Chapter 6

COMPUTER IMPLEMENTATION OF FINITE ELEMENT METHODS

The computer implementation of finite element methods entails various practical aspects that have a well-developed state-of-the-art and merit special attention. The detailed exposition of all these implementational aspects is beyond the scope of these notes. However, some of these aspects are discussed below.

6.1 Numerical integration of element matrices

All finite element methods, with the exception of those which are derived from point-collocation, are based on weak forms that are expressed as integrals the domain Ω and its boundary $\partial\Omega$ (or parts of it), see, e.g., equations (3.14) and (3.57). These integrals are ultimately evaluated as sums over integrals at the single element level, i.e., over the typical element domain Ω^e and its boundary $\partial\Omega^e$ (or parts of it). Therefore, it is important to be able to evaluate such element-wise integrals either exactly or by approximate numerical techniques. The latter case is the subject of the remainder of this section.

By way of background, consider a one-dimensional integral

$$I \;=\; \int_a^b f(x)\,dx \;, \tag{6.1}$$

where f is a real-valued function and a, b are constant integration limits. The domain (a, b) of integration can be readily mapped into the domain $(-1, 1)$ relative to a new coordinate ξ

by merely setting

$$x = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\xi \ . \tag{6.2}$$

This transformation is one-to-one and onto as long as $a \neq b$ and it establishes symmetry of the domain relative to the origin. Taking into account the preceding transformation, one may write the original integral as

$$I = \int_a^b f(x)\,dx = \int_{-1}^{1} f\Big(\frac{1}{2}(a+b) + \frac{1}{2}(b-a)\xi\Big)\frac{1}{2}(b-a)\,d\xi = \int_{-1}^{1} g(\xi)\,d\xi \ . \tag{6.3}$$

Now, the integral is evaluated numerically as

$$I = \int_{-1}^{1} g(\xi)\,d\xi \doteq \sum_{k=1}^{L} w_k g(\xi_k) \ . \tag{6.4}$$

Here, ξ_k are the *sampling points* and w_k are the corresponding *weights*. Equation $(6.4)_2$ encompasses virtually all the numerical integration methods used in one-dimensional finite elements.

Examples:

(a) The classical *trapezoidal rule* can be expressed, in view of equation $(6.4)_2$, as

$$\int_{-1}^{1} g(\xi)\,d\xi \doteq \sum_{k=1}^{2} w_k g(\xi_k) \ ,$$

where $w_1 = w_2 = 1$, $\xi_1 = -1$ and $\xi_2 = 1$. The trapezoidal rule integrates exactly all polynomials up to degree $q = 1$.

(b) *Simpson's rule* is written as

$$\int_{-1}^{1} g(\xi)\,d\xi \doteq \sum_{k=1}^{3} w_k g(\xi_k) \ ,$$

where $w_1 = w_3 = \frac{1}{3}$, $w_2 = \frac{4}{3}$, and $\xi_1 = -1$, $\xi_2 = 0$, and $\xi_3 = 1$. Simpson's rule integrates exactly all polynomials up to degree $q = 3$. Notice that Simpson's rule attains accuracy of two additional orders of magnitude as compared to the trapezoidal rule despite only adding one extra function evaluation. This is due to the optimal placement $\xi_2 = 0$ of the interior sampling point.

The trapezoidal and Simpson rules are special cases of the *Newton-Cotes closed* numerical integration formulae. Given that function evaluations are computationally expensive and need to be repeated for all element-based integrals, one may reasonably ask if the Newton-Cotes formulae are optimal, i.e., whether they furnish the maximum possible polynomial degree of accuracy for the given cost. It turns out that the answer to this question is, in fact, negative. This point can be argued as follows: recalling equation $(6.4)_2$, it is clear that the right-hand side contains L weights and L coordinates of the sampling points to be determined, hence a total of $2L$ "tunable" parameters. Suppose that one wishes to exactly integrate with such a formula a polynomial of degree q, expressed as

$$P(\xi) \;=\; a_0 + a_1\xi + \ldots + a_q\xi^q \;, \tag{6.5}$$

over the canonical domain $(-1, 1)$. This implies that

$$\int_{-1}^{1} P(\xi)\, d\xi \;=\; \sum_{k=1}^{L} w_k g(\xi_k) \;, \tag{6.6}$$

namely

$$\int_{-1}^{1} (a_0 + a_1\xi + \ldots + a_q\xi^q)\, d\xi \;=\; \sum_{k=1}^{L} w_k(a_0 + a_1\xi_k + \ldots + a_q\xi_k^q) \;. \tag{6.7}$$

Since the constant coefficients a_k, $k = 0, 1, \ldots, q$, are independent of each other and arbitrary, it follows from the above equation that

$$\Big[\frac{\xi^{i+1}}{i+1}\Big]_{-1}^{1} \;=\; \sum_{k=1}^{L} w_k\xi_k^i \;=\; \begin{cases} \dfrac{2}{i+1} & i \text{ even} \\ 0 & i \text{ odd} \end{cases} \quad , \quad i = 0, 1, \ldots, q \;. \tag{6.8}$$

The $q + 1$ equations in (6.8) contain $2L$ unknowns, which means that a unique solution can be expected if, and only if, $q = 2L - 1$. It turns out that the system (6.8) possesses such a unique solution, which yields the *Gaussian quadrature* rules. These are optimal in terms of accuracy and are used extensively in finite element computations.

Consider now the Gaussian quadrature rules corresponding to the cases $L = 1, 2, 3$:

(a) When $L = 1$, it is clear that, owing to symmetry, $\xi_1 = 0$ and $w_1 = 2$. This is the well-known *mid-point rule*, which is exact for integration of polynomials of degree up to $q = 1$.

(b) When $L = 2$, symmetry dictates that $\xi_1 = -\xi_2$ and $w_1 = w_2 = 1$. The value of ξ_1 can be deduced from (6.8) taking into account that this rule should be exact for the integration of polynomials of degree up to $q = 3$. Indeed, taking $i = 2$ in (6.8) leads to $-\xi_1 = \xi_2 = \dfrac{1}{\sqrt{3}}$.

(c) When $L = 3$, symmetry necessitates that $w_1 = w_3$ and, also, $\xi_1 = -\xi_3$, and $\xi_2 = 0$. Appealing again to equation (6.8), one may find that $w_1 = w_3 = \frac{5}{9}$, $w_2 = \frac{8}{9}$, and $-\xi_1 = \xi_3 = \sqrt{\frac{3}{5}}$.

Similar results may be obtained for higher-order accurate Gaussian quadrature formulae.

The preceding formulae are readily applicable to integration over multi-dimensional domains which are products of one-dimensional domains. These include rectangles in two dimensions and orthogonal parallelepipeds in three dimensions. This observation is particularly relevant to general isoparametric quadrilateral and hexahedral elements which are mapped to the physical domain from squares and cubes. Taking, for example, the case of an isoparametric quadrilateral, write a typical integral as

$$I \;=\; \int_{\Omega^e} f(x,y)\,dxdy \;=\; \int_{-1}^{1}\int_{-1}^{1} f\big(\hat{x}(\xi,\eta),\hat{y}(\xi,\eta)\big)\,J\,d\xi d\eta \;=\; \int_{-1}^{1}\int_{-1}^{1} g(\xi,\eta)\,d\xi d\eta\;. \tag{6.9}$$

Since the coordinates ξ and η are independent, one may write

$$\int_{-1}^{1}\int_{-1}^{1} g(\xi,\eta)\,d\xi d\eta \;\doteq\; \int_{-1}^{1}\Big\{\sum_{k=1}^{L} w_k g(\xi_k,\eta)\Big\}\,d\eta \;\doteq\; \sum_{l=1}^{L} w_l \sum_{k=1}^{L} w_k g(\xi_k,\eta_l) \;=\; \sum_{k=1}^{L}\sum_{l=1}^{L} w_k w_l g(\xi_k,\eta_l)\;. \tag{6.10}$$

Generally, multi-dimensional integrals over product domains can be evaluated numerically using multiple summations (one per dimension). Figure 6.1 illustrates certain two-dimensional integration rules over square domains and the degree of polynomials of the form $\xi^{q_1}\eta^{q_2}$ that are integrated exactly by each of the rules.

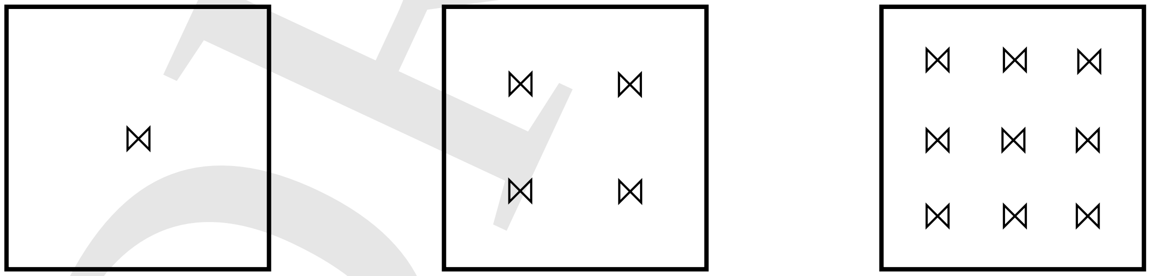


Figure 6.1: *Two-dimensional Gauss quadrature rules for $q_1, q_2 \leq 1$ (left), $q_1, q_2 \leq 3$ (center), and $q_1, q_2 \leq 5$ (right)*

Remark:

☛ It can be shown that the locations of the Gauss point in the domain $(-1, 1)$ are solutions of the *Legendre polynomials* P_k. These are defined by the recurrence formula

$$P_{k+1}(\xi) \;=\; \frac{(2k+1)\xi P_k(\xi) - kP_{k-1}(\xi)}{k+1} \;,$$

with $P_0(\xi) = 1$ and $P_1(\xi) = \xi$. The Legendre polynomials satisfy the property

$$\int_{-1}^{1} P_i(\xi)P_j(\xi)\, d\xi \;=\; \begin{cases} \dfrac{2}{i+1} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

This orthogonality property plays an essential role in establishing the aforementioned connection between Legendre polynomials and Gauss points.

Integration over triangular and tetrahedral domains can be performed either by using the exact formulae presented in Chapter 5 for polynomial functions of the area or volume coordinates or by approximate formulae of the form

$$I \;=\; \int_{\Omega^e} g(L_1, L_2, L_3)\, dA \;\doteq\; A \sum_{k=1}^{L} w_k g(L_{1k}, L_{2k}, L_{3k}) \tag{6.11}$$

for straight-edge triangles of area A, or

$$I \;=\; \int_{\Omega^e} g(L_1, L_2, L_3, L_4)\, dA \;\doteq\; V \sum_{k=1}^{L} w_k g(L_{1k}, L_{2k}, L_{3k}, L_{4k}) \tag{6.12}$$

for straight-edge, flat face tetrahedra of volume V. Figure 6.2 depicts two simple integration rules for triangles.

6.2 Assembly of global element arrays

The purpose of this section is to establish the manner in which one may start with weak forms at the element level, assemble all the element-wide information and derive global equations, whose solution yields the finite element approximation of interest.

Weak forms emanating from Galerkin, least squares, collocation or variational approaches can be written without any restrictions in any subdomain of the original domain Ω over which a differential equation is assumed to hold. Indeed, if a differential equation applies

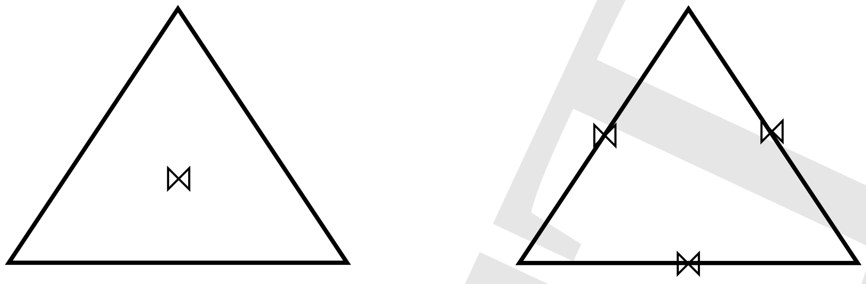


Figure 6.2: *Integration rules in triangular domains for $q \leq 1$ (left) and $q \leq 2$ (right). On the left, the integration point is located at the barycenter of the triangle and the weight is $w_1 = 1$. On the right, the integration points are located at the mid-edges and the weights are $w_1 = w_2 = w_3 = 1/3$*

over Ω, then it also applied over any subset of Ω. Further, assuming that the finite element approximation and weighting functions are smooth, the use of integration by parts and the divergence theorem is allowable. Therefore, weak forms such as (3.14) can be written over the domain Ω^e of a given element, i.e.,

$$\int_{\Omega^e} \left[\frac{\partial w_h}{\partial x_1} k \frac{\partial u_h}{\partial x_1} + \frac{\partial w_h}{\partial x_2} k \frac{\partial u_h}{\partial x_2} + w_h f \right] d\Omega + \int_{\partial\Omega^e \cap \Gamma_q} w_h \bar{q} \, d\Gamma + \int_{\partial\Omega^e \setminus \partial\Omega} w_h q \, d\Gamma = 0 \ . \tag{6.13}$$

The two boundary integral terms in (6.13) differ from those in (3.14). The first boundary term applies to the part of the element boundary $\partial\Omega^e$, if any, that happens to lie on the exterior Neumann boundary Γ_q of the domain Ω. The second boundary term refers to the interior part of the element boundary (i.e., the portion of the element boundary that is shared with other elements), which is subject to a (yet unknown) Neumann boundary condition specifying the flux q between two neighboring elements.

Starting from equation (6.13), one may write corresponding element-wide weak forms for all finite elements and add them together. This leads to

$$\sum_e \int_{\Omega^e} \left[\frac{\partial w_h}{\partial x_1} k \frac{\partial u_h}{\partial x_1} + \frac{\partial w_h}{\partial x_2} k \frac{\partial u_h}{\partial x_2} + w_h f \right] d\Omega + \sum_e \int_{\partial\Omega^e \cap \Gamma_q} w_h \bar{q} \, d\Gamma + \sum_{e,e' \text{neighbors}} \int_{C_{e,e'}} w_h [\![q]\!] \, d\Gamma = 0 \ , \tag{6.14}$$

where $C_{e,e'}$ denotes an edge shared between two contiguous elements e and e' and $[\![q]\!]$ denotes the jump of the normal flux q from element e to element e' across $C_{e,e'}$. Equation (6.14) may be readily rewritten as

$$\int_\Omega \left[\frac{\partial w_h}{\partial x_1} k \frac{\partial u_h}{\partial x_1} \; + \; \frac{\partial w_h}{\partial x_2} k \frac{\partial u_h}{\partial x_2} \; + \; w_h f \right] d\Omega \; + \; \int_{\Gamma_q} w_h \bar{q} \, d\Gamma \\ + \sum_{e,e' \text{neighbors}} \int_{C_{e,e'}} w_h [\![q]\!] \, d\Gamma \; = \; 0 \; , \qquad (6.15)$$

if one assumes that $\sum_e \int_{\Omega^e} d\Omega \, = \, \Omega$ and $\sum_e \int_{\partial\Omega^e \cap \Gamma_q} d\Gamma \, = \, \Gamma_q$. Note that, in general, the preceding conditions are satisfied only in an approximate sense, due to the error in domain and boundary discretization. Either way, it is readily apparent that the weak form in (6.15) differs from the original form in (3.14) because of the introduction of the finite element fields (w_h, u_h) and the jump conditions in the last term of the left-hand side.

The finite element assembly operation entails the summation of all the element-wide discrete weak forms to form the global discrete weak form from which one may derive a system of algebraic equations, whose solution provides the scalar coefficients that define u_h, see, e.g., equation $(3.24)_1$. Hence, for each element, one may derive an equation of the form

$$\sum_{i=1}^{n} \beta_i \Big(\sum_{j=1}^{n} K_{ij}^e \, u_j^e \; - \; F_i^e - F_i^{\text{int},e} \Big) \; = \; 0 \; , \qquad (6.16)$$

where K_{ij}^e are the components of the element stiffness matrix, F_i^e are the components of the forcing vector contributed by the domain Ω^e and the boundary $\partial\Omega^e \cap \Gamma_q$. Lastly, $F_i^{\text{int},e}$ are the components of the forcing vector due to the boundary term on $\partial\Omega^e \setminus \partial\Omega$. This term is unknown at the outset, as the element interior boundary fluxes are not specified in the original boundary-value problem.[1] Recalling the arbitrariness of the weighting function coefficients β_i, it follows immediately that for each element

$$\sum_{j=1}^{n} K_{ij}^e \, u_j^e \; - \; F_i^e - F_i^{\text{int},e} \; = \; 0 \quad , \quad i = 1, 2, \ldots, n \; . \qquad (6.17)$$

The assembly operation now amounts to combining all element-wide state equations of the form (6.17) into a global system that applies to the whole domain Ω. Symbolically, this can be represented by way of an assembly operator $\underset{e}{\mathbb{A}}$, such that

$$\underset{e}{\mathbb{A}} \left[\sum_{j=1}^{n} K_{ij}^e \, u_j^e \; - \; F_i^e - F_i^{\text{int},e} \right] \; = \; 0 \; , \qquad (6.18)$$

[1]This is precisely why one cannot, in general, solve the original boundary-value problem on a direct element-by-element basis.

or

$$\sum_{J=1}^{N} K_{IJ} u_J - F_I \;=\; 0 \quad , \quad I = 1, 2, \ldots, N \; . \tag{6.19}$$

Here,

$$[K_{IJ}] \;=\; \underset{e}{\mathbb{A}}[K^e_{ij}] \quad , \quad [F_I] \;=\; \underset{e}{\mathbb{A}}[F^e_i] \; . \tag{6.20}$$

Note that the assembled contributions of the interior boundary fluxes are neglected, namely

$$[F^{\text{int}}_I] \;=\; \underset{e}{\mathbb{A}}[F^{\text{int},e}_i] \;\doteq\; 0 \; . \tag{6.21}$$

This assumption is made in finite element methods by necessity, although it clearly induces an error. Indeed, the interior boundary fluxes are unknown at the outset, so that including the corresponding forces in the assembled state equations is not an option. On the other hand, omission of these interelement jump terms is justified by the fact that the exact solution of the differential equation guarantees flux continuity across any surface, hence in an asymptotic sense (i.e., as the approximation becomes more accurate), the force contributions of these jumps tend to vanish.

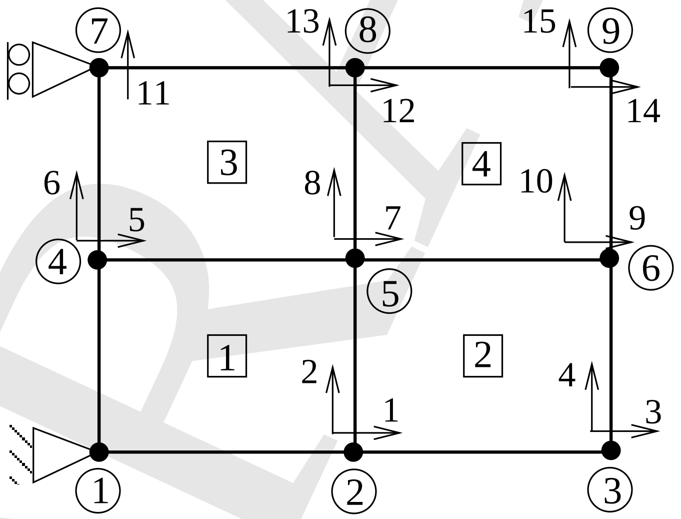


Figure 6.3: *Finite element mesh depicting global node and element numbering, as well as global degree of freedom assignments*

It is instructive here to illustrate the action of the assembly operator $\underset{e}{\mathbb{A}}$ by means of an example. To this end, consider a simple finite element mesh of 4-noded rectangular elements with two-degrees of freedom per node, see Figure 6.3. All *active* degrees of freedom (i.e., those that are not fixed) are numbered in increasing order of the global node numbers. In this manner, one forms the ID array

$$[\text{ID}] \;=\; \begin{bmatrix} 0 & 1 & 3 & 5 & 7 & 9 & 0 & 12 & 14 \\ 0 & 2 & 4 & 6 & 8 & 10 & 11 & 13 & 15 \end{bmatrix} \tag{6.22}$$

by looping over all nodes. The dimension of this array is `ndf`$\times$`numnp`, where `ndf` denotes the number of degrees of freedom per node before any boundary conditions are imposed and `numnp` denotes the total number of nodes in the mesh. Taking into account the ID array and the local nodal numbering convention for 4-noded elements (see Figure 5.36), one may now generate the LM array as

$$[\text{LM}] \;=\; \begin{bmatrix} 0 & 1 & 5 & 7 \\ 0 & 2 & 6 & 8 \\ 1 & 3 & 7 & 9 \\ 2 & 4 & 8 & 10 \\ 7 & 9 & 12 & 14 \\ 8 & 10 & 13 & 15 \\ 5 & 7 & 0 & 12 \\ 6 & 8 & 11 & 13 \end{bmatrix} \tag{6.23}$$

by looping over all elements. The dimension of this array is (`ndf*nen`)$\times$`nel`, where `nen` is the total number of nodes per element and `nel` the total number of elements in the mesh. Each column of the LM array contains the list of globally numbered degrees of freedom in the corresponding order to that of the local degrees of freedom of the element. As a result, the task of assemblying an element-wide array, say $[K^4_{ij}]$ into the global stiffness array is reduced to identifying the correspondence between local and global degrees of freedom for each component of $[K^4_{ij}]$ by direct reference to the LM array. For instance, the entry K^4_{12} is added to the global stiffness (of dimension 15×15) in the 7th row/8th column entry, as dictated by the first two entries of the 4th column of the LM array in (6.23). Likewise, the entry F^3_5 is added to the global forcing vector (of dimension 15×1) in the 12th row, as dictated by the 5th row of the 3rd column of the LM array. The preceding data structure shows that the task of assemblying global arrays amounts to a simple reindexing of the local arrays with the aid of the LM array.

6.3 Algebraic equation solving by Gaussian elimination and its variants

The system of linear algebraic equations (6.19) obtained upon assemblying the local arrays into their global counterparts can be solved using a number of different and well-established methods. These include

(1) Iterative methods, such as Jacobi, Gauss-Seidel, steepest descent, conjugate gradient, and multigrid.

(2) Direct methods, such as Gauss elimination (or LU decomposition), Choleski, frontal.

Generally, direct methods are used for relatively small systems (say, for $N \leq 10^6$), owing to their relatively high expense. Iterative methods are typically more efficient for large systems.

One of the most important properties of finite element methods is that they lead to stiffness matrices that are "banded", which means that the non-zero terms are situated around the major diagonal. This is an immediate implication of the "local" nature of the finite element approximation. Indeed, the fact that the interpolation functions have small support (see the discussion in Section 5.1) implies that coupling between degrees of freedom is restricted only to neighboring elements. Figure 6.4 illustrates a typical example of the profile of a finite element stiffness matrix:

$$[K] = \begin{bmatrix} * & * & & & & & \\ * & * & * & * & * & & \\ & * & * & * & * & & \\ & * & * & * & * & & \\ & * & * & * & * & * & \\ & & & & * & * & * \\ & & & & & * & * \end{bmatrix}$$

Figure 6.4: *Profile of a typical finite element stiffness matrix ($*$ denotes a non-zero entry)*

It is important to note that the profile of the finite element stiffness matrix is generally symmetric, even if the entries themselves are not. This is because the coupling between two degrees of freedom is generally bilateral, namely one degree of freedom affects the other and vice-versa. In the model stiffness matrix of Figure 6.4, one may define the half-bandwidth b

as the maximum height of a column above the major diagonal for which the highest entry is non-zero (in Figure 6.4, $b = 3$).

The significance of the bandedness of finite element stiffnesses becomes apparent when one considers the cost of solving a system of N linear algebraic equations. In particular, the solution of N such equations using standard Gauss elimination without accounting for bandedness requires approximately $\frac{2}{3}N^3$ flops,[2] when N is "large". If the stiffness matrix is banded, with half-bandwidth $b \ll N$, the cost is reduced to approximately $2Nb^2$. To appreciate the difference, consider a system of $N = 10^5$ equations with half-bandwidth $b = 10^3$. Standard Gauss elimination on a single processor that delivers 500 MFLOPS[3] takes approximately

$$\frac{\frac{2}{3}(10^5)^3}{500 \times 10^6} \;=\; \frac{2}{15}10^7 \text{ sec} \;\doteq\; 15 \text{ days} \; . \tag{6.24}$$

On the other hand, using a banded Gauss elimination solver requires

$$\frac{2(10^3)^2(10^5)}{500 \times 10^6} \;=\; \frac{2}{5}10^3 \text{ sec} \;\doteq\; 7 \text{ minutes} \; . \tag{6.25}$$

In addition to the substantial savings in solution time, the banded structure of the stiffness matrix allows for compact storage of its components, which reduces the associated memory requirements.

6.4 Finite element modeling: mesh design and generation

Finite element modeling is a relatively complex undertaking. It requires:

- Complete and unambiguous understanding of the boundary/initial-value problem (question: what are the relevant differential equations and boundary and/or initial conditions?)
- Familiarity with the nature of the solutions to this class of problems (question: is the finite element solution consistent with physically-motivated expectations?).
- Experience in geometric modeling (question: how does one create a finite element mesh that accurately represents the domain of interest?)

[2]A *flop* is a floating-point operation, namely an addition, subtraction, multiplication or division.
[3]MFLOPS stands for Millions of Floating-point Operations per Second.

- Deep knowledge of the technical aspects of the finite element method (questions: how does one impose Dirichlet boundary conditions, input equivalent nodal forces, choose element types, number of element integration points, etc?)

Creating sophisticated finite element models typically involves two well-tested steps. These are:

- Simplification of the model to the highest possible degree without loss of any of its salient features.

- Decomposition of the reduced model into simpler submodels, meshing of the submodels, and tying of these back into the full model.

Two aspects of mesh modeling that merit special attention are symmetry and optimal node numbering.

6.4.1 Symmetry

If the differential equation, boundary conditions, and domain are all symmetric with respect to certain axes or planes, the finite element analyst can exploit the symmetry(ies) to simplify the task of modeling. The most important step here is to apply the appropriate boundary conditions on the symmetry axis or plane.

Some typical examples of symmetry are illustrated in Figure 6.5 below.

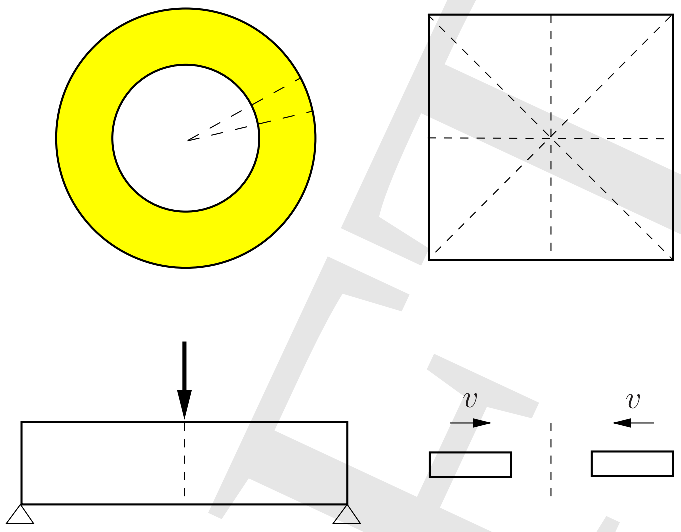


Figure 6.5: *Representative examples of symmetries in the domains of differential equations (corresponding symmetries in the boundary conditions, loading, and equations themselves are assumed)*

6.4.2 Optimal node numbering

The manner in which finite element nodes are globally numbered may play an important role in the shape and size of the profile of the resulting finite element stiffness matrix. This point is illustrated by means of a simple example, as in Figure 6.6, where the nodes of the same mesh are numbered in two distinct and regular ways, i.e., in row-wide or column-wise. Assuming that each node has two active degrees of freedom, row-wise numbering leads to a half-bandwidth $b_A = 17$, while column-wise numbering leads to $b_B = 7$.

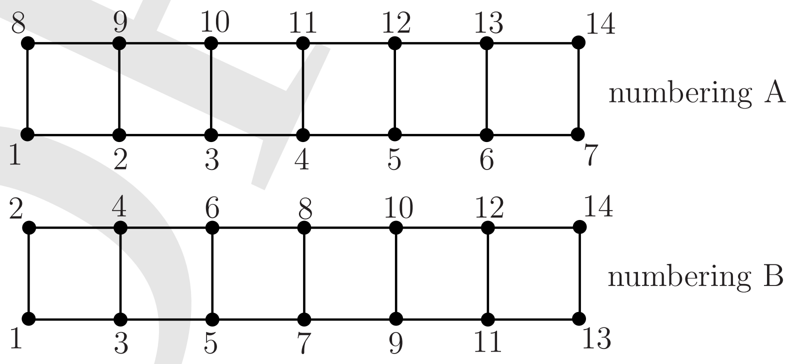


Figure 6.6: *Two possible ways of node numbering in a finite element mesh*

Generally, global node numbering should be done in a manner that minimizes the half-bandwidth of the stiffness matrix. This becomes quite challenging when creating parts of a mesh separately and then tying all of them together. Several algorithms have been devised to perform optimal (or, more often, nearly optimal) node numbering. Most commercial element codes have a built-in node numbering algorithm and the user does not have to occupy him/herself with this task.

6.5 Computer program organization

All commercial and (many) stand-along research/education finite element codes contain three basic modules: input (pre-processing), solution and output (post-processing).

The input module concerns primarily the generation of the finite element mesh and the application of boundary conditions. In this module, one may also specify the physics of the problem together with the values of any required constants, as well as the element type and other related parameters. The solution module concerns the determination of the element arrays, the assembly of the global arrays, and the solution of the resulting algebraic systems (linear or non-linear). The output module handles the computation of any quantities of interest at the mesh or individual element level and the visualization of the solution.

Some of the desirable features of finite element codes are:

- General-purpose, namely employing a wide range of finite element methods to solve diverse problems (e.g., time-independent/dependent, linear/non-linear, multi-physics, etc.)

- Full non-linearity, namely designed at the outset to treat all problems as non-linear and handling linear problems as a trivial special case.

- Modularity, namely able to incorporate new elements written by (advanced) users and finite element programmers without requiring that they know (or have access) to all parts of the program.

In recent years, there is a trend toward integration of computer-aided design software tools into finite element codes to provide "one-stop shopping" for engineering analysis/design needs. At the same time, there is a reverse trend toward diversification, where analysts use different software products for different tasks (e.g., separate software for pre-processing, solution and post-processing).

6.6 Exercises

Problem 1
A function $f(x,y) : \mathbb{R}^2 \mapsto \mathbb{R}$ varies linearly within a 3-node triangular element with domain Ω_e (i.e., $f(x,y) = a + bx + cy$ in the element). Show that the average value of this function over the element domain, defined as

$$f_{av} \;=\; \frac{1}{A}\int_{\Omega_e} f(x,y)\, d\Omega \;,$$

where A is the area of the element, is equal to the average of the three nodal values of f.

Hint: evaluate the above integral using area coordinates.

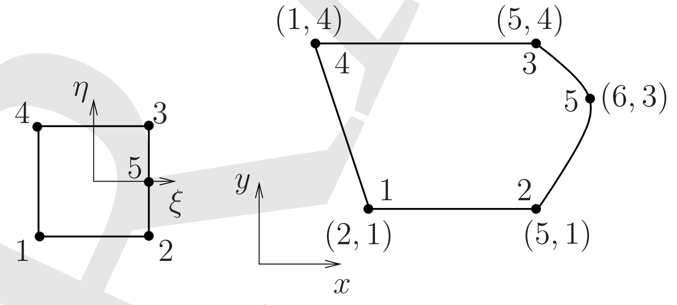
Problem 2
A 5-noded quadrilateral element is constructed by mapping isoparametrically the square domain $\Omega^e_\square$ of the natural space (ξ,η) into the partially curved domain Ω_e of the physical space (x,y), as in the figure below.

(a) Determine the shape functions of the element in the natural domain.

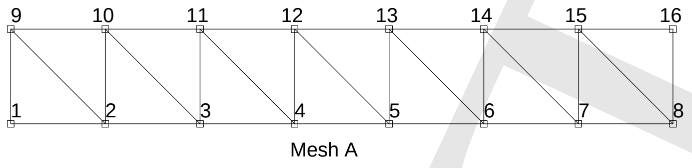
(b) Show that the isoparametric mapping is one-to-one for all points (ξ,η) of domain $\Omega_e^\square$.

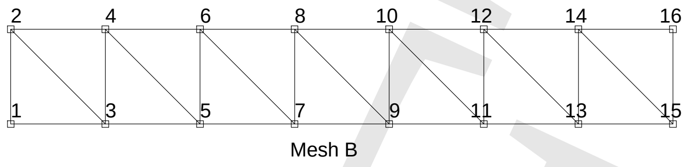
(c) Determine the minimum number of Gauss points per direction required to compute the area of the element exactly in the physical domain.

(d) Evaluate the exact area of the element in the physical domain using Gaussian quadrature.

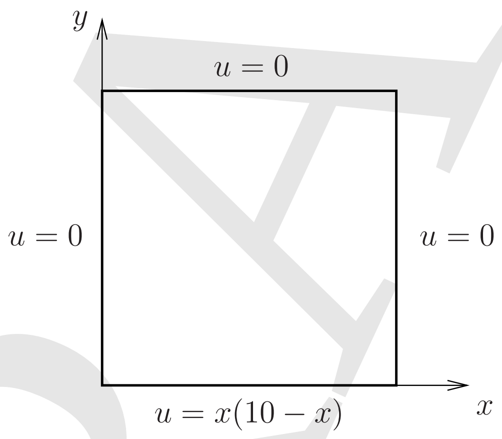


Problem 3
The two finite element meshes shown below are used in the solution of a planar problem for which every node has two active degrees of freedom numbered in increasing order with the global node numbers. For each mesh, indicate which entries of the stiffness matrix are generally non-zero by blacking them out in a 32×32 grid (you may use the grid on engineering paper). What is the half-bandwidth of each mesh for the given problem?





<u>**Problem 4**</u>
Write a finite element code to find the steady-state temperature distribution on a square rigid-body of side length $a = 10.0$, for which the boundary conditions are shown in the following figure. Assume that there is no heat supply per unit area.



Specifically, solve the problem using four uniform meshes with a total number of 4, 16, 64 and 256 4-node quadrilateral isoparametric elements using exact integration for all element arrays. Submit a copy of the code including any input files. Plot contours of the computed temperature distribution throughout the body. The exact solution of the problem can be found in series form as

$$u(x,y) \;=\; \sum_{n=1}^{\infty} \frac{0.2}{\sinh n\pi} \Big\{ \int_0^{10} z(10-z) \sin \frac{n\pi z}{10}\, dz \Big\} \sin \frac{n\pi x}{10} \sinh \frac{n\pi(10-y)}{10} \ .$$

Compute the exact solution at the center of the body ($x = 5, y = 5$) and plot the error (%) at this point as a function of the size h of the elements (use both decimal and log-log scale). Comment on the rate of convergence of the finite element solution. You may use the FEAP solutions of the same problem for comparison purposes when debugging your code.