Chapter 11

Scalar fields

11.1 Introduction

The physical behavior governing a variety of problems in engineering can be described as scalar field problems. That is, where a scalar quantity varies over a continuum. We usually need to compute the value of the scalar quantity, its gradient, and sometimes its integral over the solution domain. Typical applications of scalar fields include: electrical conduction, heat transfer, irrotational fluid flow, magnetostatics, seepage in porous media, torsion stress analysis, etc. Often these problems are governed by the well known Laplace and Poisson differential equations. The analytic solution of these equations in two- and three-dimensional field problems can present a formidable task, especially in the case where there are complex boundary conditions and irregularly shaped regions. The finite element formulation of this class of problems by using Galerkin or variational methods has proven to be a very effective and versatile approach to the solution. Previous difficulties associated with irregular geometry and complex boundary conditions are virtually eliminated. The following development will be introduced through the details of formulating the solution to the steady-state heat conduction problem. The approach is general, however, and by redefining the physical quantities involved the formulation is equally applicable to other problems involving the Poisson equation.

11.2 Variational formulation

We can obtain from any book on heat transfer the governing differential equation for steady and un-steady (transient) state heat conduction. The most general form of the heat conduction equation, in the material principal coordinate directions is the transient three-dimensional equation:

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial \theta}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial \theta}{\partial y}\right) + \frac{\partial}{\partial z}\left(k_z \frac{\partial \theta}{\partial z}\right) + Q = \frac{\partial}{\partial t}\left(\rho c_p \theta\right) \tag{11.1}$$

where, k_x, k_y, k_z = thermal conductivity coefficients, θ = temperature, Q = heat generation per unit volume, ρ = density, and c_p = specific heat at constant pressure. If we focus our attention to the two-dimensional ($\partial/\partial z = 0$) steady-state ($\partial/\partial t = 0$) problem, such as Fig. 11.1, the governing equation becomes

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial \theta}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial \theta}{\partial y}\right) + Q = 0 \tag{11.2}$$

in which k_x, k_y, and Q are known. Equations 11.1 or 2, along with the boundary (and initial) conditions specify the problem completely. The most commonly encountered boundary conditions are those in which the temperature, θ, is specified on the boundary,

$$\theta = \theta(s) \text{ on } \Gamma_D,$$

or the normal heat flux into the boundary, q_s, is specified:

$$k_x \frac{\partial \theta}{\partial x} n_x + k_y \frac{\partial \theta}{\partial y} n_y + q_s = k_n \frac{\partial \theta}{\partial n} + q_s = 0 \text{ on } \Gamma_q$$

or a normal heat flux due to convection:

$$k_n \frac{\partial \theta}{\partial n} + h(\theta - \theta_\infty) = 0, \text{ on } \Gamma_h \tag{11.3}$$

where n_x and n_y are the direction cosines of the outward normal to the boundary surface, q_s represents the known heat flux per unit of surface, and $h(\theta - \theta_\infty)$ is the convection heat loss per unit area due to a convection coefficient h and a surrounding fluid at a temperature of θ_∞. Only one of these two last two items is non-zero on a particular surface. Note that the last two surface integrals could be written in a more general form if we combine them into a *mixed* or *Robin* condition written as:

$$k_n \frac{\partial \theta}{\partial n} + h\,\theta + g = 0, \tag{11.4}$$

where g is either a known influx (when $h = 0$), or $h\,\theta_\infty$ on a convection surface.

In Section 2.13.2 we illustrated how to apply the Galerkin method to this equation. As stated previously, an alternative formulation to the above heat conduction problem is possible using the calculus of variations. It has been shown that if a variational form exists for a differential equation then both the Galerkin form and the Euler variational form will yield exactly the same element matrix definitions. Euler's theorem of the calculus of variations states that if the integral

$$I(u) = \int_\Omega f(x, y, z, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z})\, d\Omega + \int_\Gamma (gu + hu^2/2)\, d\Gamma \tag{11.5}$$

is to be minimized, the necessary and sufficient condition for this minimum to be reached is that the unknown function $u(x, y, z)$ satisfy the following differential equation

$$\frac{\partial}{\partial x}\frac{\partial f}{\partial(\partial u/\partial x)} + \frac{\partial}{\partial y}\frac{\partial f}{\partial(\partial u/\partial y)} + \frac{\partial}{\partial z}\frac{\partial f}{\partial(\partial u/\partial z)} - \frac{\partial f}{\partial u} = 0 \tag{11.6}$$

within the region Ω, provided u satisfies the essential boundary conditions on Γ_D and

$$n_x k_x \frac{\partial \theta}{\partial x} + n_y k_y \frac{\partial \theta}{\partial y} + n_z k_z \frac{\partial \theta}{\partial z} + g + h\theta = 0 = k_n \frac{\partial \theta}{\partial n} + g + h\theta$$

on the remainder of Γ. We can verify that the minimization of the volume integral

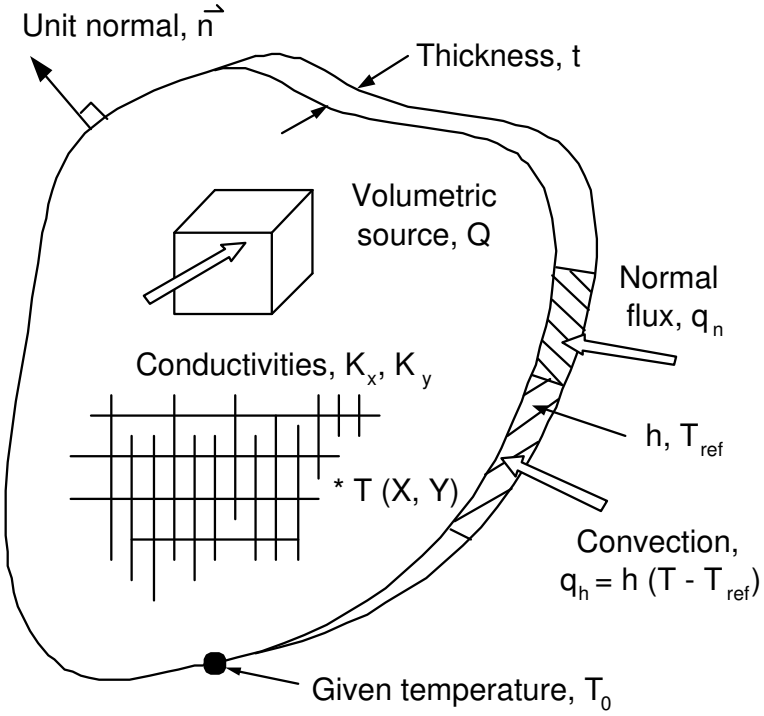


Figure 11.1 *An anisotropic heat transfer region*

$$I = \int_{\Omega} \left[\frac{1}{2} \left\{ k_x \left(\frac{\partial \theta}{\partial x}\right)^2 + k_y \left(\frac{\partial \theta}{\partial y}\right)^2 + k_z \left(\frac{\partial \theta}{\partial z}\right)^2 \right\} - Q\theta \right] d\Omega \qquad (11.7)$$

$$+ \int_{\Gamma} \left[g\theta + h\theta^2/2 \right] d\Gamma$$

leads directly to the formulation equivalent to Eq. 11.2 for the steady-state case. It should also be noted that the surface Γ will be split into different regions for each distinct set of surface input. One of those segments will usually be a Dirichlet region, Γ_D, and that surface integral represents the unknown resultant reaction fluxes at the nodes that get lumped into the RHS of the algebraic system. The functional volume contribution is

$$f = \tfrac{1}{2} \left\{ k_x \left(\frac{\partial \theta}{\partial x}\right)^2 + k_y \left(\frac{\partial \theta}{\partial y}\right)^2 + k_z \left(\frac{\partial \theta}{\partial z}\right)^2 \right\} - Q\theta.$$

Thus, if f is to be minimized it must satisfy Eq. 11.6. Here

$$\frac{\partial f}{\partial(\partial\theta/\partial x)} = k_x \frac{\partial \theta}{\partial x}, \quad \frac{\partial f}{\partial(\partial\theta/\partial y)} = k_y \frac{\partial \theta}{\partial y}, \quad \frac{\partial f}{\partial(\partial\theta/\partial z)} = k_z \frac{\partial \theta}{\partial z}, \quad \frac{\partial f}{\partial \theta} = -Q$$

so Eq. 11.6 results in

$$\frac{\partial}{\partial x}(k_x \frac{\partial \theta}{\partial x}) + \frac{\partial}{\partial y}(k_y \frac{\partial \theta}{\partial y}) + \frac{\partial}{\partial z}(k_z \frac{\partial \theta}{\partial z}) + Q = 0$$

verifying that the function f does lead to correct steady state formulation, if the boundary conditions are also satisfied. Euler also stated that the natural boundary condition associated with Eq. 11.5 on a surface with a unit normal vector $\vec{n}$ is

$$n_x \left\{ \frac{\partial f}{\partial(\partial u / \partial x)} \right\} + n_y \left\{ \frac{\partial f}{\partial(\partial u / \partial y)} \right\} + n_z \left\{ \frac{\partial f}{\partial(\partial u / \partial z)} \right\} + g + hu = 0$$

on the boundary where the value of u is not prescribed. If both g and h are non-zero this type of boundary condition is a Robin, or mixed, condition since it imposes a linear combination on the solution and the normal gradient on part of the boundary.

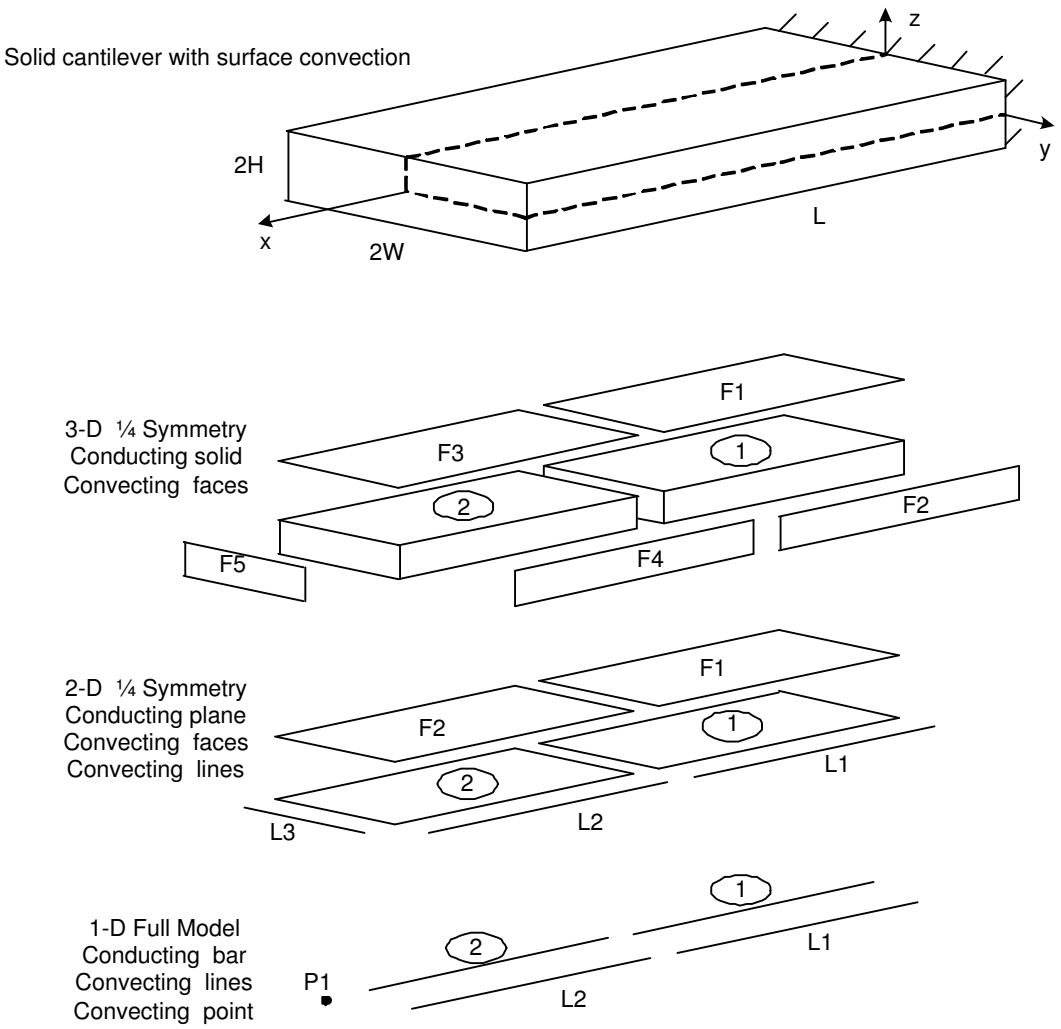


Figure 11.2 *Three-dimensional thermal models and their approximations*

The element and boundary matrices arising from Eq. 11.7 and the Galerkin method will be identical. Therefore we have the tools to build non-homogeneous, anisotropic thermal models of solids of complex geometry. As illustrated in Fig. 11.2 this will often require combining the effects of conduction elements and convection elements at shared nodes. Today commercial codes can quickly generate and solve fine meshes for 3-D thermal studies. However, it is still common to approximate some 3-D solids by 2-D models as seen in that figure. If we allow the specified thickness to vary from element to element we sometimes call this a $2\,\frac{1}{2}$-D model. The convection elements in a 2-D model can occur over the face of a 2-D conduction element and/or along an edge having a specified thickness. Likewise, 1-D approximations can have perimeter convection effects, as line elements, combined with the 1-D conduction elements. They can also have convection over an end area which is represented as a point convection element. Later we will see that a single computer implementation can handle all the combinations shown in the above figure. It is still recommended that various types of finite element models be compared to each other as a means for validating the results to problems for which the answer is not known. Sometimes a model that can be solved by hand gives a useful validation of results from a commercial code. For simplicity, in the next section we look at 2-D models that can yield matrices that can be manipulated in closed form, and then return later to numerically integrated elements for general 3-D use.

11.3 Element and boundary matrices

From Eqs. 11.1 and 11.7 it is clearly seen that the two-dimensional functional required for the steady-state analysis is

$$I = \int_A \left[\tfrac{1}{2} \left\{ k_x (\frac{\partial \theta}{\partial x})^2 + k_y (\frac{\partial \theta}{\partial y})^2 \right\} - Q\theta \right] t \, dA + \int_\Gamma (g\theta + h\theta^2/2) t \, ds \tag{11.8}$$

where t is the thickness of the domain. We will proceed in exactly the same manner as we did for the previous variational formulations. That is, we will assume that the area integral is the sum of the integrals over the element areas. Likewise, the boundary integral where the temperature is not specified is assumed to be the sum of the boundary segment integrals. Thus, $I = \sum_e I^e + \sum_b I^b$ where the element contributions are

$$I^e = \int_{A^e} \left[\tfrac{1}{2} \left\{ k_x (\frac{\partial \theta}{\partial x})^2 + k_y (\frac{\partial \theta}{\partial y})^2 \right\} - Q\theta \right] t \, dA$$

and the boundary segment contributions are

$$I^b = \int_{\Gamma^b} (g\theta + h\theta^2/2) t \, ds.$$

If we make the usual interpolation assumptions in the element and on its typical edge then we can express these quantities as

$$I^e = \tfrac{1}{2} \mathbf{D}^{e^T} \mathbf{S}^e \mathbf{D}^e - \mathbf{D}^{e^T} \mathbf{C}^e, \qquad I^b = \tfrac{1}{2} \mathbf{D}^{b^T} \mathbf{S}^b \mathbf{D}^b - \mathbf{D}^{b^T} \mathbf{C}^b.$$

Here the element matrices are the orthotropic conduction square matrix

$$\mathbf{S}^e = \int_{A^e} (k_x^e \, \mathbf{H}_x^{e^T} \mathbf{H}_x^e + k_y^e \, \mathbf{H}_y^{e^T} \mathbf{H}_y^e) t^e \, da = \int_{A^e} \mathbf{B}^{e^T} \mathbf{E}^e \mathbf{B}^e t^e \, da \tag{11.9}$$

the internal source vector

$$\mathbf{C}_Q^e = \int_{A^e} \mathbf{H}^{e^T} Q^e t^e \, da \tag{11.10}$$

the surface square matrix and source vector from convection

$$\mathbf{S}_h^b = \int_{\Gamma^b} h^b \mathbf{H}^{b^T} \mathbf{H}^b t^b \, ds, \quad \mathbf{C}_h^b = \int_{\Gamma^b} \theta_\infty^b h^b \mathbf{H}^{b^T} t^b \, ds \tag{11.11}$$

and/or the source vector due to a specified inward heat flow

$$\mathbf{C}_q^b = \int_{\Gamma^b} q^b \mathbf{H}^{b^T} t^b \, ds \tag{11.12}$$

where $\mathbf{H}$ denotes the shape functions and $\mathbf{H}_x = \partial \mathbf{H} / \partial x$, etc., are rows of the solution interpolation gradient, $\mathbf{B}^e$, and where the general constitutive matrix is $\mathbf{E}$. The symmetric array $\mathbf{E}^e$ reduces to its diagonal orthotropic form when $k_{xy} = 0 = k_{yx}$, and to the common isotropic form, $\mathbf{E}^e = k\mathbf{I}$, when $k_{xx} = k_{yy} = k$. For this class of problem there is only one unknown temperature per node. Once again, if $\mathbf{D}$ denotes all of these unknowns then $\mathbf{D}^e \subset \mathbf{D}$ and $\mathbf{D}^b \subset \mathbf{D}$. Figure 11.3 illustrates where these typical conduction, convection, and source terms are inserted in the algebraic equations of thermal equilibrium.

Likewise, Figure 11.4 reminds us that a Dirichlet (essential) boundary condition assigns a value to one of the degrees of freedom and introduces unknown reaction source terms; that a Neumann (flux) condition only contributes known terms to the source vector; while a Robin (mixed) boundary condition couples the flux linearly to the boundary value and introduces known terms into both the system square matrix and source vector.

Many analysis problems have features that allow the analyst to reduce greatly the cost of FEA through the use of *symmetry, anti-symmetry,* or *cyclic symmetry* and *coupled nodes*. The system equations are sparse and can often be described by the bandwidth, say B, and the total number of equations, say M. Solving the equations is very expensive and has an operations count that is proportional to the product B^2M. The storage required by the system is proportional to BM. Whenever possible we try to reduce these two parameters. Partial models allow us to reduce them very easily and still generate all the information we require. For example, a half-symmetry model would usually reduce both B and M by a factor of two and thereby reduce the solution costs by a factor of eight and reduce the storage requirement by a factor of four.

When a model has a plane of geometric, material, and support symmetry, it is not usually necessary to analyze the whole model. Conditions of symmetry or anti-symmetry in the source terms can be applied to the planes of symmetry in order to produce a partial model that includes the effects of the other removed parts of the complete model. In order to employ a partial analysis involving a symmetry plane, it is necessary for the following quantities to be symmetric with respect to that plane: the geometry, the material regions, the material properties, and the essential boundary conditions (on the

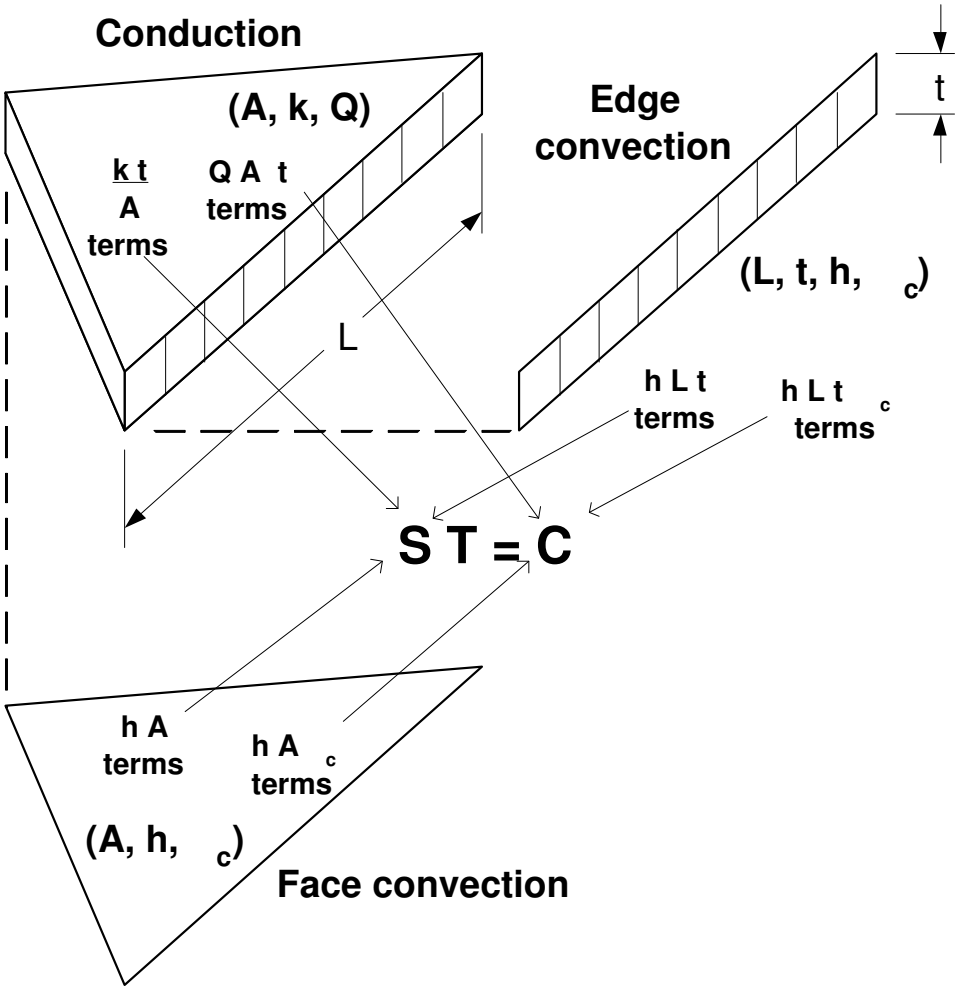


Figure 11.3 *Contributions from conducting, convecting and source regions*

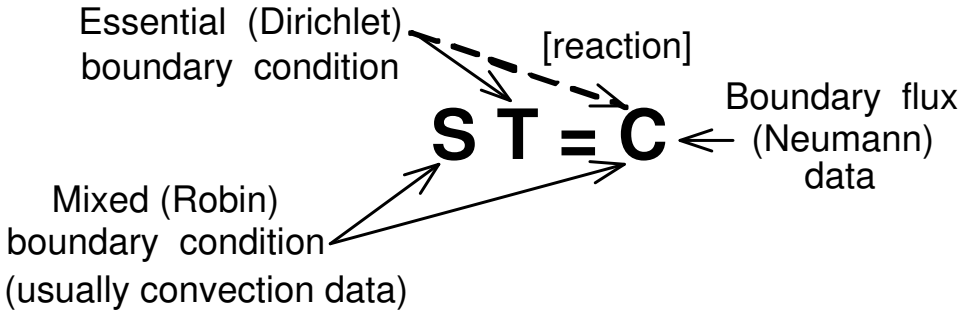
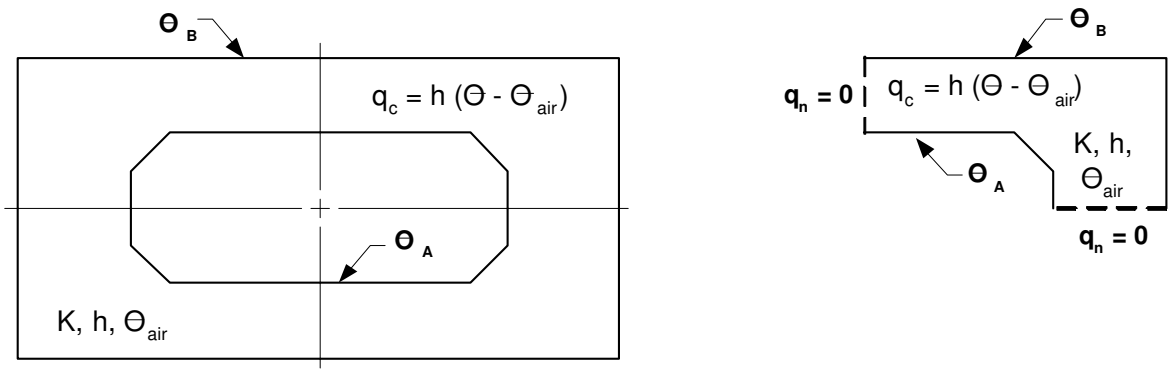
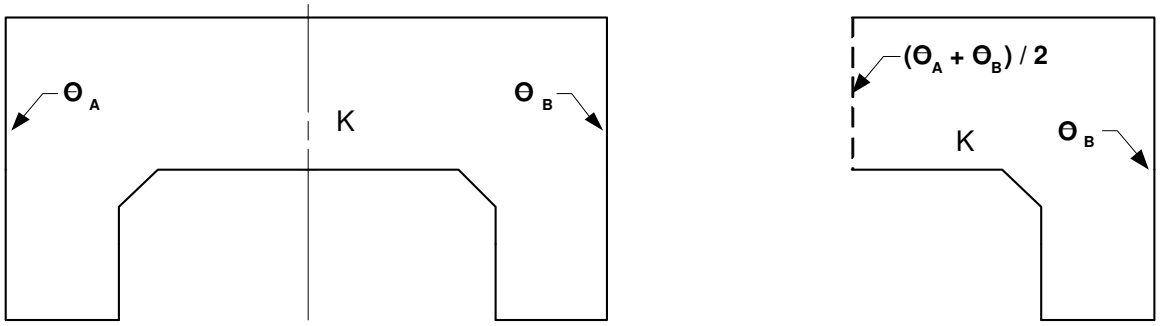


Figure 11.4 *Contributions of boundary conditions to algebraic system*

temperature). If these conditions are not quite fulfilled, then the analyst will have to exercise judgement before selecting a partial model. Even if a full model is selected for eventual use, an approximate partial model can give useful insight and aid in planning the details of the full model to be run later.



a) Quarter symmetry model



b) Half anti-symmetry model

Figure 11.5 *Typical symmetry and anti-symmetry thermal models*

In thermal models any symmetry plane has a zero heat flux and temperature gradient normal to that plane. That is because the temperatures are mirror images of each other and have the same sign when moving normal to the plane. The state of zero heat flux normal to a boundary is a natural boundary condition in a finite element analysis (but not in finite differences). We obtain that condition by default. If you desire a different type of condition to apply on a boundary, then we must prescribe either the temperature (the essential condition) or a different nonzero normal flux. We cannot give both. When you prescribe one of them, the other becomes a 'reaction' to be determined from the final result. Thermal anti-symmetry means the temperatures approach the plane with temperature increments of opposite signs but equal magnitudes. Thus, the temperature must be a known mean temperature on a plane of anti-symmetry. When that essential boundary condition is imposed, the necessary normal heat flow through that plane can be found as a 'reaction' from the final solution.

To give some specific examples of these concepts, some typical thermal and stress problems will be represented. Figure 11.5 shows a planar rectangular region with homogeneous and isotropic conduction properties, k, and internal and external specified edge temperatures, θ_A and θ_B. In addition, it has free convection, q_h, over its face to a fluid with homogeneous convection properties, h and θ_∞. Such a system has double

symmetry, which allows us to employ a one-quarter model with consistent boundary conditions. Doing this cuts M by a factor of 4 and reduces B by at least a factor of 2 and possibly much more. Thus, the cost drops by at least a factor of 16. The alternate partial model form still requires the essential conditions on θ_A and θ_B and the face convection data for q_h. The change is that the heat flow is zero on the new boundary lines formed by the symmetry planes. This is a natural condition in FEA and requires no input data (other than the geometry of the new line). That figure also shows a simple anti-symmetric domain where it is relatively clear to determine the value of the middle temperature to be assigned as an essential boundary condition. The temperatures on either side of the centerline are both either positive or negative increments of the relative temperature change. Commercial visualization codes often have the ability to plot a full region from a partial analysis region and the identification of the symmetry and anti-symmetry planes.

11.4 Linear triangular element

If we select the three node (linear) triangle then the element interpolation functions, $\mathbf{H}^e$, are given in unit coordinates by Eq. 9.6 and in global coordinates by Eqs. 9.13 and 9.14. From either set of equations we note that

$$\begin{aligned} \mathbf{H}^e_x &= \partial \mathbf{H}^e / \partial x = \lfloor\, b_1 \;\; b_2 \;\; b_3 \,\rfloor^e / 2A^e = \mathbf{d}_x \\ \mathbf{H}^e_y &= \partial \mathbf{H}^e / \partial y = \lfloor\, c_1 \;\; c_2 \;\; c_3 \,\rfloor^e / 2A^e = \mathbf{d}_y. \end{aligned} \tag{11.13}$$

Since these are constant we can evaluate the integral by inspection if the conductivities are also constant:

$$\mathbf{S}^e = \frac{k^e_x t^e}{4A^e} \begin{bmatrix} b_1 b_1 & b_1 b_2 & b_1 b_3 \\ b_2 b_1 & b_2 b_2 & b_2 b_3 \\ b_3 b_1 & b_3 b_2 & b_3 b_3 \end{bmatrix}^e + \frac{k^e_y t^e}{4A^e} \begin{bmatrix} c_1 c_1 & c_1 c_2 & c_1 c_3 \\ c_2 c_1 & c_2 c_2 & c_2 c_3 \\ c_3 c_1 & c_3 c_2 & c_3 c_3 \end{bmatrix}^e . \tag{11.14}$$

This is known as the *element conductivity matrix*. Note that this allows for different conductivities in the x- and y- directions. Equations 11.14 show that the conduction in the x-direction depends on the size of the element in the y-direction, and vice versa. If the internal heat generation, Q, is also constant then Eq. 11.10 yields:

$$\mathbf{C}^{e\,T}_Q = \frac{Q^e A^e t^e}{3} \lfloor 1 \quad 1 \quad 1 \rfloor. \tag{11.15}$$

This internal source vector shows that a third of the internal heat generated, $Q^e A^e t^e$, is equally lumped to each of the three nodes. On a typical boundary segment the edge interpolation can also be given by a linear form. The exact integrals can be evaluated for a constant Jacobian. For example, if the coefficient, h, and the surrounding temperature, T^b_∞, are constant then the boundary segment square and column matrices are:

$$\mathbf{S}^b_h = \frac{h^b L^b t^b}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad \mathbf{C}^b_h = \frac{T^b_\infty h^b L^b t^b}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}, \tag{11.16}$$

where $L^b t^b$ represents the surface area over which the convection occurs. Similarly if a constant normal flux, q, is given over a similar surface area then the resultant boundary flux vector is

$$\mathbf{C}_q^b = \frac{q^b L^b t^b}{2} \left\{ \begin{array}{c} 1 \\ 1 \end{array} \right\}. \tag{11.17}$$

In this case half the total normal flux is lumped at each of the two nodes on the segment.

```
!  ..............................................................     ! 1
!     ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***     ! 2
!  ..............................................................     ! 3
!           (Stored as application source example 204.)               ! 4
                                                                       ! 5
! Linear Triangle, K_xx U,xx + 2 K_xy U,xy + K_yy U,yy + Q = 0        ! 6
  REAL(DP) :: X_I, X_J, X_K, Y_I, Y_J, Y_K     ! Global coordinates  ! 7
  REAL(DP) :: A_I, A_J, A_K, B_I, B_J, B_K     ! Standard geometry   ! 8
  REAL(DP) :: C_I, C_J, C_K, X_CG, Y_CG, TWO_A ! Standard geometry   ! 9
  REAL(DP) :: THICK                            ! Element thickness   !10
                                                                      !11
!   DEFINE NODAL COORDINATES, CCW: I, J, K                           !12
   X_I = COORD (1,1) ;    X_J = COORD (2,1) ;    X_K = COORD (3,1)  !13
   Y_I = COORD (1,2) ;    Y_J = COORD (2,2) ;    Y_K = COORD (3,2)  !14
                                                                      !15
!   DEFINE CENTROID COORDINATES (QUADRATURE POINT)                   !16
   X_CG = (X_I + X_J + X_K)/3.d0 ;  Y_CG = (Y_I + Y_J + Y_K)/3.d0   !17
                                                                      !18
!   GEOMETRIC PARAMETERS: H_I (X,Y) = (A_I + B_I*X + C_I*Y)/TWO_A    !19
   A_I = X_J * Y_K - X_K * Y_J ; B_I = Y_J - Y_K ; C_I = X_K - X_J  !20
   A_J = X_K * Y_I - X_I * Y_K ; B_J = Y_K - Y_I ; C_J = X_I - X_K  !21
   A_K = X_I * Y_J - X_J * Y_I ; B_K = Y_I - Y_J ; C_K = X_J - X_I  !22
                                                                      !23
!   CALCULATE TWICE ELEMENT AREA                                     !24
   TWO_A = A_I + A_J + A_K              ! = B_J*C_K - B_K*C_J also   !25
                                                                      !26
!   DEFINE 2 BY 3 GRADIENT MATRIX, B (= DGH)                         !27
   B (1, 1:3) = (/ B_I, B_J, B_K /) / TWO_A  ! DH/DX, row 1          !28
   B (2, 1:3) = (/ C_I, C_J, C_K /) / TWO_A  ! DH/DY, row 2          !29
                                                                      !30
!   DEFINE PROPERTIES: 1-K_xx, 2-K_yy, 3-K_xy, 4-Source, 5-thick     !31
   E (1, 1) = GET_REAL_LP (1) ; E (1, 2) = GET_REAL_LP (3)           !32
   E (2, 2) = GET_REAL_LP (2) ; E (2, 1) = E (1, 2) ; THICK = 1      !33
   IF ( EL_REAL >= 5 ) THICK = GET_REAL_LP (5)                       !34
   E = E * THICK  ! for proper flux recovery                         !35
                                                                      !36
!   CONDUCTION MATRIX, WITH CONSTANT JACOBIAN (t in E)               !37
   S = MATMUL ( TRANSPOSE (B), MATMUL (E, B) ) * TWO_A * 0.5d0       !38
                                                                      !39
!   SOURCE VECTOR: C(1:3) = SOURCE_PER_UNIT_AREA * AREA / 3          !40
   C = GET_REAL_LP (4) * THICK * TWO_A / 6.d0                        !41
                                                                      !42
!  SAVE ONE POINT RULE TO AVERAGING, OR ERROR ESTIMATOR              !43
   LT_QP = 1 ; CALL STORE_FLUX_POINT_COUNT ! Save LT_QP              !44
   CALL STORE_FLUX_POINT_DATA ( (/ X_CG, Y_CG /), E, B )             !45
                                                                      !46
!  End of application dependent code 204.my_el_sq_inc                !47
!  ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***          !48
```

Figure 11.6 *Anisotropic linear triangle conduction element*

If one wished to code this simple element in closed form it is very easy to do as shown in Fig. 11.6. There it is assumed that each element has four or five real, or floating point, properties of K_{xx}, K_{yy}, plus the anisotropic value K_{xy} not used above, and the source per unit area of Q. There the thickness is assumed to be unity for all elements unless it is provided as an optional fifth property. If one wanted to allow more general element families, then numerical integration would be required and the coding is a little longer, as we will see shortly. If we wish to allow for only a constant normal flux along any straight line edge segment then it is quite simple to implement Eq. 11.12 in closed form, as shown in Fig. 11.7. A common use of two-dimensional models is to predict the temperature in thin cooling fins. Then Eq. 11.11 would be applied over the face(s) of the element to define the convection matrices, which represent the most common kind of mixed, or Robin, boundary conditions. Again, for the linear triangle the closed form equations are easy to implement, if we assume constant data over each mixed boundary segment (face). The implementation is given in Fig. 11.8. We will shortly illustrate these matrix definitions with some simple applications. These codes are saved as example 204.

11.5 Linear triangle applications

Since the three node triangle is so widely used in finite element analysis we will examine its more common applications in detail.

11.5.1 Internal source

Consider a uniform square of material that has its exterior perimeter maintained at a constant temperature while its interior generates heat at a constant rate. We note that the solution will be symmetric about the square's centerlines as well as about its two diagonals. This means that we only need to utilize one-eighth of the region in the analysis. For simplicity we will assume that the material is homogeneous and isotropic so $k_x = k_y = k$. The planes of symmetry have zero normal heat flux, $q = 0$. That condition is a natural boundary condition in a finite element analysis. That is true since $\mathbf{C}_q$ in Eq. 11.12 is identically zero when the normal flux, q, is zero. The remaining essential condition is that of the known external boundary temperature as shown in Fig. 11.9. For this model we have first selected a crude mesh suitable for a hand solution, then we will give results for a finer mesh using the *MODEL* program. As shown in the figure we will use four elements and six nodes. The last three nodes have the known temperature and the first three are the unknown internal temperatures. For this homogeneous region the data are:

Element	k^e	Q^e	*Topology*	t^e
1	8	6	1, 2, 3	1
2	8	6	2, 4, 5	1
3	8	6	5, 3, 2	1
4	8	6	3, 5, 6	1

From the geometry in the figure we determine that the element geometric properties from Eq. 9.14 are:

```
! ..............................................................    ! 1
! ***  SEG_COL_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***      ! 2
! ..............................................................    ! 3
!           (Stored as application source example 204.)           ! 4
! Given normal flux on a straight boundary segment (BS) edge:      ! 5
! Standard form: -K_n * U,n = Q_NORMAL_SEG, where Q_NORMAL_SEG     ! 6
! is from control keywords normal_flux, flux_thick, or via        ! 7
! optional flux segment real properties: 1-flux, 2-thickness      ! 8
 REAL(DP) :: EDGE_L, THICK ! Edge length, thickness               ! 9
                                                                  !10
   ! Get the edge length, and thickness of edge                   !11
   THICK = 1   ! Default in all cases                             !12
   IF ( FLUX_THICK /= 1.d0   ) THICK = FLUX_THICK ! line only     !13
   EDGE_L = SQRT ( (COORD(2,1) - COORD(1,1)) **2 &                !14
                 + (COORD(2,2) - COORD(1,2)) **2 )                !15
                                                                  !16
   ! Override keyword option via segment properties               !17
   IF ( SEG_REAL > 0 ) Q_NORMAL_SEG = GET_REAL_SP (1)             !18
   IF ( SEG_REAL > 1 ) THICK        = GET_REAL_SP (2)             !19
                                                                  !20
   C (1) = Q_NORMAL_SEG * THICK * EDGE_L / 2                      !21
   C (2) = Q_NORMAL_SEG * THICK * EDGE_L / 2                      !22
!  End of application dependent code 204.my_seg_col_inc           !23
```

Figure 11.7 *Straight linear edge flux source element*

```
! ..............................................................   ! 1
!  ***  MIXED_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***    ! 2
! ..............................................................   ! 3
!           (Stored as application source example 204.)           ! 4
! Global CONVECT_COEF set by keyword convect_coef is available     ! 5
! Global CONVECT_TEMP set by keyword convect_temp is available     ! 6
! Standard form: -K_n * U,n = CONVECT_COEF ( U - CONVECT_TEMP)     ! 7
                                                                  ! 8
! Linear Triangle Boundary Face Convection Matrices                ! 9
  REAL(DP) :: X_I, X_J, X_K, Y_I, Y_J, Y_K  ! Global coordinates  !10
  REAL(DP) :: A_I, A_J, A_K, TWO_A          ! Standard geometry   !11
                                                                  !12
!   DEFINE NODAL COORDINATES, CCW: I, J, K                        !13
   X_I = COORD (1,1) ;   X_J = COORD (2,1) ;   X_K = COORD (3,1)  !14
   Y_I = COORD (1,2) ;   Y_J = COORD (2,2) ;   Y_K = COORD (3,2)  !15
                                                                  !16
!   GEOMETRIC PARAMETERS, TWICE ELEMENT AREA                      !17
   A_I = X_J * Y_K - X_K * Y_J ; A_J = X_K * Y_I - X_I * Y_K      !18
   A_K = X_I * Y_J - X_J * Y_I ; TWO_A = A_I + A_J + A_K          !19
                                                                  !20
!   Get convection data from keyword or optional properties       !21
   IF ( MIXED_REAL > 0 ) THEN   ! override keyword                !22
     CONVECT_COEF = GET_REAL_MX (1)  ! convection coefficient     !23
     CONVECT_TEMP = GET_REAL_MX (2)  ! convection temperature     !24
   END IF                            ! properties supplied       !25
                                                                  !26
!   FACE CONVECTION SQUARE MATRIX, WITH CONSTANT JACOBIAN         !27
   S = CONVECT_COEF * TWO_A / 24                              &   !28
       * RESHAPE ( (/ 2, 1, 1, 1, 2, 1, 1, 1, 2 /), (/3, 3/))    !29
                                                                  !30
!   FACE CONVECTION SOURCE VECTOR                                 !31
   C = CONVECT_TEMP * CONVECT_COEF * TWO_A / 6 * (/ 1, 1, 1 /)    !32
!  ***  END MIXED_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***      !33
```

Figure 11.8 *Face convection for a linear triangle segment*

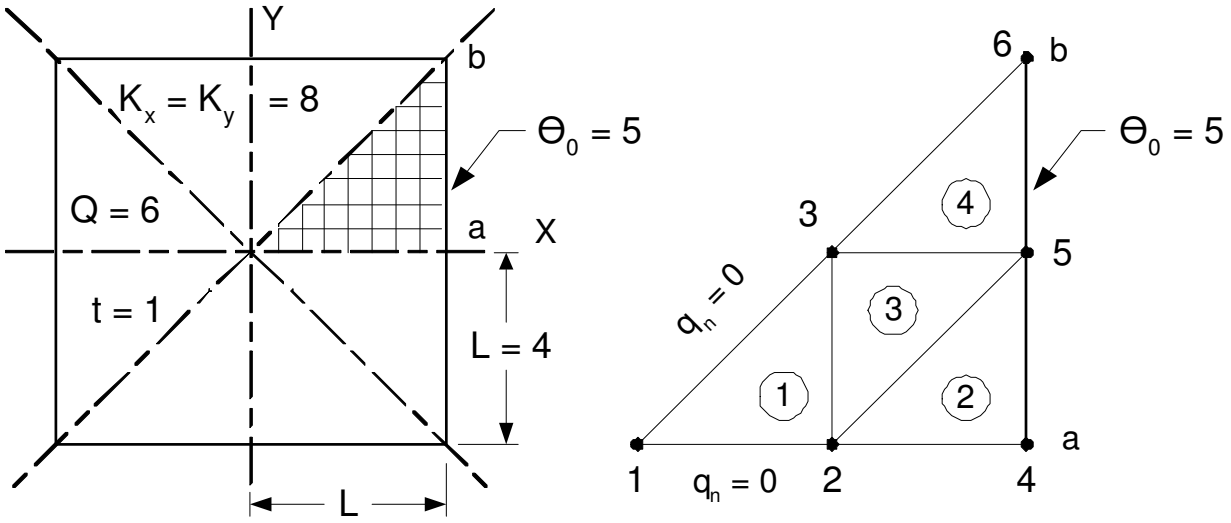


Figure 11.9 *A one-eighth symmetry model of a square*

```
title "CONDUCTION EXAMPLE, T3, INTERNAL SOURCE"                    ! 1
example  204 ! Application source code library numbe               ! 2
b_rows     2 ! Number of rows in the B (operator) matrix           ! 3
dof        1 ! Number of unknowns per node                         ! 4
el_nodes   3 ! Maximum number of nodes per element                 ! 5
elems      4 ! Number of elements in the system                    ! 6
gauss      1 ! Maximum number of quadrature points                 ! 7
nodes      6 ! Number of nodes in the mesh                         ! 8
shape      2 ! Element shape, 1=line, 2=tri, 3=quad, 4=hex         ! 9
space      2 ! Solution space dimension                            !10
el_homo      ! Element properties are homogeneous                  !11
el_real    4 ! Number of real properties per element               !12
remarks    9 ! Number of user remarks, e.g. property names         !13
end          ! Terminate the keyword control, remarks follow       !14
Conduction example of Section 11.4                                 !15
K_x = K_y = 8, Q = 6, K_xy = 0                                     !16
L_1_4 = L_4_6 = 4, Thickness = 1                                   !17
with 1/8 symmetry, so natural BC on                                !18
edges 1_6 and 1_4 ia q_n = 0                                       !19
EBC on edge 4_6 is T = 5                                           !20
K_x T,xx + 2K_xy T,xy + K_y T,yy + Q = 0                           !21
[gauss > 0 turns on flux averaging                                 !22
and possibly post-processing]                                      !23
   1    0 0.   0. ! node, ebc flag, x, y                           !24
   2    0 2.   0. ! node, ebc flag, x, y                           !25
   3    0 2.   2. ! node, ebc flag, x, y                           !26
   4    1 4.   0. ! node, ebc flag, x, y                           !27
   5    1 4.   2. ! node, ebc flag, x, y                           !28
   6    1 4.   4. ! node, ebc flag, x, y                           !29
  1  1  2  3      ! elem, three nodes                               !30
  2  2  4  5      !                                                 !31
  3  2  5  3                                                       !32
  4  3  5  6                                                       !33
   4    1 5.      ! node, dof, value of EBC                        !34
   5    1 5.                                                       !35
   6    1 5.                                                       !36
  1 8. 8. 0. 6.   ! elem, K_x, K_y, K_xy, Q (homogeneous)          !37
```

Figure 11.10 *Sample data for square bar thermal analysis*

$$\begin{array}{cccc} & e=1,2,4 & & e=3 \\ i & 1 & 2 & 3 & 1 & 2 & 3 \\ b_i & -2 & 2 & 0 & 2 & -2 & 0 \\ c_i & 0 & -2 & 2 & 0 & 2 & -2 \\ & A^e=2 & & & A^e=2 & & \end{array}$$

From Eq. 11.14 the conduction square matrix for elements 1, 2, and 4 are

$$\mathbf{S}^e = \frac{8(1)}{4(2)}\begin{bmatrix} 4 & -4 & 0 \\ -4 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{8(1)}{4(2)}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & -4 \\ 0 & -4 & 4 \end{bmatrix} = \begin{bmatrix} 4 & -4 & 0 \\ -4 & 8 & -4 \\ 0 & -4 & 4 \end{bmatrix}. \tag{11.18}$$

Since element 3 results from a 180° rotation of element 1, it happens to have exactly the same S^e. Assembling the four element matrices gives the six system equations $\mathbf{S}\,\mathbf{D} = \mathbf{C}$ where

$$\mathbf{S} = \begin{bmatrix} +4 & -4 & 0 & 0 & 0 & 0 \\ -4 & (+8+4+4) & (-4-4) & -4 & 0 & 0 \\ 0 & (-4-4) & (+4+8+4) & 0 & (-4-4) & 0 \\ 0 & -4 & 0 & +8 & -4 & 0 \\ 0 & 0 & (-4-4) & -4 & (+4+4+8) & -4 \\ 0 & 0 & 0 & 0 & -4 & +4 \end{bmatrix}$$

and

$$\mathbf{C} = \frac{QAt}{3}\begin{Bmatrix} 1 \\ 1+1+1 \\ 1+1+1 \\ 1 \\ 1+1+1 \\ 1 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ q_4 \\ q_5 \\ q_6 \end{Bmatrix} = \begin{Bmatrix} 4 \\ 12 \\ 12 \\ 4 \\ 12 \\ 4 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ q_4 \\ q_5 \\ q_6 \end{Bmatrix}.$$

In the above vector the qs are the nodal heat flux reactions required to maintain the specified external temperature. Since the last three equations have essential boundary conditions applied we can reduce the first three to

$$\begin{bmatrix} 4 & -4 & 0 \\ -4 & 16 & -8 \\ 0 & -8 & 16 \end{bmatrix}\begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{Bmatrix} = \begin{Bmatrix} 4 \\ 12 \\ 12 \end{Bmatrix} - \theta_4\begin{Bmatrix} 0 \\ -4 \\ 0 \end{Bmatrix} - \theta_5\begin{Bmatrix} 0 \\ 0 \\ -8 \end{Bmatrix} - \theta_6\begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}. \tag{11.19}$$

Substituting the data that the exterior surface temperatures are $\theta_4 = \theta_5 = \theta_6 = 5$ yields the reduced source term

$$\mathbf{C}^* = \begin{Bmatrix} 4 \\ 12 \\ 12 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 20 \\ 0 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 40 \end{Bmatrix} = \begin{Bmatrix} 4 \\ 32 \\ 52 \end{Bmatrix}.$$

Solving for the interior temperatures using the inverse

$$\mathbf{S}^{*-1} = \frac{1}{512}\begin{bmatrix} 192 & 64 & 32 \\ 64 & 64 & 32 \\ 32 & 32 & 48 \end{bmatrix}$$

and multiplying by $\mathbf{C}^*$ yields:

$$\mathbf{\Theta}^* = \begin{Bmatrix} 8.750 \\ 7.750 \\ 7.125 \end{Bmatrix} = \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{Bmatrix}.$$

Substituting these values into the original system equations will give the exterior nodal heat flux values (*thermal reactions*) required by this problem. For example, the fourth row reaction equation yields: $-4\theta_2 + 8\theta_4 - 4\theta_5 = -4(7.75) + 8(5) - 4(5) = 4 + q_4$, or simply $-15 = q_4$. The other two nodal fluxes are $q_5 = -29, q_6 = -4$. These sum to –48. The internal heat generated was $\sum_e Q^e A^e t^e = +48$, which is equal and opposite.

```
*** INPUT SOURCE RESULTANTS ***                                    ! 1
ITEM       SUM          POSITIVE      NEGATIVE                    ! 2
   1    4.8000E+01    4.8000E+01    0.0000E+00                     ! 3
                                                                   ! 4
*** REACTION RECOVERY ***                                          ! 5
NODE, PARAMETER,     REACTION, EQUATION                            ! 6
   4,  DOF_1,      -1.5000E+01     4                               ! 7
   5,  DOF_1,      -2.9000E+01     5                               ! 8
   6,  DOF_1,      -4.0000E+00     6                               ! 9
                                                                  !10
REACTION RESULTANTS                                               !11
PARAMETER,    SUM          POSITIVE     NEGATIVE                  !12
 DOF_1,     -4.8000E+01   0.0000E+00  -4.8000E+01                  !13
                                                                  !14
***  OUTPUT OF RESULTS IN NODAL ORDER  ***                        !15
NODE,  X-Coord,    Y-Coord,    DOF_1,                             !16
   1  0.0000E+00  0.0000E+00  8.7500E+00                          !17
   2  2.0000E+00  0.0000E+00  7.7500E+00                          !18
   3  2.0000E+00  2.0000E+00  7.1250E+00                          !19
   4  4.0000E+00  0.0000E+00  5.0000E+00                          !20
   5  4.0000E+00  2.0000E+00  5.0000E+00                          !21
   6  4.0000E+00  4.0000E+00  5.0000E+00                          !22
                                                                  !23
*** SUPER_CONVERGENT AVERAGED NODAL FLUXES ***                    !24
NODE,   X-Coord,    Y-Coord,    FLUX_1,     FLUX_2,               !25
   1  0.0000E+00  0.0000E+00 -3.3333E-01 -4.5833E+00              !26
   2  2.0000E+00  0.0000E+00 -7.3333E+00 -2.0833E+00              !27
   3  2.0000E+00  2.0000E+00 -4.8333E+00 -2.0833E+00              !28
   4  4.0000E+00  0.0000E+00 -1.4333E+01  4.1667E-01              !29
   5  4.0000E+00  2.0000E+00 -1.1833E+01  4.1667E-01              !30
   6  4.0000E+00  4.0000E+00 -9.3333E+00  4.1667E-01              !31
```

Figure 11.11 *Results for square bar thermal analysis*

Thus, we have verified that the generated heat equals the heat outflow. Of course, this must be true for all steady state heat conduction problems. Note that while we started with six equations from the integral formulation only three were independent equations for the unknown temperatures. The other equations were independent equations for the thermal reactions necessary to maintain the essential boundary conditions on the temperature. One does not have to assemble and solve the reaction set but it is a recommended procedure. The input data for this example are given in Fig. 11.10 and selected outputs are given in Fig. 11.11.

Replacing the previous mesh with one of 64 elements and applying the *MODEL* code gives the temperature results shown as contours and a surface in Figs. 11.12 and 13, respectively. In the first we get a visual check that the contours appear perpendicular to the insulated boundaries and parallel to the Dirichlet boundary. From Fig. 11.13 we get the impression that the temperature gradient and flux would also be smooth. However, they are constant in each element and must be made continuous by the SCP fit as seen in Fig. 11.14, which is a 2-D generalization of Fig. 2.21 in that the element flux values are constant. From the symmetry boundary conditions we expect zero heat flux at the $(0, 0)$. The continuous flux, in Fig. 11.15, misses that so more refinement is needed. The discontinuous element flux vectors are in Fig. 11.16 while the continuous nodal flux vectors are shown in Fig. 11.17. The last figure should have the vectors parallel to the insulated boundaries and they seem to do that reasonably well. The energy norm error estimates in Fig. 11.18 exceed 2 percent and are about ten times larger than we usually want. The projected maximum element sizes for a new mesh to reduce those errors are shown in Fig. 11.19.

The same four element T3 mesh was used in the text by Kwon [14], a former student of mine, to analyze a half symmetry triangular region, of Fig. 11.20, with given normal flux on the right edge, a null essential boundary condition along the bottom, and the inclined edge insulated. The modified input data and selected output are shown in Figs. 11.21 and 22, respectively. In this case the heat flux in (+) normal to the right edge was 2 per unit length, acting over the side length of 4, for a resultant input of 8. Here there were only two such flux line segments (with a local nodal resultant at each end of 2). Therefore, the source effects were simply lumped at the three nodes by inspection. The keyword *loads*, at line 12, flagged the presence of such sources and their numeric values were read in lines 38-40. (Such values are terminated by reading the source at the last degree of freedom, which is usually zero.)

For this class of problem we expect that the sum of the external sources will be equal and opposite to the reactions at the essential boundary conditions. Figure 11.22 shows that the expected result is obtained, as are the expected temperatures. The lumping of the heat flux was easy only because the flux was constant and we could get the boundary length it acted on by inspection. Usually we would have to supply the programming and data input necessary to formulate the boundary flux resultant arrays, $\mathbf{C}_q^e$. For the edge of the T3 element we need only the two nodal contributions. Implementing such a 'boundary flux segment' takes relatively little coding as shown in Fig. 11.7, but it requires much more flexibility in the data input options and the ability to recover those data. In most of the previous examples we considered only a mesh with a

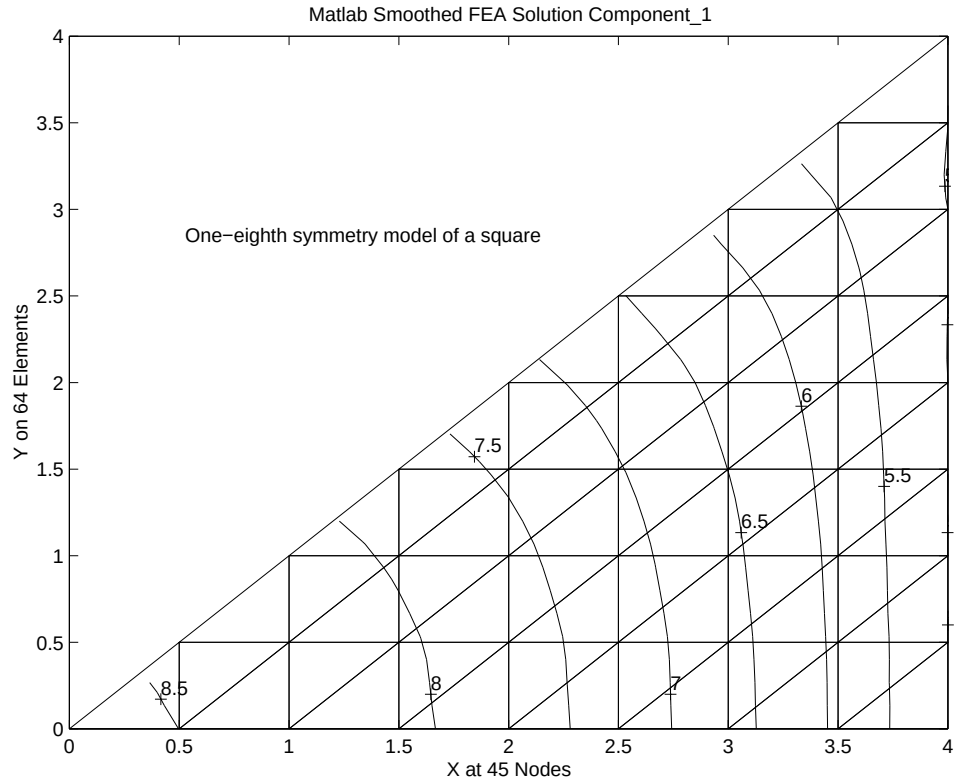


Figure 11.12 *Temperatures on the square segment*

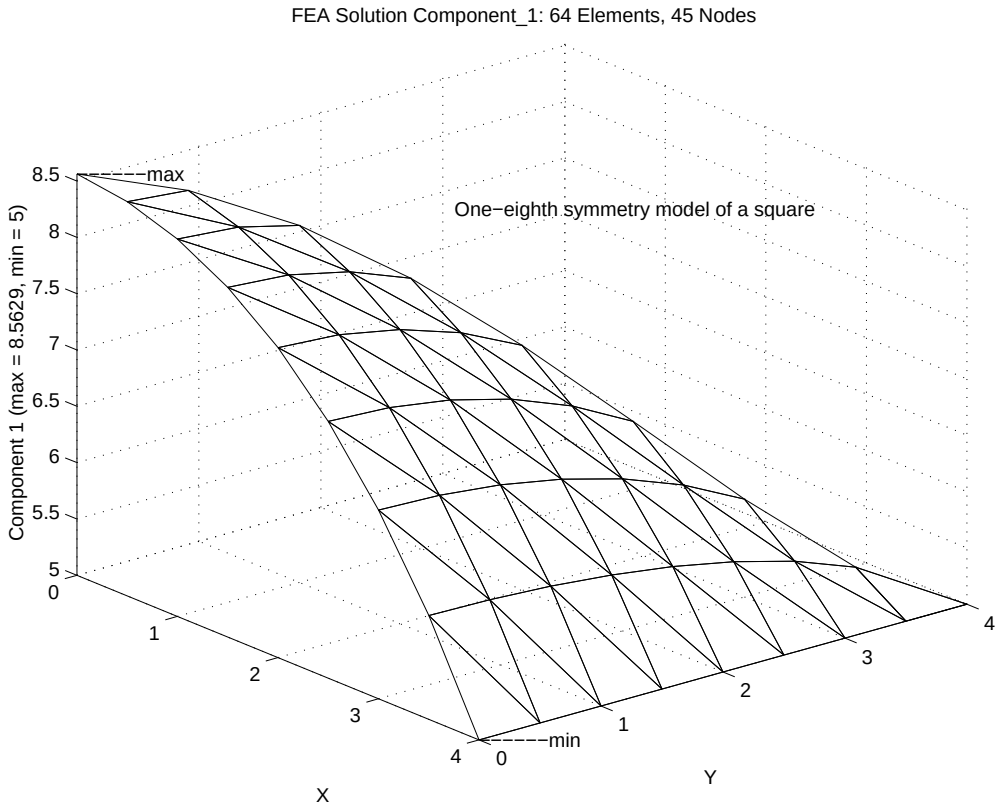


Figure 11.13 *Temperature carpet plot over the square*

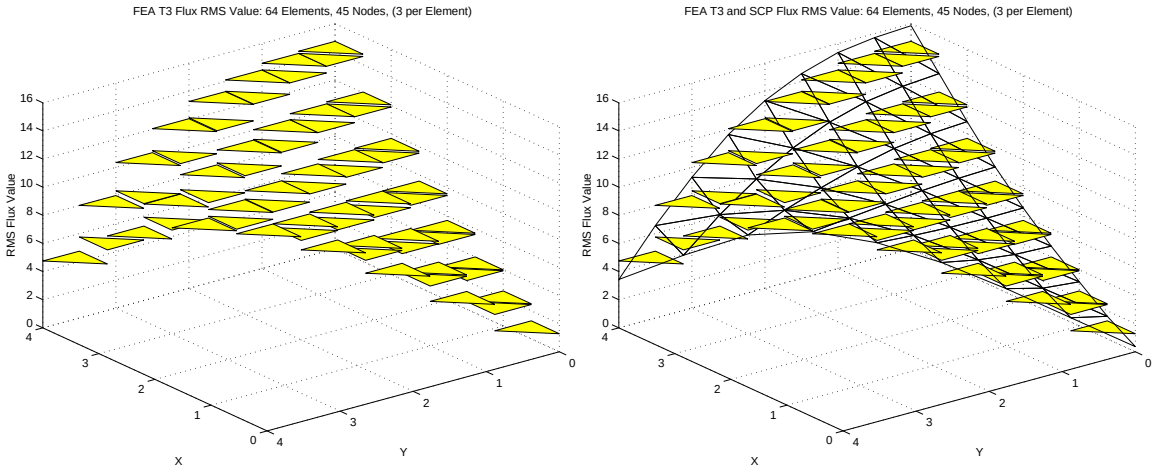


Figure 11.14 *Constant T3 fluxes and their SCP fit*

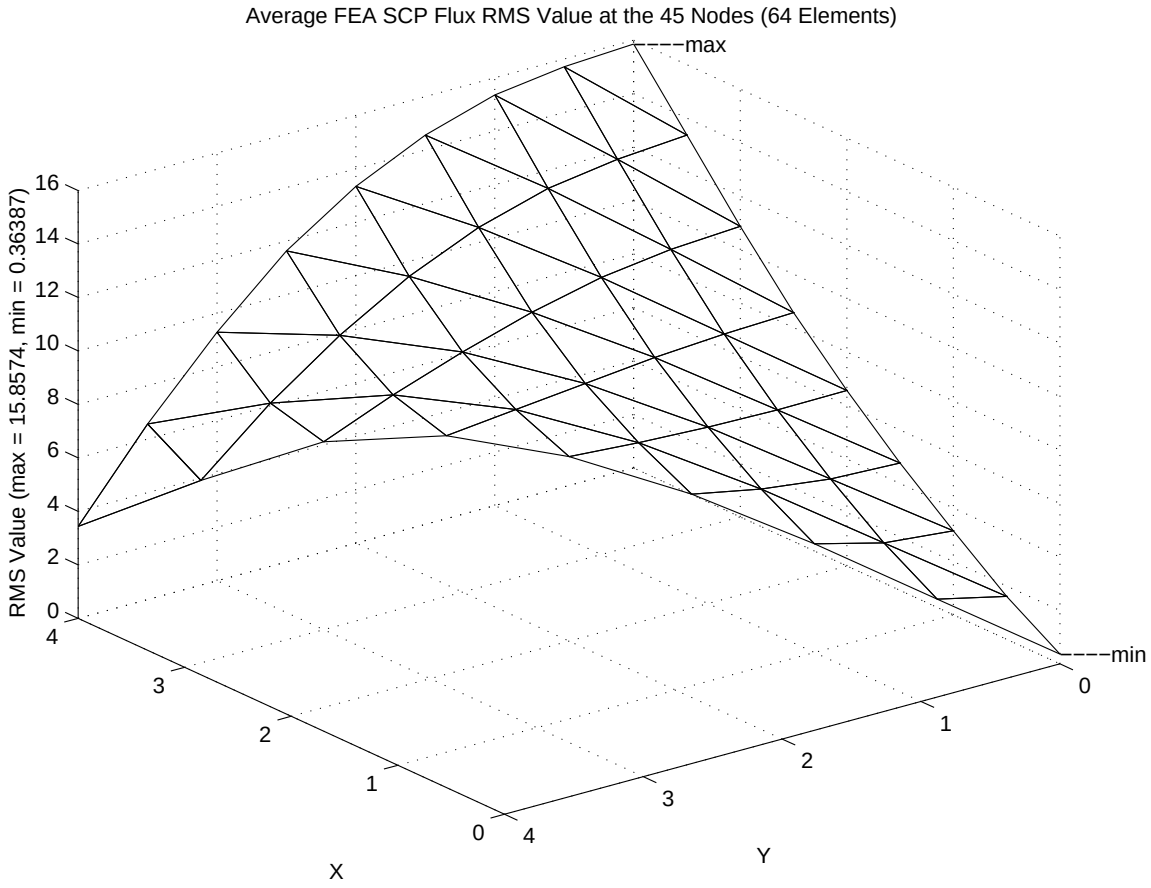


Figure 11.15 *Details of the SCP continuous nodal fluxes*

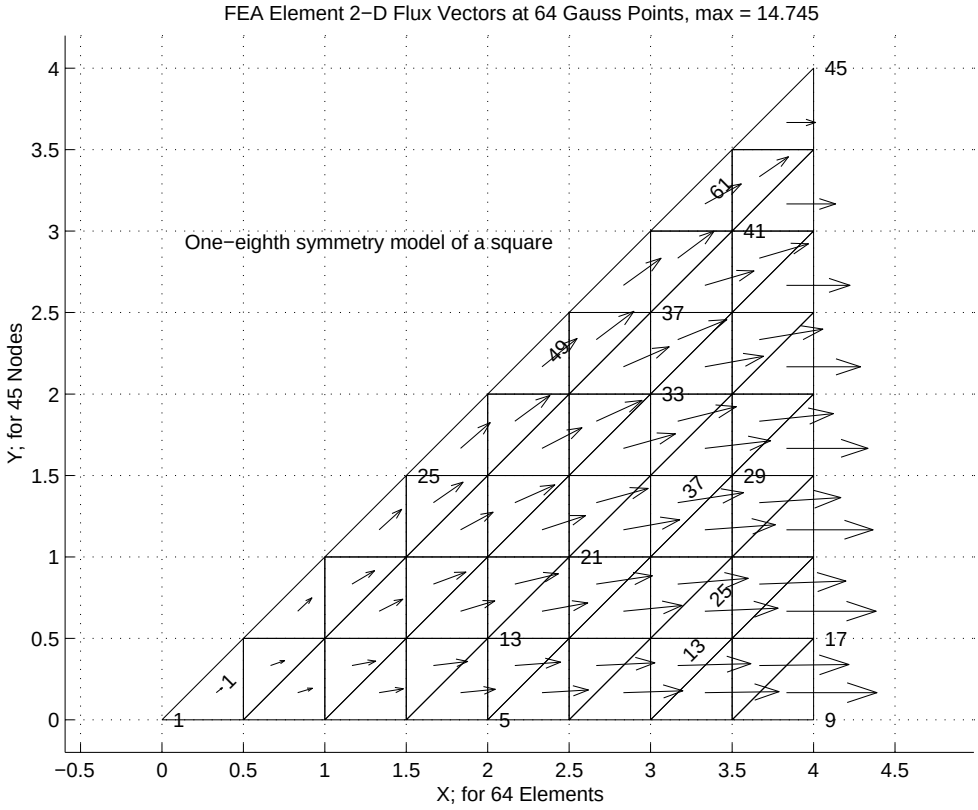


Figure 11.16 *Constant element flux vectors*

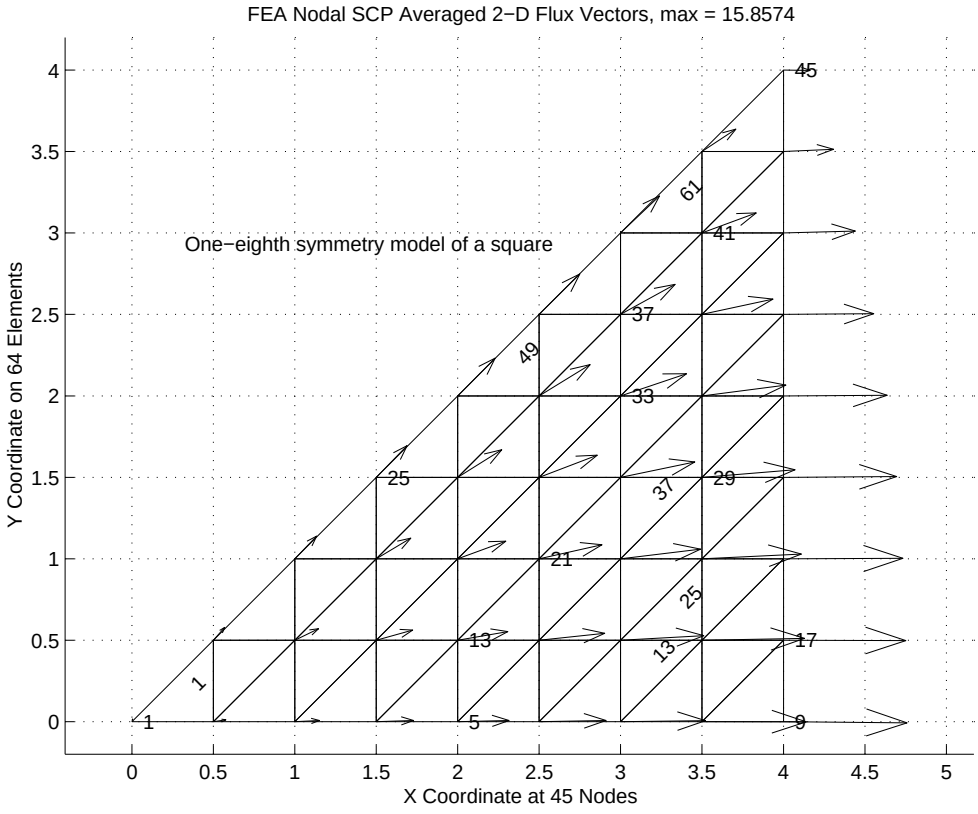


Figure 11.17 *Averaged flux vector over the square*

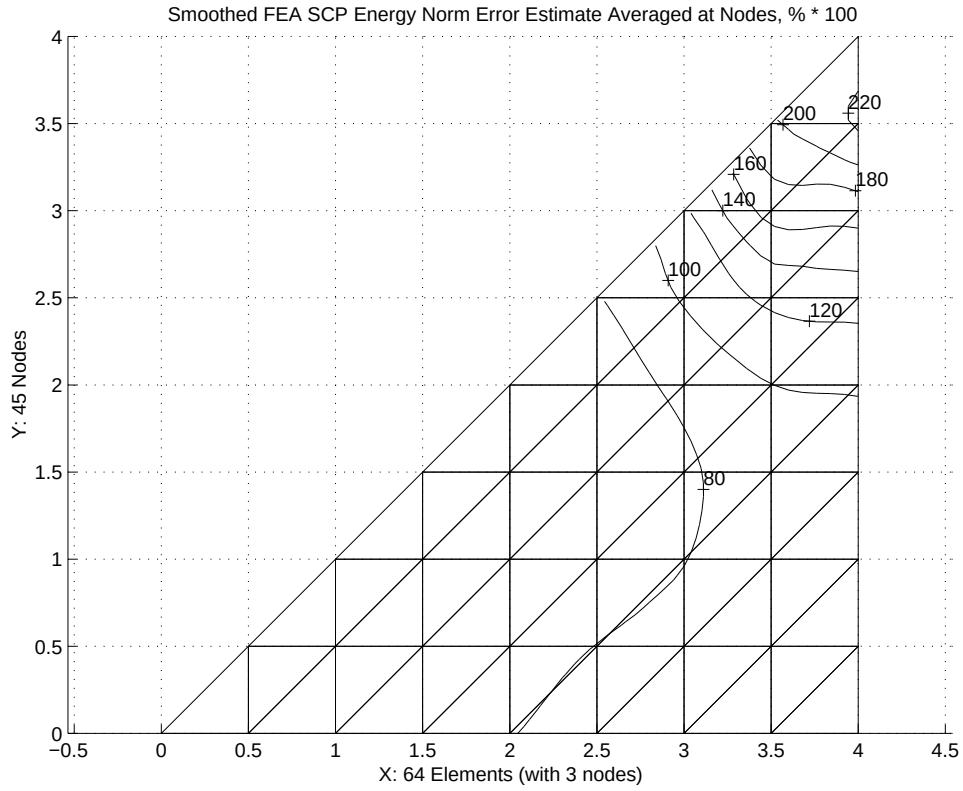


Figure 11.18 *Contours of energy norm error*

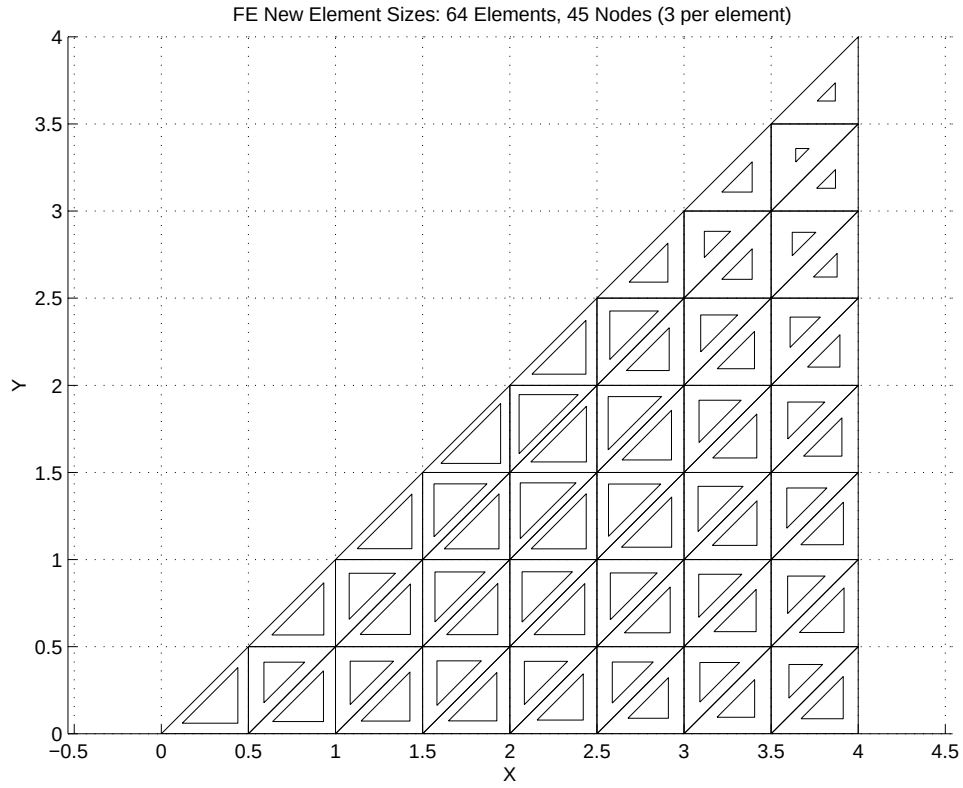


Figure 11.19 *Suggested maximum element sizes for new mesh*

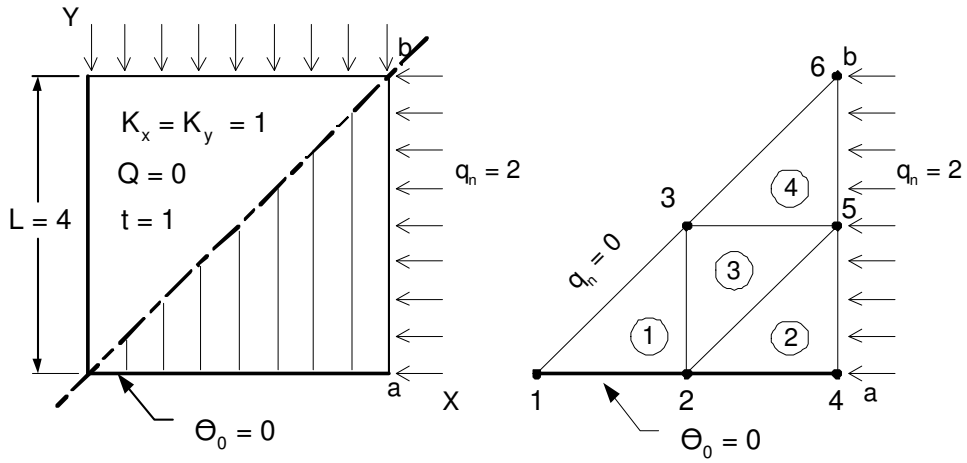


Figure 11.20 *Square with two flux sides*

```
title "T3 Conduction with given flux. Kwon example 5.4.1"    ! 1
example 204 ! Application source code library numbe          ! 2
b_rows    2 ! Number of rows in the B (operator) matrix      ! 3
dof       1 ! Number of unknowns per node                    ! 4
el_nodes  3 ! Maximum number of nodes per element            ! 5
elems     4 ! Number of elements in the system               ! 6
gauss     1 ! Maximum number of quadrature points            ! 7
nodes     6 ! Number of nodes in the mesh                    ! 8
space     2 ! Solution space dimension                       ! 9
el_homo     ! Element properties are homogeneous             !10
el_real   4 ! Number of real properties per element          !11
loads       ! An initial source vector is input              !12
remarks   9 ! Number of user remarks, e.g. property names    !13
end         ! Terminate the keyword control, remarks follow  !14
Kwon example 5.4.1, conduction with given flux.  6 <- q_n    !15
K_x = K_y = 1, 0 = 6, K_xy = 0                    / | <- q_n !16
L_1_4 = L_4_6 = 4, Thickness = 1                 /  | <- q_n !17
Edge 1_3_6 is insulated, so q_n = 0.            /(4)| <- q_n !18
Edge 4_5_6 has q_n =2, per unit length.    3-----5 <- q_n    !19
EBC on edge 1_2_4 is T = 0                / |(3)/| <- q_n    !20
K_x T,xx + 2K_xy T,xy + K_y T,yy + Q = 0 /  | /  | <- q_n    !21
Solution gives T_3 = 3, T_6 = 11.       /(1)|/(2)| <- q_n    !22
Edge 4_5_6 lumped flux = 2, 4, 2.      1----2----4 <- q_n    !23
   1    1 0.   0. ! node, ebc flag, x, y                     !24
   2    1 2.   0.                                            !25
   3    0 2.   2.                                            !26
   4    1 4.   0.                                            !27
   5    0 4.   2.                                            !28
   6    0 4.   4.                                            !29
  1  1  2  3      ! elem, three nodes                         !30
  2  2  4  5                                                 !31
  3  2  5  3                                                 !32
  4  3  5  6                                                 !33
   1    1 0.      ! node, dof, value of EBC                  !34
   2    1 0.                                                 !35
   4    1 0.                                                 !36
  1 1. 1. 0. 0.   ! elem, K_x, K_y, K_xy, Q (homogeneous)    !37
4 1 2.0  ! node, dof, lumped heat flux                       !38
5 1 4.0  ! node, dof, lumped heat flux                       !39
6 1 2.0  ! node, dof, lumped heat flux                       !40
```

Figure 11.21 *Triangular region with an edge flux*

single type of element. But it is common to mix elements like triangles, quadrilaterals, and line elements, so long as they have compatible edge interpolations. If we are going to allow multiple types of elements to be mixed then we must be able to define the number of nodes, shape, integration rule, properties, etc. for each. We will also split how we think about their purpose. Most will simply be thought of as 'standard elements', while others will exist to treat 'boundary flux segments' or to treat 'mixed (Robin) segment regions'. Their nodal connectivities and properties will be input in that order. The following examples will introduce some of the new free format keyword controls that *MODEL* employs to distinguish between these element types and the amount of data the user wishes to assign to each (if any). Table 11.1 lists most of the controls that can be selected to define combinations of element types.

To extend the previous lumped flux example to one that uses the source in Fig. 11.7 we must modify the prior data to allow the combination of some conduction elements with two flux boundary segment elements. The new controls and corresponding data are given in Fig. 11.23. Since a constant normal flux is quite common a control option *normal_flux* (line 15) is made available to the user for assigning a value to a global variable, Q_NORMAL_SEG, that can be used in application dependent arrays. Should the normal flux data not be constant everywhere one could define a different constant on each edge by using properties defined for each boundary segment (BS). The new data, in Fig. 11.23, yield exactly the same temperature results as before. The keyword *segments* 2 (line 14) caused the boundary integral calculations of Fig. 11.7 to be invoked, and told the system that two segments needed to be read (at lines 38-39). Expanding the mesh size to include 64 conduction triangles and 8 edge flux boundary segments yields the mesh (note right side) and the temperatures of Figs. 11.24 and 25, respectively.

11.5.2 Face convection

Two-dimensional models that combine the conduction through an area with convection from one or both of its faces often approximate cooling fins. We refer to such regions (points, lines, or surfaces) as 'mixed segments' or Robin segments. For the linear triangle it is again practical to write the matrices, defined in Eq. 11.11, in closed form and they were given in Fig. 11.8 for constant data as:

$$\mathbf{S}_h^b = \frac{h^b A^b}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}, \quad \mathbf{C}_h^b = \frac{h_b A^b \Theta_\infty^b}{3}\begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}. \tag{11.20}$$

Again, it is not uncommon for the convection data, h^b, θ_∞, to be constant. To allow for that condition, two user keyword control options are provided: *convect_coef* and *convect_temp*. The second one (the surrounding fluid temperature) defaults to zero. They can be used to set the corresponding values of two global variables (with the same names) for possible use in application dependent matrices, as was done in Fig. 11.8. Should the mixed segment (MX) data not be constant everywhere one could define different constants on each face by using properties defined for each mixed segment.

```
*** INITIAL FORCING VECTOR DATA ***                                    ! 1
NODE   PARAMETER     VALUE    EQUATION                                 ! 2
   4         1   2.00000E+00       4                                  ! 3
   5         1   4.00000E+00       5                                  ! 4
   6         1   2.00000E+00       6                                  ! 5
*** RESULTANTS ***                                                     ! 6
COMPONENT         SUM        POSITIVE      NEGATIVE                   ! 7
 IN_1,         8.0000E+00    8.0000E+00    0.0000E+00                 ! 8
                                                                       ! 9
*** REACTION RECOVERY ***                                             !10
NODE, PARAMETER,     REACTION, EQUATION                               !11
   1,  DOF_1,        0.0000E+00     1                                  !12
   2,  DOF_1,       -3.0000E+00     2                                  !13
   4,  DOF_1,       -5.0000E+00     4                                  !14
REACTION RESULTANTS                                                   !15
PARAMETER,    SUM          POSITIVE     NEGATIVE                      !16
 DOF_1,     -8.0000E+00   0.0000E+00  -8.0000E+00                      !17
                                                                      !18
***  OUTPUT OF RESULTS IN NODAL ORDER  ***                            !19
NODE,  X-Coord,    Y-Coord,    DOF_1,                                 !20
   1  0.0000E+00  0.0000E+00  0.0000E+00                               !21
   2  2.0000E+00  0.0000E+00  0.0000E+00                               !22
   3  2.0000E+00  2.0000E+00  3.0000E+00                               !23
   4  4.0000E+00  0.0000E+00  0.0000E+00                               !24
   5  4.0000E+00  2.0000E+00  6.0000E+00                               !25
   6  4.0000E+00  4.0000E+00  1.0000E+01                               !26
```

Figure 11.22 *Selected results for triangle with an edge flux*

Table 11.1 *Typical keywords for multiple element types*

```
WORD,     VALUES ! REMARKS                                    [DEFAULT]
area_thick   1.5 ! Global thickness of 2-D domain                  [1]
el_segment     3 ! Maximum nodes on element boundary segment       [0]
el_types       1 ! Number of different types of elements           [1]
type_parm    2 1 ! Parametric space for each element type          [d]
type_nodes   3 2 ! Number of analysis nodes for element types      [2]
type_gauss   1 2 ! Number of Gauss points in each element type     [0]

segments       1 ! Number of element segments with flux input      [0]
normal_flux   5. ! Constant normal flux on all flux segments      [0]
flux_thick   1.2 ! Thickness of all flux load lines or points     [1]
seg_int        1 ! Number of integer properties per segment       [0]
seg_pt_flux    1 ! Segment flux components input at flux nodes    [1]
seg_real       3 ! Number of real properties per segment          [0]
seg_thick    1.2 ! Thickness of all flux and mixed segments       [1]

mixed_segs     1 ! Number of mixed boundary condition segments    [0]
mixed_int      0 ! Number of integer properties per mixed_bc      [0]
mixed_real     3 ! Number of real properties per mixed_bc         [0]
convect_coef  1. ! Convection coefficient on all mixed segments   [0]
convect_temp  1. ! Convection temperature on all mixed segments   [0]
convect_thick 2. ! Thickness of all convection lines or points    [1]
convect_vary     ! Convection, different on all mixed segments    [F]

type_shape   2 2 ! Shape code of each element type                 [1]
type_geom    3 2 ! Number of geometric nodes for type  [type_nodes]
```

```
title "T3 Conduction with given flux. Via flux elements"      ! 1
example  204 ! Application source code library numbe          ! 2
remarks    9 ! Number of user remarks, e.g. property names    ! 3
b_rows     2 ! Number of rows in the B (operator) matrix      ! 4
dof        1 ! Number of unknowns per node                    ! 5
elems      4 ! Number of elements in the system               ! 6
nodes      6 ! Number of nodes in the mesh                    ! 7
space      2 ! Solution space dimension                       ! 8
el_homo      ! Element properties are homogeneous             ! 9
el_real    4 ! Number of real properties per element         !10
el_types     2 ! Number of different types of elements       !11
type_nodes 3 2 ! Number of analysis nodes for element types  !12
type_shape 2 1 ! Shape code of each element type             !13
segments     2 ! Number of element segments with flux input  !14
normal_flux 2. ! Constant normal flux on all flux segments   !15
el_segment   2 ! Maximum nodes on element boundary segment   !16
no_error_est   ! Do NOT compute SCP element error estimates  !17
end ! Terminate keyword control input, remarks follow        !18
Kwon example 5.4.1, conduction with given flux.  6 <- q_n    !19
K_x = K_y = 1, 0 = 6, K_xy = 0                  / | <- q_n   !20
L_1_4 = L_4_6 = 4, Thickness = 1               /  | <- q_n   !21
Edge 1_3_6 is insulated, so q_n = 0.          /(4)| <- q_n   !22
Edge 4_5_6 has q_n =2, per unit length.    3-----5 <- q_n   !23
EBC on edge 1_2_4 is T = 0                / |(3)/| <- q_n   !24
K_x T,xx + 2K_xy T,xy + K_y T,yy + Q = 0 /  | /  | <- q_n   !25
Solution gives T_3 = 3, T_6 = 11.        /(1)|/(2)| <- q_n   !26
Edges 4_5, 5_6 normal flux             1----2----4 <- q_n   !27
   1    1 0.   0. ! node, ebc flag, x, y                     !28
   2    1 2.   0.                                            !29
   3    0 2.   2.                                            !30
   4    1 4.   0.                                            !31
   5    0 4.   2.                                            !32
   6    0 4.   4. ! last node                                !33
1  1 1  2  3  ! standard elem, el_type, three nodes          !34
2  1 2  4  5  ! standard elem, el_type, three nodes          !35
3  1 2  5  3  ! standard elem, el_type, three nodes          !36
4  1 3  5  6  ! standard elem, el_type, three nodes          !37
1 2 4 5 ! flux segment, el_type, two edge nodes              !38
2 2 5 6 ! flux segment, el_type, two edge nodes              !39
 1    1 0.  ! node, dof, value of EBC                       !40
 2    1 0.  ! node, dof, value of EBC                       !41
 4    1 0.  ! node, dof, value of EBC                       !42
1 1. 1. 0. 0. ! elem, K_x, K_y, K_xy, Q (homogeneous)       !43
```

Figure 11.23 *Combining edge flux segments with conduction*

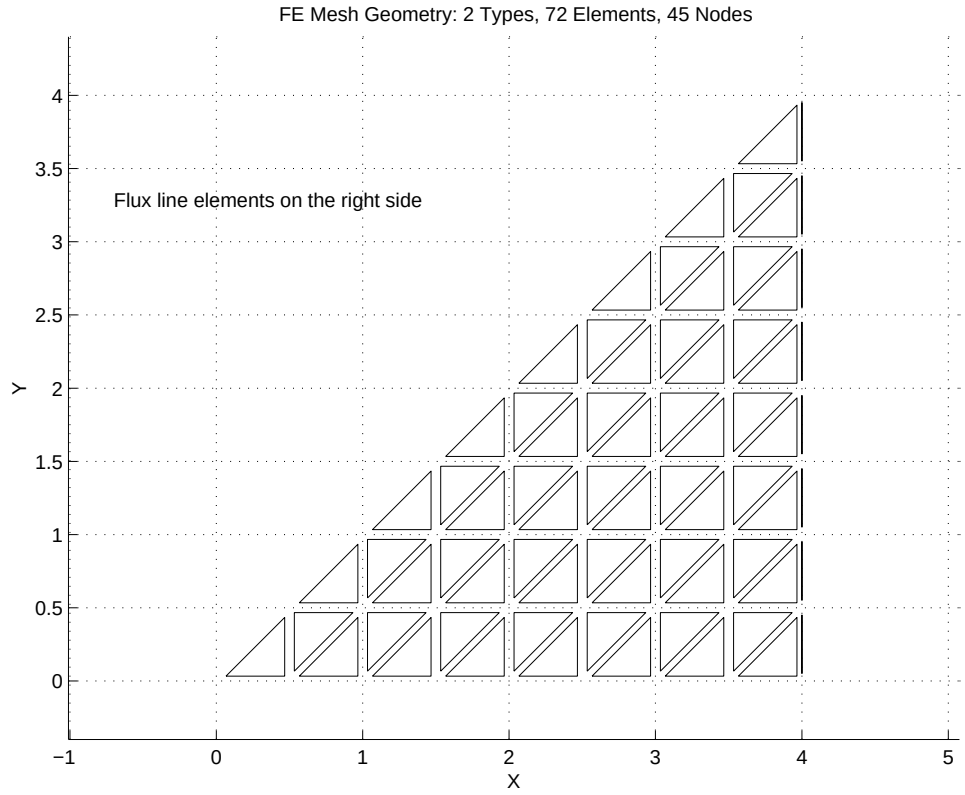


Figure 11.24 *Exploded conduction triangles and flux line segments*

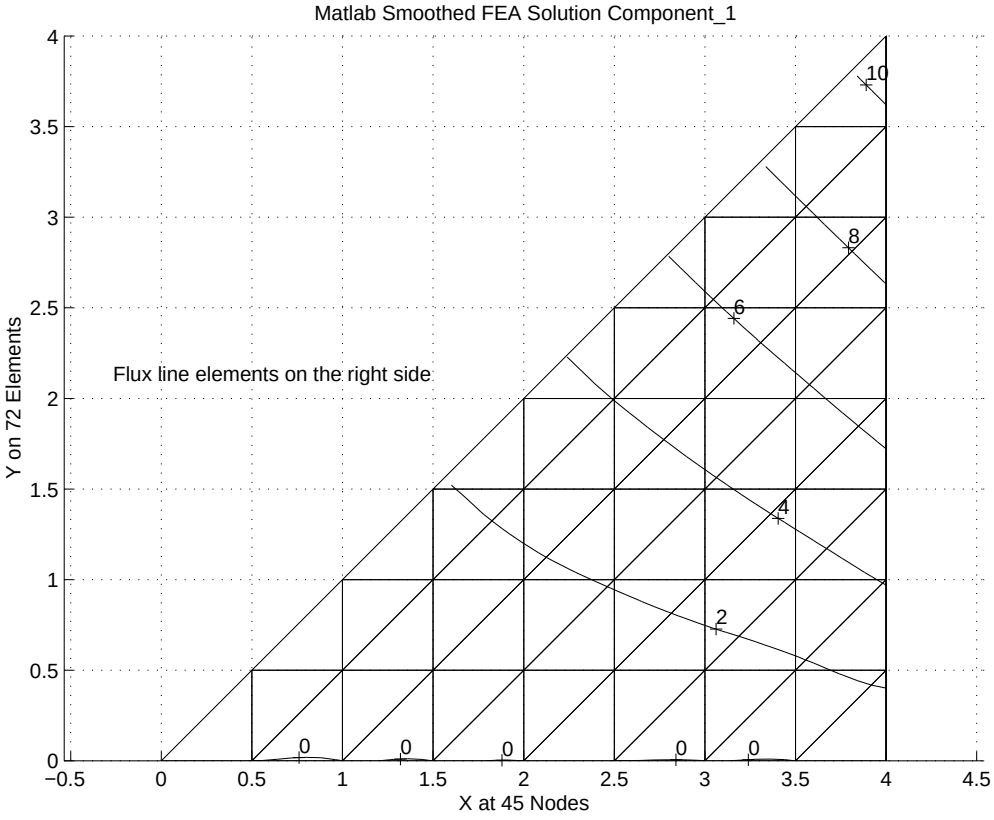


Figure 11.25 *Temperatures from edge flux sources*

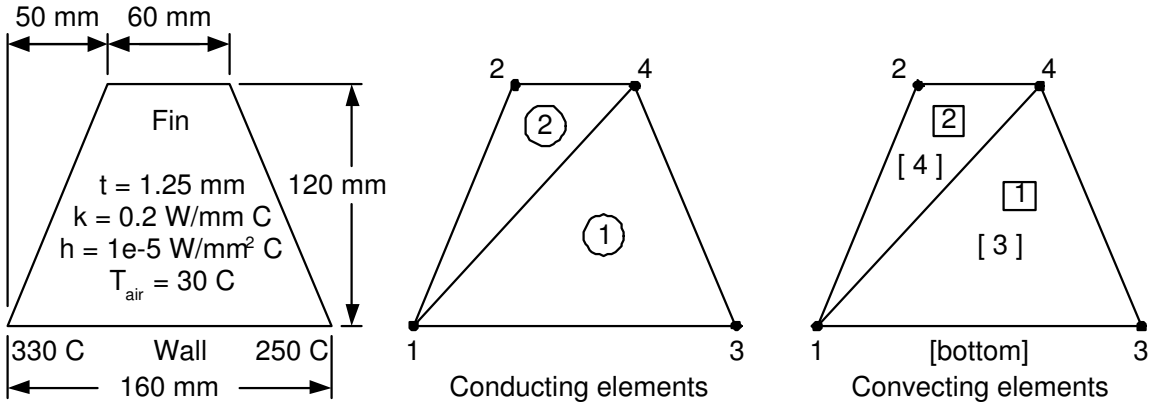


Figure 11.26 *A cooling fin with air convection on its faces*

As a simple example of a cooling fin we will consider the trapezoidal fin [2] given in detail by Allaire. He used a two linear triangle model that had convection on the top and bottom faces, and a linearly varying given temperature on the wall edge. The given problem is shown in Fig. 11.26. The corresponding data to invoke the face convection segments, as well as the standard conduction, are given in Fig. 11.27. The keyword *mixed_segs* 4 (line 14) is one way to cause the convection calculations of Fig. 11.8 to be invoked and the four sets of connectivity data to be read (lines 40-43). Two convecting face elements were on top of the fin and two were on the bottom. The model is so crude there is little precision in the temperatures in the selected output, which is given in Fig. 11.28. That figure also shows the numerical values of the matrices from each conducting and convecting element should the reader wish to verify their calculation. The system reactions, on lines 46-52, are significant. They show that to maintain the temperatures given in the two essential boundary conditions a total of 57.73 W of heat must flow into the fin. Shortly, when we consider the post-processing calculation for the convection heat loss we will find that exactly the same amount of heat flow is convected away. That is a reminder that a finite element is always flux conserving, when the fluxes are calculated properly (contrary to a common misconception).

When numerical integration is used *MODEL* lists the flux at each quadrature point. Here we have used a closed form expression for $\mathbf{S}^e$, but it corresponds to a one-point integration rule (as seen in Table 9.3). The necessary flux data was saved for the centroid of each element (in lines 17, 35, and 45 of Fig. 11.6). Then the nodal temperatures were gathered and multiplied by $-\mathbf{E}^e\mathbf{B}^e$ to yield the flux vector $\mathbf{q}^e = -\mathbf{E}^e\mathbf{B}^e\mathbf{T}^e$ at the point. If we try to use the element gradients (since the centroid is far from the boundary) to estimate the heat flow we get about 40.3 W for 42 percent flux error. For a more accurate heat loss calculation we would need to evaluate Eq. 7.37 by summing over each convection (mixed) segment. Such a post-processing implementation for the linear triangle convecting face is given in Fig. 11.29. The above flux recovery and the convection losses are given in Fig. 11.30. There (in line 13) we see the convection loss matches the thermal reactions of 57.73 W (line 52 of Fig. 11.28), as expected. The heat balances and element heat flux vectors for this crude mesh are shown in Fig. 11.31.

```
title "Fin face convection, Allaire, Basics FEM p. 343-353 2T3"    ! 1
example  204 ! Application source code library number              ! 2
elems      2 ! Number of elements in the system                    ! 3
nodes      4 ! Number of nodes in the mesh                         ! 4
space      2 ! Solution space dimension                            ! 5
b_rows     2 ! Number of rows in the B (operator) matrix           ! 6
dof        1 ! Number of unknowns per node                         ! 7
el_homo      ! Element properties are homogeneous                  ! 8
el_real    5 ! Number of real properties per element               ! 9
remarks   13 ! Number of user remarks, e.g. property names         !10
convect_coef 1.e-5 ! Convection coefficient on all mixed segments  !11
convect_temp    30 ! Convection temperature on all mixed segments  !12
el_segment       3 ! Maximum nodes on element boundary segment     !13
mixed_segs       4 ! Number of mixed boundary condition segments   !14
el_types         2 ! Number of different types of elements         !15
type_gauss    1  1 ! Number of Gauss points in each element type   !16
type_nodes    3  3 ! Number of nodes on each element type          !17
type_shape    2  2 ! Shape code of each element type               !18
post_mixed         ! Post-process mixed segments, create n_file2   !19
end ! Terminate the keyword control, remarks follow                !20
Trapezoidal fin: 160 mm and 60 mm by 120 mm high (no edge conv.)   !21
Units: x,y-mm,  k-W/mm C, h-W/mm^2 C, t-C, Q-W/mm^3, q-W/mm^2      !22
Face convection uses constant convect_coef, convect_temp           !23
Conduction: K_xx U,xx + 2K_xy U,xy + K_yy U,yy + Q = 0             !24
PROP(1) = CONDUCTIVITY K_XX, PROP(2) = CONDUCTIVITY K_YY           !25
PROP(3) = CONDUCTIVITY K_XY, PROP(4) = SOURCE PER UNIT AREA        !26
PROP(5) = THICKNESS (DEFAULT 1.0), here 1.25mm                     !27
Convection: -K_n * U,n = CONVECT_COEF ( U - CONVECT_TEMP)          !28
Optional mixed properties: 1=CONVECT_COEF, 2=CONVECT_TEMP          !29
The conduction reaction total of 57.73 W is equal and opposite     !30
to the convection loss integral of 57.73 W, (actual is about       !31
57.66 W) BUT using element gradients gives 40.34 W for 42 %        !32
flux error.  (Reverse the signs on flux listings.)                 !33
 1    1    0.0    0.0 ! node, bc-flag,x, y                        !34
 3    1  160.0    0.0                                            !35
 2    0   50.0  120.0                                            !36
 4    0  110.0  120.0                                            !37
1 1    1 3 4 ! elem, el_type, 3 nodes. conducting                  !38
2 1    2 1 4 ! elem, el_type, 3 nodes. conducting                  !39
1 2    1 3 4 ! face, el_type, 3 nodes. convecting, top             !40
2 2    2 1 4 ! face, el_type, 3 nodes. convecting, top             !41
3 2    1 3 4 ! face, el_type, 3 nodes. convecting, bottom          !42
4 2    2 1 4 ! face, el_type, 3 nodes. convecting, bottom          !43
  1  1  330. ! node, dof, value                                    !44
  3  1  250. ! node, dof, value                                    !45
 1 0.2 0.2 0.0 0.0 1.25 ! el, k_x, k_y, K_xy, Q, thick            !46
```

Figure 11.27 *Cooling fin element types data*

```
BEGINNING STANDARD ELEMENT ASSEMBLY  (debug output)            ! 1
S matrix:                              ! conduction            ! 2
ROW/COL      1          2          3                           ! 3
   1  1.10E-01 -5.79E-02 -5.21E-02                             ! 4
   2 -5.79E-02  1.73E-01 -1.15E-01                             ! 5
   3 -5.21E-02 -1.15E-01  1.67E-01                             ! 6
S matrix:                              ! conduction            ! 7
ROW/COL      1          2          3                           ! 8
   1  4.60E-01 -1.15E-01 -3.45E-01                             ! 9
   2 -1.15E-01  6.25E-02  5.21E-02                            !10
   3 -3.45E-01  5.21E-02  2.93E-01                            !11
BEGINNING MIXED_BC SEGMENTS ASSEMBLY                          !12
S matrix:                             ! convect top           !13
ROW/COL      1          2          3                          !14
   1  1.60E-02  8.00E-03  8.00E-03                            !15
   2  8.00E-03  1.60E-02  8.00E-03                            !16
   3  8.00E-03  8.00E-03  1.60E-02                            !17
C matrix:                             ! convect top           !18
ROW/COL      1          2          3                          !19
   1  9.60E-01  9.60E-01  9.60E-01                            !20
S matrix:                             ! convect top           !21
ROW/COL      1          2          3                          !22
   1  6.00E-03  3.00E-03  3.00E-03                            !23
   2  3.00E-03  6.00E-03  3.00E-03                            !24
   3  3.00E-03  3.00E-03  6.00E-03                            !25
C matrix:                             ! convect top           !26
ROW/COL      1          2          3                          !27
   1  3.60E-01  3.60E-01  3.60E-01                            !28
S matrix:                              ! convect bottom       !29
ROW/COL      1          2          3                          !30
   1  1.60E-02  8.00E-03  8.00E-03                            !31
   2  8.00E-03  1.60E-02  8.00E-03                            !32
   3  8.00E-03  8.00E-03  1.60E-02                            !33
C matrix:                              ! convect bottom       !34
ROW/COL      1          2          3                          !35
   1  9.60E-01  9.60E-01  9.60E-01                            !36
S matrix:                              ! convect bottom       !37
ROW/COL      1          2          3                          !38
   1  6.00E-03  3.00E-03  3.00E-03                            !39
   2  3.00E-03  6.00E-03  3.00E-03                            !40
   3  3.00E-03  3.00E-03  6.00E-03                            !41
C matrix:                              ! convect bottom       !42
ROW/COL      1          2          3                          !43
   1  3.60E-01  3.60E-01  3.60E-01                            !44
                                                             !45
*** REACTION RECOVERY ***                                    !46
NODE, PARAMETER,     REACTION, EQUATION                      !47
   1,  DOF_1,       3.9927E+01     1                         !48
   3,  DOF_1,       1.7806E+01     3                         !49
REACTION RESULTANTS                                          !50
PARAMETER,    SUM          POSITIVE     NEGATIVE            !51
 DOF_1,      5.7733E+01   5.7733E+01   0.0000E+00            !52
                                                             !53
***  OUTPUT OF RESULTS IN NODAL ORDER  ***                   !54
NODE,  X-Coord,    Y-Coord,    DOF_1,                        !55
  1  0.0000E+00  0.0000E+00  3.3000E+02                       !56
  2  5.0000E+01  1.2000E+02  2.0556E+02                       !57
  3  1.6000E+02  0.0000E+00  2.5000E+02                       !58
  4  1.1000E+02  1.2000E+02  1.7817E+02                       !59
```

Figure 11.28 *Selected convecting fin results*

```
! .....................................................................   ! 1
!  *** POST_PROCESS_MIXED PROBLEM DEPENDENT STATEMENTS FOLLOW ***        ! 2
! .....................................................................   ! 3
!          (Stored as application source example 204.)                  ! 4
! Global CONVECT_COEF set by keyword convect_coef is available           ! 5
! Global CONVECT_TEMP set by keyword convect_temp is available           ! 6
! H_INTG (LT_N) Integral of interpolation functions, H, available        ! 7
                                                                         ! 8
! Linear triangle face convection heat loss recover                      ! 9
  REAL(DP) :: X_I, X_J, X_K, Y_I, Y_J, Y_K     ! Global coordinates !10
  REAL(DP) :: A_I, A_J, A_K, B_I, B_J, B_K     ! Standard geometry  !11
  REAL(DP) :: C_I, C_J, C_K, X_CG, Y_CG, TWO_A ! Standard geometry  !12
  REAL(DP), SAVE :: Q_LOSS, TOTAL       ! Face and total heat loss  !13
                                                                     !14
  LOGICAL, SAVE :: FIRST = .TRUE.  ! printing                        !15
                                                                     !16
   IF ( FIRST ) THEN                   ! first call                  !17
     FIRST = .FALSE. ; WRITE (6, 5)    ! print headings              !18
     5 FORMAT ('*** CONVECTION HEAT LOSS ***', /, &                  !19
     &         'ELEMENT    HEAT_LOST')                               !20
     TOTAL = 0.d0                                                    !21
   END IF ! first call                                               !22
                                                                     !23
!   DEFINE NODAL COORDINATES, CCW: I, J, K                          !24
   X_I = COORD (1,1) ;    X_J = COORD (2,1) ;    X_K = COORD (3,1)  !25
   Y_I = COORD (1,2) ;    Y_J = COORD (2,2) ;    Y_K = COORD (3,2)  !26
                                                                     !27
!   GEOMETRIC PARAMETERS: H_I (X,Y) = (A_I + B_I*X + C_I*Y)/TWO_A   !28
   A_I = X_J * Y_K - X_K * Y_J ; B_I = Y_J - Y_K ; C_I = X_K - X_J  !29
   A_J = X_K * Y_I - X_I * Y_K ; B_J = Y_K - Y_I ; C_J = X_I - X_K  !30
   A_K = X_I * Y_J - X_J * Y_I ; B_K = Y_I - Y_J ; C_K = X_J - X_I  !31
                                                                     !32
!   CALCULATE TWICE ELEMENT AREA                                    !33
   TWO_A = A_I + A_J + A_K              ! = B_J*C_K - B_K*C_J also  !34
                                                                     !35
!   HEAT LOST FROM THIS FACE: Integral over face of h * (T - T_inf) !36
   H_INTG (1:3) = TWO_A / 6           ! Integral of H array        !37
   D (1:3) = D(1:3) - CONVECT_TEMP   ! Temp difference at nodes    !38
   Q_LOSS = CONVECT_COEF * DOT_PRODUCT (H_INTG, D)  ! Face loss    !39
   TOTAL  = TOTAL + Q_LOSS            ! Running total              !40
                                                                   !41
   PRINT '(I6, ES15.5)', IE, Q_LOSS                                !42
   IF ( IE == N_MIXED ) PRINT *, 'TOTAL = ', TOTAL                 !43
!  *** END POST_PROCESS_MIXED PROBLEM DEPENDENT STATEMENTS ***     !44
```

Figure 11.29 *Convecting mixed segment heat loss recovery*

```
*** FLUX COMPONENTS AT ELEMENT INTEGRATION POINTS ***                 ! 1
ELEMENT, PT,  X-Coord,    Y-Coord,    FLUX_1,     FLUX_2             ! 2
      1  1  9.0000E+01  4.0000E+01  1.2500E-01  2.0172E-01            ! 3
      2  1  5.3333E+01  8.0000E+01  1.1412E-01  2.1169E-01            ! 4
                                                                     ! 5
*** CONVECTION HEAT LOSS ***                                         ! 6
 ELEMENT      HEAT_LOST                                              ! 7
     1      2.13815E+01                                             ! 9
     2      7.48482E+00                                            !10
     3      2.13815E+01                                            !11
     4      7.48482E+00                                            !12
 TOTAL =   57.73267      <=== note <===                             !13
```

Figure 11.30 *Additional convecting fin results*

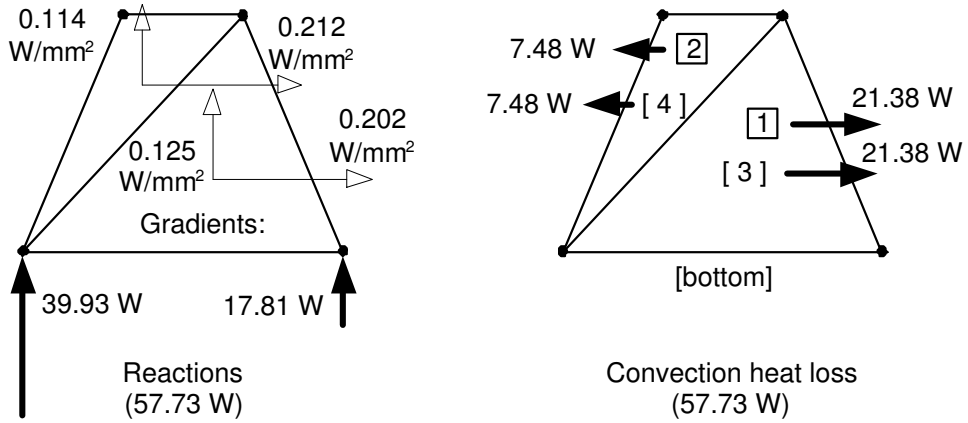


Figure 11.31 *A cooling fin heat balance*

With the numerically integrated formulations we have the ability to use any number of the linear, quadratic, and cubic elements in the *MODEL* library. We simply must generate more data to improve the accuracy. Mesh generators are used for that purpose. Here we will divide each edge with 5 nodes leading to a total of 25 nodes. Then we could use 32 T3, 16 Q4, 8 T6, 4 Q9, or 2 T15 elements in the mesh. Here we will graphically summarize the linear triangles and cubic quadrilateral results. The meshes, temperature contours, and the smoothed heat flux vectors, from the SCP, are shown in Fig. 11.32. The flux vectors should be parallel to the three outer edges since we assumed them insulated. If we revised the data to include edge convection on those three edges we would get more correct results, but they are so thin it probably is not worth the effort. The fin temperature distribution is given as a surface in Fig. 11.33. There the dashed vertical lines can be used with the left (z-axis) scale to obtain local nodal values.

The most important aspect of selecting various element types is assuring that the proper number of quadrature points are selected for each element or segment shape and polynomial degree. Here the conduction element integrand is a lower degree polynomial than for the convection segment. For the T3 element we specified a 1 point rule for conduction, and a 3 point rule for the convection matrix. The most important data changes for this Q9 element calculation are given in Fig. 11.34, while selected output results are in Fig. 11.35. In the latter we note that the conduction reactions and the convection heat loss results are again equal and opposite (lines 26 and 57), but slightly lower than in the very crude two element model.

Examining the energy norm error estimates for the linear triangle mesh in Fig. 11.32 (left) it exceeds 9 percent, as shown in Fig.11.36 so the projected mesh refinement is obtained as illustrated in Fig. 11.37. Employing those suggested element sizes to create a new mesh, with 215 T3 elements, one gets the temperature surface shown in Fig. 11.38. Note that the minimum temperature has changed from about 210 C to about 203 C. Since the surrounding air temperature is low (30 C) there is a corresponding reduction of total convection heat loss to 55. 48 W. Of course, the essential boundary condition reactions resultant is equal and opposite to that value. One might find these relatively small changes in temperatures and heat flows to be enough to cease refining the solution.

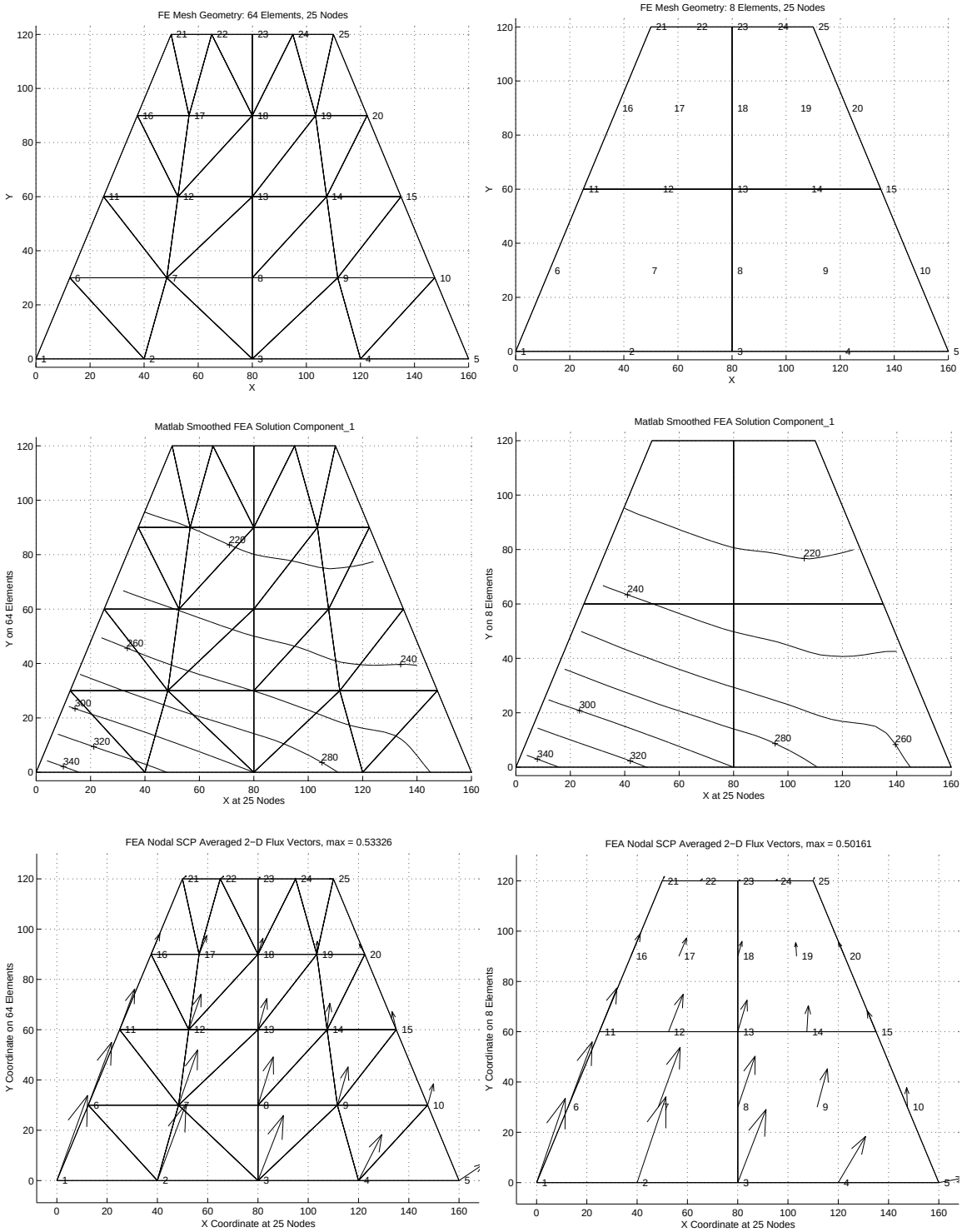


Figure 11.32 *Mesh, temperature, and smoothed flux vectors for the T3 and Q9 fin*

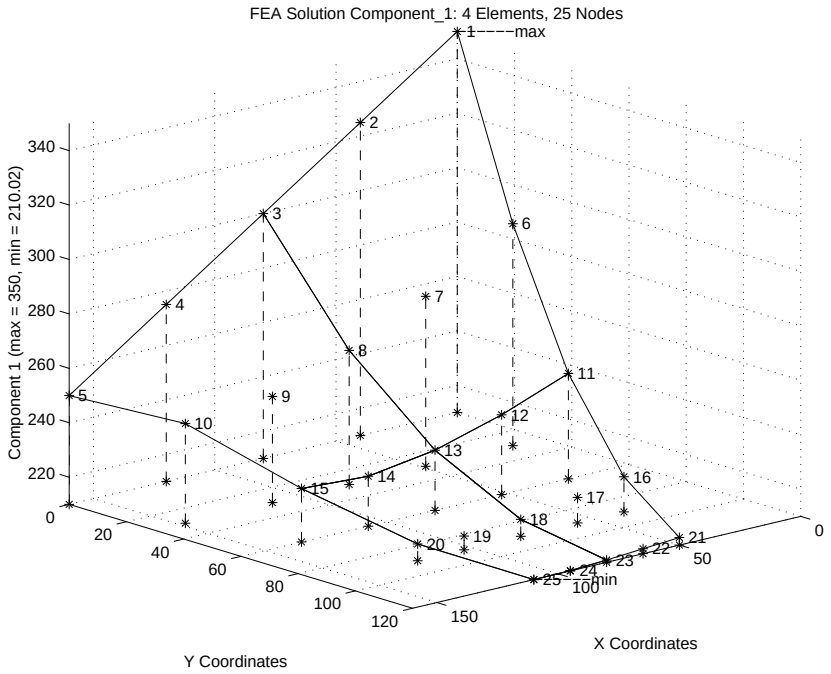


Figure 11.33 *Carpet-graph of fin temperatures, for Q9 elements*

```
title "Fin face convection, Allaire, Four Q9 elements"            ! 1
example        209 ! Application source code library number       ! 2
elems            4 ! Number of elements in the system             ! 3
nodes           25 ! Number of nodes in the mesh                  ! 4
convect_coef 2.e-5 ! Convection coefficient on mixed segments     ! 5
el_segment       9 ! Maximum nodes on element boundary segment    ! 6
mixed_segs       4 ! Number of mixed boundary condition segments ! 7
type_gauss    9 16 ! Number of Gauss points in each element type ! 8
type_nodes    9  9 ! Number of nodes on each element type        ! 9
type_shape    3  3 ! Shape code of each element type             !10
. . .                                                           !11
Here h is doubled for two face convection on 4 segment.         !12
    1    1    0.0000    0.0000 ! node, bc-flag,x, y             !13
    2    1   40.0000    0.0000                                  !14
. . .                                                           !15
   24    0   95.0000  120.0000                                  !16
   25    0  110.0000  120.0000 ! last node                       !17
1 1  1  3 13 11  2  8 12  6  7 ! elem, type, 9 nodes. conducting !18
2 1  3  5 15 13  4 10 14  8  9 ! conducting                      !19
3 1 11 13 23 21 12 18 22 16 17 ! conducting                      !20
4 1 13 15 25 23 14 20 24 18 19 ! conducting                      !21
1 2  1  3 13 11  2  8 12  6  7 ! convecting                      !22
2 2  3  5 15 13  4 10 14  8  9 ! convecting                      !23
3 2 11 13 23 21 12 18 22 16 17 ! convecting                      !24
4 2 13 15 25 23 14 20 24 18 19 ! convecting                      !25
  1  1  3.50000E+02   ! bc along wall                           !26
  2  1  3.25000E+02                                            !27
  3  1  3.00000E+02                                            !28
  4  1  2.75000E+02                                            !29
  5  1  2.50000E+02                                            !30
1  0.2 0.2 0.0 0.0 1.25 ! el, kx, ky, kxy, Q, thick            !31
```

Figure 11.34 *Data changes for the Q9 model*

```
TITLE: "Fin face convection, Allaire, Four Q9 elements"                    ! 1
. . .                                                                      ! 2
*** MIXED BOUNDARY CONDITION SEGMENTS ***                                  ! 3
 NOTE: CONSTANT CONVECTION ON ALL MIXED SEGMENTS USING                     ! 4
 CONVECTION COEFFICIENT =    2.0000000000000002E-05                        ! 5
 CONVECTION TEMPERATURE =   30.0000000000000000                            ! 6
SEGMENT, TYPE, 9 NODES ON THE SEGMENT                                      ! 7
      1     2   1    3   13   11    2    8   12    6    7                 ! 8
      2     2   3    5   15   13    4   10   14    8    9                 ! 9
      3     2  11   13   23   21   12   18   22   16   17                 !10
      4     2  13   15   25   23   14   20   24   18   19                 !11
. . .                                                                      !12
 *** SYSTEM GEOMETRIC PROPERTIES ***                                       !13
VOLUME   =  1.32000E+04                                                    !14
CENTROID =  8.00000E+01  5.09091E+01                                       !15
. . .                                                                      !16
*** REACTION RECOVERY ***                                                  !17
NODE, PARAMETER,     REACTION, EQUATION                                    !18
   1,  DOF_1,       6.4546E+00     1                                       !19
   2,  DOF_1,       2.5437E+01     2                                       !20
   3,  DOF_1,       1.1474E+01     3                                       !21
   4,  DOF_1,       1.4669E+01     4                                       !22
   5,  DOF_1,      -4.2698E-01     5                                       !23
REACTION RESULTANTS                                                        !24
PARAMETER,    SUM          POSITIVE     NEGATIVE                          !25
 DOF_1,      5.7608E+01   5.8035E+01  -4.2698E-01                          !26
                                                                           !27
*** EXTREME VALUES OF THE NODAL PARAMETERS ***                             !28
PARAMETER         MAXIMUM,   NODE      MINIMUM,   NODE                     !29
 DOF_1,        3.5000E+02,      1   2.1002E+02,     25                     !30
. . .                                                                      !31
 *** SUPER_CONVERGENT AVERAGED NODAL FLUXES ***                            !32
NODE,   X-Coord,    Y-Coord,    FLUX_1,     FLUX_2,                       !33
   1  0.0000E+00  0.0000E+00  1.5958E-01  4.6654E-01                       !34
   2  4.0000E+01  0.0000E+00  1.5970E-01  4.7551E-01                       !35
   3  8.0000E+01  0.0000E+00  1.5709E-01  4.0524E-01                       !36
   4  1.2000E+02  0.0000E+00  1.5177E-01  2.5573E-01                       !37
   5  1.6000E+02  0.0000E+00  1.4372E-01  2.6974E-02                       !38
   6  1.2500E+01  3.0000E+01  1.3393E-01  3.6177E-01                       !39
   7  4.8396E+01  3.0000E+01  1.2167E-01  3.3147E-01                       !40
   8  8.0000E+01  3.0000E+01  9.6232E-02  2.8267E-01                       !41
. . .                                                                      !42
  19  1.0340E+02  9.0000E+01 -4.2837E-03  7.5790E-02                       !43
  20  1.2250E+02  9.0000E+01 -3.3886E-02  7.8877E-02                       !44
  21  5.0000E+01  1.2000E+02  1.2224E-02  2.4631E-02                       !45
  22  6.5000E+01  1.2000E+02  1.3099E-02  1.1080E-02                       !46
  23  8.0000E+01  1.2000E+02  1.2498E-02  5.4736E-03                       !47
  24  9.5000E+01  1.2000E+02  1.0419E-02  7.8118E-03                       !48
  25  1.1000E+02  1.2000E+02  6.8624E-03  1.8095E-02                       !49
                                                                           !50
*** CONVECTION HEAT LOSS ***                                               !51
ELEMENT    HEAT_LOST                                                       !52
     1    2.02094E+01                                                     !53
     2    1.79895E+01                                                     !54
     3    9.83119E+00                                                     !55
     4    9.57818E+00                                                     !56
 TOTAL =   57.6082     <=== note <===                                      !57
```

Figure 11.35 *Selected cubic quadrilateral (Q9) fin results*

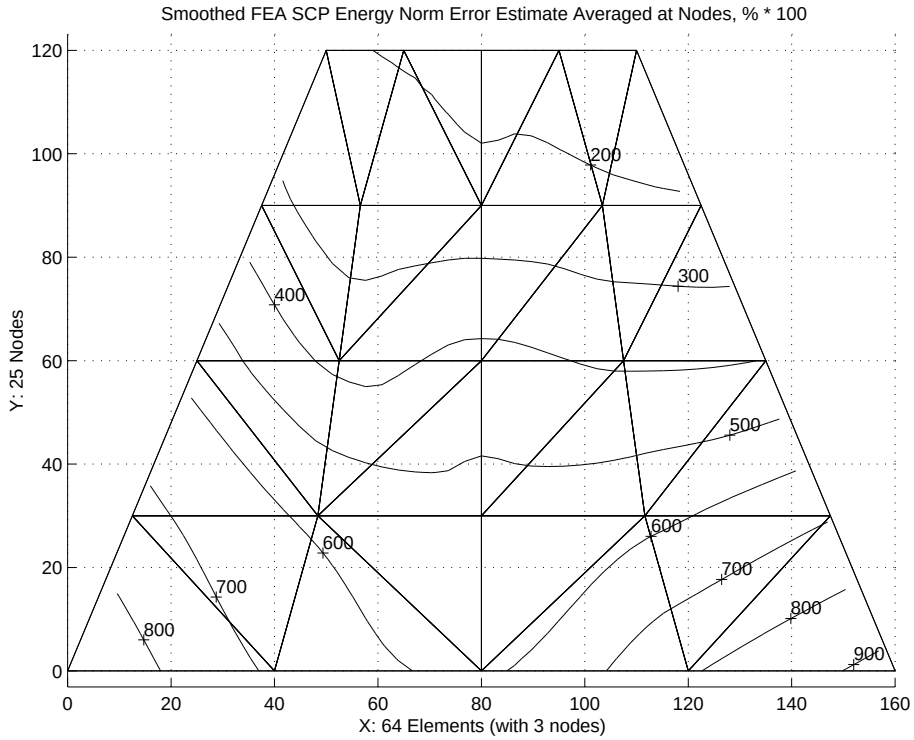


Figure 11.36 *Error norm levels for the 32 T3 model (max 9 percent)*

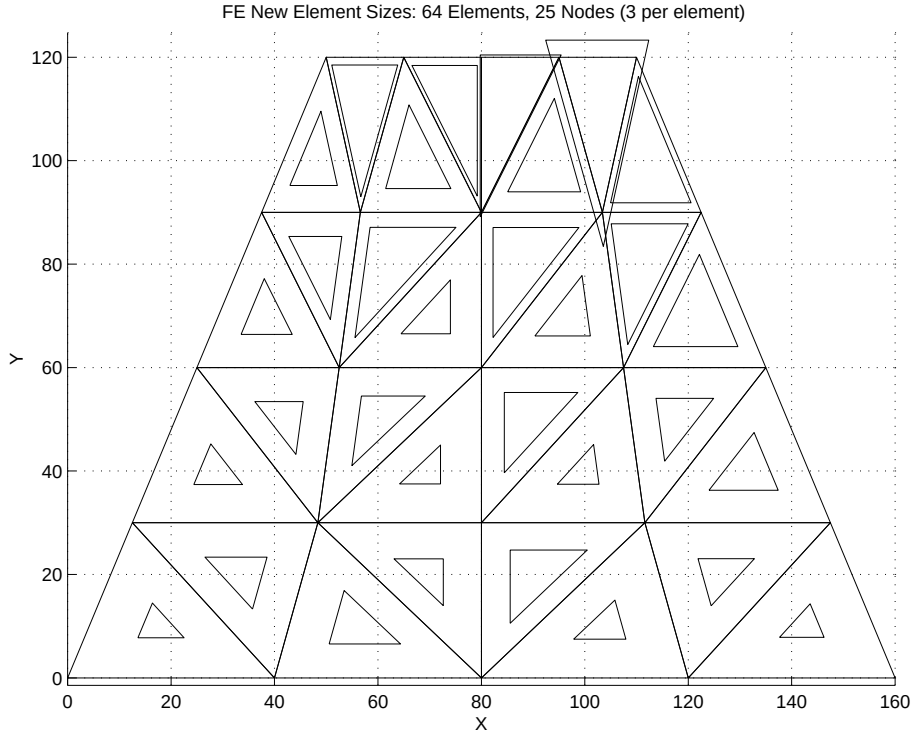


Figure 11.37 *First computed size changes for 32 T3 model*

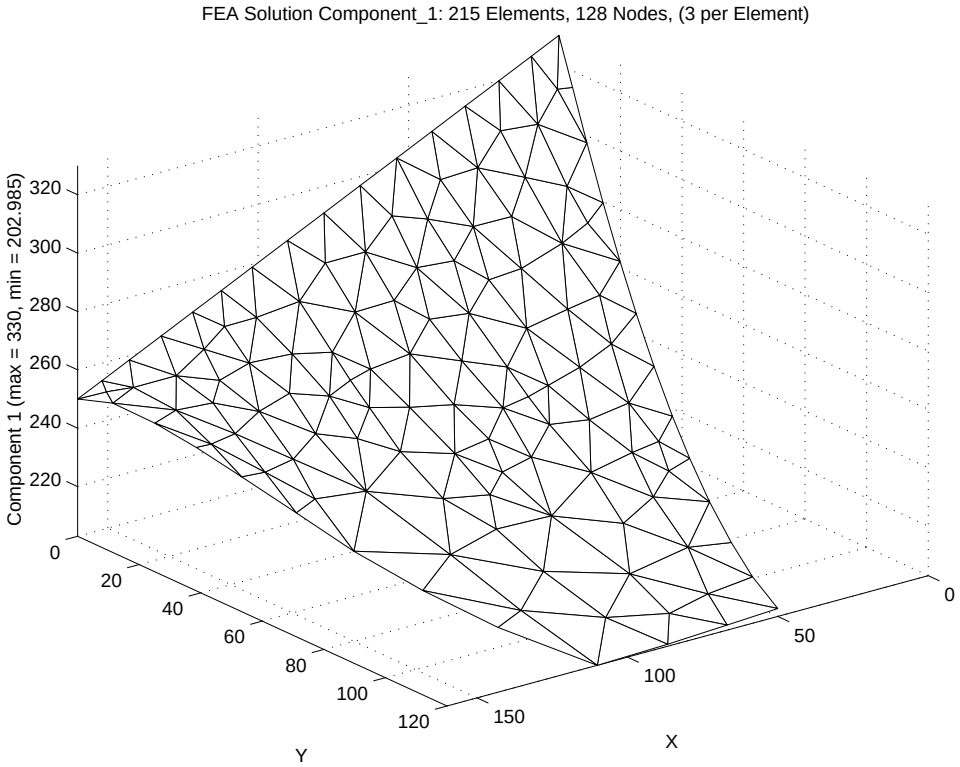


Figure 11.38 *Temperature surface for revised T3 model*

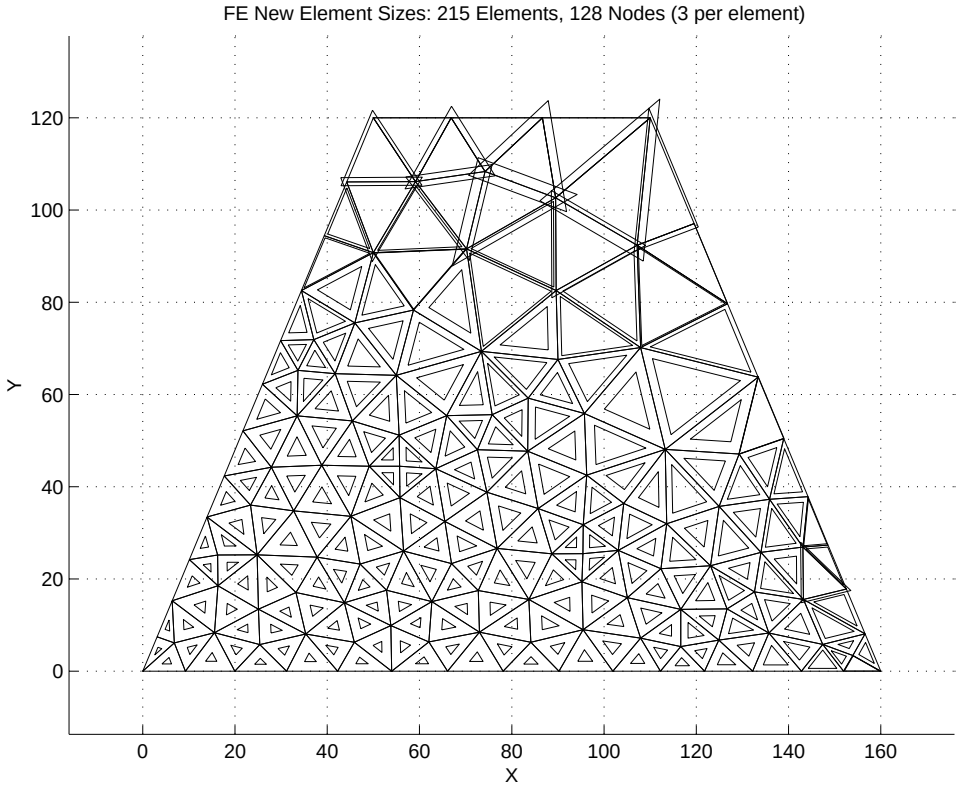


Figure 11.39 *Second computed size changes for T3 model*

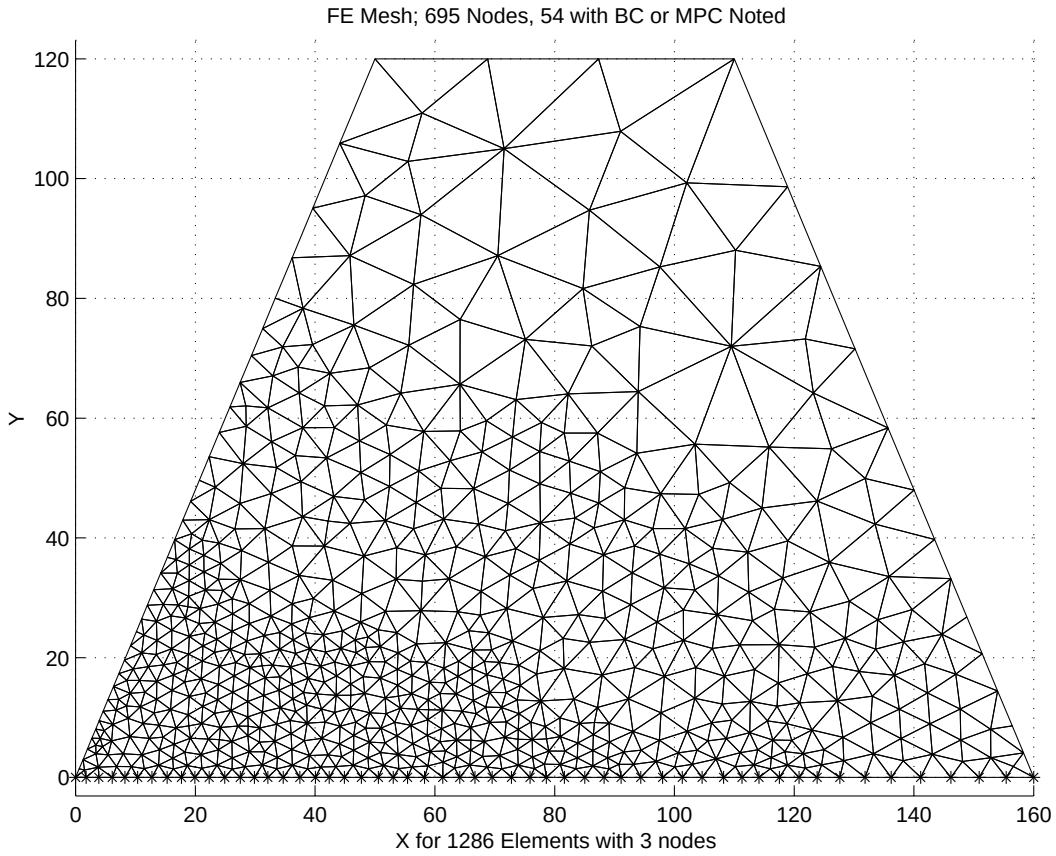


Figure 11.40 *Final mesh for T3 element model*

However, that mesh still has more than a 2.2 percent error level so another mesh size adjustment is computed. It is shown in Fig. 11.39 where again regions of high error project the need for elements much smaller than the current mesh and some elements near the free edge might be enlarged. The next stage of mesh generation creates a model with 695 nodes and 1286 T3 elements as shown in Fig. 11.40. One continues in this way until an acceptable error estimate is obtained.

11.6 Bilinear rectangles *

When quadrilateral elements have a parallelogram shape in physical space, their Jacobian is constant. If it takes the form of a rectangle with sides parallel to the global axes, then the Jacobian matrix is a constant diagonal matrix that allows the easy analytic evaluation of the element matrices. Consider a Q4 element mapped into a rectangular element that is parallel to the global axes so that the lengths parallel to the x and y axes are L_x and $L_{y,}$ respectively. This mapping gives a constant Jacobian:

$$
\begin{aligned}
x &= \bar{x} + a\,L_x/2\,, \qquad & \frac{\partial x}{\partial a} &= \frac{L_x}{2}\,, \qquad & \frac{\partial x}{\partial b} &\equiv 0\,,\\
y &= \bar{y} + b\,L_y/2\,, & \frac{\partial y}{\partial a} &= 0\,, & \frac{\partial y}{\partial b} &= \frac{L_y}{2}\,,
\end{aligned}
$$

$$
\mathbf{J} = \tfrac{1}{2}\begin{bmatrix} L_x & 0 \\ 0 & L_y \end{bmatrix}, \qquad |J| = \frac{L_x\,L_y}{4} = \frac{A^e}{4}\,.
$$

By inspection of the linear geometry mappings (or from the inverse Jacobian) we see $\partial/\partial x = \partial/\partial a\ \partial a/\partial x = 2/L_x\ \partial/\partial a,$ and likewise $\partial/\partial y = 2/L_y\ \partial/\partial b$ so that the typical term in the condition matrix due to k_x is

$$
\mathbf{S}_{x_{i,\,j}} = \frac{4}{L_x^2}\int\limits_{\Omega^e} k_x^e\ H_{i,\,a}\ H_{j,\,a}\ d\Omega, \tag{11.21}
$$

but $d\,\Omega = |J|\ d\,\square$ so that if k_x is constant

$$
\mathbf{S}_{x_{i,\,j}} = \frac{k_x^e\ L_y^e}{L_x^e}\int\limits_{\square} H_{i,\,a}\ H_{j,\,a}\ d\square.
$$

The interpolation functions are $H_j = (\,1 + a_j\,a\,)\,(\,1 + b_j\,b\,)/4$, so that a typical local derivative is $H_{j,\,a} = a_j\,(\,1 + b_j\,b\,)/4$ and the integrand becomes

$$
H_{i,\,a}\ H_{j,\,a} = a_i\ a_j\Big[\,1 + (\,b_i + b_j\,)\,b + b_i\ b_j\ b^2\,\Big]/16. \tag{11.22}
$$

Invoking numerical integration in $\square$ gives

$$
\mathbf{S}_{x_{i,\,j}} = \frac{a_i\ a_j\ k_x^e\ L_y^e}{16\ L_x^e}\ \sum_{q=1}^{Q}\Big[\,1 + (\,b_i + b_j\,)\,b_q + b_i\,b_j\,b_q^2\,\Big]w_q.
$$

Since this expression is quadratic in b, we need to pick only two points in the b direction. Likewise, for similar terms in the $\mathbf{S}_y$ matrix, we need two points in the a direction. Using the $Q = 4$ rule, we note that $w_q = 1$ and is constant, and that two $b_q = 1/\sqrt{3},$ while two are $-1/\sqrt{3}$. Thus, the linear terms cancel so that

$$
\mathbf{S}_{x_{i,\,j}} = \frac{a_i\,a_j\,k_x^e\,L_y^e}{12\,L_x^e}\Big[\,3 + b_i\,b_j\,\Big]. \tag{11.23}
$$

For the chosen local numbering, we have

j	a_j	b_j
1	−1	−1
2	1	−1
3	1	1
4	−1	1

so that the conduction matrix contributions are

$$\mathbf{S}_x = \frac{k_x^e\, L_y^e}{6\, L_x^e}\begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} \tag{11.24}$$

which agrees with the exact integration. Likewise, for a constant k_y^e:

$$\mathbf{S}_y = \frac{k_y^e\, L_x^e}{6\, L_y^e}\begin{bmatrix} 2 & 1 & -1 & -2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ -2 & -1 & 1 & 2 \end{bmatrix}. \tag{11.25}$$

Note that the sum of the terms in each row and each column is zero. This is typical for Lagrangian interpolation and serves as a useful visual check when doing hand calculations. The typical convention square (capacity) matrix term is

$$\begin{aligned} \mathbf{M}_{i,\,j} &= \int_{\Omega^e} \zeta^e\, H_i\, H_j\, d\,\Omega \\ &= \frac{L_x^e\, L_y^e}{64} \int_{\square^e} \zeta^e (1 + a_i\, a)(1 + a_j\, a)(1 + b_i\, b)(1 + b_j\, b)\; d\,\square \end{aligned} \tag{11.26}$$

so for constant ζ^e per unit area

$$\mathbf{M}_{i,\,j} = \frac{\zeta^e\, L_x^e\, L_y^e}{64} \int_{\square} [\,(1 + (a_i + a_j)\, a + a_i\, a_j\, a^2)\,(1 + (b_i + b_j)\, b + b_i\, b_j\, b^2)\,]\, d\,\square.$$

Again the four point rule is valid, and since $a_q = \pm 1/\sqrt{3}$, the linear terms cancel. Since $w_q \equiv 1$, we get

$$\mathbf{M}_{i,\,j} = \frac{\zeta^e\, L_x^e\, L_y^e}{64} [\, 4(1 + a_i\, a_j/3)\,(1 + b_j\, b_i/3)\,]$$

or

$$\mathbf{M} = \frac{\zeta^e\, L_x^e\, L_y^e}{36}\begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix}. \tag{11.27}$$

In this last matrix the sum of all the terms in the square brackets is 36, which assures that the total 'mass', $\zeta^e\, L_x^e\, L_y^e$, is properly accounted for. This is true for Lagrangian interpolation but not for Hermite elements. The edge boundary matrices are the same as for the linear triangle, since both are linear along their edges.

11.7 General 2-d elements

If we select a higher order element such as the isoparametric quadrilateral then some of the above integrations are more difficult to evaluate. In the notation of Chapter 9, Eq. 9.32 with $\mathbf{d}_x = \partial\mathbf{H}/\partial x$, etc. the conduction contribution becomes

$$\mathbf{S}^e \;=\; \int_{A^e} \left(\; k_x^e \mathbf{d}_x^{e^T} \mathbf{d}_x^e + k_y^e \mathbf{d}_y^{e^T} \mathbf{d}_y^e \; \right) t^e \; dA \;=\; \int_{A^e} \mathbf{B}^{e^T} \; \mathbf{E}^e \; \mathbf{B}^e \; t^e \; dA. \tag{11.28}$$

If we allow for general quadrilaterals and/or curved sides then we will need numerical integration. Thus, we write

$$\mathbf{S}^e = \sum_{i=1}^{n_q} W_i \; \left| J^{e_i} \right| (\; k_{x_i}^e \; \mathbf{d}_{x_i}^{e^T} \; \mathbf{d}_{x_i}^e + k_{y_i}^e \; \mathbf{d}_{y_i}^{e^T} \; \mathbf{d}_{y_i}^e \;) \; t_i^e. \tag{11.29}$$

Typically, we would place the conductivities, k_x^e etc., in the constitutive matrix, $\mathbf{E}^e$, especially if they are fully anisotropic. In that case the integration is more clearly cast into a form involving matrix products:

$$\mathbf{S}^e = \sum_{i=1}^{n_q} W_i \; \left| J^{e_i} \right| (\mathbf{B}^{e^T} \; \mathbf{E}^e \; \mathbf{B}^e) \; t_i^e. \tag{11.30}$$

Similar expressions are available for the mass (or capacity) matrix and source vectors.

11.8 Numerically integrated arrays

The processes given above can be generalized to handle any curvilinear 1-, 2-, or 3-dimensional element by using general interpolation libraries, quadrature rule libraries, and numerical integration. First we will consider the conduction matrix and volumetric source vector as shown in Fig. 11.41 which accepts any of the elements in the *MODEL* library. Next, we will sometimes have a known flux acting across the normal to a given segment of the boundary. This is usually called a Neumann condition. It degenerates to a natural boundary condition when the flux is zero because the integral of zero is zero and there is no need to invoke the calculation. Since the flux can act normal to any segment of the mesh the general differential geometry considerations, of Sec. 8.6, may come into play. That requires a more general numerical integration process because the geometric mappings are not always one to one. That is, the normal flux can occur on a doubly curved 3-dimensional surface, a flat surface in the analysis plane, a curved edge, or simply a point. Thus, a full and more expensive use of differential geometry replaces a simple (one to one) Jacobian calculation shown before.

In this case only a column vector (that may degenerate to a scalar) must be computed as given in Fig. 11.42. There, in line 26, the general parametric measure is computed via function *PARM_GEOM_METRIC* (in Fig. 9.12) instead of the determinant of the usual Jacobian matrix shown in previous examples. Of course, they give the same results when the space mapping is one to one (straight line to straight line, or flat area to flat area). Finally, we will consider a pair of mixed or Robin type of matrices. Here they will represent heat convection data. This similar coding always also forms a square matrix and a column vector (which may degenerate to scalars) as listed in Fig. 11.43. It includes logic to identify either a globally constant set of convection data, or a different set of convection constants on each segment. One could also allow the convection data to be input as nodal properties if they vary over the segments but those details are omitted here to save space. Again *PARM_GEOM_METRIC*, line 42, allows for a general non-flat convection surface, or convection along a non-straight space curve.

```
! ..............................................................    ! 1
!    ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***    ! 2
! ..............................................................    ! 3
!    ANISOTROPIC POISSON EQUATION IN 1-, 2-, or 3-DIMENSIONS        ! 4
!            VIA NUMERICALLY INTEGRATED ELEMENTS                    ! 5
!       (K_ij * U,i),j + Q = 0;  1 <= (i,j) <= N_SPACE              ! 6
 REAL(DP) :: CONST, DET, THICK         ! integration                 ! 7
 REAL(DP) :: SOURCE                    ! data                        ! 8
 INTEGER  :: IP                        ! loop counter                ! 9
!           (Stored as application source example 209.)             !10
! Convection added in MIXED_SQ_MATRIX, known flux in SEG_COL_MATRIX !11
                                                                    !12
! Conduction properties assumed, (order in GET_REAL_LP (n)):        !13
! 1-D problem, K_xx, Q, Thickness                                   !14
! 2-D problem, K_xx, K_yy, K_xy, Q, Thickness                       !15
! 3-D problem, K_xx, K_yy, K_zz, K_xy, K_xz, K_yz, Q                !16
  CALL POISSON_ANISOTROPIC_E_MATRIX (E) ! for 1-, 2-, or 3-D        !17
                                                                    !18
!   CHECK FOR KEYWORD GLOBAL CONSTANTS: scalar_source & area_thick  !19
  SOURCE = 0 ; IF ( SCALAR_SOURCE /= 0.d0 ) SOURCE = SCALAR_SOURCE  !20
  THICK  = 1 ; IF ( AREA_THICK    /= 1.d0 ) THICK  = AREA_THICK     !21
  IF ( EL_REAL > 0 ) THEN     ! Get local element constant values,  !22
    SELECT CASE (N_SPACE)     ! for source or thickness             !23
      CASE (1) ; IF ( EL_REAL > 1 ) SOURCE = GET_REAL_LP (2)        !24
                 IF ( EL_REAL > 2 ) THICK  = GET_REAL_LP (3)        !25
      CASE (2) ; IF ( EL_REAL > 3 ) SOURCE = GET_REAL_LP (4)        !26
                 IF ( EL_REAL > 4 ) THICK  = GET_REAL_LP (5)        !27
      CASE (3) ; IF ( EL_REAL > 6 ) SOURCE = GET_REAL_LP (7)        !28
    END SELECT ! for properties options                             !29
  END IF ! element data provided                                    !30
                                                                    !31
  CALL STORE_FLUX_POINT_COUNT ! Save LT_QP, for post-processing     !32
                                                                    !33
  DO IP = 1, LT_QP          ! NUMERICAL INTEGRATION LOOP            !34
    H   = GET_H_AT_QP (IP)    ! EVALUATE INTERPOLATION FUNCTIONS    !35
    XYZ = MATMUL (H, COORD)   ! FIND GLOBAL PT, ISOPARAMETRIC       !36
    DLH = GET_DLH_AT_QP (IP)  ! FIND LOCAL DERIVATIVES, dH / dr     !37
    AJ  = MATMUL (DLH, COORD) ! FIND JACOBIAN AT THE PT             !38
    CALL INVERT_JACOBIAN (AJ, AJ_INV, DET, N_SPACE)       ! inverse !39
    IF ( AXISYMMETRIC ) THICK = TWO_PI * XYZ (1) ! via axisymmetric !40
    CONST  = DET * WT(IP) * THICK                                   !41
                                                                    !42
    DGH = MATMUL (AJ_INV, DLH)          ! Physical gradient, dH / dx !43
    B   = COPY_DGH_INTO_B_MATRIX (DGH) ! B = DGH                    !44
                                                                    !45
!   VARIABLE VOLUMETRIC SOURCE, via keyword use_exact_source        !46
!   Defaults to file my_exact_source_inc if no exact_case key       !47
    IF ( USE_EXACT_SOURCE )  CALL        & ! analytic Q             !48
         SELECT_EXACT_SOURCE (XYZ, SOURCE) ! via exact_case key     !49
    C = C + CONST * SOURCE * H          ! source resultant          !50
                                                                    !51
!     CONDUCTION SQUARE MATRIX (THICKNESS IN E)                     !52
    S = S + CONST * MATMUL ((MATMUL (TRANSPOSE (B), E)), B)         !53
                                                                    !54
!-->  SAVE COORDS, E AND DERIVATIVE MATRIX, FOR POST PROCESSING     !55
    CALL STORE_FLUX_POINT_DATA (XYZ, (E * THICK), B)                !56
  END DO                                                            !57
!   ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***        !58
```

Figure 11.41 *The generalized volumetric contributions*

```
! ..................................................                  ! 1
!   ***  SEG_COL_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***       ! 2
! ..................................................                  ! 3
! Given normal flux on an element face or edge:                       ! 4
! Standard form: -K_n * U,n = Q_NORMAL_SEG (in System_Constants)      ! 5
!          (Stored as application source example 209.)                ! 6
 INTEGER  :: IQ                     ! loops                          ! 7
 REAL(DP) :: CONST, DET, THICK      ! flux area                      ! 8
  ! Get normal flux from keyword, or segment property                ! 9
  IF ( SEG_REAL > 0 ) Q_NORMAL_SEG= GET_REAL_SP (1)                  !10
                                                                     !11
  THICK = 1   ! Default flux line segment real property # 2          !12
  IF ( SEG_REAL > 1 ) THICK = GET_REAL_SP (2)                        !13
                                                                     !14
  IF ( LT_N > 1 ) THEN ! Not a point value                           !15
                                                                     !16
    DO IQ = 1, LT_QP ! NUMERICAL INTEGRATION LOOP                    !17
                                                                     !18
      H = GET_H_AT_QP (IQ) ! BOUNDARY INTERPOLATION FUNCTIONS        !19
!       FIND GLOBAL COORD, XYZ = H*COORD (ISOPARAMETRIC)             !20
      XYZ = MATMUL (H, COORD)                                        !21
!        FIND LOCAL DERIVATIVES                                      !22
      DLH = GET_DLH_AT_QP (IQ)  ! dH / dr                            !23
                                                                     !24
!        FORM DETERMINATE OF GENERALIZED JACOBIAN                    !25
      DET   = PARM_GEOM_METRIC (DLH, COORD) ! dX / dr                !26
      IF ( AXISYMMETRIC ) THICK = TWO_PI * XYZ (1) ! keyword         !27
      CONST = DET * WT(IQ) * THICK                                   !28
                                                                     !29
!       GET NORMAL FLUX COMPONENT                                    !30
      IF ( USE_EXACT_FLUX ) CALL SELECT_EXACT_NORMAL_FLUX &          !31
           (XYZ, Q_NORMAL_SEG) ! via keyword use_exact_flux          !32
      C = C + Q_NORMAL_SEG * CONST * H    ! Source vector            !33
    END DO                                                           !34
                                                                     !35
  ELSE ! This is a point value                                       !36
                                                                     !37
    IF ( USE_EXACT_FLUX ) CALL SELECT_EXACT_NORMAL_FLUX &            !38
         (COORD (1, :), Q_NORMAL_SEG) ! via use_exact_flux           !39
    C (1) = Q_NORMAL_SEG                                             !40
                                                                     !41
  END IF ! boundary segment type                                     !42
! End application dependent flux 202.my_seg_col_inc_2               !43
```

Figure 11.42 *Computing a Neumann flux contribution*

If one wants to later recover the convection heat loss as a physical check or to better understand the problem then the necessary integral of $\mathbf{H}^b$ is saved, in line 58, for later use with the constant segment convection properties. Keyword *post_mixed* is used in the data to activate a file unit (*N_FILE*2) for these purposes. After a solution has been obtained that keyword also causes the convection heat loss calculations, in Fig. 11.44, to be carried out, as discussed in Eq. 7.37 and Section 11.5.2.

These numerically integrated forms for evaluating the matrices were applied to the previous linear triangle examples and gave exactly the same results as the explicit coded forms in Figs. 11.6-8. A one-point rule was used in most cases but the face convection square matrix required a three-point rule since it involved the product of two linear functions, which resulted in the integrand being a quadratic polynomial. The quadrature

based formulation allows for curved element boundaries or other reasonable varying Jacobians. They allow use of any point, 1-, 2-, or 3-dimensional element in the library.

11.9 Strong diagonal gradient SCP test case

This problem is defined in terms of the Poisson equation where the solution has been chosen in advance to give zero values on the boundary of a unit square and to exhibit a relatively sharp transition of gradients along a region near the diagonal of the square domain. This test case has been used by Oden 18 and Zienkiewicz and Zhu 23 for testing various error estimators. The exact solution is given by

$$u(x, y) = xy(1 - x)(1 - y)\, Tan^{-1}(\alpha(\xi - \xi_0))$$

where $\xi = (x + y)/\sqrt{2}$ with $\xi_0 = 0.8$ and $\alpha = 20$. The contours of the exact solution are plotted in Fig. 11.45, and the corresponding exact gradient contours are in Fig. 11.46. When it is substituted into Poisson's equation the algebraic definition of the source term per unit area, Q, is obtained and placed in routine to be used in the numerical integration of the element matrices. Since the source term is rather complicated this represents a case where one may want to have different subroutines for evaluating the element square and column matrices since the latter requires more quadrature points than the former. The initial mesh for this problem is a rather crude one consisting of 50 identical 6 noded (complete quadratic) elements. They are shown in Fig. 11.47 along with the flags on the boundary which indicate the nodes where the null essential boundary conditions were applied. A crude mesh is chosen so that the reader can see the differences in the exact and approximate solutions and verify that they are decreased as the model is later refined. Usually we do not know the exact result and must rely on the computed error measures when deciding to stop an analysis.

When the solution is computed with this mesh one gets the contour levels shown in Fig. 11.48 which are compared to corresponding exact levels in Fig. 11.49. The wiggles appearing in the contours provide an 'eyeball' check that indicates that the mesh is too crude in this region and will need refining if it is an important part of the solution domain. In this case the the region of wiggles is where the contours are the closest together which means the solution is rapidly changing and the mesh is indeed too coarse to be accepted. Of course, our error estimator will lead us to the same conclusion but it is still wise to employ a little common sense along the way.

Post-processing for the gradients at the quadrature points of the elements yields the distribution of flux vectors shown in Fig. 11.50. While we could compare contour values of the gradient components it makes more sense in this case to use the true shape two-dimensional flux vectors. Here the vectors represent the item obtained from Eq. 11.8 by using a standard post-processing technique once the $\mathbf{\Phi}^e$ of the element have been gathered from the system $\mathbf{\Phi}$ and multiplied by the matrices $\mathbf{E}^e\ \mathbf{B}^e$ at each quadrature point in the element. They are the data that are processed in the SCP method to obtain continuous nodal flux estimates.

In this case we have selected element based patches using all adjacent element neighbors, as shown in Fig. 11.2. The resulting SCP nodal flux vectors are shown in

```
!    ***  MIXED_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***   ! 1
! ..........................................................        ! 2
! Mixed or Robin boundary condition, Standard form:                 ! 3
!   K_n * U,n + ROBIN_1_SEG * U + ROBIN_2_SEG = 0                   ! 4
!          (Stored as application source example 209.)              ! 5
 INTEGER  :: IQ                ! Integration loop                   ! 6
 REAL(DP) :: CONST, DET, THICK ! face area                          ! 7
                                                                    ! 8
  IF ( N_FILE2 > 0 ) H_INTG = 0 ! via post_mixed to post-process    ! 9
 !  GET ROBIN TERMS, IF GLOBAL CONSTANTS                           !10
 !    Set in keywords robin_square and robin_column, or via        !11
 !    convect_coef, convect_temp for convection special case       !12
  THICK = 1.d0           ! if an edge                              !13
  IF ( CONVECTION ) THEN ! constant convection all segments        !14
    ROBIN_1_SEG =  CONVECT_COEF                                     !15
    ROBIN_2_SEG = -CONVECT_COEF * CONVECT_TEMP                      !16
    IF ( LT_PARM < 2 ) THICK =  CONVECT_THICK ! point or line       !17
  END IF ! Globally constant convection                             !18
                                                                    !19
 !  GET ROBIN TERMS, IF LOCAL SEGMENT CONSTANTS                     !20
  IF ( MIXED_REAL > 0 ) THEN ! are local data                       !21
    IF ( CONVECT_VARY ) THEN ! data are coeff, temperature, thick   !22
      ROBIN_1_SEG = GET_REAL_MX (1) ! convection coeff              !23
      ROBIN_2_SEG = 0.d0            ! default temperature effect    !24
      IF (MIXED_REAL > 1) ROBIN_2_SEG = -ROBIN_1_SEG   &            !25
                          * GET_REAL_MX (2) ! coeff*temp            !26
      IF (MIXED_REAL > 2 .AND. LT_PARM < 2) THICK = GET_REAL_MX (3) !27
    ELSE ! a non-convection Robin condition                         !28
      ROBIN_1_SEG = GET_REAL_MX (1) ; ROBIN_2_SEG = 0.d0            !29
      IF (MIXED_REAL > 1) ROBIN_2_SEG = GET_REAL_MX (2)             !30
      IF (MIXED_REAL > 2 .AND. LT_PARM < 2) THICK = GET_REAL_MX (3) !31
    END IF ! convection or general Robin data                       !32
  END IF ! local data                                               !33
                                                                    !34
  IF ( LT_N > 1 ) THEN ! Not a point, must integrate line or surf   !35
    DO IQ = 1, LT_QP           ! NUMERICAL INTEGRATION LOOP         !36
      H   = GET_H_AT_QP (IQ)   ! BOUNDARY INTERPOLATION FUNCTIONS   !37
      XYZ = MATMUL (H, COORD)  ! FIND GLOBAL COORD, (ISOPARAMETRIC) !38
      DLH = GET_DLH_AT_QP (IQ) ! FIND LOCAL DERIVATIVES, dH / dr    !39
                                                                    !40
!        FORM DETERMINATE OF GENERALIZED JACOBIAN, Fig 9.12         !41
      DET   = PARM_GEOM_METRIC (DLH, COORD)        ! dX / dr        !42
      IF ( AXISYMMETRIC ) THICK = TWO_PI * XYZ (1) ! axisymmetric   !43
      CONST = DET * WT(IQ) * THICK                                  !44
      IF ( N_FILE2 > 0 ) H_INTG = H_INTG + H * DET * WT(IQ)         !45
                                                                    !46
!       FORM MIXED ARRAYS                                          !47
      S = S + ROBIN_1_SEG * CONST * OUTER_PRODUCT (H, H) ! Sq       !48
      C = C - ROBIN_2_SEG * CONST * H                    ! Source   !49
    END DO                                                          !50
                                                                    !51
  ELSE ! This is a point value                                      !52
    IF ( AXISYMMETRIC ) THICK = TWO_PI * COORD (1, 1)               !53
    S (1, 1) =  ROBIN_1_SEG * THICK ; C (1) = -ROBIN_2_SEG * THICK  !54
    IF ( N_FILE2 > 0 ) H_INTG = THICK ! actually area at pt, input  !55
                                                                    !56
  END IF ! boundary segment type                                    !57
  IF ( N_FILE2 > 0 ) WRITE (N_FILE2) H_INTG ! via post_mixed        !58
! End mixed condition BC my_mixed_sq_inc                            !59
```

Figure 11.43 *General Robin or convection contributions*

```
! .....................................................................     ! 1
!  *** POST_PROCESS_MIXED PROBLEM DEPENDENT STATEMENTS FOLLOW ***          ! 2
! .....................................................................     ! 3
!           (Stored as application source example 209.)                    ! 4
! Global CONVECT_COEF  set by keyword convect_coeff is available           ! 5
! Global CONVECT_TEMP  set by keyword convect_temp  is available           ! 6
! H_INTG (LT_N) Integral of interpolation functions, H, available          ! 7
                                                                           ! 8
!               convection heat loss recovery                              ! 9
 REAL(DP) :: THICK                        ! line width                    !10
 REAL(DP), SAVE :: Q_LOSS, TOTAL = 0.d0 ! Face and total heat loss        !11
 LOGICAL,  SAVE :: FIRST = .TRUE.         ! printing                      !12
                                                                          !13
  IF ( FIRST ) THEN                       ! first call                    !14
    FIRST = .FALSE. ; WRITE (6, 5)        ! print headings                !15
    5 FORMAT ('*** CONVECTION HEAT LOSS ***', /, &                        !16
    &         'ELEMENT    HEAT_LOST')                                     !17
  END IF ! first call                                                     !18
                                                                          !19
  IF ( N_FILE2 > 0 ) THEN ! H already integrated in MIXED SQ.             !20
    READ (N_FILE2) H_INTG ! via keyword post_mixed                        !21
  ELSE                                                                    !22
    PRINT *,'WARNING: NEED post_mixed IN CONTROL'                         !23
    N_WARN = N_WARN + 1                                                   !24
    STOP 'POST_PROCESS_MIXED 208,209, or 302 DATA NOT SAVED'              !25
  END IF ! H_INTG                                                         !26
                                                                          !27
!  GET ROBIN TERMS, IF GLOBAL CONSTANTS                                   !28
!    Set in keywords robin_square and robin_column, or via                !29
!    convect_coef, convect_temp for convection special case               !30
  THICK = 1.d0           ! if an edge                                     !31
  IF ( CONVECTION ) THEN ! constant convection all segments               !32
    ! Then CONVECT_COEF, CONVECT_TEMP, CONVECT_THICK global               !33
    IF ( LT_PARM < 2 ) THICK = CONVECT_THICK ! point or line              !34
  END IF ! Globally constant convection                                   !35
                                                                          !36
 !  GET ROBIN TERMS, IF LOCAL SEGMENT CONSTANTS                           !37
  IF ( MIXED_REAL > 0 ) THEN ! are local data                             !38
    IF ( CONVECT_VARY ) THEN ! data: coeff, temperature, thick            !39
      CONVECT_COEF = GET_REAL_MX (1) ! convection coeff                   !40
      CONVECT_TEMP = 0.d0            ! default temperature                !41
      IF (MIXED_REAL > 1) CONVECT_TEMP = GET_REAL_MX (2) ! temp           !42
      IF (MIXED_REAL > 2 .AND. LT_PARM < 2) THICK = GET_REAL_MX (3)       !43
    ELSE ! a non-convection Robin condition                               !44
      PRINT *,'WARNING: NEED KEY convect_coef OR convect_vary '           !45
      N_WARN = N_WARN + 1                                                 !46
      STOP 'POST_PROCESS_MIXED 208,209, or 302 KEYWORD ERROR'             !47
    END IF                                                                !48
  END IF ! local data                                                     !49
                                                                          !50
!   HEAT LOST FROM THIS FACE: Integral over face of h * (T - T_inf)       !51
   D (1:LT_N) = D(1:LT_N) - CONVECT_TEMP  ! Temp difference at nodes      !52
   Q_LOSS = CONVECT_COEF  * DOT_PRODUCT (H_INTG, D)  ! Face loss          !53
   IF ( LT_PARM < 2 ) Q_LOSS = Q_LOSS * THICK        ! line or point      !54
   TOTAL  = TOTAL + Q_LOSS                           ! Running total      !55
   PRINT '(I6, ES15.5)', IE, Q_LOSS                  ! Each segment       !56
   IF ( IE == N_MIXED ) PRINT *, 'TOTAL = ', TOTAL   ! Last segment       !57
!  *** END POST_PROCESS_MIXED PROBLEM DEPENDENT STATEMENTS ***            !58
```

Figure 11.44 *General convection heat loss post-processing*

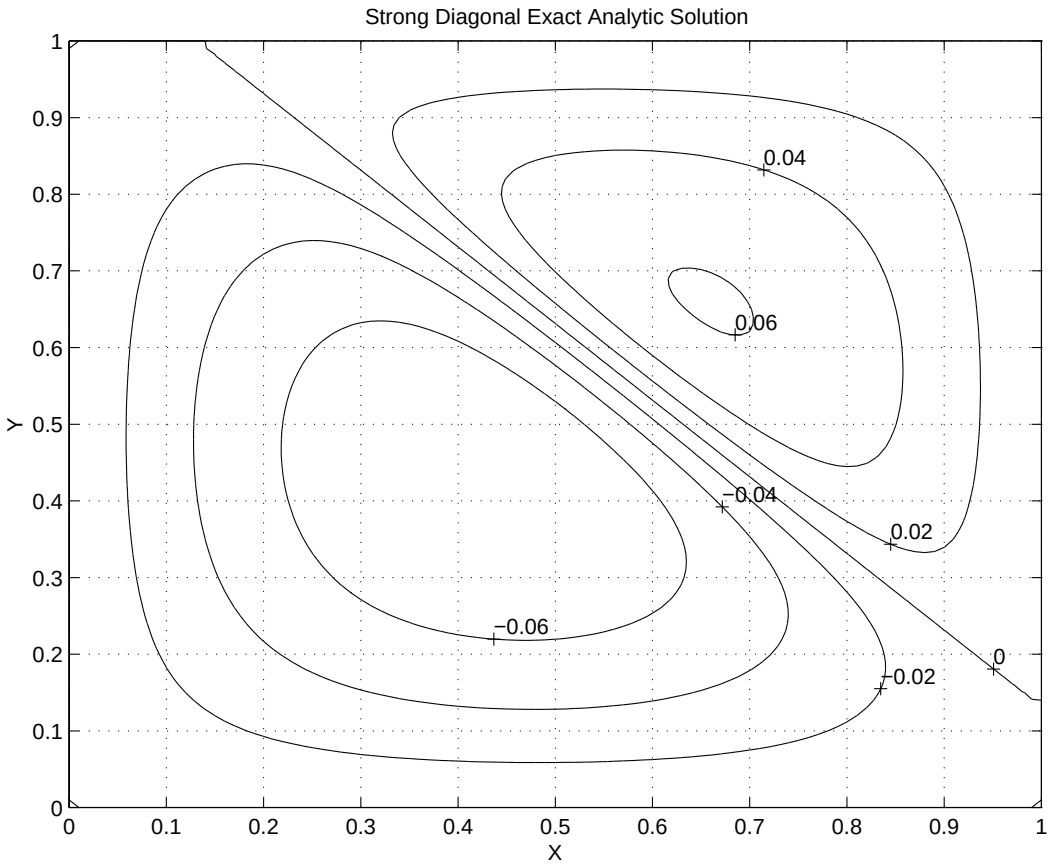


Figure 11.45 *Contours of the analytic solution*

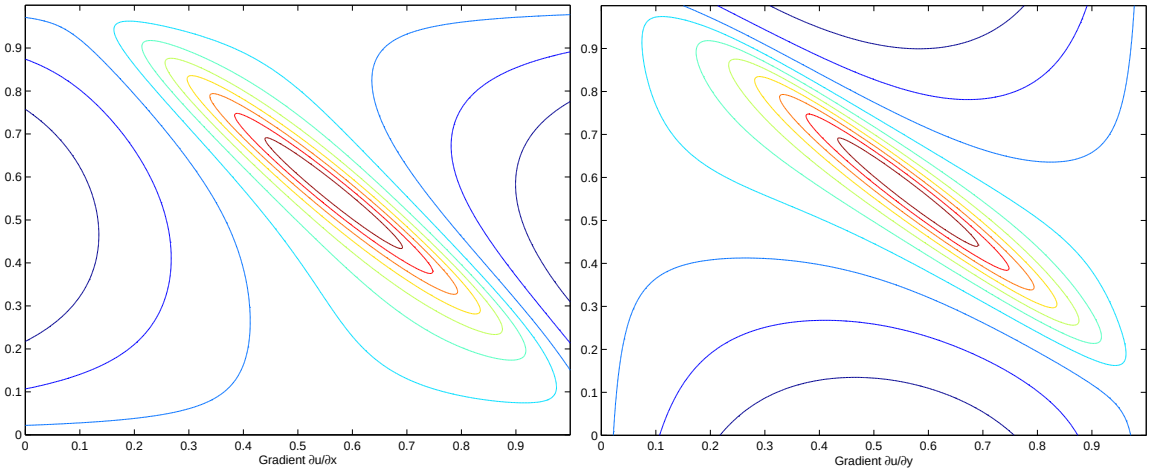


Figure 11.46 *Contours of the analytic gradient components*

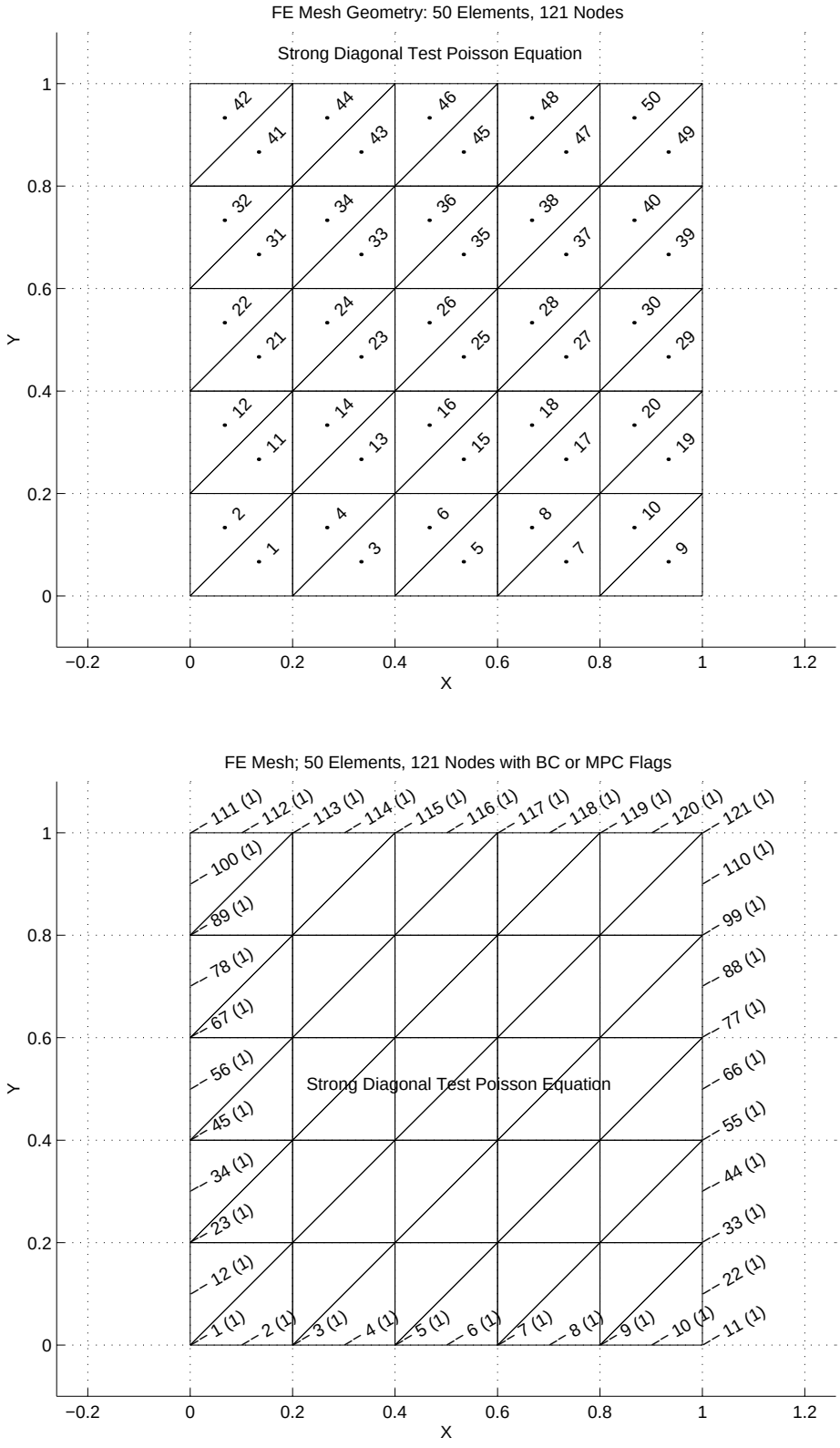


Figure 11.47 *Initial quadratic element mesh and Dirichlet nodes*

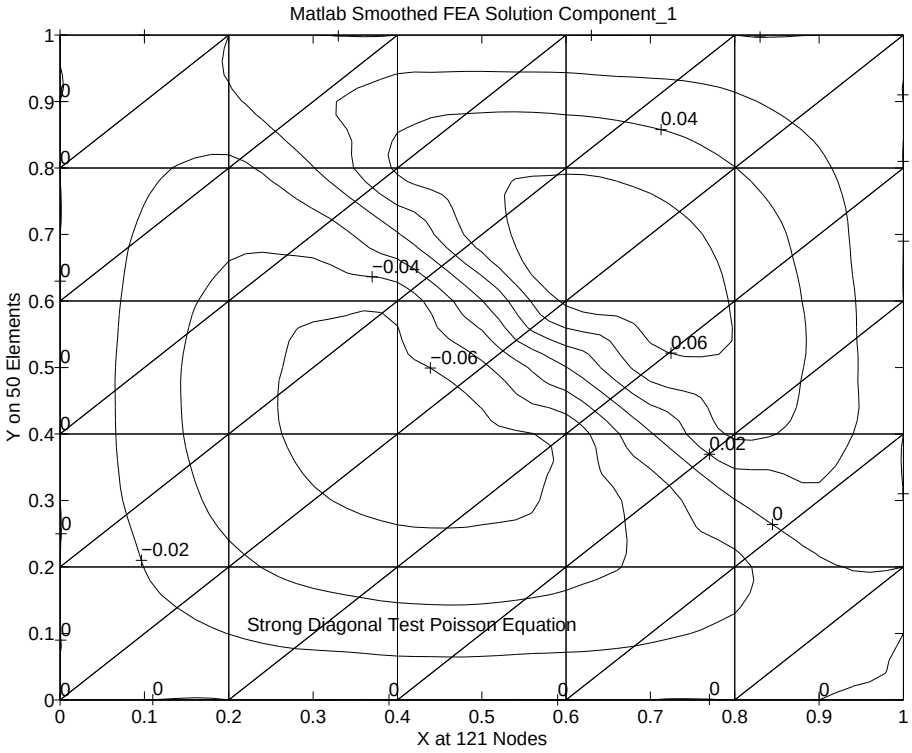


Figure 11.48 *Initial finite element solution contours*

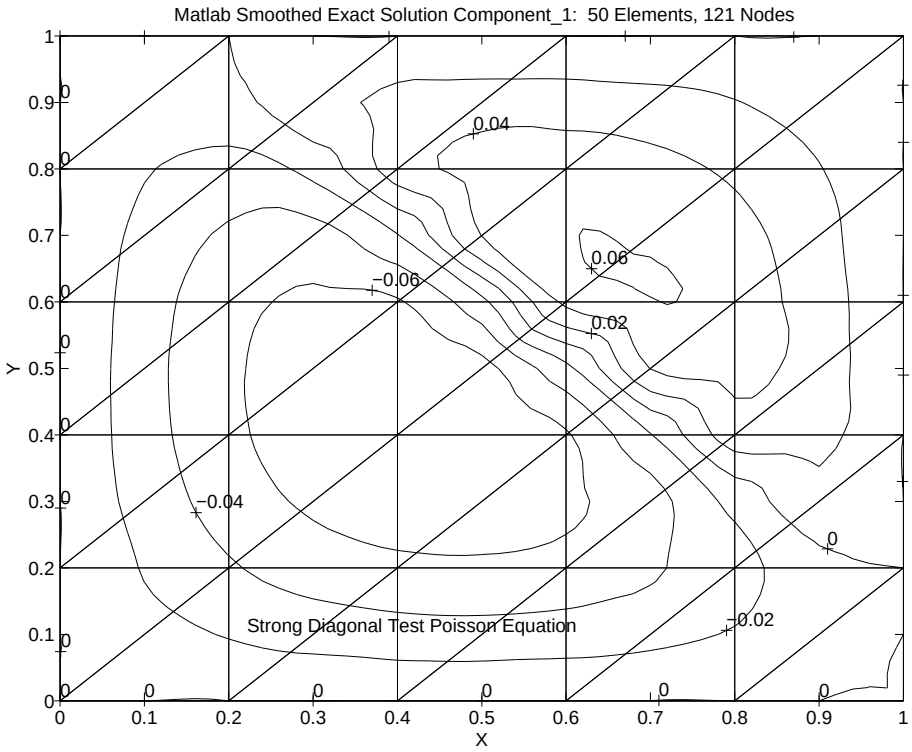


Figure 11.49 *Exact solution evaluated at the nodes (only)*

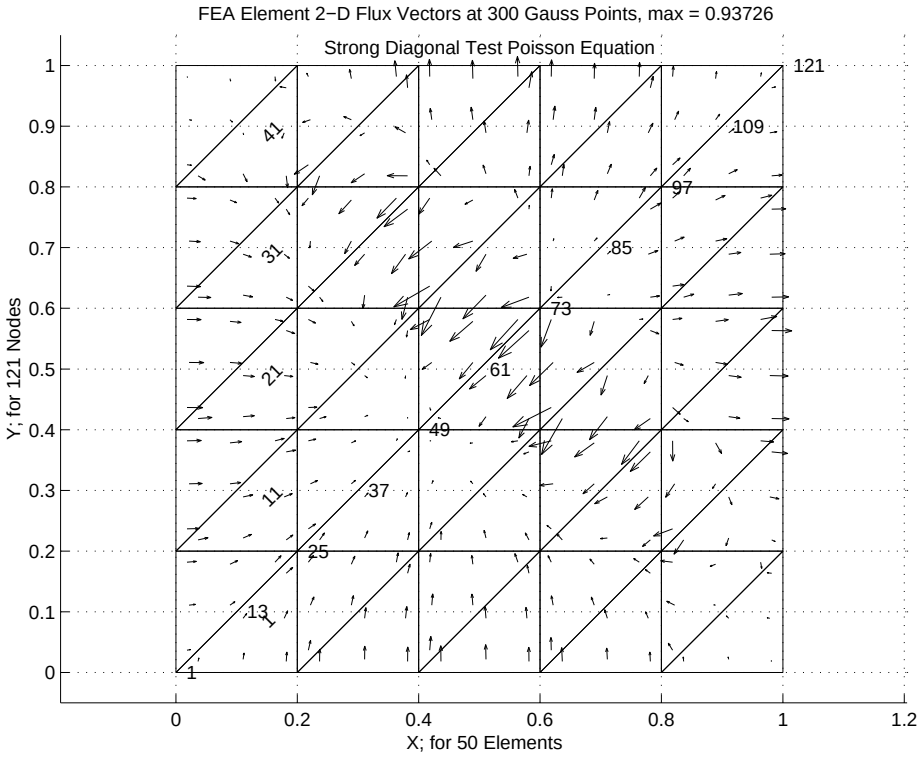


Figure 11.50 *Initial element quadrature point flux vectors*

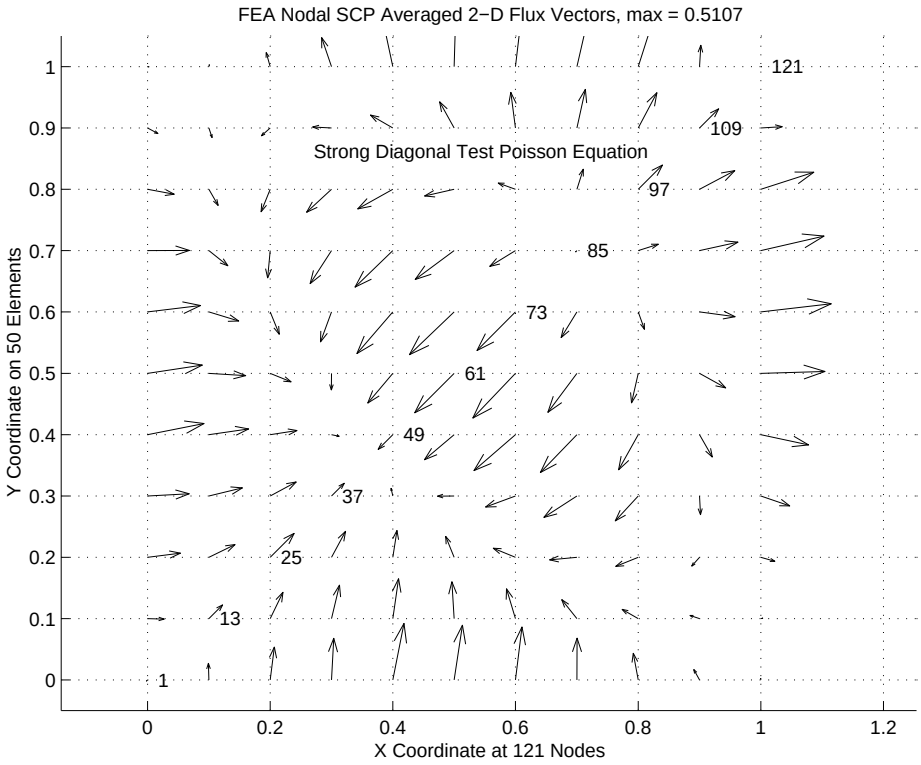


Figure 11.51 *Initial finite element SCP nodal flux vectors*

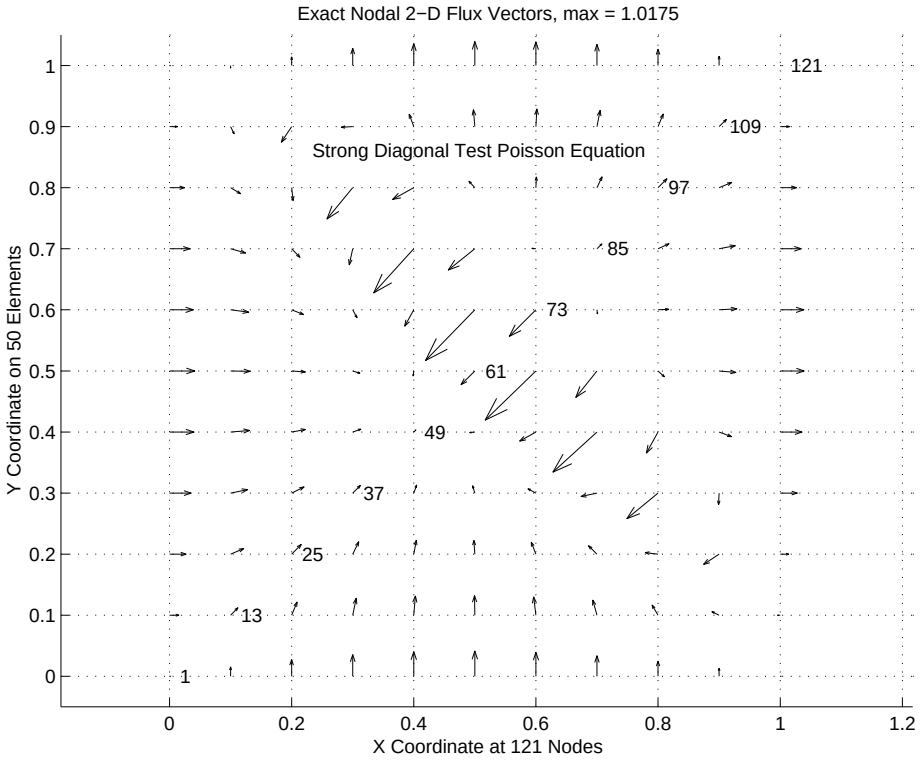


Figure 11.52 *Exact nodal flux vectors*

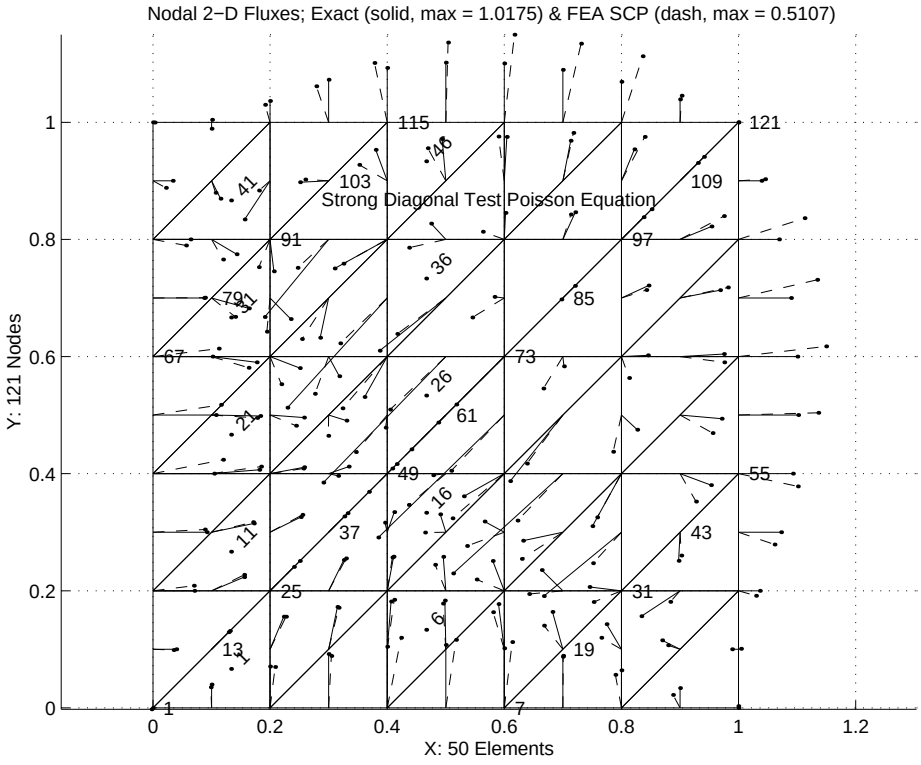


Figure 11.53 *Nodal SCP and exact nodal flux vectors*

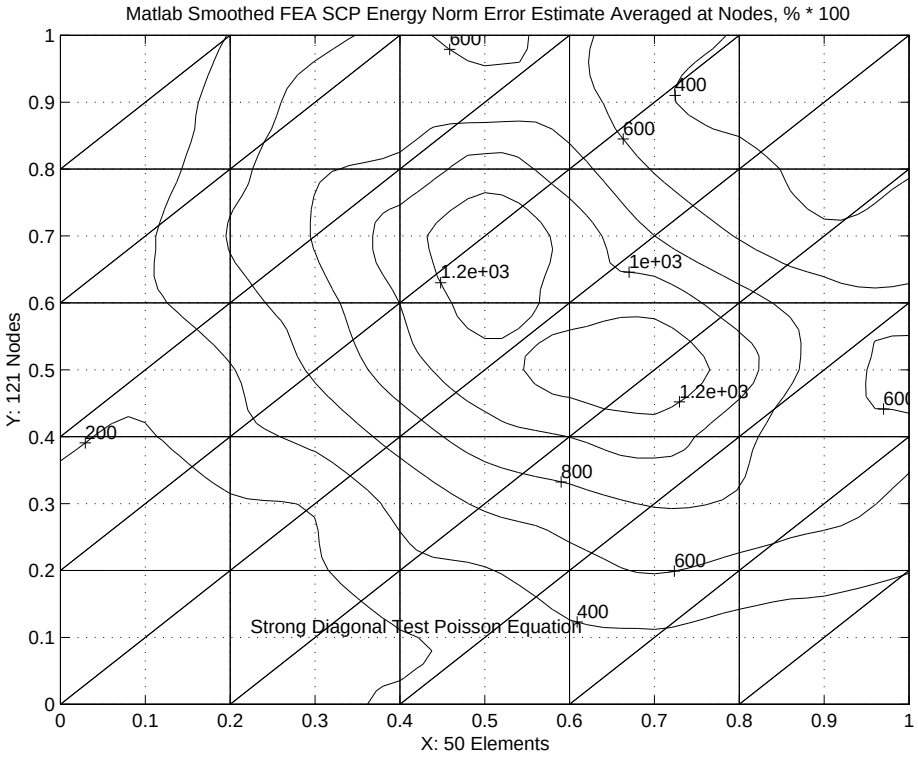


Figure 11.54 *Initial finite element energy norm error estimate*

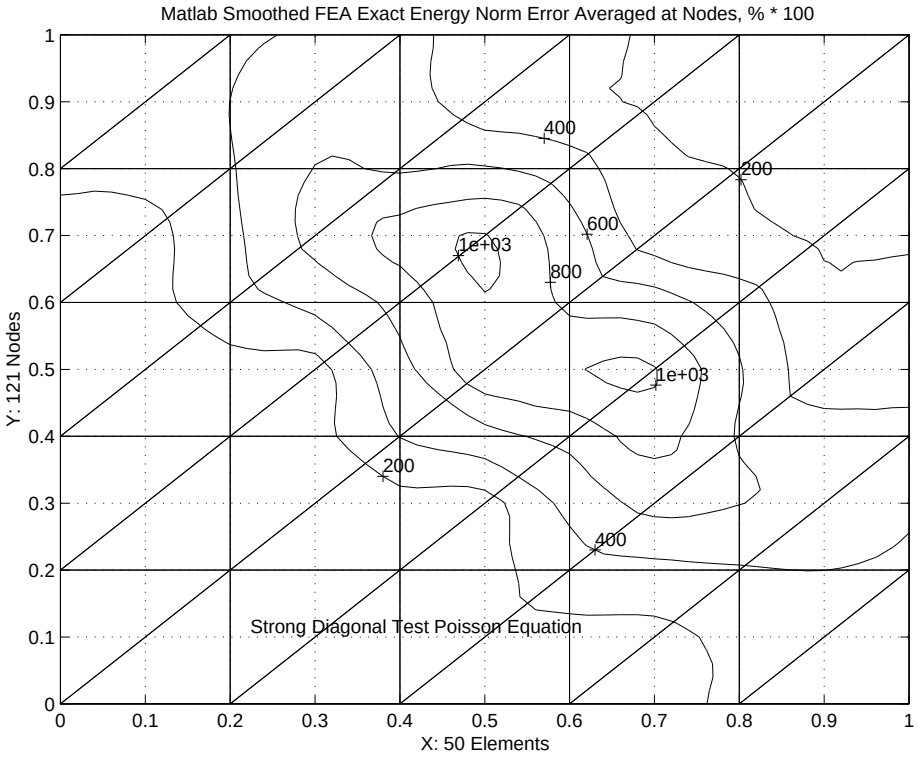


Figure 11.55 *Exact energy norm error estimate*

Fig. 11.51. That figure represents the continuous nodal fluxes, **a** computed in Eqs. 2.45 and 6.1, that will be interpolated to serve as a way to define the flux error estimate at any point in Eqs. 2.46 and 5.20. These vectors represent the results of the averaging obtained from the SCP. They can be used in a typical post-processing activity and/or further processed to yield an error indicator. If the error in the energy is to be computed then we use the last version of Eqs. 5.16 and 6.3, but with $\boldsymbol{\sigma}$ replaced with $\boldsymbol{\sigma}^*$. The integrals are carried out at the element level in a process similar to that used in formulating the element matrices.

An important difference at this stage is that more quadrature points are probably needed, because of the polynomial degree of **P** in Eq. 2.45 is of higher degree than the **B** matrix, and this will require the computation of the standard flux estimate, $\mathbf{E}^e \mathbf{B}_j^e \mathbf{u}^e$, rather than recovering the saved values of **B** that were used in the SCP averaging process. That is an additional cost that must be paid to extend the SCP averaging process to an error indicator.

The corresponding exact nodal flux vectors are given in Figs. 11.52. They are not shown to the same scale. The crude mesh is estimating the maximum flux value to be 0.5107 versus a maximum of 1.0175 from the exact values. This is because the crude mesh is spanning the diagonal region where the exact flux changes from very large to very small fluxes. As we refine the mesh we expect these flux vectors to approach each other. In Fig. 11.53 we see the error (magnitude and direction) in the continuous SCP nodal fluxes and the exact nodal fluxes. In this case they are seen at the same scale. The solid line gives the magnitude and direction of the exact vector, while the dashed line gives the approximate vector. Wherever they are in agreement one should only be able to see the solid line. They start from the same point where they were computed and each ends with a dot instead of an arrowhead so as to reduce the clutter as the mesh is refined. Here we see some vectors (along the horizontal, vertical and 45 degree diagonal) agree on direction but still have significant differences in magnitudes. Most other vectors also have large errors in directions. It is reassuring to see these differences vanish as the error indicator gets smaller.

The comparison of the SCP energy norm error estimate and the exact error in the energy norm is seen by examining Figs. 11.54 and 55. Even though they have slightly different shapes it is reassuring that even with this crude mesh the SCP error estimating process is giving very similar locations for the highest error. The SCP estimate is higher than the exact values but we expect them to approach the same values as the mesh is refined. The same two energy norm errors are represented as surfaces in Figs. 11.56 and 57. It is informative to see how the energy norm estimate we will compute compares to the true error in the value of the solution variable, ϕ. Those values are shown as contours and a surface plot in Figs. 11.58 and 59, respectively. While representing a different measure of error, with different units, both forms are showing the peak error occurring in the same general regions. Note that the exact function error is zero on the (entire) boundary where essential boundary conditions are applied while the exact and SCP energy norm estimates of the error are not zero in those regions. It is common for energy based error estimates to have their largest error in such regions. This suggests that while the function values are accurate there the gradient is not. That is another reason why one may want to use somewhat smaller element sizes near the boundaries.

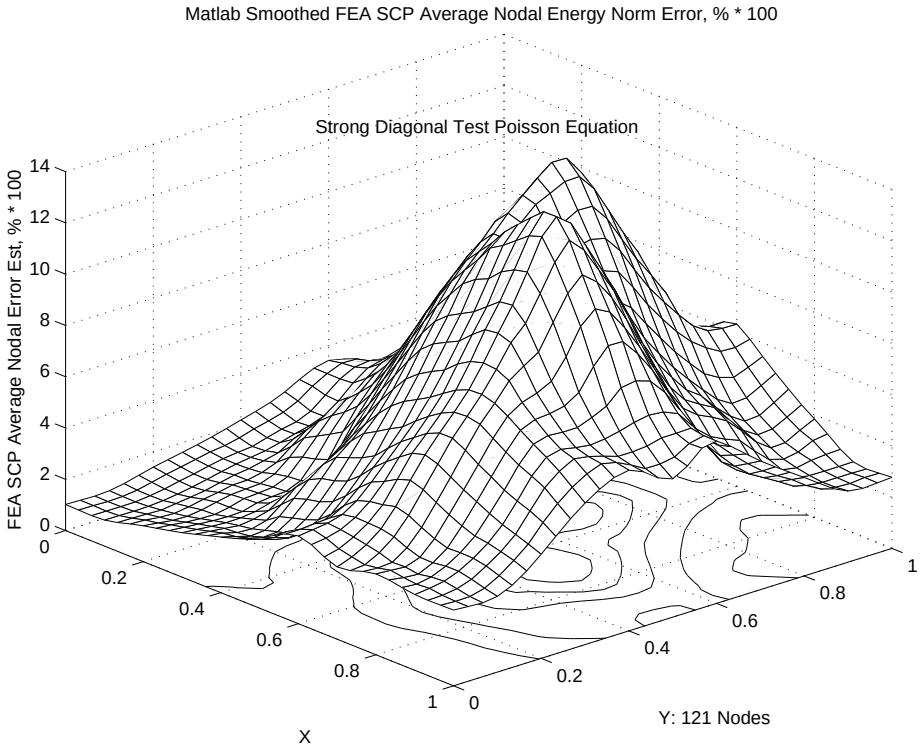


Figure 11.56 *Surface of SCP energy norm error estimate*

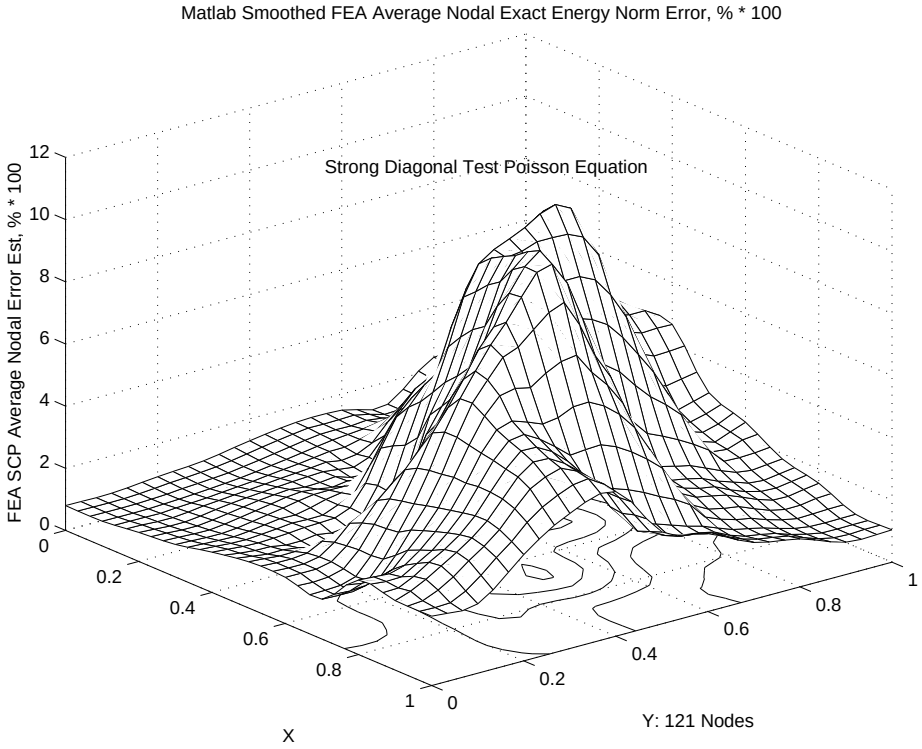


Figure 11.57 *Surface of exact energy norm error estimate*

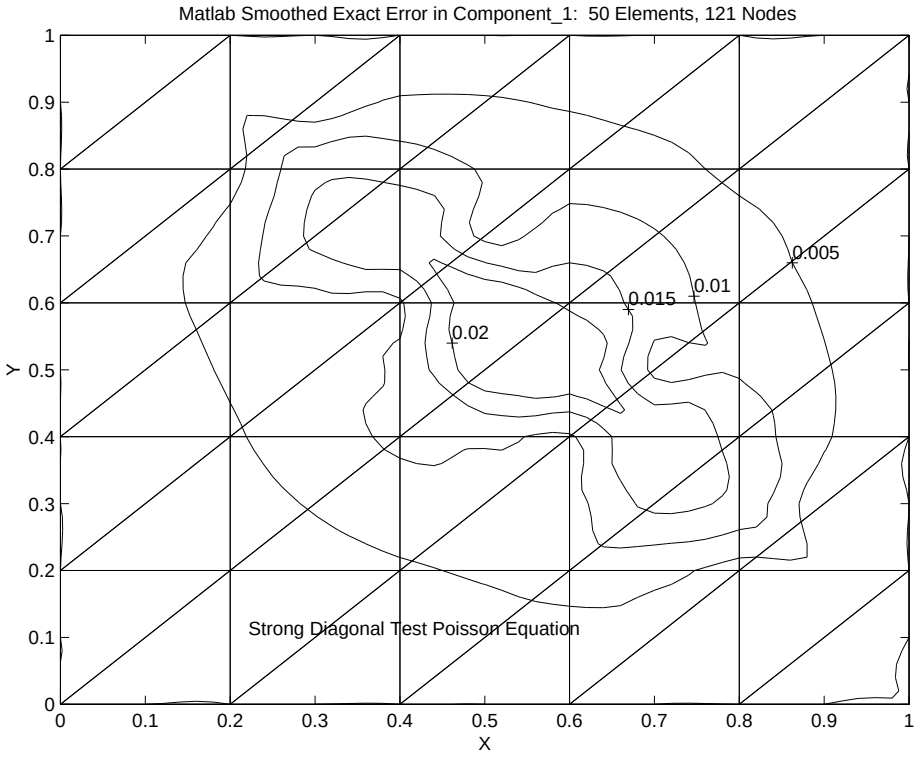


Figure 11.58 *Contours of exact solution error at the nodes (only)*

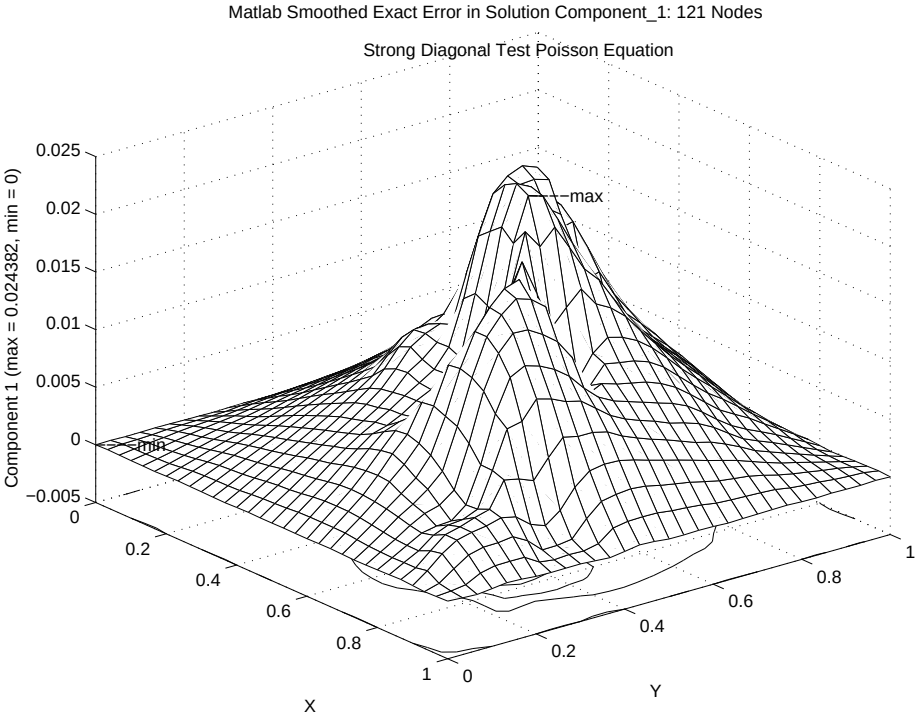


Figure 11.59 *Surface of exact solution error at the nodes (only)*

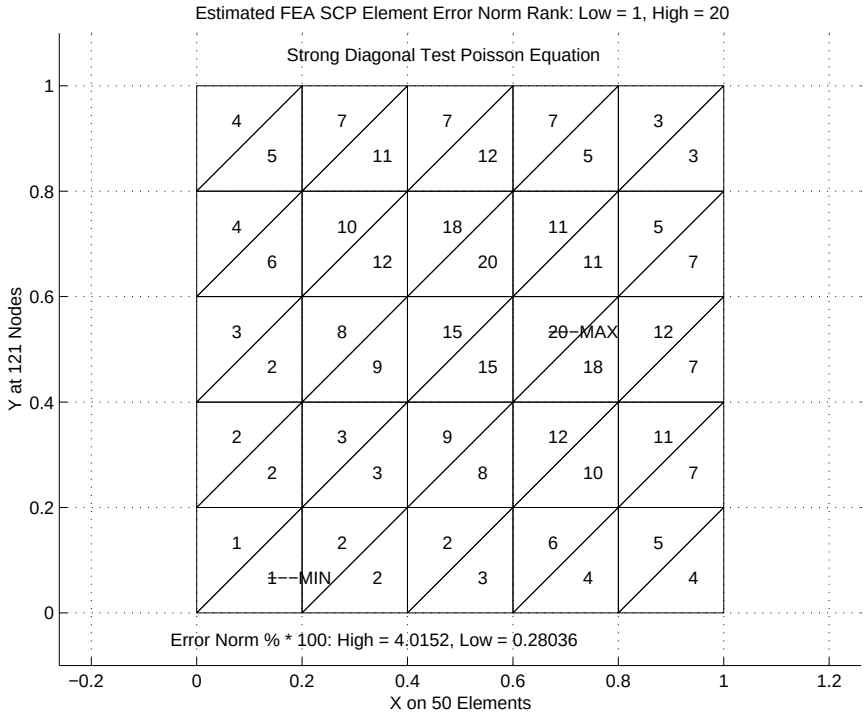


Figure 11.60 *Relative ranking of element error estimates*

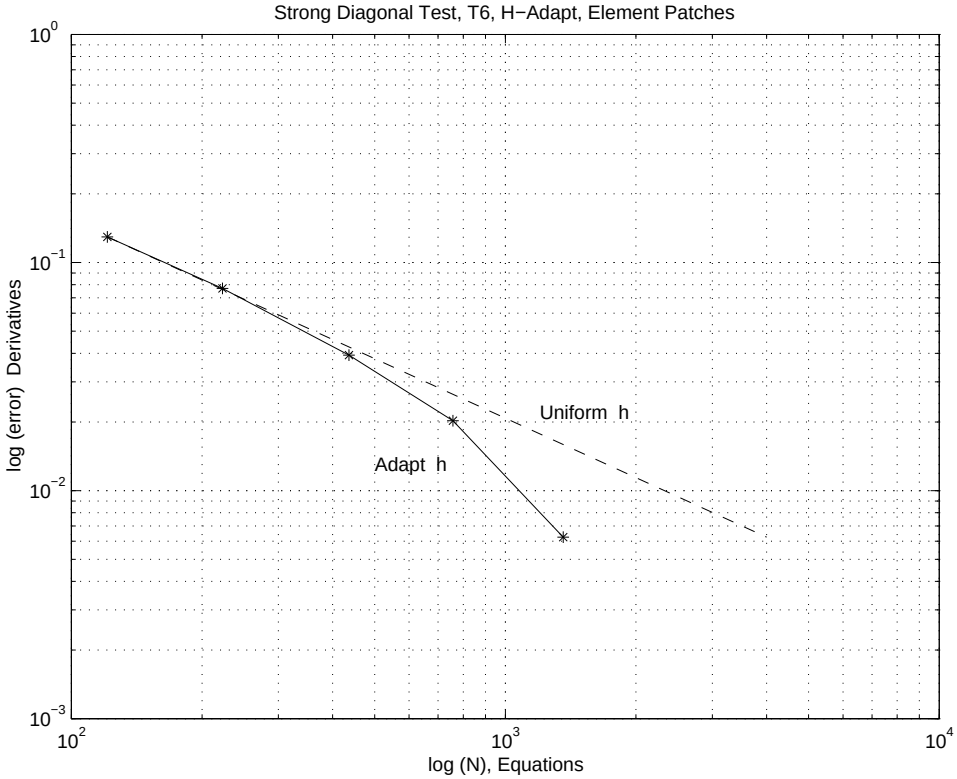


Figure 11.61 *Reduction in the error*