

METODA ELEMENTÓW SKOŃCZONYCH

X. The Finite Element Method

X.1. Introduction

Many CFD practitioners prefer finite volume methods because the derivation of the discrete equations is based directly on the underlying physical principles, thus resulting in “physically sound” schemes. From a mathematical point of view, finite volume, difference, and element methods are closely related, and it is difficult to decide that one approach is superior to the others; these spatial discretization methods have different advantages and disadvantages.

Today the Finite Element Method (FEM) has been widely employed in solving field problems arising in modern industrial practices. The text in this chapter (section) is short introduction to the application of the FEM to the analysis of fluid flow which is a very common phenomenon in many processes of manufacturing and engineering.

In this chapter / section we shall introduce the reader to a finite element treatment of the equations of motion for various problems of fluid mechanics. Much of the activity in fluid mechanics has however pursued a finite difference formulation and more recently a derivative of this known as the finite volume technique. Competition between the newcomer of finite elements and established techniques of finite differences have appeared on the surface and led to a much slower adoption of the finite element process in fluid mechanics than in structures. The reasons for this are perhaps simple. In solid mechanics or structural problems, the treatment of continua arises only on special occasions. The engineer often dealing with structures composed of bar-like elements does not need to solve continuum problems. Thus his interest has focused on such continua only in more recent times. In fluid

mechanics, practically all situations of flow require a two or three dimensional treatment and here approximation was frequently required. This accounts for the early use of finite differences in the 1950s before the finite element process was made available. However, as it was pointed out in book [Zie2000a], there are many advantages of using the finite element process. This not only allows a fully unstructured and arbitrary domain subdivision to be used but also provides an approximation which in self-adjoint problems is always superior to or at least equal to that provided by finite differences.

One advantage of the finite element method over finite difference methods is the relative ease with which the boundary conditions of the problem are handled [Bur1985]. Many physical problems have boundary conditions involving derivatives and the boundary of the region is irregularly shaped. Boundary conditions of this type are very difficult to handle using finite difference techniques, since each boundary condition involving a derivative must be approximated by a difference quotient at the grid points, and irregular shaping of the boundary makes placing the grid points difficult.

The construction procedure in the FEM is independent of the particular boundary conditions of the problem.

X.2. The method of weighted residuals (Galerkin's method)

Some physical problems can be stated directly in the frame of variational principle which consists of determining the function which makes a certain integral statement called functional stationary. However the form of the variational principle is not always obvious and such a principle does not exist for many continuum problems.

As an alternative to solve such differential equations we may use a variety of weighted residual methods. Weighted residual methods are numerical techniques which can be used to solve a single or set of partial differential equations. Consider such a set in domain Ω with boundary $\delta\Omega=\Gamma$, where u is the exact solution and may represent a single variable or a column

vector of variables. where at least the first order gradient in the variable is prescribed.

Applying the method of weighted residuals involves basically two steps. The first step is to assume the general functional behavior of the dependent field variable in some way so as to approximately satisfy the given differential equation and boundary conditions. Substitution of this approximation into the original differential equation and boundary conditions then results in some error called a residual. This residual is required to vanish in some average sense over the entire solution domain. The second step is to solve the equation (or equations) resulting from the first step and thereby specialize the general functional form to a particular function, which then becomes the approximate solution sought. According to Galerkin's method, the weighting functions are chosen to be the same as the approximating functions.

Let us consider differential equation

$$L(u) = f \quad (x,y)$$

defined within a domain Ω and with boundary conditions specified at the boundary of Γ .

An operator L is said to be linear if and only if it satisfies the relation

$$L(a \cdot u + b \cdot v) = aL(u) + bL(v) \quad (\text{x.y})$$

for any scalars a and b and dependent variables u and v . When an operator does not satisfy the above condition it is said to be nonlinear. The function u (i.e. solution) is not only required to satisfy the operator equation, it is also required to satisfy the boundary conditions associated with the operator.

In the weighted-residual method the solution u is approximated by expressions \bar{u} of the form

$$\bar{u} = S_0 + \sum_{j=1}^N u_j S_j \quad (\text{x.y})$$

where S_j are trial functions, and S_0 must satisfy all the specified boundary conditions ($S_0 = 0$ if all the specified boundary conditions are homogeneous) of the problem, and S_i must satisfy the following conditions:

- S_j should be such that $L(S_j)$ is well defined and nonzero, i.e. sufficiently differentiable;
- S_j must satisfy at least the homogeneous form of the essential boundary conditions of the problem;
- for any N , the set $\{S_j, j = 1, 2, \dots, N\}$ is linearly independent.

We begin by introducing the error, or residual, R_Ω in the approximation (by substitution of the approximation \bar{u} into the operator equation) which is defined by

$$R_{\Omega} = L(\bar{u}) - f \quad (\text{x.y})$$

where \bar{u} contains trial functions and satisfies the Dirichlet boundary conditions of $\bar{u} = u_0$ at $\Gamma_1 \subseteq \Gamma$. If the residual is smaller the approximation is better. It should be noted that R_{Ω} is a function of position in Ω . Now we attempt to reduce this residual as close to zero as possible. If we have

$$\int_{\Omega} T_i R_{\Omega} d\Omega = 0 \quad (\text{x.y})$$

where $T_i, i = 1, 2, \dots, M$ is a set of arbitrary functions and $M \rightarrow \infty$, then it can be said that the residual R_{Ω} vanishes. Here T_i are called weighting functions which, in general, are not the same as the approximation (trial) functions S_i . Expanding above equation we have

$$\int_{\Omega} T_i(L(\bar{u}) - f) d\Omega = 0. \quad (\text{x.y})$$

A function \bar{u} that satisfies above equation for every function T_i in Ω is a weak solution of the differential equation, whereas the strong solution \bar{u} satisfies the differential equation at every point of Ω .

When the operator L is linear above equation can be simplified to the form

$$\sum_{j=1}^N \left(\int_{\Omega} T_i L(S_j) d\Omega \right) u_j = \int_{\Omega} T_i (f - L(S_0)) d\Omega \quad (\text{x.y})$$

or

$$\sum_{j=1}^N A_{ij} u_j = f_i . \quad (\text{x.y})$$

where

$$A_{ij} = \int_{\Omega} T_i L(S_j) d\Omega \quad (\text{x.y})$$

and

$$f_i = \int_{\Omega} T_i (f - L(S_0)) d\Omega . \quad (\text{x.y})$$

Note that the coefficients of matrix \mathbf{A} is not symmetric $A_{ij} \neq A_{ji}$.

The weighted-residual method (when $T_i \neq S_i$) is also sometimes referred to as the Petrov-Galerkin method. For different choices of T_i the method is known by different names. We outline below the most frequently used methods.

The Galerkin method. For $T_i = S_i$ the weighted-residual method is known as the Galerkin method. When the operator is a linear differential operator of even order, the Galerkin method reduces to the Ritz method. In this case the resulting matrix will be symmetric because half of the differentiation can be transformed to the weight functions.

The least-squares method. The least-squares method seeks a solution in the form (XXX) $\bar{u} = S_0 + \sum_{j=1}^N u_j S_j$ and determines the constants u_j by minimizing the integral of the square of the residual

$$\frac{\partial}{\partial u_i} \int_{\Omega} R_{\Omega}^2 d\Omega = 0 \quad (\text{x.y})$$

or

$$\int_{\Omega} \frac{\partial R_{\Omega}}{\partial u_i} R_{\Omega} d\Omega = 0 \quad (\text{x.y})$$

A comparison of Eq.(XXX) with Eq.(XXX) shows that $T_i = \frac{\partial R_{\Omega}}{\partial u_i}$. If L is a linear operator Eq.(XXX) becomes

$$\sum_{j=1}^N \left(\int_{\Omega} L(S_i) L(S_j) d\Omega \right) u_j = \int_{\Omega} L(S_i) (f - L(S_0)) d\Omega$$

which yields a symmetric matrix but requires the same order of differentiation as the operator equation.

The collocation method. The collocation method seeks approximate solution \bar{u} by requiring the residual $R_\Omega = R_\Omega(\mathbf{x}, \mathbf{u})$ in the equation to be identically to zero at N selected points $\mathbf{x}_i, i = 1, 2, \dots, N$ in the domain Ω

$$R_\Omega(\mathbf{x}_i, u_j) = 0. \quad (\text{x.y})$$

The selection of the points \mathbf{x}_i is crucial in obtaining a well conditioned system of equations and ultimately in obtaining an accurate solution. The collocation points can be shown to be a special case of Eq. (XXX) $\int_\Omega T_i R_\Omega d\Omega = 0$ for

$T_i = \delta(\mathbf{x} - \mathbf{x}_i)$, where $\delta(\mathbf{x})$ is the Dirac delta function

$$\int_{\Omega} f(\mathbf{x})\delta(\mathbf{x}-\xi)d\Omega = f(\xi). \quad (\text{x.y})$$

The Courant method. To so-called Courant method combines the basic concepts of the Ritz method and the least-squares method (for linear operator). The method seeks approximate solution \bar{u} by minimizing the modified quadratic functional

$$I_p(\bar{u}) = I(\bar{u}) + \frac{\alpha}{2} \|L(\bar{u}) - f\|^2 \quad (\text{x.y})$$

Where $I(u)$ is the quadratic functional associated with $L(u) = f$, when L is linear, and α is the penalty parameter (preassigned). Obviously the statement make sense only for operator equation that admit functional formulation.

X.3. Time discretization schemes

In time-dependent (unsteady) problems, the undetermined unknown parameters u_j are assumed to be functions of time, while the trial functions S_j are assumed to depend on spatial coordinates. This leads to two stages of solution, both of which employ approximation methods. In the solution of unsteady problems we can first consider the spatial approximation and the time (or timelike) approximation next. Such a strategy is commonly known as semidiscrete approximation in space. The spatial discretisation leave us with the first order ordinary differential equations with respect to time. There are numerous ways of accomplishing the discretisation of the time domain.

The first order system of equations

$$\mathbf{M} \dot{\mathbf{w}} + \mathbf{K} \mathbf{w} = \mathbf{r} \quad (\text{x.1})$$

where $\dot{\mathbf{w}} = \frac{\partial \mathbf{w}}{\partial t}$ needs to be discretised in time. If we consider only single PDE we have $\mathbf{w} = \mathbf{u}$ else \mathbf{w} is set of unknown parameters of few variables (e.g. velocity, temperature, pressure). The most commonly used method for such a system is the generalized mid-point or trapezoidal family of methods. The trapezoidal method applied to can be written as follows

$$\mathbf{M}(\mathbf{w}_{n+1}, t_{n+1}) \dot{\mathbf{w}}_{n+1} + \mathbf{K}(\mathbf{w}_{n+1}, t_{n+1}) \mathbf{w}_{n+1} = \mathbf{r}_{n+1} \quad (\text{x.2})$$

and

$$\frac{\mathbf{w}_{n+1} + \mathbf{w}_n}{2} = \frac{\mathbf{w}_{n+1} - \mathbf{w}_n}{\delta t_n} \quad (\text{x.3})$$

Substituting (x.3) in (x.2) we obtain

$$\left(\frac{2\mathbf{M}_{n+1}}{\delta t} + \mathbf{K}_{n+1} \right) \mathbf{w}_{n+1} = \left(\frac{2}{\delta t} \mathbf{w}_n + \dot{\mathbf{w}}_n \right) \mathbf{M}_{n+1} + \mathbf{r}_{n+1} \quad (\text{x.4})$$

The method involves the calculation of the derivatives on the right hand side. Here $\mathbf{M}_{n+1} = \mathbf{M}(\mathbf{w}_{n+1}, t_{n+1})$. The generalized mid-point family of methods is written as

$$\mathbf{M}(\mathbf{w}_{n+\alpha}, t_{n+\alpha}) \dot{\mathbf{w}}_{n+\alpha} + \mathbf{K}(\mathbf{w}_{n+\alpha}, t_{n+\alpha}) \mathbf{w}_{n+\alpha} = \mathbf{r}(\mathbf{w}_{n+\alpha}, t_{n+\alpha}) \quad (\text{x.5})$$

where

$$\mathbf{w}_{n+1} = (1 - \alpha) \mathbf{w}_n + \alpha \mathbf{w}_{n+1}, \quad (\text{x.6})$$

$$\dot{\mathbf{w}}_{n+\alpha} = \frac{\mathbf{w}_{n+1} - \mathbf{w}_n}{\delta t_n}, \quad (\text{x.7})$$

$$t_{n+1} = t_n + \alpha \delta t_n = t_n + \alpha \delta t . \quad (\text{x.8})$$

Substituting (x.6)-(x.8) into (x.5) we obtain

$$\left(\frac{\mathbf{M}_{n+\alpha}}{\delta t} + \alpha \mathbf{K}_{n+\alpha} \right) \mathbf{w}_{n+1} = \left(\frac{\mathbf{M}_{n+\alpha}}{\delta t} - (1 - \alpha) \mathbf{K}_{n+\alpha} \right) \mathbf{w}_n + \mathbf{r}_{n+\alpha} . \quad (\text{x.9})$$

No calculation of derivatives is necessary for this method. By changing the values of α from 0 to 1, different members of this family of methods are identified. We can obtain a number of well-known difference schemes by choosing the value of α :

$\alpha = 0$	forward difference (Euler) scheme (conditionally stable);
$\alpha = 1/2$	midpoint rule (Crank-Nicolson) scheme (unconditionally stable);
$\alpha = 2/3$	Galerkin method (unconditionally stable);
$\alpha = 1$	backward difference (backward Euler) scheme (conditionally stable).

All, except the forward Euler of the above schemes are implicit (they require matrix inversion for solution). As far as accuracy is concerned the midpoint rule is to be preferred. The generalized midpoint rule conserves linear and quadratic quantities, while the trapezoidal rule conserves only the linear ones.

It must be pointed out that one can expect better results if smaller steps are used. In practice one wishes to take as large a time step as possible to cut down the computational expense. Larger time steps, in addition to decreasing the accuracy of the solution, can introduce some unwanted, numerically induced oscillations into the solution. Thus an estimate of an upper bound on time step proves to be very useful.

The system of equations for incompressible flow is non-linear and therefore an iterative solution is necessary within one time step. The fully discretised using the mentioned in this chapter generalized mid-point family of methods given in Eq.(xxxxxxx) may be written as

$$\left(\frac{\mathbf{M}_{n+\alpha}^p}{\delta t} + \alpha \mathbf{K}_{n+\alpha}^p \right) \mathbf{w}_{n+1}^{p+1} = \left(\frac{\mathbf{M}_{n+\alpha}^p}{\delta t} - (1 - \alpha) \mathbf{K}_{n+\alpha}^p \right) \mathbf{w}_{n+\alpha}^p + \mathbf{r}_{n+\alpha}^p. \quad (\text{x.9})$$

where p represents the iteration number. Above equation may be solved until the norm $\left| \mathbf{w}_{n+1}^{p+1} - \mathbf{w}_{n+1}^p \right|$ falls below an acceptable tolerance. To speed up convergence within one time step the Newton-Raphson method may be used [Hua1999].

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 discretised 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.

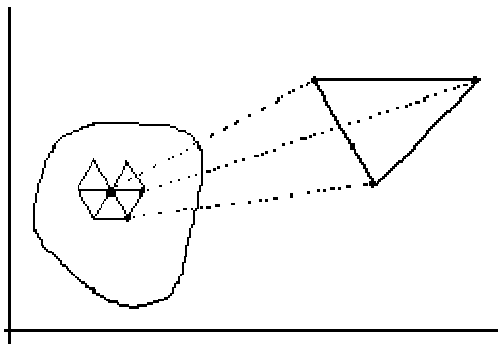


Figure 3.1: 2D triangular mesh.

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

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

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

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

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

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

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

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

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

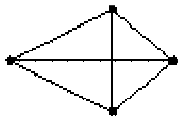
 <p>3-node triangle, linear approximation</p>	 <p>6-node triangle, quadratic approximation</p>



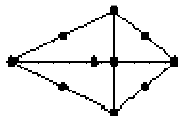
4-node quadrilateral, bi-linear approximation



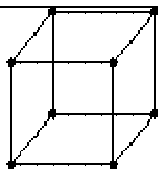
8-node quadrilateral, bi-quadratic approximation (serendipity element)



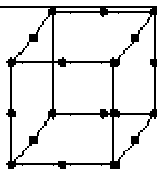
4-node tetrahedron, linear approximation



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

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

defined within a domain Ω and with boundary conditions specified at the boundary of Γ . The boundary conditions typically specify u 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:

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

The PDE will be satisfied in the weak sense provided

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

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

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

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

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

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:

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

where

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

and

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

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

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

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

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

which is equivalent to

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

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

The Crank–Nicolson approximation uses the algorithm:

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

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

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

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

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

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

The PDE will be satisfied in the weak sense provided

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

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

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

The weak form of the PDE thus gives

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

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

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

where

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

and

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

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

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

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

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

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

$$\begin{aligned}
 & \int_{\Omega} T_j \frac{\partial u}{\partial t} dV + \int_{\Omega} T_j L(u) dV = \\
 & = \int_{\Omega} T_j \frac{\partial u}{\partial t} dV + \int_{\Omega} T_j L_1(u) dV + \int_{\Omega} \nabla T_j L_2(u) dV - \int_{\Omega} L_2(u) \nabla T_j dV = \quad . \quad (\text{x.y}) \\
 & = \int_{\Omega} T_j \frac{\partial u}{\partial t} dV + \int_{\Omega} T_j L_1(u) dV + \int_{\partial\Omega} \mathbf{n} T_j L_2(u) dS - \int_{\Omega} L_2(u) \nabla T_j dV = \int_{\Omega} T_j f dV
 \end{aligned}$$

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

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

where

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

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

and

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

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)

$$G(u) = g \quad (x,y)$$

where the value of variable is prescribed;

- b) natural (Neumann)

$$S(u) = s \quad (x,y)$$

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

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

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

for midside nodes $\xi_i = 0$

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

for midside nodes $\eta_i = 0$

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

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:

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

and

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

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

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

may be obtained by the chain rule of differentiation

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

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

where

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

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

The terms

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

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

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

which is termed the Jacobian matrix \mathbf{J} . The inverse of the Jacobian is then evaluated

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

An element area of the element is given as

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

For an isoparametric element we have

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

and

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

X.10.3. Numerical integration

We can adopt a numerical integration procedure to evaluate such integrals

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

or

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

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

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

r	$\bar{\xi}_i$	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:

$$\begin{aligned}x &= \sum_{i=1}^3 L_i x_i \\y &= \sum_{i=1}^3 L_i y_i \\1 &= \sum_{i=1}^3 L_i\end{aligned} \tag{x.y}$$

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

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

in which

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

and

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

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

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

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 :

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

where

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

and

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

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

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

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

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

It is easy to verify that corner nodes shape functions are

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

And mid-sides nodes shape functions are as follows

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

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

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

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:

$$\begin{aligned}x &= \sum_{i=1}^4 L_i x_i \\y &= \sum_{i=1}^4 L_i y_i \\z &= \sum_{i=1}^4 L_i z_i\end{aligned} \quad (\text{x.y})$$

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

Solving Eq. (xxxx) gives

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

in which

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

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

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

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 :

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

where

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

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

and

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

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

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

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

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

It is easy to verify that corner nodes shape functions are

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

And mid-sides nodes shape functions are as follows

$$N_{1100} = L_1L_2, N_{1010} = L_1L_3, N_{1001} = L_1L_4, \\ N_{0110} = L_2L_3, N_{0101} = L_2L_4, N_{0011} = L_3L_4 \quad (\text{x.y})$$

The following exact integration expression is valid

$$\iiint_V L_1^a L_2^b L_3^c L_4^d dx dy dz = \frac{a! b! c! d!}{(a+b+c+3)!} 6V . \quad (\text{x.y})$$

(x.y)

X.12. Finite element model of non-isothermal flow

In order to demonstrate temporal discretization we use equation for Newtonian three dimensional flow. Beginning with the dimensionless incompressible Navier-Stokes equations

$$\left(\frac{\partial v_k}{\partial t} + v_1 \frac{\partial v_k}{\partial x_1} + v_2 \frac{\partial v_k}{\partial x_2} + v_3 \frac{\partial v_k}{\partial x_3} \right) + \frac{\partial p}{\partial x_k} - Pr \nabla^2 v_k = f_k \quad (\text{x.y})$$

for $k=1,2,3$ (three dimensional problem) and the continuity equation (the conservation of mass)

$$\frac{\partial v_1}{\partial x_1} + \frac{\partial v_2}{\partial x_2} + \frac{\partial v_3}{\partial x_3} = 0. \quad (\text{x.y})$$

For incompressible flow under non-isothermal conditions, the energy conservation equation must also be included in the formulation

$$\frac{\partial T}{\partial t} + v_1 \frac{\partial T}{\partial x_1} + v_2 \frac{\partial T}{\partial x_2} + v_3 \frac{\partial T}{\partial x_3} = \nabla^2 T + PrEc \eta \Phi . \quad (5.8)$$

We have to notice that considered problem is non-linear. If we linearize the governing equations by approximating the nonlinear terms (e.g. convective terms) we can solve the set of equations using any of iterative scheme. Using naïve linearizing method the equations can be rewritten as

$$\left(\frac{\partial v_k}{\partial t} + \tilde{v}_1 \frac{\partial v_k}{\partial x_1} + \tilde{v}_2 \frac{\partial v_k}{\partial x_2} + \tilde{v}_3 \frac{\partial v_k}{\partial x_3} \right) + \frac{\partial p}{\partial x_k} - Pr \nabla^2 v_k = f_k \quad (x.y)$$

$$\frac{\partial v_1}{\partial x_1} + \frac{\partial v_2}{\partial x_2} + \frac{\partial v_3}{\partial x_3} = 0. \quad (\text{x.y})$$

$$\frac{\partial T}{\partial t} + \tilde{v}_1 \frac{\partial T}{\partial x_1} + \tilde{v}_2 \frac{\partial T}{\partial x_2} + \tilde{v}_3 \frac{\partial T}{\partial x_3} = \nabla^2 T + PrEc \eta \Phi. \quad (5.8)$$

where \tilde{v}_k , for $k=1,2,3$ denotes the value of velocity components from previous iteration step. To achieve better approximation and convergence of iterative process we can use one of the method described in previous Chapter.

The following shape functions will be used to approximate the field variables

$$v_1 = \sum_{i=1}^{nu} N_i v_{1i}, \quad v_2 = \sum_{i=1}^{nu} N_i v_{2i}, \quad v_3 = \sum_{i=1}^{nu} N_i v_{3i}, \quad p = \sum_{i=1}^{np} NP_i p_i, \quad (\text{x.y})$$

$$T = \sum_{i=1}^{nc} NT_i c_i .$$

Applying the Galerkin weighted residual procedure to our governing equations using the above shape functions we obtain the Galerkin finite element method (GFEM) equations discretized in space. This is illustrated in the following lines.

Substitution of (x,y) into the governing equation yields

$$\begin{aligned}
& \int_{\Omega} N_i \left(N_j \dot{v}_{kj} + N_l \tilde{v}_{1l} \frac{\partial N_j}{\partial x_1} v_{kj} + N_l \tilde{v}_{2l} \frac{\partial N_j}{\partial x_2} v_{kj} + N_l \tilde{v}_{3l} \frac{\partial N_j}{\partial x_3} v_{kj} \right) d\Omega \\
& + \int_{\Omega} N_i \frac{\partial NP_j}{\partial x_k} p_j d\Omega + \\
& - \int_{\Omega} Pr N_i \left(\frac{\partial^2 N_j}{\partial x_1^2} v_{kj} + \frac{\partial^2 N_j}{\partial x_2^2} v_{kj} + \frac{\partial^2 N_j}{\partial x_3^2} v_{kj} \right) d\Omega \quad \cdot \quad (x.y) \\
& = \int_{\Omega} N_i f_k d\Omega
\end{aligned}$$

and

$$\begin{aligned}
& \int_{\Omega} N_i \left(NT_j \dot{c}_j + N_l \tilde{v}_{1l} \frac{\partial NT_j}{\partial x_1} c_j + N_l \tilde{v}_{2l} \frac{\partial NT_j}{\partial x_2} c_j + N_l \tilde{v}_{3l} \frac{\partial NT_j}{\partial x_3} c_j \right) d\Omega \\
& - \int_{\Omega} N_i \left(\frac{\partial^2 NT_j}{\partial x_1^2} c_j + \frac{\partial^2 NT_j}{\partial x_2^2} c_j + \frac{\partial^2 NT_j}{\partial x_3^2} c_j \right) d\Omega \quad . \quad (\text{x.y}) \\
& = \int_{\Omega} Pr Ec N_i \eta \Phi d\Omega
\end{aligned}$$

In the continuity equation we must use the pressure shape functions NP_i for the weighting functions as the continuity equation will only be enforced at the pressure modes

$$\int_{\Omega} NP_i \frac{\partial N_j}{\partial x_1} v_{1i} d\Omega + \int_{\Omega} NP_i \frac{\partial N_j}{\partial x_2} v_{2i} d\Omega + \int_{\Omega} NP_i \frac{\partial N_j}{\partial x_3} v_{3i} d\Omega = 0 \quad (\text{x.y})$$

By applying integration by parts to the integral expression of equations, we can obtain expressions containing lower-order derivatives, and hence we can use approximating functions with lower-order interelement continuity. For all second-order terms and some first-order terms, the integration is done by parts. This gives:

$$\int_{\Omega} T_i (D_m Expr) dV = - \int_{\Omega} (D_m T_i) Expr dV + \int_{\Gamma} T_i n_m Expr dS \quad (\text{x.y})$$

where $D_m = \frac{\partial}{\partial x_m}$ and $Expr$ is expression.

When integration by parts is possible, it also offers a convenient way to introduce the natural boundary conditions that must be satisfied on some portion of the boundary. Although the boundary terms containing the natural boundary conditions appear in the equations for each element, in the assembly

of the element equations only the boundary elements give nonvanishing contributions.

For second order derivative using formula (XXX) we obtain

$$\int_{\Omega} N_i \frac{\partial^2 N_j}{\partial x_s^2} v_{kj} d\Omega = \int_{\Gamma} N_i \frac{\partial N_j}{\partial x_s} n_s v_{kj} d\Gamma - \int_{\Omega} \frac{\partial N_i}{\partial x_s} \frac{\partial N_j}{\partial x_s} v_{kj} d\Omega \quad (\text{x.y})$$

When integration by parts is applied to the momentum equation we obtain

$$\begin{aligned}
& \int_{\Omega} N_i \left(N_j \dot{v}_{kj} + N_l \tilde{v}_{1l} \frac{\partial N_j}{\partial x_1} v_{kj} + N_l \tilde{v}_{2l} \frac{\partial N_j}{\partial x_2} v_{kj} + N_l \tilde{v}_{3l} \frac{\partial N_j}{\partial x_3} v_{kj} \right) d\Omega \\
& + \int_{\Omega} N_i \frac{\partial NP_j}{\partial x_k} p_j d\Omega + \\
& + \int_{\Omega} Pr \left(\frac{\partial N_i}{\partial x_1} \frac{\partial N_j}{\partial x_1} v_{kj} + \frac{\partial N_i}{\partial x_2} \frac{\partial N_j}{\partial x_2} v_{kj} + \frac{\partial N_i}{\partial x_3} \frac{\partial N_j}{\partial x_3} v_{kj} \right) d\Omega \quad . \quad (x.y) \\
& - \int_{\Gamma} Pr N_i \left(\frac{\partial N_j}{\partial x_1} n_1 v_{kj} + \frac{\partial N_j}{\partial x_2} n_2 v_{kj} + \frac{\partial N_j}{\partial x_3} n_3 v_{kj} \right) d\Gamma \\
& = \int_{\Omega} N_i f_k d\Omega
\end{aligned}$$

and when we use it for the energy conservation equation this gives

$$\begin{aligned}
& \int_{\Omega} N_i \left(NT_j \dot{c}_j + N_l \tilde{v}_{1l} \frac{\partial NT_j}{\partial x_1} c_j + N_l \tilde{v}_{2l} \frac{\partial NT_j}{\partial x_2} c_j + N_l \tilde{v}_{3l} \frac{\partial NT_j}{\partial x_3} c_j \right) d\Omega \\
& + \int_{\Omega} \left(\frac{\partial N_i}{\partial x_1} \frac{\partial NT_j}{\partial x_1} c_j + \frac{\partial N_i}{\partial x_2} \frac{\partial NT_j}{\partial x_2} c_j + \frac{\partial N_i}{\partial x_3} \frac{\partial NT_j}{\partial x_3} c_j \right) d\Omega \\
& - \int_{\Gamma} N_i \left(\frac{\partial NT_j}{\partial x_1} n_1 c_j + \frac{\partial NT_j}{\partial x_2} n_2 c_j + \frac{\partial NT_j}{\partial x_3} n_3 c_j \right) d\Gamma \\
& = \int_{\Omega} Pr Ec N_i \eta \Phi d\Omega
\end{aligned} \tag{x.y}$$

We may write the final set of the GFEM equations discretized in space in a fully coupled form as follows

$$\begin{bmatrix} \mathbf{M1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M2} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{M3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{MT} \end{bmatrix} \cdot \begin{bmatrix} \dot{\mathbf{v}}_1 \\ \dot{\mathbf{v}}_2 \\ \dot{\mathbf{v}}_3 \\ \dot{\mathbf{p}} \\ \dot{\mathbf{c}} \end{bmatrix} + \begin{bmatrix} \mathbf{K11} & \mathbf{0} & \mathbf{0} & \mathbf{K14} & \mathbf{0} \\ \mathbf{0} & \mathbf{K22} & \mathbf{0} & \mathbf{K24} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{K33} & \mathbf{K34} & \mathbf{0} \\ \mathbf{K41} & \mathbf{K42} & \mathbf{K43} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K55} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{p} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \mathbf{f1} \\ \mathbf{f2} \\ \mathbf{f3} \\ \mathbf{0} \\ \mathbf{fT} \end{bmatrix} \quad (\text{x.y})$$

where

$$\mathbf{M1}_{ij} = \mathbf{M2}_{ij} = \mathbf{M3}_{ij} = \int_{\Omega} N_i N_j d\Omega \quad (\text{x.y})$$

$$\mathbf{MT}_{ij} = \int_{\Omega} N_i N T_j d\Omega \quad (\text{x.y})$$

$$\begin{aligned}
\mathbf{K11}_{ij} &= \int_{\Omega} N_i \left(N_l \tilde{v}_{1l} \frac{\partial N_j}{\partial x_1} + N_l \tilde{v}_{2l} \frac{\partial N_j}{\partial x_2} + N_l \tilde{v}_{3l} \frac{\partial N_j}{\partial x_3} \right) d\Omega \\
&+ \int_{\Omega} Pr \left(\frac{\partial N_i}{\partial x_1} \frac{\partial N_j}{\partial x_1} + \frac{\partial N_i}{\partial x_2} \frac{\partial N_j}{\partial x_2} + \frac{\partial N_i}{\partial x_3} \frac{\partial N_j}{\partial x_3} \right) d\Omega \quad . \quad (\text{x.y}) \\
&- \int_{\Gamma} Pr N_i \left(\frac{\partial N_j}{\partial x_1} n_1 + \frac{\partial N_j}{\partial x_2} n_2 + \frac{\partial N_j}{\partial x_3} n_3 \right) d\Gamma
\end{aligned}$$

$$\mathbf{K14}_{ij} = \int_{\Omega} N_i \frac{\partial NP_j}{\partial x_1} d\Omega \quad (\text{x.y})$$

$$\mathbf{K22}_{ij} = \mathbf{K33}_{ij} = \mathbf{K11}_{ij} \quad (\text{x.y})$$

$$\mathbf{K24}_{ij} = \mathbf{K34}_{ij} = \mathbf{K14}_{ij} \quad (\text{x.y})$$

$$\mathbf{K41}_{ij} = \int_{\Omega} NP_i \frac{\partial N_j}{\partial x_1} v_{1i} d\Omega, \quad \mathbf{K42}_{ij} = \int_{\Omega} NP_i \frac{\partial N_j}{\partial x_2} v_{2i} d\Omega, \quad (\text{x.y})$$
$$\mathbf{K43}_{ij} = \int_{\Omega} NP_i \frac{\partial N_j}{\partial x_3} v_{3i} d\Omega$$

$$\begin{aligned}
\mathbf{K55}_{ij} = & \int_{\Omega} N_i \left(N_l v_{1l} \frac{\partial NT_j}{\partial x_1} + N_l \tilde{v}_{2l} \frac{\partial NT_j}{\partial x_2} + N_l \tilde{v}_{3l} \frac{\partial NT_j}{\partial x_3} \right) d\Omega \\
& + \int_{\Omega} \left(\frac{\partial N_i}{\partial x_1} \frac{\partial NT_j}{\partial x_1} + \frac{\partial N_i}{\partial x_2} \frac{\partial NT_j}{\partial x_2} + \frac{\partial N_i}{\partial x_3} \frac{\partial NT_j}{\partial x_3} \right) d\Omega \quad . \quad (\text{x.y}) \\
& - \int_{\Gamma} N_i \left(\frac{\partial NT_j}{\partial x_1} n_1 + \frac{\partial NT_j}{\partial x_2} n_2 + \frac{\partial NT_j}{\partial x_3} n_3 \right) d\Gamma
\end{aligned}$$

$$\begin{aligned}
\mathbf{f1}_i = \int_{\Omega} N_i f_1 d\Omega, \quad \mathbf{f2}_i = \int_{\Omega} N_i f_2 d\Omega, \quad \mathbf{f3}_i = \int_{\Omega} N_i f_3 d\Omega, \\
\mathbf{fT}_i = \int_{\Omega} Pr Ec N_i \eta \Phi d\Omega \quad (\text{x.y})
\end{aligned}$$

X.13. Finite element model of non-Newtonian flow

In this chapter a method for analyzing transient non-Newtonian flow is presented. We begin with the Navier-Stokes equation for non-Newtonian fluids represented by conservation of momentum

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} + \nabla p - Pr \nabla \cdot \mathbf{S} = \mathbf{f} \quad (\text{x.y})$$

and conservation of mass for incompressible fluid

$$\nabla \cdot \mathbf{v} = 0. \quad (\text{x.y})$$

To solve transient flow of non-Newtonian fluids the above equations are linearized. Let r denote the number of linearization iterations and the

superscript n denotes the previous time step and $n+1$ denotes the current time step. The linearization is carried out as follows:

$$\nabla \cdot (\mathbf{v})_{n+1}^r = 0 \quad (\text{x.y})$$

$$\begin{aligned} & \left(\frac{(\mathbf{v})_{n+1}^r - \mathbf{v}_n}{\delta t} + (\mathbf{v})_{n+1}^{r-1} \cdot \nabla (\mathbf{v})_{n+1}^r \right) \\ & + \nabla (p)_{n+1}^r - Pr(\eta)_{n+1}^{r-1} \nabla^2 (\mathbf{v})_{n+1}^r \\ & - Pr \nabla (\eta)_{n+1}^{r-1} \left(\nabla (\mathbf{v})_{n+1}^{r-1} + \left(\nabla (\mathbf{v})_{n+1}^{r-1} \right)^T \right) = (\mathbf{f})_{n+1}^{r-1} \end{aligned} \quad (\text{x.y})$$

Rearranging terms we obtain:

$$\nabla \cdot (\mathbf{v})_{n+1}^r = 0 \quad (\text{x.y})$$

$$\begin{aligned}
& \left(\frac{(\mathbf{v})_{n+1}^r}{\delta t} + (\mathbf{v})_{n+1}^{r-1} \cdot \nabla (\mathbf{v})_{n+1}^r \right) + \nabla (p)_{n+1}^r - Pr(\eta)_{n+1}^{r-1} \nabla^2 (\mathbf{v})_{n+1}^r = \\
& = (\mathbf{f})_{n+1}^{r-1} + \frac{\mathbf{v}_n}{\delta t} + Pr \nabla (\eta)_{n+1}^{r-1} \left(\nabla (\mathbf{v})_{n+1}^{r-1} + \left(\nabla (\mathbf{v})_{n+1}^{r-1} \right)^T \right)
\end{aligned} \tag{x.y}$$

In the above:

$$(\mathbf{v})_{n+1}^0 = \mathbf{v}_n, \quad (p)_{n+1}^0 = p_n, \quad (\eta)_{n+1}^0 = \eta_n. \tag{x.y}$$

The governing equations for non-Newtonian flow summarized above must first be discretized in space. The following shape functions will be used to approximate the field variables

$$(v_1)_{n+1}^r = \sum_{i=1}^{nu} N_i \{ (v_{1i})_{n+1}^r \}, \quad (v_2)_{n+1}^r = \sum_{i=1}^{nu} N_i \{ (v_{2i})_{n+1}^r \}, \tag{x.y}$$

$$(v_3)^r_{n+1} = \sum_{i=1}^{nu} N_i \{(v_{3i})^r_{n+1}\}, (p)^r_{n+1} = \sum_{i=1}^{nu} NP_i \{(p_i)^r_{n+1}\}$$

Due to time step process and iteration technique applied in algorithm we have to use shape functions with additional indexes.

Substitution of (x.y) into the virtual work equation yields

$$\int_{\Omega} NP_i \frac{\partial N_j}{\partial x_1} \{(v_{1j})^r_{n+1}\} d\Omega + \int_{\Omega} NP_i \frac{\partial N_j}{\partial x_2} \{(v_{2j})^r_{n+1}\} d\Omega + \int_{\Omega} NP_i \frac{\partial N_j}{\partial x_3} \{(v_{3j})^r_{n+1}\} d\Omega = 0 \quad (x.y)$$

$$\begin{aligned}
& \int_{\Omega} N_i \left(\frac{N_j}{\delta} + (v_1)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_1} + (v_2)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_2} + (v_3)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_3} \right) \{(v_{1j})_{n+1}^r\} d\Omega \\
& + \int_{\Omega} N_i \frac{\partial NP_j}{\partial x_1} \{(p_j)_{n+1}^r\} d\Omega \\
& - \int_{\Omega} N_i (\eta)_{n+1}^{r-1} \left(\frac{\partial^2 N_j}{\partial x_1^2} + \frac{\partial^2 N_j}{\partial x_2^2} + \frac{\partial^2 N_j}{\partial x_3^2} \right) \{(v_{1j})_{n+1}^r\} d\Omega = \\
& = \int_{\Omega} N_i \left((f_1)_{n+1}^{r-1} + \frac{(v_1)_n}{\delta} + 2Pr \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_1} \frac{\partial (v_1)_{n+1}^{r-1}}{\partial x_1} \right) d\Omega \tag{x.y} \\
& + \int_{\Omega} N_i \left(Pr \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_2} \left(\frac{\partial (v_1)_{n+1}^{r-1}}{\partial x_2} + \frac{\partial (v_2)_{n+1}^{r-1}}{\partial x_1} \right) \right) d\Omega \\
& + \int_{\Omega} N_i \left(Pr \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_3} \left(\frac{\partial (v_1)_{n+1}^{r-1}}{\partial x_3} + \frac{\partial (v_3)_{n+1}^{r-1}}{\partial x_1} \right) \right) d\Omega
\end{aligned}$$

$$\begin{aligned}
& \int_{\Omega} N_i \left(\frac{N_j}{\delta} + (v_1)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_1} + (v_2)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_2} + (v_3)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_3} \right) \left\{ (v_{2j})_{n+1}^r \right\} d\Omega \\
& + \int_{\Omega} N_i \frac{\partial NP_j}{\partial x_2} \left\{ (p_j)_{n+1}^r \right\} d\Omega \\
& - \int_{\Omega} N_i (\eta)_{n+1}^{r-1} \left(\frac{\partial^2 N_j}{\partial x_1^2} + \frac{\partial^2 N_j}{\partial x_2^2} + \frac{\partial^2 N_j}{\partial x_3^2} \right) \left\{ (v_{2j})_{n+1}^r \right\} d\Omega = \\
& = \int_{\Omega} N_i \left((f_2)_{n+1}^{r-1} + \frac{(v_2)_n}{\delta} + Pr \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_1} \left(\frac{\partial (v_1)_{n+1}^{r-1}}{\partial x_2} + \frac{\partial (v_2)_{n+1}^{r-1}}{\partial x_1} \right) \right) d\Omega \quad (x.y) \\
& + \int_{\Omega} Pr N_i \left(2 \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_2} \frac{\partial (v_2)_{n+1}^{r-1}}{\partial x_2} \right) d\Omega \\
& \int_{\Omega} Pr N_i \left(\frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_3} \left(\frac{\partial (v_2)_{n+1}^{r-1}}{\partial x_3} + \frac{\partial (v_3)_{n+1}^{r-1}}{\partial x_2} \right) \right) d\Omega
\end{aligned}$$

$$\begin{aligned}
& \int_{\Omega} N_i \left(\frac{N_j}{\delta} + (v_1)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_1} + (v_2)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_2} + (v_3)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_3} \right) \left\{ (v_{3j})_{n+1}^r \right\} d\Omega \\
& + \int_{\Omega} N_i \frac{\partial NP_j}{\partial x_3} \left\{ (p_j)_{n+1}^r \right\} d\Omega \\
& - \int_{\Omega} Pr N_i (\eta)_{n+1}^{r-1} \left(\frac{\partial^2 N_j}{\partial x_1^2} + \frac{\partial^2 N_j}{\partial x_2^2} + \frac{\partial^2 N_j}{\partial x_3^2} \right) \left\{ (v_{3j})_{n+1}^r \right\} d\Omega = \\
& = \int_{\Omega} N_i \left((f_3)_{n+1}^{r-1} + \frac{(v_3)_n}{\delta} + Pr \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_1} \left(\frac{\partial (v_3)_{n+1}^{r-1}}{\partial x_1} + \frac{\partial (v_1)_{n+1}^{r-1}}{\partial x_3} \right) \right) d\Omega \tag{x.y} \\
& + \int_{\Omega} Pr N_i \left(\frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_2} \left(\frac{\partial (v_3)_{n+1}^{r-1}}{\partial x_2} + \frac{\partial (v_2)_{n+1}^{r-1}}{\partial x_3} \right) \right) d\Omega \\
& + \int_{\Omega} Pr N_i \left(2 \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_3} \frac{\partial (v_3)_{n+1}^{r-1}}{\partial x_3} \right) d\Omega
\end{aligned}$$

By applying integration by parts to the integral expression of equations, we can obtain following expressions containing lower-order derivatives

$$\begin{aligned}
& \int_{\Omega} N_i \left(\frac{N_j}{\delta r} + (v_1)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_1} + (v_2)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_2} + (v_3)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_3} \right) \{(v_{1j})_{n+1}^r\} d\Omega \\
& + \int_{\Omega} N_i \frac{\partial NP_j}{\partial x_1} \{(p_j)_{n+1}^r\} d\Omega \\
& - Pr \int_{\Omega} (\eta)_{n+1}^{r-1} \left(\frac{\partial N_i}{\partial x_1} \frac{\partial N_j}{\partial x_1} + \frac{\partial N_i}{\partial x_2} \frac{\partial N_j}{\partial x_2} + \frac{\partial N_i}{\partial x_3} \frac{\partial N_j}{\partial x_3} \right) \{(v_{1j})_{n+1}^r\} d\Omega = \\
& = \int_{\Omega} N_i \left((f_1)_{n+1}^{r-1} + \frac{(v_1)_n}{\delta r} + 2Pr \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_1} \frac{\partial (v_1)_{n+1}^{r-1}}{\partial x_1} \right) d\Omega \tag{x.y} \\
& + Pr \int_{\Omega} N_i \left(\frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_2} \left(\frac{\partial (v_1)_{n+1}^{r-1}}{\partial x_2} + \frac{\partial (v_2)_{n+1}^{r-1}}{\partial x_1} \right) \right) d\Omega \\
& + Pr \int_{\Omega} N_i \left(\frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_3} \left(\frac{\partial (v_1)_{n+1}^{r-1}}{\partial x_3} + \frac{\partial (v_3)_{n+1}^{r-1}}{\partial x_1} \right) \right) d\Omega \\
& - \int_{\Gamma} N_i \frac{\partial (v_1)_{n+1}^r}{\partial \mathbf{n}} d\Gamma
\end{aligned}$$

$$\begin{aligned}
& \int_{\Omega} N_i \left(\frac{N_j}{\delta t} + (v_1)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_1} + (v_2)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_2} + (v_3)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_3} \right) \left\{ (v_{2j})_{n+1}^r \right\} d\Omega \\
& + \int_{\Omega} N_i \frac{\partial NP_j}{\partial x_2} \left\{ (p_j)_{n+1}^r \right\} d\Omega \\
& - Pr \int_{\Omega} (\eta)_{n+1}^{r-1} \left(\frac{\partial N_i}{\partial x_1} \frac{\partial N_j}{\partial x_1} + \frac{\partial N_i}{\partial x_2} \frac{\partial N_j}{\partial x_2} + \frac{\partial N_i}{\partial x_3} \frac{\partial N_j}{\partial x_3} \right) \left\{ (v_{2j})_{n+1}^r \right\} d\Omega = \\
& = \int_{\Omega} N_i \left((f_2)_{n+1}^{r-1} + \frac{(v_2)_n}{\delta t} + Pr \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_1} \left(\frac{\partial (v_1)_{n+1}^{r-1}}{\partial x_2} + \frac{\partial (v_2)_{n+1}^{r-1}}{\partial x_1} \right) \right) d\Omega \quad (\text{x.y}) \\
& + Pr \int_{\Omega} N_i \left(2 \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_2} \frac{\partial (v_2)_{n+1}^{r-1}}{\partial x_2} \right) d\Omega \\
& + Pr \int_{\Omega} N_i \left(\frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_3} \left(\frac{\partial (v_2)_{n+1}^{r-1}}{\partial x_3} + \frac{\partial (v_3)_{n+1}^{r-1}}{\partial x_2} \right) \right) d\Omega \\
& - \int_{\Gamma} N_i \frac{\partial (v_2)_{n+1}^r}{\partial \mathbf{n}} d\Gamma
\end{aligned}$$

$$\begin{aligned}
& \int_{\Omega} N_i \left(\frac{N_j}{\delta t} + (v_1)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_1} + (v_2)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_2} + (v_3)_{n+1}^{r-1} \frac{\partial N_j}{\partial x_3} \right) \left\{ (v_{3j})_{n+1}^r \right\} d\Omega \\
& + \int_{\Omega} N_i \frac{\partial NP_j}{\partial x_3} \left\{ (p_j)_{n+1}^r \right\} d\Omega \\
& - Pr \int_{\Omega} (\eta)_{n+1}^{r-1} \left(\frac{\partial N_i}{\partial x_1} \frac{\partial N_j}{\partial x_1} + \frac{\partial N_i}{\partial x_2} \frac{\partial N_j}{\partial x_2} + \frac{\partial N_i}{\partial x_3} \frac{\partial N_j}{\partial x_3} \right) \left\{ (v_{3j})_{n+1}^r \right\} d\Omega = \\
& = \int_{\Omega} N_i \left((f_3)_{n+1}^{r-1} + \frac{(v_3)_n}{\delta t} + Pr \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_1} \left(\frac{\partial (v_3)_{n+1}^{r-1}}{\partial x_1} + \frac{\partial (v_1)_{n+1}^{r-1}}{\partial x_3} \right) \right) d\Omega \quad (x.y) \\
& + Pr \int_{\Omega} N_i \left(\frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_2} \left(\frac{\partial (v_3)_{n+1}^{r-1}}{\partial x_2} + \frac{\partial (v_2)_{n+1}^{r-1}}{\partial x_3} \right) \right) d\Omega \\
& + Pr \int_{\Omega} N_i \left(2 \frac{\partial (\eta)_{n+1}^{r-1}}{\partial x_3} \frac{\partial (v_3)_{n+1}^{r-1}}{\partial x_3} \right) d\Omega \\
& - \int_{\Gamma} N_i \frac{\partial (v_3)_{n+1}^r}{\partial \mathbf{n}} d\Gamma
\end{aligned}$$

To complete calculation use technique presented in Section XXX.

(x.y)

References

[Wuj1998] Wu J., Sheng-Tao Y., Bo-Nan J.,
Simulation of two-fluid flows by the least-squares finite element method using
a Continuum Surface Tension model,
Int. J. Numer. Meth. Engng. **42**, 583-600, 1998.

[Bra1992] Brackbill J.U., Kothe D.B., Zemach C.,
A continuum method for modeling surface tension,
J. Comput. Phys. **100**, 335-354, 1992.

[Hue1975] Huebner K.H.,
The Finite Element Method for Engineers,
Wiley, Toronto, 1975.

[Tay1981] Taylor C., Hughes T.G.,
Finite Element Programming of the Navier-Stokes Equations,
Pineridge, Swansea, 1981.

[Hin1979] Hinton E., Owen D.R.J.,
An Introduction to Finite Element Computations,
Pineridge, Swansea, 1979.

[Hua1999] Hou-Cheng Huang, Zheng-Hua Li and Asif S. Usmani,
Finite Element Analysis of Non-Newtonian Flow,
Springer-Verlang, London, 1999.

[Zie2000a] 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.

[Kar2001] Hvistendahl Karlsen K., Lie K.-A., Natvig J. R., Nordhaug H. F.,
and Dahle H. K., Operator Splitting Methods for Systems of Convection–
Diffusion equations: Nonlinear Error Mechanisms and Correction Strategies,
Journal of Computational Physics 173, 636–663 (2001).

[Tur1996] Turek S., A comparative study of time-stepping techniques for the
incompressible Navier-Stokes equations: From fully implicit non-linear
schemes to semi-implicit projection methods. Int. J. Num. Meth. In Fluids, 22,
pp.987-1011, 1996.

[Bur1985] Burden R.L., Faires J.D., Numerical Analysis, Third Edition, PWS-
Kent, Boston, 1985.

[Str1999] Stręk T. Zastosowanie Maple do wyznaczania wielomianów wyższych stopni dla trójkątnych elementów skończonych, *Pro Dialog* , 8, 1999, p. 63-68.

[Sil1969] Silvester P. Higher-Order Polynomial Triangular Finite Elements for Potential Problems, *Int. J. Eng. Sci.* 7, pp. 849-861, 1966.

[Lan0000] Lantangen H.P., Mardal K.-A., Winther R., Numerical methods for Incompressible Viscous Flow, NO INFORMATION !!!