Finite Element Concepts and Implementation

Written by:
Daniel V. Swenson
X. Jack Xin
Liang-Wu Cai

Department of Mechanical and Nuclear Engineering
Kansas State University
Manhattan, KS 66506

Preface

Approach

The goal of these notes is to provide a complete presentation of the Galerkin weighted residual approach to the finite element method. Doing this should result in a thorough understanding of the topics discussed, while not providing a completely general discussion of all methods. Many excellent books can be used to fill in gaps in the presentation. Some recommended texts include:

> Hughes, T. J. R., *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis,* Prentice-Hall, Inc., 1987.
>
> Bathe, K.-J., *Finite Element Procedures,* Prentice-Hall, Inc., 1996.
>
> Cook, R.D., Malkus, D.S., Plesha, M.E., and Witt, R.J., *Concepts and Applications of Finite Element Analysis,* 4thEdition, John Wiley & Sons, 2002.

In these notes, our approach is similar to Hughes (1987), but presented at a more elementary level. After an introduction to the method, we implement two-dimensional elasticity using the eight-noded quadrilateral element. This introduces the topics of shape functions, numerical integration, assembly of the global equations, solution, and postprocessing.

We believe that understanding comes from implementing all fundamental concepts in code. Therefore, we provide pseudo-coding of all functions, but expect the student to implement details. In our classes, we provide a solver for use by students.

When the student completes the course they should have a thorough understanding of a finite element program. They should also be in a position to learn additional concepts of the method from other texts.

In the appendices, we discuss topics that may be of selective interest to students.

Distribution

These notes are provided free of charge to any reader. Any reader may use the notes with two stipulations:

- Please reference these notes whenever they are used. The reference should be, *Finite Element Concepts and Implementation*, Daniel Swenson, X. J. Xin, & Liang-Wu Cai,

Contents

1 A Simple One-Dimensional Example of Discretization 1
1.1 Objective 1
1.2 One-Dimensional Spring System 1
1.3 Equations Using Nodal Equilibrium 2
1.4 Equations Using Spring Element 4
1.5 Solution Using Gauss Elimination 6
1.6 Check Solution 7
1.7 From Real Structure to Model 8
1.8 Hints to Questions 9
2 Weighted Residuals 10
2.1 Objective 10
2.2 Weighted Residual Method 10
2.3 How Weighted Residuals Work 11
2.4 Example Using One Term Approximation 13
2.4.1 Collocation Method 14
2.4.2 Subdomain Method 15
2.4.3 Least Squares Method 15
2.4.4 Galerkin Method 16
2.4.5 Comparison of Solutions 16
2.5 Using More Terms in Approximation 17
2.6 Hints to Questions 17
3 Strong Form and Weak Form of the Differential Equation 18
3.1 Purpose 18
3.2 Weak Form of Differential Equation 18
3.3 Approximate Solution Using the Weak Formulation 20
3.4 Example Solution Using Single Term Approximation 22
3.5 Example Solution Using Two Terms 23
3.6 Hints to Questions 24
3.7 References 25

4 Hat Functions 26
4.1 Objective 26
4.2 Hat Functions 26
4.3 Solution Using Hat Functions 28
4.4 Hints to Questions 33
4.5 References 33
5 A First Finite Element and Shape Functions 34
5.1 Objective 34
5.2 Desired Characteristics of an Effective Computational Method 34
5.3 A Linear Finite Element 34
5.4 Equivalence Between Global Integration of Hat Functions and Summing the Integration of Element Shape Functions 37
5.5 Shape Functions in Natural Coordinates 39
5.5.1 Linear Shape Functions 39
5.5.2 Quadratic Shape Functions 40
5.6 Lagrange Polynomials 42
5.7 Hints to Questions 44
5.8 References 44
6 Generalized Matrix Form of the Finite Element Equations 45
6.1 Purpose 45
6.2 Matrix Derivation of Equations 45
6.3 The Global Matrix Equations 47
6.4 The Element Stiffness Matrix 47
6.5 Hints to Questions 50
6.6 References 50
7 Weak Statement of Elasticity 51
7.1 Purpose 51
7.2 Brief Review of Elasticity 51
7.2.1 Relation Between Surface Tractions and Stress (First Derivation) 52
7.2.2 Equilibrium 53
7.2.3 Strain-Displacement Relations 54
7.2.4 Stress-Strain Relations (Plane Strain) 56
7.2.5 Stress-Strain Relations (Plane Stress) 58
7.3 Elastic Finite Element Formulation 59
7.4 Hints to Questions 61

7.5 References 61
8 Elasticity Using Finite Elements 62
8.1 Purpose 62
8.2 Finite Element Statement 62
8.3 Hints to Questions 66
8.4 References 66
9 Natural Coordinates and Numerical Integration 67
9.1 Purpose 67
9.2 Change of Variables 67
9.3 Gauss-Legendre Integration 68
9.4 Example 71
9.5 Higher Order Gauss-Legendre Integration 73
9.6 Application of Numerical Integration to Evaluation of Element Matrices 73
9.7 Numerical Integration in 2D 75
9.8 Integration of Body Forces and Surface Tractions 77
9.9 Implementation 77
9.10 Hints to Questions 78
9.11 References 78
10 More on Shape Functions and Isoparametric Elements 79
10.1 Purpose 79
10.2 Finite Element Statement 79
10.3 The Serendipity Element (8-Noded Quadratic, Q8, Element) 81
10.4 Details of the Element Matrices 83
10.5 The Material Matrix 86
10.6 Body Forces 87
10.7 Point Forces and Surface Tractions 88
10.8 Hints to Questions 89
10.9 References 90
11 Some Theoretical Aspects of FEM 91
11.1 Purpose 91
11.2 Well-posedness of a Problem (Hughes, §2.1) 91
11.3 Continuity of Displacements, Strains and Stresses (Hughes, §1.Appendix and §3.1) 91
11.4 Galerkin Formulation; Symmetry and Positive-definiteness of K (Hughes, §2.8) 92

11.5 "Best Approximation" and Error Estimates: Why Standard FEM Usually Works and Why Sometimes It Does Not (Hughes, §4.1) 92
11.6 Reduced Integration (Hughes, §4.4) 94
11.7 Zero-energy Mode and Rank Deficiency (Hughes, §4.6) 95
11.8 Patch Test (Hughes, §4.6) 96
11.9 Nonconfirming Elements (Hughes, §4.7) 96
11.10 Mixed or Hybrid Elements (Hughes, §4.3) 97
11.11 References 97
12 Computer Implementation 98
12.1 Purpose 98
12.2 Design of the Finite Element Program 98
12.2.1 femElement Class 101
12.2.2 femElementQ8 Class 103
12.2.3 femNode Class 104
12.2.4 femDof Class 105
12.2.5 femData Class 106
12.2.6 femEssentialBC Class 108
12.2.7 femNaturalBC Class 109
12.2.8 femPointBC Class 110
12.2.9 femSolve Class 111
12.2.10 femMaterial Class 112
12.2.11 femMaterialElastic Class 113
12.2.12 femGauss3Pt Class 114
12.2.13 utlMatrix Class 115
12.2.14 utlMatrixFull Class 116
12.2.15 utlMatrixBandedSymm Class 117
12.2.16 utlVector Class 118
12.2.17 utlStress2D Class 119

1 A Simple One-Dimensional Example of Discretization

1.1 Objective

Our objective is to introduce one view of the finite element method using discrete elements. In this view, the finite element method consists of the following steps:

1. Dividing the problem into elements.
2. Obtaining a stiffness matrix for each element.
3. Assembling each element stiffness matrix into a global matrix.
4. Solving the global matrix equations.

This approach is that used for frame analysis by civil engineers. It also is the basis of the first papers on finite element analysis of a continuum (Clough, 1960). In later lectures we will develop a more mathematical approach, where the finite element method will be obtained using weighted residuals.

In this lecture, we will first illustrate the solution of a spring system by writing equations of equilibrium. We will then write the stiffness matrix for a spring element and assemble these into a global matrix. Finally, the solution using Gauss elimination simulates a computer solution.

1.2 One-Dimensional Spring System

The example problem we will use is shown in Figure 1-1. This figure shows a system of connected springs with a load. What we will call "nodes" (points at which elements are connected) are indicated using circles and the "elements" (springs) are indicated using squares. The displacement of each node is given by u_i, where i is the node number.

Question 1: How dose the numbering of nodes and elements affect the solution?

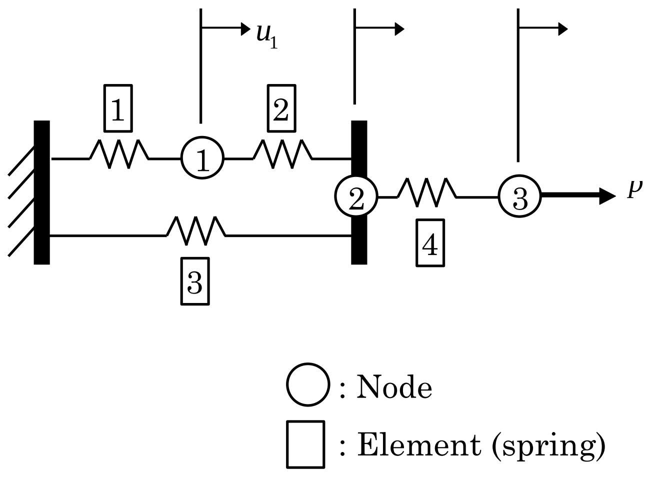


Figure 1-1: Spring System

1.3 Equations Using Nodal Approach

Recall that the force in a spring is related to the stretching:

$$f = k\delta \tag{1.1}$$

where k is the spring stiffness and δ is the spring stretching. The free body diagram at node 1 is:

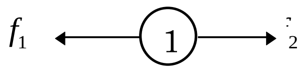


Figure 1-2: FBD at node 1

The equation of equilibrium is:

$$\sum F_X = 0$$

$$-f_1 + f_2 = 0$$

or

$$-k_1(u_1) + k_2(u_2 - u_1) = 0 \tag{1.2}$$

Similarly at node 2 (Figure 1-3):

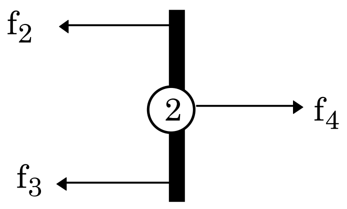


Figure 1-3: FBD at node 2

$$
\begin{aligned}
\sum F_X &= 0 \\
-f_2 - f_3 + f_4 &= 0 \\
-k_2(u_2 - u_1) - k_3(u_2) + k_4(u_3 - u_2) &= 0
\end{aligned}
\tag{1.3}
$$

And at node 3 (Figure 1-4):

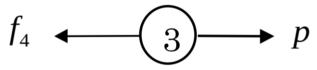


Figure 1-4: FBD at node 3

$$
\begin{aligned}
\sum F_X &= 0 \\
-f_4 + P &= 0 \\
-k_4(u_3 - u_2) + p &= 0
\end{aligned}
\tag{1.4}
$$

Put **Error! Reference source not found.**, **Error! Reference source not found.** and **Error! erence source not found.** together:

$$
\begin{aligned}
&(\mathrm{k}_1 + \mathrm{k}_2)\mathrm{u}_1 - (\mathrm{k}_2)\mathrm{u}_2 = 0 \\
&-(\mathrm{k}_2)\mathrm{u}_1 + (\mathrm{k}_2 + \mathrm{k}_3 + \mathrm{k}_4)\mathrm{u}_2 - (\mathrm{k}_4)\mathrm{u}_3 = 0 \\
&-(\mathrm{k}_{41})\mathrm{u}_2 + (\mathrm{k}_4)\mathrm{u}_3 = \mathrm{p}
\end{aligned}
$$

and rewrite using matrix form:

$$
\boxed{\begin{bmatrix} \mathrm{k}_1 + \mathrm{k}_2 & -\mathrm{k}_2 & 0 \\ -\mathrm{k}_2 & \mathrm{k}_2 + \mathrm{k}_3 + \mathrm{k}_4 & -\mathrm{k}_4 \\ 0 & -\mathrm{k}_4 & \mathrm{k}_4 \end{bmatrix} \begin{Bmatrix} \mathrm{u}_1 \\ \mathrm{u}_2 \\ \mathrm{u}_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ \mathrm{p} \end{Bmatrix}}
\tag{1.5}
$$

or:

$$
\mathbf{Ku} = \mathbf{p} \tag{1.6}
$$

In finite element terminology, $\mathbf{K}$ is the stiffness matrix, $\mathbf{u}$ is the displacement vector and $\mathbf{p}$ is the load vector. Note: Bold upper case is a matrix, bold lower case is a vector.

1.4 Equations Using Element Approach

Eq. **Error! Reference source not found.** was obtained using equilibrium at each node. We ill now show how the same result can be obtained using the concept of a spring element with an associated element stiffness matrix. A typical spring is shown in Figure 1-5. It has nodes i and j, associated displacements (or degrees of freedom) u_i and u_j, and stiffness k.

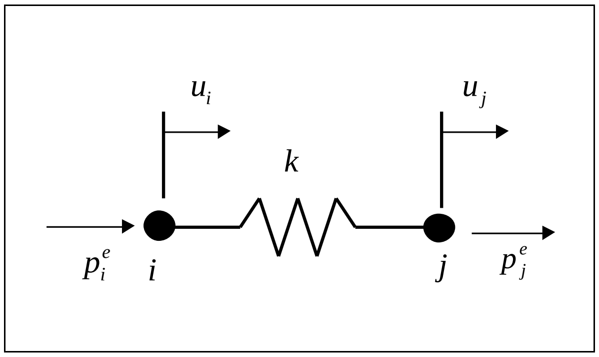


Figure 1-5: Typical spring element

The stiffness matrix may be thought of as the matrix that relates forces on an element to displacements of the element, i.e.

$$\begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{Bmatrix} u_i \\ u_j \end{Bmatrix} = \begin{Bmatrix} p_i^e \\ p_j^e \end{Bmatrix} \tag{1.7}$$

The coefficients of the stiffness matrix are the forces at the nodes that correspond to a unit displacement at each node, holding all other degrees of freedom (DOF's) fixed. For example, giving node *i* a unit displacement, while holding node *j* fixed results in the forces:

$$\begin{aligned} p_i^e &= ku_i \\ p_j^e &= -ku_i \end{aligned} \tag{1.8}$$

This gives the first column of coefficients in the element stiffness matrix:

$$\begin{bmatrix} k & ? \\ -k & ? \end{bmatrix} \begin{Bmatrix} u_i \\ u_j \end{Bmatrix} = \begin{Bmatrix} p_i^e \\ p_j^e \end{Bmatrix} \tag{1.9}$$

Similarly, giving node *j* a unit displacement, while holding node *i* fixed results in the forces:

$$\begin{aligned} p_i^e &= -ku_j \\ p_j^e &= ku_j \end{aligned} \tag{1.10}$$

This completes the element stiffness matrix.

$$\boxed{\begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \begin{Bmatrix} u_i \\ u_j \end{Bmatrix} = \begin{Bmatrix} p_i^e \\ p_j^e \end{Bmatrix}} \tag{1.11}$$

Note that the element matrix is symmetric.

To assemble the global stiffness matrix using the element approach, we must define a relationship between the element node numbering scheme and the global numbering scheme. We will do this for element 2 first, since this element is not associated with a boundary condition. Referring to Figure 1-1, we see that for element 2, node *i* corresponds to global node 1, and that node *j* corresponds to global node number 2. Letting *i*=1 and *j*=2, we can take the associated element stiffness coefficients and place them in the global matrix.

$$\begin{bmatrix} k_2 & -k_2 & 0 \\ -k_2 & k_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} f_1^2 \\ f_2^2 \\ 0 \end{Bmatrix} \tag{1.12}$$

We now return to element 1. This element is a bit tricky, because one of its nodes corresponds to a known zero displacement in Figure 1-1. Therefore for element 1, node *i* does not have an associated global node number, but that node *j* has a global node number of 1. Therefore, only the *j* local node contributes to the global matrix.

$$\begin{bmatrix} k_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} f_1^1 \\ 0 \\ 0 \end{Bmatrix} \tag{1.13}$$

Proceeding in the same manner, for element 3 the contributions to the global matrix will be:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & k_3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ f_2^3 \\ 0 \end{Bmatrix} \tag{1.14}$$

And finally, for element 4 the contributions to the global matrix will be:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & k_4 & -k_4 \\ 0 & -k_4 & k_4 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ f_2^4 \\ f_3^4 \end{Bmatrix} \tag{1.15}$$

To obtain the complete global stiffness matrix, we add **Error! Reference source not found.**, REF b13 \h **Error! Reference source not found.**, **Error! Reference source not found.** and **Error! Reference source not found.** (the procedure in FEM is referred to as **assembly**), giving:

$$\begin{bmatrix} (k_1 + k_2) & -(k_2) & 0 \\ -(k_2) & (k_2 + k_3 + k_4) & -(k_4) \\ 0 & -(k_4) & (k_4) \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} f_1^1 + f_1^2 \\ f_2^1 + f_2^3 + f_2^4 \\ f_3^4 \end{Bmatrix} \tag{1.16}$$

Now, we can simplify the right hand side by recognizing that this is just a statement of equilibrium at each node. This gives:

$$\boxed{\begin{bmatrix} (k_1 + k_2) & -(k_2) & 0 \\ -(k_2) & (k_2 + k_3 + k_4) & -(k_4) \\ 0 & -(k_4) & (k_4) \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ p \end{Bmatrix}} \tag{1.17}$$

Clearly, this is the same as obtained using nodal equilibrium, **Error! Reference source not ound.**. We have thus demonstrated how an element approach can be used to obtain the global stiffness matrix. The advantage of the element approach is that assembling the matrix can be easily automated for computer use.

*Question 2: If an additional node, which is numbered 0, is put between the spring system and the leftside wall, how will **Error! Reference source not found.** look like?*

*Question 3: The stiffness matrix in FEM usually has a lot of zero elements. How to judge, from the physical structure of the system being studied, whether an element of **K** is zero or not?*

Question 4: The assembly involves putting element stiffnesses together to form the global stiffness matrix. Illustrate graphically how to decide the position in the global matrix.

1.5 Solution Using Gauss Elimination

As a simple reminder and to simulate the actual computer solution, we will solve **Error! eference source not found.** using Gauss elimination. Assume that k_1=500 lb/in, k_2=250 lb/in, k_3=2000 lb/in, k_4=1000 lb/in, and P=1000 lb, then the global matrix becomes:

$$\begin{bmatrix} (750) & -(250) & 0 \\ -(250) & (3250) & -(1000) \\ 0 & -(1000) & (1000) \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 1000 \end{Bmatrix} \qquad (1.18)$$

First, normalize row 1:

$$\begin{bmatrix} (1) & -(0.3333) & 0 \\ -(250) & (3250) & -(1000) \\ 0 & -(1000) & (1000) \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 1000 \end{Bmatrix} \qquad (1.19)$$

Next, zero the remaining coefficients in column 1 by multiplying row 1 by the appropriate factor and subtracting row 1 from each of the remaining rows.

$$\begin{bmatrix} (1) & -(0.3333) & 0 \\ 0 & (3166.7) & -(1000) \\ 0 & -(1000) & (1000) \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 1000 \end{Bmatrix} \qquad (1.20)$$

Normalize row 2 and repeat:

$$\begin{bmatrix} (1) & (-0.3333) & 0 \\ 0 & (1) & -(0.3158) \\ 0 & 0 & (684.2) \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 1000 \end{Bmatrix} \qquad (1.21)$$

Finally backsubstitute to obtain the final answer:

$$\mathbf{u} = \begin{Bmatrix} 0.1538 \\ 0.4616 \\ 1.4615 \end{Bmatrix} \qquad (1.22)$$

Question 5: In what situations, apart from singular matrix, will the Gauss elimination method become ill behaved? How to overcome the problem?

Question 6: If a truss member, which has an area A, length L and Young's modulus E, is made equuivalent to a spring, what should be the value of k for the srpring?

1.6 Check Solution

It is always necessary to view numerical results with skepticism until confidence in the solution can be justified. We will check our solution by calculating the forces in the springs. For spring 1, the force is:

$$\begin{aligned} f_1 &= k_1(u_1) \\ &= 250(0.1538) \\ f_1 &= 77\ lb \end{aligned} \tag{1.23}$$

The other forces are calculated in a similar manner and plotted in Figure 1-6.

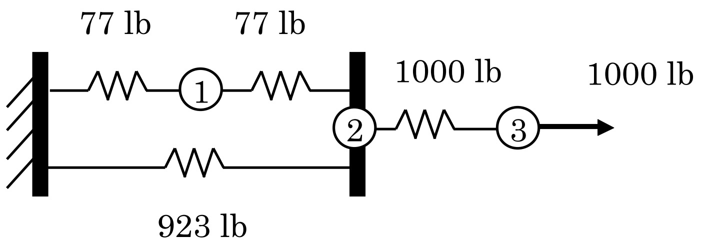


Figure 1-6: Spring force

As can be seen, the results satisfy equilibrium.

1.7 From Real Structure to Model

You may question the value or the usefulness of the spring system as shown in Figure 1-1, because most mechanical structures are not real spring systems. The value of the spring system, however, is that it can be used to model many real mechanical structures. An example is shown in Figure 1-7 below.

In Figure 1-7 (a), a rather complicated structure (shaded) is subjected to a force P. The problem seems difficult to solve, because the cross-section of the structure is non-uniform. We can, however, devide the structure into 4 elements with uniform cross-section, as shown in Figure 1-7 (b). Each element acts as a uniform bar under axial loading. Mechanically, a uniform bar loaded axially can be modeled exactly by a spring, with the following equivalent spring constant:

$$k = \frac{L}{AE} \tag{1.24}$$

where L is the axial length of the bar, A is the cross-section area, and E is the Young’s modulus of the bar material. With the proper spring constant for each spring element, the spring system in Figiure 1-1 can be used to solve the structure in Figure 1-7 (a). The displacement of node 3 in the spring system in Figure 1-1 will represent the displacement of the loading point in Figure 1-7.

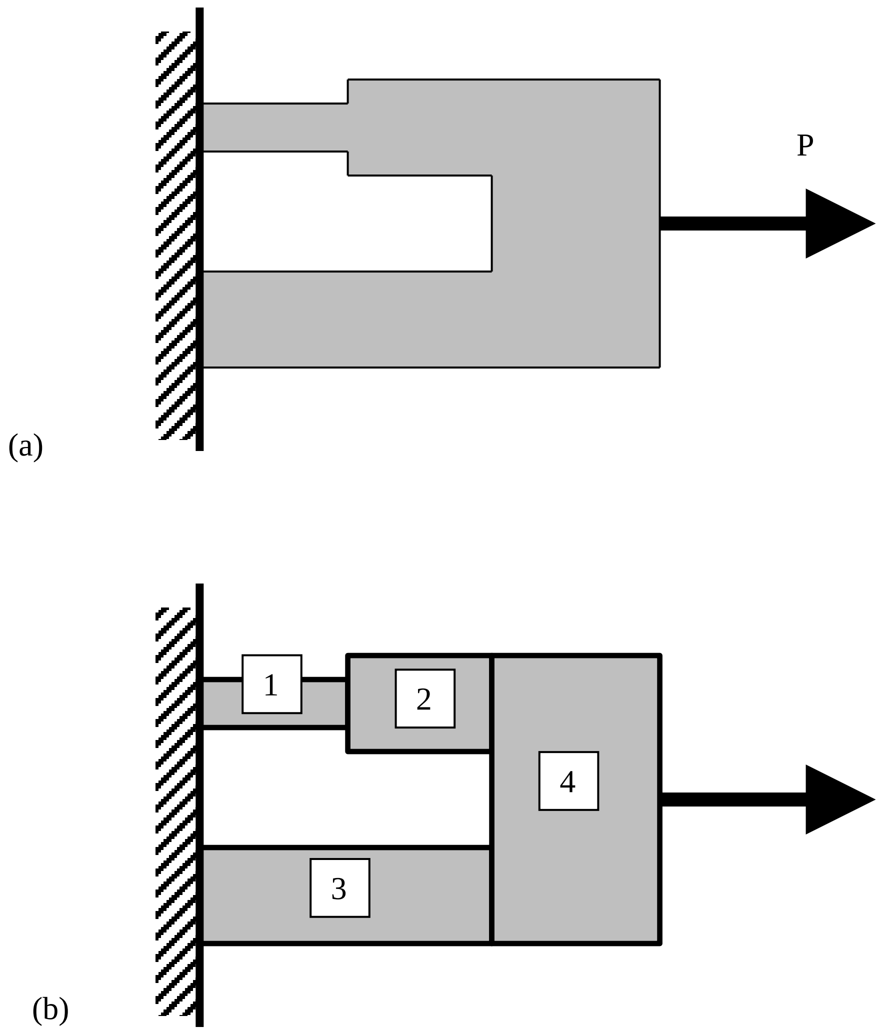


Figure 1-7: From a structure to an element model

1.8 Hints to Questions

Q1: The solutions will be the same at the same physical locations.
Q2: (omit)
Q3: Look at the connectivity between two nodes.
Q4: Use the relation between local numbering and global numbering.
Q5: When the diagonal term is not dominating. Use row and column pivoting.
Q6: k=AE/L

2 Weighted Residuals

2.1 Objective

Our objective is to introduce the method of weighted residuals for obtaining approximate solutions to differential equations.

We will first introduce the method of weighted residuals using a single term approximation and show how different methods result in different results. We then demonstrate the Galerkin method using multiple terms in the approximation.

2.2 Weighted Residual Methods

Weighted residual methods are another way to develop approximate solutions. Recall that in the finite difference method we developed an approximation to the differential equation at a *point*. In contrast, in weighted residual methods, we assume the form of the *global* solution and then adjust parameters to obtain the best global fit to the actual solution.

Let us be more precise in our definitions. We are given a body B with boundary S. As shown in Figure 2-1, the boundary is divided into two regions, a region S_u with essential (Dirichlet) boundary conditions and a region S_f with natural (Neumann) boundary conditions. The essential boundary conditions are specifications of the solution on the boundary (for example, known boundary displacements), while the natural boundary conditions are specifications of derivatives of the solution (for example, surface tractions). *All points on the boundary must have one or the other type of specified boundary condition for the problem to be well defined mathematically*.

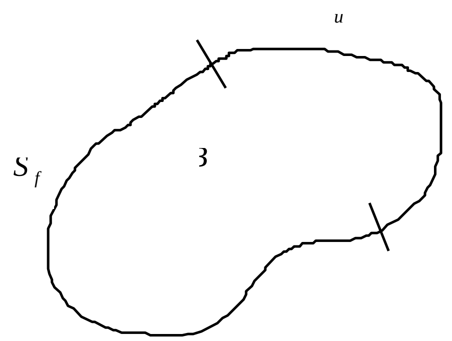


Figure 2-1: General body with boundary

In this lecture, we will present the weighted residual methods in ways that require the approximating function to satisfy *both the essential and natural boundary conditions*. In the next lecture, we will show how the weak form of the differential equation can be used to develop methods that loosen this requirement, so that only the essential boundary conditions must be satisfied by our approximating function. The basic step in weighted residual methods is to assume a solution of the form:

$$u_n = \sum_{j=1}^{n} a_j \phi_j \tag{2.1}$$

Our job is to solve for the coefficients a_i that give a best fit (in some sense) to the exact solution.

2.3 How Weighted Residuals Work

Let us first demonstrate how weighted residuals work using a bar subjected to body and end loads (Figure 2-2).

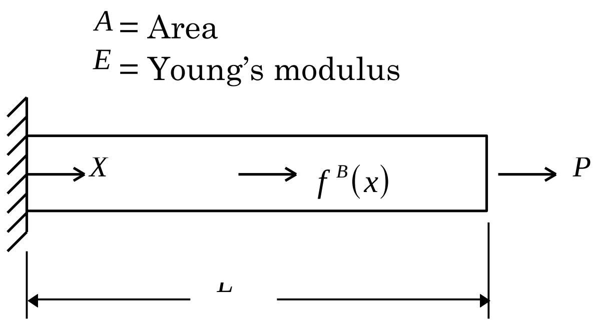


Figure 2-2 Uniform bar with body and tip loads

For static equilibrium, the summation of the forces is zero:

$$\sum F_x = 0$$

or:

$$-A\sigma\big|_x + f^B(x)\Delta x + A\sigma\big|_{x+\Delta x} = 0$$

Rearranging and assuming constant area:

$$A\frac{\left(\sigma\big|_{x+\Delta x} - \sigma\big|_x\right)}{\Delta x} + f^B(x) = 0$$

Taking the limit as $\Delta x \to 0$,

$$A\frac{d\sigma}{dx} + f^B(x) = 0 \tag{2.2}$$

For an elastic material, the stress is related to the strain by,

$$\sigma = E\varepsilon \tag{2.3}$$

where E is Young's modulus. The strain is related to the displacements by:

$$\varepsilon = \frac{du}{dx} \tag{2.4}$$

Substituting (2.4) and (2.3) into (2.2),

$$A\frac{d}{dx}\left(E\frac{du}{dx}\right) + f^B(x) = 0$$

Assuming Young's modulus is constant, with $f^B(x) = b$, gives,

$$AE\frac{d^2u}{dx^2} + b = 0 \qquad \text{for } 0 \le x \le L \tag{2.5}$$

with BC's:

$$u\big|_{x=0} = 0 \tag{2.6}$$

$$EA\frac{du}{dx}\bigg|_{x=L} = P \tag{2.7}$$

Multiply (2.5) by a weighting function $w(x)$:

$$w\left(EA\frac{d^2u}{dx^2} + b\right) = 0 \tag{2.8}$$

Now integrate over the body:

$$\int_0^L w\left(EA\frac{d^2u}{dx^2} + b\right)dx = 0 \tag{2.9}$$

This is called the weighted residual formulation. It is called this because if we assume an approximate solution ϕ_n (that satisfies all boundary conditions) then,

$$AE\frac{d^2\phi_n}{dx^2} + b = R(x) \neq 0 \tag{2.10}$$

Instead, we have an error (residual) that is a function of x. Thus (2.9) is really a weighting of the residual over the body:

$$\int_0^L wR\,dx = 0 \qquad (2.11)$$

We have taken the error (residual), multiplied by a weighting function and set the weighted integral to zero.

Question 1: Why can we say the weighted residual form of an equation is more general than the traditional concept of equation?

2.4 Example Using One Term Approximation

To see how this allows us to obtain values for the coefficients, let us do an example. We will pick the approximate function to satisfy all boundary conditions.

$$u_n = \frac{P}{AE}x + a_1 \sin\left(\frac{\pi}{2L}x\right) \qquad (2.12)$$

Clearly, this function satisfies the essential boundary condition (2.6), as well as the natural boundary condition (2.7). This can be verified easily by substituting (2.12) in the left hand side of (2.6) and (2.7). Note that the firt term in (2.12) satifies the essential boundary condition at the left end and the natural boundary condition at the rigth end, while the second term in (2.12) is zero at the the left end and its slope is zero at the right end.

Taking the derivatives:

$$\begin{aligned} \frac{du_n}{dx} &= \frac{P}{AE} + a_1 \frac{\pi}{2L}\cos\left(\frac{\pi}{2L}x\right) \\ \frac{d^2u_n}{dx^2} &= -a_1\left(\frac{\pi}{2L}\right)^2 \sin\left(\frac{\pi}{2L}x\right) \end{aligned} \qquad (2.13)$$

Substitute (2.13) into (2.10) to obtain the residual,

$$-AEa_1\left(\frac{\pi}{2L}\right)^2 \sin\left(\frac{\pi}{2L}x\right) + b = R(x) \qquad (2.14)$$

For simplicity, pick $A = E = P = L = b = 1$. Then (2.14) becomes:

$$R(x) = -a_1\frac{\pi^2}{4}\sin\left(\frac{\pi}{2}x\right) + 1 \qquad (2.15)$$

Figure 2-3 plots the residual (2.15) for different values of a_1.

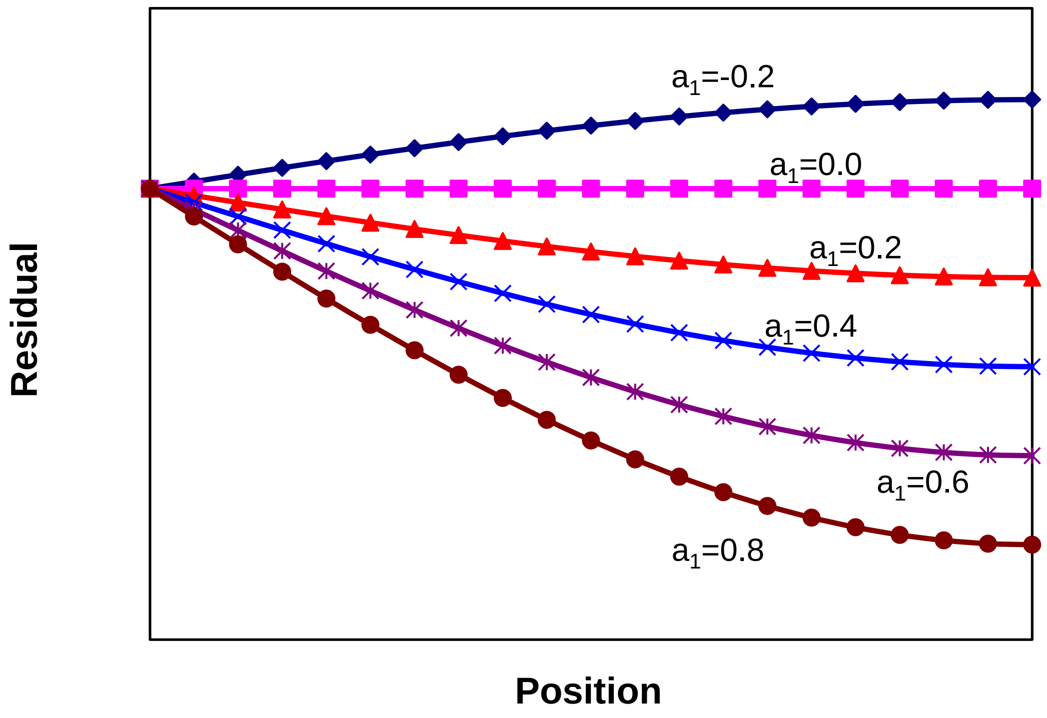


Figure 2-3: Residual over body

Now it is clear why we integrate. The error is a function of x. We can weight the error (residual) in any way we want over the interval and for the integral to be zero. Depending on how we weight the residual, we get different solutions.

2.4.1 Collocation Method

In this case, we force the residual to be zero at a specific location ("nail-down" method). That is:

$$w(x) = \delta(x - x_i)$$

Then,

$$\int_0^1 \delta(x - x_i) R dx = 0 \qquad (2.16)$$

or

$$R(x_i) = 0$$

If we pick $x_i = 0.5$, that is, we want the residual to equal zero at the midpoint, then inspection of Figure 2-3 shows that a value of $a_1 \approx 0.60$ satisfies that condition. Solving exactly,

$$-a_1\frac{\pi^2}{4}\sin\left(\frac{\pi}{4}\right)+1=0$$

$$\underline{a_1=0.573} \tag{2.17}$$

2.4.2 Subdomain Method

Alternately, let us weight the residual uniformly over the interval ("glue" method). That is:

$$w(x)=1$$

Then,

$$\int_0^1 1R\,dx=0 \tag{2.18}$$

By inspection, we can again see that a value of $a_1=0.6$ will result in approximately equal areas above and below the x axis. Let us solve exactly:

$$\int_0^1 1\left(-a_1\frac{\pi^2}{4}\sin\left(\frac{\pi}{2}x\right)+1\right)dx=0$$

$$\left(a_1\frac{\pi}{2}\cos\left(\frac{\pi}{2}x\right)+x\right)\Bigg|_0^1=0$$

$$1-a_1\frac{\pi}{2}=0$$

$$\underline{a_1=\frac{2}{\pi}=0.637} \tag{2.19}$$

2.4.3 Least Squares Method

In least squares, we require that the squared residual be minimized with respect to the adjusting parameter, i.e.

$$\text{Minimize}\left(\int_0^1 R^2dx\right)$$

or

$$\int_0^1 R\frac{\partial R}{\partial a_i}dx=0$$

This is equivalent to select $\frac{\partial R}{\partial a_i}$ as the weighting function. For our particular problem, there is only one adjusting parameter a_1, and

$$\frac{\partial R}{\partial a_1} = -\frac{\pi^2}{4}\sin\left(\frac{\pi}{2}x\right)$$

$$\int_0^1 R\frac{\partial R}{\partial a_i}dx = \int_0^1\left(-a_1\frac{\pi^2}{4}\sin\left(\frac{\pi}{2}x\right)+1\right)\left(-\frac{\pi^2}{4}\sin\left(\frac{\pi}{2}x\right)\right)dx = 0$$

$$-a_1\frac{\pi^2}{4}\int_0^1\left(\sin^2\left(\frac{\pi}{2}x\right)\right)dx + \int_0^1\left(\sin\left(\frac{\pi}{2}x\right)\right)dx = 0$$

$$-a_1\frac{\pi^2}{4}\left(\frac{1}{2}x - \frac{1}{2\pi}\sin(\pi x)\right)\Bigg|_0^1 + \left(-\frac{1}{\pi}\cos(\pi x)\right)\Bigg|_0^1 = 0$$

$$-a_1\frac{\pi^2}{8} + \frac{2}{\pi} = 0$$

which leads to
$$\underline{a_1 = \frac{16}{\pi^3} = 0.516}$$

2.4.4 Galerkin Method

Finally, if we use the approximating function as the weighting function (except that we do not include the term in the approximating function that satisfies the essential boundary conditions, which does not enter into this case) then we have:

$$w(x) = d_1\sin\left(\frac{\pi}{2}x\right)$$

and

$$\int_0^1\left(d_1\sin\left(\frac{\pi}{2}x\right)\right)\left(-a_1\frac{\pi^2}{4}\sin\left(\frac{\pi}{2}x\right)+1\right)dx = 0 \qquad (2.20)$$

$$\underline{a_1 = \frac{16}{\pi^3} = 0.516} \qquad (2.21)$$

Note that for our particular example, the least-squares method and the Galerkin's method yield identical results. In general, however, the two methods may give different answers.

Question 2: How do you judge whether an approximate method is legitimate or acceptable? If there are two approximate methods, with what criterion can you say one is better than the other?

Question 3: Theoretically, how can you obtain an excat solution (or as acuurate as you like) with an approximate method?

2.4.5 Comparison of Solutions

Figure 2-4 compares the analytical and approximate solutions. This shows that the Galerkin method gives a good approximation to the actual solution.

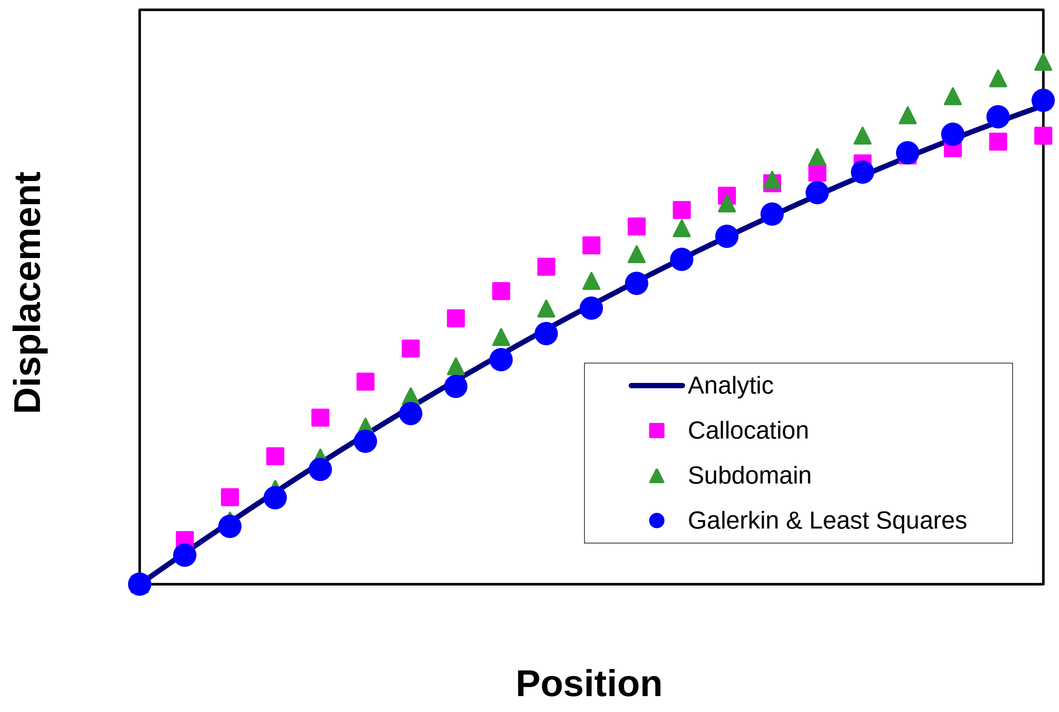


Figure 2-4: Comparison of analytic and approximate solutions. Note that the results for the least squares method are the same as for the Galerkin's method.

2.5 Using More Terms in Approximation

It would be possible to solve this problem using more terms in the approximate solution, but practically speaking, this is not a good approach if we must satisfy natural boundary conditions. It is much better to proceed to the weighted residuals formulation based on the weak form of the differential equation. This formulation will automatically incorporate the natural boundary conditions, making it much simpler to develop the approximate solution. This will be dealt with in next lecture.

2.6 Hints to Questions

Q1: The weighted residual approach extends the concept of solution which includes the exact solution.
Q2: The method must not exclude the exact solution if found; the method should produce the exact solution for constant function.
Question 3: The bases for functional space should be complete, and number of terms must be large enough.

3 Strong Form and Weak Form of the Differential Equation

3.1 Purpose

The purpose of this lecture is to introduce the weak form of the differential equation. This is used for the basis of our final finite element formulation. It will give us a symmetric stiffness matrix (for the elasticity problem) and clearly shows the natural boundary condition (surface traction) terms.

We will use the one-dimensional elastic rod as our prototype for the derivation.

3.2 Weak Form of Differential Equation

The strong statement of the differential equation is the one we derived using the differential element (previous lecture). It is given by:

$$AE\frac{d^2u}{dx^2}+b=0 \quad \text{for } 0\le x\le L \tag{3.1}$$

with the boundary conditions:

$$u\big|_{x=0}=0 \qquad \text{(an essential b.c.)} \tag{3.2}$$

$$AE\frac{du}{dx}\bigg|_{x=L}=P \qquad \text{(a natural b.c.)} \tag{3.3}$$

We call the differential equation (3.1) with boundary conditions (3.2) and (3.3) the ***strong form of the differential equation***. This is the conventional form of differential equation in almost all calculus courses.

Now let us proceed to derive the ***weak form***. Similar to lecture 2, we multiply (3.1) by a weighting function w. This function is chosen *to be zero at all essential boundary conditions (but not the natural boundary conditions)*. We then integrate over the body:

$$\int_0^L w\left(EA\frac{d^2u}{dx^2}+b\right)dx=0 \tag{3.4}$$

Now separate the integral:

$$EA\int_0^L w\left(\frac{d^2u}{dx^2}\right)dx + \int_0^L wb\,dx = 0 \tag{3.5}$$

Recall integration by parts, where:

$$\frac{d}{dx}\left(w\frac{du}{dx}\right) = \frac{dw}{dx}\frac{du}{dx} + w\frac{d^2u}{dx^2} \tag{3.6}$$

Rearranging gives:

$$w\frac{d^2u}{dx^2} = \frac{d}{dx}\left(w\frac{du}{dx}\right) - \frac{dw}{dx}\frac{du}{dx} \tag{3.7}$$

Substitute (3.7) into (3.5) gives:

$$EA\int_0^L \frac{d}{dx}\left(w\frac{du}{dx}\right)dx - EA\int_0^L \frac{dw}{dx}\frac{du}{dx}dx + \int_0^L wb\,dx = 0 \tag{3.8}$$

By the fundamental theorem of calculus:

$$EA\int_0^L \frac{d}{dx}\left(w\frac{du}{dx}\right)dx = EA\left(w\frac{du}{dx}\right)\Bigg|_L - EA\left(w\frac{du}{dx}\right)\Bigg|_0 \tag{3.9}$$

However, we have specifically chosen w so that it is zero at essential boundary conditions, $x = 0$. As a result, the second term on the right hand side is zero, and (3.9) becomes:

$$EA\int_0^L \frac{d}{dx}\left(w\frac{dT}{dx}\right)dx = w(L)\left(EA\frac{du}{dx}\right)\Bigg|_L = w(L)P \tag{3.10}$$

Substituting into (3.8) gives:

$$\underline{EA\int_0^L \frac{dw}{dx}\frac{du}{dx}dx = \int_0^L wb\,dx + w(L)P} \tag{3.11}$$

Note that the ***weak statement is equivalent to the strong statement***. This is carefully discussed in Hughes, 1987. The advantage of the weak statement is that **we have reduced the order of the differential by one and made possible symmetry of the resulting matrix formulation when we approximate the true solution**. We have also explicitly handled the natural boundary conditions, so that we can include them in our analysis. Note also that the weak form is still a weighted residual statement. The weak form does not show the weighted residual formulation as clearly, but it is there.

Question 1: What are the similarities and differences between the weak form (3.11) and the weighted residual form in previous lecture?

Question 2: What is the advantage of reducing the order of the differentiation of the trial function u?

Questions 3: What is the advantage of having a symmetric stiffness matrix?

3.3 Approximate Solution Using the Weak Formulation

The weak form will be the starting point of our approximate solution. We approximate the solution using a function in which the first known term (or terms) satisfies the essential boundary conditions. The rest of the approximate solution consists of functions that are zero at the essential boundary conditions and are multiplied by coefficients to be determined:

$$u_n = \phi_0 + \sum_{j=1}^{n} a_j \phi_j \tag{3.12}$$

The derivative of the approximate solution is given by:

$$\frac{du_n}{dx} = \frac{d\phi_0}{dx} + \sum_{j=1}^{n} a_j \frac{d\phi_j}{dx} \tag{3.13}$$

We will use the Galerkin approach, so the weighting function will be the same as the approximate function, but without the essential boundary condition terms:

$$w_n = \sum_{i=1}^{n} d_i \phi_i \tag{3.14}$$

and the derivatives of the weighing function are:

$$\frac{dw_n}{dx} = \sum_{i=1}^{n} d_i \frac{d\phi_i}{dx} \tag{3.15}$$

Substituting (3.13) and (3.15) into (3.11):

$$AE\int_0^L \left(\sum_{i=1}^{n} d_i \frac{d\phi_i}{dx} \right)\left(\frac{d\phi_0}{dx} + \sum_{j=1}^{n} a_j \frac{d\phi_j}{dx} \right) dx = \int_0^L \left(\sum_{i=1}^{n} d_i \phi_i \right) b dx + \left(\sum_{i=1}^{n} d_i \phi(L)_i \right) P$$

Rearranging,

$$\sum_{i=1}^{n} d_i \left[AE\int_0^L \left(\frac{d\phi_i}{dx}\right)\left(\frac{d\phi_0}{dx} + \sum_{j=1}^{n} a_j \frac{d\phi_j}{dx}\right)dx - \int_0^L \phi_i b dx - \phi(L)_i P \right] = 0 \qquad (3.16)$$

Since this must hold true for all d_i, then:

$$AE\int_0^L \left(\frac{d\phi_i}{dx}\right)\left(\frac{d\phi_0}{dx} + \sum_{j=1}^{n} a_j \frac{d\phi_j}{dx}\right)dx - \int_0^L \phi_i b dx - \phi(L)_i P = 0$$

or

$$AE\int_0^L \left(\frac{d\phi_i}{dx}\right)\left(\frac{d\phi_0}{dx} + \sum_{j=1}^{n} a_j \frac{d\phi_j}{dx}\right)dx = \int_0^L \phi_i b dx + \phi(L)_i P\,,\ \ i = 1,\ldots,n \qquad (3.17)$$

Since the terms satisfying the essential boundary conditions are known, we will move them to the right hand side also:

$$\underline{AE\int_0^L \left(\frac{d\phi_i}{dx}\right)\left(\sum_{j=1}^{n} a_j \frac{d\phi_j}{dx}\right)dx = \int_0^L \phi_i b dx - AE\int_0^L \left(\frac{d\phi_i}{dx}\right)\left(\frac{d\phi_0}{dx}\right)dx + \phi(L)_i P\,,\ \ i = 1,\ldots,n} \quad (3.18)$$

The term on the left contains the unknown coefficients. The first term on the right comes from body loads. The second term is the contribution of the essential boundary conditions. The third term is the contribution of the natural boundary conditions. Note that it is not necessary that all these terms appear in every problem.

Question 4: In what aspects are (3.18) and (3.11) different?

This is really a set of n equations with n unknowns. We can expand (3.18) to see the individual terms:

$$\begin{bmatrix} EA\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx & EA\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_2}{dx}\right)dx & \ldots & EA\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_n}{dx}\right)dx \\ EA\int_0^L \left(\frac{d\phi_2}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx & EA\int_0^L \left(\frac{d\phi_2}{dx}\right)\left(\frac{d\phi_2}{dx}\right)dx & \ldots & EA\int_0^L \left(\frac{d\phi_2}{dx}\right)\left(\frac{d\phi_n}{dx}\right)dx \\ \ldots & \ldots & \ldots & \ldots \\ EA\int_0^L \left(\frac{d\phi_n}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx & EA\int_0^L \left(\frac{d\phi_n}{dx}\right)\left(\frac{d\phi_2}{dx}\right)dx & \ldots & EA\int_0^L \left(\frac{d\phi_n}{dx}\right)\left(\frac{d\phi_n}{dx}\right)dx \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ \ldots \\ a_n \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ \ldots \\ f_n \end{Bmatrix} \quad (3.19)$$

where:

$$\begin{Bmatrix} f_1 \\ f_2 \\ \cdots \\ f_n \end{Bmatrix} = \begin{Bmatrix} \int_0^L \phi_1 b dx + \phi_1(L)P - \int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_0}{dx}\right) dx \\ \int_0^L \phi_2 b dx + \phi_2(L)P - \int_0^L \left(\frac{d\phi_2}{dx}\right)\left(\frac{d\phi_0}{dx}\right) dx \\ \cdots \\ \int_0^L \phi_N b dx + \phi_N(L)P - \int_0^L \left(\frac{d\phi_N}{dx}\right)\left(\frac{d\phi_0}{dx}\right) dx \end{Bmatrix} \tag{3.20}$$

There is beauty and power here. We now have a general method to develop approximate solutions to differential equations. We must select appropriate shape functions, but as we will see, we can automate that using finite elements. If we write it in matrix form, this will be:

$$\mathbf{Ka} = \mathbf{f} \tag{3.21}$$

This is the basis of our future finite element development. As derived, we have assumed that the shape functions are global functions over the entire body. We will show that the finite element formulation arises when we use local shape functions.

Question 5: From (3.19), list some reasons that the Galerkin's method is better than the other method such as collocation and sub-domain methods.

3.4 Example Solution Using Single Term Approximation

To illustrate obtaining a solution using the weak formulation, let us solve the same problem as done in previous lecture. This is an elastic rod with body and end forces. We assume the approximate solution:

$$u_n = a_1 \phi_1 \tag{3.22}$$

where:

$$\phi_1 = \sin\left(\frac{\pi}{2L} x\right) \tag{3.23}$$

This function was chosen to be zero at the essential boundary condition. The derivative is:

$$\frac{d\phi_1}{dx} = \frac{\pi}{2L}\cos\left(\frac{\pi}{2L} x\right) \tag{3.24}$$

Substituting into (3.19) and (3.20) (assume $A = E = L = P = b = 1$) gives:

$$\left[\frac{\pi^2}{4}\int_0^1\left(\cos\left(\frac{\pi}{2}x\right)\cos\left(\frac{\pi}{2}x\right)\right)dx\right]\{a_1\}=\left\{\int_0^1\sin\left(\frac{\pi}{2}x\right)dx+1\right\} \tag{3.25}$$

Evaluating the integrals:

$$\left[\frac{\pi^2}{4}\left(\frac{1}{2}x+\frac{1}{2\pi}\sin(\pi x)\right)\Bigg|_0^1\right]\{a_1\}=\left\{-\frac{2}{\pi}\cos\left(\frac{\pi}{2}x\right)\Bigg|_0^1+1\right\} \tag{3.26}$$

$$\frac{\pi^2}{8}a_1=\frac{2}{\pi}+1$$
$$\underline{a_1=1.326} \tag{3.27}$$

This is a rough approximation, since we only have one term in the approximate solution.

3.5 Example Solution Using Two Terms

We will now do an example solution using two terms in the approximation. We will add a linear term to our previous solution:

$$u_n=\sum_{j=1}^{2}a_j\phi_j \tag{3.28}$$

where:

$$\phi_1=x \tag{3.29}$$
$$\phi_2=\sin\left(\frac{\pi}{2L}x\right) \tag{3.30}$$

These functions both are zero at the essential boundary condition. The derivatives are:

$$\frac{d\phi_1}{dx}=1 \tag{3.31}$$
$$\frac{d\phi_2}{dx}=\frac{\pi}{2L}\cos\left(\frac{\pi}{2L}x\right) \tag{3.32}$$

(3.20) and (3.21) now become:

$$\begin{bmatrix} EA\int_0^L(1)(1)dx & EA\int_0^L(1)\left(\frac{\pi}{2L}\cos\left(\frac{\pi}{2L}x\right)\right)dx \\ EA\int_0^L\left(\frac{\pi}{2L}\cos\left(\frac{\pi}{2L}x\right)(1)\right)dx & EA\int_0^L\left(\frac{\pi}{2L}\cos\left(\frac{\pi}{2L}x\right)\frac{\pi}{2L}\cos\left(\frac{\pi}{2L}x\right)\right)dx \end{bmatrix}\begin{Bmatrix}a_1\\a_2\end{Bmatrix}=\begin{Bmatrix}f_1\\f_2\end{Bmatrix} \tag{3.33}$$

$$\begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} = \begin{Bmatrix} \int_0^L xb\,dx + LP \\ \int_0^L \sin\left(\frac{\pi}{2L}x\right)b\,dx + 1P \end{Bmatrix} \tag{3.34}$$

Again, assume $A = E = L = P = b = 1$. Performing the integrations in (3.33) and (3.34) gives:

$$\begin{bmatrix} 1 & \frac{\pi}{2} \\ \frac{\pi}{2} & \frac{\pi^2}{8} \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} \frac{1}{2}+1 \\ \frac{2}{\pi}+1 \end{Bmatrix} \tag{3.35}$$

Solving:

$$\underline{\begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} 0.915 \\ 0.584 \end{Bmatrix}} \tag{3.36}$$

Both solutions are plotted in Figure 3-1.

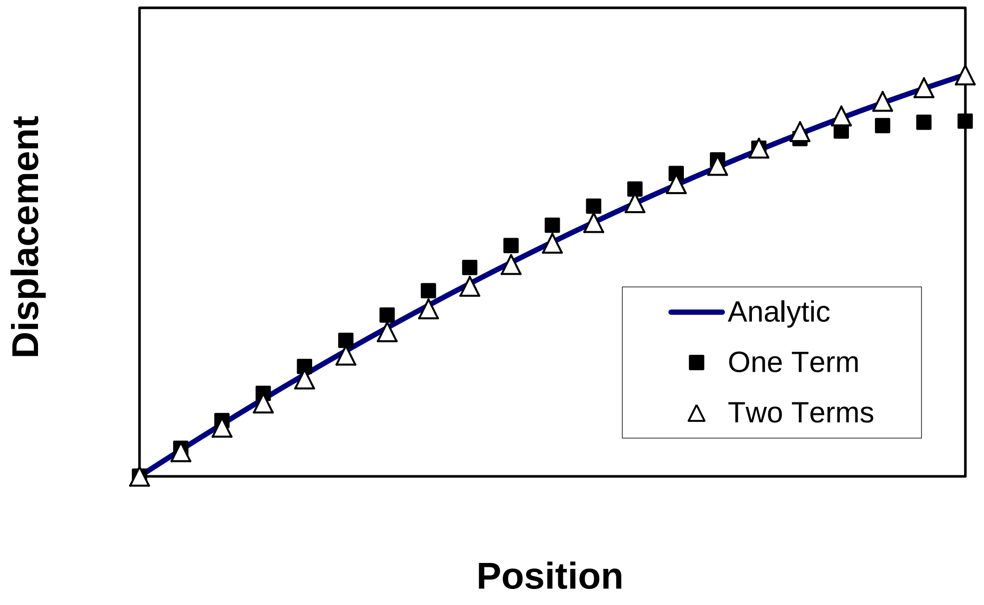


Figure 3-1. Comparison of solutions

3.6 Hints to Questions

Q1: (omit)
Q2: Allows for the use of piecewise function; weakens the requirements of differentiability.
Q3: Increases computational efficiency; reduce storage requirement.

Q4: One is exact, while the other is approximate.
Q5: (omit).

3.7 References

Hughes, T. J. R., 1987, *The Finite Element Method. Linear Static and Dynamic Finite Element Analysis*, Chapter 1, Prentice-Hall, Inc.

4 Hat Functions

4.1 Objective

Our objective is to use hat functions as a prelude to defining a finite element and its associated shape functions.

We will first describe hat functions. We will then show that hat functions are equivalent to defining shape functions over an element. The student will also see that the global integration is equivalent to the summation of integration over all elements.

4.2 Hat Functions

We are now approaching the final definition of a finite element. Before we get there, we will first look at special hat functions that are global functions, but only non-zero locally. We will use these functions in our previously derived Galerkin weighted residual formulation that we developed from the weak form of the differential equation. Again, we will be solving the elastic bar. We will define nodes on the bar as shown below:

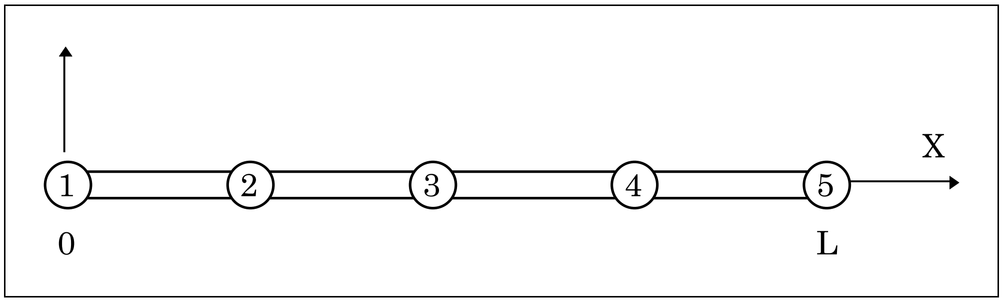


Figure 4-1:Discretization for hat functions

The associated hat functions are shown in Figure 4-2.

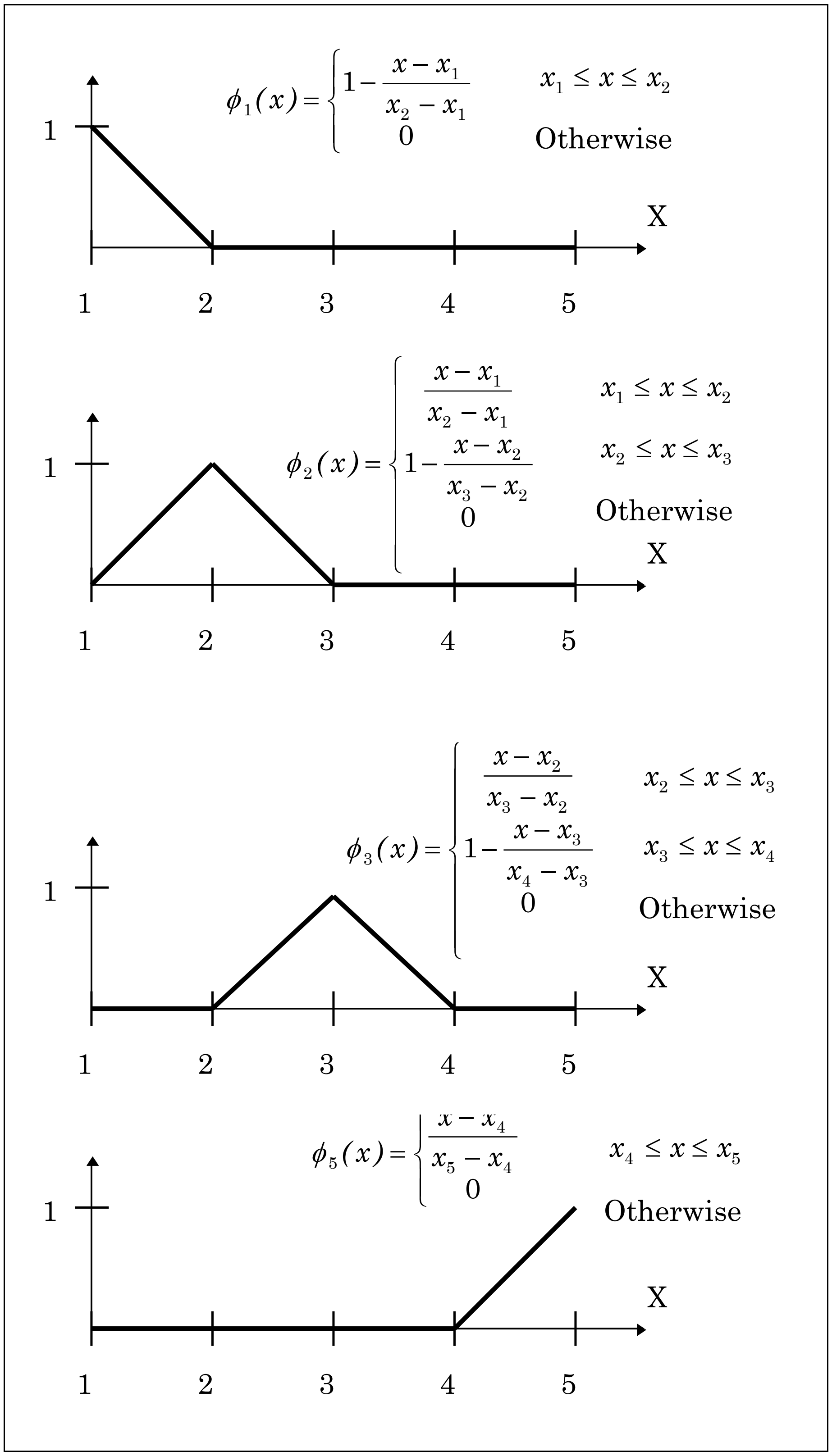

$\phi_1(x)=\begin{cases}1-\dfrac{x-x_1}{x_2-x_1} & x_1\le x\le x_2\\ 0 & \text{Otherwise}\end{cases}$
$\phi_2(x)=\begin{cases}\dfrac{x-x_1}{x_2-x_1} & x_1\le x\le x_2\\ 1-\dfrac{x-x_2}{x_3-x_2} & x_2\le x\le x_3\\ 0 & \text{Otherwise}\end{cases}$
$\phi_3(x)=\begin{cases}\dfrac{x-x_2}{x_3-x_2} & x_2\le x\le x_3\\ 1-\dfrac{x-x_3}{x_4-x_3} & x_3\le x\le x_4\\ 0 & \text{Otherwise}\end{cases}$
$\phi_5(x)=\begin{cases}\dfrac{x-x_4}{x_5-x_4} & x_4\le x\le x_5\\ 0 & \text{Otherwise}\end{cases}$
1
X
1 2 3 4 5

Figure 4-2: Hat functions (Node 4 skipped)

Note that once the hat function for x=0, $\phi_0(x)$ for $x_{-1} < x < x_1$, is defined, i.e., the hat function centered at x_i can be simply expressed by $\phi_i(x) = \phi_0(x - x_i)$

The hat functions can be used to create a multi-linear (piecewise linear) curve by adding them together and using different coefficients multiplying each hat function, as illustrated in Figure 4-3.

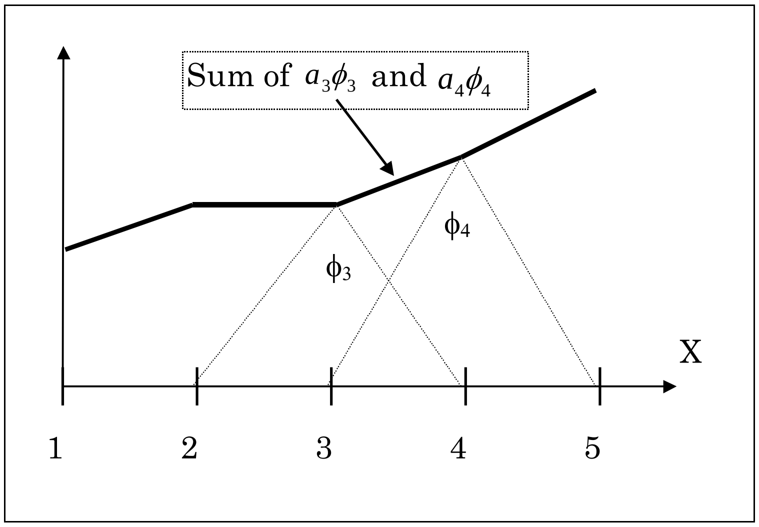


Figure 4-3: Multi-linear curve generated using hat functions with variable coefficients

Question 1: Compare the hat function $\phi_i(x)$ *and the Dirac delta function* δ *:*

$$f(x) \cong \sum_{i}^{N} f(x_i)\phi_i(x) = \sum_{i}^{N} f(x_i)\phi_0(x - x_i)$$

$$f(x) = \int_{-\infty}^{\infty} f(\xi)\delta(x - \xi)d\xi$$

4.3 Solution Using Hat Functions

We will now use the hat functions to solve a problem. We will use the Galerkin method developed in Lecture 3. The problem we will solve is the same elastic rod problem with body and end loads. The rod is divided as shown in Figure 4-1.

The approximate solution is developed using the hat functions shown in Figure 4-2. Note that there is an essential boundary condition at node 1 and a natural boundary condition at node 5. The value of the essential boundary conditions is zero. Recall that the first term (terms) in the approximate solution satisfy the essential boundary conditions, the rest are zero at the essential boundary conditions. Because the essential boundary condition is zero, we do not need any special functions to satisfy it. However, note that using the function:

$$\phi_0(x) = u_1\phi_1(x) \tag{4.1}$$

Would satisfy the essential boundary condition and could be used in a problem where the essential boundary condition is not zero (see the homework).

Then, the hat functions associated with nodes 2, 3, 4, and 5 will be the ones we will use in our approximate solution.

$$\boxed{u_n(x) = a_2\phi_2(x) + a_3\phi_3(x) + a_4\phi_4(x) + a_5\phi_5(x)} \tag{4.2}$$

The above numbering system corresponds to that in Figure 4-1 and Figure 4-2. For fitting the approximating function to the matrix form of the Galerkin method derived using the weak form of the differential equation in Lecture 3, we will recognize that we should map 2 to 1, 3 to 2, 4 to 3, and 5 to 4. That is, $n = 4$. Then,

$$\phi_1 = \begin{cases} \dfrac{x - x_1}{x_2 - x_1} & \text{for } x_1 \le x \le x_2 \\ 1 - \dfrac{x - x_2}{x_3 - x_2} & \text{for } x_2 \le x \le x_3 \end{cases} \tag{4.3}$$

and,

$$\frac{d\phi_1}{dx} = \begin{cases} \dfrac{1}{x_2 - x_1} & \text{for } x_1 \le x \le x_2 \\ \dfrac{-1}{x_3 - x_2} & \text{for } x_2 \le x \le x_3 \end{cases} \tag{4.4}$$

We will now evaluate the first row of the in the Galerkin formulation of Lecture 3. The first coefficient is given by (where we have assumed $E = A = 1$):

$$k_{11} = \int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx \tag{4.5}$$

For ϕ_1 , which is zero except for $x_1 \le x \le x_3$, we will divide the integral into two parts as shown below:

$$\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx = \int_0^{x_2} \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx + \int_{x_2}^{x_3} \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx \tag{4.6}$$

Substituting (4.4) into (4.6) and using the values for the positions of the nodes ($L = 1$), gives:

$$\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx = \int_0^{0.25} \left(\frac{1}{0.25}\right)\left(\frac{1}{0.25}\right)dx + \int_{0.25}^{0.5} \left(\frac{-1}{0.25}\right)\left(\frac{-1}{0.25}\right)dx \tag{4.7}$$

Evaluating the integrals,

$$\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx = 16(x)\Big|_0^{0.25} + 16(x)\Big|_{0.25}^{0.5}$$

$$\underline{\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx = 8 = k_{11}} \qquad (4.8)$$

The second term in the Galerkin formulation of Lecture 3 is given by:

$$\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_2}{dx}\right)dx \qquad (4.9)$$

Carefully examining Figure 4-2, we see that the only region over the body that the two approximating functions interact is over the region $x_2 \le x \le x_3$. Then the integral becomes:

$$\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_2}{dx}\right)dx = \int_{x_2}^{x_3} \left(\frac{-1}{x_3 - x_2}\right)\left(\frac{1}{x_3 - x_2}\right)dx \qquad (4.10)$$

Evaluating:

$$\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_2}{dx}\right)dx = \int_{0.25}^{0.5} \left(\frac{-1}{0.25}\right)\left(\frac{1}{0.25}\right)dx$$

or,

$$\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_2}{dx}\right)dx = -16x\Big|_{0.25}^{0.5}$$

and,

$$\underline{\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_2}{dx}\right)dx = -4 = k_{12}} \qquad (4.11)$$

The third term in the Galerkin formulation of Lecture 3 is given by:

$$\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_3}{dx}\right)dx \qquad (4.12)$$

Looking again at Figure 4-2, we see that the two shape functions do not interact. Therefore, the integral is zero.

The remainder of the integrals can be evaluated by the student.

We will also evaluate the right hand side of the Galerkin equation developed in Lecture 3. The first term to be evaluated is the one that includes the body force:

$$\int_0^L \phi_1 dx \tag{4.13}$$

where we have assumed $b = 1$. As before, we divide the integral into two parts:

$$\int_0^L \phi_1 dx = \int_0^{x_2} \left(\frac{x - x_1}{x_2 - x_1} \right) dx + \int_{x_2}^{x_3} \left(1 - \frac{x - x_2}{x_3 - x_2} \right) dx \tag{4.14}$$

Evaluating:

$$\int_0^L \phi_1 dx = \int_0^{0.25} \left(\frac{x - 0}{0.25} \right) dx + \int_{0.25}^{0.5} \left(1 - \frac{x - 0.25}{0.25} \right) dx$$

Integrating,

$$\int_0^L \phi_1 dx = 4\frac{x^2}{2}\bigg|_0^{0.25} + \left(2x - 2x^2\right)\Big|_{0.25}^{0.5} dx$$

or,

$$\underline{\int_0^L \phi_1 dx = \frac{1}{8} + \frac{1}{8} = \frac{1}{4}} \tag{4.15}$$

The second term on the right hand side is:

$$\phi_1(L)P \tag{4.16}$$

Clearly, $\phi_1(L) = 0$, so the natural boundary condition term is zero. For this problem, the essential boundary conditions are zero, therefore that term also does not contribute anything.

Similar integrations are performed for the other terms. These should be done by the student. The only row in which the natural boundary condition term contributes is for 4, where $\phi_4(L)P = 1$. When all integrations are completed, the coefficients obtained using the hat functions will be those given below:

$$\begin{bmatrix} 8 & -4 & 0 & 0 \\ -4 & 8 & -4 & 0 \\ 0 & -4 & 8 & -4 \\ 0 & 0 & -4 & 4 \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{Bmatrix} = \begin{Bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/8 + 1 \end{Bmatrix} \tag{4.17}$$

Solving (4.17) gives:

$$\mathbf{a} = \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{Bmatrix} = \begin{Bmatrix} 0.0 \\ 0.46875 \\ 1.21875 \\ 1.5 \end{Bmatrix} \tag{4.18}$$

The comparison with the analytic solution is given in Figure 4-4. As for the finite difference solution, the solution is exact at the nodes. This is only true in the one-dimensional case.

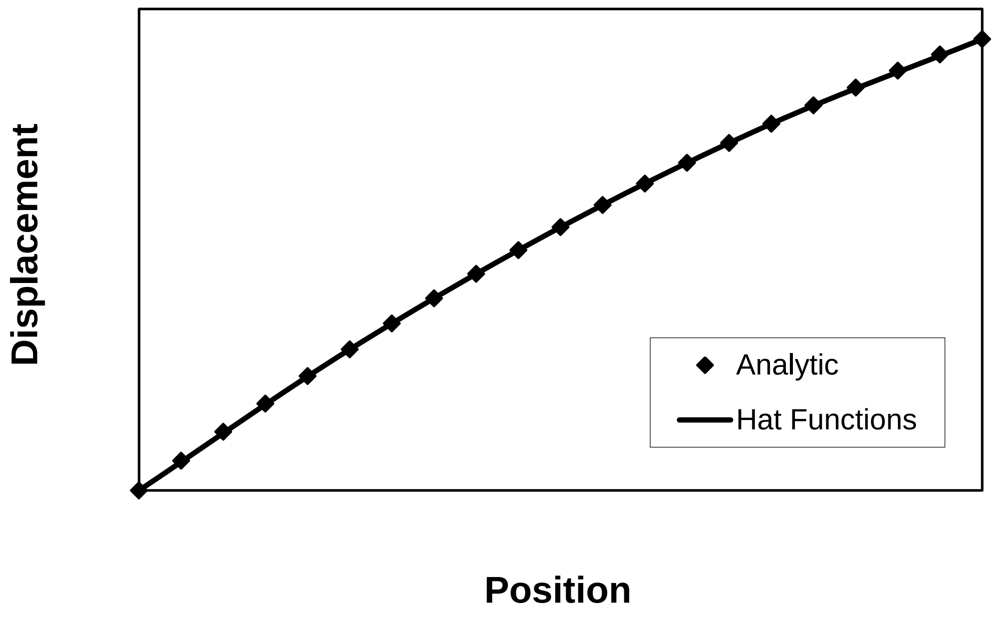


Figure 4-4: Comparison of analytic and hat function solutions

We have now obtained a multi-linear approximation to the solution.

Question 2: Compare the hat function approach with the multiple term approach in Lecture 3. What are the respective advantages and disadvantages?

4.4 Hints to Questions

Q1: Similar in concept; discrete versus continuous.
Q2: Think of computational efficiency and smoothness of the function.

4.5 References

Hughes, T. J. R., 1987, *The Finite Element Method. Linear Static and Dynamic Finite Element Analysis*, Chapter 1, Prentice-Hall, Inc.

5 A First Finite Element and Shape Functions

5.1 Objective

Our objective is to develop our first finite element approximation and to demonstrate a general method of obtaining element shape functions in natural coordinates. This is an extension of the hat functions developed in Lecture 4.

We first discuss the desirable characteristics of a numerical technique for analysis of general problems. We then derive the shape functions for an element and show the equivalence between global integration using hat functions and the summation of integrals over each element. We also introduce the concept of nature coordinates, and demonstrate how to use the Lagrange polynomials to derive the shape function.

5.2 Desired Characteristics of an Effective Computational Method

As discussed in Reddy (*An Introduction to the Finite Element Method*, 2nd Edition, 1993), "an effective computational method should have the following characteristics:

- It should have a sound mathematical as well as physical basis (i.e., yield convergent solutions and be applicable to practical problems).
- It should not have limitations with regard to the geometry, the physical composition of the domain, or the nature of the loading.
- The formulative procedure should be independent of the shape of the domain and the specific form of the boundary conditions.
- The method should be flexible enough to allow different degrees of approximation without reformulating the entire problem.
- It should involve a systematic procedure that can be automated for use on digital computers".

These criteria are met by the finite element method.

5.3 A Linear Finite Element

We will now begin our derivation of a simple finite element. Our first element will be a linear element with two nodes, Figure 5-1.

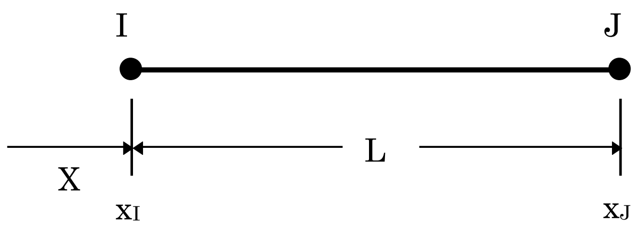


Figure 5-1: Simple element

We will assume that the displacement in the element is linear:

$$u = c_1 + c_2 x \tag{5.1}$$

Then, we can evaluate the displacements at the nodes using the coordinates of the nodes:

$$u_I = c_1 + c_2 x_I \tag{5.2}$$
$$u_J = c_1 + c_2 x_J \tag{5.3}$$

We can assemble in matrix form and solve:

$$\begin{bmatrix} 1 & x_I \\ 1 & x_J \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} = \begin{Bmatrix} u_I \\ u_J \end{Bmatrix} \tag{5.4}$$

Solving:

$$\begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} = \left(\frac{1}{x_J - x_I} \right) \begin{bmatrix} x_J & -x_I \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_I \\ u_J \end{Bmatrix} \tag{5.5}$$

Substituting the solution back into (5.1), the displacement is now given by:

$$u = \frac{x_J u_I - x_I u_J}{x_J - x_I} + \left(\frac{u_J - u_I}{x_J - x_I} \right) x \tag{5.6}$$

Rearranging:

$$u = \left(\frac{x_J - x}{x_J - x_I} \right) u_I + \left(\frac{x - x_I}{x_J - x_I} \right) u_J \tag{5.7}$$

This is a simple, but important result. Given the nodal displacements (u_I and u_J) and a position on the element (x), we can interpolate to find the displacement at that point. The interpolation functions are typically called “shape functions”:

$$u(x) = N_I(x)u_I + N_J(x)u_J \text{ or } u(x) = \sum_i N_i(x)u_i \tag{5.8}$$

These are plotted in Figure 5-2. Both are linear and both range from a value of 1 at the node they are associated with to 0 at the other node.

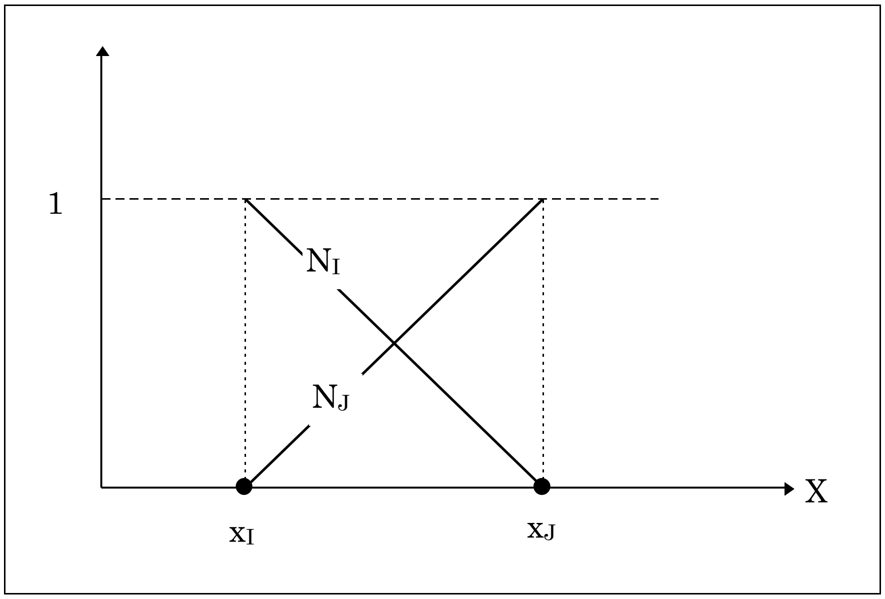


Figure 5-2: Linear shape functions

We can now see the connection between the shape functions and the hat functions we used in our previous lecture. They are in fact the same over the interior of the element, Figure 5-3.

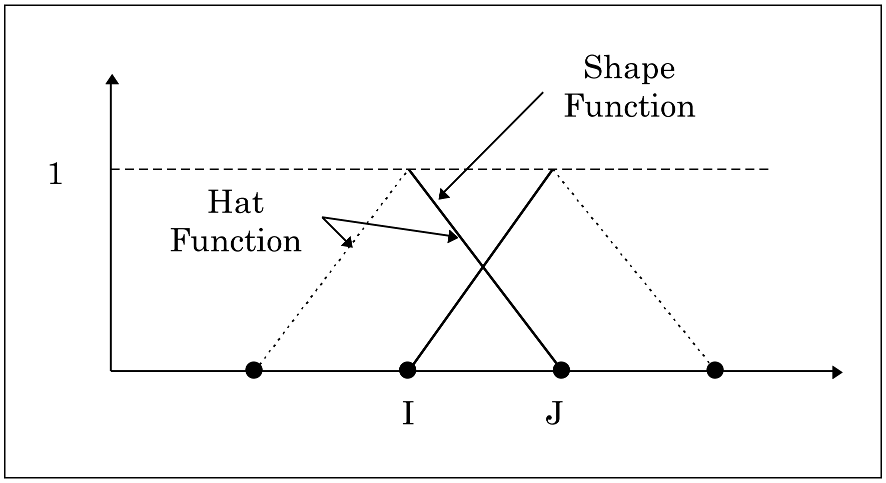


Figure 5-3: Comparison of hat and shape functions

Since the hat functions are non-zero locally, the only non-zero terms over the interval I to J occur with the two element shape functions. Therefore, integrating over each element and adding, give the same result as integrating over the entire length.

5.4 Equivalence Between Global Integration of Hat Functions and Summing the Integration of Element Shape Functions

With such a long title, this section must be significant. It is, in the sense that it shows that using hat functions and integrating globally is equivalent to integrating over each element, then summing the integrals. We will first show the global hat function and element shape function integrations schematically, Figure 5-4.

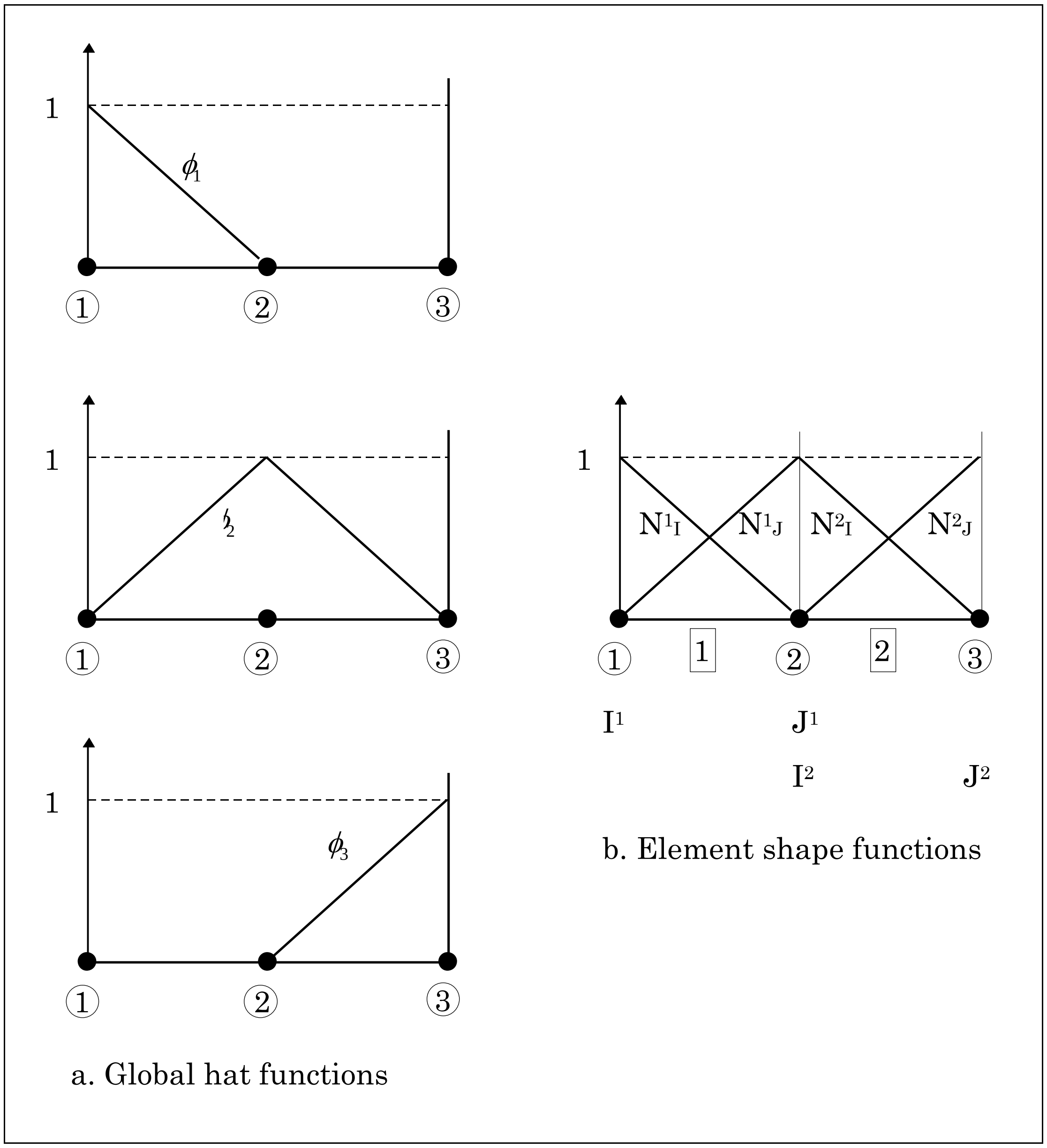


Figure 5-4: Global and element shape functions

We have already derived the Galerkin approximation in Lecture 3:

$$\begin{bmatrix} EA\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx & EA\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_2}{dx}\right)dx & EA\int_0^L \left(\frac{d\phi_1}{dx}\right)\left(\frac{d\phi_3}{dx}\right)dx \\ EA\int_0^L \left(\frac{d\phi_2}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx & EA\int_0^L \left(\frac{d\phi_2}{dx}\right)\left(\frac{d\phi_2}{dx}\right)dx & EA\int_0^L \left(\frac{d\phi_2}{dx}\right)\left(\frac{d\phi_3}{dx}\right)dx \\ EA\int_0^L \left(\frac{d\phi_3}{dx}\right)\left(\frac{d\phi_1}{dx}\right)dx & EA\int_0^L \left(\frac{d\phi_3}{dx}\right)\left(\frac{d\phi_2}{dx}\right)dx & EA\int_0^L \left(\frac{d\phi_3}{dx}\right)\left(\frac{d\phi_3}{dx}\right)dx \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \end{Bmatrix} \quad (5.9)$$

Our previous work has made us familiar with this approach and has demonstrated how well it works.

We now show that the same result is obtained by integrating element shape functions and then summing the integrals. For the element, the form of the integrals is the same as in (5.9). For element 1 (I=1, J=2) and the integral is given by:

$$\begin{bmatrix} EA\int_0^{L^1} \left(\frac{dN_I^1}{dx}\right)\left(\frac{dN_I^1}{dx}\right)dx & EA\int_0^{L^1} \left(\frac{dN_I^1}{dx}\right)\left(\frac{dN_J^1}{dx}\right)dx \\ EA\int_0^{L^1} \left(\frac{dN_J^1}{dx}\right)\left(\frac{dN_I^1}{dx}\right)dx & EA\int_0^{L^1} \left(\frac{dN_J^1}{dx}\right)\left(\frac{dN_J^1}{dx}\right)dx \end{bmatrix} \begin{Bmatrix} u_I^1 \\ u_J^1 \end{Bmatrix} = \begin{Bmatrix} f_I^1 \\ f_J^1 \end{Bmatrix} \quad (5.10)$$

For element 2 (I=2, J=3) the integral is given by:

$$\begin{bmatrix} EA\int_0^{L^2} \left(\frac{dN_I^2}{dx}\right)\left(\frac{dN_I^2}{dx}\right)dx & EA\int_0^{L^2} \left(\frac{dN_I^2}{dx}\right)\left(\frac{dN_J^2}{dx}\right)dx \\ EA\int_0^{L^2} \left(\frac{dN_J^2}{dx}\right)\left(\frac{dN_I^2}{dx}\right)dx & EA\int_0^{L^2} \left(\frac{dN_J^2}{dx}\right)\left(\frac{dN_J^2}{dx}\right)dx \end{bmatrix} \begin{Bmatrix} u_I^2 \\ u_J^2 \end{Bmatrix} = \begin{Bmatrix} f_I^2 \\ f_J^2 \end{Bmatrix} \quad (5.11)$$

Here x is understood as measured locally based on the left node of element 2. We now add the two element integrals together to assemble a global stiffness matrix. As this is done we must map from the element nodes (I, J) to the global nodes (1, 2, 3). For instance, for element 1, I=1 and J=2. Thus the I,I integral term for element 1 will go in the 1,1 global coefficient, and the I,J integral term will go in the 1,2 global coefficient. Similarly, for element 2 where I=2 and J=3, the I,I term will go to the 2,2 global coefficient and the I,J term will go to the 2,3 coefficient. When all terms are summed, we get the matrix in (5.12). The reader should verify this.

$$\begin{bmatrix} \int_0^{L^1}\left(\frac{dN_I^1}{dx}\right)\left(\frac{dN_I^1}{dx}\right)dx & \int_0^{L^1}\left(\frac{dN_I^1}{dx}\right)\left(\frac{dN_J^1}{dx}\right)dx & 0 \\ \int_0^{L^1}\left(\frac{dN_{J1}^1}{dx}\right)\left(\frac{dN_{I1}^1}{dx}\right)dx & \left(\int_0^{L^1}\left(\frac{dN_J^1}{dx}\right)\left(\frac{dN_J^1}{dx}\right)dx + \int_0^{L^2}\left(\frac{dN_I^2}{dx}\right)\left(\frac{dN_I^2}{dx}\right)dx\right) & \int_0^{L^2}\left(\frac{dN_I^2}{dx}\right)\left(\frac{dN_J^2}{dx}\right)dx \\ 0 & \int_0^{L^2}\left(\frac{dN_J^2}{dx}\right)\left(\frac{dN_I^2}{dx}\right)dx & \int_0^{L^2}\left(\frac{dN_J^2}{dx}\right)\left(\frac{dN_J^2}{dx}\right)dx \end{bmatrix} \quad (5.12)$$

Note in the above expression, the integration variable x should be measured relative to the left node of the element. Now, using Figure 5-4, the reader can see that (5.9) and (5.12) are equivalent. Because the hat and element shape functions are local, we can integrate over each element and sum the element contributions. Comparison between (5.9) and (5.12) indicates that integration using global hat functions is equivalent to the sum of integration over the element shape function. In the following lectures, we will show how to perform numerical integration for the integrals in the stiffness matrix and, ultimately, how to solve the system equations of FEM. At this point, however, it is helpful to devote some attention to the concept of natural coordinate and a general method to derive shape functions.

Question 1: What are the advantages of using elment shape functions over using the global hat functions?

5.5 Shape Functions in Natural Coordinates

5.5.1 Linear Shape Functions

In the preceeding sections, we derived shape functions for a linear element in the actual element coordinates:

$$\lfloor \mathbf{N} \rfloor = \lfloor N_1 \quad N_2 \rfloor = \left\lfloor \left(\frac{x_J - x}{x_J - x_I}\right) \quad \left(\frac{x - x_I}{x_J - x_I}\right) \right\rfloor \quad (5.13)$$

We will now introduce the idea of natural coordinates. Our natural coordinates will run from -1 to 1. That is, in the natural coordinate system, $\zeta_I = -1$ and $\zeta_J = 1$.

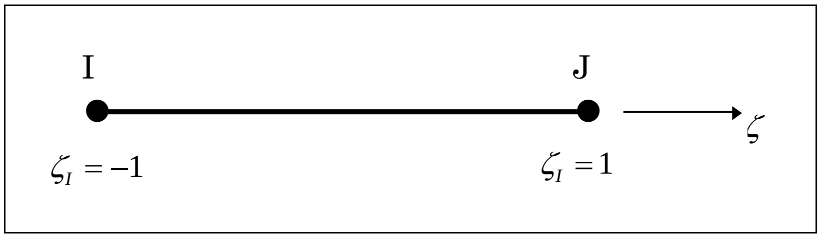


Figure 5-5: Linear element

The natural coordinate system turns out to be a convenient way to perform numerical integration. We will use a mapping from the natural coordinates to the real coordinates for the element. In one dimensional system, there is a very simple relation between the actual coordinates and the natural coordinates: $x = \text{(midpoint of element)} + \frac{\zeta}{2} \times \text{(length of element)}$. For two dimensional system, we will introduce the isoparametric elements for which the mapping between the actual coordinates and natural coordinates is performed conveniently using the shape functions for the displacements.

In the natural coordinate system, (5.13) becomes:

$$\lfloor \mathbf{N} \rfloor = \left\lfloor \left(\frac{1-\zeta}{2}\right) \quad \left(\frac{\zeta+1}{2}\right) \right\rfloor \tag{5.14}$$

As illustrated in Figure 5-6, these shape functions have a value of 1 at the associated node and 0 at the other node. They interpolate linearly.

Question 2: What are the advantages of using the natural coordinates?

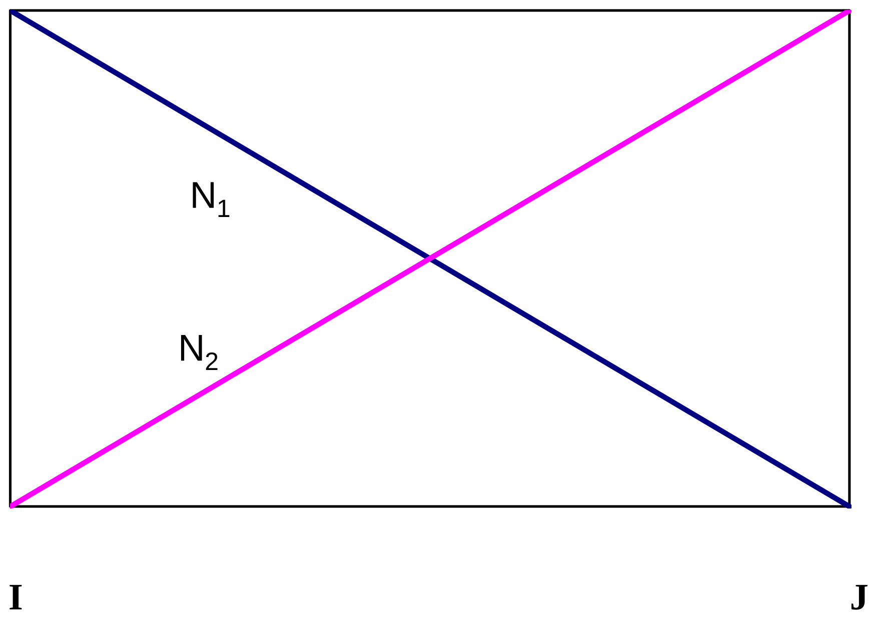


Figure 5-6: Linear shape functions

5.5.2 Quadratic Shape Functions

If we use quadratic interpolation over an element, we have three coefficients to solve for, so we use three nodes (Figure 5-7).

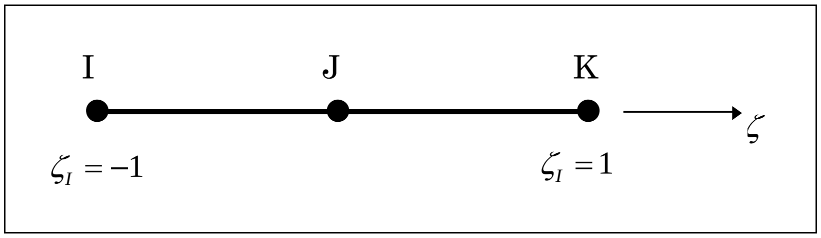


Figure 5-7 Quadratic element

As for the linear element we assume a quadratic interpolation function:

$$u = c_1 + c_2\zeta + c_3\zeta^2 \tag{5.15}$$

Since the displacements at the three nodes are known, we can write:

$$\begin{aligned} u_I &= c_1 + c_2(-1) + c_3(-1)^2 \\ u_J &= c_1 \\ u_K &= c_1 + c_2(1) + c_3(1)^2 \end{aligned} \tag{5.16}$$

or in matrix form,

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \end{Bmatrix} = \begin{Bmatrix} u_I \\ u_J \\ u_K \end{Bmatrix} \tag{5.17}$$

Solving:

$$\begin{Bmatrix} c_1 \\ c_2 \\ c_3 \end{Bmatrix} = \begin{Bmatrix} u_J \\ \frac{1}{2}(u_I - 2u_J + u_K) \\ \frac{1}{2}(u_K - u_I) \end{Bmatrix} \tag{5.18}$$

Substituting into (5.15) gives:

$$u = u_J + \frac{1}{2}(u_I - 2u_J + u_K)\zeta + \frac{1}{2}(u_K - u_I)\zeta^2 \tag{5.19}$$

Finally, collecting terms gives:

$$u = \left(\frac{\zeta}{2}\right)(\zeta - 1)u_I + \left(1 - \zeta^2\right)u_J + \left(\frac{\zeta}{2}\right)(\zeta + 1)u_K \tag{5.20}$$

or,

$$u = N_I u_I + N_J u_J + N_K u_K \tag{5.21}$$

Where we now have derived the quadratic shape functions. They are plotted in Figure 5-8.

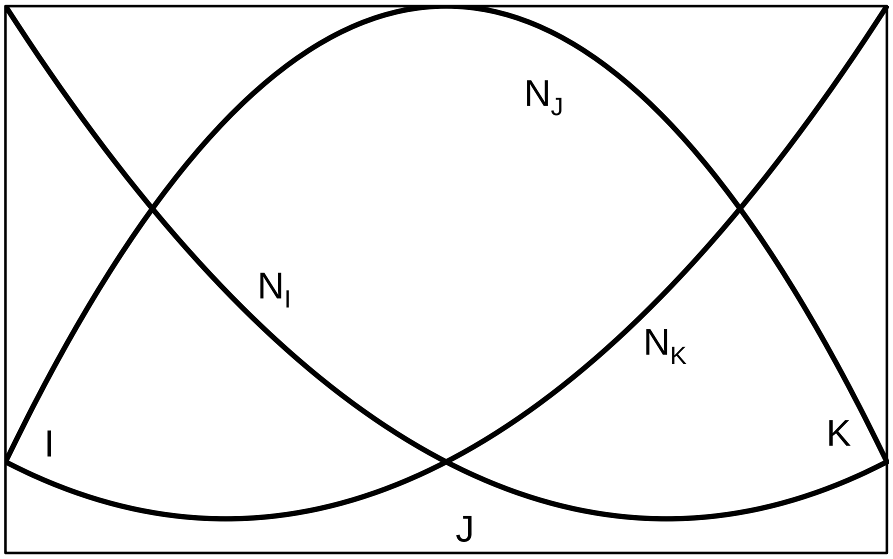


Figure 5-8: Quadratic shape functions

5.6 Lagrange Polynomials

Interpolation functions obtained using the dependent unknown (not its derivatives) are Lagrange interpolation functions. They have the properties (Reddy):

1. $N_i(\zeta_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$
2. $\sum N_i(\zeta) = 1$

It is easy to prove the above relations from the concept of shape functions. Using the shape function, the displacement u can be expressed as

$$u(x) = \sum_i^n N_i(x) u_i$$

where u_i is the displacement at node i, $N_i(x)$ is the shape function for node i. Since the expression should hold for rigid body displacement, inserting $u(x) = \text{const.}$ into the above equation, we find that $\sum_i^n N_i(x) = 1$, or $\sum N_i(\zeta) = 1$ in natural coordinates. As interpolation function, $u(x)$ should become u_i when x is at node i. This indicates that $N_i(x_i)=1$ and $N_j(x_i)=0$. Property 2 also indicates that the sum of the derivatives of the shape function is zero, i.e. $\sum \frac{dN_i(\zeta)}{d\zeta}=0$.

We have seen the above properties in the functions we derived directly. The shape function is equal to 1 at its associated node and 0 at the other nodes. Thus the coefficient multiplying a shape is the actual value at a node. Also, the sum of all the shape functions at any position is 1. As a result, if the displacements at the nodes are the same, the values between the nodes will also be constant.

In general, the Lagrange functions are derived for any number of nodes, n. We first define a function that is zero at all nodes except the ith node:

$$N_i = c_i(\zeta-\zeta_1)(\zeta-\zeta_2)\cdots(\zeta-\zeta_{i-1})(\zeta-\zeta_{i+1})\cdots(\zeta-\zeta_n) \qquad (5.22)$$

Where the subscripts indicate the value of ζ at that node. In addition, we force this function to equal 1 at node i:

$$1 = c_i(\zeta_i-\zeta_1)(\zeta_i-\zeta_2)\cdots(\zeta_i-\zeta_{i-1})(\zeta_i-\zeta_{i+1})\cdots(\zeta_i-\zeta_n) \qquad (5.23)$$

We then solve (5.23) for c_i and substitute into (5.22) to get:

$$N_i = \frac{(\zeta-\zeta_1)(\zeta-\zeta_2)\cdots(\zeta-\zeta_{i-1})(\zeta-\zeta_{i+1})\cdots(\zeta-\zeta_n)}{(\zeta_i-\zeta_1)(\zeta_i-\zeta_2)\cdots(\zeta_i-\zeta_{i-1})(\zeta_i-\zeta_{i+1})\cdots(\zeta_i-\zeta_n)} \qquad (5.24)$$

As an illustration of the use of (5.24), let us obtain the shape functions for $n=3$. Then:

$$N_1(\zeta) = \frac{(\zeta-\zeta_2)(\zeta-\zeta_3)}{(\zeta_1-\zeta_2)(\zeta_1-\zeta_3)} \qquad (5.25)$$

Similarly at node 2:

$$N_2(\zeta) = \frac{(\zeta-\zeta_1)(\zeta-\zeta_3)}{(\zeta_2-\zeta_1)(\zeta_{2i}-\zeta_3)} \qquad (5.26)$$

at node 3:

$$N_3(\zeta) = \frac{(\zeta-\zeta_1)(\zeta-\zeta_2)}{(\zeta_3-\zeta_1)(\zeta_3-\zeta_2)} \qquad (5.27)$$

that is

$$N_1(\zeta) = \frac{\zeta}{2}(\zeta - 1) \tag{5.28}$$

$$N_2(\zeta) = (1 - \zeta^2) \tag{5.29}$$

$$N_3(\zeta) = \frac{\zeta}{2}(\zeta + 1) \tag{5.30}$$

As expected, these shape functions are the same as we derived directly.

The method of deriving the shape function from Lagrange polynomials can be easily extended to 2 or 3 dimensions. We will work on that in later lectures.

5.7 Hints to Questions

Q1: Unified expression; convenience in developing higher order interpolation.
Q2: Intuitive appeal; convenience for numerical integration.

5.8 References

Hughes, T. J. R., 1987, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Chapter 1, Prentice-Hall, Inc.

6 Generalized Matrix Form of the Finite Element Equations

6.1 Purpose

Our purpose is to write the finite element equations using matrix notation. We will then show how the element stiffness matrix and load vector is obtained.

We will again use the one-dimensional elastic rod as our prototype for the derivation. We will then calculate the element stiffness matrix and load vector for a simple linear element.

6.2 Matrix Derivation of Equations

We start from the weak statement of the differential equation (Lecture 3):

$$EA\int_0^L \frac{dw}{dx}\frac{du}{dx}dx = \int_0^L wb\,dx + w(L)P \tag{6-1}$$

with the boundary conditions:

$$u\big|_{x=0} = 0 \qquad \text{(an essential boundary condition)} \tag{6-2}$$

$$AE\frac{du}{dx}\bigg|_{x=L} = P \qquad \text{(a natural boundary condition)} \tag{6-3}$$

As before, we approximate the solution using a function in which the first known term (or terms) combines to satisfy the essential boundary conditions. The rest of the approximate solution consists of functions that are zero at the essential boundary conditions and are multiplied by coefficients to be determined. We will represent the shape functions using N rather than ϕ, to show that these shape functions are only locally non-zero. Then:

$$u^{(m)} = N_0u_o + N_1u_1 + N_2u_2 + ... + N_mu_m \tag{6-4}$$

or:

$$u^{(m)} = N_0u_o + \lfloor \mathbf{N} \rfloor\{\mathbf{u}\} \tag{6-5}$$

We also need the derivative of the displacements:

$$\frac{du^{(m)}}{dx} = \frac{dN_0}{dx}u_o + \left\lfloor \frac{d\mathbf{N}}{dx} \right\rfloor\{\mathbf{u}\}$$

or:

$$\frac{du^{(m)}}{dx} = B_0 u_o + \lfloor \mathbf{B} \rfloor \{\mathbf{u}\} \tag{6-6}$$

Where the $\mathbf{B}$ matrix contains the derivatives of the $\mathbf{N}$ matrix.

We will use the Galerkin method, so we use the same functions for the weighting function as we used to approximate the solution. As before, we do not include the term that satisfies the essential boundary condition. Then, the weighting function is given by:

$$w^{(m)} = \lfloor \mathbf{N} \rfloor \{\mathbf{d}\} \tag{6-7}$$

Note that since $w^{(m)}$ is a scalar, we may also write:

$$w^{(m)} = \{\mathbf{d}\}^{\mathbf{T}} \lfloor \mathbf{N} \rfloor^{\mathbf{T}} \tag{6-8}$$

The derivative of the weighting function is given by:

$$\frac{dw^{(m)}}{dx} = \{\mathbf{d}\}^{\mathbf{T}} \lfloor \mathbf{B} \rfloor^{\mathbf{T}} \tag{6-9}$$

Substituting 6-9 and 6-6 into 6-1:

$$AE\int_0^L \{\mathbf{d}\}^{\mathbf{T}} \lfloor \mathbf{B} \rfloor^{\mathbf{T}} \left(B_0 u_0 + \lfloor \mathbf{B} \rfloor \{\mathbf{u}\}\right) dx = \int_0^L \{\mathbf{d}\}^{\mathbf{T}} \lfloor \mathbf{N} \rfloor^{\mathbf{T}} b\, dx + \{\mathbf{d}\}^{\mathbf{T}} \lfloor \mathbf{N}(L) \rfloor^{\mathbf{T}} P \tag{6-10}$$

Rearranging:

$$\{\mathbf{d}\}^{\mathbf{T}} \left[AE\int_0^L \lfloor \mathbf{B} \rfloor^{\mathbf{T}} \left(B_0 u_0 + \lfloor \mathbf{B} \rfloor \{\mathbf{u}\}\right) dx - \int_0^L \lfloor \mathbf{N} \rfloor^{\mathbf{T}} b\, dx - \lfloor \mathbf{N}(L) \rfloor^{\mathbf{T}} P \right] = 0 \tag{6-11}$$

Since this must hold true for all $\{\mathbf{d}\}$, then:

$$\left[AE\int_0^L \lfloor \mathbf{B} \rfloor^{\mathbf{T}} \left(B_0 u_0 + \lfloor \mathbf{B} \rfloor \{\mathbf{u}\}\right) dx - \int_0^L \lfloor \mathbf{N} \rfloor^{\mathbf{T}} b\, dx - \lfloor \mathbf{N}(L) \rfloor^{\mathbf{T}} P \right] = 0 \tag{6-12}$$

or

$$AE\int_0^L \lfloor \mathbf{B} \rfloor^{\mathbf{T}} \lfloor \mathbf{B} \rfloor \{\mathbf{u}\} dx = \int_0^L \lfloor \mathbf{N} \rfloor^{\mathbf{T}} b\, dx + \lfloor \mathbf{N}(L) \rfloor^{\mathbf{T}} P - AE\int_0^L \lfloor \mathbf{B} \rfloor^{\mathbf{T}} B_0 u_0 dx \tag{6-13}$$

With respect to the integration, $\{\mathbf{u}\}$ is constant, so:

$$AE\int_0^L \lfloor \mathbf{B} \rfloor^{\mathbf{T}} \lfloor \mathbf{B} \rfloor dx \{\mathbf{u}\} = \int_0^L \lfloor \mathbf{N} \rfloor^{\mathbf{T}} b\, dx + \lfloor \mathbf{N}(L) \rfloor^{\mathbf{T}} P - AE\int_0^L \lfloor \mathbf{B} \rfloor^{\mathbf{T}} B_0 u_0 dx \tag{6-14}$$

The reader should note that we can still view these shape functions as global, so this gives us a global set of equations. If we write it in matrix form, this will be:

$$[\mathbf{K}]\{\mathbf{u}\} = \{\mathbf{f}\} \tag{6-15}$$

Here $[\mathbf{K}]$ is called the stiffness matrix, $\{\mathbf{u}\}$ is the unknown displacement vector, and $\{\mathbf{f}\}$ is the load vector.

6.3 The Global Matrix Equations

In Lecture 5 we showed that, using shape functions that are only locally non-zero, we could perform the integration over individual elements and then sum the contribution of each element to obtain the same global matrix as given using global integration. If we use this approach, 6-14 becomes:

$$\sum_{e=1}^{numel} AE\int_0^{L_e} \lfloor \mathbf{B} \rfloor^{\mathbf{T}} \lfloor \mathbf{B} \rfloor dx \{\mathbf{u}\} = \sum_{e=1}^{numel} \int_0^{L_e} \lfloor \mathbf{N} \rfloor^{\mathbf{T}} b\, dx + \sum_{e=1}^{numel_f} \lfloor \mathbf{N}(L) \rfloor^{\mathbf{T}} P - \sum_{e=1}^{numel_u} AE\int_0^{L} \lfloor \mathbf{B} \rfloor^{\mathbf{T}} B_0 u_0 dx \tag{6-16}$$

Note that the boundary terms are only included for the elements with boundary conditions. In addition, there is an implied mapping from the local element numbering system to the global degrees of freedom. The summation signs therefore also indicate assembling.

Question 1: What are the advantages of using the matrix form?

6.4 The Element Stiffness Matrix

Let us now look at an individual element stiffness matrix:

$$[\mathbf{K}^{\mathbf{e}}] = AE\int_0^{L_e} \lfloor \mathbf{B} \rfloor^{\mathbf{T}} \lfloor \mathbf{B} \rfloor dx \tag{6-17}$$

In Lecture 5 we derived the 1-D linear element shape functions:

$$\lfloor \mathbf{N} \rfloor = \left\lfloor \left(\frac{x_J - x}{x_J - x_I} \right) \quad \left(\frac{x - x_I}{x_J - x_I} \right) \right\rfloor \tag{6-18}$$

Taking the derivatives:

$$\lfloor \mathbf{B} \rfloor = \left\lfloor \left(\frac{-1}{L}\right) \quad \left(\frac{1}{L}\right) \right\rfloor \tag{6-19}$$

where $L = x_J - x_I$, the length of the element.

Now, substitute 6-19 into 6-17, to obtain:

$$\left[\mathbf{K^e}\right] = AE\int_0^{L_e} \left\lfloor \begin{matrix} \left(\frac{-1}{L}\right) \\ \left(\frac{1}{L}\right) \end{matrix} \right\rfloor \left\lfloor \left(\frac{-1}{L}\right) \quad \left(\frac{1}{L}\right) \right\rfloor dx \tag{6-20}$$

$$\left[\mathbf{K^e}\right] = AE\int_0^{L_e} \begin{bmatrix} \left(\frac{1}{L^2}\right) & \left(\frac{-1}{L^2}\right) \\ \left(\frac{-1}{L^2}\right) & \left(\frac{1}{L^2}\right) \end{bmatrix} dx \tag{6-21}$$

$$\left[\mathbf{K^e}\right] = \frac{AE}{L^2}\int_0^{L_e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} dx \tag{6-22}$$

and finally, integrating:

$$\left[\mathbf{K^e}\right] = \frac{AE}{L^2}\begin{bmatrix} L & -L \\ -L & L \end{bmatrix} \tag{6-23}$$

or:

$$\left[\mathbf{K^e}\right] = \frac{AE}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{6-24}$$

This is our old friend. Go back to Lecture 1, where we developed the stiffness matrix for a spring using intuition. That matrix was the same as 6-24. It all makes sense.

We now look at the right hand side of 6-16.

$$\left\{\mathbf{f^e}\right\} = \int_0^{L_e} \lfloor \mathbf{N} \rfloor^T b\,dx + \lfloor \mathbf{N}(L) \rfloor^T P + AE\int_0^{L} \lfloor \mathbf{B} \rfloor^T B_0 u_0 dx \tag{6-25}$$

We will look at the first term on the right had side:

$$\left\{\mathbf{f}^{\mathbf{e}}\right\}_{\text{BodyLoads}} = \int_0^{L_e} \lfloor \mathbf{N} \rfloor^{\mathbf{T}} b\, dx \tag{6-26}$$

Substituting for the shape functions gives:

$$\left\{\mathbf{f}^{\mathbf{e}}\right\}_{\text{BodyLoads}} = \int_0^{L_e} \begin{Bmatrix} \left(\dfrac{x_J - x}{x_J - x_I}\right) \\ \left(\dfrac{x - x_I}{x_J - x_I}\right) \end{Bmatrix} b\, dx \tag{6-27}$$

or,

$$\left\{\mathbf{f}^{\mathbf{e}}\right\}_{\text{BodyLoads}} = \frac{b}{L} \int_0^{L_e} \begin{Bmatrix} (x_J - x) \\ (x - x_I) \end{Bmatrix} dx \tag{6-28}$$

After integrating, we obtain:

$$\left\{\mathbf{f}^{\mathbf{e}}\right\}_{\text{BodyLoads}} = bL \begin{Bmatrix} \left(\dfrac{1}{2}\right) \\ \left(\dfrac{1}{2}\right) \end{Bmatrix} \tag{6-29}$$

That is, for a constant body force, the equivalent nodal loads each equal to one-half the total load. If the body load was not constant, we could use the shape functions to interpolate the body load and we would obtain different equivalent nodal loads.

Question 2: What is the physical meaning of (6-26)?

The second term of 6-25 is:

$$\left\{\mathbf{f}^{\mathbf{e}}\right\}_{\text{NaturalBC}} = \lfloor \mathbf{N}(L) \rfloor^{\mathbf{T}} P \tag{6-30}$$

This term is only active on the boundaries with natural boundary conditions. In addition, since we are using shape functions that evaluate to 1 at the nodes, 6-30 turns out to be a statement that the loads at the nodes are directly included in the load vector in the appropriate global locations.

The last term of 6-25 is:

$$\left\{\mathbf{f}^{\mathbf{e}}\right\}_{\text{EssentialBC}} = AE \int_0^{L} \lfloor \mathbf{B} \rfloor^{\mathbf{T}} B_0 u_0 dx \tag{6-31}$$

This term is just like the element stiffness matrix (times the displacement vector). In fact, using element shape functions allows us to evaluate the entire element stiffness matrix and then move the terms associated with known displacements to the right hand side. (6-31) will disappear if the essential boundary condition is homogeneous.

We now have completed a discussion of how we can assemble the global stiffness matrix by summing the individual element matrices.

6.5 Hints to Questions

Q1: Faster to write; shows more clearly the functional relation between physical quantities.
Q2: Weighted interpolation of body forces onto the nodes.

6.6 References

Hughes, T. J. R., 1987, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Chapter 2, Prentice-Hall, Inc.

7 Weak Statement of Elasticity

7.1 Purpose

Our purpose is to develop a weak statement of elasticity. We first briefly review elasticity. We then derive the weak statement of elasticity.

7.2 Brief Review of Elasticity

In the following derivations, we assume the stresses on an infinitesimal block are as shown in Figure 7-1:

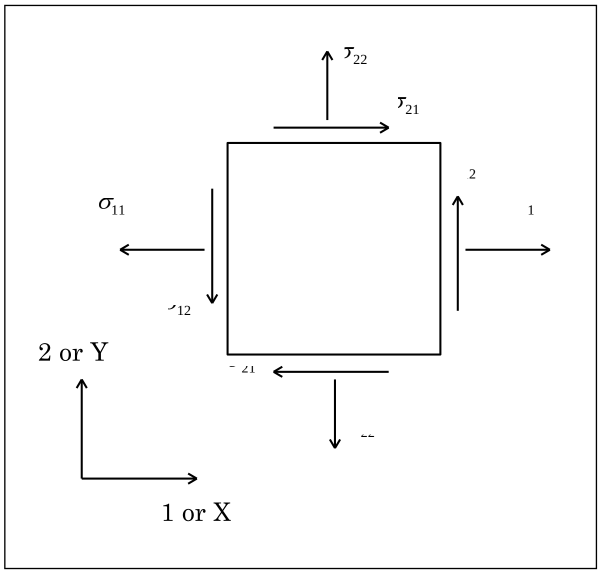


Figure 7-1: Stresses on infinitesimal block

We will equivalently either use indices or x and y to indicate directions. Note that moment equilibrium of the block implies symmetry of the stress tensor. We can write the components of the stress tensor either as a matrix or as a vector:

$$\sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix} = \begin{bmatrix} \sigma_x & \tau_{xy} \\ \tau_{xy} & \sigma_y \end{bmatrix} \tag{7-1}$$

or:

$$\sigma = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{Bmatrix} = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \tag{7-2}$$

Note that the (2,1) element of the matrix in (7-1) is written as σ_{12} rather than σ_{21}. This is because the stress tensor is symmetric and $\sigma_{12} = \sigma_{21}$ as indicated later. Very often, the stress tensor is denoted as σ_{ij}, where i and j may assume the value of 1 or 2 in two dimensional problems, or 1, 2, or 3 in three dimensional problems. Note that *the first index, i, denotes the direction of the outward normal of the plane on which the stress is defined, while the second index, j, refers to the direction the traction is pointing.*

7.2.1 Relation Between Surface Tractions and Stress (First Derivation)

We first establish a relationship between surface tractions and stress. To do this, we look at a wedge of unit thickness:

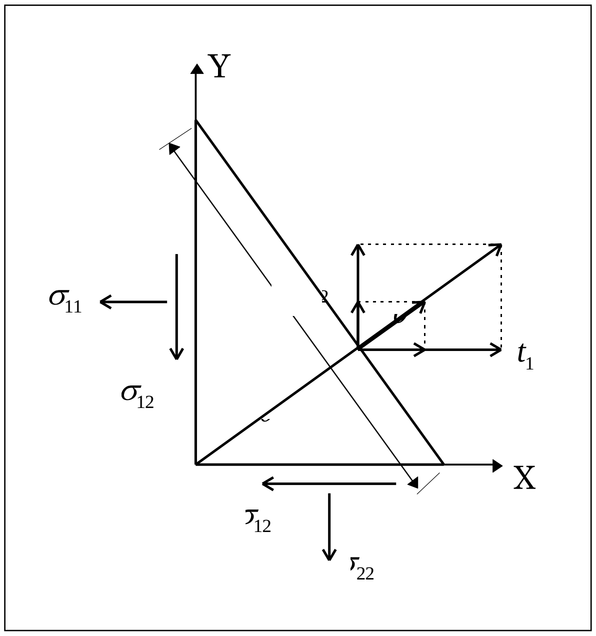


Figure 7-2: Infinitesimal wedge

Equilibrium in the X direction gives:

$$t_1 l - \sigma_{11} l \cos\theta - \sigma_{12} l \sin\theta = 0$$
$$t_1 = \sigma_{11}\cos\theta + \sigma_{12}\sin\theta \qquad (7\text{-}3)$$

and in the Y direction:

$$t_2 l - \sigma_{22} l \sin\theta - \sigma_{12} l \cos\theta = 0$$
$$t_2 = \sigma_{22}\sin\theta + \sigma_{12}\cos\theta \qquad (7\text{-}4)$$

Looking at the normal:

$$n_1 = \cos\theta$$
$$n_2 = \sin\theta \qquad (7\text{-}5)$$

Then 7-3 and 7-4 become:

$$t_1 = \sigma_{11}n_1 + \sigma_{12}n_2$$
$$t_2 = \sigma_{22}n_2 + \sigma_{12}n_1 \qquad (7\text{-}6)$$

or:

$$t_i = \sum_{j=1}^{2}\sigma_{ij}n_j \qquad (7\text{-}7)$$

or $t_i = \sigma_{ij}n_j$ for short. Repeated index indicates summation from 1 to 2 for 2D, or from 1 to 3 for 3D.

7.2.2 Equilibrium

We next look at the stresses on a differential element:

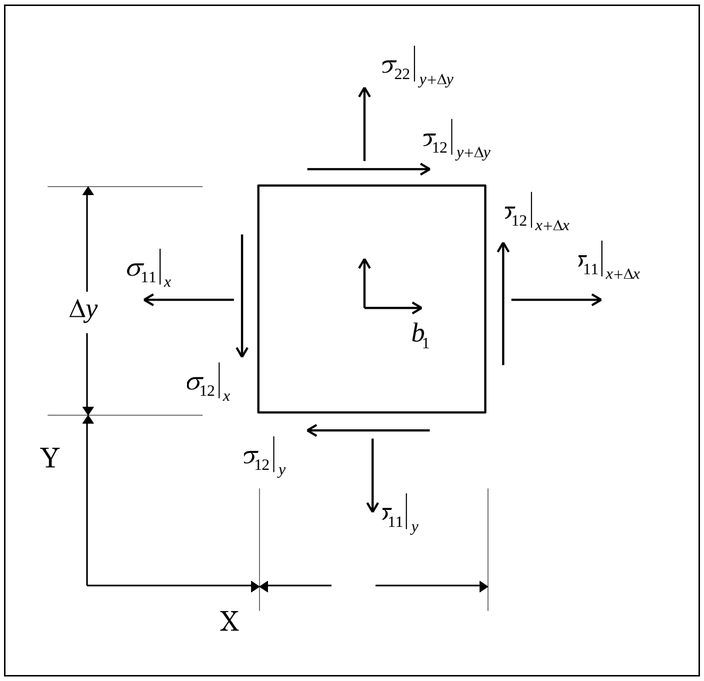

$\sigma_{22}|_{y+\Delta y}$
$\sigma_{12}|_{y+\Delta y}$
$\sigma_{12}|_{x+\Delta x}$
$\sigma_{11}|_{x+\Delta x}$
$\sigma_{11}|_{x}$
Δy
b_1
$\sigma_{12}|_{x}$
Y
$\sigma_{12}|_{y}$
$\sigma_{11}|_{y}$
X

Figure 7-3: Differential element

Here, b_i is a body force per unit volume.

Equilibrium in the X direction gives:

$$\sum F_x = 0$$
$$-\sigma_{11}\big|_x \Delta y + \sigma_{11}\big|_{x+\Delta x}\Delta y - \sigma_{12}\big|_y \Delta x + \sigma_{12}\big|_{y+\Delta y}\Delta x + b_1\Delta x\Delta y = 0 \qquad (7\text{-}8)$$

Rearranging and divide by $\Delta x \Delta y$:

$$\frac{\sigma_{11}\big|_{x+\Delta x} - \sigma_{11}\big|_{x}}{\Delta x} + \frac{\sigma_{12}\big|_{y+\Delta y} - \sigma_{12}\big|_{y}}{\Delta y} + b_1 = 0 \qquad (7\text{-}9)$$

Taking the limit as $\Delta x \to 0, \Delta y \to 0$:

$$\frac{\partial \sigma_{11}}{\partial x_1} + \frac{\partial \sigma_{12}}{\partial x_2} + b_1 = 0 \qquad (7\text{-}10)$$

A similar statement of equilibrium in the Y direction gives:

$$\frac{\partial \sigma_{22}}{\partial x_2} + \frac{\partial \sigma_{12}}{\partial x_1} + b_2 = 0 \qquad (7\text{-}11)$$

7-10 and 7-11 can be combined as:

$$\sigma_{ij,j} + b_i = 0 \text{, i=1, 2} \qquad (7\text{-}12)$$

Equilibrium of the moment on the material element leads to $\sigma_{ij} = \sigma_{ji}$. The stress tensor is therefore symmetric.

7.2.3 Strain-Displacement Relations

As for all these relations we are deriving, there are different approaches to the same end. In this case, we will use a simple derivation assuming small displacements (deformed and original lengths are approximately the same) and small rotations (sinθ=θ and cosθ=1).

The displacements are defined as functions of position:

$$\begin{aligned} u_1 &= u_1(x, y) \\ u_2 &= u_2(x, y) \end{aligned} \qquad (7\text{-}13)$$

or equivalently:

$$\mathbf{u} = \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{Bmatrix} u_x \\ u_y \end{Bmatrix} \qquad (7\text{-}14)$$

Figure 7-4 shows an element that has first undergone a rigid translation, then a stretching deformation, a uniform rotation, and a shear.

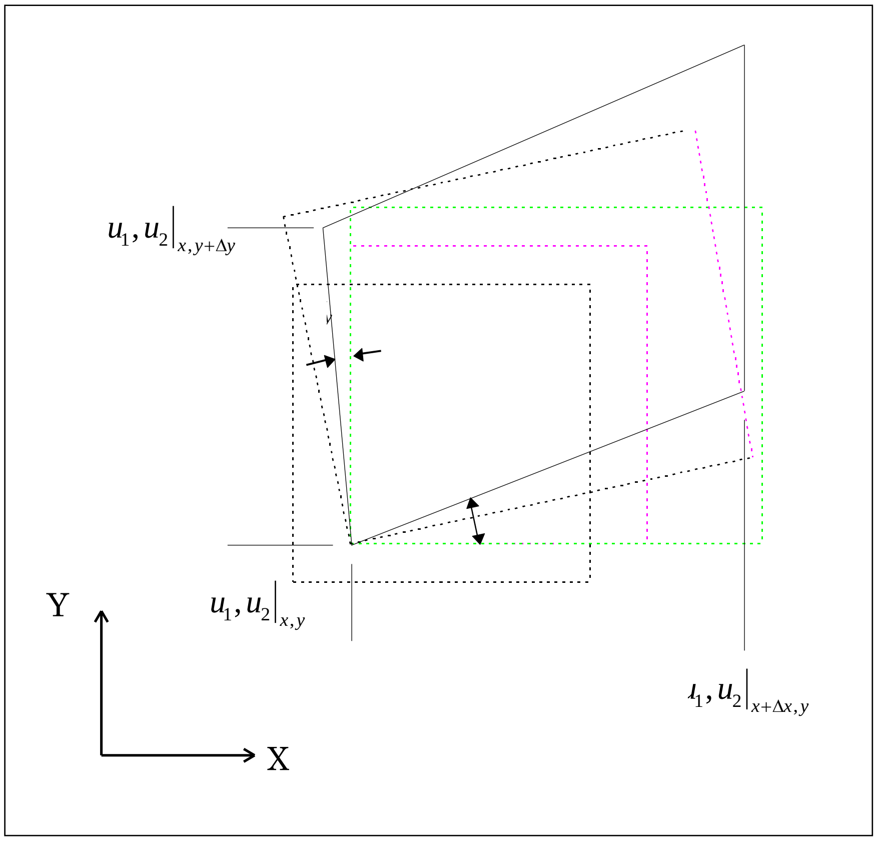


Figure 7-5: Schematic of deformed element

A normal strain is the stretching, that is, change in length divided by length:

$$\varepsilon_1 = \lim_{\Delta x \to 0}\left(\frac{u_1|_{x+\Delta x,y} - u_1|_{x,y}}{\Delta x} \right)$$

$$\varepsilon_1 = \frac{\partial u_1}{\partial x} \tag{7-15}$$

Similarly,

$$\varepsilon_2 = \lim_{\Delta y \to 0}\left(\frac{u_2|_{x,y+\Delta x} - u_2|_{x,y}}{\Delta y} \right)$$

$$\varepsilon_2 = \frac{\partial u_2}{\partial x} \tag{7-16}$$

Note that a rigid body displacement does not induce a strain.

The shear strain is measured by the change in angle of the deformed element. In this case:

$$\theta_x = \lim_{\Delta x \to 0}\left(\frac{u_2|_{x+\Delta x,y} - u_2|_{x,y}}{\Delta x}\right)$$

$$\theta_x = \frac{\partial u_2}{\partial x} \qquad (7\text{-}17)$$

and

$$\theta_y = \lim_{\Delta y \to 0}\left(\frac{u_1|_{x,y+\Delta y} - u_1|_{x,y}}{\Delta y}\right)$$

$$\theta_y = \frac{\partial u_1}{\partial y}$$

The engineering shear strain is measured using the total change in angle:

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \qquad (7\text{-}18)$$

The tensor shear strain is one-half the engineering shear strain:

$$\varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \qquad (7\text{-}19)$$

or

$$\varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \qquad (7\text{-}20)$$

or, for all strains:

$$\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \qquad (7\text{-}21)$$

From the definition of the strain, it is trivial to show that $\varepsilon_{ij} = \varepsilon_{ji}$. The strain tensor is therefore also symmetric.

Question 1: $\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right)$ *represents pure deformation (extension and shear). What does the displacement (deformation) gradient* $u_{i,j}$ *represent?*

7.2.4 Stress-Strain Relations (Plane Strain)

In order to understand properly the stress-strain relations in 2D, it is necessary to start from the 3D relations. The generalized Hooke's law links the stresses to the strains or the strains to the stresses as follows:

Stiffness matrix

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{Bmatrix} = \frac{E}{(1+\upsilon)\ (1-2\nu)} \begin{bmatrix} 1-\upsilon & \upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & 1-\upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & \upsilon & 1-\upsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\upsilon}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\upsilon}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\upsilon}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ 2\varepsilon_{xy} \\ 2\varepsilon_{xz} \\ 2\varepsilon_{yz} \end{Bmatrix} \quad (7\text{-}22)$$

Compliance matrix

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ 2\varepsilon_{xy} \\ 2\varepsilon_{xz} \\ 2\varepsilon_{yz} \end{Bmatrix} = \frac{1}{E} \begin{bmatrix} 1 & -\upsilon & -\upsilon & 0 & 0 & 0 \\ -\upsilon & 1 & -\upsilon & 0 & 0 & 0 \\ -\upsilon & -\upsilon & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2(1+\upsilon) & 0 & 0 \\ 0 & 0 & 0 & 0 & 2(1+\upsilon) & 0 \\ 0 & 0 & 0 & 0 & 0 & 2(1+\upsilon) \end{bmatrix} \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{Bmatrix} \quad (7\text{-}23)$$

Even though we wish to deal with 2-D elastic problems only, the above expressions show clearly that, if the Poisson's ratio of the material is non-zero and the stress-strain state is not trivially zero, it is impossible for both the normal stress and normal strain (σ_{33} and ε_{33}) to be zero simultaneously. It is therefore necessary to distinguish two types of plane problems in elasticity: *plane strain and plane stress*. In plane strain, the thickness dimension is much larger then the in-plane dimensions ("thick plate"), and all out-of-plane strain components (ε_{13}, ε_{23}, and ε_{33}) are zero. For plane strain, although the out-of-plane shear stresses (σ_{13} and σ_{23}) are zero, the normal stress in the thickness direction (σ_{33}) is usually non-zero. In plane stress, the thickness dimension is much smaller then the in-plane dimensions ("thin plate"), and all out-of-plane stress components (σ_{13}, σ_{23}, and σ_{33}) are zero. For plane stress, although the out-of-plane shear strains (ε_{13} and ε_{23}) are zero, the normal strain (ε_{33}) in the thickness direction is usually non-zero.

In this course, we will focus on the plane strain problems.

For plane strain, the Hooke’s law reduces (taking the submatrix from the 3D stiffness matrix) to the following:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} = \frac{E}{(1+\upsilon)\ (1-2\nu)} \begin{bmatrix} (1-\upsilon) & \upsilon & 0 \\ \upsilon & (1-\upsilon) & 0 \\ 0 & 0 & \dfrac{1-2\upsilon}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ 2\varepsilon_{xy} \end{Bmatrix} \qquad (7\text{-}24)$$

If we define the Lame' constants:

$$\lambda = \frac{\nu E}{(1+\upsilon)(1-2\upsilon)}$$

$$G = \frac{E}{2(1+\upsilon)}$$

we can then write:

$$\sigma_{ij} = 2G\varepsilon_{ij} + \lambda\varepsilon_{kk}\delta_{ij}$$

We must invert (7-24), instead of taking the submatrix of the 3D compliance matrix, to obtain the expression for the strains in terms of the stresses:

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ 2\varepsilon_{xy} \end{Bmatrix} = \frac{(1+\upsilon)}{E} \begin{bmatrix} 1-\upsilon & -\upsilon & 0 \\ -\upsilon & 1-\upsilon & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix}$$

Question 2: Why the plane strain compliance matrix is not a submatrix of the 3D case?

7.2.5 Stress-Strain Relations (Plane Stress)

For completeness, the stress-strain relations for plane stress are also given in the following:

Compliance (taking the submatrix of the 3D compliance matrix):

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ 2\varepsilon_{xy} \end{Bmatrix} = \frac{1}{E} \begin{bmatrix} 1 & -\upsilon & 0 \\ -\upsilon & 1 & 0 \\ 0 & 0 & 2(1+\upsilon) \end{bmatrix} \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix}$$

Stiffness (the inverse of the above):

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} = \frac{E}{1-\upsilon^2} \begin{bmatrix} 1 & \upsilon & 0 \\ \upsilon & 1 & 0 \\ 0 & 0 & \frac{1-\upsilon}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ 2\varepsilon_{xy} \end{Bmatrix}$$

Question 3: Why the plane stress stiffness matrix is not a submatrix of the 3D case?

7.3 Elastic Finite Element Formulation

Recall the definition of our problem:

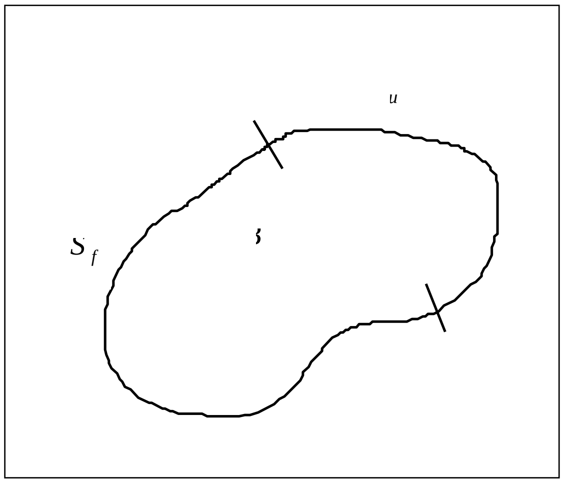


Figure 7-6: Problem definition

We start with equilibrium (the strong form of the differential equation, 7-12):

$$\sigma_{ij,j} + b_i = 0 \tag{7-25}$$

Again, we multiply by a weighting function that is zero at the essential boundary conditions. We will call this weighting function δu to emphasize the equivalence between the weak statement and the variational approach:

$$\left(\sigma_{ij,j} + b_i\right)\delta u_i = 0 \tag{7-26}$$

We then integrate over the volume:

$$\int_V \left(\sigma_{ij,j} + b_i\right)\delta u_i dV = 0 \tag{7-27}$$

or

$$\int_V \sigma_{ij,j}\delta u_i dV + \int_V b_i \delta u_i dV = 0 \tag{7-28}$$

We integrate the left hand term by parts:

$$\int_V \sigma_{ij,j}\delta u_i dV = \int_V \left(\sigma_{ij}\delta u_i\right)_{,j} dV - \int_V \sigma_{ij}\delta u_{i,j} dV \tag{7-29}$$

But we can simplify the term on the right:

$$\frac{\sigma_{ij}\delta u_{i,j}}{2} = \frac{\sigma_{ij}\delta u_{i,j}}{2} \tag{7-30}$$

$$\frac{\sigma_{ij}\delta u_{i,j}}{2} = \frac{\sigma_{ji}\delta u_{i,j}}{2} \qquad \text{(symmetry of stress tensor)} \tag{7-31}$$

$$\frac{\sigma_{ij}\delta u_{i,j}}{2} = \frac{\sigma_{ji}\delta u_{i,j}}{2} = \frac{\sigma_{ij}\delta u_{j,i}}{2} \qquad \text{(switching indices)} \tag{7-32}$$

$$\sigma_{ij}\delta u_{i,j} = \frac{1}{2}\sigma_{ij}\left(\delta u_{i,j} + \delta u_{j,i}\right) \qquad \text{(sum 7-30, 7-32)} \tag{7-33}$$

or, using the strain-displacement definition (7-21):

$$\sigma_{ij} u_{i,j} = \frac{1}{2}\sigma_{ij}\delta\varepsilon_{ij} \tag{7-34}$$

Substituting 7-34 into 7-29 gives:

$$\int_V \sigma_{ij,j}\delta u_i dV = \int_V \left(\sigma_{ij}\delta u_i\right)_{,j} dV - \int_V \sigma_{ij}\delta\varepsilon_{ij} dV \tag{7-35}$$

We now use the divergence theorem on the second term of 7-35 to convert the volume integral to a surface integral:

$$\int_V \sigma_{ij,j}\delta u_i dV = \int_S \sigma_{ij}\delta u_i n_j dS - \int_V \sigma_{ij}\delta\varepsilon_{ij} dV \tag{7-36}$$

We use the relationship between surface traction and stress (7-7) to obtain:

$$\int_V \sigma_{ij,j}\delta u_i dV = \int_S t_i \delta u_i dS - \int_V \sigma_{ij}\delta\varepsilon_{ij} dV \tag{7-37}$$

Finally, we substitute 7-37 into 7-28 to obtain:

$$\int_S t_i \delta u_i dS - \int_V \sigma_{ij}\delta\varepsilon_{ij} dV + \int_V b_i \delta u_i dV = 0 \tag{7-38}$$

or

$$\int_V \sigma_{ij}\delta\varepsilon_{ij}dV = \int_V b_i\delta u_i dV + \int_S t_i\delta u_i dS \qquad (7\text{-}39)$$

This is the weak form of the differential equation. Along with the essential boundary conditions, it describes our problem. Note that the surface integration is only performed over the surface that has *natural boundary conditions*. It is also called the "principal of virtual work." The left hand side corresponds to internal work, while the right hand side corresponds to external and body force work. Also note that the strains and displacements are self-consistent, as are the stresses and loads, but that the stress terms do not need to be consistent with the strain terms.

This forms the basis of our finite element approximation.

Question 4: What is the physical meaning of the left hand side of (7-39)? What is the physical meaning of the right hand side of (7-39)? What is the physical meaning of (7-39)?

7.4 Hints to Questions

Q1: Deformation as well as rigid body rotation.
Q2: Because the normal stress in thickness is non-zero.
Q3: Because the normal strain in thickness is non-zero.
Q4: Internal strain energy; work of external forces; energy conservation.

7.5 References

Hughes, T. J. R., 1987, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Inc.

8 Elasticity Using Finite Elements

8.1 Purpose

Our purpose is to develop a finite element formulation for two dimensional elasticity. We continue from the weak form of the elasticity equation and develop the finite element formulation using element shape functions.

8.2 Finite Element Statement

In Lecture 7 we derived the weak form of the elasticity equation:

$$\int_V \sigma_{ij}\delta\varepsilon_{ij}dV = \int_V b_i\delta u_i dV + \int_S t_i\delta u_i dS \tag{8-1}$$

For convenience, we now introduce matrix notation where:

$$\sigma = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{Bmatrix} \tag{8-2}$$

$$\delta\varepsilon = \begin{Bmatrix} \delta\varepsilon_{11} \\ \delta\varepsilon_{22} \\ \delta\varepsilon_{33} \\ \delta\gamma_{12} \\ \delta\gamma_{23} \\ \delta\gamma_{31} \end{Bmatrix} \tag{8-3}$$

$$\delta\mathbf{u} = \begin{Bmatrix} \delta u_1 \\ \delta u_2 \\ \delta u_3 \end{Bmatrix} \tag{8-4}$$

$$\mathbf{t} = \begin{Bmatrix} t_1 \\ t_2 \\ t_3 \end{Bmatrix} \tag{8-5}$$

The reader should note that the use of engineering shear strain (twice the tensorial shear strain) is deliberate. This is because the full stress and strain tensors have nine components, and

$$\sigma_{ij}\delta\varepsilon_{ij} = \{\sigma_{11} \quad \sigma_{22} \quad \sigma_{33} \quad \sigma_{12} \quad \sigma_{23} \quad \sigma_{31} \quad \sigma_{21} \quad \sigma_{32} \quad \sigma_{13}\}\begin{Bmatrix} \delta\varepsilon_{11} \\ \delta\varepsilon_{22} \\ \delta\varepsilon_{33} \\ \delta\varepsilon_{12} \\ \delta\varepsilon_{23} \\ \delta\varepsilon_{31} \\ \delta\varepsilon_{21} \\ \delta\varepsilon_{32} \\ \delta\varepsilon_{13} \end{Bmatrix}$$

$$= \{\sigma_{11} \quad \sigma_{22} \quad \sigma_{33} \quad \sigma_{12} \quad \sigma_{23} \quad \sigma_{31}\}\begin{Bmatrix} \delta\varepsilon_{11} \\ \delta\varepsilon_{22} \\ \delta\varepsilon_{33} \\ 2\delta\varepsilon_{12} \\ 2\delta\varepsilon_{23} \\ 2\delta\varepsilon_{31} \end{Bmatrix}$$

Substituting into 8-1:

$$\int_V \delta\varepsilon^{T}\sigma dV = \int_V \delta\mathbf{u}^{\mathbf{T}}\mathbf{b}dV + \int_S \delta\mathbf{u}^{\mathbf{T}}\mathbf{t}dS \qquad (8\text{-}6)$$

Question 1: Why is it necessary to use the engineering shear strain, $\delta\gamma_{12} = 2\delta\varepsilon_{12}$, etc., in expression (8-3)?

Questions 2: What is the physical meaning of (8-6)?

We now use shape functions to interpolate the solution. We will use functions that have a unit value at nodes, so we indicate the nodal values by $\boldsymbol{\delta}\mathbf{u}_{\mathbf{a}}$:

$$\boldsymbol{\delta}\mathbf{u} = \mathbf{N}\boldsymbol{\delta}\mathbf{u}_{\mathbf{a}}$$

Note that in 3D, each node has three degrees of freedom (u, v and w), and the expression in the above becomes

$$\begin{Bmatrix} \delta u \\ \delta v \\ \delta w \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & \cdots \\ 0 & N_1 & 0 & 0 & N_2 & 0 & \ldots \\ 0 & 0 & N_1 & 0 & 0 & N_2 & \ldots \end{bmatrix} \begin{Bmatrix} \delta u_1 \\ \delta v_1 \\ \delta w_1 \\ \delta u_2 \\ \delta v_2 \\ \delta w_2 \\ \vdots \end{Bmatrix}$$

where the superscript (*) denotes node number. Note that the shape functions are functions of x, y, and z, but the nodal displacements $\boldsymbol{\delta u_a}$ are not.

We can also write

$$\boldsymbol{\delta u}^{\mathbf{T}} = \boldsymbol{\delta u_a}^{\mathbf{T}} \mathbf{N}^{\mathbf{T}} \tag{8-7}$$

And similarly:

$$\boldsymbol{\delta \varepsilon} = \mathbf{B} \boldsymbol{\delta u_a}$$

$$\boldsymbol{\delta \varepsilon}^{\mathbf{T}} = \boldsymbol{\delta u_a}^{\mathbf{T}} \mathbf{B}^{\mathbf{T}} \tag{8-8}$$

Note here the formulation is in 3D, and the shape function should also be understood as function of x, y, and z, i.e. 3D shape function. Later, we will specialize the formulas to 2D problems for simplicity in programming. We will elaborate on 2D shape functions in later lectures. Substituting 8-7 and 8-8 into 8-6 gives:

$$\int_V \boldsymbol{\delta u_a}^{\mathbf{T}} \mathbf{B}^{\mathbf{T}} \boldsymbol{\sigma} dV - \int_V \boldsymbol{\delta u_a}^{\mathbf{T}} \mathbf{N}^{\mathbf{T}} \mathbf{b} dV - \int_S \boldsymbol{\delta u_a}^{\mathbf{T}} \mathbf{N}^{\mathbf{T}} \mathbf{t} dS = 0 \tag{8-9}$$

Since the nodal displacements associated with the weighting function are constant with respect to the integration:

$$\boldsymbol{\delta u_a}^{\mathbf{T}} \left(\int_V \mathbf{B}^{\mathbf{T}} \boldsymbol{\sigma} dV - \int_V \mathbf{N}^{\mathbf{T}} \mathbf{b} dV - \int_S \mathbf{N}^{\mathbf{T}} \mathbf{t} dS \right) = 0 \tag{8-10}$$

And since this must hold true for all values of the weighting function constants:

$$\int_V \mathbf{B}^{\mathbf{T}} \boldsymbol{\sigma} dV = \int_V \mathbf{N}^{\mathbf{T}} \mathbf{b} dV + \int_S \mathbf{N}^{\mathbf{T}} \mathbf{t} dS \tag{8-11}$$

By recognizing we can do the integration over each element and sum the integrals, we obtain:

$$\sum_{e=1}^{numel}\int_{V_e}\mathbf{B}^{\mathbf{T}}\boldsymbol{\sigma}dV = \sum_{e=1}^{numel}\int_{V_e}\mathbf{N}^{\mathbf{T}}\mathbf{b}dV + \sum_{e=1}^{numel}\int_{S_{et}}\mathbf{N}^{\mathbf{T}}\mathbf{t}dS \qquad (8\text{-}12)$$

Although not our final endpoint, 8-12 is an important finite element statement. It is used in "explicit" dynamic finite element codes. Such codes are used for problems that involve large displacements and large strains, such as automobile crash simulation or weapons simulations. The primary development of explicit codes occurred at the National Laboratories by people including Sam Key (HONDO, Sandia) and Hallquist (DYNA2D/3D, Livermore). These codes have now been commercialized (Livermore Associates, Hallquist) and Abaqus Explicit (Taylor and Flannegan).

Our path is not quite finished. We will now assume an elastic problem with small strains. The displacements are interpolated using shape functions:

$$\mathbf{u} = \mathbf{N_0 u_0} + \mathbf{N u_a} \qquad (8\text{-}13)$$

Note that these are now the real nodal displacements, not a weighting function (variation). As before, we have included shape functions, $\mathbf{N_0}$, that satisfy the essential boundary conditions. Taking the appropriate derivatives of the shape functions we can write:

$$\mathbf{e} = \mathbf{B_0 u_0} + \mathbf{B u_a} \qquad (8\text{-}14)$$

and:

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon} \qquad (8\text{-}15)$$

Where the $\mathbf{D}$ matrix defines the elastic relation between strain and stress. Using 8-14:

$$\boldsymbol{\sigma} = \mathbf{D}\left[\mathbf{B_0 u_0} + \mathbf{B u_a}\right] \qquad (8\text{-}16)$$

Substituting 8-16 into 8-12 gives:

$$\sum_{\mathrm{e=1}}^{\mathrm{numel}}\int_{\mathrm{V_e}}\mathbf{B}^{\mathbf{T}}\mathbf{D}\left[\mathbf{B_0 u_0} + \mathbf{B u_a}\right]\mathrm{dV} = \sum_{\mathrm{e=1}}^{\mathrm{numel}}\int_{\mathrm{V_e}}\mathbf{N}^{\mathbf{T}}\mathbf{b}\mathrm{dV} + \sum_{\mathrm{e=1}}^{\mathrm{numel}}\int_{\mathrm{S_{et}}}\mathbf{N}^{\mathbf{T}}\mathbf{t}\mathrm{dS} \qquad (8\text{-}17)$$

and rearranging:

$$\sum_{e=1}^{numel}\int_{V_e}\mathbf{B}^{\mathbf{T}}\mathbf{D}\mathbf{B}dV\mathbf{u_a} = \sum_{e=1}^{numel}\int_{V_e}\mathbf{N}^{\mathbf{T}}\mathbf{b}dV + \sum_{e=1}^{numel}\int_{S_{et}}\mathbf{N}^{\mathbf{T}}\mathbf{t}dS - \sum_{e=1}^{numel}\int_{S_{et}}\mathbf{B}^{\mathbf{T}}\mathbf{D}\mathbf{B_0}dV\mathbf{u_0} \qquad (8\text{-}18)$$

This is our final finite element statement. In the next lecture, we define the shape functions for our elements.

Question 3: Is the stiffness matrix $\mathbf{D}$ *the same as (7-22) in lecture 7?*

8.3 Hints to Questions

Q1: In expression $\sigma_{ij}\delta\varepsilon_{ij}$, shear terms will appear twice. Using $\delta\gamma_{12} = 2\delta\varepsilon_{12}$ in expression (8-3) takes care of it.
Q2: Internal virtual energy = external virtual work.
Q3: No, because $\delta\gamma_{12} = 2\delta\varepsilon_{12}$ is used in (8-3). $\mathbf{D}$ should be

$$\mathbf{D} = \frac{E}{(1+\upsilon)(1-2\nu)}\begin{bmatrix} 1-\upsilon & \upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & 1-\upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & \upsilon & 1-\upsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\upsilon}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\upsilon}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\upsilon}{2} \end{bmatrix}$$

8.4 References

Hughes, T. J. R., 1987, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Chapter 3, Prentice-Hall, Inc.

9 Natural Coordinates and Numerical Integration

9.1 Purpose

Our purpose is to show how we can integrate the element stiffness matrices numerically.

In general, we will not integrate element stiffness matrices directly because, in most cases, there is no simple closed form integral. Since numerical integration formulae are given in natural coordinates, we need to map our real coordinates to the natural coordinates. We will then demonstrate integration in the natural coordinate system. We derive the mapping first, then implement numerical integration. Integration for both 1D and 2D will be discussed.

9.2 Change of Variables

From calculus, the change of variables formula is:

$$\int_a^b f(x)dx = \int_{\zeta_1}^{\zeta_2} f(g(\zeta))\left[\frac{d(g(\zeta))}{d\zeta}\right]d\zeta \tag{9-1}$$

where:

$$x = g(\zeta)$$

We want to integrate the element stiffness matrix. The relation between the stiffness matrix in real coordinates and the natural coordinates is given by:

$$\left[\mathbf{K^e}\right] = AE\int_0^{L_e}\lfloor\mathbf{B}\rfloor^T\lfloor\mathbf{B}\rfloor dx = AE\int_{-1}^{1}\lfloor\mathbf{B}(\zeta)\rfloor^T\lfloor\mathbf{B}(\zeta)\rfloor\left(\frac{dx}{d\zeta}\right)d\zeta \tag{9-2}$$

We have already derived the shape functions and their derivatives in terms of the natural coordinate, but we still need to define the relationship between x and ζ. We do this using the same shape functions we used for interpolating the displacement. For a linear (two-noded) one-dimensional element:

$$x = N_I x_I + N_J x_J \tag{9-3}$$

Then:

$$\frac{dx}{d\zeta} = \frac{d}{d\zeta}\left(\frac{1}{2}(1-\zeta)\right)x_I + \frac{d}{d\zeta}\left(\frac{1}{2}(1+\zeta)\right)x_J \tag{9-4}$$

Simplifying:

$$\frac{dx}{d\zeta} = \left(\frac{1}{2}(-1)\right)x_I + \left(\frac{1}{2}(1)\right)x_J = \frac{L^e}{2} \tag{9-5}$$

To illustrate, let us integrate the element stiffness matrix of 6-17. To do this, we first need to obtain an expression for the derivative (with respect to x) of the shape functions expressed in natural coordinates $\frac{dN_I(\zeta)}{dx}$. For example, we will illustrate this using the first shape function:

$$\frac{dN_I(\zeta)}{d\zeta} = \frac{dN_I(\zeta)}{dx}\frac{dx}{d\zeta} \tag{9-6}$$

We can evaluate the term on the left hand side and the second term on the right hand side of 9-6:

$$\left(\frac{1}{2}(-1)\right) = \frac{dN_I(\zeta)}{dx}\frac{L^e}{2} \tag{9-7}$$

or:

$$\frac{dN_I(\zeta)}{dx} = \left[\frac{L^e}{2}\right]^{-1}\left(\frac{-1}{2}\right) = \left(\frac{-1}{L^e}\right) \tag{9-8}$$

We will look at the first term in the matrix:

$$AE\int_{-1}^{1} B_I(\zeta)B_I(\zeta)\left(\frac{dx}{d\zeta}\right)d\zeta = AE\int_{-1}^{1}\left(\frac{-1}{L^e}\right)\left(\frac{-1}{L^e}\right)\frac{L^e}{2}d\zeta = \frac{AE}{L^e} \tag{9-9}$$

This is the same value we obtained in Lecture 6. Although a simple example, the same concept applies to more complicated integrals.

Questiuon 1: What are the possible advantages of changing variables in calculating an integral?

9.3 Gauss-Legendre Integration

Numerical integration involves calculating a function at selected locations, multiplying the value by a weight, and then summing to obtain the integral, i.e.

$$\int_{-1}^{1} f(\zeta)d\zeta \approx \sum_{i=1}^{numgp} w_i f(\zeta_i) \tag{9-10}$$

Let us illustrate. Figure 9-1 shows a linear function over the interval -1 to 1.

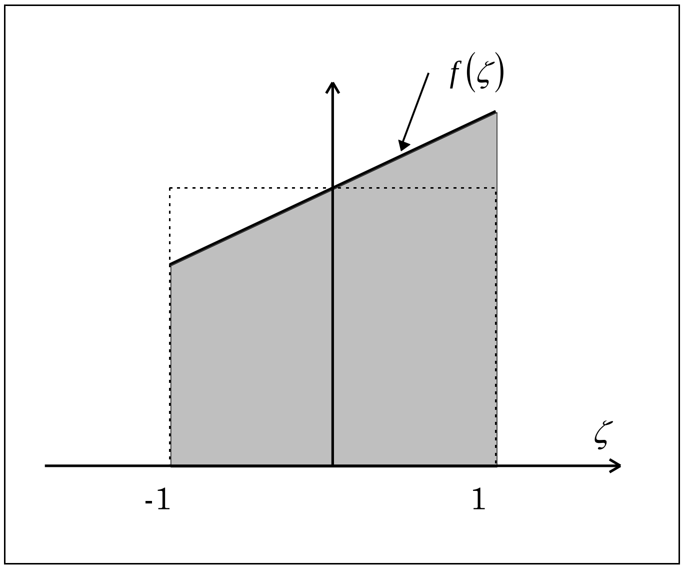


Figure 9-1: Integration of linear function

Clearly, the linear function can be evaluated exactly using one point:

$$\int_{-1}^{1} f(\zeta)d\zeta = \sum_{i=1}^{1} w_i f(\zeta_i) = 2f(0) \tag{9-11}$$

That is, if we evaluate the function at $\zeta = 0$ and multiply that value by 2, we will obtain the exact integral.

If the function is of higher order than linear, the one-point numerical integration is only approximate. We can increase the accuracy by selecting more integration points. Many numerical integration schemes use evenly spaced integration locations such as the Simpson's formula and the Newton-Cotes formula. The Gauss-Legendre integration, however, evaluates the integral at cleverly picked locations and weights leading to high accuracy at low cost. This is illustrated in Figure 9-2.

We will derive the equations for Guass-Legendre integration for a cubic polynomial. To do this, we let both the weight and the locations at which we will evaluate the function be variable. We will calculate the optimum points for 2 Gauss points:

$$\int_{-1}^{1} f(\zeta)d\zeta = \sum_{i=1}^{2} w_i f(\zeta_i) = w_1 f(\zeta_1) + w_2 f(\zeta_2) \tag{9-12}$$

We require that the above expression be exact for a polynomial up to degree 3 and use this requirement to determine the values of the weights (w_1 and w_1) and the locations (ζ_1 and ζ_2). Taking $f(\zeta)$ to be 1, ζ , ζ^2 , and ζ^3 in turn leads to:

$$\int_{-1}^{1} 1 d\zeta = 2 = w_1 + w_2 \tag{9-13}$$

$$\int_{-1}^{1} \zeta d\zeta = 0 = w_1\zeta_1 + w_2\zeta_2 \tag{9-14}$$

$$\int_{-1}^{1} \zeta^2 d\zeta = \frac{2}{3} = w_1\zeta_1{}^2 + w_2\zeta_2{}^2 \tag{9-15}$$

$$\int_{-1}^{1} \zeta^3 d\zeta = 0 = w_1\zeta_1{}^3 + w_2\zeta_2{}^3 \tag{9-16}$$

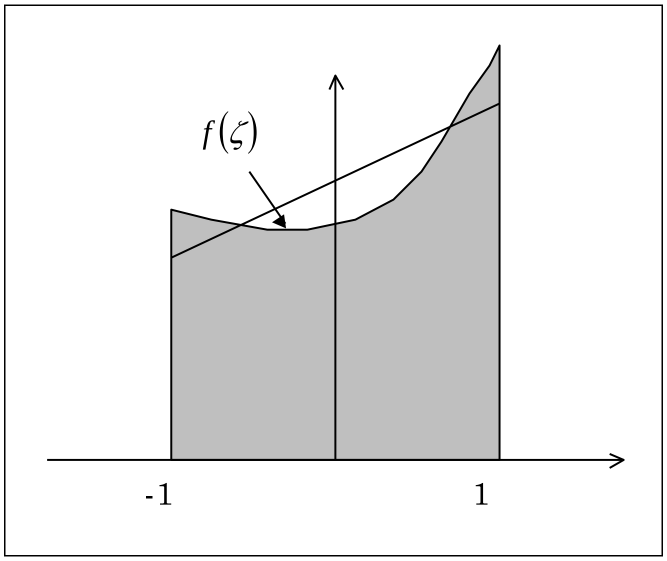


Figure 9-2: Sketch showing integration of higher order curve

We can now solve for the four unknowns. First we use 9-14 and 9-16. Arranging in matrix form:

$$\begin{bmatrix} \zeta_1 & \zeta_2 \\ \zeta_1{}^3 & \zeta_2{}^3 \end{bmatrix} \begin{Bmatrix} w_1 \\ w_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \tag{9-17}$$

For this to be true, for non-zero weights, the determinant must equal zero:

$$\zeta_1\zeta_2{}^3 - \zeta_1{}^3\zeta_2 = 0 \tag{9-18}$$

for $\zeta_1 \neq 0$ and $\zeta_2 \neq 0$:

$$\zeta_2{}^2 - \zeta_1{}^2{}_2 = 0 \tag{9-19}$$

or:

$$\zeta_2 = \pm\zeta_1 \tag{9-20}$$

For two distinct points:

$$\zeta_2 = -\zeta_1 \tag{9-21}$$

Using this information in 9-14, gives:

$$\mathrm{w}_1 = \mathrm{w}_2 \tag{9-22}$$

Then, using 9-13 gives:

$$\mathrm{w}_1 = \mathrm{w}_2 = 1 \tag{9-23}$$

and using 9-15 gives:

$$\zeta_1 = -\zeta_2 = \frac{1}{\sqrt{3}} \tag{9-24}$$

Finally, we get:

$$\int_{-1}^{1} \mathrm{f}(\zeta)\,\mathrm{d}\zeta = 1\mathrm{f}\left(\frac{1}{\sqrt{3}}\right) + 1\mathrm{f}\left(\frac{1}{\sqrt{3}}\right) \tag{9-25}$$

That is, we can obtain the exact integral for a polynomial of up to order $N = 3$ using two Gauss points with:

$$\mathrm{w}_1 = \mathrm{w}_2 = 1$$

and

$$\zeta_1 = \frac{1}{\sqrt{3}},\ \zeta_2 = -\frac{1}{\sqrt{3}}.$$

For polynomials of order greater than three, the numerical integral is only approximate.

Question 1: Why is Guassian integration effecient?

9.4 Example

To illustrate, let us integrate the following function:

$$f(\zeta) = 100 + 50\zeta + 75\zeta^2 \tag{9-26}$$

The exact result is:

$$\int_{-1}^{1} f(\zeta)d\zeta = 250 \tag{9-27}$$

Using our derived Gauss method:

$$\int_{-1}^{1} f(\zeta)d\zeta = (1)f\left(-\frac{1}{\sqrt{3}}\right) + (1)f\left(\frac{1}{\sqrt{3}}\right) \tag{9-28}$$

Evaluating gives:

$$\int_{-1}^{1} f(\zeta)d\zeta \cong (1)\left[100 + 50\left(-\frac{1}{\sqrt{3}}\right) + 75\left(-\frac{1}{\sqrt{3}}\right)\right] + (1)\left[100 + 50\left(\frac{1}{\sqrt{3}}\right) + 75\left(\frac{1}{\sqrt{3}}\right)\right] \tag{9-29}$$

or

$$\int_{-1}^{1} f(\zeta)d\zeta = 250 \tag{9-30}$$

This is the exact integral, as expected, since the order of the function being integrated was two.

As stated, if the function is of order $N > 3$, the numerical integral will be approximate. For example, if we the function is:

$$f(\zeta) = \cos(\zeta) \tag{9-31}$$

The exact integral from -1 to 1 is:

$$\int_{-1}^{1} \cos(\zeta)d\zeta = 1.682942 \tag{9-32}$$

While the approximate integral using two Gauss points is:

$$\int_{-1}^{1} \cos(\zeta)d\zeta = 1.675823 \tag{9-33}$$

Thus, the numerical integral is approximate. Using more Gauss points would lead to a more accurate numerical integral.

Question 2: For one-dimensional, 3-noded quadratic elements, to obtain accurate element stiffness matrix, how many integration points are needed?

9.5 Higher Order Gauss-Legendre Integration

Table for higher order integration using more Gauss points are given in many books. Gives values for up to three Gauss points. A polynomial of order $N \le 2n-1$, where n is the number of Gauss points, will be integrated exactly.

n	ζ_i	w_i
1	0.0	2.0
2	±0.577350269189626 (or $\left(\sqrt{3}\right)^{-1}$)	1.0
3	±0.774596669241483 (or $\sqrt{0.6}$)	0.555555555555555 (or 5/9)
	0.0	0.888888888888889 (or 8/9)

Table 9-1: Higher order Gauss-Legendre integration

9.6 Application of Numerical Integration to Evaluation of Element Matrices

We will now demonstrate how numerical integration can be used to calculate an element stiffness matrix. We previously changed variables for the element stiffness matrix in 6-17:

$$\left[\mathbf{K}^{\mathbf{e}}\right] = AE\int_0^{L_e}\lfloor\mathbf{B}\rfloor^{\mathbf{T}}\lfloor\mathbf{B}\rfloor dx = AE\int_{-1}^{1}\lfloor\mathbf{B}(\zeta)\rfloor^{\mathbf{T}}\lfloor\mathbf{B}(\zeta)\rfloor\left(\frac{dx}{d\zeta}\right)d\zeta \tag{9-34}$$

We have also shown how we can implement a numerical integration scheme where we evaluate the function at a given number of points and multiply the values by a weight and then sum the results:

$$\int_{-1}^{1} f(\zeta)\,d\zeta = \sum_{i=1}^{n} w_i f\left(\zeta_i\right) \tag{9-35}$$

Substituting 9-34 into 9-35 gives:

$$\left[\mathbf{K}^{\mathbf{e}}\right] \cong \sum_{i=1}^{n} w_i\left[AE\lfloor\mathbf{B}(\zeta_i)\rfloor^{\mathbf{T}}\lfloor\mathbf{B}(\zeta_i)\rfloor\left(\frac{dx(\zeta_i)}{d\zeta}\right)\right] \tag{9-36}$$

This maybe looks a bit complicated, but all it means is that we will evaluate the term in square brackets at however many Gauss points we are using, multiply by the weights, and then summing the result.

Recall, for a linear element:

$$\lfloor \mathbf{B}(\zeta) \rfloor = \left\lfloor \left(\frac{dN_I(\zeta)}{dx} \right) \quad \left(\frac{dN_J(\zeta)}{dx} \right) \right\rfloor \tag{9-37}$$

and we have already shown (9-8) that:

$$\frac{dN_I(\zeta)}{dx} = \left(\frac{-1}{L^e} \right) \tag{9-38}$$

Similarly:

$$\frac{dN_J(\zeta)}{dx} = \left(\frac{1}{L^e} \right) \tag{9-39}$$

Also, in 9-5 we derived that (for a linear element):

$$\frac{dx}{d\zeta} = \frac{L^e}{2} \tag{9-40}$$

Substituting 9-38, 9-39, and 9-40 into 9-37, we obtain:

$$\left[\mathbf{K}^{\mathbf{e}}\right] \cong \sum_{i=1}^{n} w_i \left[AE \begin{Bmatrix} \left(\frac{-1}{L^e} \right) \\ \left(\frac{1}{L^e} \right) \end{Bmatrix} \left\lfloor \left(\frac{-1}{L^e} \right) \quad \left(\frac{1}{L^e} \right) \right\rfloor \left(\frac{L^e}{2} \right) \right] \tag{9-41}$$

or:

$$\left[\mathbf{K}^{\mathbf{e}}\right] \cong \sum_{i=1}^{n} w_i \left[\frac{AE}{L^e} \begin{bmatrix} \left(\frac{1}{2} \right) & \left(-\frac{1}{2} \right) \\ \left(-\frac{1}{2} \right) & \left(\frac{1}{2} \right) \end{bmatrix} \right] \tag{9-42}$$

Obviously, this is a simple integral. The function we are integrating is a constant, so we only need to use one Gauss point to integrate it exactly. In that case, $n = 1$, $w_i = 2$, and $\zeta_i = 0$ (not used for constant function). Implementing the Gauss-Legendre integration, gives:

$$\left[\mathbf{K^e}\right] \cong 2\left[\frac{\mathrm{AE}}{\mathrm{L^e}}\begin{bmatrix}\left(\frac{1}{2}\right) & \left(-\frac{1}{2}\right)\\ \left(-\frac{1}{2}\right) & \left(\frac{1}{2}\right)\end{bmatrix}\right] \tag{9-43}$$

and we finally obtain our old friend:

$$\left[\mathbf{K^e}\right] \cong \frac{\mathrm{AE}}{\mathrm{L^e}}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix} \tag{9-44}$$

In this case, the integration is exact, but in general, the numerical integration will be approximate.

9.7 Numerical Integration in 2D

In Lecture 8 we derived our finite element statement:

$$\sum_{e=1}^{numel}\int_{V_e}\mathbf{B}^{\mathbf{T}}\mathbf{D}\mathbf{B}dV\mathbf{u_a} = \sum_{e=1}^{numel}\int_{V_e}\mathbf{N}^{\mathbf{T}}\mathbf{b}dV + \sum_{e=1}^{numel}\int_{S_{et}}\mathbf{N}^{\mathbf{T}}\mathbf{t}dS - \sum_{e=1}^{numel}\int_{S_{et}}\mathbf{B}^{\mathbf{T}}\mathbf{D}\mathbf{B_0}dV\mathbf{u_0} \tag{9-45}$$

and in later lectures we will present the details of the $\mathbf{B}$, $\mathbf{D}$, and $\mathbf{N}$ matrices. The integration for each element will be performed numerically. As for one-dimensional numerical integration, we first transform from real coordinates to natural coordinates (note that in two dimensions, integration over the volume corresponds performing area integration multiplied by the thickness). In two dimensions, the change of variables formula is:

$$\int_{y_1}^{y_2}\int_{x_1}^{x_2} f(x,y)\,dxdy = \int_{\eta_1}^{\eta_2}\int_{\zeta_1}^{\zeta_2} f\left(g_1(\zeta,\eta), g_2(\zeta,\eta)\right)\left|\mathbf{J}\right| d\zeta d\eta \tag{9-46}$$

where:

$$x = g_1(\zeta,\eta)$$
$$y = g_2(\zeta,\eta)$$
$$\left|\mathbf{J}\right| = \begin{vmatrix}\frac{\partial x}{\partial\zeta} & \frac{\partial y}{\partial\zeta}\\ \frac{\partial x}{\partial\eta} & \frac{\partial y}{\partial\eta}\end{vmatrix} = \begin{vmatrix}\frac{\partial g_1}{\partial\zeta} & \frac{\partial g_2}{\partial\zeta}\\ \frac{\partial g_1}{\partial\eta} & \frac{\partial g_2}{\partial\eta}\end{vmatrix} = \text{Determinant of Jacobian matrix.}$$

In our case, the integration of the element stiffness matrix is transformed as:

$$\int_{V_e} \mathbf{B}^T\mathbf{D}\mathbf{B}dV = t\int_{-1}^{1}\int_{-1}^{1} \mathbf{B}^T\mathbf{D}\mathbf{B}|\mathbf{J}|d\zeta d\eta \tag{9-47}$$

where t is the element thickness and the $\mathbf{B}$, $\mathbf{D}$, and $\mathbf{J}$ matrices are as given in Lecture 10, where they are written in terms of ζ and η.

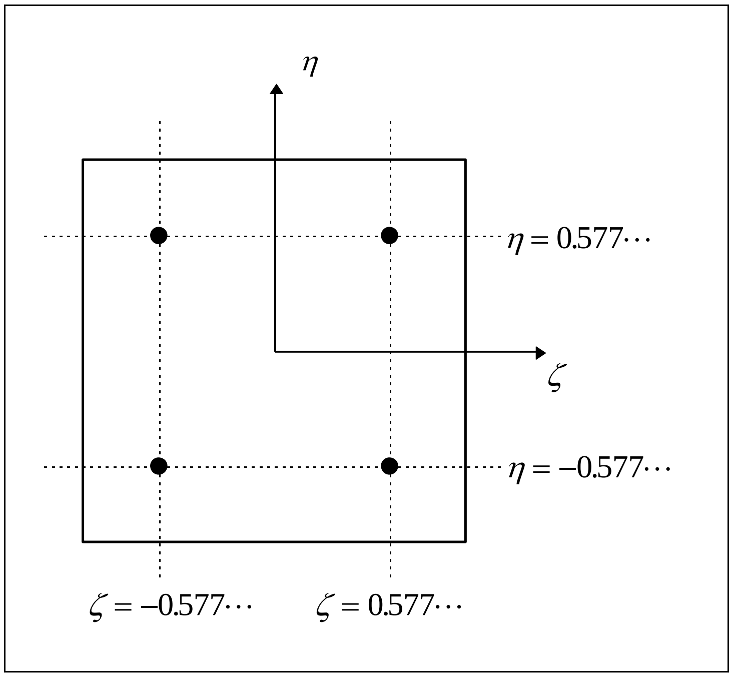


Figure 9-3: Location of 2x2 Gauss points

Similar to the one-dimensional case, in 2D we integrate numerically by evaluating the function at selected points in each dimension, multiplying the value by a weighting function, and summing over all the integration points. In general this is:

$$t\int_{-1}^{1}\int_{-1}^{1} \mathbf{B}^T\mathbf{D}\mathbf{B}|\mathbf{J}|d\zeta d\eta = t\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\mathbf{B}(\zeta_i,\eta_j)^T\mathbf{D}\mathbf{B}(\zeta_i,\eta_j)|\mathbf{J}|(\zeta_i,\eta_j)\right]w_i w_j \tag{9-48}$$

The Gauss points are defined in two dimensions using a product of the one-dimensional locations previously derived.

More precise values of the locations and weights are given below.

Order	**Location**	**Weight**
2x2	± 0.577350269189626 (or $\left(\sqrt{3}\right)^{-1}$)	1.000000000000000
3x3	± 0.774596669241483 (or $\sqrt{0.6}$)	0.555555555555556 (or 5/9)

	0.000000000000000	0.888888888888888 (or 8/9)

Table 2: Gauss points

As before, 2x2 Gauss point integration is precise for cubic polynomials and 3x3 integration is precise for 5th order polynomials. In our work, we will use 3x3 integration. This will be exact integration for parallelepiped elements with the side nodes at the midpoints (the user should verify this).

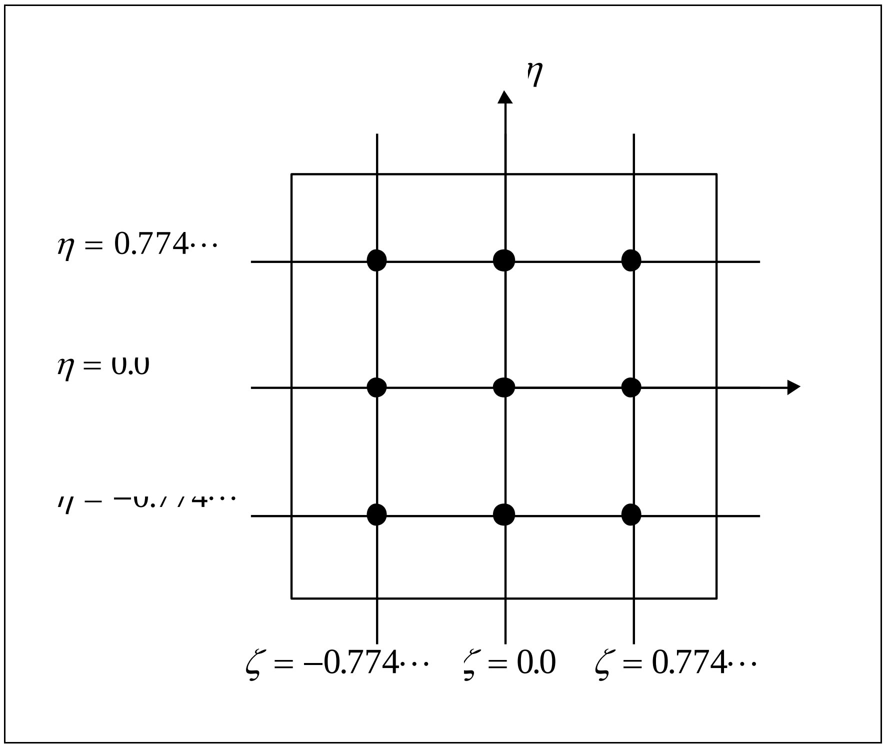


Figure 9-4: Location of 3x3 Gauss points

9.8 Integration of Body Forces and Surface Tractions

See next lecture.

9.9 Implementation

Numerical integration is typically implemented using nested loops for each direction of integration. For instance, 9-48 would be implemented as nested loops corresponding to the ζ and η natural coordinates. Inside the inner most loop, all matrices are evaluated, multiplied, and the sum is accumulated to obtain the integral. This is discussed in more detail in later lectures.

9.10 Hints to Questions

Q1: Because both the locations and weights are optimized.
Q2: 2 integration points.

9.11 References

Hughes, T. J. R., 1987, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Inc.

10 More on Shape Functions and Isoparametric Elements

10.1 Purpose

Our purpose is to briefly describe element shape functions. We will then develop the shape functions for an eight-noded quadrilateral element.

Our goal here is not to describe all element shape functions. Instead, we will focus on the eight-noded element. This element is a common element for elasticity analysis. We will see that it performs well. The reader will be referred to other texts for a more complete discussion of shape functions.

10.2 Finite Element Statement

We have already derived one-dimensional elements with linear and quadratic shape functions (Figure 10-1).

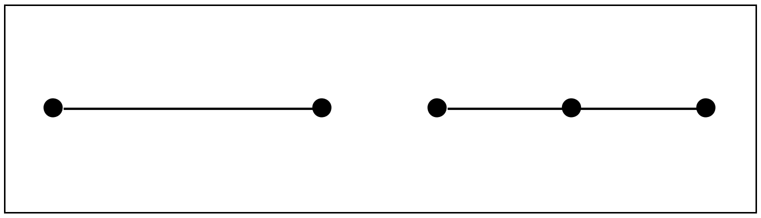


Figure 10-1: One-dimensional elements

Similar elements can be derived in two dimensions. These include triangular elements (Figure 10-2).

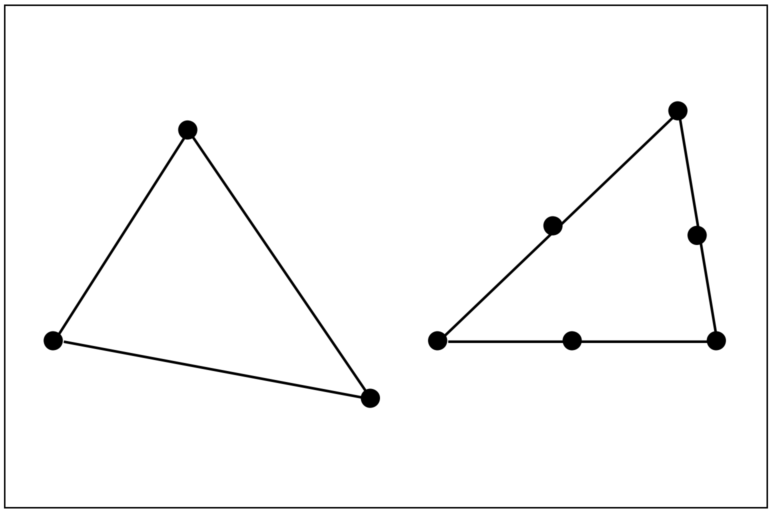


Figure 10-2: Triangular two-dimensional elements

We can also develop two-dimensional linear and quadratic elements (Figure 10-3).

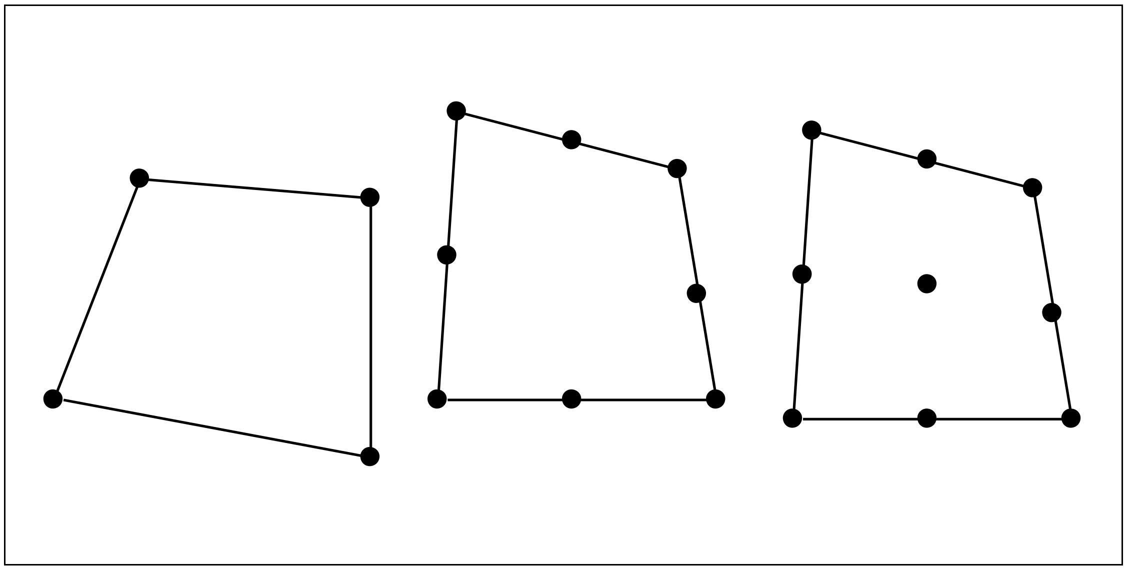


Figure 10-3: Two-dimensional linear and quadratic elements

Finally, the same families of elements can be developed in three-dimensions (Figure 10-4).

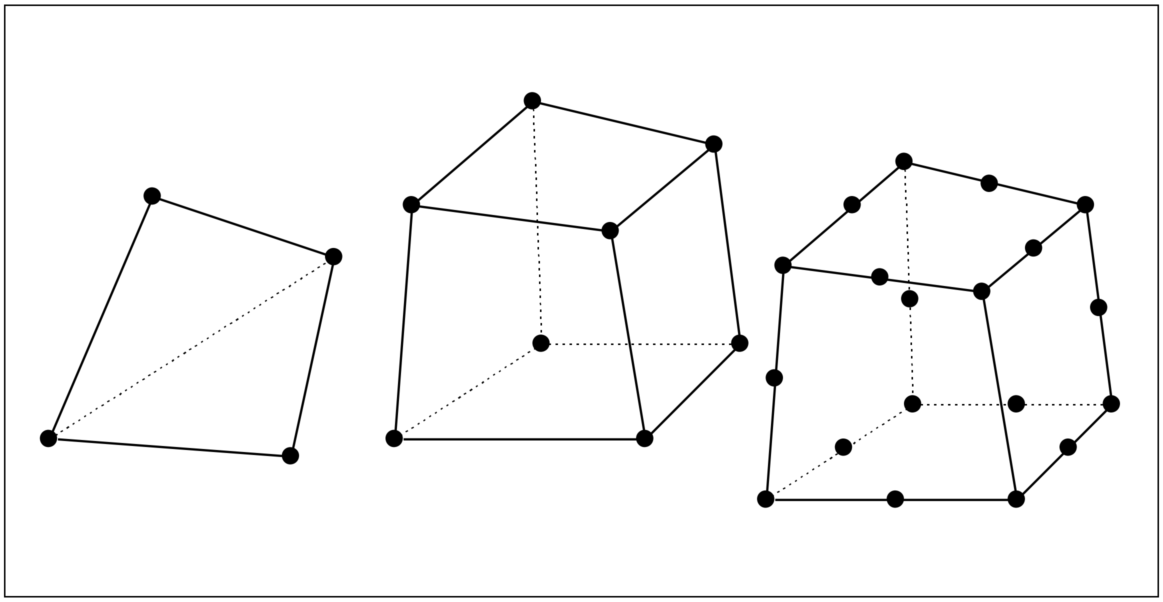


Figure 10-4: Some three-dimensional elements

Most commercial finite element codes will include all the above elements (and others). There is no answer to what element is "best." Large deformation nonlinear analyses typically use simple four-noded (2-D) or eight-noded (3-D) elements. This technology was originally developed at the national laboratories to solve the large deformation problems associated with weapons. Eight-noded (2-D) and twenty-noded (3-D) elements are often used for elasticity. One advantage is that fewer elements are needed for a given accuracy than using the simpler elements. Another advantage is the capability to move the side node to the quarter-points to represent the singularity around a crack tip. The bottom line is that, at the present time, the analyst has the responsibility to ensure that the mesh being used to represent a problem is appropriate. This must be done knowing the type of element being used and intuition into the particular problem being solved. In the future (and in some current codes) automatic mesh refinement will help ensure a desired accuracy of solution.

10.3 The Serendipity Element (8-Noded Quadratic, Q8, Element)

The 2D shape functions $N_i(\zeta,\eta)$ are defined by the following relations:

$$u(\zeta,\eta) = \sum_i N_i(\zeta,\eta) u_i$$
$$v(\zeta,\eta) = \sum_i N_i(\zeta,\eta) v_i$$

where u_i and v_i are the x and y displacements of node i. Similar to the one-dimensional case, the 2D shape functions satisfy the following conditions:

1) $$N_i(\zeta_j,\eta_j) = \delta_{ij}$$
2) $$\sum_i N_i(\zeta,\eta) = 1$$

We will focus on one particular element, the eight-noded quadrilateral (Q8), for two-dimensional problems. This element is a square in natural coordinates (Figure 10-5).

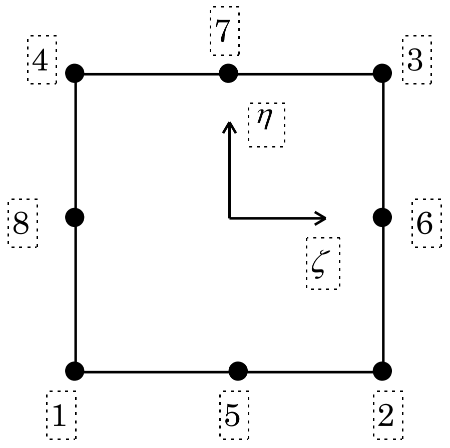


Figure 10-5: Serendipity element (Q8)

The formula for shape function of node i can be generally expressed by

$$N_i = C\prod(3\text{ lines passing other nodes but i})$$

Every node, except node i, must be passed at least once by one or more of the three lines. The constant C is determined using the condition $N_i(\zeta_i, \eta_i) = 1$. For example, for node 1, draw three lines, one passing nodes 5 and 8 ($\zeta + \eta + 1 = 0$), one passing nodes 2, 6 and 3 ($\zeta - 1 = 0$), and one passing nodes 4, 7 and 3 ($\eta - 1 = 0$). Then $N_1 = C(\zeta + \eta + 1)(\zeta - 1)(\eta - 1)$. From $N_1(\zeta = -1, \eta = -1) = 1$, we find that $C = -1/4$, N_1 is therefore determined to be $N_1 = -\frac{1}{4}(\zeta - 1)(\eta - 1)(\zeta + \eta + 1)$.

Question 1: Why do we require the equation of 3 lines in the above formula?

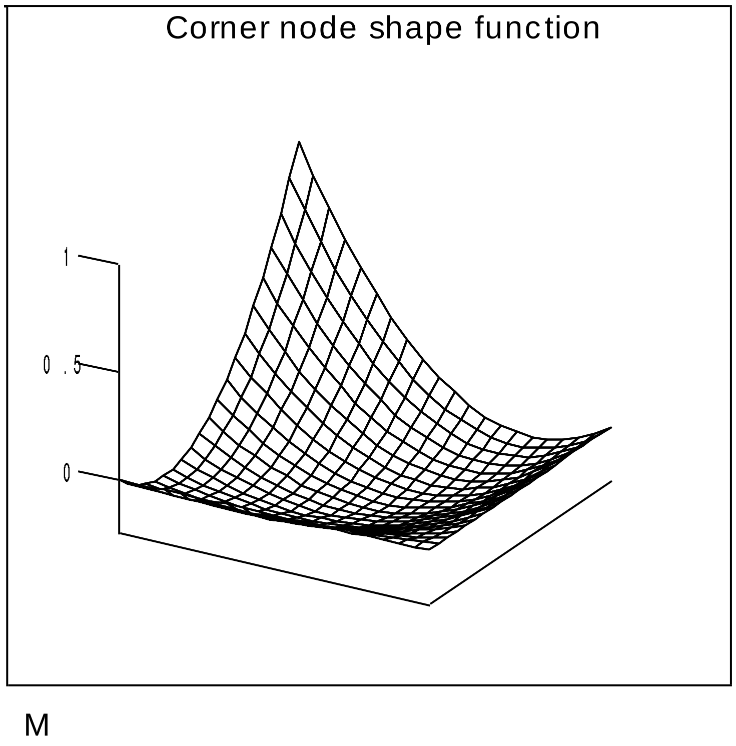


Figure 10-6: Corner node shape function

Following the procedures described above, the shape function for all nodes can be obtained. The shape functions for this element are:

$$N_1 = -\frac{1}{4}(1 - \zeta)(1 - \eta)(\zeta + \eta + 1)$$

$$N_2 = \frac{1}{4}(1 + \zeta)(1 - \eta)(\zeta - \eta - 1)$$

$$N_3 = \frac{1}{4}(1 + \zeta)(1 + \eta)(\zeta + \eta - 1)$$

$$N_4 = -\frac{1}{4}(1-\zeta)(1+\eta)(\zeta-\eta+1)$$

$$N_5 = \frac{1}{2}\left(1-\zeta^2\right)(1-\eta)$$

$$N_6 = \frac{1}{2}\left(1-\eta^2\right)(1+\zeta)$$

$$N_7 = \frac{1}{2}\left(1-\zeta^2\right)(1+\eta)$$

$$N_8 = \frac{1}{2}\left(1-\eta^2\right)(1-\zeta) \qquad (10\text{-}1)$$

The shape functions are plotted in Figure 10-6 and Figure 10-7. Note that the values are 1.0 at the associated node and 0.0 at all other nodes. The user should verify that along an edge ($\zeta = 1.0$, etc.) the shape functions simplify to the previously derived one-dimensional quadratic functions.

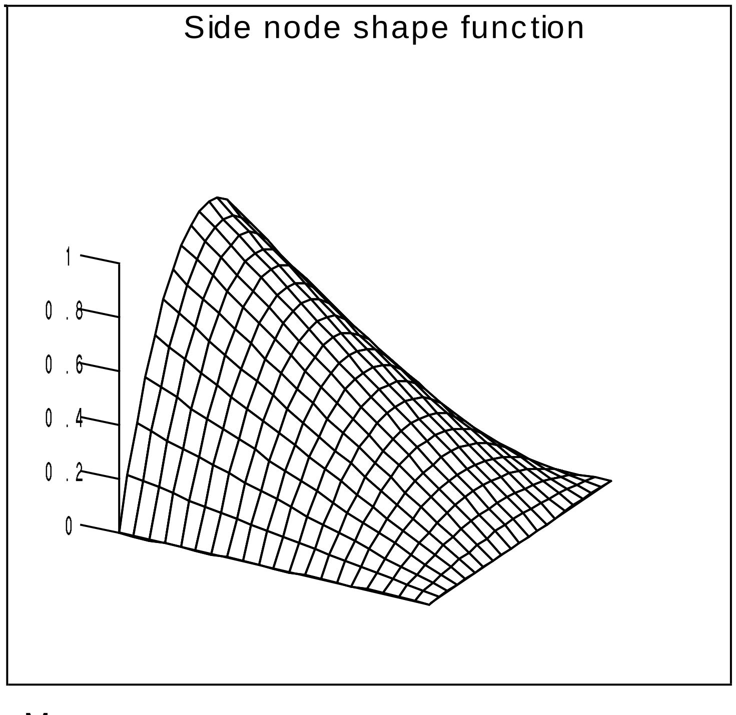


Figure 10-7: Side node shape function

10.4 Details of the Element Matrices

We now can give details of the **N** and **B** matrices defined in the previous lecture. The **N** matrix interpolates the nodal displacements to the interior of the element:

$$\mathbf{u} = \mathbf{N}\mathbf{u}_{\mathbf{a}} \tag{10-2}$$

For two dimensional elasticity, each node has displacements in the X and Y directions. The interpolation then is:

$$\begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & \cdots & N_8 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & \cdots & 0 & N_8 \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ \vdots \\ u_8 \\ v_8 \end{Bmatrix} \tag{10-3}$$

Note that the u(x, y) depends on only the x-component of nodal displacements, while v(x, y) only one the y-component.

The **B** matrix relates strains to displacements. Since we use the shape functions to interpolate displacements:

$$u = N_1u_1 + N_2u_2 + N_3u_3 + N_4u_4 + N_5u_5 + N_6u_6 + N_7u_7 + N_8u_8 \tag{10-4}$$

and then,

$$\varepsilon_x = \frac{\partial u}{\partial x} = \frac{\partial N_1}{\partial x}u_1 + \frac{\partial N_2}{\partial x}u_2 + \frac{\partial N_3}{\partial x}u_3 + \cdots + \frac{\partial N_8}{\partial x}u_8 \,. \tag{10-5}$$

Similarly,

$$\varepsilon_y = \frac{\partial v}{\partial x} = \frac{\partial N_1}{\partial x}v_1 + \frac{\partial N_2}{\partial x}v_2 + \frac{\partial N_3}{\partial x}v_3 + \cdots + \frac{\partial N_8}{\partial x}v_8 \tag{10-6}$$

and,

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = \frac{\partial N_1}{\partial y}u_1 + \frac{\partial N_1}{\partial x}v_1 + \frac{\partial N_2}{\partial y}u_2 + \frac{\partial N_2}{\partial x}v_2 + \cdots + \frac{\partial N_3}{\partial y}u_3 + \frac{\partial N_8}{\partial x}v_8 \tag{10-7}$$

Combined, these define the **B** matrix:

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_x \\ \gamma_{xy} \end{Bmatrix} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \cdots & \frac{\partial N_8}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & \cdots & 0 & \frac{\partial N_8}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \cdots & \frac{\partial N_8}{\partial y} & \frac{\partial N_8}{\partial x} \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_8 \\ v_8 \end{Bmatrix} \tag{10-8}$$

or:

$$\boldsymbol{\varepsilon} = \mathbf{Bu_a} \tag{10-9}$$

So, if we evaluate the $\mathbf{B}$ matrix at any point in the element, we can obtain the strains by multiplying it by the nodal displacements.

We need the derivatives of the shape functions with respect to the global coordinates, but we know the derivatives with respect to the natural coordinates because we express the shape functions in terms of the natural coordinates. To obtain the derivatives with respect to global coordinates, we use the chain rule:

$$\begin{aligned} \frac{\partial N_1}{\partial \zeta} &= \frac{\partial N_1}{\partial x}\frac{\partial x}{\partial \zeta} + \frac{\partial N_1}{\partial y}\frac{\partial y}{\partial \zeta} \\ \frac{\partial N_1}{\partial \eta} &= \frac{\partial N_1}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial N_1}{\partial y}\frac{\partial y}{\partial \eta} \end{aligned} \tag{10-10}$$

or:

$$\begin{Bmatrix} \frac{\partial N_1}{\partial \zeta} \\ \frac{\partial N_1}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \frac{\partial N_1}{\partial x} \\ \frac{\partial N_1}{\partial y} \end{Bmatrix} \tag{10-11}$$

This defines the Jacobian matrix. By using (10-11) , it is implied that the relation between the real and natural coordinates are known. We can choose the use the shape function that interpolate the displacements to interpolate the geometry (positions), that is,

$$x = N_1x_1 + N_2x_2 + N_3x_3 + \cdots + N_8x_8 \tag{10-12}$$

from which

$$\frac{\partial x}{\partial \zeta} = \frac{\partial N_1}{\partial \zeta}x_1 + \frac{\partial N_2}{\partial \zeta}x_2 + \cdots + \frac{\partial N_8}{\partial \zeta}x_8 \tag{10-13}$$

with similar expressions for $\frac{\partial y}{\partial \zeta}$, $\frac{\partial x}{\partial \eta}$, and $\frac{\partial y}{\partial \eta}$. Elements that use the same shape functions to interpolate both the displacements and positions are called *isoparametric elements*. We can then invert the Jacobian matrix to obtain the derivatives with respect to global coordinates:

$$\begin{Bmatrix} \frac{\partial N_1}{\partial x} \\ \frac{\partial N_1}{\partial y} \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \frac{\partial N_1}{\partial \zeta} \\ \frac{\partial N_1}{\partial \eta} \end{Bmatrix} \qquad (10\text{-}14)$$

Question 2: Is it necessary to use isoparametric formulation?

10.5 The Material Matrix

The material matrix, $\mathbf{D}$, relates the stresses to the strains:

$$\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon} \qquad (10\text{-}15)$$

For elasticity, the $\mathbf{D}$ matrix is given by Hooke's law. For plane strain:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & 0 \\ \nu & (1-\nu) & 0 \\ 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \qquad (10\text{-}16)$$

and,

$$\sigma_z = \nu(\sigma_x + \sigma_y) \qquad (10\text{-}17)$$

Similarly, for plane stress:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} = \frac{E}{(1-\nu^2)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \qquad (10\text{-}18)$$

and,

$$\sigma_z = 0 \qquad (10\text{-}19)$$

We have now defined all the matrices that are needed to evaluate the stiffness matrix and load vectors defined in the previous lecture. It is difficult, and frequently impossible, to perform the integration analytically. The Gauss-Legendre numerical integration introduced in previous lecture are used to evaluate the integration numerically.

The element stiffness matrix is then

$$\mathbf{K}^e = \int_{V_e} \mathbf{B}^{\mathbf{T}}\mathbf{D}\mathbf{B}\,dV = t\int_{-1}^{1}\int_{-1}^{1} \mathbf{B}^{\mathbf{T}}\mathbf{D}\mathbf{B}\left|\mathbf{J}\right| d\zeta d\eta$$

or

$$\mathbf{K}^{\mathrm{e}} = \mathrm{t}\int_{-1}^{1}\int_{-1}^{1} \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_1}{\partial y} \\ 0 & \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} \\ \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_2}{\partial y} \\ 0 & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} \\ \cdots & \cdots & \cdots \\ \frac{\partial N_8}{\partial x} & 0 & \frac{\partial N_8}{\partial y} \\ 0 & \frac{\partial N_8}{\partial y} & \frac{\partial N_8}{\partial x} \end{bmatrix} \mathbf{D} \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \cdots & \frac{\partial N_8}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & \cdots & 0 & \frac{\partial N_8}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \cdots & \frac{\partial N_8}{\partial y} & \frac{\partial N_8}{\partial x} \end{bmatrix} \left|\mathbf{J}\right| \mathrm{d}\zeta \mathrm{d}\eta$$

Here t is the thickness of the body.

Question 3: What is the size of the element stiffness matrix for Q8 element?

Lecture 9 gives details of numerical integration in 2D.

10.6 Body Forces

For the body force term at the element level, the numerical integration formula is:

$$\int_{V_e} \mathbf{N}^{\mathbf{T}}\mathbf{b}\,\mathrm{d}V = \mathrm{t}\int_{-1}^{1}\int_{-1}^{1} \mathbf{N}^{\mathbf{T}}\mathbf{b}\left|\mathbf{J}\right|\mathrm{d}\zeta\mathrm{d}\eta = \mathrm{t}\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\mathrm{w}_i \mathrm{w}_j \mathbf{N}(\zeta_i,\eta_j)^{\mathbf{T}}\mathbf{b}(\zeta_i,\eta_j)\left|\mathbf{J}\right|(\zeta_i,\eta_j)\right]$$

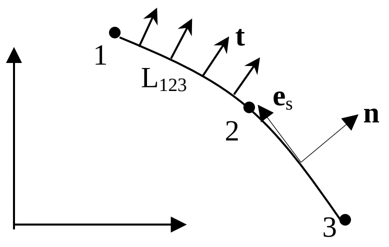
Note that the body force in 2D is a vector of 2 components, $\mathbf{b} = \left\{\mathrm{b}_x \quad \mathrm{b}_y\right\}^{\mathrm{T}}$, while $\mathbf{N}^{\mathrm{T}}$ is 16 by 2, therefore the above expression results in a vector of size 16.

10.7 Point Forces and Surface Tractions

When point forces are present, the mesh is usually generated in such a way that there will be a node at each loading point. The force vector then is simply the components of the point force, e.g. in 2D

$$\int_{S_{et}} \mathbf{N}^{\mathbf{T}}\mathbf{t}dS \rightarrow \begin{Bmatrix} f_{1x} \\ f_{1y} \\ \cdots \\ f_{8y} \end{Bmatrix}$$

For the surface tractions in 2D, the integration becomes one-dimensional. Depending on the orientation, the formula takes slightly different forms. To derive a generalized formula, let's consider a curved element edge L_{123} passing 3 nodes (Q8 element) as show in the figure below:



Assuming that the element is traversed counter clockwisely (i.e. the material is on the left side of the boundry), then the tangent direction of an arc line lement is

$$\mathbf{e}_s = \frac{dx}{ds}\mathbf{e}_1 + \frac{dy}{ds}\mathbf{e}_2$$

and the outward normal to the boundary is $\mathbf{n} = \mathbf{e}_s \times \mathbf{e}_3 = \frac{dy}{ds}\mathbf{e}_1 - \frac{dx}{ds}\mathbf{e}_2$

Assuming the sufcae traction $\mathbf{t}$ is expressed locally in terms of normal and tangent components t_n and t_s, i.e.

$$\mathbf{t} = t_n\mathbf{n} + t_s\mathbf{e}_s$$

The equivalent nodal forces corresponding to the three nodes on L_{123} are then

$$\begin{Bmatrix} f_{1x} \\ f_{1y} \\ f_{2x} \\ f_{2y} \\ f_{3x} \\ f_{3y} \end{Bmatrix} = \int_{L_{123}} \mathbf{N}^{\mathbf{T}}\mathbf{t}dS = t\int_{L_{123}} \mathbf{N}^{\mathbf{T}}\left(t_n\mathbf{n}+t_s\mathbf{e}_s\right)ds = t\int_{L_{123}} \begin{bmatrix} N_1 & 0 \\ 0 & N_1 \\ N_2 & 0 \\ 0 & N_2 \\ N_3 & 0 \\ 0 & N_3 \end{bmatrix}\left(t_n \begin{Bmatrix} \frac{dy}{ds} \\ -\frac{dx}{ds} \end{Bmatrix} + t_s \begin{Bmatrix} \frac{dx}{ds} \\ \frac{dy}{ds} \end{Bmatrix} \right) ds$$

here N_i is the 1D shape function for L_{123}, i.e. (lecture 5)

$$N_1(\zeta) = \frac{\zeta}{2}(\zeta-1)$$
$$N_2(\zeta) = \left(1-\zeta^2\right)$$
$$N_3(\zeta) = \frac{\zeta}{2}(\zeta+1)$$

Using isoparametric formulation

$$x = \sum_{i=1}^{3} N_i(\zeta)x_i\,,\ y = \sum_{i=1}^{3} N_i(\zeta)y_i$$

we have $$dx = \left(\sum_{i=1}^{3}\frac{dN_i(\zeta)}{dx}x_i\right)d\zeta\,,\ dy = \left(\sum_{i=1}^{3}\frac{dN_i(\zeta)}{dy}y_i\right)d\zeta$$

Therefore the component form of the force vector is

$$f_{ix} = t\int_{L_{123}} N_i\left(t_n dy + t_s dx\right) = t\int_{-1}^{1} N_i\left(t_n\sum_{i=1}^{3}\frac{dN_i(\zeta)}{dy}y_i + t_s\sum_{i=1}^{3}\frac{dN_i(\zeta)}{dx}x_i\right)d\zeta$$

and $$f_{iy} = t\int_{L_{123}} N_i\left(-t_n dx + t_s dy\right) = t\int_{-1}^{1} N_i\left(-t_n\sum_{i=1}^{3}\frac{dN_i(\zeta)}{dx}x_i + t_s\sum_{i=1}^{3}\frac{dN_i(\zeta)}{dy}y_i\right)d\zeta$$

The above integration can be readily evaluated using 3-point Gauss integration in 1D.

Question 4: Why is the integration formula of surface tractions different from the others?

10.8 Hints to Questions

Q1: Because Q3 is quadratic.
Q2: No.
Q3: 16×16.

Q4: Because its dimensionality is reduced by 1.

10.9 References

Hughes, T. J. R., 1987, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Inc.

11 Some Theoretical Aspects of FEM

11.1 Purpose

This lecture covers some theoretical aspects of Finite Element Method. We will avoid strict theoretical proofs, keep the mathematics to a minimum, and focus on concepts and useful conclusions. Hughes' book gives fairly detailed mathematical treatments on the issues discussed here. The relevant sections of Hughes' book that deal with the topics discussed are listed in the subtitles for convenience in reading. The readers are encouraged to study the textbook for deeper understanding of these issues.

11.2 Well-posedness of a Problem (Hughes, §2.1)

Do not attempt to "solve" a problem without a precise knowledge of what the problem is. Make sure a problem is well defined and well posed before attempting to solve it.

11.3 Continuity of Displacements, Strains and Stresses (Hughes, §1.Appendix and §3.1)

The FEM formulation we introduced so far is called the *finite element displacement approach* because the basic variables are the displacements, while the strains and stresses are derived variables. In order for the FEM solution to converge to the exact solution as the mesh is refine, certain continuity/discontinuity conditions must be satisfied. The basic convergence requirements are that the shape functions be:

1. smooth (i.e., at least C^1 which means the first order derivatives are continuous) on each element interior
2. continuous across each element boundary
3. complete

Conditions 1 and 2 guarantee that first derivatives of the shape functions have, at worst, finite jumps across the element interfaces, and that all integrals for the computation of element arrays are well defined (instead of divergent). Shape functions satisfy Conditions 1 and 2 are of class C^0, and the corresponding elements are referred to as C^0-elements.

The shape functions for C^0-elements are continuous across element boundaries, while their derivatives are typically discontinuous (but the first derivatives are square-integrable). Consequently, the displacements for C^0-elements are continuous across the element boundaries, but the strains and stresses are discontinuous across the element boundaries. *Increasing the order of the shape function does not help in this regard*. To calculate the stresses and strains on the boundary, a commonly used technique is to extrapolate stresses and strains from the Gaussian points to the boundaries or nodes, then take the average values from surrounding elements.

Condition 3 means that the shape function must contain all constant and linear monomials so that constant strain field can be reproduced exactly. For example, in 1D, the shape function must contain at least $c_0 + c_1 x$, while in 2D at least $c_0 + c_1 x + c_2 y$. This requirement can be illustrated in the following scenario: As the finite element mesh is further and further refined, the exact solution and its derivatives approach constant values over each element domain. In elasticity, the presence of all monomials through linear terms means that an element may exactly present all rigid body motions (i.e. translations and rotations) and constant strain states. We thus often speak of completeness in terms of the ability to present rigid body motions and constant strains.

11.4 Galerkin Formulation; Symmetry and Positive-definiteness of K (Hughes, §2.8)

Assuming that the material constitutive matrix is positive-definite (which is true for elastic materials), and that suitable boundary conditions are incorporated (which means rigid body motions are precluded in the solution), the Galerkin formulation leads to a *symmetric* and *positive-definite* stiffness matrix **K**.

The symmetry and positive-definiteness of **K** offer a number of advantages in solution process including reduced storage requirement, well-behaved governing equations and efficient solution methods.

11.5 "Best Approximation" and Error Estimates: Why Standard FEM Usually Works and Why Sometimes It Does Not (Hughes, §4.1)

The weak form for 1D rod problem

$$EA\int_0^L \frac{dw}{dx}\frac{du}{dx}dx = \int_0^L wb\,dx + w(L)P \tag{11-20}$$

or 2D elasticity problem

$$\int_V \delta\varepsilon^T \sigma dV = \int_V \delta\mathbf{u}^T\mathbf{b}dV + \int_S \delta\mathbf{u}^T\mathbf{t}dS \tag{11-21}$$

can be written using a more general notation

$$a(\mathbf{w},\mathbf{u}) = (\mathbf{w}, \boldsymbol{f}) + (\mathbf{w}, \boldsymbol{h})_\Gamma \tag{11-22}$$

where $\mathbf{u}$ is the trial solution, $\mathbf{w}$ is the weighting function, $a(\square,\square)$, $(\square,\square)$, and $(\square,\square)_\Gamma$ are operators. For example, for 1D rod problem, $a(\square,\square)$ is defined as $a(\square,\square) = EA\int_0^L \frac{d(\square)}{dx}\frac{d(\square)}{dx}dx$. We assume that $a(\square,\square)$, $(\square,\square)$, and $(\square,\square)_\Gamma$ are symmetric and bilinear.

For FEM, the solution is usually approximate. If we denote the approximate trial solution and weighting function as $\mathbf{u}^h$ and $\mathbf{w}^h$ respectively (h indicates some measure of the element size), and the weak form is

$$a\left(\mathbf{w}^h, \mathbf{u}^h\right) = \left(\mathbf{w}^h, \boldsymbol{f}\right) + \left(\mathbf{w}^h, \boldsymbol{h}\right)_\Gamma \tag{11-23}$$

The term $a(\mathbf{w},\mathbf{w})$ is generally regarded as the (strain) energy inner product, and its square root the energy norm.

Let $\mathbf{e} = \mathbf{u}^h - \mathbf{u}$ denote the *error* in the FEM approximation, then it can be proved that

A. $a\left(\mathbf{w}^h, \mathbf{e}\right) = 0$ for all $\mathbf{w}^h \in \boldsymbol{V^h}$ (finite-dimensional collection of weighting functions)

B. $a(\mathbf{e},\mathbf{e}) \le a\left(\mathbf{U}^h - \mathbf{u}, \mathbf{U}^h - \mathbf{u}\right)$ for all $\mathbf{U}^h \in \boldsymbol{S^h}$ (finite-dimensional collection of trial solutions)

Part A. means that the error is ***orthogonal*** to the subspaces $\boldsymbol{V^h} \subset \boldsymbol{V}$ (collection of weighting functions), or we can say, $\mathbf{u}^h$ is the projection of $\mathbf{u}$ (the exact solution) onto $\boldsymbol{S^h}$ with respect to the operator $a(\square,\square)$, since $a\left(\mathbf{w}^h, \mathbf{e}\right) = 0$ leads to $a\left(\mathbf{w}^h, \mathbf{u}^h\right) = a\left(\mathbf{w}^h, \mathbf{u}\right)$.

Part B. means that there is no member of $\boldsymbol{S^h}$ that is a better approximation than $\mathbf{u}^h$ (the solution of the Galerkin finite element problem). This is referred to as the ***best approximation property***. We can also say that derivatives of $\mathbf{u}^h$ best fit $\mathbf{u}$ in a weighted sense. In elasticity, this means the accuracy of strains and stresses is optimized.

It can also be proved that, if the essential boundary conditions are homogeneous, then

$$a(\mathbf{u},\mathbf{u}) = a\left(\mathbf{u}^h, \mathbf{u}^h\right) + a(\mathbf{e},\mathbf{e})$$

This is in analogue to the Pythagorean theorem. From which we can easily show that

$$a\left(\mathbf{u}^h, \mathbf{u}^h\right) \le a(\mathbf{u},\mathbf{u})$$

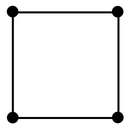
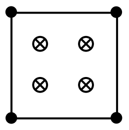
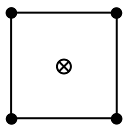
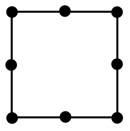
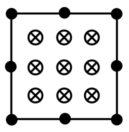
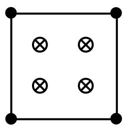
that is, (for displacement approach) the *approximate solution always underestimates the strain energy*. Such displacement solution is frequently referred to as the *lower bound* solution. Another way to put it is that the displacement FEM appears to be *stiffer* than the exact solution is. [Note that the total potential energy, which is the strain energy minus the external work, is a different quantity. *While the displacement FEM always underestimates the strain energy, it always overestimates the total potential*. Do not get confused.]

The preceding mathematical results provide a degree of confidence in the theoretical soundness of the finite element method. In practice, for typical problems, the results obtained for Galerkin FEM are generally very good.

What could possibly go wrong? When there are no functions in $\boldsymbol{S}^h$ that is a good approximation to $\mathbf{u}$. This can happen for constrained media problems such as for incompressible or nearly incompressible materials (Poisson's ratio is close to 0.5). Conventional Galerkin FEM may be so poor for these problems that is practically useless.

11.6 Reduced Integration (Hughes, §4.4)

To achieve accurate results in numerical integration, sufficient integration points are needed. The number of integration points for normal Gaussian quadrature is shown in the table below.

	Degree of polynomial in integrand of K_{ij}	Full Gaussian quadrature	Reduced Gaussian quadrature
4-node quadrilateral 	quadratic	2×2 	one-point 
8-node quadrilateral (Q8) 	4th order	3×3 	2×2 

Sometimes, reduced integration (i.e. fewer integration points) may be desirable for two reasons. First, since the expense of generating a **K** matrix by numerical integration is proportional to the number of integration points, using reduced integration lowers computational cost. Second, a reduced integration tends to soften an element, thus countering the overly stiff behavior associated with an assumed displacement field. Softening comes about because certain higher order polynomial terms happen to vanish at Gaussian points of a low-order rule, so that these terms make no contribution to strain energy. In other words, with fewer sampling points, some of the more complicated displacement modes offer less resistance to deformation. Reduced integration is particularly effective in overcoming some numerical difficulties such as locking for incompressible materials. The number of Gaussian points for commonly used reduced integration is also listed in the above table.

In general, reduced integration may be able to simultaneously reduce cost, reduce accuracy in the evaluation of integral expressions, and *increase* the accuracy of a finite element analysis.

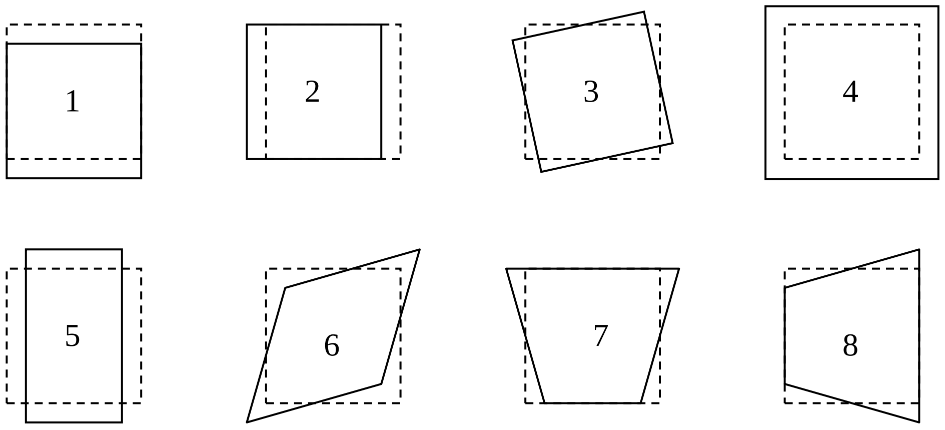
Reduced integration should not be used if cost reduction is the only motivation.

11.7 Zero-energy Mode and Rank Deficiency (Hughes, §4.6)

An instability in elements may occur which is also called a *spurious mode*, a *mechanism*, a *kinematic mode*, an *hourglass mode*, or a *zero-energy mode*. The term zero-energy mode refers to a nodal displacement vector $\{\mathbf{u}_a\}$ that is not a rigid-body motion but nevertheless produces zero strain energy $\{\mathbf{u}_a\}[\mathbf{K}]\{\mathbf{u}_a\}/2$. Zero-energy modes arise because of shortcoming in the element formulation, such as use of a low-order Gaussian integration.

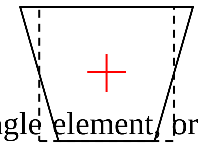
An example of zero-energy modes for 4-node bilinear element:

There are 8 independent deformation modes shown in the following

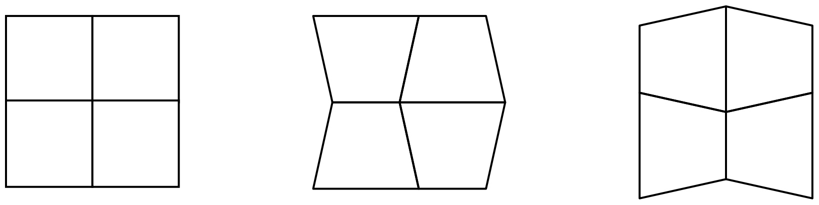


- mode 1 to 3: rigid body motions
- mode 4 to 6: non-zero energy modes
- mode 7 and 8: zero energy mode if one-point integration is used, because a single Gaussian point at the center dose not sense these mode as $\varepsilon_x = \varepsilon_y = \gamma_{xy} = 0$ at the center. These two spurious modes disappear if higher order Gaussian integration (2 or higher) is used.

To detect a zero energy mode, draw a small cross at each Gaussian point. For the given deformation mode, if the length of both lines of all crosses remains the same and the angle between the two lines remains 90 degree (which means all strain components are zero at all Gaussian points), it is a zero-energy mode. This is illustrated in the figure below.



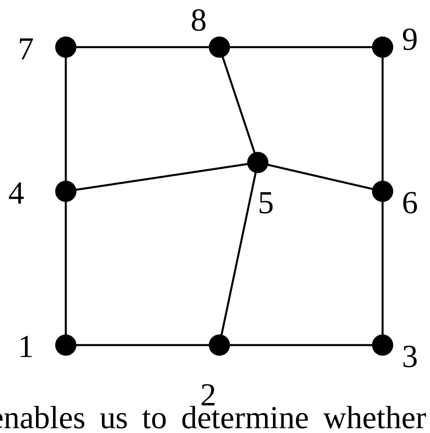
Hourglass mode can occur in a single element, or a mesh of elements. It is called hourglass modes because of their physical shape. Occurrence of hourglass mode indicates that the solution of the problem is not unique, and the global stiffness is singular or nearly singular. A good FEM program should give warnings when hourglass mode is detected.



For an element with N nodes in 2D, the size of the **K** matrix is 2N, and there will be 2N independent deformation modes. As 3 modes are rigid body motions, the rank of **K** is 2N-3 if no zero-energy mode exists. If there are M zero-energy modes, the rank of **K** will be reduced by M, i.e. rank(**K**)=2N-3-M. Therefore use of reduced integration could lead to rank deficiency. Whether or not this is bad depends on the particular problem being investigated. Some commercial software such as LS-DYNA recommends reduced integration with hourglass control when solving the equation.

11.8 Patch Test (Hughes, §4.6)

According to Hughes, the patch test is the most practically useful technique for accessing element behavior. The patch test is said to be passed if the states 1, x, and y (i.e. constant stress states) are exactly representable by an arbitrary "patch" of elements. An example is shown in the figure below. In constructing an arbitrary patch, it is important to use an irregular geometry, because some elements pass the patch test in certain special configurations but not others. Apply constant strain loading conditions to the patch (e.g. $\varepsilon_{xx} = \text{const.}$, others=0). If the solution at all nodes (in particular node 5) is exact, and the displacement gradients are also exact, then the patch test is passed.



Basically, the patch test enables us to determine whether or not an element satisfies the completeness condition. The patch test is frequently applied to elements employing nonstandard features (e.g. selective integration, incompatible interpolations, etc.)

11.9 Nonconfirming Elements (Hughes, §4.7)

C^0-elements (or elements with higher continuity) in which displacements are continuous across the element boundaries are called compatible elements. Other finite elements have also been proposed in which C^0 continuity is violated, resulting in functions that do not possess square-integrable first derivatives. Dirac delta distributions exist in the derivatives of such functions, and the product of these is not well defined. Elements of this type are referred to as ***incompatible***, or ***nonconfirming***.

There is no guarantee that convergence will be achieved for incompatible elements. Some incompatible elements, such as Wilson's incompatible modes and Taylor's incompatible modes, have been developed successfully, and they generally give better results in the stresses than conventional compatible elements. Patch test is the basic requirement in developing incompatible elements.

11.10 Mixed or Hybrid Elements (Hughes, §4.3)

It is also possible to develop elements in which both the displacements and stresses are independent variables. Such an approach is called the mixed or hybrid approach. Because the stresses are obtained directly from the solution instead of being postprocessed from the derivatives of the displacement field, the accuracy in stresses is usually better than the displacement approach. The hybrid approach is especially effective certain problems such as incompressible or nearly incompressible materials in which the pressure is taken as an independent variable.

11.11 References

Hughes, T. J. R., 1987, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Inc.

12 Computer Implementation

12.1 Purpose

The goal of this lecture is to lay the framework for an object-oriented finite element program. We will do this at two levels, first we will implement the simplest program that still illustrates the general approach. Next we will discuss a more general approach, applicable to multi-physics problems.

12.2 Design of the Finite Element Program

The general flow of a linear finite element solution is shown in Figure 12-1.

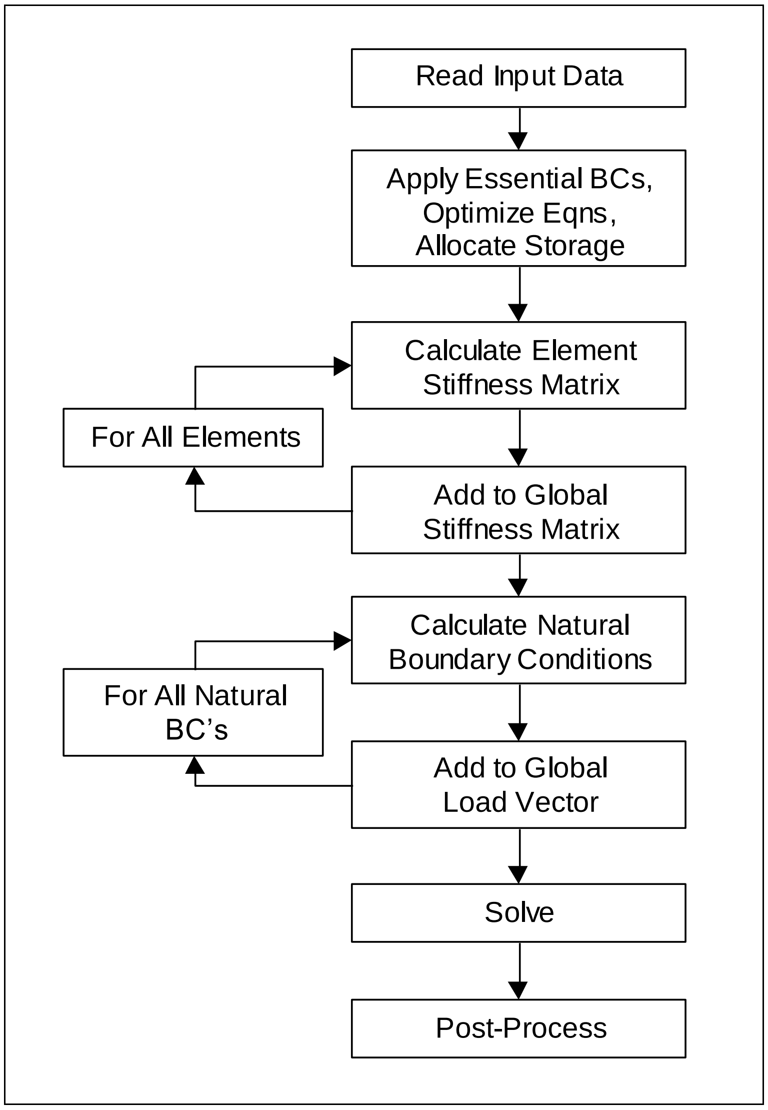

Read Input Data
Apply Essential BCs, Optimize Eqns, Allocate Storage
Calculate Element Stiffness Matrix
For All Elements
Add to Global Stiffness Matrix
Calculate Natural Boundary Conditions
For All Natural BC's
Add to Global Load Vector
Solve
Post-Process

Figure 12-1: Flow of finite element solution

Figure 12-1 shows the solution procedure as a linear process for which a procedural design is appropriate. While such a design can certainly be implemented, an object-oriented approach offers the potential for future expansion and easier maintenance.

In object-oriented design, we define objects, each with their own data and functions. These objects interact with each other through public interfaces. Ideally, such an approach allows each object to be independent of other objects and enhances reliability and maintainability.

With respect to the finite element program, some objects are intuitive, such as femNode and femElement objects. Also, examining Figure 12-1, it also seems natural to have an object that stores all the problem data (femData) and a separate object that accesses the data to perform the solution (femSolve). As our discussion evolves, other classes will be identified. Also, note the format used for our class names; the first three letters identify a grouping of classes, followed by the class name with the first letter capitalized. In C++, the grouping would typically be a subproject in a larger project and the files would be found in a subdirectory with the same name (i.e. fem for finite element classes). For a variable or function, the first letter is lower case, followed by upper case letters to help readability (as appropriate). For example, getElementK() is a function and d_material is a data member variable in a class (the data of a class begins with d_).

The femElement class will be the base class of all elements. From the element class we will derive elements of different types, for example eight-noded quadrilateral elements (Q8) and six-noded triangular elements (T6). Please remember that this is our simple implementation, a more general implementation could separate the element functions (interpolation, integration) from the physics of the problem. In this simple implementation, we are assuming all elements solve only one type of differential equation. The elements will be organized as shown in Figure 12-2.

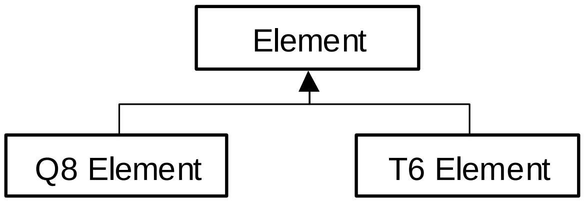


Figure 12-2: Element classes

The reason we define a base class and derive element types from it, is that this allows us to create different element types when we read in the data, but to still access each element in a common manner (polymorphism). Let us be specific. In the femModel class we will have an array of element pointers. The length of the array is the number of elements (`d_numElems`). We dynamically allocate only the required memory for our particular problem.

```
femElement **elems;
```

```
      elems = new femElement*[d_numElems];
```

Then, as we are reading our data, we can create new elements that are either Q8 or T6, for example:

```
      elems[5] = new femElementQ8();
      elems[6] = new femElementT6();
      etc.
```

Each element in our array is now a specific element type. The power of this approach is that we can now access the virtual functions defined in the femElement interface without concern for what kind of element it is. For example, when assembling the global stiffness matrix, we can obtain each element stiffness as follows:

```
      for(int i=0; i<d_numElems; i++)
      {
            elems[i]->getElementK(ke);
      }
```

Where ke is a matrix where we can store the element stiffness matrix (it is passed by reference so data can be modified and accessed). The point is that we no longer care what type of element we are dealing with, we just tell each element to do its appropriate task. This same pattern is used for all the data for which we do not know the number of items we will have until we read the problem data. This includes nodes, elements, materials, essential boundary conditions, point boundary conditions, and natural boundary conditions.

In a more advanced implementation, these items would be accessed using iterators. In our implementation femData will have a function that returns the number of items and then provides access to a particular item. For example, to loop through each node and get the node coordinates, we use the following example code.

```
      femData dat;
      femNode* node;
      double x, y;
      for(int i=0; i<dat.getNumNodes(); i++)
      {
            node = dat.getNode(i);
            x = node->getX();
            y = node->getY();
      }
```

Finally, an absolute design rule is that ***NO*** data is public.

12.2.1 femElement Class

Description

The femElement class is the base class for all elements. The femElement class defines pure virtual functions that define the external interface. It also stores information common to all element types.

The basic information contained in the femElement class is an id, a pointer to a material, the number of nodes in the element, and an array of pointers to the nodes. When the element is constructed, all the data is passed. It is possible to access the node id's, which is needed when postprocessing. All other functions are pure virtual and defined in the specific element type.

Definition

```
class femElement
{

      protected:

      int d_id;
      femMaterial* d_material;
      int d_numNodes;
      femNode** d_nodes;

      //never to be used constructors
      femElement();
      femElement(const femElement &elem);

  public:

      // Constructors
      femElement(int id, femMaterial* mat, int numNodes, femNode*
nodes[]);

   //Destructor
   virtual ~femElement();

   // Functions
      int getID() const;
      int getNumNodes() const;
      void getNodes(femNode* nodes[]) const;
      void printData(ostream &out) const;
      virtual int getNumDofs() const = 0;
      virtual int getBandwidth() const = 0;
      virtual void getElementKF(utlMatrix &ke, utlVector &fe, int
eqns[],
            ostream &out) = 0;
      virtual void calcData() = 0;
      virtual void getNodalData(utlStress2D stresses[]) = 0;
      virtual void printResults(ostream &out) const = 0;
};
```

12.2.2 femElementQ8 Class

Description

We will only illustrate a single element derived from the base class, the Q8 element. However, such a derivation allows the easy addition of other element types, such at the T6 element.

The femElementQ8 class implements the eight-noded quadrilateral element. It is derived from the femElement base class.

Definition

```
class femElementQ8: public femElement
{

      private:

      utlStress2D d_stress[3][3];

      // Never to be used constructors.
      femElementQ8();
      femElementQ8(const femElementQ8 &elem);

      public:

      // Constructors
      femElementQ8(int id, femMaterial* mat, int numNodes,
            femNode* enodes[]);

   // Destructor
   virtual ~femElementQ8();

   // Functions
      virtual int getNumDofs() const;
   virtual int getBandwidth() const;
      virtual void getElementKF(utlMatrix &ke, utlVector &fe, int
eqns[],
            ostream &out);
      void calcData();
      void getNodalData(utlStress2D stresses[]);
      virtual void printResults(ostream &out) const;

      private:

      void getBMatrix(double r, double s, utlMatrix &b, double
&detj);
};
```

12.2.3 femNode Class

Description

The femNode class stores geometry, equation numbers for each degree of freedom (dof), the solution, and boundary condition data. In this implementation, we illustrate two dof's at the node, UX and UY. The data for these dofs is stored in arrays. The appropriate value is accessed using a enumerated type as follows:

```
enum DOF {UX=0, UY};
```

For example, we can then return the UX dof value by:

```
return d_dof[UX];
```

Definition

```
class femNode
{

      private:

      int d_id;
      double d_x, d_y;
      femDof* d_dof[2];

      // Never used constructors
      femNode();
      femNode(const femNode& node);

      public:

      // Constructors
      femNode(int id);

   // Destructor
   virtual ~femNode();

   // Functions
      int getID() const;
      void setCoords(double x, double y);
   double getX() const;
      double getY() const;
      void createActiveDof(DOF dof);
      femDof* getDof(DOF dof) const;
      void printData(ostream &out) const;
      void printResults(ostream &out) const;
};
```

12.2.4 femDof Class

Description

The femDof class implements a degree of freedom. Each dof has an equation number with associated value. Dof's with either no boundary condition or a natural boundary condition are active. If the dof has an essential boundary condition, the dof is not active. During matrix assembly, if the boundary condition is essential, the appropriate element stiffness coefficient is multiplied by the boundary condition value and subtracted from the global load vector. If it is a natural boundary condition the value is added to the load vector. After the solution, the results are stored in the value.

Definition

```
enum DOF {UX=0, UY};

class femDof
{
      private:

      double d_value;
      int d_eqn;
      bool d_active;

      //never used copy constructor
      femDof (femDof &dof);

      public:

      // Constructors
      femDof();

   //Destructor
   virtual ~femDof();

   // Functions
      void setNotActive();
      bool isActive() const;
      int setEqn(int currEqn);
      int getEqn() const;
      void setValue(double value);
      double getValue() const;
      void print(ostream &out) const;
};
```

12.2.5 femData Class

Description
The femData class stores all problem data. It provides a means to access all elements, nodes, material, and boundary condition data. Memory for all data is allocated dynamically after the size of the required storage is known. Simple arrays are used for the nodes and materials. These are allocated by:

```
d_nodes = new femNode[d_numNodes];
```

Because we want to use polymorphism for the elements, we allocate an array of element pointers by:

```
d_elems = new femElement*[d_numElems];
```

We then save a particular element using:

```
m_elems[i] = new femElementQ8();
```

The class also returns pointers to requested nodes and elements. In a more advanced implementation, this would be accomplished using iterators.

Definition

```
class femData
{
      private:

      femNode** d_nodes;
      femMaterial** d_matprops;
      femElement** d_elems;
      femEssentialBC** d_essentialBCs;
      femPointBC** d_pointBCs;
      femNaturalBC** d_naturalBCs;
      char* d_title;
      int d_numNodes;
      int d_numMats;
      int d_numElems;
      int d_probType;
      int d_numEssentialBCs;
      int d_numPointBCs;
      int d_numNaturalBCs;
      int d_neq;
      int d_bandw;

      public:

      // Constructors
      femData();

      //Destructor
```

```
        virtual ~femData();

           // Functions
           void readData(istream &inp);
           void writeData(ostream &out);
           int getNumNodes();
           int getNumElems();
           int getNumEssentialBCs();
           int getNumPointBCs();
           int getNumNaturalBCs();
           femNode *getNode(int node);
           femElement *getElem(int elem);
           femEssentialBC *getEssentialBC(int ebc);
           femPointBC *getPointBC(int pbc);
           femNaturalBC *getNaturalBC(int nbc);
           void writeResults(ostream &out);
           void writePlotFile(ostream &plt);
};
```

12.2.6 femEssentialBC Class

Description
The femEssentialBC class stores essential boundary condition data.

Definition

```
class femEssentialBC
{
      private:

      femNode* d_node;
      DOF d_dof;
      double d_value;

      //never used constructors
      femEssentialBC();
      femEssentialBC (femEssentialBC &ebc);

      public:

      // Constructors
      femEssentialBC(femNode* node, DOF dof, double value);

   //Destructor
    virtual ~femEssentialBC();

   // Functions
      femNode* getNode() const;
      DOF getDof() const;
      double getValue() const;
};
```

12.2.7 *femNaturalBC Class*

Description

The femNaturalBC class stores natural boundary condition data. Data is given for two corner nodes of an element. To use this, a search is performed to identify the element. Then the value is assumed to be positive acting away from the element.

Definition

```
class femNaturalBC
{
      private:

      femNode* d_node1;
      femNode* d_node2;
      double d_value;

      //never used constructors
      femNaturalBC();
      femNaturalBC (femNaturalBC &ebc);

  public:

      // Constructors
      femNaturalBC(femNode* node1, femNode* node2, double value);

   //Destructor
   virtual ~femNaturalBC();

   // Functions
      femNode* getNode1() const;
      femNode* getNode2() const;
      double getValue() const;
};
```

12.2.8 femPointBC Class

Description
The femPointBC class stores point boundary condition data (a special case of natural boundary condition data).

Definition

```
class femPointBC
{
      private:

      femNode* d_node;
      DOF d_dof;
      double d_value;

      //never used constructors
      femPointBC();
      femPointBC (femPointBC &ebc);

      public:

      // Constructors
      femPointBC(femNode* node, DOF dof, double value);

   //Destructor
   virtual ~femPointBC();

   // Functions
      femNode* getNode() const;
      DOF getDof() const;
      double getValue() const;
};
```

12.2.9 femSolve Class

The femSolve class manages the solution. The idea is to keep the solution independent of the problem data and storage.

Description
This class manages the solution, including assembly of the global stiffness matrix and load vector.

Responsibilities
The femSolve class numbers the equations,

Attributes
```
private:
utlMatrix *m_k;
utlVector *m_f;
utlVector *m_sol;
int m_neq;
int m_bandw;
```

Interface
```
public:
// Constructors
femSolve();

//Destructor
virtual ~femSolve();

// Functions
void solve(modModel &mod, ostream &out);

private:
int setEquationNumbers(modModel &mod);
int getBandwidth(modModel &mod);
void assembleRHS(modModel &mod, utlVector *f);
void assembleK(modModel &mod, utlMatrix *k, utlVector *f,
      ostream &out);
void saveSolution(modModel &mod, utlVector *sol);
void calcData(modModel &mod);
```

12.2.10 femMaterial Class

Description

Th femMaterial class is the base material class. It provides the interface to all materials.

Definition

```
class femMaterial
{
      protected:

      int d_id;

   //Never used constructors.
      femMaterial();
      femMaterial(const femMaterial &mat);

      public:

      //Constructors
      femMaterial(int id);

   //Destructor
   virtual ~femMaterial();

   //Functions
      int getID() const;
      virtual void getDMatrixStress(utlMatrix &d) = 0;
      virtual void printData(ostream &out) const = 0;
};
```

12.2.11 femMaterialElastic Class

Description

Th femMaterialElastic class implements an elastic material.

Definition

```
class femMaterialElastic: public femMaterial
{
      protected:

      double d_e;       //young's modulus
      double d_nu;      //poisson's ration
      double d_k;       //conductivity
      double d_thick; //thickness

      // never used copy constructor
      femMaterialElastic(const femMaterialElastic &mat);

      public:

      // Constructors
      femMaterialElastic();
      femMaterialElastic(int id, double e, double nu,
            double thick);

    //Destructor
    virtual ~femMaterialElastic();

    // Functions
      virtual void printData(ostream &out) const;
      virtual void getDMatrixStress(utlMatrix &d);
};
```

12.2.12 femGauss3Pt Class

Description
The femGauss3Pt class provides three gauss point data.

Definition

```
class femGauss3Pt
{
      private:

      int d_numgp;
      double d_point[3];
      double d_weight[3];

      public:

      //Constructor
      femGauss3Pt();

   //Destructor
   virtual ~femGauss3Pt();

   // Functions
      int getNumPts() const;
      double getGaussPt(int gp) const;
      double getWeight(int gp) const;
};
```

12.2.13 utlMatrix Class

Description

The utlMatrix class is the base matrix class.

Definition

```
class utlMatrix
{
      protected:
   double **m_coeff;          // Coefficients of matrix
   int m_nr, m_nc;        // Number of columns and rows

      // Never to be used copy constructor
      utlMatrix(const utlMatrix &matrix);
      // Never to be used Constructor
      utlMatrix();

      public:

      //Destructor
   virtual ~utlMatrix();

      // Functions
      void zero();
      int getNumRows() const;
      int getNumCols() const;
      void print(ostream &out);
   virtual void setCoeff(int i, int j, double value) = 0;
      virtual void addCoeff(int i, int j, double value) = 0;
      virtual double getCoeff(int i, int j) = 0;
      virtual void mult(utlMatrix &b, utlMatrix &c) = 0;
      virtual void mult(utlVector &b, utlVector &c) = 0;
      virtual void trans(utlMatrix &a) = 0;
      virtual void gauss(utlVector *b, utlVector *x) = 0;
};
```

12.2.14 utlMatrixFull Class

Description

The utlMatrixFull class stores a complete matrix.

Definition

```
class utlMatrixFull: public utlMatrix
{
      protected:
      //never to be used constructors
      utlMatrixFull();
   utlMatrixFull(const utlMatrixFull &matrix);

      public:

      // Constructors
   utlMatrixFull(int nr, int nc);

      //Destructor
   virtual ~utlMatrixFull();

      // Functions
      virtual void setCoeff(int i, int j, double value);
      virtual void addCoeff(int i, int j, double value);
      virtual double getCoeff(int i, int j);
      virtual void mult(utlMatrix &b, utlMatrix &c);
      virtual void mult(utlVector &b, utlVector &c);
   virtual void trans(utlMatrix &a);
      virtual void gauss(utlVector *b, utlVector *x);
};
```

12.2.15 utlMatrixBandedSymm Class

Description

The utlMatrixBandedSymm class stores a banded symmetric matrix. Used for storing the global element matrix.

Definition

```
class utlMatrixBandedSymm: public utlMatrix
{
      protected:

      //never to be used constructors
      utlMatrixBandedSymm();
   utlMatrixBandedSymm(const utlMatrixBandedSymm &matrix);

      public:

      // Constructors
   utlMatrixBandedSymm(int nr, int nc);

      //Destructor
   virtual ~utlMatrixBandedSymm();

      // Functions
      virtual void setCoeff(int i, int j, double value);
      virtual void addCoeff(int i, int j, double value);
      virtual double getCoeff(int i, int j);
      virtual void mult(utlMatrix &b, utlMatrix &c);
      virtual void mult(utlVector &b, utlVector &c);
    virtual void trans(utlMatrix &a);
      virtual void gauss(utlVector *b, utlVector *x);
};
```

12.2.16 utlVector Class

Description

The utlVector class stores a vector.

Definition

```
class utlVector
{
      protected:

      double *m_coeff;        // Coefficients of matrix
   int m_nr;                        // Number of rows

      //never to be used copy constructor
      utlVector(const utlVector &vector);

      public:

      // Constructors
      utlVector();
      utlVector(int nr);

   //Destructor
   virtual ~utlVector();

      // Functions
      void zero();
      int getNumRows() const;
      void setCoeff(int i, double value);
      void addCoeff(int i, double value);
      double getCoeff(int i);
      void print(ostream &out);
};
```

12.2.17 utlStress2D Class

Description

The utlStress2D class stores stress data.

Definition

```
class utlStress2D
{
      protected:

      // stress data
      double m_sigxx, m_sigyy, m_sigzz, m_sigxy;

      // never used constructors
      utlStress2D(utlStress2D&);

      public:

      // constructors
      utlStress2D();
      utlStress2D(double xx, double yy,
                        double zz, double xy);

      // destructor
   virtual ~utlStress2D();

      //functions
      void setSigXX(double xx);
      void setSigYY(double yy);
      void setSigZZ(double zz);
      void setSigXY(double xy);
      void addSigXX(double xx);
      void addSigYY(double yy);
      void addSigZZ(double zz);
      void addSigXY(double xy);
      double getSigXX() const;
      double getSigYY() const;
      double getSigZZ() const;
      double getSigXY() const;
      void zero();
};
```