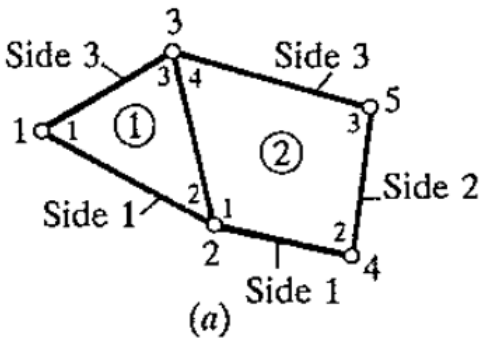
we have the matrix, $A = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}$ The array of interpolating function is thus given by

$$\Psi = \{1 \quad x \quad y\}[A]^{-1} \tag{5.4.1}$$

5.5 Assembly

The process of assembling the element equations remains the same. The only difference is that when elements of different order are connected, the row size is changed, and the missing entry is denoted by a $\times$. For example, The connectivity matrix in this case becomes,

Figure 5.6: Assembly



Global →	Local
K_{11}	K_{11}^1
K_{12}	K_{12}^1
K_{22}	$K_{22}^1 + K_{11}^2$
K_{14}	0
K_{15}	0
K_{23}	$K_{23}^1 + K_{14}^2$

$$B = \begin{bmatrix} 1 & 2 & 3 & \times \\ 2 & 4 & 5 & 3 \end{bmatrix}$$

The correspondence between the local and global nodes is derived in the same way. The figure shows correspondence between the local and global nodes.

5.6 Solution of the problem corresponding to four elements

Consider the weak form (5.3.3) of the problem. Using the approximation of the form, $u_h(x,y) = \sum_{j=1}^{n} u_j \psi_j(x,y)$ and $w = \psi_i$, we obtain the element equations,

$$\sum_{j=1}^{n} \int_0^1 \int_0^x \left(\frac{\partial \psi_i^e}{\partial x} \frac{\partial \psi_j^e}{\partial x} + \frac{\partial \psi_i^e}{\partial y} \frac{\partial \psi_j^e}{\partial y} - \psi_i f \right) dy dx = 0 \quad i = 1, 2, 3, \ldots, n \tag{5.6.1}$$

Using the interpolation functions derived earlier, we construct the matrix system

$$[K^e] = \frac{1}{2} \begin{bmatrix} 1 & -1 & 0 \\ -2 & 3 & -2 \\ 0 & -1 & -1 \end{bmatrix} \tag{5.6.2}$$

Using the fact that, the elements are identical to each other, together with the connectivity matrix,

$$B = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 3 & 2 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \tag{5.6.3}$$

We assemble the system, apply the boundary conditions (set $U_4, U_5, U_6 = 0$), to get

$$\frac{1}{2}\begin{bmatrix} 1 & -1 & 0 \\ -1 & 4 & -2 \\ 0 & -2 & 4 \end{bmatrix}\begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \frac{f}{24}\begin{Bmatrix} 1 \\ 3 \\ 3 \end{Bmatrix} \tag{5.6.4}$$

Solving this system, we get the solution to the problem at the global nodes 4,5 & 6. Due to the ssymmetry of the problem about $y = x$, the same value exists on the other side of the triangle. If $f = 1$, we have,

$$U_1 = 0.3125$$
$$U_2 = 0.2292$$
$$U_3 = 0.1771$$

Conclusion

The Finite element method is a widely used method to solve problems in engineering and applied sciences. Software like ANSYS, COMSOL etc. that are being used by many engineers alike, run on the basis of Finite element method although other methods like Finite Volume Method, Finite Difference Method are being used for commercial purposes.

Numerical methods are always preferred over the traditional analytical methods owing to the former's simpicity in application. Numerical methods can be comfortably implemented in computers to get solutions to problems that cannot be solved otherwise. A lot of numerical methods are available in literature that are used to construct solutions for engineering problems.

The finite element method has a much systematic way to construct approximation functions than the traditional variational methods. Methods like Ritz method imposes strict condtions on the approximation functions like differentiablity etc. Moreover the finite element approach discussed in this article can very well be generalized to solve engineering problems, provided proper data and boundary conditions are incorporated.

Although these methods provide us with the solutions that we require, some questions still remain unanswered. Are these solutions unique? Are these stable? The answers to these questions are not discussed in this article as the entire finite element method is constructed using rigourous mathematical principles. Further study can be done on the uniqueness and stabilty of the finite element solution to some common problems like heat conduction etc.

References

1. ***An Introduction to the Finite Element Method, Third Edition, J.N.Reddy***
2. ***Mathematical Theories of Finite Element Methods, Sussanne C. Brenner, Ridgway Scott (2007)***
3. ***in.mathworks.com* - MATLAB and Simulink for Technical Computing - MathWorks India**
4. ***LaTeX- www.latex-project.org***