Finite Element Method

Tomasz Stręk
Institute of Applied Mechanics,
Poznan University of Technology
ul. Jana Pawla II 24, 60-965 Poznan, Poland
Room 438
www.strek.h2g.pl

DATE: 2020.03.02

X.4. Overview of the finite element method

There is an extensive literature on finite elements, both for theory and applications. Popular books include those by Huebner [Hue1975] (a definitive work from an engineering perspective), Hinton and Owen [Hin1979], Zienkiewicz and Taylor [Zie2000a, Zie2000b, Zie2000c].

In this chapter, we give a sketch of the finite element procedures. This sketch introduces important concepts of local approximation functions (linear and quadratic), the Galerkin method, treatment of boundary conditions, and assembly and solution of global matrices.

The governing equations of given problem must first be discrtetised spatially to obtain the finite element equations. The conventional Galerkin weighted residual technique discussed in previous section/chapter is the most powerful and general method available to achieve finite element spatial discretisation for any set of differential equations.

X.5. Local approximations

In the finite element method, the solution u of a PDE is approximated by low-order polynomials on local elements. The local elements constitute the mesh; typical elements used are triangles and quadrilaterals in 2D, and tetrahedra and hexahedra in 3D.

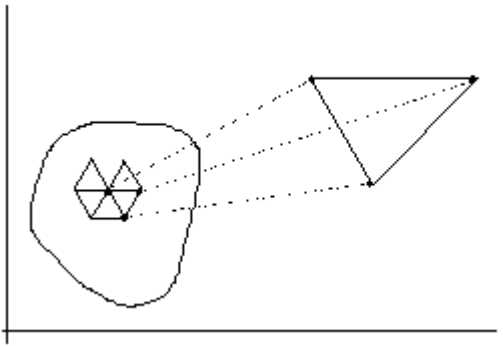


Figure 3.1: 2D triangular mesh.

To give a simple example, consider a triangular mesh in 2D (Figure 3.1).

We concentrate on the single triangle with corner nodes $\{i,j,k\}$, and let the values of u at the nodes be $\{u_i,u_j,u_k\}$. We approximate u within the local element by

$$\bar{u} = \left[N_i(x,y), N_j(x,y), N_k(x,y)\right] \cdot \left[u_i, u_j, u_k\right]^T \qquad \text{(x.y)}$$

where $\{N_i(x,y), N_j(x,y), N_k(x,y)\}$ are interpolation functions. In the simplest case, these are linear polynomials such that

$$N_l(x_p, y_p) = \delta_{lp} \qquad \text{(x.y)}$$

where δ_{lp} is the Kronecker symbol.

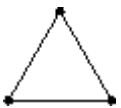
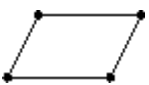
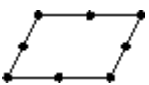
For example, if the local element is the triangle with nodes at $(0,0),(1,0),(1,1)$, the three linear interpolation functions are

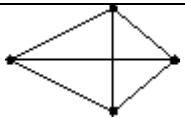
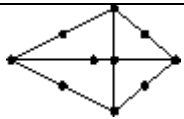
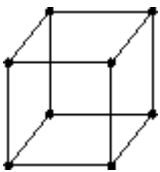
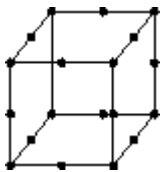
$$N_1 = 1 - x\,,\ N_2 = x - y\,,\ N_3 = y \qquad \text{(x.y)}$$

and, given nodal values (u_i, u_j, u_k), the linear approximation to u in the element is

$$\bar{u} = (1-x)u_i + (x-y)u_j + yu_k\,. \qquad \text{(x.y)}$$

We can use (for example) the following element types:

 3-node triangle, linear approximation	 6-node triangle, quadratic approximation
 4-node quadrilateral, bi-linear approximation	 8-node quadrilateral, bi-quadratic approximation (serendipity element)

 4-node tetrahedron, linear approximation	 10-node tetrahedron, quadratic approximation
 8-node hexahedron, tri-linear approximation	 20-node hexahedron, tri-quadratic approximation (serendipity element)

Figures X. Examples of finite element.

X.6. Calculation of the nodal values

X.5.1. Solution of steady problems

The nodal values are pointwise approximations to the solution of a system of PDEs

$$L(u) = f \qquad \text{(x.y)}$$

defined within a domain Ω and with boundary conditions specified at the boundary of Γ. The boundary conditions typically specify and/or its derivatives. If u is specified on the boundary, it is known as a Dirichlet boundary condition. A natural boundary condition specifies the value of terms arising from integration by parts, such as the flux. The PDE system might be a well-specified problem in its own right, or it might result from an algorithm applied to a more complex problem. Domain Ω can be in 2 or 3 dimensions, whilst both L and u

can have multiple components. In the above, f represents a forcing term for the PDEs, and $L(u)$ typically includes derivatives of u up to second order.

We represent the solution $u(x)$ of the PDE system as follows:

$$u(x) = \sum u_i S_i(x) \qquad \text{(x.y)}$$

The PDE will be satisfied in the weak sense provided

$$\int_\Omega T_j(x)\{L(u) - f\}dV = 0\,, \quad j = 1,...,N_e \qquad \text{(x.y)}$$

for a given set of test functions T_j. If L has multiple components, then T has a corresponding number. In the Galerkin method as implemented in many codes, the shape functions S_i and the test functions T_j are identical. However, since the shape functions do not have second derivatives everywhere, we

usually integrate some terms by parts prior to the substitution of the shape function representation for u. In the finite element method the shape function S_i for each node is continuous and identically zero outside the elements of which the node is a part. Within each of those elements, S_i is a low-order polynomial which takes the value one at node j and zero at all other nodes.

To accomplish the integration by parts, we symbolically decompose the operator L into first- and second-order operators

$$L = L_1 + \nabla L_2 \qquad \text{(x.y)}$$

Here both L_1 and L_2 are first-order operators. L_2 may be vector or tensor valued, with possibly a reduction operation when the grad is applied. The weak form of the PDE thus gives

$$\int_\Omega T_j L(u) dV = \int_\Omega T_j L_1(u) dV + \int_\Omega \nabla T_j L_2(u) dV - \int_\Omega L_2(u) \nabla T_j dV =$$
$$= \int_\Omega T_j L_1(u) dV + \int_{\partial\Omega} \mathbf{n} T_j L_2(u) dS - \int_\Omega L_2(u) \nabla T_j dV = \qquad \text{(x.y)}$$
$$= \int_\Omega T_j f dV$$

where $\partial\Omega$ is the boundary and $\mathbf{n}$ is the unit outward normal. Using this integrated form of the PDE, it is now possible to approximate u using the shape functions. This process is known as assembly, and the end result is a finite dimensional system over the N_e nodes:

$$\sum_{j=1}^{N_e} K_{ij} u_j = r_i, \; i = 1, \ldots, N_e \qquad \text{(x.y)}$$

where

$$K_{ij} = \int_{\Omega} S_i L_1(S_j) dV - \int_{\Omega} L_2(S_j) \nabla S_i dV \qquad \text{(x.y)}$$

and

$$r_i = -\int_{\partial\Omega} \mathbf{n} S_i L_2(u) dS + \int_{\Omega} S_i f dV . \qquad \text{(x.y)}$$

The matrix $\mathbf{K}$ is called the stiffness or global matrix, and vector $\mathbf{r}$ is called the load vector.

X.6.2. Solution of time dependent problems

X.6.2.1. First strategy

Let us consider time-dependent PDE equation

$$\frac{\partial u}{\partial t} + L(u) = f \qquad \text{(x.y)}$$

defined within a domain Ω and with boundary conditions specified at the boundary of Γ. $L(u)$ typically includes derivatives of u up to second order.

The numerical strategies are based on discretizing governing equations first in time, to get a set of simpler partial differential equations, and then discretizing the time-discrete equations in space. There are two main discretizing in time schemes: backward Euler and Crank–Nicolson.

The backward Euler method uses the algorithm

$$\frac{u_{n+1} - u_n}{\delta t} + L(u_{n+1}) = f \qquad \text{(x.y)}$$

which is equivalent to

$$u_{n+1} + \delta t L(u_{n+1}) = \delta t(u_n + f). \qquad \text{(x.y)}$$

In the heart of the algorithm, the equation (x.y) is assembled and solved at each timestep.

The Crank–Nicolson approximation uses the algorithm:

$$\frac{u_{n+1}-u_n}{\delta t}+\frac{1}{2}L\left(u_{n+1}+u_n\right)=f\ . \qquad \text{(x.y)}$$

Next define $\delta u = u_{n+1}-u_n$ and verify that δu satisfies

$$\delta u+\frac{\delta t}{2}L\left(\delta u\right)=-\delta t L\left(u_n\right)+\delta t\ f\ . \qquad \text{(x.y)}$$

The above equation is assembled and solved for δu at each timestep.

Let us consider the backward Euler approximation with the finite element method

$$u_{n+1} + \delta t L(u_{n+1}) = \delta t(u_n + f). \qquad \text{(x.y)}$$

The PDE will be satisfied in the weak sense provided

$$\int_\Omega T_j(x)\{u_{n+1} + \delta t L(u_{n+1}) - \delta t(u_n + f)\} dV = 0, \quad j = 1,...,N_e \qquad \text{(x.y)}$$

for a given set of test functions T_j. To accomplish the integration by parts, we symbolically decompose the operator L into first- and second-order operators

$$L = L_1 + \nabla L_2 \qquad \text{(x.y)}$$

The weak form of the PDE thus gives

$$
\begin{aligned}
&\int_{\Omega} T_j\left(u_{n+1} + \delta t L\left(u_{n+1}\right)\right) dV = \\
&= \int_{\Omega} T_j u_{n+1} dV + \int_{\Omega} T_j L_1\left(u_{n+1}\right) dV + \int_{\Omega} \nabla T_j L_2\left(u_{n+1}\right) dV - \int_{\Omega} L_2\left(u_{n+1}\right) \nabla T_j dV = \\
&= \int_{\Omega} T_j u_{n+1} dV + \int_{\Omega} T_j L_1\left(u_{n+1}\right) dV + \int_{\partial\Omega} \mathbf{n} T_j L_2\left(u_{n+1}\right) dS - \int_{\Omega} L_2\left(u_{n+1}\right) \nabla T_j dV = \\
&= \int_{\Omega} T_j \delta t\left(u_n + f\right) dV
\end{aligned}
\qquad \text{(x.y)}
$$

The end result is a finite dimensional system over the N_e nodes:

$$\sum_{j=1}^{N_e} K_{ij} u_j = r_i, \ i = 1,\ldots,N_e \qquad \text{(x.y)}$$

where

$$K_{ij} = \int_{\Omega} S_i S_j dV + \int_{\Omega} S_i L_1(S_j) dV - \int_{\Omega} L_2(S_j) \nabla S_i dV \qquad \text{(x.y)}$$

and

$$r_i = -\int_{\partial\Omega} \mathbf{n} S_i L_2(u_{n+1}) dS + \int_{\Omega} S_i (u_n + f) dV . \qquad \text{(x.y)}$$

X.6.2.2. Second strategy

Strategies for time-dependent problems presented in previous sections were based on discretizing governing equations first in time and then discretizing the time-discrete equations in space.

One can write finite element shape functions to include the time variable and thus incorporate it into the general finite element method procedure [Hua1999]. However, due to the conceptual simplicity of the time dimension simpler finite difference approximations presented in the previous section are generally favoured. Most schemes currently used are constructed in this way. We can also discretizing governing equations first in space.

Let us consider time-dependent PDE equation

$$\frac{\partial u}{\partial t} + L(u) = f \qquad \text{(x.y)}$$

defined within a domain Ω and with boundary conditions specified at the boundary of Γ. $L(u)$ typically includes derivatives of u up to second order.

The main numerical strategies are based on discretizing governing equations first in time, to get a set of simpler partial differential equations, and then discretizing the time-discrete equations in space.

The PDE will be satisfied in the weak sense provided

$$\int_{\Omega} T_j(x)\left\{\frac{\partial u}{\partial t} + L(u) - f\right\} dV = 0\,, \quad j = 1,\ldots,N_e \qquad \text{(x.y)}$$

for a given set of test functions T_j.

To accomplish the integration by parts, we symbolically decompose the operator L into first- and second-order operators $L = L_1 + \nabla L_2$. Here both L_1 and L_2 are first-order operators. The weak form of the PDE thus gives

$$\int_\Omega T_j \frac{\partial u}{\partial t} dV + \int_\Omega T_j L(u) dV =$$

$$= \int_\Omega T_j \frac{\partial u}{\partial t} dV + \int_\Omega T_j L_1(u) dV + \int_\Omega \nabla T_j L_2(u) dV - \int_\Omega L_2(u) \nabla T_j dV = \quad . \quad \text{(x.y)}$$

$$= \int_\Omega T_j \frac{\partial u}{\partial t} dV + \int_\Omega T_j L_1(u) dV + \int_{\partial\Omega} \mathbf{n} T_j L_2(u) dS - \int_\Omega L_2(u) \nabla T_j dV = \int_\Omega T_j f dV$$

The end result is a finite dimensional system over the N_e nodes:

$$\sum_{j=1}^{N_e} M_{ij}\dot{u}_j + \sum_{j=1}^{N_e} K_{ij}u_j = r_i,\ i = 1,\ldots,N_e \qquad \text{(x.y)}$$

where

$$M_{ij} = \int_{\Omega} S_i S_j dV \qquad \text{(x.y)}$$

$$K_{ij} = \int_{\Omega} S_i L_1(S_j) dV - \int_{\Omega} L_2(S_j) \nabla S_i dV \qquad \text{(x.y)}$$

and

$$r_i = -\int_{\partial\Omega} \mathbf{n} S_i L_2(u) dS + \int_{\Omega} S_i f dV\,. \qquad \text{(x.y)}$$

The matrix $\mathbf{M}$ is called the mass matrix.

X.7. Assembly and sub-assembly

Although the components of $\mathbf{K}$ are written as integrals over the whole mesh, they are in fact zero everywhere except on elements containing both node i and node j. Nodes that have no element in common have a zero entry; hence $\mathbf{K}$ is a sparse matrix. Assembly in FEM is carried out element by element. Each pair of nodes of the element generates a component to be added to $\mathbf{K}$. These components are added into an element matrix, prior to being added into the global matrix. In this process, subsets of the global vectors required as data for assembly, including the coordinates, are selected and sorted into a standard nodal order for the element. This is referred to as the local level; the vectors are called local vectors. The integrals making up $\mathbf{K}$ have to be evaluated, and this is done by Gauss quadrature. Standard interpolation formulae are used to calculate the quantities concerned at quadrature points, and weighted sums of these values are used to approximate the integral.

X.8. Boundary conditions

The finite element method distinguishes between essential and natural boundary conditions:

a) essential (Dirichlet)

$$G(u) = g \qquad \text{(x.y)}$$

where the value of variable is prescribed;

b) natural (Neumann)

$$S(u) = s \qquad \text{(x.y)}$$

As an introduction to these concepts, consider the weak form of the left-hand side of Laplace's equation:

$$\int_{\Omega} S_i \nabla^2 u \, dV = -\int_{\Omega} \nabla S_i \cdot \nabla u \, dV + \int_{\partial\Omega} S_i \mathbf{n} \cdot \nabla u \, dS \qquad \text{(x.y)}$$

The last term is an integral over the boundary of the normal derivative (or flux). This is called a natural boundary condition; the boundary integrands represent a physical quantity (for example, flux in a diffusion problem, or stress in a linear elasticity problem). The condition is implemented by substituting directly if the integrand is known, or by substituting an expression involving unknowns. Natural boundary conditions are specified at the time that the PDE problem is specified. On the other hand, an absolute specification $u_i = c_i$ at some set of boundary nodes is called an essential boundary condition. This is enforced by including it in the set of equations, replacing the equation which had been formed by using S_i as a test function.

To illustrate this point, suppose that node 3 is a boundary node with value $u_3 = U$. The third row of the matrix is independent of other nodal values and is given by

$$[0,0,1,0,\ldots,0]\cdot[u_1,u_2,u_3,\ldots,u_N]=U \qquad \text{(x.y)}$$

Incorporation of this boundary condition into the matrix system gives the new system

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} & \ldots \\ K_{21} & K_{22} & K_{23} & K_{24} & \ldots \\ 0 & 0 & 1 & 0 & \ldots \\ K_{41} & K_{42} & K_{43} & K_{44} & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \ldots \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ U \\ r_4 \\ \ldots \end{bmatrix} \qquad \text{(x.y)}$$

If the matrix $\mathbf{K}$ is symmetric it is necessary to do a further elimination to regain symmetry:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & K_{14} & \dots \\ K_{21} & K_{22} & 0 & K_{24} & \dots \\ 0 & 0 & 1 & 0 & \dots \\ K_{41} & K_{42} & 0 & K_{44} & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \dots \end{bmatrix} = \begin{bmatrix} r_1 - UK_{13} \\ r_2 - UK_{23} \\ U \\ r_4 - UK_{43} \\ \dots \end{bmatrix} \qquad \text{(x.y)}$$

To summarise the discussion so far, essential boundary conditions are implemented by modification of the global matrix and right-hand side (RHS) vector, whilst natural boundary conditions are often accounted for in the RHS vector alone. These concepts are so important, however, that we now provide a more detailed commentary.

In the Galerkin procedure, a term such as $D_m Exp$ (where D_m denotes partial differentiation with respect to x_m, m could be i or j, and Exp might or might not involve suffices, an unknown or another differentiation) is multiplied by a test function T over the region Ω with boundary $\partial\Omega$. For all second-order terms and some first-order terms, the integration is done by parts. This gives:

$$\int_{\Omega} T\, D_m Exp\, dV = -\int_{\Omega} (D_m T) Exp\, dV + \int_{\partial\Omega} T\, n_m Exp\, dS\,. \qquad \text{(x.y)}$$

The application of boundary conditions in the finite element method requires either that some information is used to replace the integrand resulting from second-order terms, when T is non-zero there, or that for such test functions the whole equation is replaced by an essential condition. So when we implements such terms, we adds only the second integral into the equations - either to the sparse matrix or the right-hand side vector.

The boundary integrals often have physical significance, and it is best to try to formulate the equations to take advantage of this. In fact, many second-order equations correspond to one of the following patterns:

- rate of change of heat with time = div (flux),
- change of momentum with time = div (stress).

For steady equations the rates would be zero. The divergences are integrated by parts, and the boundary integrands will be the normal components of either the flux or the stress. On interior boundaries, integrals are generated on each side, and the net integrand is the difference.

There are three possible specifications on any particular boundary.
1. We can assert that the integrand (flux, stress, ...) is zero on an outside boundary or the integrand is continuous on an interior boundary.
2. We can set the boundary integral, by including the appropriate value, which will be added to the left-hand side.

3. We can specify a Dirichlet condition, in which case the equation, with its boundary integrals, will be overwritten.

X.9. The solution stage

The finite element solution is obtained by solving

$$\sum_{j=1}^{N} K_{ij} u_j = r_i \ , \ i = 1,...,N \qquad \text{(x.y)}$$

where the right-hand side is made up of boundary integrals from natural boundary conditions, terms from essential boundary conditions and boundary integrals. The matrix system is invariably large and sparse, and often symmetric positive definite. To solve the matrix system we can use both direct and indirect. The above description illustrates concepts underlying the use of finite elements method.

X.10. Shape functions in local coordinate system

X.10.1. Basic two-dimensional *C*(0) rectangular elements

The shape functions for the four noded rectangular element in local coordinate system can be abbreviated to

$$N_i = \frac{1}{4}(1+\xi_i\xi)(1+\eta_i\eta) \qquad \text{(x.y)}$$

where

i	1	2	3	4
ξ_i	-1	1	1	-1
η_i	-1	-1	1	1

I	ξ_i	η_i
1	-1	-1
2	1	-1
3	1	1
4	-1	1

The shape functions for the eight noded rectangular element can be summarised: for corner nodes

$$N_i = \frac{1}{4}(1+\xi_i\xi)(1+\eta_i\eta)(\xi_i\xi+\eta_i\eta-1) \qquad \text{(x.y)}$$

for midside nodes $\xi_i = 0$

$$N_i = \frac{1}{2}(1-\xi^2)(1+\eta_i\eta) \qquad \text{(x.y)}$$

for midside nodes $\eta_i = 0$

$$N_i = \frac{1}{2}(1+\xi_i\xi)(1-\eta^2) \qquad \text{(x.y)}$$

where

i	1	2	3	4	5	6	7	8
ξ_i	-1	0	1	1	1	0	-1	-1
η_i	-1	-1	-1	0	1	1	1	0

i	ξ_i	η_i
1	-1	-1
2	0	-1
3	1	-1
4	1	0
5	1	1
6	0	1
7	-1	1
8	-1	0

X.10.2. Isoparametric elements

We can generalise these elements by using the isoparametric representation. Consider an isoparametric formulation for an *m*-node element. We can express the geometry of such elements using the nodal coordinates *x* and *y* of element and the shape functions of element described above. Thus at any point within an element the Cartesian coordinates may be obtained from the expressions:

$$x(\xi,\eta) = \sum_{i=1}^{m} N_i(\xi,\eta) x_i \qquad \text{(x.y)}$$

and

$$y(\xi,\eta) = \sum_{i=1}^{m} N_i(\xi,\eta) y_i \qquad \text{(x.y)}$$

The Cartesian derivative of any function *f* defined over the element using the expression:

$$f(\xi,\eta)=\sum_{i=1}^{m} N_i(\xi,\eta)\, f_i \qquad \text{(x.y)}$$

may be obtained by the chain rule of differentiation

$$\frac{\partial f}{\partial x}=\frac{\partial f}{\partial \xi}\frac{\partial \xi}{\partial x}+\frac{\partial f}{\partial \eta}\frac{\partial \eta}{\partial x} \qquad \text{(x.y)}$$

$$\frac{\partial f}{\partial y}=\frac{\partial f}{\partial \xi}\frac{\partial \xi}{\partial y}+\frac{\partial f}{\partial \eta}\frac{\partial \eta}{\partial y} \qquad \text{(x.y)}$$

where

$$\frac{\partial f}{\partial \xi} = \sum_{i=1}^{m} \frac{\partial N_i}{\partial \xi} f_i \qquad \text{(x.y)}$$

$$\frac{\partial f}{\partial \eta} = \sum_{i=1}^{m} \frac{\partial N_i}{\partial \eta} f_i \qquad \text{(x.y)}$$

The terms

$$\frac{\partial \xi}{\partial x} \quad \frac{\partial \eta}{\partial x} \quad \frac{\partial \xi}{\partial y} \quad \frac{\partial \eta}{\partial y} \qquad \text{(x.y)}$$

can be obtained using the following procedure. First we evaluate the matrix

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \displaystyle\sum_{i=1}^{m} \frac{\partial N_i}{\partial \xi} x_i & \displaystyle\sum_{i=1}^{m} \frac{\partial N_i}{\partial \xi} y_i \\ \displaystyle\sum_{i=1}^{m} \frac{\partial N_i}{\partial \eta} x_i & \displaystyle\sum_{i=1}^{m} \frac{\partial N_i}{\partial \eta} y_i \end{bmatrix} \qquad \text{(x.y)}$$

which is termed the Jacobian matrix **J**. The inverse of the Jacobian is then evaluated

$$\mathbf{J}^{-1} = \begin{bmatrix} \dfrac{\partial \xi}{\partial x} & \dfrac{\partial \eta}{\partial x} \\ \dfrac{\partial \xi}{\partial y} & \dfrac{\partial \eta}{\partial y} \end{bmatrix} = \frac{1}{\det \mathbf{J}} \begin{bmatrix} \dfrac{\partial y}{\partial \eta} & -\dfrac{\partial y}{\partial \xi} \\ -\dfrac{\partial x}{\partial \eta} & \dfrac{\partial x}{\partial \xi} \end{bmatrix} \qquad \text{(x.y)}$$

An element area of the element is given as

$$dxdy = \det \mathbf{J}\, d\xi d\eta \qquad \text{(x.y)}$$

For an isoparametric element we have

$$\int_{\Omega_{xy}} f(x,y)dxdy = \int_{\Omega_{\xi\eta}} f(\xi,\eta)\det \mathbf{J}\, d\xi d\eta = \int_{\Omega_{\xi\eta}} g(\xi,\eta)\, d\xi d\eta \qquad \text{(x.y)}$$

and

$$\int_{\Gamma_{xy}} f(x,y)dxdy = \int_{\Gamma_{\xi\eta}} f(\xi,\eta)\det \mathbf{J}\, d\Gamma_{\xi\eta} = \int_{\Gamma_{\xi\eta}} g(\xi,\eta)\, d\Gamma_{\xi\eta} \qquad \text{(x.y)}$$

X.10.3. Numerical integration

We can adopt a numerical integration procedure to evaluate such integrals

$$\int_{\Omega_{\xi\eta}} g(\xi,\eta)\, d\xi d\eta = \int_{-1}^{1}\int_{-1}^{1} g(\xi,\eta)\, d\xi d\eta \qquad \text{(x.y)}$$

or

$$\int_{\Gamma_{\xi\eta}} g(\xi,\eta) d\Gamma_{\xi\eta} = \begin{cases} \int_{-1}^{1} g(\pm 1,\eta) d\eta \\ \int_{-1}^{1} g(\xi,\pm 1) d\xi \end{cases} \qquad \text{(x.y)}$$

The *r*-point Gauss-Legendre integration rule have the form:

$$\int_{-1}^{1}\int_{-1}^{1} g(\xi,\eta)\, d\xi d\eta = \sum_{i=1}^{r}\sum_{j=1}^{r} w_i w_j g(\overline{\xi_i},\overline{\eta_j}) \qquad \text{(x.y)}$$

r	$\overline{\xi_i}$	w_i
1	0.00000	2.00000
2	0.577530 -0.577530	1.00000 1.00000
3	0.00000 0.774597 -0.774597	8/9 5/9 5/9
4	0.861136 -0.861136 0.339981 -0.339981	0.347855 0.347855 0.652145 0.652145

Note that *r*-point rule can integrate exactly polynomial functions of degree *2r-1* or less. This type of formulation enables us to use elements of the very general nature.

X.11. Shape functions for triangular and tetrahedral element family

X.11.1. Triangular element family

The advantage of an arbitrary triangular shape in approximating to any boundary shape has been amply demonstrated in [Zie2000a]. The number of nodes in each member of the family is now such that a complete polynomial expansion, of the order needed for interelement compatibility, is ensured. This follows by comparison with the Pascal triangle in which we see the number of nodes coincides exactly with the number of polynomial terms required. Direct generation of shape functions will be preferred - and indeed will be shown to be particularly easy. Before proceeding further it is useful to define a special set of normalized coordinates for a triangle (area coordinates) [Zie200a].

While Cartesian directions parallel to the sides of a rectangle were a natural choice for that shape, in the triangle these are not convenient. A new set

of coordinates, L_1, L_2, L_3 for a triangle 1,2,3 is defined by the following linear relation between these and the Cartesian system:

$$x = \sum_{i=1}^{3} L_i x_i$$
$$y = \sum_{i=1}^{3} L_i y_i \qquad \text{(x.y)}$$
$$1 = \sum_{i=1}^{3} L_i$$

To every set, L_1, L_2, L_3 (which are not independent, but are related by the third equation), there corresponds a unique set of Cartesian coordinates.

At point j: $L_i = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$ for $j = 1,2,3$.

Solving Eq. (xxxx) gives

$$L_i = \frac{a_i + b_i x + c_i y}{2\Delta} \qquad \text{(x.y)}$$

in which

$$\Delta = \frac{1}{2} det \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} \qquad \text{(x.y)}$$

and

$$a_1 = x_2 y_3 - x_3 y_2\,,\ b_1 = y_2 - y_3\,,\ c_1 = x_3 - x_2 \qquad \text{(x.y)}$$

etc., with cyclic rotation of indices 1,2 and 3.

Relation between the Cartesian coordinates and area coordinates implicates that geometric place for L_i, $i=1,2,3$, are lines parallel to edge $j-k$ ($i \neq j \neq k$) with $L_i = 0$.

For the first element of the triangular series (linear element with three nodes placed at the vertices of triangle) the shape functions are simply the area coordinates. Thus

$$N_i = L_i \qquad \text{(x.y)}$$

for $i=1,2,3$. This is obvious as each individually gives unity at one node i, zero at others, and varies linearly everywhere.

To derive shape functions for other elements a simple recurrence relation can be derived. However, it is very simple to write an arbitrary triangle of order m. We can use Silvester's formula [Sil1969] to generate shape functions of order m:

$$N_{abc}(L_1,L_2,L_3)=P_a(L_1)P_b(L_2)P_c(L_3) \qquad \text{(x.y)}$$

where

$$P_0(L_i)=1$$
$$P_s(L_i)=\prod_{j=1}^{s}\frac{mL_i-j+1}{j} \qquad \text{(x.y)}$$

and

$$a+b+c=m. \qquad \text{(x.y)}$$

Shape functions are generated for all nodes, all sequences of (a,b,c) which satisfies $a+b+c=m$. Indices a,b,c in above equations denotes node position in triangle. The area coordinates of this node we can evaluate using

$$L_1 = \frac{a}{m}, L_2 = \frac{b}{m}, L_3 = \frac{c}{m}. \qquad \text{(x.y)}$$

Number of nodes of shape function (interpolating polynomial) of order m is equal to $(m+1)(m+2)/2$. For example, if we would like to evaluate shape functions of second order we have to calculate following expressions

$$N_{200}, N_{020}, N_{002}, N_{110}, N_{101}, N_{011}. \qquad \text{(x.y)}$$

It is easy to verify that corner nodes shape functions are

$$N_{200} = L_1(2L_1 - 1), N_{020} = L_2(2L_2 - 1), N_{002} = L_3(2L_3 - 1) \qquad \text{(x.y)}$$

And mid-sides nodes shape functions are as follows

$$N_{110} = 4L_1L_2, N_{101} = 4L_1L_3, N_{011} = 4L_2L_3. \qquad \text{(x.y)}$$

We can notice that in high-order shape functions in all cases three nodes of triangle element are placed at vertices of triangle and others on boundary or inside the triangle.

When element matrices have to be evaluated it will follow that we are faced with integration of quantities defined in terms of area coordinates over the triangular region. It is useful to note in this context the following exact integration expression

$$\iint_{\Delta} L_1^a L_2^b L_3^c dx dy = 2\Delta \int_0^1 \int_0^{1-L_2} L_1^a L_2^b L_3^c dL_1 dL_2 = \frac{a!\, b!\, c!}{(a+b+c+2)!} 2\Delta . \qquad \text{(x.y)}$$

In paper [Str1999] procedure automatically generating shape functions using symbolic computations has been presented.

X.11.2. Tetrahedral element family

Firstly, once again complete polynomials in three coordinates are achieved at each stage. Secondly, as faces are divided in a manner identical with that of the previous triangles, the same order of polynomial in two coordinates in the plane of the face is achieved and element compatibility ensured [Zie2000a].

Once again special coordinates L_1, L_2, L_3, L_4 for a tetrahedral 1,2,3,4 are introduced defined by:

$$x = \sum_{i=1}^{4} L_i x_i$$

$$y = \sum_{i=1}^{4} L_i y_i \qquad \text{(x.y)}$$

$$z = \sum_{i=1}^{4} L_i z_i$$

$$1 = \sum_{i=1}^{4} L_i$$

Solving Eq. (xxxx) gives

$$L_i = \frac{a_i + b_i x + c_i y + d_i z}{6V} \qquad \text{(x.y)}$$

in which

$$V = \frac{1}{6} det \begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix} \qquad \text{(x.y)}$$

For the first element of the triangular series (linear element with three nodes placed at the vertices of triangle) the shape functions are simply the area coordinates. Thus

$$N_i = L_i \qquad \text{(x.y)}$$

for $i = 1,2,3,4$. This is obvious as each individually gives unity at one node i, zero at others, and varies linearly everywhere.

We can use again Silvester's formula [Sil1969] to generate shape functions of order m:

$$N_{abcd}(L_1, L_2, L_3, L_4) = P_a(L_1) P_b(L_2) P_c(L_3) P_d(L_4) \qquad \text{(x.y)}$$

where

$$P_0(L_i) = 1 \qquad \text{(x.y)}$$

$$P_s(L_i) = \prod_{j=1}^{s} \frac{mL_1 - j + 1}{j}$$

and

$$a + b + c + d = m. \qquad \text{(x.y)}$$

Shape functions are generated for all nodes, all sequences of (a,b,c) which satisfies $a + b + c + d = m$. Indices a,b,c,d in above equations denotes node position in triangle. The area coordinates of this node we can evaluate using

$$L_1 = \frac{a}{m}, L_2 = \frac{b}{m}, L_3 = \frac{c}{m}, L_4 = \frac{d}{m}. \qquad \text{(x.y)}$$

Number of nodes of shape function (interpolating polynomial) of order m is equal to $(m+1)(m+2)(m+3)/6$. For example, if we would like to evaluate shape functions of second order we have to calculate following expressions

$$N_{2000}, N_{0200}, N_{0020}, N_{0002}$$
$$N_{1100}, N_{1010}, N_{1001}, N_{0110}, N_{0101}, N_{0011}. \qquad \text{(x.y)}$$

It is easy to verify that corner nodes shape functions are

$$N_{2000} = L_1(2L_1 - 1), N_{0200} = L_2(2L_2 - 1), N_{0020} = L_3(2L_3 - 1), N_{0002} = L_4(2L_4 - 1) \qquad \text{(x.y1)}$$

And mid-sides nodes shape functions are as follows

$$N_{1100} = L_1 L_2, N_{1010} = L_1 L_3, N_{1001} = L_1 L_4,$$
$$N_{0110} = L_2 L_3, N_{0101} = L_2 L_4, N_{0011} = L_3 L_4 \qquad \text{(x.y)}$$

The following exact integration expression is valid

$$\iiint_V L_1^a L_2^b L_3^c L_4^d dxdydz = \frac{a!\,b!\,c!\,d!}{(a+b+c+3)!} 6V \,. \qquad \text{(x.y)}$$

Literatura

[Hinton1979] Hinton E., Owen D.R.J., *An Introduction to Finite Element Computations*, Pineridge, Swansea, 1979.

[Huang1999] Hou-Cheng Huang, Zheng-Hua Li and Asif S. Usmani, *Finite Element Analysis of Non-Newtonian Flow*, Springer-Verlang, London, 1999.

[Huebner1975] Huebner K.H., *The Finite Element Method for Engineers*, Wiley, Toronto, 1975.

[Taylor1981] Taylor C., Hughes T.G., *Finite Element Programming of the Navier-Stokes Equations*, Pineridge, Swansea, 1981.

[Zienkiewicz2000] Zienkiewicz O.C. ,Taylor R.L., The Finite Element Method, Volume 1: The Basis (Fifth edition), Butterworth-Heinemann, Oxford, 2000.

[Zie2000b] Zienkiewicz O.C. ,Taylor R.L., The Finite Element Method, Volume 2: Solid Mechanics (Fifth edition), Butterworth-Heinemann, Oxford, 2000.
[Zie2000c] Zienkiewicz O.C. ,Taylor R.L., The Finite Element Method, Volume 3: Fluid Dynamics (Fifth edition), Butterworth-Heinemann, Oxford, 2000.
O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, The Finite Element Method: Its Basis and Fundamentals Butterworth-Heinemann, Elsevier, 2014.
O.C. Zienkiewicz, R.L. Taylor, D.D. Fox, The Finite Element Method for Solid and Structural Mechanics, Butterworth-Heinemann, Elsevier, 2014.
O.C. Zienkiewicz, R.L. Taylor, P. Nithiarasu, The Finite Element Method for Fluid Dynamics, Butterworth-Heinemann, Elsevier, 2014
[Rymarz1993] Rymarz Cz., *Mechanics of Continuum Media (in Polish: Mechanika ośrodków ciągłych)*, Wydawnictwo Naukowe PWN, Warsaw, 1993.