INTRODUCTION TO THE FINITE ELEMENT METHOD

G. P. Nikishkov

2004 Lecture Notes. University of Aizu, Aizu-Wakamatsu 965-8580, Japan
niki@u-aizu.ac.jp

Contents

Chapter 1

Introduction

1.1 What is the finite element method

The finite element method (FEM) is a numerical technique for solving problems which are described by partial differential equations or can be formulated as functional minimization. A domain of interest is represented as an assembly of *finite elements*. Approximating functions in finite elements are determined in terms of nodal values of a physical field which is sought. A continuous physical problem is transformed into a discretized finite element problem with unknown nodal values. For a linear problem a system of linear algebraic equations should be solved. Values inside finite elements can be recovered using nodal values.

Two features of the FEM are worth to be mentioned:

1) Piece-wise approximation of physical fields on finite elements provides good precision even with simple approximating functions (increasing the number of elements we can achieve any precision).

2) Locality of approximation leads to sparse equation systems for a discretized problem. This helps to solve problems with very large number of nodal unknowns.

1.2 How the FEM works

To summarize in general terms how the finite element method works we list main steps of the finite element solution procedure below.

1. *Discretize the continuum.* The first step is to divide a solution region into finite elements. The finite element mesh is typically generated by a preprocessor program. The description of mesh consists of several arrays main of which are nodal coordinates and element connectivities.

2. *Select interpolation functions.* Interpolation functions are used to interpolate the field variables over the element. Often, polynomials are selected as interpolation functions. The degree of the polynomial depends on the number of nodes assigned to the element.

3. *Find the element properties.* The matrix equation for the finite element should be established which relates the nodal values of the unknown function to other parameters. For this task different approaches can be used; the most convenient are: the variational approach and the Galerkin method.

4. *Assemble the element equations.* To find the global equation system for the whole solution region we must assemble all the element equations. In other words we must combine local element equations for all elements used for discretization. Element connectivities are used for the assembly process. Before solution, boundary conditions (which are not accounted in element equations) should be imposed.

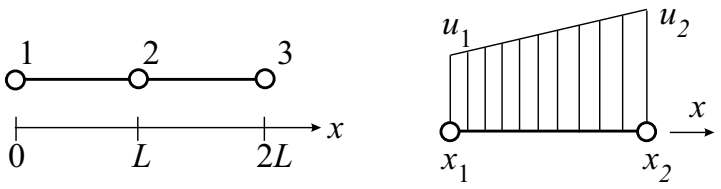


Figure 1.1: Two one-dimensional linear elements and function interpolation inside element.

5. *Solve the global equation system.* The finite element global equation sytem is typically sparse, symmetric and positive definite. Direct and iterative methods can be used for solution. The nodal values of the sought function are produced as a result of the solution.

6. *Compute additional results.* In many cases we need to calculate additional parameters. For example, in mechanical problems strains and stresses are of interest in addition to displacements, which are obtained after solution of the global equation system.

1.3 Formulation of finite element equations

Several approaches can be used to transform the physical formulation of the problem to its finite element discrete analogue. If the physical formulation of the problem is known as a differential equation then the most popular method of its finite element formulation is the *Galerkin method*. If the physical problem can be formulated as minimization of a functional then *variational formulation* of the finite element equations is usually used.

1.3.1 Galerkin method

Let us use simple one-dimensional example for the explanation of finite element formulation using the Galerkin method. Suppose that we need to solve numerically the following differential equation:

$$a\frac{d^2u}{dx^2} + b = 0, \qquad 0 \leq x \leq 2L \tag{1.1}$$

with boundary conditions

$$\begin{aligned} &u|_{x=0} = 0 \\ &a\frac{du}{dx}|_{x=2L} = R \end{aligned} \tag{1.2}$$

where u is an unknown solution. We are going to solve the problem using two linear one-dimensional finite elements as shown in Fig. 1.1.

Fist, consider a finite element presented on the right of Figure. The element has two nodes and approximation of the function $u(x)$ can be done as follows:

$$\begin{aligned} &u = N_1u_1 + N_2u_2 = [N]\{u\} \\ &[N] = [N_1 \;\; N_2] \\ &\{u\} = \{u_1 \;\; u_2\} \end{aligned} \tag{1.3}$$

where N_i are the so called *shape functions*

$$\begin{aligned} N_1 &= 1 - \frac{x - x_1}{x_2 - x_1} \\ N_2 &= \frac{x - x_1}{x_2 - x_1} \end{aligned} \tag{1.4}$$

which are used for interpolation of $u(x)$ using its nodal values. Nodal values u_1 and u_2 are unknowns which should be determined from the discrete global equation system.

After substituting u expressed through its nodal values and shape functions, in the differential equation, it has the following approximate form:

$$a\frac{d^2}{dx^2}[N]\{u\} + b = \psi \tag{1.5}$$

where ψ is a nonzero residual because of approximate representation of a function inside a finite element. The Galerkin method provides residual minimization by multiplying terms of the above equation by shape functions, integrating over the element and equating to zero:

$$\int_{x_1}^{x_2} [N]^T a\frac{d^2}{dx^2}[N]\{u\}dx + \int_{x_1}^{x_2} [N]^T b dx = 0 \tag{1.6}$$

Use of integration by parts leads to the following discrete form of the differential equation for the finite element:

$$\int_{x_1}^{x_2} \left[\frac{dN}{dx}\right]^T a\left[\frac{dN}{dx}\right]dx\{u\} - \int_{x_1}^{x_2} [N]^T b dx - \left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} a\frac{du}{dx}|_{x=x_2} + \left\{ \begin{array}{c} 1 \\ 0 \end{array} \right\} a\frac{du}{dx}|_{x=x_1} = 0 \tag{1.7}$$

Usually such relation for a finite element is presented as:

$$\begin{aligned} &[k]\{u\} = \{f\} \\ &[k] = \int_{x_1}^{x_2} \left[\frac{dN}{dx}\right]^T a\left[\frac{dN}{dx}\right]dx \\ &\{f\} = \int_{x_1}^{x_2} [N]^T b dx + \left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} a\frac{du}{dx}|_{x=x_2} - \left\{ \begin{array}{c} 1 \\ 0 \end{array} \right\} a\frac{du}{dx}|_{x=x_1} \end{aligned} \tag{1.8}$$

In solid mechanics $[k]$ is called *stiffness matrix* and $\{f\}$ is called *load vector*. In the considered simple case for two finite elements of length L stiffness matrices and the load vectors can be easily calculated:

$$\begin{aligned} &[k_1] = [k_2] = \tfrac{a}{L}\left[\begin{array}{cc} 1 & -1 \\ -1 & 1 \end{array} \right] \\ &\{f_1\} = \tfrac{bL}{2}\left\{ \begin{array}{c} 1 \\ 1 \end{array} \right\}, \quad \{f_2\} = \tfrac{bL}{2}\left\{ \begin{array}{c} 1 \\ 1 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ R \end{array} \right\} \end{aligned} \tag{1.9}$$

The above relations provide finite element equations for the two separate finite elements. A global equation system for the domain with 2 elements and 3 nodes can be obtained by an assembly of element equations. In our simple case it is clear that elements interact with each other at the node with global number 2. The assembled global equation system is:

$$\frac{a}{L}\left[\begin{array}{ccc} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{array} \right] \left\{ \begin{array}{c} u_1 \\ u_2 \\ u_3 \end{array} \right\} = \frac{bL}{2}\left\{ \begin{array}{c} 1 \\ 2 \\ 1 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 0 \\ R \end{array} \right\} \tag{1.10}$$

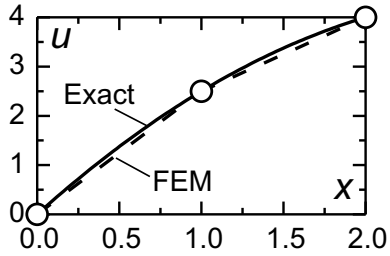


Figure 1.2: Comparison of finite element solution and exact solution.

After application of the boundary condition $u(x=0)=0$ the final appearance of the global equation system is:

$$\frac{a}{L}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \frac{bL}{2}\begin{Bmatrix} 0 \\ 2 \\ 1 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ R \end{Bmatrix} \tag{1.11}$$

Nodal values u_i are obtained as results of solution of linear algebraic equation system. The value of u at any point inside a finite element can be calculated using the shape functions. The finite element solution of the differential equation is shown in Fig. 1.2 for $a=1, b=1, L=1$ and $R=1$.

Exact solution is a quadratic function. The finite element solution with the use of the simplest element is piece-wise linear. More precise finite element solution can be obtained increasing the number of simple elements or with the use of elements with more complicated shape functions. It is worth noting that at nodes the finite element method provides exact values of u (just for this particular problem). Finite elements with linear shape functions produce exact nodal values if the sought solution is quadratic. Quadratic elements give exact nodal values for the cubic solution *etc.*

1.3.2 Variational formulation

The differential equation

$$\begin{aligned} &a\frac{d^2u}{dx^2} + b = 0, \qquad 0 \le x \le 2L \\ &u|_{x=0} = 0 \\ &a\frac{du}{dx}|_{x=2L} = R \end{aligned} \tag{1.12}$$

with $a = EA$ has the following physical meaning in solid mechanics. It describes tension of the one dimensional bar with cross-sectional area A made of material with the elasticity modulus E and subjected to a distributed load b and a concentrated load R at its right end as shown in Fig 1.3.
Such problem can be formulated in terms of minimizing the potential energy functional Π:

$$\begin{aligned} &\Pi = \int_L \frac{1}{2}a\left(\frac{du}{dx}\right)^2 dx - \int_L budx - Ru|_{x=2L} \\ &u|_{x=0} = 0 \end{aligned} \tag{1.13}$$

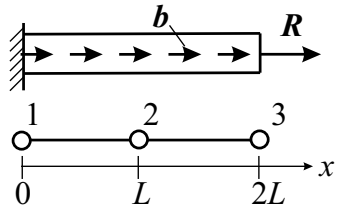


Figure 1.3: Tension of the one dimensional bar subjected to a distributed load and a concentrated load.

Using representation of $\{u\}$ with shape functions (1.3)-(1.4) we can write the value of potential energy for the second finite element as:

$$\begin{aligned}\Pi_e = &\int_{x_1}^{x_2} \frac{1}{2} a\{u\}^T \left[\frac{dN}{dx}\right]^T \left[\frac{dN}{dx}\right]\{u\}dx \\ &-\int_{x_1}^{x_2} \{u\}^T [N]^T b dx - \{u\}^T \left\{ \begin{array}{c} 0 \\ R \end{array} \right\}\end{aligned} \tag{1.14}$$

The condition for the minimum of Π is:

$$\delta\Pi = \frac{\partial \Pi}{\partial u_1}\delta u_1 + ... + \frac{\partial \Pi}{\partial u_n}\delta u_n = 0 \tag{1.15}$$

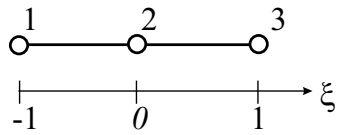
which is equivalent to

$$\frac{\partial \Pi}{\partial u_i} = 0\ , \quad i = 1...n \tag{1.16}$$

It is easy to check that differentiation of Π in respect to u_i gives the finite element equilibrium equation which is coincide with equation obtained by the Galerkin method:

$$\int_{x_1}^{x_2} \left[\frac{dN}{dx}\right]^T EA \left[\frac{dN}{dx}\right] dx\{u\} - \int_{x_1}^{x_2} [N]^T b dx - \left\{ \begin{array}{c} 0 \\ R \end{array} \right\} = 0 \tag{1.17}$$

Example. Obtain shape functions for the one-dimensional quadratic element with three nodes. Use local coordinate system $-1 \leq \xi \leq 1$.



Solution. With shape functions, any field inside element is presented as:

$$u(\xi) = \sum N_i u_i\ , \quad i = 1,\ 2,\ 3$$

At nodes the approximated function should be equal to its nodal value:

$$\begin{aligned} &u(-1) = u_1 \\ &u(0) = u_2 \\ &u(1) = u_3 \end{aligned}$$

Since the element has three nodes the shape functions can be quadratic polynomials (with three coefficients). The shape function N_1 can be written as:

$$N_1 = \alpha_1 + \alpha_2\xi + \alpha_3\xi^2$$

Unknown coefficients α_i are defined from the following system of equations:

$$\begin{aligned} &N_1(-1) = \alpha_1 - \alpha_2 + \alpha_3 = 1 \\ &N_1(0) = \alpha_1 = 0 \\ &N_1(1) = \alpha_1 + \alpha_2 + \alpha_3 = 0 \end{aligned}$$

The solution is: $\alpha_1 = 0, \;\; \alpha_2 = -1/2, \;\; \alpha_3 = 1/2$. Thus the shape function N_1 is equal to:

$$N_1 = -\frac{1}{2}\xi(1-\xi)$$

Similarly it is possible to obtain that the shape functions N_2 and N_3 are equal to:

$$\begin{aligned} N_2 &= 1 - \xi^2 \\ N_3 &= \frac{1}{2}\xi(1+\xi) \end{aligned}$$

Chapter 2

Finite Element Equations for Heat Transfer

2.1 Problem Statement

Let us consider an isotropic body with temperature dependent heat transfer. A basic equation of heat transfer has the following appearance:

$$-\left(\frac{\partial q_x}{\partial x}+\frac{\partial q_y}{\partial y}+\frac{\partial q_z}{\partial z}\right)+Q=\rho c\frac{\partial T}{\partial t} \tag{2.1}$$

Here q_x, q_y and q_z are components of heat flow through the unit area; $Q=Q(x,y,z,t)$ is the inner heat generation rate per unit volume; ρ is material density; c is heat capacity; T is temperature and t is time. According to the Fourier's law the components of heat flow can be expressed as follows:

$$\begin{aligned} q_x &= -k\frac{\partial T}{\partial x} \\ q_y &= -k\frac{\partial T}{\partial y} \\ q_z &= -k\frac{\partial T}{\partial z} \end{aligned} \tag{2.2}$$

where k is the thermal conductivity coefficient of the media. Substitution of Fourier's relations gives the following basic heat transfer equation:

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)+Q=\rho c\frac{\partial T}{\partial t} \tag{2.3}$$

It is assumed that *boundary conditions* can be of the following types:

1. Specified temperature

   $T_s=T(x,y,z,t)$ on S_1

2. Specified heat flow

   $q_xn_x+q_yn_y+q_zn_z=-q_s$ on S_2

3. Convection boundary conditions

$$q_x n_x + q_y n_y + q_z n_z = h(T_s - T_e) \text{ on } S_3$$

where h is the convection coefficient, T_s is an unknown surface temperature and T_e is a known environmental temperature.

4. Radiation

$$q_x n_x + q_y n_y + q_z n_z = \sigma \varepsilon T_s^4 - \alpha q_r \text{ on } S_4$$

where σ is the Stefan-Boltzmann constant; ε is the surface emission coefficient; α is the surface absorption coefficient and q_r is incoming heat flow per unit surface area.

For transient problems it is necessary to specify a temperature field for a body at the time t = 0:

$$T(x, y, z, 0) = T_0(x, y, z) \tag{2.4}$$

2.2 Finite element discretization of heat transfer equations

A domain V is divided into finite elements connected at nodes. We are going to write all relations for a finite element. Global equations for the domain can be assembled from finite element equations using connectivity information.

Shape functions N_i are used for interpolation of temperature inside a finite element:

$$\begin{array}{l} T = [N]\{T\} \\ [N] = [\ N_1 \quad N_2 \quad ...\ ] \\ \{T\} = \{\ T_1 \quad T_2 \quad ...\ \} \end{array} \tag{2.5}$$

Differentiation of the temperature interpolation equation gives the following interpolation relation for temperature gradients:

$$\left\{ \begin{array}{c} \partial T/\partial x \\ \partial T/\partial y \\ \partial T/\partial z \end{array} \right\} = \left[\begin{array}{ccc} \partial N_1/\partial x & \partial N_2/\partial x & ... \\ \partial N_1/\partial y & \partial N_2/\partial y & ... \\ \partial N_1/\partial z & \partial N_2/\partial z & ... \end{array} \right] \{T\} = [B]\{T\} \tag{2.6}$$

Here $\{T\}$ is a vector of temperatures at nodes; $[N]$ is a matrix of shape functions and $[B]$ is a matrix for temperature gradients interpolation.

Using Galerkin method, we can rewrite the basic heat transfer equation in the following form:

$$\int\limits_V \left(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z} - Q + \rho c \frac{\partial T}{\partial t} \right) N_i dV = 0 \tag{2.7}$$

Applying the divergence theorem to the first three terms, we arrive to the relations:

$$\begin{array}{l} \int\limits_V \rho c \frac{\partial T}{\partial t} N_i dV - \int\limits_V \left[\begin{array}{ccc} \frac{\partial N_i}{\partial x} & \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial z} \end{array} \right] \{q\} dV \\ = \int\limits_V Q N_i dV - \int\limits_S \{q\}^T \{n\} N_i dS \\ \{q\}^T = [\ q_x \quad q_y \quad q_z\ ] \\ \{n\}^T = [\ n_x \quad n_y \quad n_z\ ] \end{array} \tag{2.8}$$

where $\{n\}$ is an outer normal to the surface of the body. After insertion of boundary conditions into the above equation, the discretized equations are as follows:

$$\begin{aligned}&\int_V \rho c \frac{\partial T}{\partial t} N_i dV - \int_V \left[\begin{array}{ccc} \frac{\partial N_i}{\partial x} & \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial z} \end{array}\right] \{q\} dV \\ &= \int_V Q N_i dV - \int_{S_1} \{q\}^T \{n\} N_i dS \\ &+ \int_{S_2} q_s N_i dS - \int_{S_3} h(T - T_e) N_i dS - \int_{S_4} (\sigma \varepsilon T^4 - \alpha q_r) N_i dS\end{aligned} \tag{2.9}$$

It is worth noting that

$$\{q\} = -k[B]\{T\} \tag{2.10}$$

The discretized finite element equations for heat transfer problems have the following finite form:

$$\begin{aligned}&[C]\{\dot{T}\} + ([K_c] + [K_h] + [K_r])\{T\} \\ &= \{R_T\} + \{R_Q\} + \{R_q\} + \{R_h\} + \{R_r\}\end{aligned} \tag{2.11}$$

$$\begin{aligned}[C] &= \int_V \rho c [N]^T [N] dV \\ [K_c] &= \int_V k [B]^T [B] dV \\ [K_h] &= \int_{S_3} h [N]^T [N] dS \\ [K_r]\{T\} &= \int_{S_4} \sigma \varepsilon T^4 [N] dS \\ [R_T] &= -\int_{S_1} \{q\}^T \{n\} [N]^T dS \\ [R_Q] &= \int_V Q [N]^T dV \\ [R_q] &= -\int_{S_2} q_s [N]^T dS \\ [R_h] &= -\int_{S_3} h T_e [N]^T dS \\ [R_r] &= -\int_{S_4} \alpha q_r [N]^T dS\end{aligned} \tag{2.12}$$

2.3 Different Type Problems

Equations for different type problems can be deducted from the above general equation :

Stationary linear problem

$$([K_c] + [K_h])\{T\} = \{R_Q\} + \{R_q\} + \{R_h\} \tag{2.13}$$

Stationary nonlinear problem

$$([K_c] + [K_h] + [K_r])\{T\} = \{R_Q(T)\} + \{R_q(T)\} + \{R_h(T)\} + \{R_r(T)\} \tag{2.14}$$

Transient linear problem

$$[C]\{\dot{T}(t)\} + ([K_c] + [K_h(t)])\{T(t)\} = \{R_Q(t)\} + \{R_q(t)\} + \{R_h(t)\} \tag{2.15}$$

Transient nonlinear problem

$$[C(T)]\{\dot{T}\} + ([K_c(T)] + [K_h(T,t)] + [K_r(T)])\{T\} = \{R_Q(T,t)\} + \{R_q(T,t)\} + \{R_h(T,t)\} + \{R_r(T,t)\} \tag{2.16}$$

Chapter 3

FEM for Solid Mechanics Problems

3.1 Problem statement

Let us consider a three-dimensional elastic body subjected to surface and body forces and temperature field. In addition, displacements are specified on some surface area. For given geometry of the body, applied loads, displacement boundary conditions, temperature field and material stress-strain law, it is necessary to determine the displacement field for the body. The corresponding strains and stresses are also of interest.

The displacements along coordinate axes x, y and z are defined by the displacement vector $\{u\}$:

$$\{u\} = \{u \;\; v \;\; w\} \tag{3.1}$$

Six different strain components can be placed in the strain vector $\{\varepsilon\}$:

$$\{\varepsilon\} = \{\varepsilon_x \;\; \varepsilon_y \;\; \varepsilon_z \;\; \gamma_{xy} \;\; \gamma_{yz} \;\; \gamma_{zx}\} \tag{3.2}$$

For small strains the relationship between strains and displacements is:

$$\{\varepsilon\} = [D]\{u\} \tag{3.3}$$

where $[D]$ is the matrix differentiation operator:

$$[D] = \begin{bmatrix} \partial/\partial x & 0 & 0 \\ 0 & \partial/\partial y & 0 \\ 0 & 0 & \partial/\partial z \\ \partial/\partial y & \partial/\partial x & 0 \\ 0 & \partial/\partial z & \partial/\partial y \\ \partial/\partial z & 0 & \partial/\partial x \end{bmatrix} \tag{3.4}$$

Six different stress components are formed the stress vector $\{\sigma\}$:

$$\{\sigma\} = \{\sigma_x \;\; \sigma_y \;\; \sigma_z \;\; \tau_{xy} \;\; \tau_{yz} \;\; \tau_{zx}\} \tag{3.5}$$

which are related to strains for elastic body by the Hook's law:

$$\begin{aligned} &\{\sigma\} = [E]\{\varepsilon^e\} = [E](\{\varepsilon\} - \{\varepsilon^t\}) \\ &\{\varepsilon^t\} = \{\alpha T \;\; \alpha T \;\; \alpha T \;\; 0 \;\; 0 \;\; 0\} \end{aligned} \tag{3.6}$$

Here $\{\varepsilon^e\}$ is the elastic part of strains; $\{\varepsilon^t\}$ is the thermal part of strains; α is the coefficient of thermal expansion; T is temperature. The elasticity matrix $[E]$ has the following appearance:

$$[E] = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{3.7}$$

where λ and μ are elastic Lame constants which can be expressed through the elasticity modulus E and Poisson's ratio ν:

$$\begin{aligned} \lambda &= \frac{\nu E}{(1+\nu)(1-2\nu)} \\ \mu &= \frac{E}{2(1+\nu)} \end{aligned} \tag{3.8}$$

The purpose of finite element solution of elastic problem is to find such displacement field which provides minimum to the functional of total potential energy Π:

$$\Pi = \int_V \frac{1}{2}\{\varepsilon^e\}^T\{\sigma\}dv - \int_V \{u\}^T\{p^V\}dV - \int_S \{u\}^T\{p^S\}dS \tag{3.9}$$

Here $\{p^V\} = \{p_x^V \;\; p_y^V \;\; p_z^V\}$ is the vector of body force and $\{p^S\} = \{p_x^S \;\; p_y^S \;\; p_z^S\}$ is the vector of surface force. Prescribed displacements are specified on the part of body surface where surface forces are absent.

Displacement boundary conditions are not present in the functional of Π. Because of these, displacement boundary conditions should be implemented after assembly of finite element equations.

3.2 Finite element equations

Let us consider some abstract three-dimensional finite element having the vector of nodal displacements $\{q\}$:

$$\{q\} = \{u_1 \;\; v_1 \;\; w_1 \;\; u_2 \;\; v_2 \;\; w_2 \;\; ...\} \tag{3.10}$$

Displacements at some point inside a finite element $\{u\}$ can be determined with the use of nodal displacements $\{q\}$ and shape functions N_i:

$$\begin{aligned} u &= \sum N_i u_i \\ v &= \sum N_i v_i \\ w &= \sum N_i w_i \end{aligned} \tag{3.11}$$

These relations can be rewritten in a matrix form as follows:

$$\begin{aligned} \{u\} &= [N]\{q\} \\ [N] &= \begin{bmatrix} N_1 & 0 & 0 & N_2 & ... \\ 0 & N_1 & 0 & 0 & ... \\ 0 & 0 & N_1 & 0 & ... \end{bmatrix} \end{aligned} \tag{3.12}$$

Strains can also be determined through displacements at nodal points:

$$\begin{array}{l} \{\varepsilon\} = [B]\{q\} \\ [B] = [D][N] = [B_1 \;\; B_2 \;\; B_3 \;\; ...] \end{array} \tag{3.13}$$

Matrix $[B]$ is called the displacement differentiation matrix. It can be obtained by differentiation of displacements expressed through shape functions and nodal displacements:

$$[B_i] = \begin{bmatrix} \partial N_i/\partial x & 0 & 0 \\ 0 & \partial N_i/\partial y & 0 \\ 0 & 0 & \partial N_i/\partial z \\ \partial N_i/\partial y & \partial N_i/\partial x & 0 \\ 0 & \partial N_i/\partial z & \partial N_i/\partial y \\ \partial N_i/\partial z & 0 & \partial N_i/\partial x \end{bmatrix} \tag{3.14}$$

Now using relations for stresses and strains we are able to express the total potential energy through nodal displacements:

$$\begin{array}{l} \Pi = \int_V \frac{1}{2}([B]\{q\} - \{\varepsilon^t\})^T[E]([B]\{q\} - \{\varepsilon^t\})dV \\ -\int_V ([N]\{q\})^T\{p^V\}dV - \int_S ([N]\{q\})^T\{p^S\}dS \end{array} \tag{3.15}$$

Nodal displacements $\{q\}$ which correspond to the minimum of the functional Π are determined by the conditions:

$$\left\{\frac{\partial \Pi}{\partial q}\right\} = 0 \tag{3.16}$$

Differentiation of Π in respect to nodal displacements $\{q\}$ produces the following equilibrium equations for a finite element:

$$\begin{array}{l} \int_V [B]^T[E][B]dV\{q\} - \int_V [B]^T[E]\{\varepsilon^t\}dV \\ -\int_V [N]^T\{p^V\}dV - \int_S [N]^T\{p^S\}dS = 0 \end{array} \tag{3.17}$$

which is usually presented in the following form:

$$\begin{array}{l} [k]\{q\} = \{f\} \\ \{f\} = \{p\} + \{h\} \\ \\ [k] = \int_V [B]^T[E][B]dV \\ \{p\} = \int_V [N]^T\{p^V\}dV + \int_S [N]^T\{p^S\}dS \\ \{h\} = \int_V [B]^T[E]\{\varepsilon^t\}dV \end{array} \tag{3.18}$$

Here $[k]$ is the element stiffness matrix; $\{f\}$ is the load vector; $\{p\}$ is the vector of actual forces and $\{h\}$ is the thermal vector which represents fictitious forces for modeling thermal expansion.

3.3 Assembly of the global equation system

The aim of assembly is to form the global equation system

$$[K]\{Q\} = \{F\} \tag{3.19}$$

using element equations

$$[k_i]\{q_i\} = \{f_i\} \tag{3.20}$$

Here $[k_i]$, $[q_i]$ and $[f_i]$ are the stiffness matrix, the displacement vector and the load vector of the ith finite element; $[K]$, $\{Q\}$ and $\{F\}$ are global stiffness matrix, displacement vector and load vector.

In order to derive an assembly algorithm let us present the total potential energy for the body as a sum of element potential energies π_i:

$$\Pi = \sum \pi_i = \sum \frac{1}{2}\{q_i\}^T[k_i]\{q_i\} - \sum \{q_i\}^T\{f_i\} + \sum E_i^0 \tag{3.21}$$

where E_i^0 is the fraction of potential energy related to free thermal expansion:

$$E_i^0 = \int_{V_i} \frac{1}{2}\{\varepsilon^t\}^T[E]\{\varepsilon^t\}dV \tag{3.22}$$

Let us introduce the following vectors and a matrix where element vectors and matrices are simply placed:

$$\begin{array}{l} \{Q_d\} = \{\{q_1\}\ \{q_2\} \\ \{F_d\} = \{\{f_1\}\ \{f_2\}\ ...\} \end{array} \tag{3.23}$$

$$[K_d] = \begin{bmatrix} [k_1] & 0 & 0 \\ 0 & [k_2] & 0 \\ 0 & 0 & ... \end{bmatrix} \tag{3.24}$$

It is evident that it is easy to find matrix $[A]$ such that

$$\begin{array}{l} \{Q_d\} = [A]\{Q\} \\ \{F_d\} = [A]\{F\} \end{array} \tag{3.25}$$

The total potential energy for the body can be rewritten in the following form:

$$\begin{array}{l} \Pi = \frac{1}{2}\{Q_d\}^T[K_d]\{Q_d\} - \{Q_d\}^T\{F_d\} + \sum E_i^0 \\ = \frac{1}{2}\{Q\}^T[A]^T[K_d][A]\{Q\} - \{Q\}^T[A]^T\{F_d\} + \sum E_i^0 \end{array} \tag{3.26}$$

Using the condition of minimum of the total potential energy

$$\left\{\frac{\partial \Pi}{\partial Q}\right\} = 0 \tag{3.27}$$

we arrive at the following global equation system:

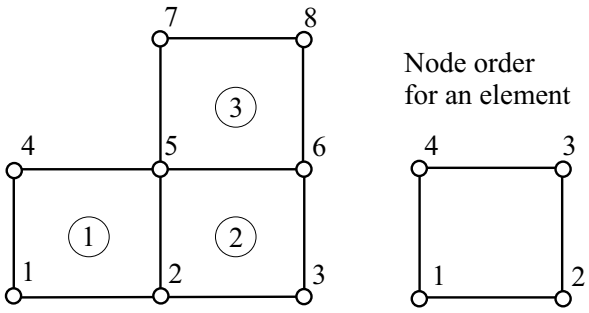
$$[A]^T[K_d][A]\{Q\} - [A]^T\{F_d\} = 0 \tag{3.28}$$

The last equation shows that algorithms of assembly the global stiffness matrix and the global load vector are:

$$\begin{aligned}[K] &= [A]^T[K_d][A] \\ \{F\} &= [A]^T\{F_d\}\end{aligned} \tag{3.29}$$

Here $[A]$ is the matrix providing transformation from global to local enumeration. Fraction of nonzero (unit) entries in the matrix $[A]$ is very small. Because of this the matrix $[A]$ is never used explicitly in actual computer codes.

Example. Write down a matrix $[A]$, which relates local (element) and global (domain) node numbers for the following finite element mesh:



Solution. To make the matrix representation compact let us assume that each node has one degree of freedom (note that in three-dimensional solid mechanics problem there are three degrees of freedom at each node). The matrix $[A]$ relates element and global nodal values in the following way:

$$\{Q_d\} = [A]\{Q\}$$

where $\{Q\}$ is a global vector of nodal values and $\{Q_d\}$ is vector containing element vectors. The explicit rewriting of the above relation looks as follows:

$$\left\{\begin{Bmatrix} Q_1 \\ Q_2 \\ Q_5 \\ Q_4 \end{Bmatrix} \begin{Bmatrix} Q_2 \\ Q_3 \\ Q_6 \\ Q_5 \end{Bmatrix} \begin{Bmatrix} Q_5 \\ Q_6 \\ Q_8 \\ Q_7 \end{Bmatrix}\right\} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ Q_5 \\ Q_6 \\ Q_7 \\ Q_8 \end{Bmatrix}$$

Chapter 4

Finite Elements

4.1 Two-dimensional triangular element

Triangular finite element was the first finite element proposed for continuous problems. Because of simplicity it can be used as an introduction to other elements. A triangular finite element in the coordinate system xy is shown in Fig. 4.1. Since the element has three nodes, linear approximation of displacements u and v is selected:

$$\begin{aligned} u(x,y) &= N_1 u_1 + N_2 u_2 + N_3 u_3 \\ v(x,y) &= N_1 v_1 + N_2 v_2 + N_3 v_3 \\ N_i &= \alpha_i + \beta_i x + \gamma_i y \end{aligned} \tag{4.1}$$

Shape functions $N_i(x,y)$ can be determined from the following equation system:

$$u(x_i, y_i) = u_i \ , \quad i = 1,\ 2,\ 3 \tag{4.2}$$

Shape functions for the triangular element can be presented as:

$$\begin{aligned} N_i &= \frac{1}{2\Delta}(a_i + b_i x + c_i y) \\ a_i &= x_{i+1} y_{i+2} - x_{i+2} y_{i+1} \\ b_i &= y_{i+1} - y_{i+2} \\ c_i &= x_{i+2} - x_{i+1} \\ \Delta &= \frac{1}{2}(x_2 y_3 + x_3 y_1 + x_1 y_2 - x_2 y_1 - x_3 y_2 - x_1 y_3) \end{aligned} \tag{4.3}$$

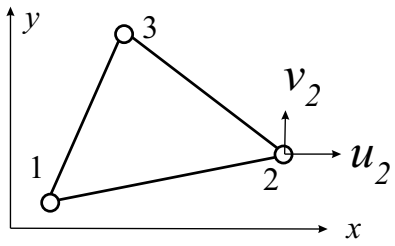


Figure 4.1: Triangular finite element is the simplest two-dimensional element.

where Δ is the element area. The matrix $[B]$ for interpolating strains using nodal displacements is equal to:

$$[B] = \frac{1}{2\Delta}\begin{bmatrix} b_1 & 0 & b_2 & 0 & b_3 & 0 \\ 0 & c_1 & 0 & c_2 & 0 & c_3 \\ c_1 & b_1 & c_2 & b_2 & c_3 & b_3 \end{bmatrix} \tag{4.4}$$

The elasticity matrix $[E]$ has the following appearance for plane problems:

$$[E] = \begin{bmatrix} \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda + 2\mu & 0 \\ 0 & 0 & \mu \end{bmatrix} \tag{4.5}$$

where λ and μ are Lame constants:

$$\begin{aligned} \lambda &= \frac{\nu E}{(1+\nu)(1-2\nu)} && \text{for plane strain} \\ \lambda &= \frac{\nu E}{1-\nu^2} && \text{for plane stress} \\ \mu &= \frac{E}{2(1+\nu)} \end{aligned} \tag{4.6}$$

Here E is the elasticity modulus and ν is the Poisson's ratio.

The stiffness matrix for the three-node triangular element can be calculated as:

$$[k] = \int_V [B]^T[E][B]dV = [B]^T[E][B]\Delta \tag{4.7}$$

Here it was taken into account that both matrices $[B]$ and $[E]$ do not depend on coordinates. It was assumed that the element has unit thickness. Since the matrix $[B]$ is constant inside the element the strains and stresses are also constant inside the triangular element.

4.2 Two-dimensional isoparametric elements

Isoparametric finite elements are based on the parametric definition of both coordinate and displacement functions. The same shape functions are used for specification of the element shape and for interpolation of the displacement field.

4.2.1 Shape functions

Linear and quadratic two-dimensional isoparametric finite elements are presented in Figure 4.2. Shape functions N_i are defined in local coordinates $\xi,\ \eta$ $(-1 \leq \xi,\ \eta \leq 1)$. The same shape functions are used for interpolations of displacements and coordinates:

$$\begin{aligned} u &= \textstyle\sum N_i u_i\ , \quad & v &= \textstyle\sum N_i v_i \\ x &= \textstyle\sum N_i x_i\ , \quad & y &= \textstyle\sum N_i y_i \end{aligned} \tag{4.8}$$

where u, v are displacement components at point with local coordinates (ξ, η); u_i, v_i are displacement values at the nodes of the finite element; x, y are point coordinates and x_i, y_i are coordinates of element nodes. Matrix form of the above relations is as follows:

$$\begin{aligned} \{u\} &= [N]\{q\} \\ \{u\} &= \{u\ \ v\} \\ \{q\} &= \{u_1\ v_1\ u_2\ v_2\ ...\} \end{aligned} \tag{4.9}$$

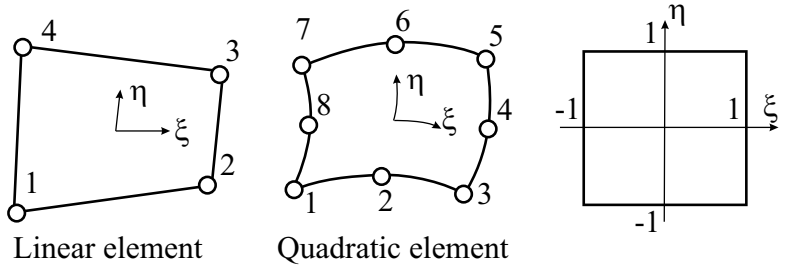


Figure 4.2: Linear and quadratic finite elements and their representation in the local coordinate system.

$$\begin{aligned} &\{x\} = [N]\{x^e\} \\ &\{x\} = \{x \;\; y\} \\ &\{x^e\} = \{x_1 \; y_1 \; x_2 \; y_2 \; ...\} \end{aligned} \tag{4.10}$$

where the interpolation matrix for nodal values is:

$$[N] = \begin{bmatrix} N_1 & 0 & N_2 & 0 & ... \\ 0 & N_1 & 0 & N_2 & ... \end{bmatrix} \tag{4.11}$$

Shape functions for linear and quadratic two-dimensional isoparametric elements are given by:
linear element (4 nodes):

$$N_i = \frac{1}{4}(1+\xi_0)(1+\eta_0) \tag{4.12}$$

quadratic element (8 nodes):

$$\begin{aligned} N_i &= \frac{1}{4}(1+\xi_0)(1+\eta_0) - \frac{1}{4}(1-\xi^2)(1+\eta_0) \\ &\quad -\frac{1}{4}(1+\xi_0)(1-\eta^2) \;\; i = 1,\; 3,\; 5,\; 7 \\ N_i &= \frac{1}{2}(1-\xi^2)(1+\eta_0)\;, \;\; i = 2,\; 6 \\ N_i &= \frac{1}{2}(1+\xi_0)(1-\eta^2)\;, \;\; i = 4,\; 8 \end{aligned} \tag{4.13}$$

In the above equations the following notation is used: $\xi_0 = \xi\xi_i \;,\; \eta_0 = \eta\eta_i$ where ξ_i, η_i are values of local coordinates ξ, η at nodes.

4.2.2 Strain-displacement matrix

For plane problem the strain vector contains three components:

$$\{\varepsilon\} = \{\varepsilon_x \;\; \varepsilon_y \;\; \gamma_{xy}\} = \left\{ \frac{\partial u}{\partial x} \quad \frac{\partial v}{\partial y} \quad \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right\} \tag{4.14}$$

The strain-displacement matrix which is employed to compute strains at any point inside the element using nodal displacements is:

$$[B] = [B_1\ B_2\ ...]$$
$$[B_i] = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 \\ 0 & \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} \end{bmatrix} \tag{4.15}$$

While shape functions are expressed through the local coordinates ξ, η the strain-displacement matrix contains derivatives in respect to the global coordinates x, y. Derivatives can be easily converted from one coordinate system to the other by means of the chain rule of partial differentiation:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{Bmatrix} \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = [J] \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} \tag{4.16}$$

where $[J]$ is the Jacobian matrix. The derivatives in respect to the global coordinates are computed with the use of inverse of the Jacobian matrix:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} \tag{4.17}$$

The components of the Jacobian matrix are calculated using derivatives of shape functions N_i in respect to the local coordinates ξ, η and global coordinates of element nodes x_i, y_i:

$$\begin{aligned} \frac{\partial x}{\partial \xi} &= \sum \frac{\partial N_i}{\partial \xi} x_i\ , \quad \frac{\partial x}{\partial \eta} = \sum \frac{\partial N_i}{\partial \eta} x_i \\ \frac{\partial y}{\partial \xi} &= \sum \frac{\partial N_i}{\partial \xi} y_i\ , \quad \frac{\partial y}{\partial \eta} = \sum \frac{\partial N_i}{\partial \eta} y_i \end{aligned} \tag{4.18}$$

The determinant of the Jacobian matrix $|J|$ is used for the transformation of integrals from the global coordinate system to the local coordinate system:

$$dV = dxdy = |J| d\xi d\eta \tag{4.19}$$

4.2.3 Element properties

Element matrices and vectors are calculated as follows:

stiffness matrix

$$[k] = \int_V [B]^T [E][B] dV \tag{4.20}$$

force vector (volume and surface loads)

$$\{p\} = \int_V [N]^T \{p^V\} dV + \int_S [N]^T \{p^S\} dS \tag{4.21}$$

thermal vector (fictitious forces to simulate thermal expansion)

$$\{h\} = \int_V [B]^T [E]\{\varepsilon^t\} dV \tag{4.22}$$

The elasticity matrix $[E]$ is given by relation (4.5).

4.2.4 Integration in quadrilateral elements

Integration of expressions for stiffness matrices and load vectors can not be performed analytically for general case of isoparametric elements. Instead, stiffness matrices and load vectors are typically evaluated numerically using Gauss quadrature over quadrilateral regions. The Gauss quadrature formula for the volume integral in two-dimensional case is of the form:

$$I = \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta) d\xi d\eta = \sum_{i=1}^{n}\sum_{j=1}^{n} f(\xi_i,\eta_j) w_i w_j \tag{4.23}$$

where $\xi_i,\ \eta_j$ are abscissae and w_i are weighting coefficients of the Gauss integration rule. Abscissae and weights of Gauss quadrature for n = 1,2,3 are given in Table:

Abscissae and weights of Gauss quadrature

n	ξ_i	w_i
1	0	2
2	$\mp 1/\sqrt{3}$	1
3	0	8/9
	$\mp\sqrt{3/5}$	5/9

To compute the nodal equivalent of the surface load, the surface integral is replaced by line integration along element side. The fraction of the surface load is evaluated as:

$$\{p\} = \int_S [N]^T \{p^S\} dS = \int_{-1}^{1} [N]^T \{p^S\} \frac{ds}{d\xi} d\xi \tag{4.24}$$

$$\frac{ds}{d\xi} = \sqrt{\left(\frac{dx}{d\xi}\right)^2 + \left(\frac{dy}{d\xi}\right)^2} \tag{4.25}$$

Here s is a global coordinate along the element side and ξ is a local coordinate along the element side. If the distributed load load is applied along the normal to the element side as shown in Fig. 4.3 then the nodal equivalent of such load is:

$$\{p\} = \int_S [N]^T p \begin{Bmatrix} \frac{dy}{ds} \\ -\frac{dx}{ds} \end{Bmatrix} \frac{ds}{d\xi} d\xi = \int_{-1}^{1} [N]^T p \begin{Bmatrix} \frac{dy}{d\xi} \\ -\frac{dx}{d\xi} \end{Bmatrix} d\xi \tag{4.26}$$

Example. Calculate nodal equivalents of a distributed load with constant intensity applied to the side of a two-dimensional quadratic element:

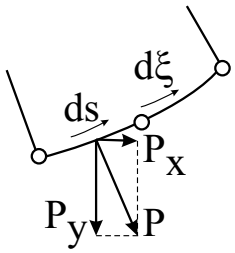
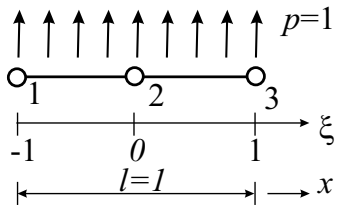


Figure 4.3: Distributed normal load on a side of a quadratic element.



Solution. The nodal equivalent of the distributed load is calculated as:

$$\{p\} = \int_{-1}^{1} [N]^T p \frac{dx}{d\xi} d\xi$$

or

$$\{p\} = \left\{ \begin{array}{c} p_1 \\ p_2 \\ p_3 \end{array} \right\} = \int_{-1}^{1} \left\{ \begin{array}{c} N_1 \\ N_2 \\ N_3 \end{array} \right\} p \frac{dx}{d\xi} d\xi, \quad \frac{dx}{d\xi} = \frac{1}{2}$$

The shape functions for the one-dimensional quadratic element are:

$$N_1 = -\frac{1}{2}\xi(1-\xi), \;\; N_2 = 1-\xi^2, \quad N_3 = \frac{1}{2}\xi(1+\xi)$$

The values of nodal forces at nodes 1, 2 and 3 are defined by integration:

$$\begin{aligned} p_1 &= -\int_{-1}^{1} \frac{1}{2}\xi(1-\xi)\frac{1}{2} d\xi = \frac{1}{6} \\ p_2 &= \int_{-1}^{1} (1-\xi^2)\frac{1}{2} d\xi = \frac{2}{3} \\ p_3 &= \int_{-1}^{1} \frac{1}{2}\xi(1+\xi)\frac{1}{2} d\xi = \frac{1}{6} \end{aligned}$$

The example shows that physical approach cannot be used for the estimation of nodal equivalents of a distributed load. It works for linear elements; however, it does not work for more complicated elements.

4.2.5 Calculation of strains and stresses

Strains at any point an element are determined using Cauchy relations (4.14) with the use of the displacement differentiation matrix (4.15):

$$\{\varepsilon\} = [B]\{q\} \tag{4.27}$$

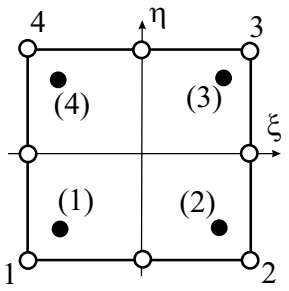


Figure 4.4: Numbering of integration points and vertices for the 8-node isoparametric element.

Stresses are calculated with the Hook's law:

$$\{\sigma\} = [E]\{\varepsilon^e\} = [E](\{\varepsilon\} - \{\varepsilon^t\}) \tag{4.28}$$

where $\{\varepsilon^t\}$ is the vector of free thermal expansion:

$$\{\varepsilon^t\} = \{\alpha T \;\; \alpha T \;\; 0\} \tag{4.29}$$

Precision of strains and stresses is significantly dependent on the point location where they are computed. The highest precision for displacement gradients are at the geometric center for the linear element and at reduced integration points 2×2 for the quadratic quadrilateral element.

For quadratic elements with 8 nodes, strains and stresses have best precision at 2×2 integration points with local coordinates $\xi,\ \eta = \pm 1/\sqrt{3}$. A possible way to create continuous stress field with reasonable accuracy consists of: 1) extrapolation of stresses from reduced integration points to nodes; 2) averaging contributions from finite elements at all nodes of the finite element model. Later stresses can be interpolated from nodes using quadratic shape functions.

Let us consider quadratic element in the local coordinate system $\xi,\ \eta$ as shown in Fig. 4.4 where integration points are numbered as (1)...(4); corner nodes have numbers 1...4. For extrapolation (inside quadratic element) we are going to employ linear shape functions. In matrix form the extrapolation relation can be presented as follows:

$$f_i = L_{i(m)} f_{(m)} \tag{4.30}$$

where $f_{(m)}$ are known function values at reduces integration points; f_i are function values at vertex nodes and $L_{i(m)}$ is the symmetric extrapolation matrix:

$$\begin{aligned} L_{i(m)} &= \begin{bmatrix} A & B & C & B \\ & A & B & C \\ & & A & B \\ & & & A \end{bmatrix} \\ A &= 1 + \frac{\sqrt{3}}{2} \\ B &= -\frac{1}{2} \\ C &= 1 + \frac{\sqrt{3}}{2} \end{aligned} \tag{4.31}$$

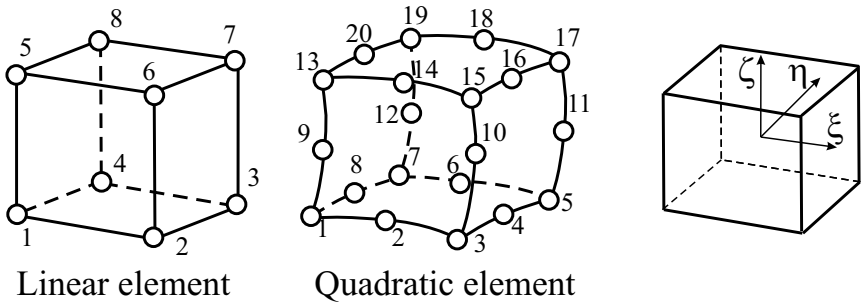


Figure 4.5: Linear and quadratic three-dimensional finite elements and their representation in the local coordinate system.

4.3 Three-dimensional isoparametric elements

4.3.1 Shape functions

Hexahedral (or brick-type) linear 8-node and quadratic 20-node three-dimensional isoparametric elements are depicted in Fig. 4.5. The term "isoparametric" means that geometry and displacement field are specified in parametric form and are interpolated with the same functions. Shape functions used for interpolation are polynomials of the local coordinates ξ, η and ζ ($-1 \leq \xi, \eta, \zeta \leq 1$). Both coordinates and displacements are interpolated with the same shape functions:

$$\begin{aligned} &\{u\} = [N]\{q\} \\ &\{u\} = \{u \;\; v \;\; w\} \\ &\{q\} = \{u_1 \; v_1 \; w_1 \; u_2 \; v_2 \; w_2 \; ...\} \end{aligned} \tag{4.32}$$

$$\begin{aligned} &\{x\} = [N]\{x^e\} \\ &\{x\} = \{x \;\; y \;\; z\} \\ &\{x^e\} = \{x_1 \; y_1 \; z_1 \; x_2 \; y_2 \; z_2 \; ...\} \end{aligned} \tag{4.33}$$

Here u, v, w are displacements at point with local coordinates ξ, η, ζ; u_i, v_i, w_i are displacement values at nodes; x, y, z are point coordinates and x_i, y_i, z_i are coordinates of nodes. The matrix of shape functions is:

$$[N] = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & ... \\ 0 & N_1 & 0 & 0 & N_2 & 0 & ... \\ 0 & 0 & N_1 & 0 & 0 & N_2 & ... \end{bmatrix} \tag{4.34}$$

Shape functions of the linear element are equal to:

$$\begin{aligned} &N_i = \frac{1}{8}(1 + \xi_0)(1 + \eta_0)(1 + \varsigma_0) \\ &\xi_0 = \xi\xi_i \;, \;\; \eta_0 = \eta\eta_i \;, \;\; \varsigma_0 = \varsigma\varsigma_i \end{aligned} \tag{4.35}$$

For the quadratic element with 20 nodes the shape functions can be written in the following form:

$$\begin{aligned}
N_i &= \frac{1}{8}(1+\xi_0)(1+\eta_0)(1+\varsigma_0)(\xi_0+\eta_0+\varsigma_0-2) \ \text{ vertices} \\
N_i &= \frac{1}{4}(1-\xi^2)(1+\eta_0)(1+\varsigma_0)\ , \quad i=2,\ 6,\ 14,\ 18 \\
N_i &= \frac{1}{4}(1-\eta^2)(1+\xi_0)(1+\varsigma_0)\ , \quad i=4,\ 8,\ 16,\ 20 \\
N_i &= \frac{1}{4}(1-\varsigma^2)(1+\xi_0)(1+\eta_0)\ , \quad i=9,\ 10,\ 11,\ 12
\end{aligned} \tag{4.36}$$

In the above relations ξ_i, η_i, ζ_i are values of local coordinates ξ, η, ζ at nodes.

4.3.2 Strain-displacement matrix

The strain vector $\{\varepsilon\}$ contains six different components of the strain tensor:

$$\{\varepsilon\} = \{\varepsilon_x \ \ \varepsilon_y \ \ \varepsilon_z \ \ \gamma_{xy} \ \ \gamma_{yz} \ \ \gamma_{zx}\} \tag{4.37}$$

The strain-displacement matrix for three-dimensional elements has the following appearance:

$$[B] = [D][N] = [B_1 \ \ B_2 \ \ B_3 \ \ ...] \tag{4.38}$$

$$[B_i] = \begin{bmatrix} \partial N_i/\partial x & 0 & 0 \\ 0 & \partial N_i/\partial y & 0 \\ 0 & 0 & \partial N_i/\partial z \\ \partial N_i/\partial y & \partial N_i/\partial x & 0 \\ 0 & \partial N_i/\partial z & \partial N_i/\partial y \\ \partial N_i/\partial z & 0 & \partial N_i/\partial x \end{bmatrix} \tag{4.39}$$

Derivatives of shape functions in respect to global coordinates are obtained as follows:

$$\begin{Bmatrix} \partial N_i/\partial x \\ \partial N_i/\partial y \\ \partial N_i/\partial z \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \partial N_i/\partial \xi \\ \partial N_i/\partial \eta \\ \partial N_i/\partial \varsigma \end{Bmatrix} \tag{4.40}$$

where the Jacobian matrix has the appearance:

$$[J] = \begin{bmatrix} \partial x/\partial \xi & \partial y/\partial \xi & \partial z/\partial \xi \\ \partial x/\partial \eta & \partial y/\partial \eta & \partial z/\partial \eta \\ \partial x/\partial \varsigma & \partial y/\partial \varsigma & \partial z/\partial \varsigma \end{bmatrix} \tag{4.41}$$

The partial derivatives of x, y, z in respect to ξ, η,ζ are found by differentiation of displacements expressed through shape functions and nodal displacement values:

$$\begin{aligned}
\frac{\partial x}{\partial \xi} &= \sum \frac{\partial N_i}{\partial \xi} x_i\ , \quad \frac{\partial x}{\partial \eta} = \sum \frac{\partial N_i}{\partial \eta} x_i\ , \quad \frac{\partial x}{\partial \zeta} = \sum \frac{\partial N_i}{\partial \zeta} x_i \\
\frac{\partial y}{\partial \xi} &= \sum \frac{\partial N_i}{\partial \xi} y_i\ , \quad \frac{\partial y}{\partial \eta} = \sum \frac{\partial N_i}{\partial \eta} y_i\ , \quad \frac{\partial y}{\partial \zeta} = \sum \frac{\partial N_i}{\partial \zeta} y_i \\
\frac{\partial z}{\partial \xi} &= \sum \frac{\partial N_i}{\partial \xi} z_i\ , \quad \frac{\partial z}{\partial \eta} = \sum \frac{\partial N_i}{\partial \eta} z_i\ , \quad \frac{\partial z}{\partial \zeta} = \sum \frac{\partial N_i}{\partial \zeta} z_i
\end{aligned} \tag{4.42}$$

The transformation of integrals from the global coordinate system to the local coordinate system is performed with the use of determinant of the Jacobian matrix:

$$dV = dxdydz = |J| d\xi d\eta d\varsigma \tag{4.43}$$

4.3.3 Element properties

Element equilibrium equation has the following form:

$$\begin{aligned} &[k]\{q\} = \{f\} \\ &\{f\} = \{p\} + \{h\} \end{aligned} \tag{4.44}$$

Element matrices and vectors are:

stiffness matrix

$$[k] = \int_V [B]^T [E][B] dV \tag{4.45}$$

force vector (volume and surface loads)

$$\{p\} = \int_V [N]^T \{p^V\} dV + \int_S [N]^T \{p^S\} dS \tag{4.46}$$

thermal vector (fictitious forces to simulate thermal expansion)

$$\{h\} = \int_V [B]^T [E] \{\varepsilon^t\} dV \tag{4.47}$$

The elasticity matrix $[E]$ is:

$$[E] = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{4.48}$$

where λ and μ are elastic Lame constants which can be expressed through the elasticity modulus E and Poisson's ratio ν:

$$\begin{aligned} \lambda &= \frac{\nu E}{(1+\nu)(1-2\nu)} \\ \mu &= \frac{E}{2(1+\nu)} \end{aligned} \tag{4.49}$$

4.3.4 Efficient computation of the stiffness matrix

Calculation of the element stiffness matrix by multiplication of three matrices involves many arithmetic operations with zeros. After performing multiplications in closed form, coefficients of the element stiffness matrix $[k]$ can be expressed as follows:

$$\begin{aligned} k_{ii}^{mn} &= \int_V \left[\beta \frac{\partial N_m}{\partial x_i} \frac{\partial N_n}{\partial x_i} + \mu \left(\frac{\partial N_m}{\partial x_{i+1}} \frac{\partial N_m}{\partial x_{i+1}} + \frac{\partial N_m}{\partial x_{i+2}} \frac{\partial N_m}{\partial x_{i+2}} \right) \right] dV \\ k_{ij}^{mn} &= \int_V \left(\lambda \frac{\partial N_m}{\partial x_i} \frac{\partial N_n}{\partial x_j} + \mu \frac{\partial N_m}{\partial x_j} \frac{\partial N_n}{\partial x_i} \right) dV \\ \beta &= \lambda + 2\mu \end{aligned} \tag{4.50}$$

Here m, n are local node numbers; i, j are indices related to coordinate axes (x_1, x_2, x_3). Cyclic rule is employed in the above equation if coordinate indices become greater than 3.

4.3.5 Integration of the stiffness matrix

Integration of the stiffness matrix for three-dimensional isoparametric elements is carried out in the local coordinate system ξ, η, ζ:

$$[k] = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} [B(\xi,\eta,\varsigma)]^T [E][B(\xi,\eta,\varsigma)]|J| d\xi d\eta d\varsigma \tag{4.51}$$

Three-time application of the one-dimensional Gauss quadrature rule leads to the following numerical integration procedure:

$$\begin{aligned} I &= \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta,\varsigma) d\xi d\eta d\varsigma \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} f(\xi_i,\eta_j,\varsigma_k) w_i w_j w_k \end{aligned} \tag{4.52}$$

Usually $2 \times 2 \times 2$ integration is used for linear elements and integration $3 \times 3 \times 3$ is applied to the evaluation of the stiffness matrix for quadratic elements. Fore more efficient integration, a special 14-point Gauss-type rule exists, which provides sufficient precision of integration for three-dimensional quadratic elements.

4.3.6 Calculation of strains and stresses

After computing element matrices and vectors, the assembly process is used to compose the global equation system. Solution of the global equation system provides displacements at nodes of the finite element model. Using disassembly, nodal displacement for each element can be obtained.

Strains inside an element are determined with the use of the displacement differentiation matrix:

$$\{\varepsilon\} = [B]\{q\} \tag{4.53}$$

Stresses are calculated with the Hook's law:

$$\{\sigma\} = [E]\{\varepsilon^e\} = [E](\{\varepsilon\} - \{\varepsilon^t\}) \tag{4.54}$$

where $\{\varepsilon^t\}$ is the vector of free thermal expansion:

$$\{\varepsilon^t\} = \{\alpha T \;\; \alpha T \;\; \alpha T \;\; 0 \;\; 0 \;\; 0\} \tag{4.55}$$

It should be noted that displacement gradients (and hence strains and stresses) have quite difference precision at different point inside finite elements. The highest precision for displacement gradients are at the geometric center for the linear element and at reduced integration points $2\times2\times2$ for the quadratic hexagonal element.

4.3.7 Extrapolation of strains and stresses

For quadratic elements, displacement derivatives have best precision at $2\times2\times2$ integration points with local coordinates $\xi,\ \eta,\ \zeta = \pm 1/\sqrt{3}$. In order to build a continuous field of strains or stresses, it is necessary to extrapolate result values from $2 \times 2 \times 2$ integration points to vertices of 20-node element (numbering of integration points and vertices is shown in Fig. 4.6).

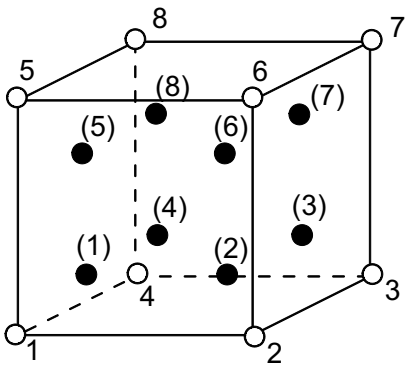


Figure 4.6: Numbering of integration points and vertices for the 20-node isoparametric element.

Results are calculated at 8 integration points, and a trilinear extrapolation in the local coordinate system $\xi,\ \eta,\ \zeta$ is used:

$$f_i = L_{i(m)} f_{(m)} \tag{4.56}$$

where $f_{(m)}$ are known function values at reduces integration points; f_i are function values at vertex nodes and $L_{i(m)}$ is the symmetric extrapolation matrix:

$$L_{i(m)} = \begin{bmatrix} A & B & C & B & B & C & D & C \\ & A & B & C & C & B & C & D \\ & & A & B & D & C & B & C \\ & & & A & C & D & C & B \\ & & & & A & B & C & B \\ & & & & & A & B & C \\ & & & & & & A & B \\ & & & & & & & A \end{bmatrix} \tag{4.57}$$

$$A = \frac{5+\sqrt{3}}{4}, \quad B = -\frac{\sqrt{3}+1}{4}$$
$$C = \frac{\sqrt{3}-1}{4}, \quad D = \frac{5-\sqrt{3}}{4}$$

Stresses are extrapolated from integration points to all nodes of elements. Values for midside nodes can be calculated as an average between values for two vertex nodal values. Then averaging of contributions from the neighboring finite elements is performed for all nodes of the finite element model. Averaging produces a continuous field of secondary results specified at nodes of the model with quadratic variation inside finite elements. Later, the results can be interpolated to any point inside element or on its surface using quadratic shape functions.

Chapter 5

Discretization

5.1 Discrete model of the problem

In order to apply finite element procedures a discrete model of the problem should be presented in numerical form. A typical description of the problem can contain:

Scalar parameters (number of nodes, number of elements etc.);

Material properties;

Coordinates of nodal points;

Connectivity array for finite elements;

Arrays of element types and element materials;

Arrays for description of displacement boundary conditions;

Arrays for description of surface and concentrated loads;

Temperature field.

Let us write down numerical information for a simple problem depicted in Fig. 5.1.

The finite element model can be described as follows:

1. Scalar parameters

```
Number of nodes       = 6
Number of elements    = 2
Number of constraints = 5
```

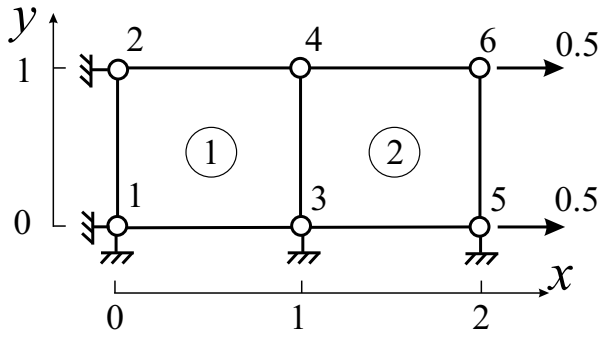


Figure 5.1: Discrete model composed of two finite elements.

```
Number of loads       = 2
```

2. Material properties

```
Elasticity modulus    = 2.0e+8 MPa
Poisson's ratio       = 0.3
```

3. Node coordinates (x_1, y_1, x_2, y_2 etc.)

```
1) 0 0  2) 0 1  3) 1 0  4) 1 1  5) 2 0  6) 2 1
```

4. Element connectivity array (counterclockwise direction)

```
1) 1 3 4 2   2) 3 5 6 4
```

5. Constraints (node, direction: $x = 1$; $y = 2$)

```
1 1  2 1  1 2  3 2  5 2
```

6. Nodal forces (node, direction, value)

```
5 1 0.5   6 1 0.5
```

While for simple example like the demonstrated above the finite element model can be coded by hand, it is not practical for real-life models. Various automatic mesh generators are used for creating finite element models for complex shapes.

5.2 Mesh generation

5.2.1 Mesh generators

The finite element models for practical analysis can contain tens of thousands or even hundreds of thousands degrees of freedom. It is not possible to create such meshes manually. Mesh generator is a software tool, which divides the solution domain into many subdomains – finite elements. Mesh generators can be of different types.

For *two-dimensional* problems, we want to mention two types: block mesh generators and triangulators.

Block mesh generators require some initial form of gross partitioning. The solution domain is partitioned in some relatively small number of blocks. Each block should have some standard form. The mesh inside block is usually generated by mapping technique.

Triangulators are typically generate irregular mesh inside arbitrary domains. Voronoi polygons and Delaunay triangulation are widely used to generate mesh. Later triangular mesh can be transformed to the mesh consisting of quadrilateral elements. Delaunay triangulation can be generalized for three-dimensional domains.

5.2.2 Mapping technique

Suppose we want to generate quadrilateral mesh inside a domain that has the shape of curvilinear quadrilateral. Mapping technique shown in Fig. 5.2 can be used for this purpose.

If each side of the curvilinear quadrilateral domain can be approximated by parabola then the domain looks like 8-node isoparametric element. The domain is mapped to a square in the local coordinate

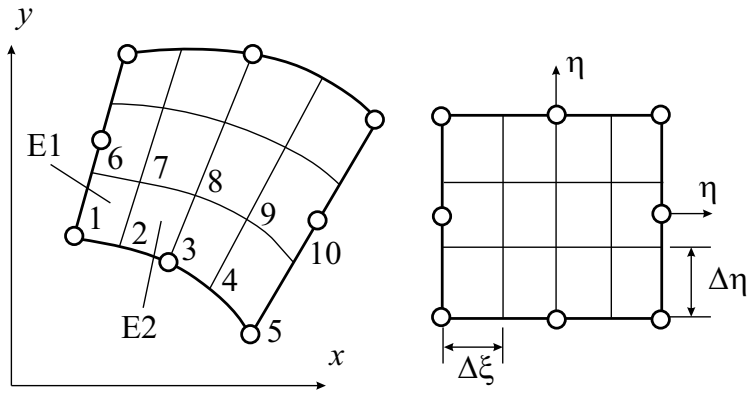


Figure 5.2: Mesh generation with mapping technique.

system $\xi,\ \eta$. The square in coordinates $\xi,\ \eta$ is divided into rectangular elements then nodal coordinates are transformed back to the global coordinate system x, y.

Algorithm of coordinate calculation for node i

n_ξ = number of elements in ξ direction
n_η = number of elements in η direction

Row: $R = (i-1)/(n_\xi + 1) + 1$
Column: $C = \mathrm{mod}((i-1), (n_\xi + 1)) + 1$

$$\Delta\xi = 2/n_\xi \qquad \Delta\eta = 2/n_\eta$$

$$\xi = -1 + \Delta\xi(C-1)$$
$$\eta = -1 + \Delta\eta(R-1)$$

$$x = \sum N_k(\xi,\eta)x_k$$
$$y = \sum N_k(\xi,\eta)y_k$$

Connectivities for element e

Element row: $R = (e-1)/n_\xi + 1$
Element column: $C = \mathrm{mod}((e-1), n_\xi) + 1$
Connectivities (global node numbers):

$$i_1 = (R-1)(n_\xi + 1) + C$$
$$i_2 = i_1 + 1$$
$$i_3 = R(n_\xi + 1) + C + 1$$
$$i_4 = i_3 - 1$$

Shifting of midside nodes closer to some corner of the domain helps to refine (make smaller elements) mesh near this corner. If refinement is done on the element side which is parallel to the local axis ξ and the size of the smallest element near the corner node is Δl then the midside node should be moved to

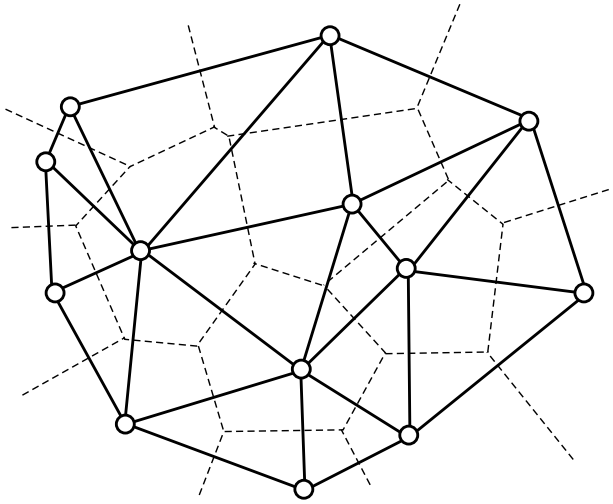


Figure 5.3: Voronoi polygons and Delaunay triangulation.

the position:

$$\alpha = \frac{l_m}{l} = \frac{1 + \frac{\Delta l}{l} n_\xi - \frac{2}{n_\xi}}{4\left(1 - \frac{1}{n_\xi}\right)}$$

Here n_ξ is the number of elements in ξ direction; l_m is a distance from the corner node to the midside node and l is the element side length.

5.2.3 Delaunay triangulation

Consider two-dimensional domain of an arbitrary shape. Let $p_1,\ p_2,\ ..., p_n$ be distinct points inside the domain and at its boundary. Voronoi poligon V_i represents a region in the plane whose points are nearer to node p_i than to any other node. Thus, V_i is an open convex polygon whose boundaries are portions of the perpendicular bisectors of the lines joining node p_i to node p_j when V_i and V_j are contiguous. Voronoi polygons are shown in Fig. 5.3 by dashed lines.

A vertex of a Voronoi polygon is shared by three adjacent polygons. Connecting of three points associated with such adjacent three polygons form a truangle, say T_k. The set of triangles T_k is called the Delaunay triangulation.

Generated triangular mesh can be used for creation of a mesh consisting of quadrilateral elements. Usually joining triangles with subsequent quality improvement is employed for this purpose.

Chapter 6

Assembly and Solution

6.1 Disassembly and assembly

Disassembly is a creation of element vectors from a given global vector. The *disassembly* operation is given by the relation:

$$\{Q_d\} = [A]\{Q\} \tag{6.1}$$

Here $[A]$ is the matrix providing correspondence between global and local numbers of nodes (or degrees of freedom), $\{Q\}$ is the global vector and $\{Q_d\}$ is the vector composed of the element vectors

$$\{Q_d\} = \{\{q_1\}\ \{q_2\}\ ...\} \tag{6.2}$$

For matrices the disassembly operation is not necessary for the finite element procedure implementation.

Assembly is the operation of joining element vectors (matrices) in a global vector (matrix). For vectors the *assembly* operation is given by the relation:

$$\{F\} = [A]^T\{F_d\} \tag{6.3}$$

and for matrices the *assembly* operation is given by the relation:

$$[K] = [A]^T[K_d][A] \tag{6.4}$$

Here $[K]$ is the global matrix and $[K_d]$ is the following matrix consisting of the element matrices:

$$[K_d] = \begin{bmatrix} [k_1] & 0 & 0 \\ 0 & [k_2] & 0 \\ 0 & 0 & ... \end{bmatrix} \tag{6.5}$$

Fraction of nonzero (unit) entries in the matrix $[A]$ is very small. Because of this the matrix $[A]$ is never used explicitly in actual computer codes.

6.2 Disassembly algorithm

Disassembly operation include matrix multiplication with the use of large matrix $[A]$, which gives correspondence between local and global enumerations. Matrix $[A]$ is almost completely composed of zeros. It has only one nonzero entry (= 1) in each row. Nonzero entries in $[A]$ provide information on global addresses where local entries should be taken. Instead of matrix multiplication it is possible just to take vector elements from their global positions and to put to correspondent positions in element vector.

Disassembly for one element vector

n = number of degrees of freedom per element
N = total number of degrees of freedom
E = number of elements
$C[E, n]$ = connectivity array
$f[n]$ = element load vector
$F[N]$ = global load vector
e = element number for which we need local vector

do $i = 1, n$
  $f[i] = F[C[e, i]]$
end do

In the above algorithm we used connectivity information related to degrees of freedom. If connectivity array contains global node numbers and each node has more than one degrees of freedom then instead of one number the block related to the node should be selected. For example, in three-dimensional elasticity problem each node is associated with three displacements (three degrees of freedom).

6.3 Assembly

6.3.1 Assembly algorithm for vectors

Instead of matrix $[A]$, assembly procedures are usually based on direct summation with the use of the element connectivity array.

Suppose that we need to assemble global load vector F using element load vectors f and connectivity array C. A pseudocode of assembly algorithm is as follows:

Assembly of the global vector

n = number of degrees of freedom per element
N = total number of degrees of freedom
E = number of elements
$C[E, n]$ = connectivity array
$f[n]$ = element load vector
$F[N]$ = global load vector

do $i = 1, N$
  $F[i] = 0$
end do
do $e = 1, E$
  generate f
  do $i = 1, n$

$F[C[e,i]] = F[C[e,i]] + f[i]$
end do
end do

Element load vectors are generated when they are necessary for assembly. It can be seen that connectivity entry $C[e,i]$ simply provides address in the global vector where the ith entry of the load vector for element e goes. We assume that connectivity array is written in terms of degrees of freedom. In actual codes the connectivity array contains node numbers which are transformed to degrees of freedom for the current assembled element.

6.3.2 Assembly algorithm for matrices

An algorithm of assembly of the global stiffness matrix K from contributions of element stiffness matrices k can be expressed by the following pseudo-code:

Assembly of the global matrix

n = number of degrees of freedom per element
N = total number of degrees of freedom in the domain
E = number of elements
$C[E,n]$ = connectivity array
$k[n,n]$ = element stiffness matrix
$K[N,N]$ = global stiffness matrix

do $i = 1, N$
do $j = 1, N$
$K[i,j] = 0$
end do
end do
do $e = 1, E$
generate k
do $i = 1, n$
do $j = 1, n$
$K[C[e,i], C[e,j]] = K[C[e,i], C[e,j]] + k[i,j]$
end do
end do
end do

Here for simplicity, element matrices are assembled fully in the full square global matrix. Since the global stiffness matrix is symmetric and sparse, these facts are used to economize space and time in actual finite element codes.

6.4 Displacement boundary conditions

Displacement boundary conditions were not accounted in the functional of the total potential energy. They can be applied to the global equation system after its assembly.

Let us consider application of the displacement boundary condition

$$Q_m = d \tag{6.6}$$

to the global equation system. Two methods can be used for the specification of the displacement boundary condition.

6.4.1 Explicit specification of displacement BC

In the explicit method, we substitute the known value of the displacement $Q_m = d$ in the mth column and move this column to the right-hand side. Then we put zeros to the mth column and mth row of the matrix except the main diagonal element, which is replaced by 1.

Explicit method:

$F_i = F_i - K_{im}d, \quad i = 1...N, i \neq m$
$F_m = d$
$K_{mj} = 0, \quad j = 1...N$
$K_{im} = 0, \quad i = 1...N$
$K_{mm} = 1$

6.4.2 Method of large number

Method of large number uses the fact that computer computations have limited precision. Results of double precision computations contain about 15-16 digits. So, addition $1.0 + 1e^{-17}$ produces 1.0 as the result.

Method of large number ($M >> K_{ij}$):

$K_{mm} = M$
$F_m = Md$

The method of large number is simpler than the explicit method of displacement boundary condition specification. The solution of the finite element problem is the same for both methods.

6.5 Solution of Finite Element Equations

6.5.1 Solution methods

Practical applications of the finite element method lead to large systems of simultaneous linear algebraic equations.

Fortunately, finite element equation systems possess some properties which allows to reduce storage and computing time. The finite element equation systems are: symmetric, positive definite and sparse. Symmetry allows to store only half of the matrix including diagonal entries. Positive definite matrices are characterized by large positive entries on the main diagonal. Solution can be carried out without pivoting. A sparse matrix contains more zero entries than nonzero entries. Sparsity can be used to economize storage and computations.

Solution methods for linear equation systems can be divided into two large groups: direct methods and iterative methods. Direct solution methods are usually used for problems of moderate size. For large problems iterative methods require less computing time and hence they are preferable.

Matrix storage formats are closely related to the solution methods. Below we consider two solution methods which are widely used in finite element computer codes. The first method is the direct LDU solution with profile global stiffness matrix. The second one is the preconditioned conjugate gradient method with sparse row-wise format of matrix storage.

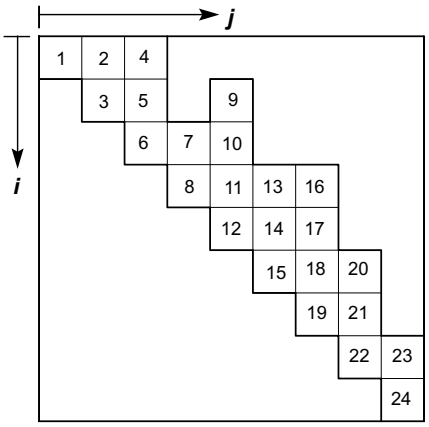


Figure 6.1: Symmetric profile storage of the global matrix.

6.5.2 Direct LDU method with profile matrix

Using symmetric profile format, the global stiffness matrix $[A] = A_{ij}$ of the order N is stored by columns as shown in Fig. 6.1. Each column starts from the first top nonzero element and ends at the diagonal element. The matrix is represented by two arrays:

$a[pcol[N+1]]$ array of doubles containing matrix elements

$pcol[N+1]$ integer pointer array for columns.

The ith element of $pcol$ contains the address of the first column element minus one. The length of the ith column is given by $pcol[i+1] - pcol[i]$. The length of the array a is equal to $pcol[N+1]$ (assuming that array indices begin from 1).

For example, for the matrix shown above, array $pcol$ has the following contents:

$pcol = \{0, 1, 3, 6, 8, 12, 15, 19, 22, 24\}$

It should be noted that proper node ordering can decrease the matrix profile significantly. Usually node ordering algorithms are based on some heuristic methods because full ordering problem with seeking global minimum is too time consuming.

We are going to present algorithms in full matrix notation a_{ij}. Then, it is necessary to have relations between two-index notation for the global stiffness matrix a_{ij} and array a used in FORTRAN or C codes. The location of the first nonzero element in the ith column of the matrix a is given by the following function:

$$FN(i) = i - (pcol[i+1] - pcol[i]) + 1). \tag{6.7}$$

The following correspondence relations can be easily obtained for a transition from two-index notation to FORTRAN/C notation for a one-dimensional array a:

$$A_{ij} \rightarrow a[i + pcol[j+1] - j] \tag{6.8}$$

Solution of symmetric linear algebraic system consists of three stages:

Factorization: $[A] = [U]^T[D][U]$
Forward solution: $\{y\} = [U]^{-T}\{b\}$
Back substitution: $\{x\} = [U]^{-1}[D]^{-1}\{y\}$

The right-looking algorithm of factorization of a symmetric profile matrix is as follows:

LDU factorization (right-looking)

do $j = 2, N$
  $Cdivt(j)$
  do $i = j, N$
    $Cmod(j, i)$
  end do
end do
do $j = 2, N$
  $Cdiv(j)$
end do

$Cdivt(j) =$
  do $i = FN(j)), j - 1$
    $t_i = A_{ij}/A_{ii}$
  end do

$Cmod(j, i) =$
  do $k = max(FN(i), FN(j)), i - 1$
    $A_{ji} = A_{ji} - t_k A_{ki}$
  end do

$Cdiv(j) =$
  do $i = FN(j)), j - 1$
    $A_{ij}/ = A_{ii}$
  end do

Forward solution with triangular matrix and back substitution are given by the pseudo-code:

Forward reduction and back substitution

do $j = 2, N$
  do $i = FN(j), j - 1$
    $b_j = b_j - A_{ij} * b_i$
  end do
end do

do $j = 1, N$
  $b_j = b_j/A_{jj}$
end do
do $i = N, 1, -1$
  do $i = FN(j), j - 1$
    $b_i = b_i - A_{ij} * b_j$
  end do
end do

6.5.3 Tuning of the LDU factorization

Do loop, which takes most time of LDU decomposition is contained in the procedure $Cmod(j,i)$. One column of the matrix is used to modify another column inside inner do loop. Two operands should be loaded from memory in order to perform one Floating-point Multiply-Add (FMA) operation. Data loads can be economized by tuning with the use of blocking technique. After unrolling two outer loops, the tuned version of the LDU decomposition is as follows:

do $j = 1, N, d$
  $Bdivt(j,d)$
  do $i = j+d, N, d$
    $BBmod(j,i)$
  end do
end do
do $j = 2, N$
  $Cdiv(j)$
end do

$Bdivt(k,d) =$
  do $j = k, k+d-1$
    do $i = FN(k)), j-1$
      $t_{ij} = A_{ij}/A_{ii}$
    end do
    do $i = j, k+d-1$
      do $l = max(FN(j), FN(i)), j-1$
        $A_{ji} = A_{ji} - t_{lj}A_{li}$
      end do
    end do
  end do

$BBmod(j,i,d=2) =$
  do $k = max(FN(j), FN(i)), j-1$
    $A_{ji} = A_{ji} - t_{kj}A_{ki}$
    $A_{j+1i} = A_{j+1i} - t_{kj+1}A_{ki}$
    $A_{ji+1} = A_{ji+1} - t_{kj}A_{ki+1}$
    $A_{j+1i+1} = A_{j+1i+1} - t_{kj+1}A_{ki+1}$
  end do
  if $j >= FN(j)$**then**
    $A_{j+1i} = A_{j+1i} - t_{jj+1}A_{ji}$
    $A_{j+1i+1} = A_{j+1i+1} - t_{jj+1}A_{ji+1}$
  end if

Procedure $BBmod(j,i,d)$ performs modification of a column block, which starts from column i by a column block, which starts from column j and contains d columns. The pseudo-code above is given for the block size $d = 2$ for brevity. Such block size is suitable for the solution of two-dimensional problems. In three-dimensional problems, where each node contains tree degrees of freedom, the block size $d = 3$ should be used. In the algorithm above, it is assumed that columns in the block start at the same row of the matrix A. This is fulfilled automatically if the column block contains columns, which are related to one node of the finite element model.

Tuning of the LDU decomposition significantly affects the solution speed. For C codes the solution time can be decreased by about two times. For Java code, tuned solution can take four times less time than untuned algorithm.

6.5.4 Preconditioned conjugate gradient method

A simple and efficient iterative method widely used for the solution of sparse systems is the conjugate gradient (CG) method. In many cases the convergence rate of CG method can be too slow for practical purposes. The convergence rate can be considerably improved by using preconditioning of the equation system:

$$[M]^{-1}[A]x = [M]^{-1}\{b\} \tag{6.9}$$

where $[M]^{-1}$ is the preconditioning matrix which in some sense approximates $[A]^{-1}$. The simplest preconditioning is diagonal preconditioning, in which $[M]$ contains only diagonal entries of the matrix $[A]$. Typical algorithm of PCG method can be presented as the following sequence of computations:

PCG algorithm

Compute $[M]$
$\{x_0\} = 0$
$\{r_0\} = \{b\}$
do $i = 0, 1...$
  $\{w_i = [M]^{-1}\{r_i\}$
  $\gamma_i = \{r_i\}^T\{w_i\}$
  if $i = 0$ $\{p_i\} = \{w_i\}$
  else $\{p_i\} = \{w_i\} + (\gamma_i/\gamma_{i-1})\{p_{i-1}\}$
  $\{w_i\} = [A]\{p_i\}$
  $\beta_i = \{p_i\}^T\{w_i\}$
  $\{x_i\} = \{x_{i-1}\} + (\gamma_i/\beta_i)\{p_i\}$
  $\{r_i\} = \{r_{i-1}\} - (\gamma_i/\beta_i)\{p_i\}$
  if $\gamma_i/\gamma_0 < \varepsilon$ **exit**
end do

In the above PCG algorithm, matrix $[A]$ is not changed during computations. This means that no additional fill arise in the solution process. Because of this, the sparse row-wise format is an efficient storage scheme for PCG iterative method. In this scheme, the values of nonzero entries of matrix $[A]$ are stored by rows along with their corresponding column indices; additional array points to the beginning of each row. Thus the matrix in the sparse row-wise format is represented by the following three arrays:

$A[pcol[N+1]]$ = array containing nonzero entries of the matrix;

$col[pcol[N+1]]$ = column numbers for nonzero entries;

$pcol[N+1]$ = pointers to the beginning of each row.

The following pseudocode explains matrix-vector multiplication $\{w\} = [A]\{p\}$ when matrix $[A]$ is stored in the sparse row-wise format:

Matrix-vector multiplication in sparse-row format

do $j = 1, N$
  $w[j] = 0$
  do $i = pcol[j], pcol[j+1] - 1$

$w[j] = w[j] + A[i] * p[coln[i]]$
end do
end do