Chapter 10

SCALAR FIELDS

10.1 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 computer the value of the scalar quantity, its gradient, and sometimes its integral over the solution domain. Typical applications of scalar fields includes: 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.

10.2 Variational formulation

We can obtain from any book on heat transfer the governing differential equation for steady and un-steady state heat conduction. The most general form of the heat conduction equation is the transient three-dimensional equation:

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k_z \frac{\partial T}{\partial z}\right) + Q = \frac{\partial}{\partial t}(\rho c T) \tag{10.1}$$

where, k_x, k_y, k_z = thermal conductivity coefficients, T = temperature, Q = heat generation per unit volume, ρ = density, and c = specific heat. If we focus our attention to the two-dimensional ($\partial / \partial z = 0$) steady-state ($\partial / \partial t = 0$) problem, such as Fig. 10.2.1, the governing equation becomes

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) + Q = 0 \tag{10.2}$$

in which k_x, k_y, and Q are known. Equations (10.1) or (10.2), together with the appropriate boundary conditions specify the problem completely. The two most commonly encountered boundary conditions are those in which the temperature, T, is specified on the boundary, i.e.,

$$T = T(s) \text{ on } \Gamma_s. \tag{10.3}$$

boundary heat flux is specified at a point, i.e.,

$$k_x \frac{\partial T}{\partial x} n_x + k_y \frac{\partial T}{\partial y} n_y + q + h(T - T_r) = 0 \text{ on } \Gamma_q.$$

or

$$k_n \frac{\partial T}{\partial n} + q + h(T - T_r) = 0, \tag{10.4}$$

where n_x and n_y are the direction cosines of the outward normal to the boundary surface, q represents the known heat flux per unit of surface, and $h(T - T_r)$ is the convection heat loss per unit area. Only one of these two terms is non-zero on a surface. Otherwise we combine them into a *mixed* or *Robin* condition with a part (h) proportional to the unknown surface value and a second part known on the surface.

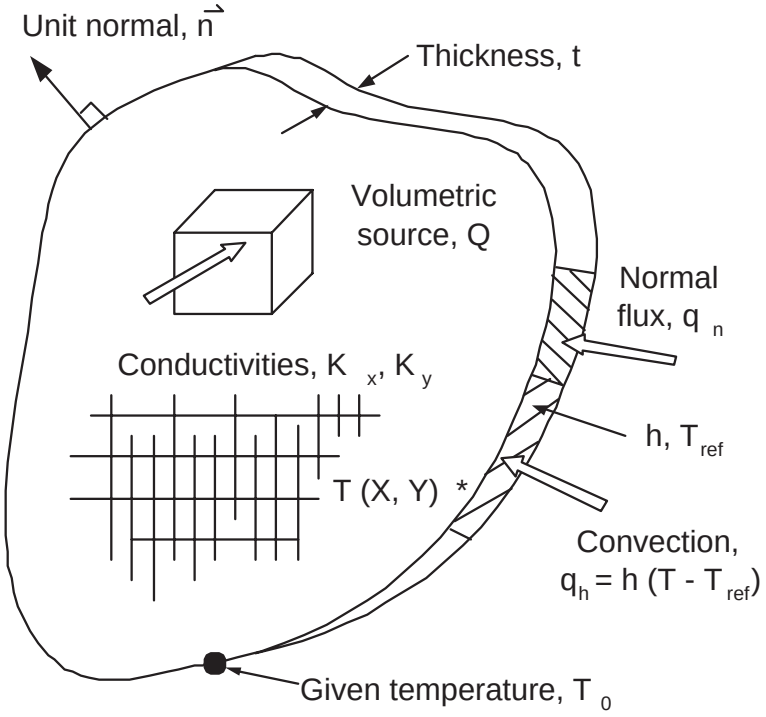


Figure 10.2.1 An anisotropic heat transfer region

As stated previously, an alternative formulation to the above heat conduction problem is possible using the calculus of variations. Euler's theorem of the calculus of variations states that if the integral

$$I(u) = \int_V f(x, y, z, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z})\, dx\, dy\, dz + \int_S (gu + hu^2/2)\, da \tag{10.5}$$

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

$$\frac{\partial}{\partial x}\frac{\partial f}{\partial(\partial u/\partial x)} + \frac{\partial}{\partial y}\frac{\partial f}{\partial(\partial u/\partial y)} + \frac{\partial}{\partial z}\frac{\partial f}{\partial(\partial u/\partial z)} - \frac{\partial f}{\partial u} = 0 \tag{10.6}$$

within the region, provided u satisfies the essential boundary conditions in both cases. We can verify that the minimization of the volume integral

$$I = \int_V \left[\frac{1}{2}\left\{k_x(\frac{\partial T}{\partial x})^2 + k_y(\frac{\partial T}{\partial y})^2 + k_z(\frac{\partial T}{\partial z})^2\right\} - QT\right] dV + \int_S \left[gT + hT^2/2\right] da \tag{10.7}$$

leads directly to the formulation equivalent to Eq. (10.2) for the steady-state case. The functional volume contribution is

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

Thus, if f is to be minimized it must satisfy Eq. (10.6). Here

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

so Eq. (10.6) results in

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

verifying that the function f does lead to correct steady state formulation, if Eq. (10.3) is also satisfied. Euler also stated that the natural boundary condition associated with Eq. (10.5) 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 forced. Here this equation simplifies to

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

where k_n is the conductivity in the direction of the unit normal vector. In the above form if $h = 0$, then $g = q$, the normal heat flux. Otherwise, it would be the external temperature convecting effect, $g = -hT_r$. This is the type of boundary condition given earlier in Eq. (10.4).

10.3 Element and Boundary Matrices

From Eqs. (10.1) and (10.7) it is clearly seen that the two-dimensional functional required for the steady-state analysis is

$$I = \int_A \left[\tfrac{1}{2} \left\{ k_x (\frac{\partial T}{\partial x})^2 + k_y (\frac{\partial T}{\partial y})^2 \right\} - QT \right] t \, dx \, dy + \int_\Gamma (gT + hT^2/2) t \, ds. \qquad (10.8)$$

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

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

and the boundary segment contributions are

$$I^b = \int_{\Gamma^b} (gT + hT^2/2) t \, ds.$$

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

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

Here the element matrices are

$$\mathbf{S}^e = \int_{A^e} (k_x^e \, \mathbf{H}_x^{e^T} \mathbf{H}_x^e + k_y^e \, \mathbf{H}_y^{e^T} \mathbf{H}_y^e) t^e \, da \qquad (10.9)$$

$$\mathbf{C}_Q^e = \int_{A^e} \mathbf{H}^{e^T} Q^e t^e \, da \qquad (10.10)$$

$$\mathbf{S}_h^b = \int_{\Gamma^b} h^b \mathbf{H}^{b^T} \mathbf{H}^b t^b \, ds, \quad \mathbf{C}_h^b = \int_{\Gamma^b} T_r^b h^b \mathbf{H}^{b^T} t^b \, ds \qquad (10.11)$$

$$\mathbf{C}_q^b = \int_{\Gamma^b} q^b \mathbf{H}^{b^T} t^b \, ds \qquad (10.12)$$

where $\mathbf{H}$ denotes the shape functions and $\mathbf{H}_x = \partial \mathbf{H} / \partial x$, etc. For this class of problem there is only one unknown temperature per node. Once again, if $\mathbf{T}$ denotes all of these unknowns then $\mathbf{T}^e \subset \mathbf{T}$ and $\mathbf{T}^b \subset \mathbf{T}$. Figure 10.3.1 illustrates where these typical conduction, convection, and source terms are inserted in the algebraic equations of thermal equilibrium. Likewise, Figure 10.3.2 reminds us that a Dirichlet (essential) boundary condition assigns a value to one of the degrees of freedom and introduces a 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 $B^2 M$. 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 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.

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 10.3.3 shows a planar rectangular region with homogeneous conduction properties, k, and internal and external specified edge temperatures, T_A and T_B. In addition, it has free convection, q_h, over its face to a fluid with homogeneous convection properties, h and T_∞. 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 T_A and T_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

that 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 either positive or negative increments of the relative temperature change. Commercial visualization codes often have the ability to plot a full region when provided a partial analysis region and the identification of the symmetry and anti-symmetry planes.

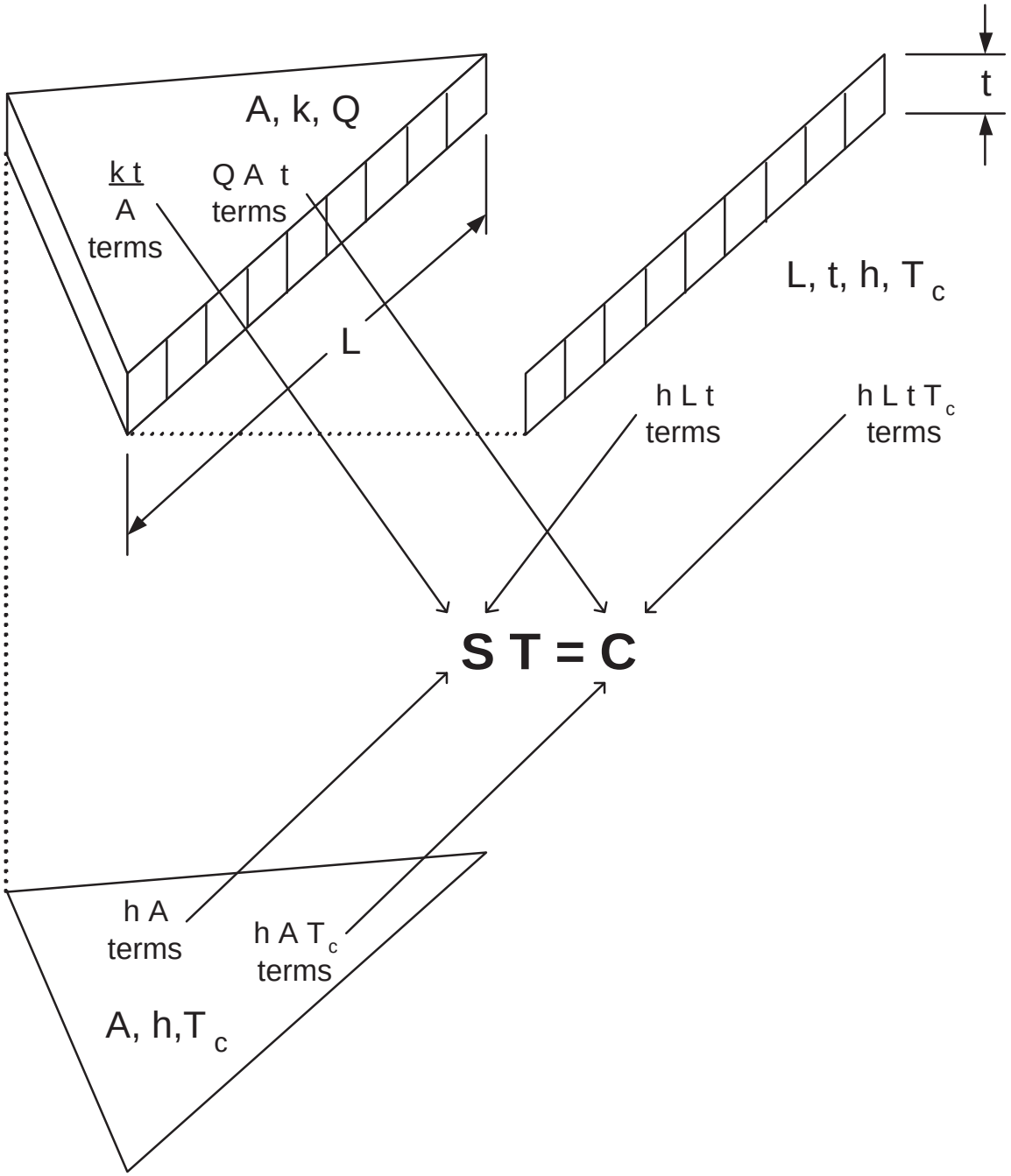


Figure 10.3.1 Contributions from conducting, convecting and source regions

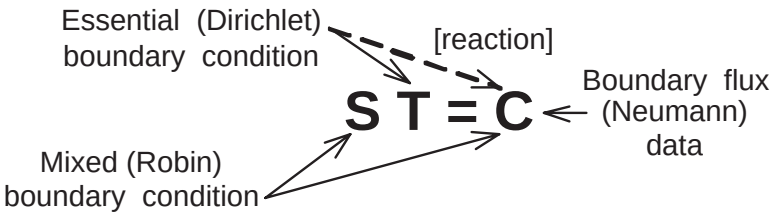
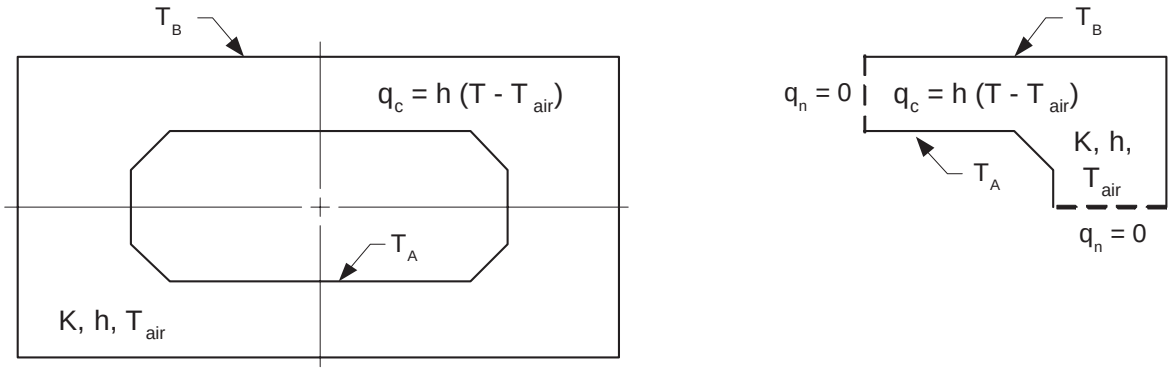
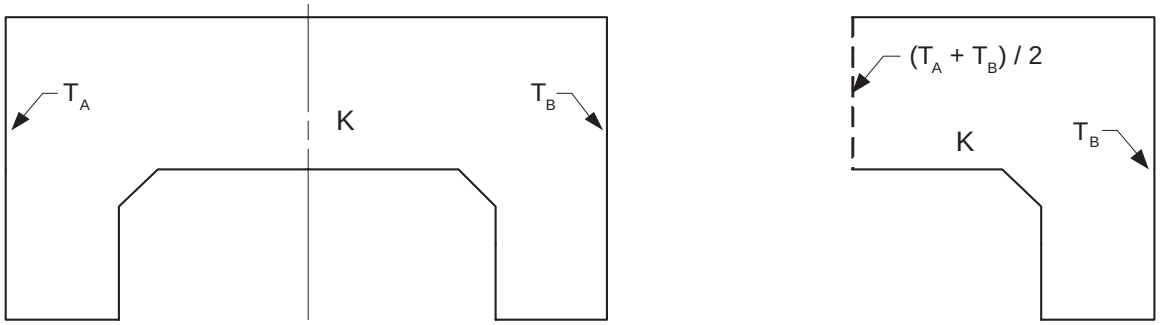


Figure 10.3.2 Contributions of boundary conditions to algebraic system



a) Quarter symmetry model



b) Half anti-symmetry model

Figure 10.3.3 Typical symmetry and anti-symmetry thermal models

10.4 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

$$\begin{aligned} \mathbf{H}^e_x &= \partial \mathbf{H}^e / \partial x = \lfloor\, b_1 \;\; b_2 \;\; b_3 \,\rfloor^e / 2A^e = \mathbf{d}_x \\ \mathbf{H}^e_y &= \partial \mathbf{H}^e / \partial y = \lfloor\, c_1 \;\; c_2 \;\; c_3 \,\rfloor^e / 2A^e = \mathbf{d}_y \,. \end{aligned} \tag{10.13}$$

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

$$\mathbf{S}^e = \frac{k_x^e t^e}{4A^e}\begin{bmatrix} b_1b_1 & b_1b_2 & b_1b_3 \\ b_2b_1 & b_2b_2 & b_2b_3 \\ b_3b_1 & b_3b_2 & b_3b_3 \end{bmatrix}^e + \frac{k_y^e t^e}{4A^e}\begin{bmatrix} c_1c_1 & c_1c_2 & c_1c_3 \\ c_2c_1 & c_2c_2 & c_2c_3 \\ c_3c_1 & c_3c_2 & c_3c_3 \end{bmatrix}^e . \quad (10.14)$$

This is known as the *element conductivity matrix*. Note that this allows for different conductivities in the x- and y- directions. Equations (10.14) 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. (10.10) can be integrated via Eq. (11.3) to yield

$$\mathbf{C}^{eT} = \frac{Q^e A^e t^e}{3}\lfloor 1 \quad 1 \quad 1 \rfloor. \quad (10.15)$$

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, is constant then the boundary segment square matrix is obtained from Eq. (10.11) as

$$\mathbf{S}^b = \frac{h^b L^b t^b}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}. \quad (10.16)$$

Similarly if a constant normal flux, q, is given then the boundary flux vector is

$$\mathbf{C}^b = \frac{q^b L^b t^b}{2}\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}. \quad (10.17)$$

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

If one wished to code this simple element in closed form it is very easy to do as shown in Fig.10.4.1. 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. 10.12 in closed form, as shown in Fig. 10.4.2. A common use of two-dimensional models is to predict the temperature in thin cooling fins. Then Eq. 10.11 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 each mixed boundary segment (face). The implementation is given in Fig. 10.4.3. We will shortly illustrate these matrix definitions with some simple applications.


```
! ..............................................................    ! 1
!    ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***    ! 2
! ..............................................................    ! 3
! Define any new array or variable types, then give statements      ! 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_PROP >= 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 10.4.1 Anisotropic linear triangle conduction element


```
! ..................................................................... ! 1
! ***  SEG_COL_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***          ! 2
! ..................................................................... ! 3
! Given normal flux on a straight boundary segment (BS) edge:          ! 4
! Standard form: -K_n * U,n = Q_NORMAL_SEG, where Q_NORMAL_SEG         ! 5
! is from control keywords normal_flux, flux_thick, or via            ! 6
! optional flux segment real properties: 1-flux, 2-thickness          ! 7
 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_1             !23
```

Figure 10.4.2 Straight linear edge flux source element

```
! ..................................................................... ! 1
!  ***  MIXED_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***        ! 2
! ..................................................................... ! 3
! Define any new array or variable types, then give statements         ! 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 10.4.3 Face convection for a linear triangle segment

10.5 Linear Triangle Applications

10.5.1 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 $k_x = k_y = k$. 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. (10.2) 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. 10.5.1. 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:

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

From Eq. (10.14) the conduction square matrix for elements 1, 2, and 4 are

$$\mathbf{S}^e = \frac{8(1)}{4(2)}\begin{bmatrix} 4 & -4 & 0 \\ -4 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{8(1)}{4(2)}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & -4 \\ 0 & -4 & 4 \end{bmatrix} = \begin{bmatrix} 4 & -4 & 0 \\ -4 & 8 & -4 \\ 0 & -4 & 4 \end{bmatrix}. \qquad (10.27)$$

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 $\mathbf{ST} = \mathbf{C}$ where

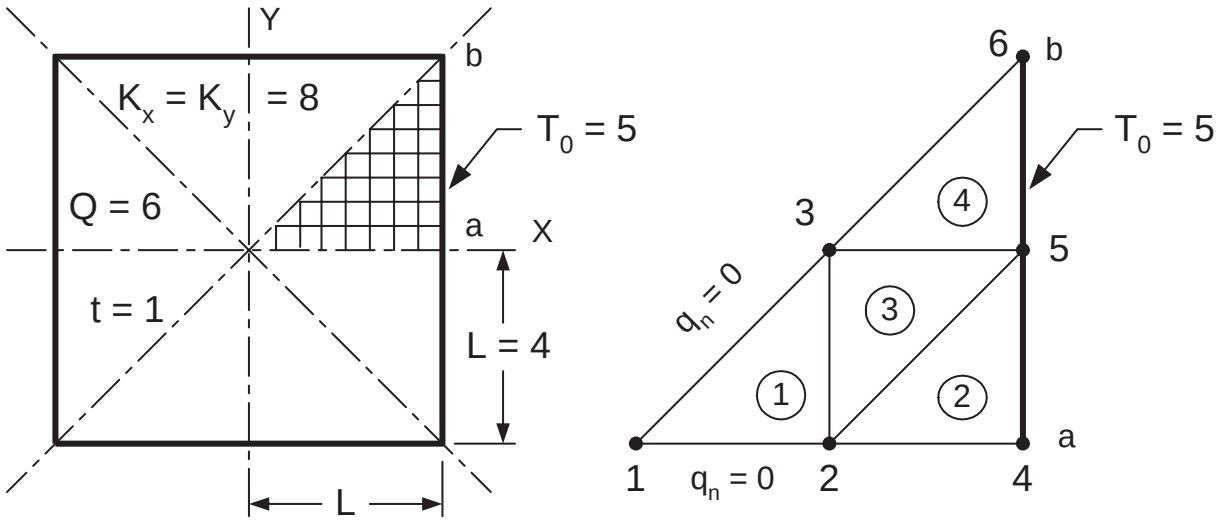


Figure 10.5.1 A one-eighth symmetry model of a square

```
title "CONDUCTION EXAMPLE, T3, INTERNAL SOURCE"                    ! 1
b_rows          2 ! Number of rows in the B (operator) matrix      ! 2
dof             1 ! Number of unknowns per node                    ! 3
el_nodes        3 ! Maximum number of nodes per element            ! 4
elems           4 ! Number of elements in the system               ! 5
gauss           1 ! Maximum number of quadrature points            ! 6
nodes           6 ! Number of nodes in the mesh                    ! 7
shape           2 ! Element shape, 1=line, 2=tri, 3=quad, 4=hex    ! 8
space           2 ! Solution space dimension                       ! 9
el_homo           ! Element properties are homogeneous             !10
el_real         4 ! Number of real properties per element          !11
remarks         9 ! Number of user remarks, e.g. property names    !12
end               ! Terminate the keyword control, remarks follow  !13
Conduction example of Section 10.4                 6               !14
K_x = K_y = 8, Q = 6, K_xy = 0                    / |              !15
L_1_4 = L_4_6 = 4, Thickness = 1                 /  |              !16
with 1/8 symmetry, so natural BC on            /(4)|              !17
edges 1_6 and 1_4 ia q_n = 0                 3-----5              !18
EBC on edge 4_6 is T = 5                    / |(3)/|              !19
K_x T,xx + 2K_xy T,xy + K_y T,yy + Q = 0 /  | /  |              !20
[gauss > 0 turns on flux averaging      /(1)|/(2)|              !21
and possibly post-processing]         1----2----4              !22
   1    0 0.   0. ! node, ebc flag, x, y                         !23
   2    0 2.   0.                                                !24
   3    0 2.   2.                                                !25
   4    1 4.   0.                                                !26
   5    1 4.   2.                                                !27
   6    1 4.   4.                                                !28
  1  1  2  3      ! elem, three nodes                             !29
  2  2  4  5                                                      !30
  3  2  5  3                                                      !31
  4  3  5  6                                                      !32
   4    1 5.      ! node, dof, value of EBC                       !33
   5    1 5.                                                     !34
   6    1 5.                                                     !35
  1 8. 8. 0. 6.   ! elem, K_x, K_y, K_xy, Q (homogeneous)        !36
```

Figure 10.5.2 Sample data for square bar thermal analysis

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

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 q's 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

$$\begin{bmatrix} 4 & -4 & 0 \\ -4 & 16 & -8 \\ 0 & -8 & 16 \end{bmatrix}\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \end{Bmatrix} = \begin{Bmatrix} 4 \\ 12 \\ 12 \end{Bmatrix} - T_4\begin{Bmatrix} 0 \\ -4 \\ 0 \end{Bmatrix} - T_5\begin{Bmatrix} 0 \\ 0 \\ -8 \end{Bmatrix} - T_6\begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}. \quad (10.28)$$

Substituting the data that the exterior surface temperatures are $T_4 = T_5 = T_6 = 5$ 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

$$\mathbf{S}^{*-1} = \frac{1}{512}\begin{bmatrix} 192 & 64 & 32 \\ 64 & 64 & 32 \\ 32 & 32 & 48 \end{bmatrix} \quad (10.29)$$

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

$$\mathbf{T}^* = \begin{Bmatrix} 8.750 \\ 7.750 \\ 7.125 \end{Bmatrix} = \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \end{Bmatrix}.$$

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 equation yields: $-4T_2 + 8T_4 - 4T_5 = -4(7.75) + 8(5) - 4(5) = 4 + q_4$, or simply $-15 = q_4$. The other two nodal fluxes are $q_5 = -29$, $q_6 = -4$ and the internal heat generated was $\sum_e Q^e A^e t^e = +48$.


```
*** 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 10.5.3 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. One must always modify the column vector for the temperatures to include essential boundary condition data. Another example of thermal analysis will be given later to show the importance of computing temperatures as input data needed in computing thermal stress. The input data for this example are given in Fig. 10.5.2 and selected outputs are given in Fig. 10.5.3.

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. 10.5.4 and 5, respectively. In the first figure we get an 'eyeball' check that the contour lines appear perpendicular to the insulated boundaries and parallel to the Dirichlet boundary. The constant element flux vectors are shown in Fig. 10.5.6 and the averaged nodal vectors (to be defined later) are shown in Fig. 10.5.7. The last figure should have the vectors parallel to the insulated boundaries and they seem to do that reasonably well.

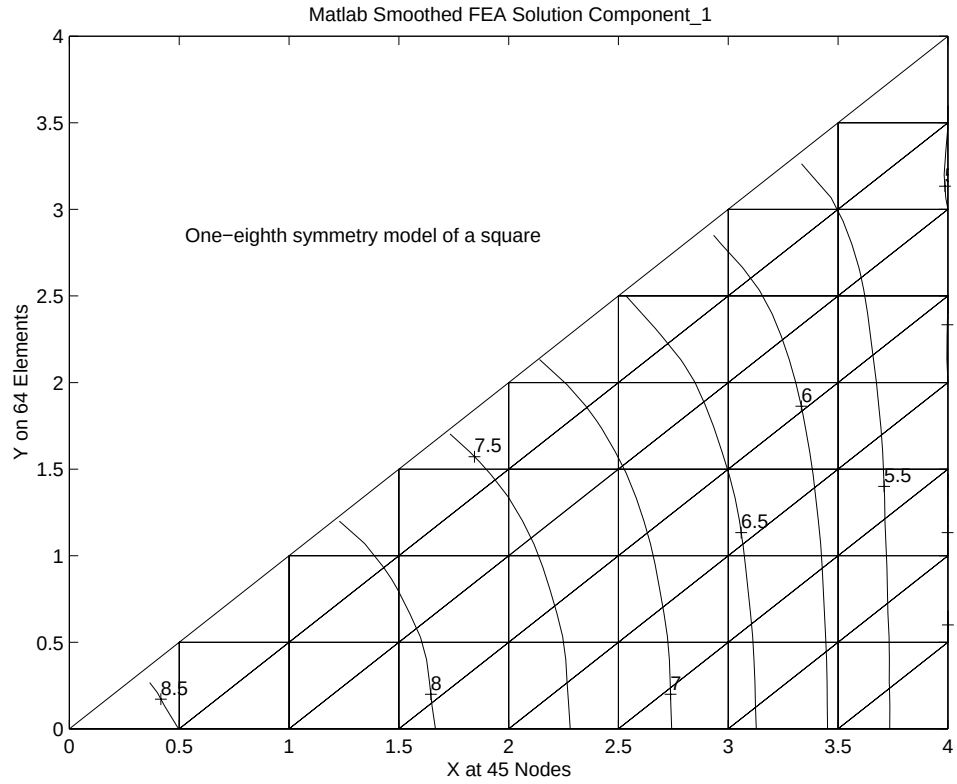


Figure 10.5.4 Temperatures on the square segment

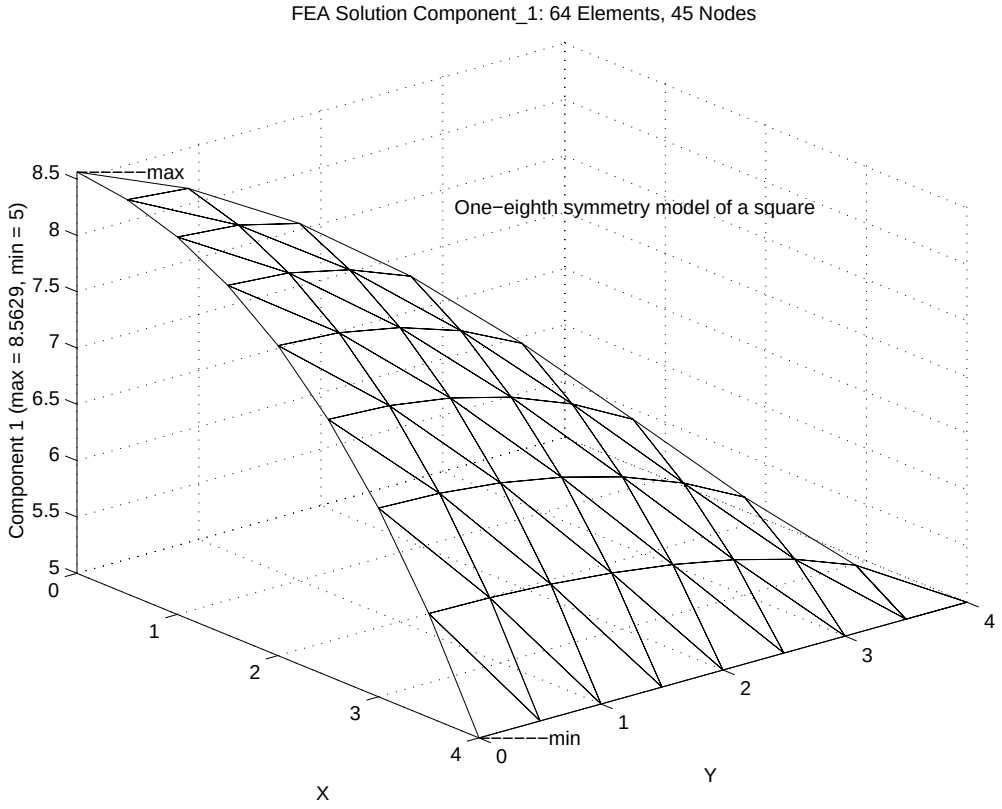


Figure 10.5.5 Temperature carpet plot over the square

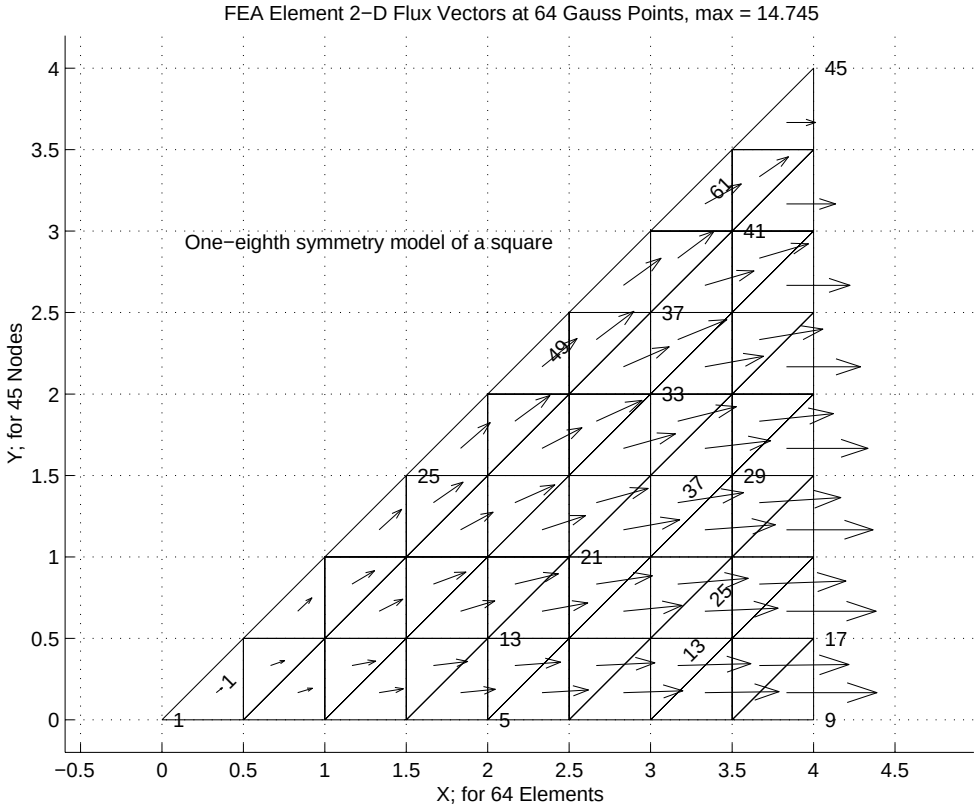


Figure 10.5.6 Constant element flux vectors

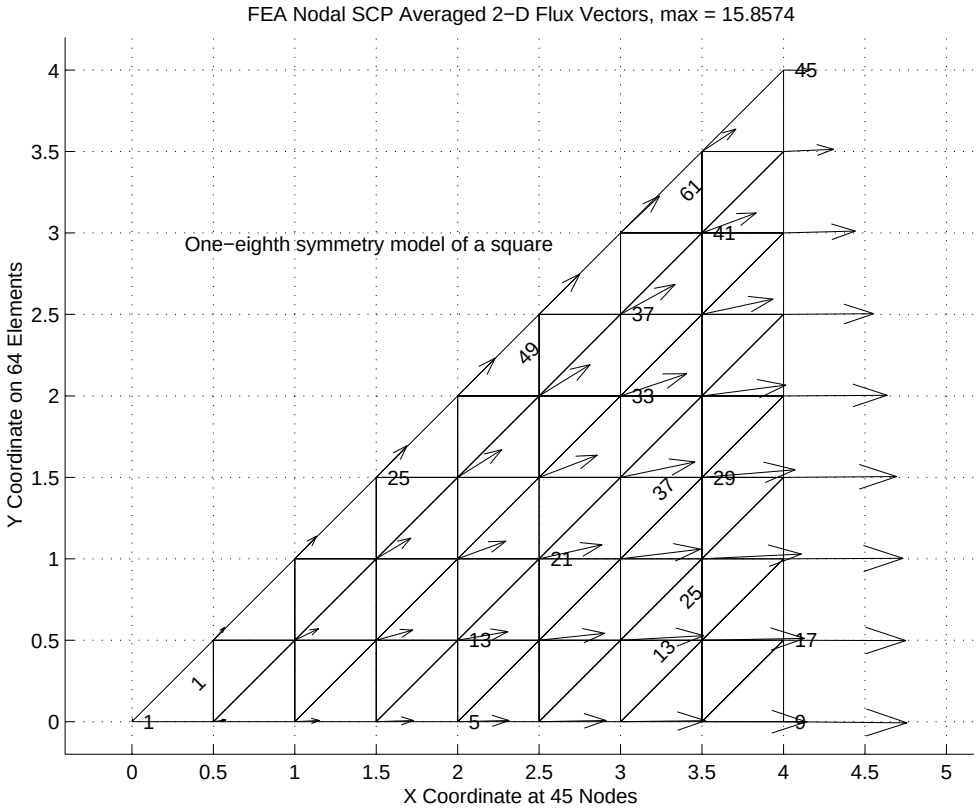


Figure 10.5.7 Averaged flux vector over the square

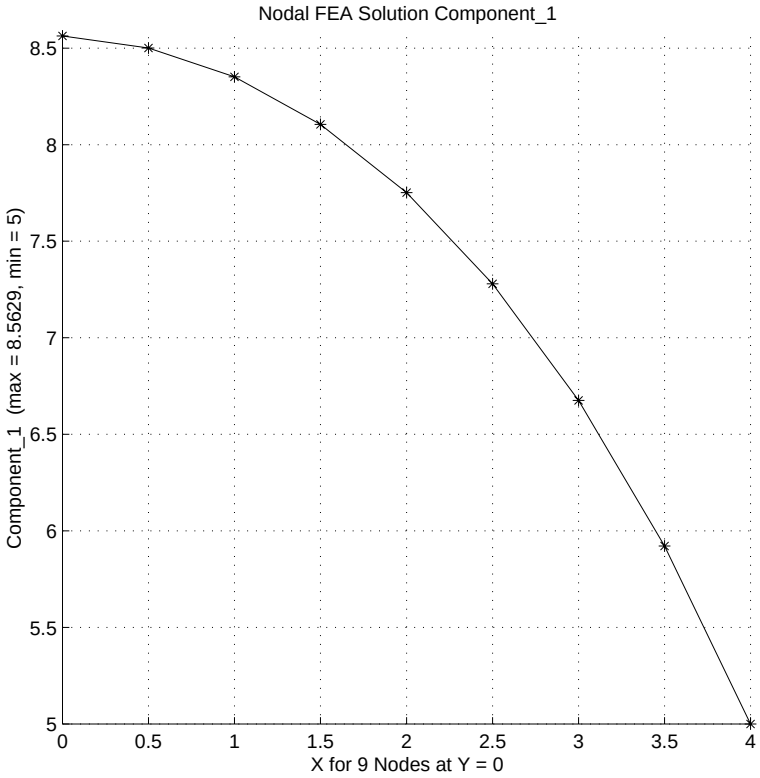


Figure 10.5.8 Temperatures along y = 0

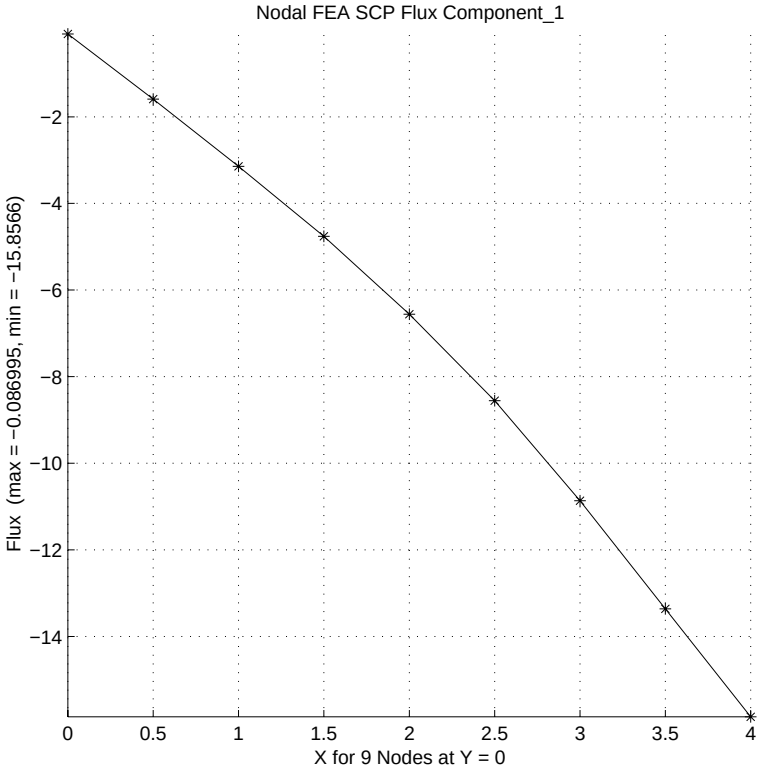


Figure 10.5.9 Heat flux along y = 0

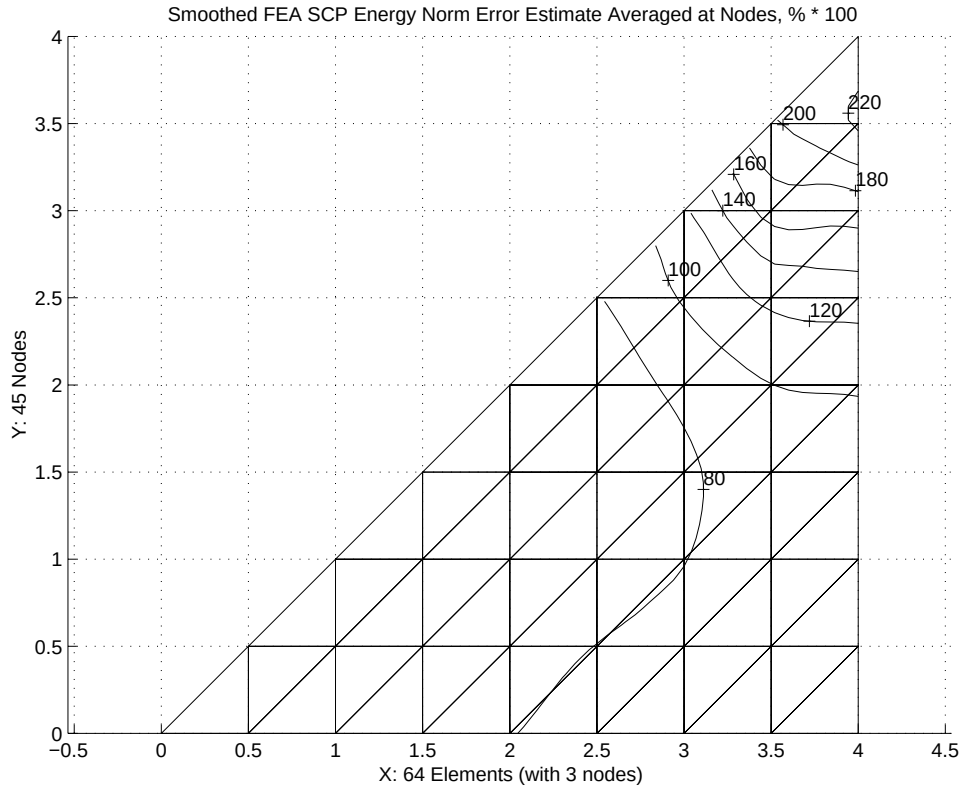


Figure 10.5.10 Contours of energy norm error

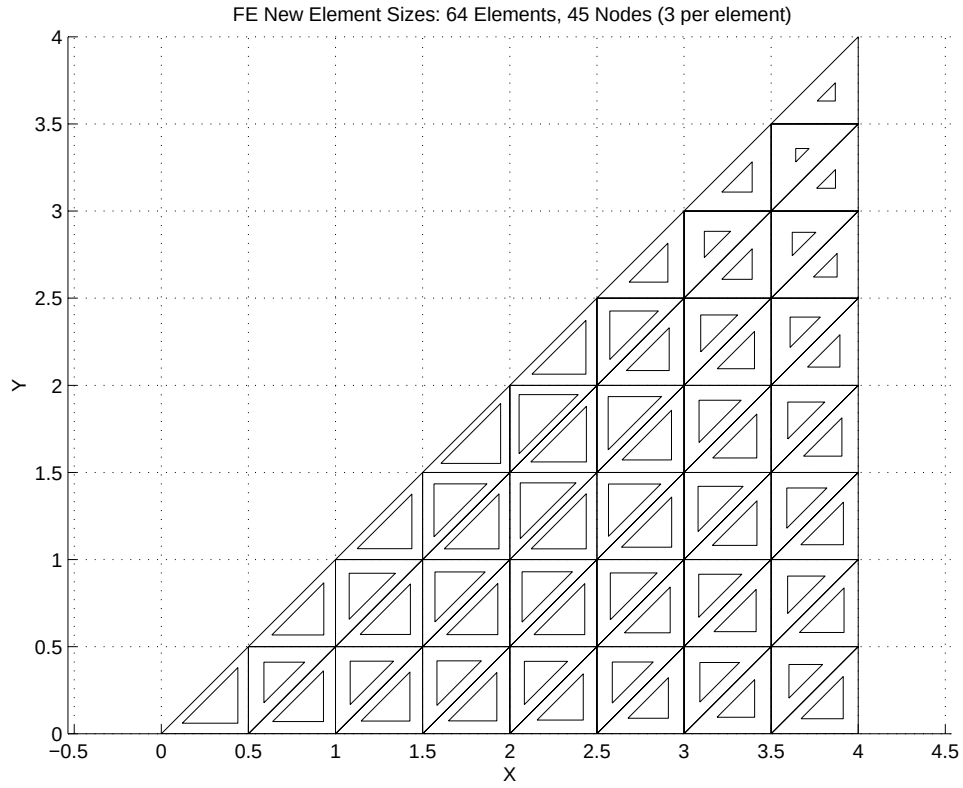


Figure 10.5.11 Suggested maximum element sizes for new mesh

The graphs of the temperature and (tangential) heat flux along the symmetry line of $y = 0$, in Figs. 10.5.8 and 9, show a rather smooth result without obvious wiggles that imply need for a new mesh. However, the energy norm error estimates in Fig. 10.5.10 exceed 2 % 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. 10.5.11.

The same four element T3 mesh was used in the text by Kwon [7], a former student of mine, to analyze a half symmetry triangular region, of Fig. 10.5.8, 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 Fig.s 10.5.9 and 10, 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 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 11, flagged the presence of such sources and their numeric values were read in lines 37-39. (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 10.5.10 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. 10.4.2, 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 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 10.1 lists most of the controls that can be select to define combinations of element types.

To extend the previous lumped flux example to one that uses the source in Fig. 10.4.2 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. 10.5.11. Since a constant normal flux is common a control option 'normal_flux' (line 14) 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

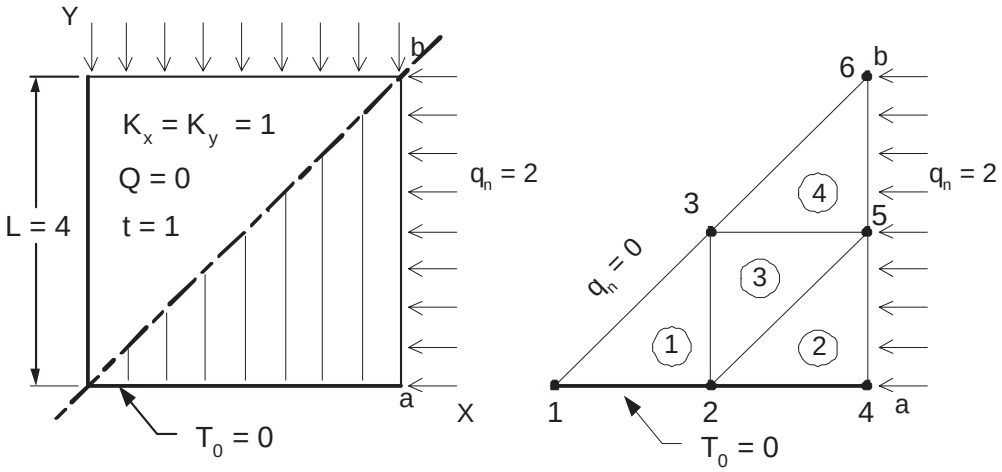


Figure 10.5.8 Square with two flux sides

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

Figure 10.5.9 Triangular region with an edge flux

each edge by using properties defined for each boundary segment (BS). The new data, in Fig. 10.5.12, yield exactly the same temperature results as before. The keyword 'segments' (line 13) caused the boundary integral calculations of Fig. 10.4.2 to be invoked, and told the system that two segments needed to be read (at lines 37-39). Expanding the mesh size to include 64 conduction triangles and 8 edge flux segments yields the mesh and temperatures of Fig.s 10.5.12 and 13, respectively.

10.5.2 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. 10.11, in closed form and they were given in Fig. 10.4.3. Again, it is not uncommon for the convection data, h^b, T_∞, to be constant. To allow for that condition, two user control options are provided: 'convect_coef' and 'convect_temp'. The second one (surrounding temperature) defaults to zero. They can be used to set the corresponding values of two global variables CONVECT_COEF, and CONVECT_TEMP, respectively, for possible use in application dependent matrices, as was done in Fig. 10.4.3. 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.

As a simple example of a cooling fin we will consider the trapezoidal fin given in detail by Allaire. [1] 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. 10.5.14. The corresponding data to invoke the face convection segments, as well as the standard conduction, are given in Fig.10.5.15. The keyword 'mixed_segs' (line 13) is what caused the convection calculations of Fig. 10.4.3 to be invoked and the four sets of connectivity data to be read (lines 41-44). Recall that 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. 10.5.16. 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.


```
*** 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 10.5.10 Selected results for triangle with an edge flux

Table 10.1 Typical Keywords for Multiple Element Types

```
COMMON_WORD  TYPICAL_VALUES    ! REMARKS                    [DEFAULT]
el_types       1 ! Number of different types of elements        [1]
el_segment     3 ! Maximum nodes on element boundary segment    [0]
type_nodes   3 3 ! Number of analysis nodes for element types   [2]
type_gauss   0 0 ! 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]
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]

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]

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

Figure 10.5.11 Combining edge flux segments with conduction

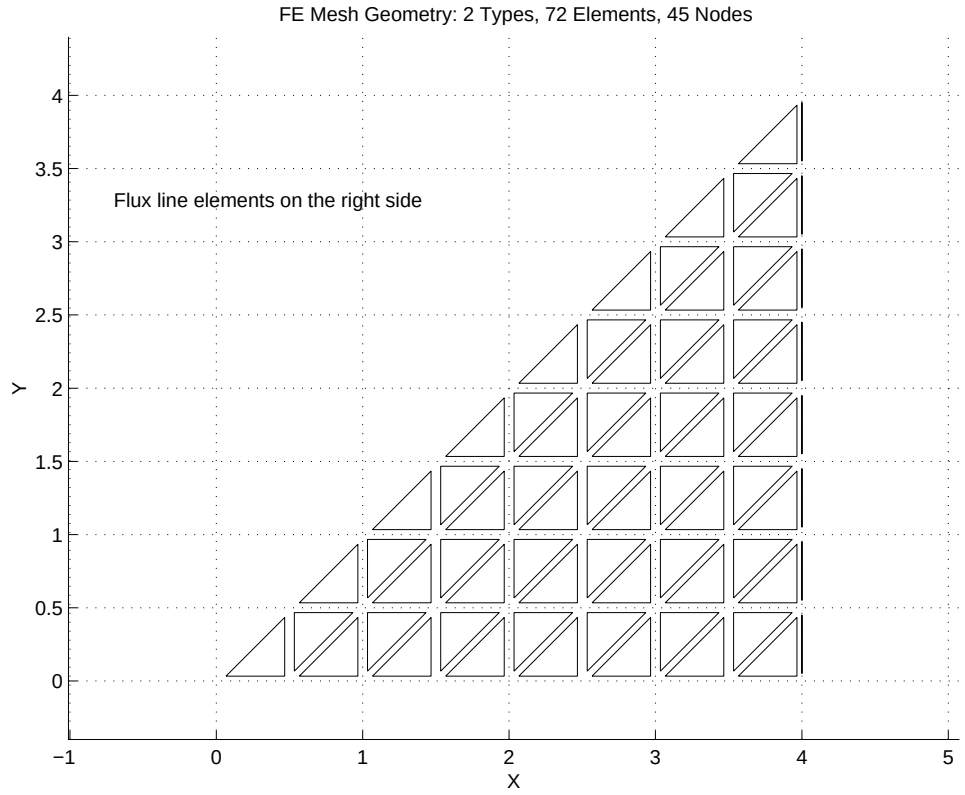


Figure 10.5.12 Exploded conduction triangles and flux line segments

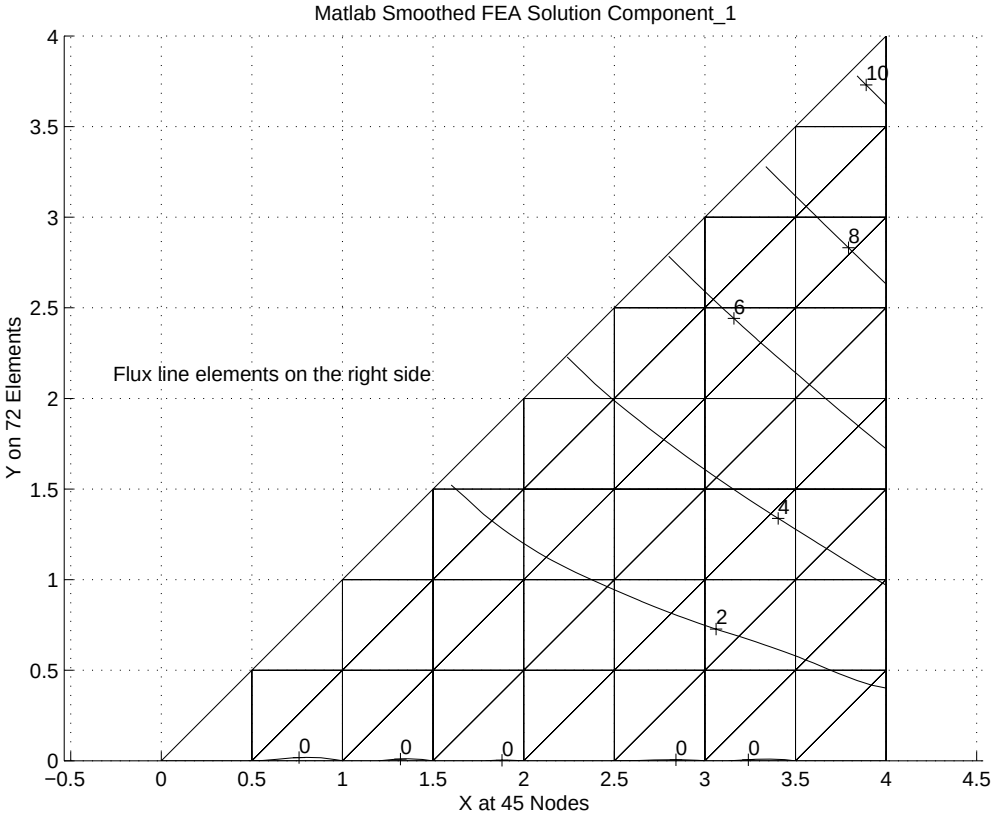


Figure 10.5.13 Temperatures from edge flux sources

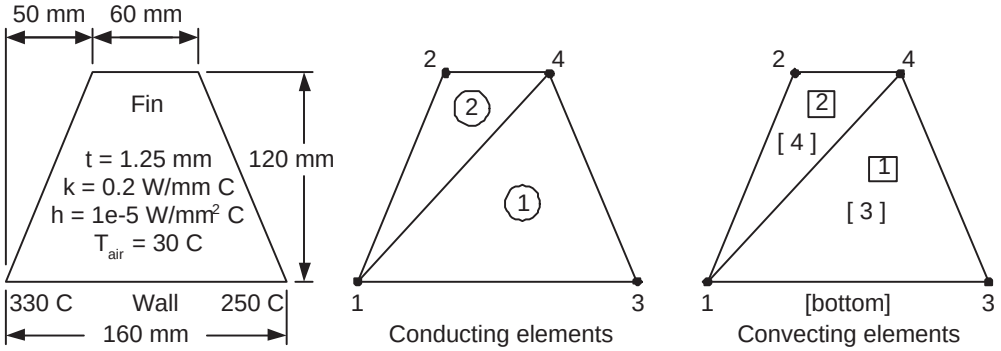


Figure 10.5.14 A cooling fin with air convection on its faces

```
title "Fin face convection, Allaire, Basics FEM 343-353 2T3"      ! 1
elems      2 ! Number of elements in the system                   ! 2
nodes      4 ! Number of nodes in the mesh                        ! 3
space      2 ! Solution space dimension                           ! 4
b_rows     2 ! Number of rows in the B (operator) matrix          ! 5
dof        1 ! Number of unknowns per node                        ! 6
el_homo      ! Element properties are homogeneous                 ! 7
el_real    5 ! Number of real properties per element              ! 8
remarks   13 ! Number of user remarks, e.g. property names        ! 9
convect_coef 1.e-5 ! Convection coefficient on all mixed segments !10
convect_temp    30 ! Convection temperature on all mixed segments !11
el_segment       3 ! Maximum nodes on element boundary segment    !12
mixed_segs       4 ! Number of mixed boundary condition segments  !13
el_types         2 ! Number of different types of elements        !14
type_gauss    1  1 ! Number of Gauss points in each element type  !15
type_nodes    3  3 ! Number of nodes on each element type         !16
type_shape    2  2 ! Shape code of each element type              !17
post_mixed         ! Post-process mixed segments, create n_tape2  !18
end ! Terminate the keyword control, remarks follow               !19
Trapezoidal fin: 160 mm and 60 mm by 120 mm high (no edge conv.)  !20
Units: x,y-mm,  k-W/mm C, h-W/mmˆ2 C, t-C, Q-W/mmˆ3, q-W/mmˆ2     !21
Face convection uses constant convect_coef, convect_temp          !22
Conduction: K_xx U,xx + 2K_xy U,xy + K_yy U,yy + Q = 0            !23
PROP(1) = CONDUCTIVITY K_XX, PROP(2) = CONDUCTIVITY K_YY          !24
PROP(3) = CONDUCTIVITY K_XY, PROP(4) = SOURCE PER UNIT AREA       !25
PROP(5) = THICKNESS (DEFAULT 1.0), here 1.25mm                    !26
Convection: -K_n * U,n = CONVECT_COEF ( U - CONVECT_TEMP)         !27
Optional mixed properties: 1=CONVECT_COEF, 2=CONVECT_TEMP         !28
The conduction reaction total of 57.73 W is equal and opposite    !29
to the convection loss integral of 57.73 W, (actual is about      !30
57.66 W) BUT using element gradients gives 40.34 W for 42 %       !31
flux error.  (Reverse the signs on flux listings.)                !32
 1    1    0.0    0.0 ! node, bc-flag,x, y                       !33
 3    1  160.0    0.0                                            !34
 2    0   50.0  120.0                                            !35
 4    0  110.0  120.0                                            !36
1 1    1 3 4 ! elem, el_type, 3 nodes. conducting                 !37
2 1    2 1 4 ! elem, el_type, 3 nodes. conducting                 !38
1 2    1 3 4 ! face, el_type, 3 nodes. convecting, top            !39
2 2    2 1 4 ! face, el_type, 3 nodes. convecting, top            !40
3 2    1 3 4 ! face, el_type, 3 nodes. convecting, bottom         !41
4 2    2 1 4 ! face, el_type, 3 nodes. convecting, bottom         !42
  1  1  330. ! node, dof, value                                   !43
  3  1  250. ! node, dof, value                                   !44
 1 0.2 0.2 0.0 0.0 1.25 ! el, k_x, k_y, K_xy, Q, thick           !45
```

Figure 10.5.15 Cooling fin element types data


```
BEGINNING STANDARD ELEMENT ASSEMBLY                          ! 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 10.5.16 Selected convecting fin results


```
! ..............................................................    ! 1
!  *** POST_PROCESS_MIXED PROBLEM DEPENDENT STATEMENTS FOLLOW ***   ! 2
! ..............................................................    ! 3
! Define any new array or variable types, then give statements      ! 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 10.5.17 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 10.5.18 Additional convecting fin results

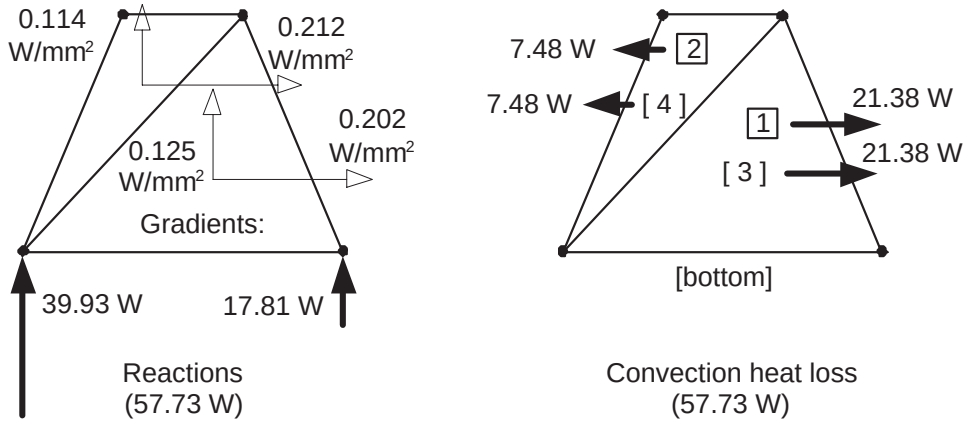


Figure 10.5.19 A cooling fin heat balance

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. 10.4.1). Then the nodal temperatures were gathered and multiplied by $-\mathbf{E}^e\mathbf{B}^e$ to yield the flux vector $\mathbf{q}^e$ 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. 3.37 by summing over each convection (mixed) segment. Such a post-processing implementation for the linear triangle convecting face is given in Fig. 10.5.17. The above flux recovery and the convection losses are given in Fig. 10.5.18. There (in line 13) we see the convection loss matches the thermal reactions of 57.73 W (line 52 of Fig. 10.5.16), as expected. The heat balances and element heat flux vectors for this crude mesh are shown in Fig. 10.5.19.

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. Ther meshes, temperature contours, and heat flux vectors are shown in Fig. 10.5.20. 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-graph in Fig.10.5.21. 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

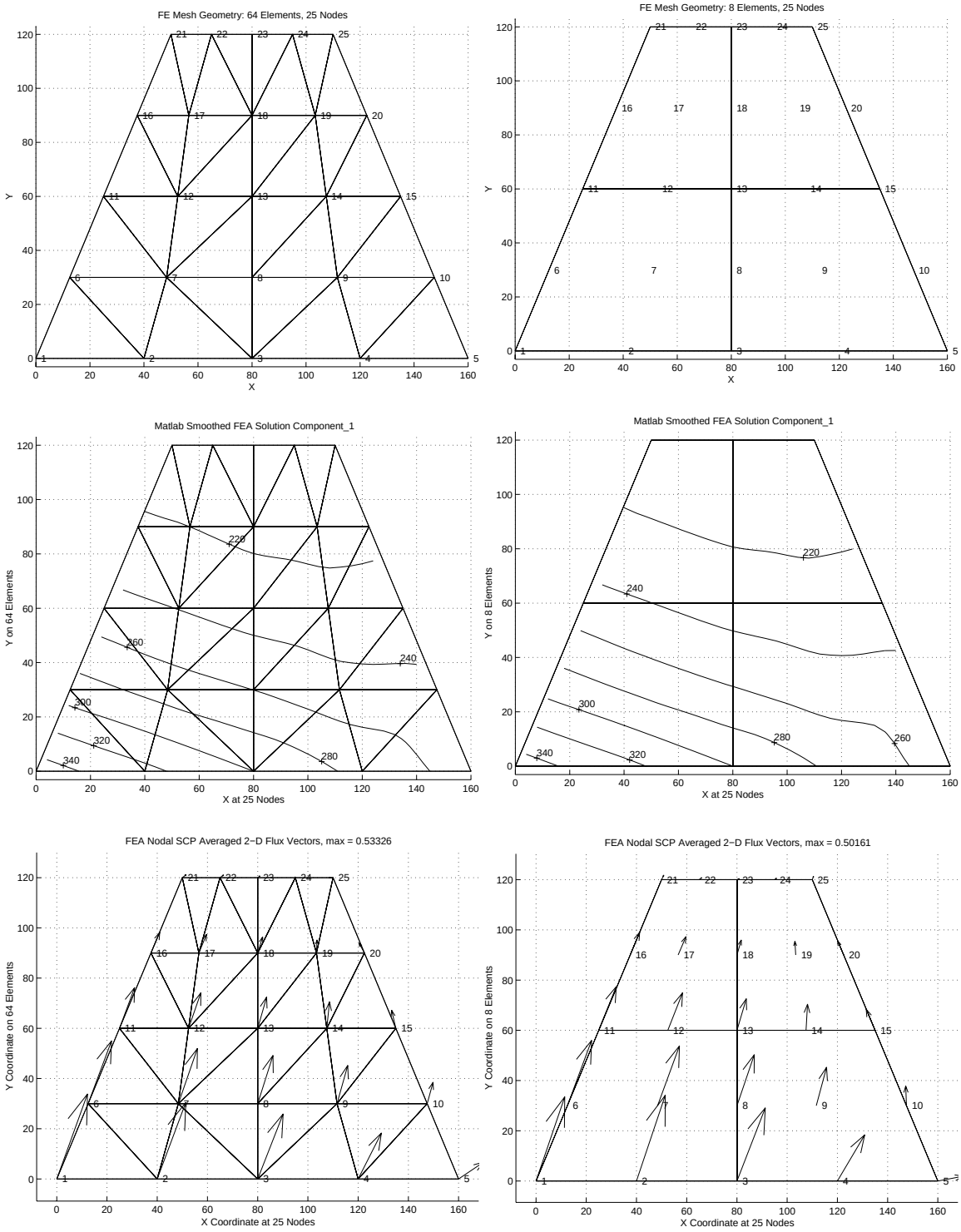


Figure 10.5.20 Mesh, temperature, and flux vectors for the T3 and Q9 fin

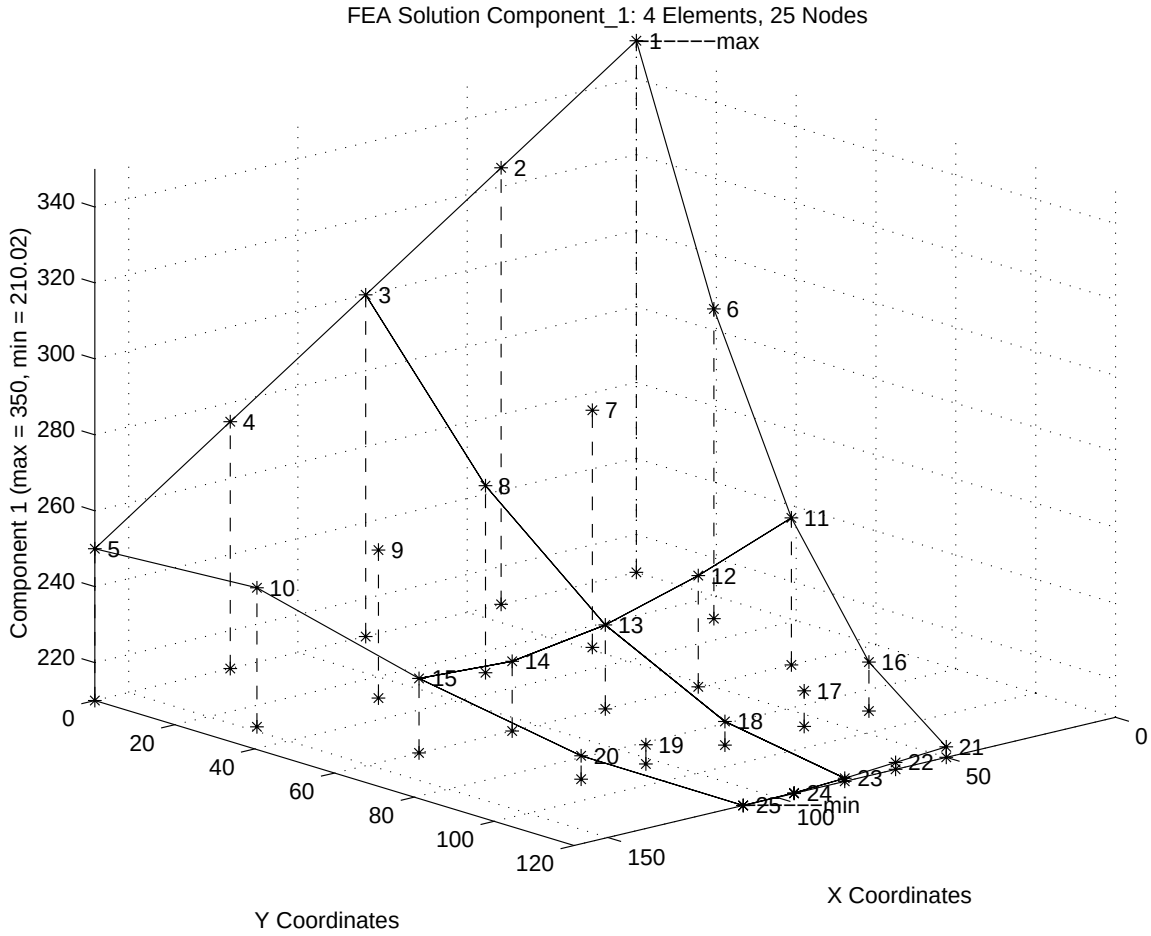


Figure 10.5.21 Carpet-graph of fin temperatures, for Q9 elements


```
title "Fin face convection, Allaire, Four Q9 elements"            ! 1
elems            4 ! Number of elements in the system             ! 2
nodes           25 ! Number of nodes in the mesh                  ! 3
convect_coef 2.e-5 ! Convection coefficient on all mixed segments ! 4
el_segment       9 ! Maximum nodes on element boundary segment    ! 5
mixed_segs       4 ! Number of mixed boundary condition segments  ! 6
type_gauss    9 16 ! Number of Gauss points in each element type  ! 7
type_nodes    9  9 ! Number of nodes on each element type         ! 8
type_shape    3  3 ! Shape code of each element type              ! 9
                                                                  !11
The conduction reaction total of 57.61 W is equal and opposite    !12
to the convection loss integral of 57.61 W.                       !13
Here h is doubled for two face convection on 4 segment.           !14
    1    1    0.0000    0.0000 ! node, bc-flag,x, y              !15
    2    1   40.0000    0.0000                                   !16
                                                                  !18
   24    0   95.0000  120.0000                                   !19
   25    0  110.0000  120.0000 ! last node                        !20
1 1  1  3 13 11  2  8 12  6  7 ! elem, type, 9 nodes. conducting  !21
2 1  3  5 15 13  4 10 14  8  9 ! conducting                       !22
3 1 11 13 23 21 12 18 22 16 17 ! conducting                       !23
4 1 13 15 25 23 14 20 24 18 19 ! conducting                       !24
1 2  1  3 13 11  2  8 12  6  7 ! convecting                       !25
2 2  3  5 15 13  4 10 14  8  9 ! convecting                       !26
3 2 11 13 23 21 12 18 22 16 17 ! convecting                       !27
4 2 13 15 25 23 14 20 24 18 19 ! convecting                       !28
  1  1  3.50000E+02   ! bc along wall                            !29
  2  1  3.25000E+02                                             !30
  3  1  3.00000E+02                                             !31
  4  1  2.75000E+02                                             !32
  5  1  2.50000E+02                                             !33
1  0.2 0.2 0.0 0.0 1.25 ! el, kx, ky, kxy, Q, thick             !34
```

Figure 10.5.22 Data changes for the Q9 model


```
TITLE: "Fin face convection, Allaire, Four Q9 elements"   ! 1
*** MIXED BOUNDARY CONDITION SEGMENTS ***                 ! 3
 NOTE: CONSTANT CONVECTION ON ALL MIXED SEGMENTS USING    ! 4
 CONVECTION COEFFICIENT =    2.00000000000000002E-05      ! 5
 CONVECTION TEMPERATURE =   30.00000000000000000          ! 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
 *** SYSTEM GEOMETRIC PROPERTIES ***                      !13
VOLUME   =  1.32000E+04                                   !14
CENTROID =  8.00000E+01  5.09091E+01                      !15
*** 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
 *** 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
  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 =   57.6082011189374654                            !57
```

Figure 10.5.23 Selected cubic quadrilateral results

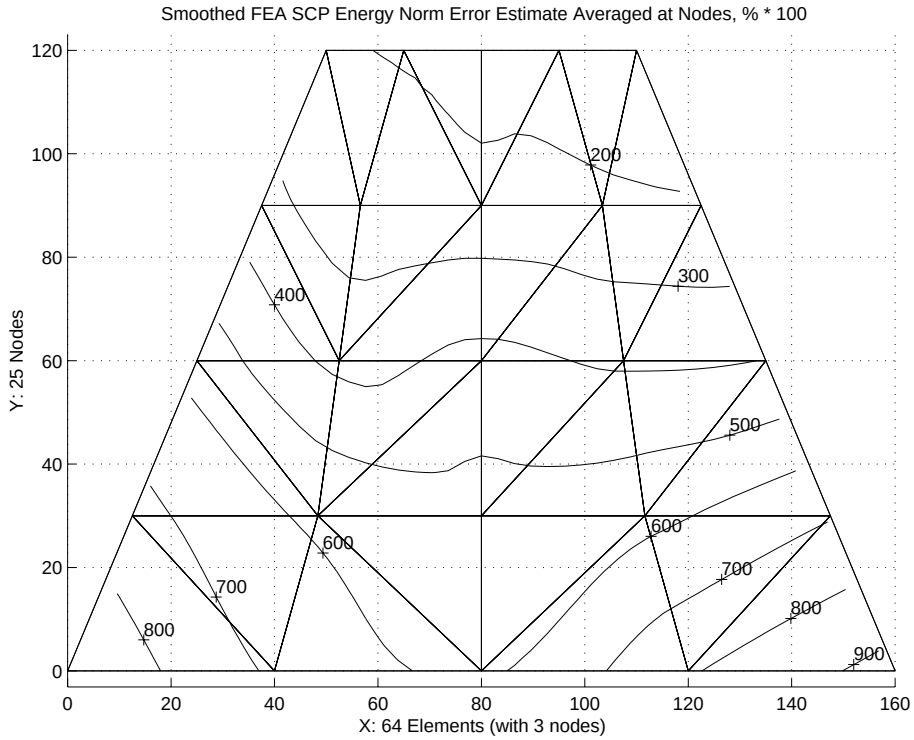


Figure 10.5.24 Error norm levels for the 32 T3 model

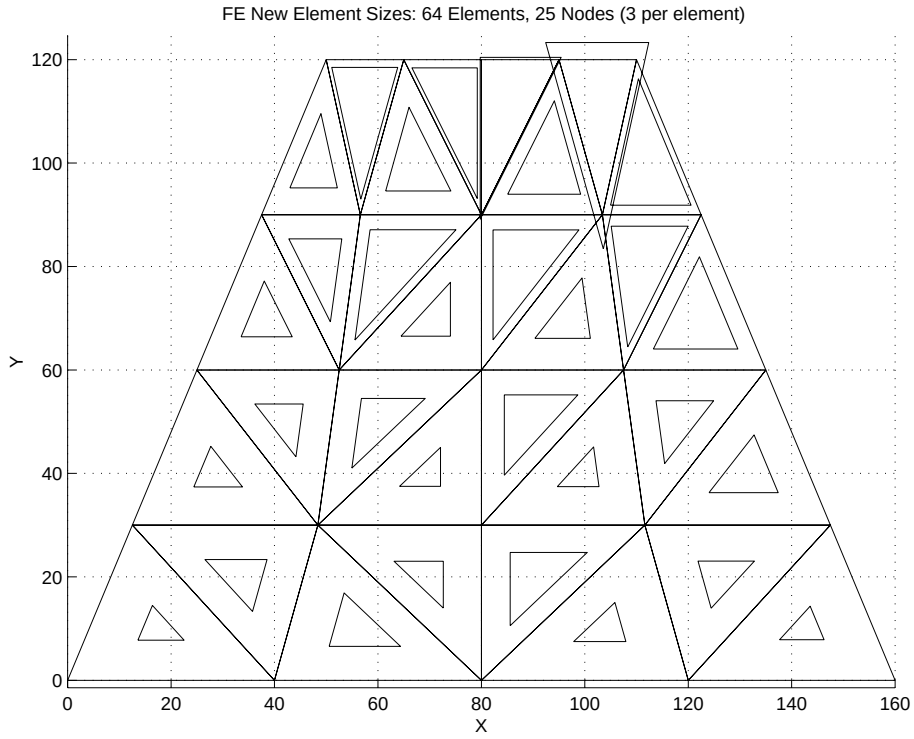


Figure 10.5.25 First computed size changes for 32 T3 model

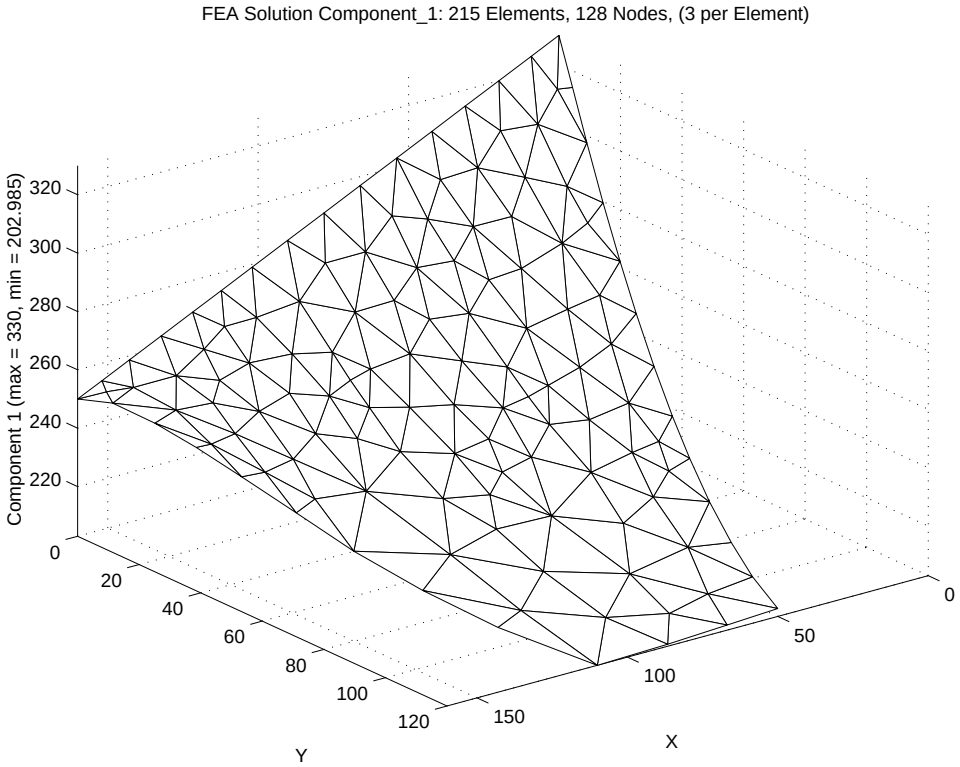


Figure 10.5.26 Solution surface for revised T3 model

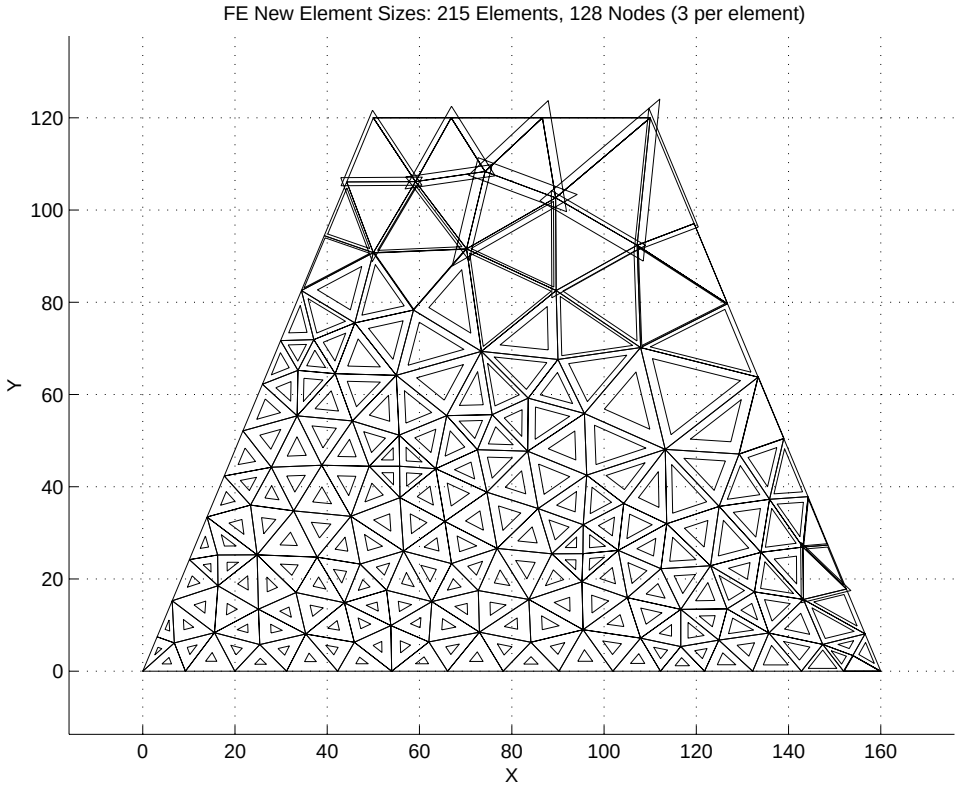


Figure 10.5.27 Second computed size changes for T3 model

changes for this Q9 element calculation are given in Fig. 10.5.22, while selected output results are in Fig. 10.5.23. 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. 10.5.20 (left) it exceeds 9 %, as shown in Fig.10.5.24 so the projected mesh refinement is obtained as illustrated in Fig. 10.5.25. Employing those suggested element sizes to create a new mesh, with 215 T3 elements, one gets the temperature surface shown in Fig. 10.5.26. 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. However, that mesh still has more than a 2.2 % error level so another mesh size adjustment is computed. It is shown in Fig. 10.5.27 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

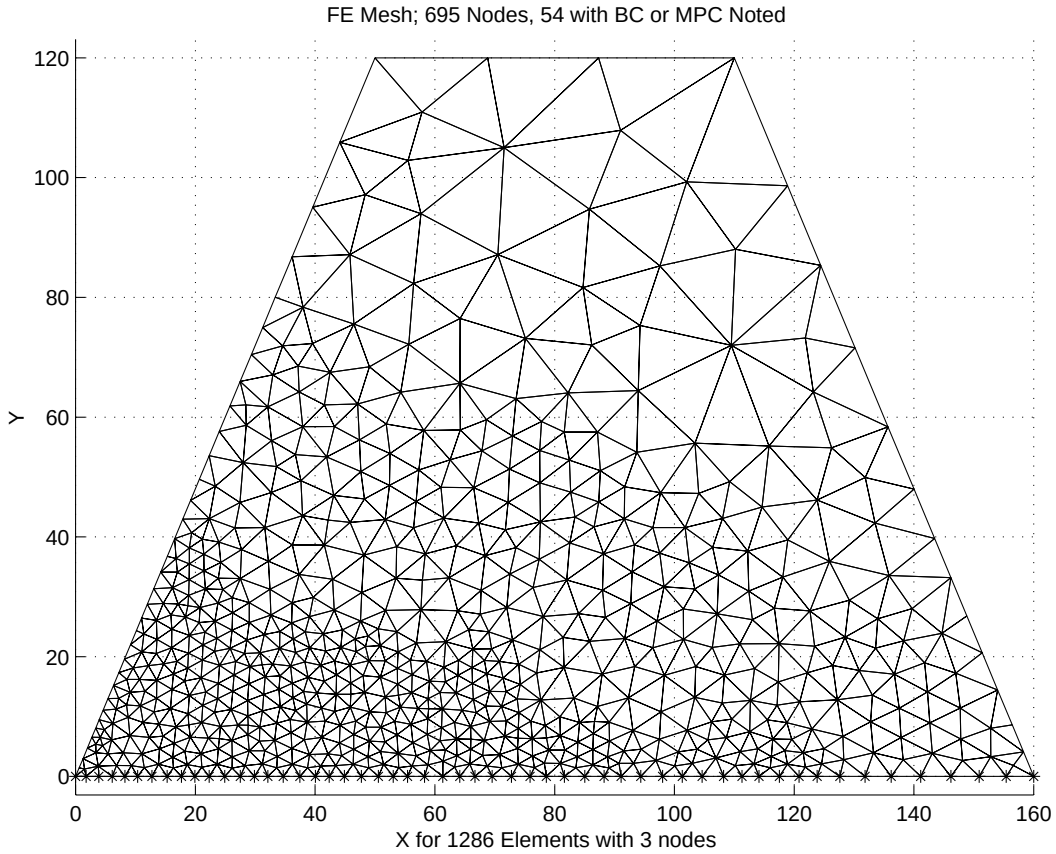


Figure 10.5.28 Final mesh for T3 element model

generation creates a model with 695 nodes and 1286 T3 elements as shown in Fig. 10.5.28. One continues in this way until an acceptable error estimate is obtained.

10.6 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 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. Note that this mapping is simple and gives a constant Jacobian:

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

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

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

$$\mathbf{S}_{x_{i,j}} = \frac{4}{L_x^2}\int_{\Omega^e} k_x^e\,H_{i,a}\,H_{j,a}\;d\,\Omega\,, \tag{10.18}$$

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

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

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

$$H_{i,a}\,H_{j,a} = a_i\,a_j\Big[\,1 + (b_i + b_j)\,b + b_i\,b_j\,b^2\,\Big]/\,16\,. \tag{10.19}$$

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

$$\mathbf{S}_{x_{i,j}} = \frac{a_i\,a_j\,k_x^e\,L_y^e}{12\,L_x^e}\Big[\,3 + b_i\,b_j\,\Big]. \tag{10.20}$$

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

$$\mathbf{S}_x = \frac{k_x^e\, L_y^e}{6\, L_x^e}\begin{bmatrix} 2 & -2 & -1 & 1 \\ -2 & 2 & 1 & -1 \\ -1 & 1 & 2 & -2 \\ 1 & -1 & -2 & 2 \end{bmatrix} \tag{10.21}$$

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

$$\mathbf{S}_y = \frac{k_y^e\, L_x^e}{6\, L_y^e}\begin{bmatrix} 2 & 1 & -1 & -2 \\ 1 & 2 & -2 & -1 \\ -1 & -2 & 2 & 1 \\ -2 & -1 & 1 & 2 \end{bmatrix}. \tag{10.22}$$

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 matrix term is

$$\begin{aligned} \mathbf{M}_{i,j} &= \int_{\Omega^e} \zeta^e\, H_i\, H_j\, d\,\Omega \\ &= \frac{L_x^e\, L_y^e}{64} \int_{\square^e} \zeta^e (1 + a_i\, a)(1 + a_j\, a)(1 + b_i\, b)(1 + b_j\, b)\; d\,\square \end{aligned} \tag{10.23}$$

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

$$\mathbf{M} = \frac{\zeta^e\, L_x^e\, L_y^e}{36}\begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix}. \tag{10.24}$$

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.

10.7 General 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 Chap. 9, Eq. (9.32) becomes

$$\mathbf{S}^e = \int_{A^e} \left(k_x^e \mathbf{d}_x^{e^T} \mathbf{d}_x^e + k_y^e \mathbf{d}_y^{e^T} \mathbf{d}_y^e \right) t^e \, da \ . \tag{10.25}$$

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

$$\mathbf{S}^e = \sum_{i=1}^{\text{NIP}} W_i \ | J^{e_i} | \left(k_{x_i}^e \ \mathbf{d}_{x_i}^{e^T} \ \mathbf{d}_{x_i}^e + k_{y_i}^e \ \mathbf{d}_{y_i}^{e^T} \ \mathbf{d}_{y_i}^e \right) t_i^e \ . \tag{10.26}$$

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:

$$\mathbf{S}^e = \sum_{i=1}^{\text{NIP}} W_i \ | J^{e_i} | \left(\mathbf{B}^{e^T} \mathbf{E}^e \mathbf{B}^e \right) t_i^e \ . \tag{10.27}$$

Similar expressions are available for the source vectors.

10.8 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. 10.8.1 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 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. 10.8.2. There, in line 21, the general parametric measure is computed via function PARM_GEOM_METRIC (in Fig. 8.7.1) 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 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


```
! ..............................................................      ! 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
                                                                      ! 6
 REAL(DP) :: CONST, DET, THICK         ! integration                  ! 7
 REAL(DP) :: SOURCE                    ! data                         ! 8
 INTEGER  :: IP                        ! counter                      ! 9
                                                                      !10
! K_xx U,xx + 2K_xy U,xy + K_yy U,yy + Q = 0                          !11
                                                                      !12
  SOURCE = 0 ; THICK = 1                ! Initialize                  !13
  CALL POISSON_ANISOTROPIC_E_MATRIX (E) ! for 1-, 2-, or 3-D          !14
                                                                      !15
!      STORE NUMBER OF POINTS FOR FLUX CALCULATIONS                   !16
  CALL STORE_FLUX_POINT_COUNT ! Save LT_QP                            !17
                                                                      !18
!-->   NUMERICAL INTEGRATION LOOP                                     !19
   DO IP = 1, LT_QP                                                   !20
     H   = GET_H_AT_QP (IP)    ! EVALUATE INTERPOLATION FUNCTIONS     !21
     XYZ = MATMUL (H, COORD)   ! FIND GLOBAL COORD, ISOPARAMETRIC     !22
     DLH = GET_DLH_AT_QP (IP)  ! FIND LOCAL DERIVATIVES               !23
     AJ  = MATMUL (DLH, COORD) ! FIND JACOBIAN AT THE PT              !24
!         FORM INVERSE AND DETERMINATE OF JACOBIAN                    !25
     CALL INVERT_JACOBIAN (AJ, AJ_INV, DET, N_SPACE)                  !26
     IF ( AXISYMMETRIC ) THICK = TWO_PI * XYZ (1) ! via axisymmetric  !27
     CONST  = DET * WT(IP) * THICK                                    !28
                                                                      !29
!         EVALUATE GLOBAL DERIVATIVES, DGH == B                       !30
     DGH = MATMUL (AJ_INV, DLH)         ! Physical gradient           !31
     B   = COPY_DGH_INTO_B_MATRIX (DGH) ! B = DGH                     !32
                                                                      !33
!      VARIABLE VOLUMETRIC SOURCE, via keyword use_exact_source       !34
!      Defaults to file my_exact_source_inc if no exact_case key      !35
     IF ( USE_EXACT_SOURCE )  CALL        & ! analytic Q              !36
          SELECT_EXACT_SOURCE (XYZ, SOURCE) ! via exact_case key      !37
     C = C + CONST * SOURCE * H         ! source resultant            !38
                                                                      !39
!      CONDUCTION SQUARE MATRIX (THICKNESS IN E)                      !40
     S = S + CONST * MATMUL ((MATMUL (TRANSPOSE (B), E)), B)          !41
                                                                      !42
!-->  SAVE COORDS, E AND DERIVATIVE MATRIX, FOR POST PROCESSING       !43
     CALL STORE_FLUX_POINT_DATA (XYZ, (E * THICK), B)                 !44
   END DO                                                             !45
!    ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***         !46
```

Figure 10.8.1 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
 INTEGER  :: IQ                    ! loops                    ! 6
 REAL(DP) :: CONST, DET, THICK     ! flux area                ! 7
                                                              ! 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)                                !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 10.8.2 Computing a Neumann flux contribution

degenerate to scalars) as listed in Fig. 10.8.3.

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 Fig.s 10.4.1-3. 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 also allow for any point, 1-, 2-, or 3-dimensional element in the library to be utilized.


```
! .....................................................          ! 1
! ***  MIXED_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***   ! 2
! .....................................................          ! 3
! Mixed or Robin boundary condition, Standard form:             ! 4
!  K_n * U,n + ROBIN_1_SEG * U + ROBIN_2_SEG = 0, set in        ! 5
!  keywords robin_square, optional robin_column, robin_thick    ! 6
 INTEGER  :: IQ                ! Integration loop               ! 7
 REAL(DP) :: CONST, DET, THICK ! face area                      ! 8
                                                               ! 9
! GET ROBIN TERMS, IF GLOBAL CONSTANTS: Set in keywords        !10
! robin_square, robin_column, robin_thick, or via convect_coef,!11
! convect_temp, convect_thick for convection special case      !12
  THICK = 1.d0                          ! Default for all cases !14
  IF ( ROBIN_THICK /= 1.d0 ) THICK = ROBIN_THICK ! from keyword !14
  IF ( CONVECTION ) THEN     ! constant convection all segments !15
    ROBIN_1_SEG =  CONVECT_COEF                                 !16
    ROBIN_2_SEG =  CONVECT_COEF * CONVECT_TEMP                  !17
    IF ( LT_PARM <= 1 ) THICK = CONVECT_THICK ! point, line only !17
  END IF ! Convection                                           !18
                                                                !19
 !  GET ROBIN TERMS, IF LOCAL MIXED SEGMENT CONSTANTS           !20
  IF ( MIXED_REAL > 0 ) THEN ! are local data                   !21
    ROBIN_1_SEG = GET_REAL_MX (1)                               !22
    ROBIN_2_SEG = GET_REAL_MX (2)                               !23
    IF ( MIXED_REAL > 2 ) THICK = GET_REAL_MX (3)               !24
  END IF ! local data                                           !25
                                                                !26
  IF ( LT_N > 1 ) THEN ! Not a point, must integrate            !27
                       ! this element type (LT)                !28
    DO IQ = 1, LT_QP           ! NUMERICAL INTEGRATION LOOP     !29
                                                                !30
      H   = GET_H_AT_QP (IQ)   ! BOUNDARY INTERPOLATION FUNCTIONS !31
      XYZ = MATMUL (H, COORD)  ! FIND GLOBAL COORD, (ISOPARAMETRIC) !32
      DLH = GET_DLH_AT_QP (IQ) ! FIND LOCAL DERIVATIVES         !33
                                                                !34
!        FORM DETERMINATE OF GENERALIZED JACOBIAN, Fig 5.4.2    !35
      DET   = PARM_GEOM_METRIC (DLH, COORD) ! dX / dr           !36
      IF ( AXISYMMETRIC ) THICK = TWO_PI * XYZ (1) ! keyword    !37
      CONST = DET * WT(IQ) * THICK ! net "surface" area         !38
                                                                !39
!       GET ROBIN DATA, if analytic, via use_exact_robin keyword !40
      IF ( USE_EXACT_ROBIN ) CALL SELECT_EXACT_ROBIN_DATA  &    !41
                             (XYZ, ROBIN_1_SEG, ROBIN_2_SEG)    !42
                                                                !43
!       MIXED SQUARE AND COLUMN TERMS                           !44
      S = S + ROBIN_1_SEG * CONST * OUTER_PRODUCT (H, H) ! Square !45
      C = C + ROBIN_2_SEG * CONST * H                    ! Source !47
                                                                !48
    END DO                                                      !49
                                                                !50
  ELSE ! This is a point value, maybe analytic expression       !51
                                                                !52
    IF ( USE_EXACT_ROBIN ) CALL SELECT_EXACT_ROBIN_DATA  &      !53
         (COORD (1, :), ROBIN_1_SEG, ROBIN_2_SEG) ! use_exact_robin !54
    IF ( AXISYMMETRIC ) THICK = TWO_PI * COORD (1, 1)           !55
    S (1, 1) = ROBIN_1_SEG * THICK       ! Sq matrix            !56
    C (1)    = ROBIN_2_SEG * THICK       ! Source vec           !57
                                                                !58
  END IF ! boundary segment type: point, line, or surface       !59
```

Figure 10.8.3 General Robin or convection contributions

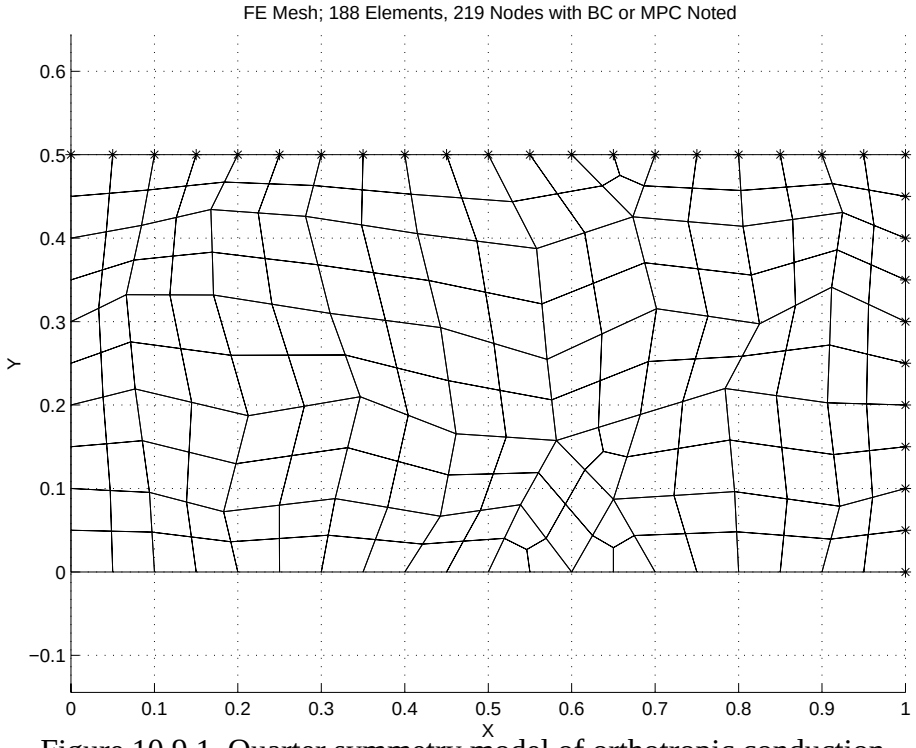


Figure 10.9.1 Quarter symmetry model of orthotropic conduction

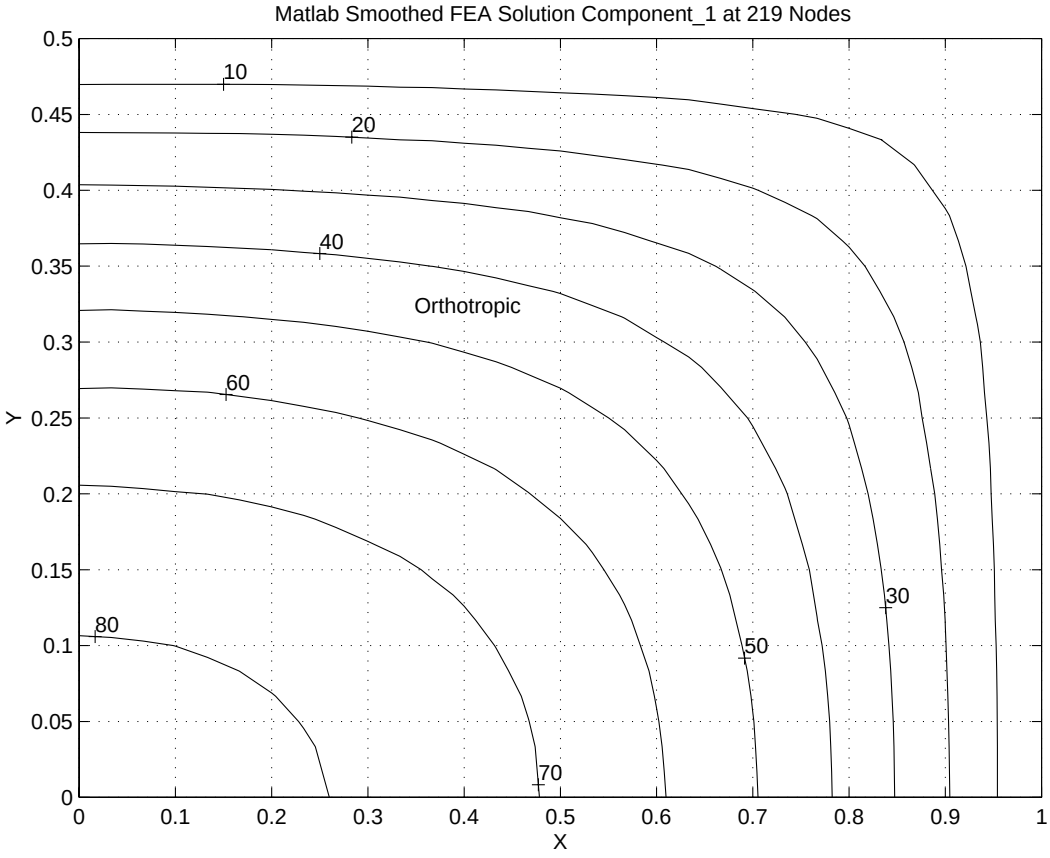


Figure 10.9.2 Temperature, $K_x = 2, K_y = 1.237, Q = 1000$

10.9 Orthotropic Conduction

We have seen that the finite element method automatically includes the ability to have directionally dependent (anisotropic) material properties. However we have not illustrated how much such properties can influence a solution. Carslaw and Jaeger [2] have given an exact infinite series solution for an orthotropic rectangular region (*exact_case* 19) with a constant internal heat source, $Q = 1000$, and with a temperature of zero around its edge. Consider a material that has thermal conductivity values of $K_x = 2.0, K_y = 1.2337$, and $K_{xy} = 0$. The problem has symmetry of the geometry, boundary conditions, source term, and conductivities with respect to both the y-axis and the x-axis. That is, one can employ a quarter symmetry model to study this problem. The top right segment is shown with a mesh of Q4 quadrilateral elements in Fig. 10.9.1. It also has asterisk marking those nodes with the null essential boundary conditions. For a region that has full lengths of 2.0 and 1.0, in the x- and y-directions, respectively, the analytic solution at the center point, (0, 0), is 83.72 degrees. The above mesh yielded a corresponding value of 83.77 degrees. That differed significantly from the 70.31 degrees that is computed if one assumes an isotropic material with a $K = 1.617$ value which is the average of the above orthotropic conductivities. Had the orthotropic conductivities been reversed (largest in the y-direction) then the center temperature would be 60.13 degrees so we see about a 30 % variation in the peak temperature as various principle material directions are considered in our study. The temperature contours for these three cases are shown in Figs.10.9.2-4.

The exact flu vectors for the isotropic material are shown in Fig. 10.9.5. The FEA vectors obtained from the SCP post-averaging look almost identical so they are not plotted. To emphasize the difference between the FEA and exact fluxes the differences in the two sets of vectors are shown in Fig. 10.9.6, to a different scale. The biggest differences occur along the edges and symmetry planes and at the top right corner where both flux vector components are approaching zero. We can see the magnitude of the heat flow in the exact and SCP flux contour plots in Figs. 10.9.7 and 8, respectively. Even with this crude mesh the agreement in the isotropic heat flow is quite good. Condidering the two orthotropic material choices we see that the heat flow changes relatively little, as Figs. 10.9.9 and 10 illustrate. The difference between the exact energy norm error and the SCP estimate is also quite close, as will be discussed in Chap. 12. The estimated new mesh sizes do not change much here with the small deviations from isotropic properties. The isotropic case suggests a relatively uniform mesh refinement as seen in the suggested new grid in Fig. 10.9.11.

An interesting question is how well dose the energy norm error estimate compare to the exact energy norm error, and how does the exact energy norm error compare to the exact error in the computed primary variable. We will illustrate the answers for the orthotropic case where $K_x > K_y$. For this crude mesh there is little correlation between the peak error in the solution value (Fig. 10.9.12) and the peak error in the energy norm. Figures 10.9.13 and 14 show the exact and SCP estimated error as measured in the energy norm. The agreement is both the relative location of the peak error and the values of the local errors are unusually good.

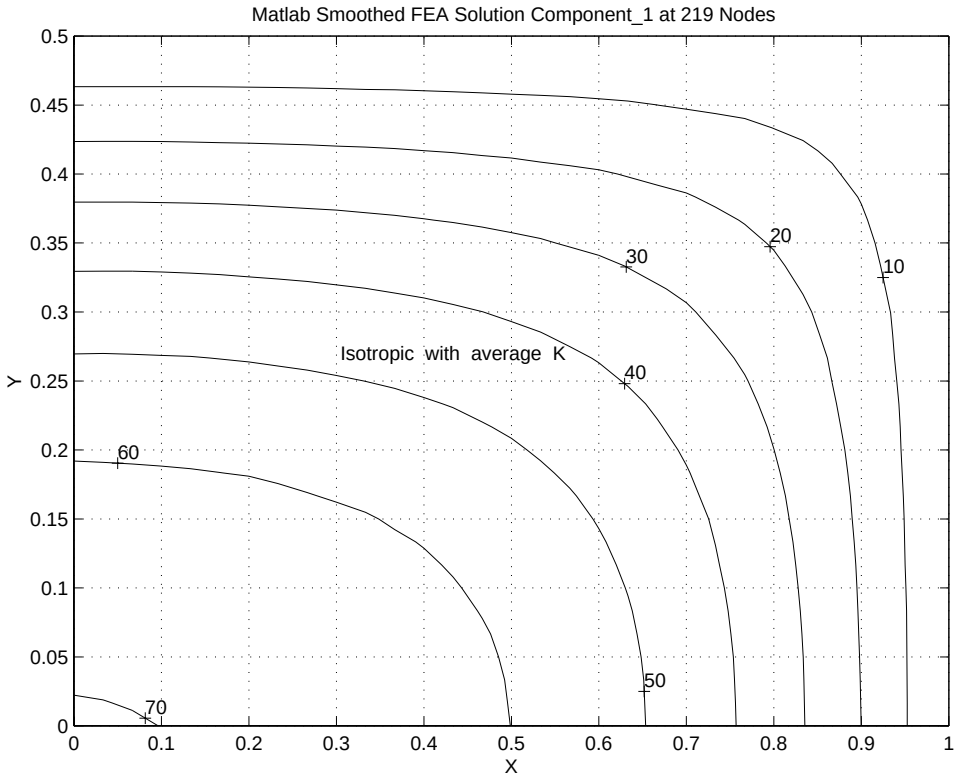


Figure 10.9.3 Temperature, $K_x = K_y = 1.617, Q = 1000$

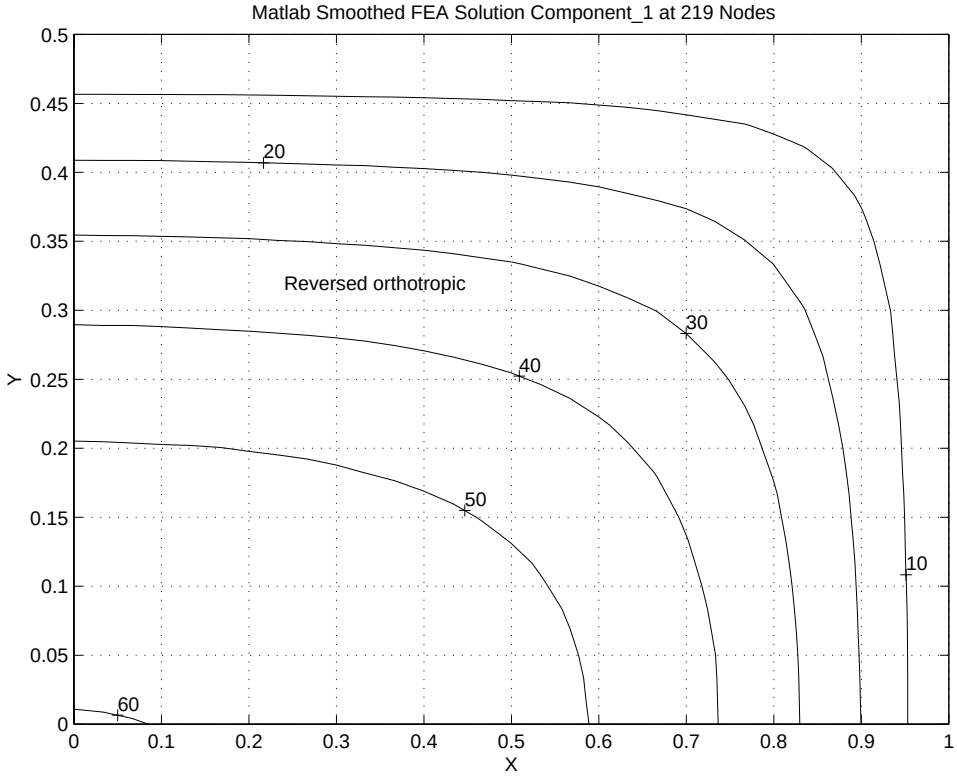


Figure 10.9.4 Temperature, $K_x = 1.2337, K_y = 2, Q = 1000$

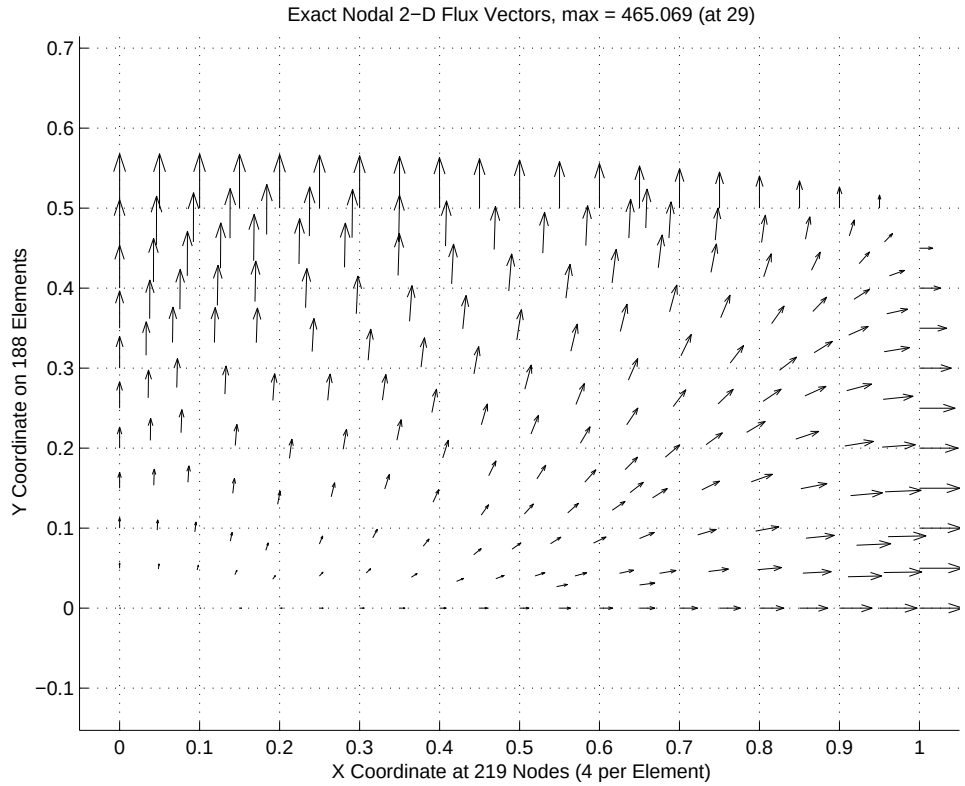


Figure 10.9.5 Exact isotropic heat flux vectors

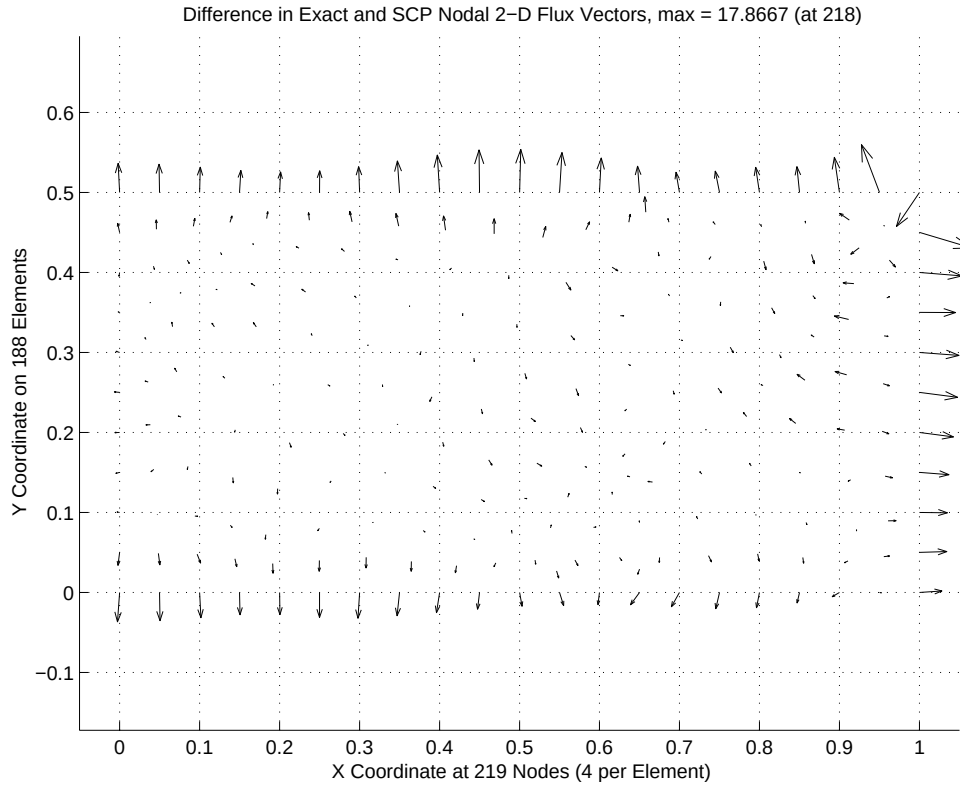


Figure 10.9.6 Differences in exact and SCP average flux vectors

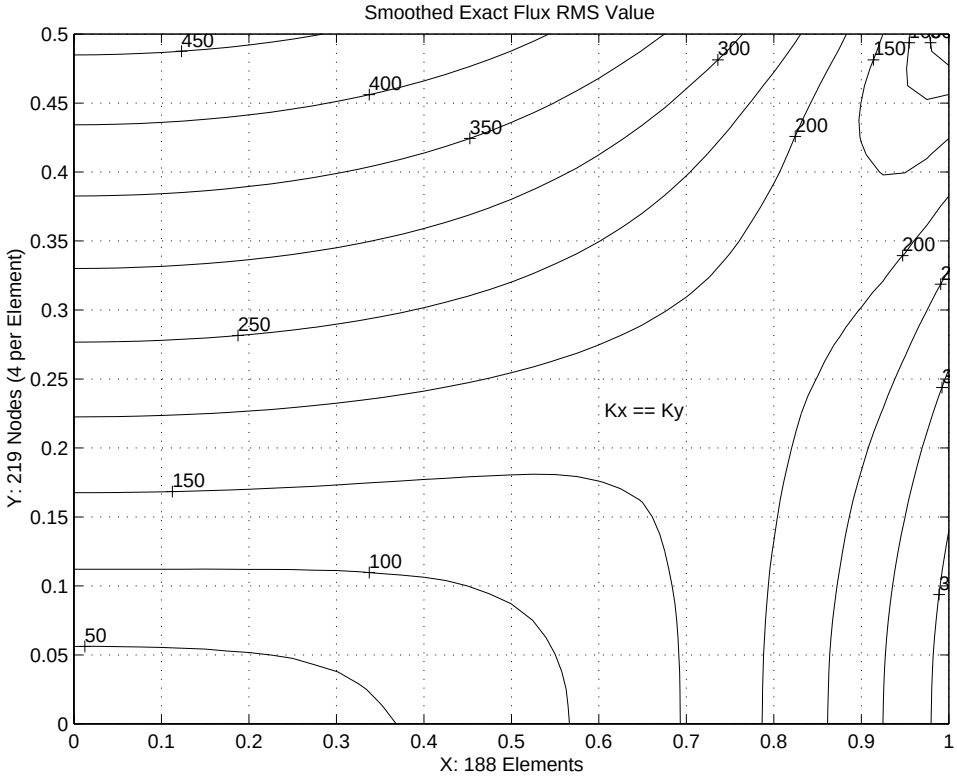


Figure 10.9.7 Exact isotropic heat flux value contours

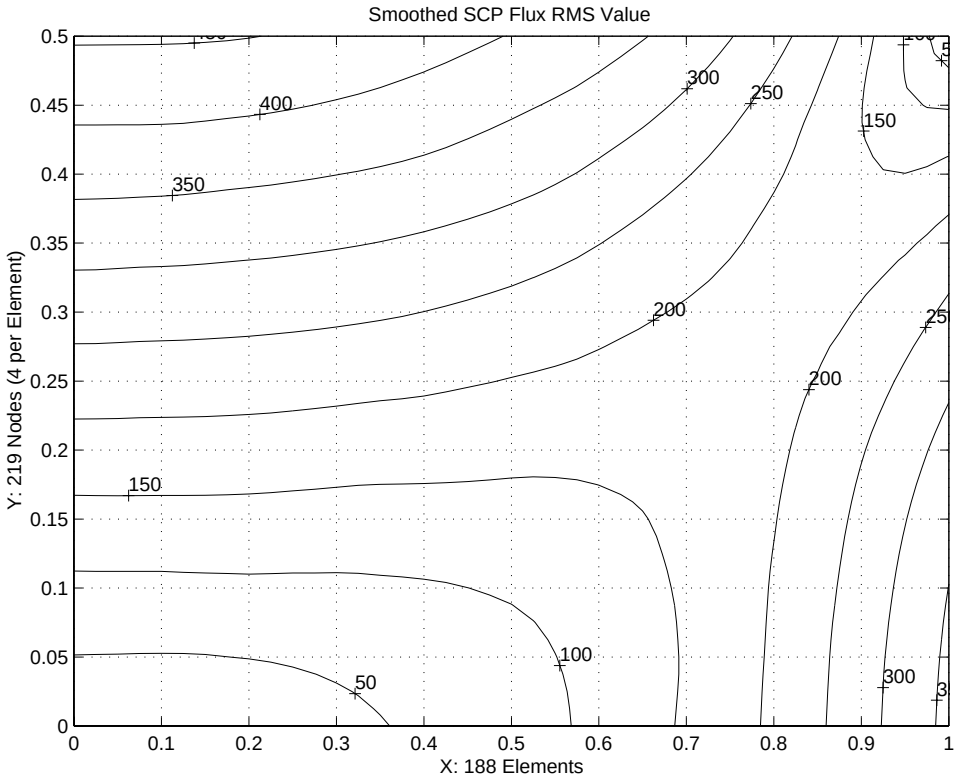


Figure 10.9.8 SCP averaged isotropic heat flux value contours

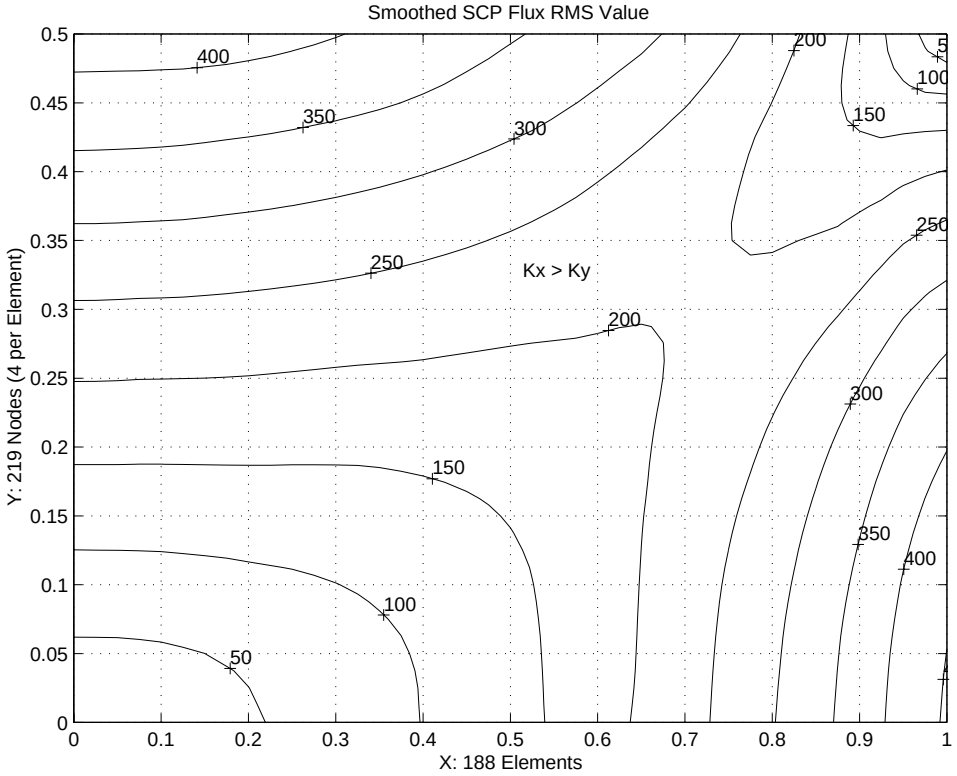


Figure 10.9.9 SCP averaged *Kx* > *Ky* heat flux value contours

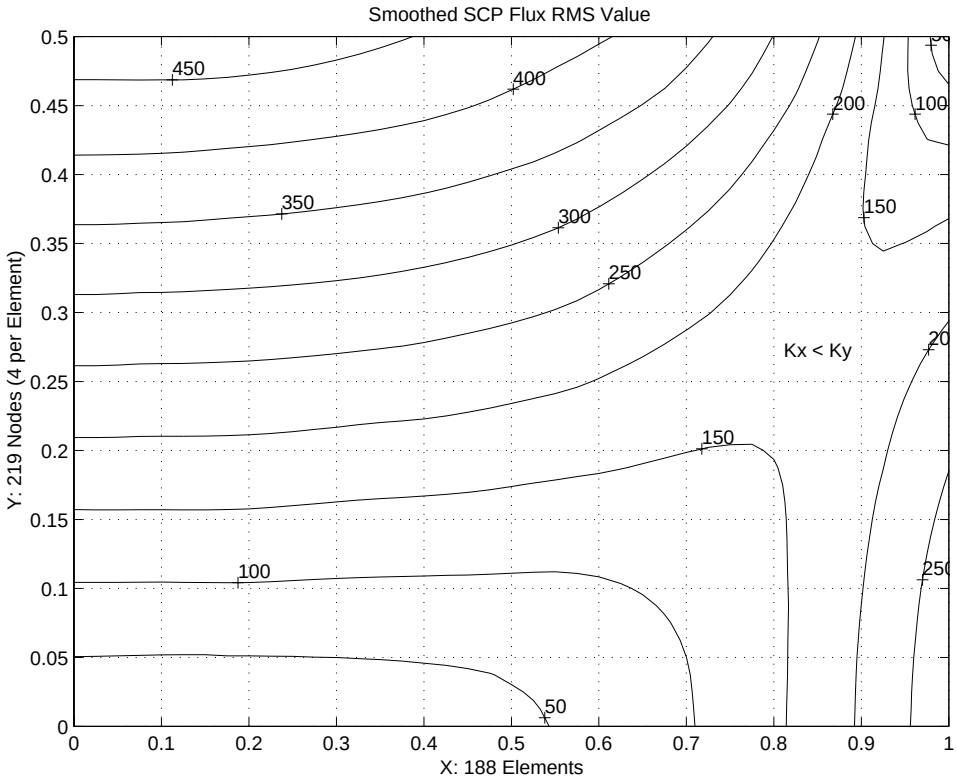


Figure 10.9.10 SCP averaged *Kx* < *Ky* heat flux value contours

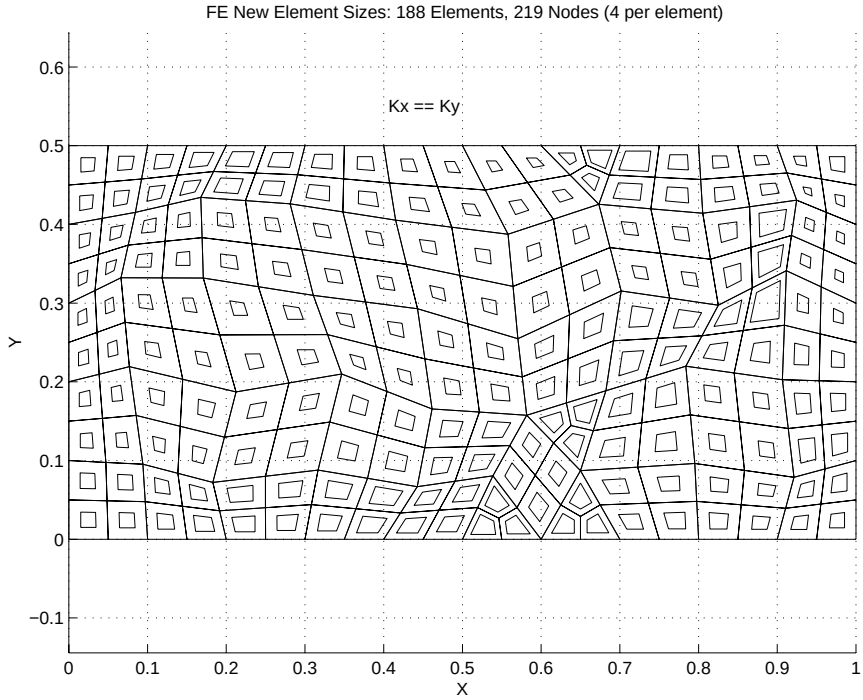


Figure 10.9.11 Suggested new mesh for isotropic material

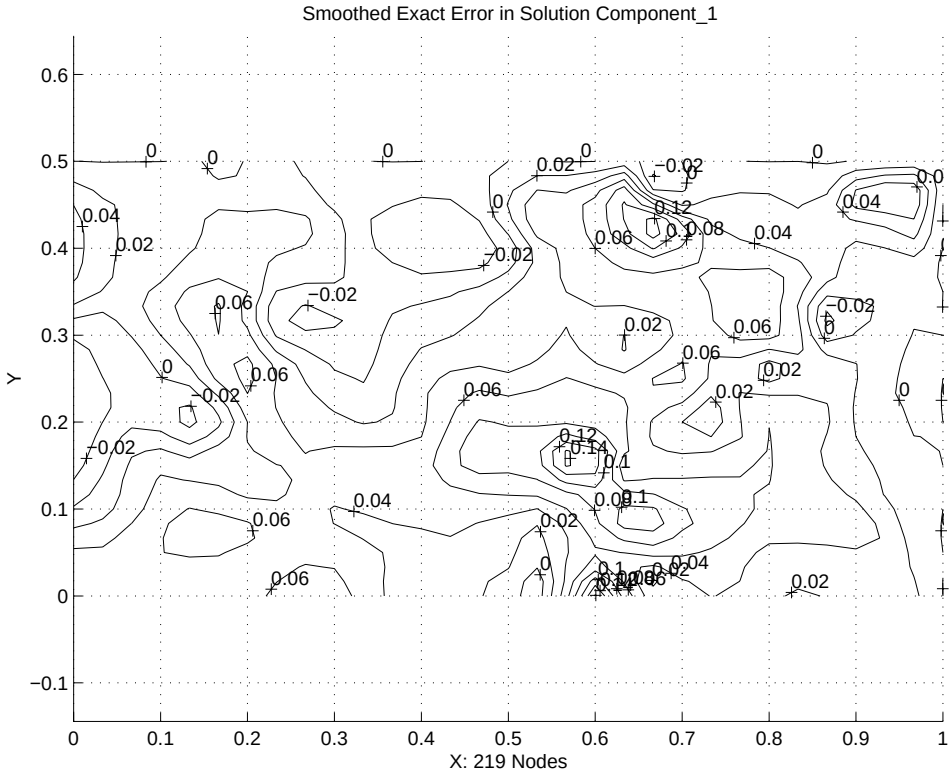


Figure 10.9.12 Exact error in the solution value, *Kx < Ky*

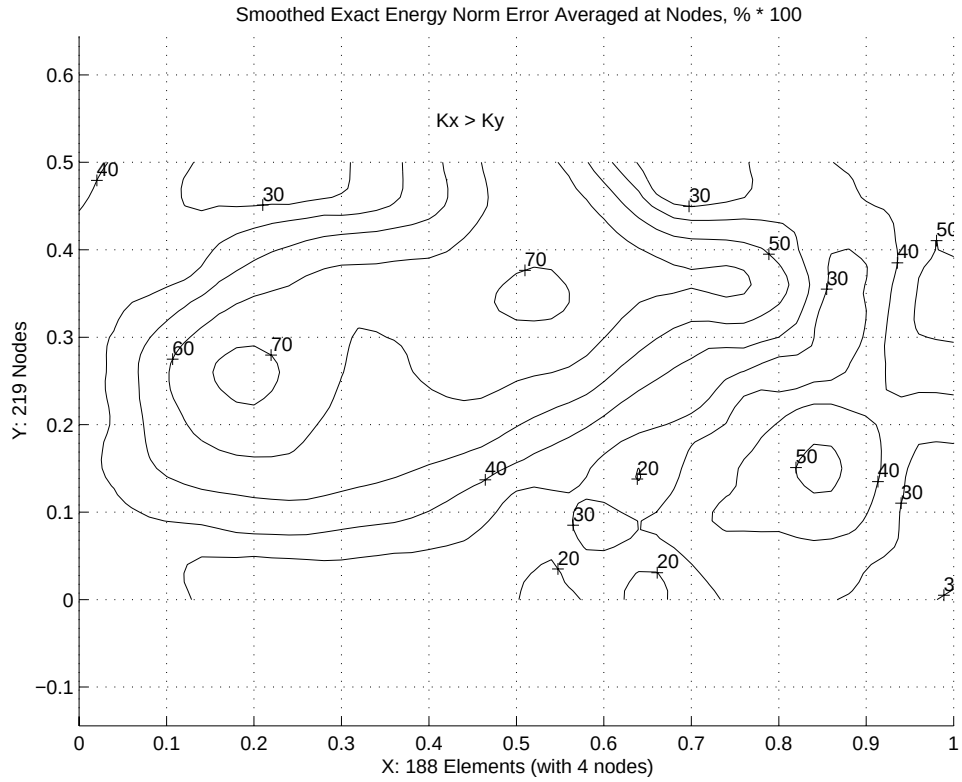


Figure 10.9.13 Exact error in the energy norm, *Kx < Ky*

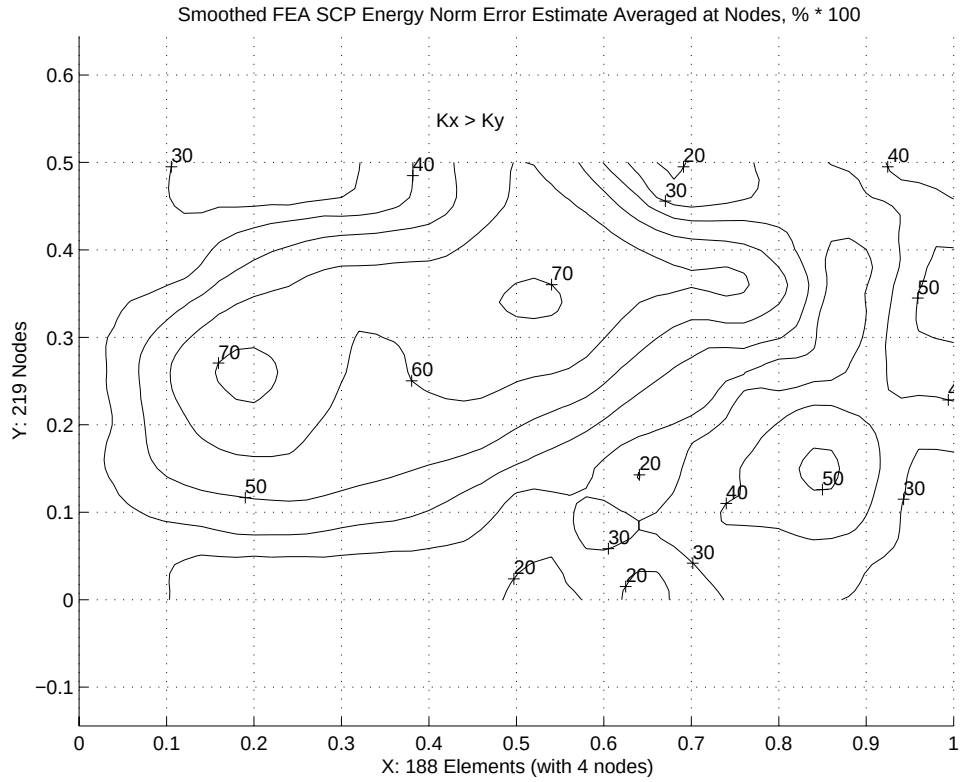


Figure 10.9.14 SCP estimated error in the energy norm, *Kx < Ky*

10.10 Axisymmetric Formulations

For a general steady state orthotropic axisymmetric material we just recast the governing differential equation, Eq. 10.2, into cylindrical coordinates:

$$\frac{1}{r} k_r \frac{\partial T}{\partial r} + k_r \frac{\partial^2 T}{\partial r^2} + k_z \frac{\partial^2 T}{\partial z^2} + Q = 0$$

or re-arranging

$$\frac{\partial}{\partial r}(k_r r \frac{\partial T}{\partial r}) + \frac{\partial}{\partial z}(k_z r \frac{\partial T}{\partial z}) + Q r = 0$$

Comparing this form to Eq. 10.2 we see that one difference is that one could consider substituting the values $(k_r r)$, $(k_z r)$, and $(Q r)$ as modified conductivities and source term in the previous formulation. That is, material constants would now become spatially varying, but that is no problem to include in a numerically integrated version like those given in the previous section. However, the change in coordinates also means that we must change the differential volume $d\Omega$, to $2\pi r\, dr\, dz$. Some analysts like to use a one radian segment rather that the full 2π body. When specifying resultant point (ring) flux data you need to know which basis is being employed. Carrying out the usual Galerkin process the integral definitions of the element and segment arrays are almost identical to those in Eq.s 10.9-11, expect that x and y are replaced by r and z, respectively, while the differential volume and surface areas change from $t^e\, da$ and $t^b\, ds$ to

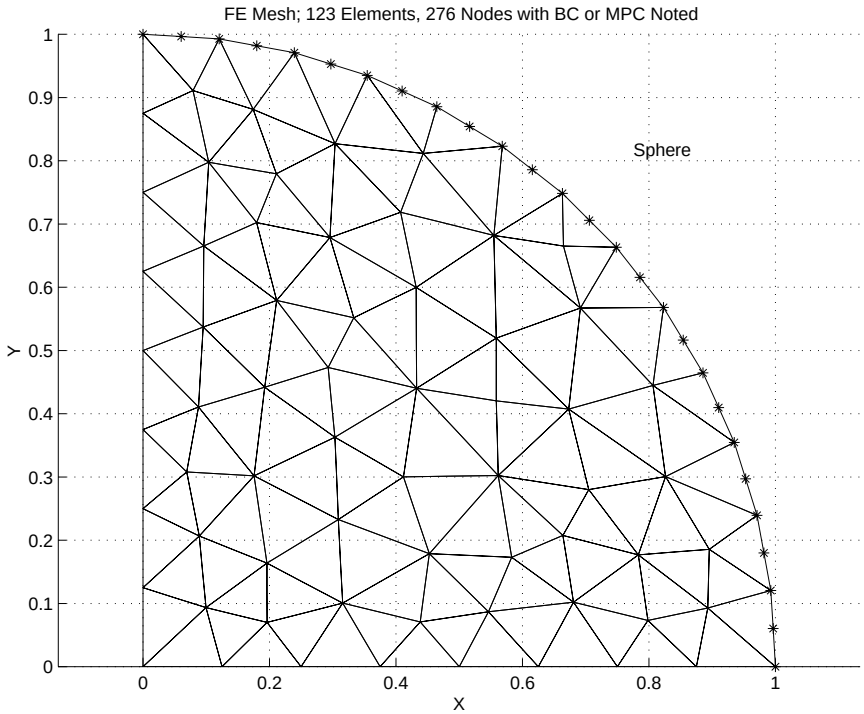


Figure 10.10.1 Half sphere model and boundary condition nodes

the corresponding axisymmetric measures of $2\pi\, r\, da$ and $2\pi\, r\, ds$. Namely,

$$\mathbf{S}^e = \int_{A^e} (k_r^e\, \mathbf{H}_r^{e^T} \mathbf{H}_r^e + k_z^e\, \mathbf{H}_z^{e^T} \mathbf{H}_z^e)\, 2\pi\, r\, da$$

$$\mathbf{C}_Q^e = \int_{A^e} \mathbf{H}^{e^T} Q^e 2\pi\, r\, da$$

$$\mathbf{S}_h^b = \int_{\Gamma^b} h^b \mathbf{H}^{b^T} \mathbf{H}^b 2\pi\, r\, ds, \quad \mathbf{C}_h^b = \int_{\Gamma^b} T_r^b h^b \mathbf{H}^{b^T} 2\pi\, r\, ds$$

$$\mathbf{C}_q^b = \int_{\Gamma^b} q^b \mathbf{H}^{b^T} 2\pi\, r\, ds$$

In other words we could use the previous formulations, but now require the thickness to be $t = 2\pi\, r$ at any point. Then we need to include options in the previous numerical integration coding in Fig.s 10.8.1-3 by inserting a new line in the numerical integration loop, after forming the Jacobian, such as:

```
IF ( AXISYMMETRIC ) THICK = TWO_PI * XYZ (1) ! via key axisymmetric  !27
```

where TWO_PI is a global program parameter, and AXISYMMETRIC is a global logical variable that is true only if the keyword 'axisymmetric' appears in the control data. Then the r coordinate is the first component of the coordinates of the point, XYZ (1).

As an axisymmetric heat transfer example we will consider the temperature of a solid homogeneous unit sphere where the temperature on the surface is unity at the north and south poles and decreases to zero at the equator. We employ a half-symmetry model as shown in Fig. 10.10.1. The mesh was created by an automatic mesh generator as the first step in an adaptive analysis to be considered later. The mesh consists of quadratic triangles with six nodes (the T6 element). They have curved edges where they approximate the surface of the sphere. The conduction matrix involves the products of the gradients (linear polynomials) and the radius, which is varying at least in a linear fashion (for straight-sided elements) or quadratically in general. Thus the square matrix is at least a cubic polynomial. From Table 9.3, we see that a cubic polynomial requires four quadrature points. Near the surface the Jacobian is not constant so a seven-point rule is not unreasonable.

Portions of the input data are shown in Fig. 10.10.2 where the main new control item is the word "axisymmetric" that flags the need for an extension of the integration rules in forming the conduction matrix, and the domain geometry properties. The exact result is known, exact_case 22, and is used for comparison purposes in the output list and plots. The computed temperatures agree with the exact values to several significant figures. Selected output results are given in Fig. 10.10.3. Note that the volume, provided as a data checking aid, is in error by less than one percent. It is inexact because we have approximated a circular arc by eight parabolic (three-noded) segments. Carpet plots of the two solutions are given in Figs. 10.10.4-5. The error estimator, to be considered later, suggests a new size for each element. They are plotted on top of the original mesh in Fig. 10.10.6.


```
title "Kreyszig unit sphere with EBC'                                  ! 1
exact_case 22 ! Exact analytic solution                                ! 2
axisymmetric  ! Problem is axisymmetric, x radius, y C/L               ! 3
b_rows      2 ! Number of rows in the B (operator) matrix              ! 4
dof         1 ! Number of unknowns per node                            ! 5
el_nodes    6 ! Maximum number of nodes per element                    ! 6
elems     123 ! Number of elements in the system                       ! 7
gauss       7 ! Maximum number of quadrature points                    ! 8
nodes     276 ! Number of nodes in the mesh                            ! 9
shape       2 ! Element shape, 1=line, 2=tri, 3=quad, 4=hex            !10
space       2 ! Solution space dimension                               !11
el_homo       ! Element properties are homogeneous                     !12
el_real     4 ! Number of real properties per element                  !13
pt_list       ! List the answers at each node point                    !14
list_exact    ! List given exact answers at nodes, etc                 !15
save_new_mesh ! Save new element sizes for adaptive mesher             !16
remarks     3 ! Number of user remarks, e.g. property names            !17
end ! Terminate the keyword control inputs, remarks follow             !18
The surface temperature BC is T=cosine(angle_from_Z)^2                 !19
so it varies from 1 to zero. R_max = 1 = Z_max.                        !20
T_exact=(R^2 + Z^2)*( cos(ang)^2 - third) + third                      !21
  1     1  6.02683E-02  9.96354E-01 ! node bc_flag, R, Z               !22
  2     1  0.00000E+00  1.00000E+00                                    !23
  3     0  3.95504E-02  9.55542E-01                                    !24
    ...                                                                !25
276     0  8.12500E-01  0.00000E+00 ! last node                        !26
  1   272   259   262   261   260   264 ! elem, 6 nodes                !27
  2   274   272   271   276   273   275                                !28
  3   268   274   271   270   275   269                                !29
    ...                                                                !30
123    21     6    10     7     5    11 ! last elem                    !31
  2      1   1.00000E+00 ! node, dof, exact bc value                   !32
  1      1   9.93937E-01                                              !33
    ...                                                                !34
259      1   1.00000E+00 ! last EBC                                    !35
1 1. 1. 0. 0. ! el kr kz krz Q                                         !36
```

Figure 10.10.2 Partial sphere input data


```
 TITLE: "Kreyszig unit sphere with EBC"                          ! 1
                                                                 ! 2
****  OPTIONS: (DEFAULT) VALUE ****                             ! 3
AXISYMMETRIC DOMAIN:         0=FALSE, 1=TRUE ..(0)    1         ! 4
                                                                 ! 5
*** SYSTEM GEOMETRIC PROPERTIES ***                              ! 6
VOLUME   =  2.08676E+00                                          ! 7
CENTROID =  5.88333E-01  3.74543E-01                             ! 8
                                                                 ! 9
*** OUTPUT OF RESULTS AND EXACT VALUES IN NODAL ORDER ***        !10
NODE,  Radius r,   Axial z,    DOF_1,      EXACT1,               !11
   1  6.0268E-02  9.9635E-01  9.9394E-01  9.9394E-01             !12
   2  0.0000E+00  1.0000E+00  1.0000E+00  1.0000E+00             !13
   3  3.9550E-02  9.5554E-01  9.4152E-01  9.4152E-01             !14
   4  0.0000E+00  9.3750E-01  9.1927E-01  9.1927E-01             !15
    ...                                                          !16
 274  7.5000E-01  0.0000E+00  1.4583E-01  1.4583E-01             !17
 275  7.7422E-01  3.6620E-02  1.3442E-01  1.3442E-01             !18
 276  8.1250E-01  0.0000E+00  1.1328E-01  1.1328E-01             !19
                                                                 !20
*** FLUX COMPONENTS AT ELEMENT INTEGRATION POINTS ***            !21
ELEM, PT,  Radius r,   Axial z,    FLUX_1,     FLUX_2,          !22
   1  1  9.2269E-01  3.0937E-02 -3.5661E+00  2.3913E-01         !23
   1  2  8.9095E-01  4.3634E-02 -3.3251E+00  3.2568E-01         !24
   1  3  9.3485E-01  5.5422E-03 -3.6607E+00  4.3406E-02         !25
   1  4  9.4226E-01  4.3634E-02 -3.7190E+00  3.4444E-01         !26
   1  5  9.7651E-01  9.4004E-03 -3.9943E+00  7.6904E-02         !27
   1  6  9.0206E-01  7.4009E-02 -3.4085E+00  5.5929E-01         !28
   1  7  8.8949E-01  9.4004E-03 -3.3141E+00  7.0049E-02         !29
    ...                                                          !30
 123  1  1.2462E-01  9.2829E-01 -6.5057E-02  9.6919E-01         !31
 123  2  1.2630E-01  9.0186E-01 -6.6820E-02  9.5426E-01         !32
 123  3  1.4331E-01  9.3536E-01 -8.6027E-02  1.1230E+00         !33
 123  4  1.0426E-01  9.4767E-01 -4.5535E-02  8.2776E-01         !34
 123  5  1.2178E-01  9.7314E-01 -6.2120E-02  9.9280E-01         !35
 123  6  9.2934E-02  9.1631E-01 -3.6177E-02  7.1340E-01         !36
 123  7  1.5916E-01  8.9544E-01 -1.0611E-01  1.1940E+00         !37
                                                                 !38
 *** SUPER_CONVERGENT AVERAGED NODAL FLUXES ***                  !39
NODE,   Radius r,   Axial z,    FLUX_1,     FLUX_2,             !40
   1  6.0268E-02  9.9635E-01 -1.5215E-02  5.0306E-01             !41
   2  0.0000E+00  1.0000E+00  7.9862E-07  7.1783E-08             !42
   3  3.9550E-02  9.5554E-01 -6.5518E-03  3.1661E-01             !43
   4  0.0000E+00  9.3750E-01  1.1258E-07  2.2950E-08             !44
   5  9.9819E-02  9.5190E-01 -4.1737E-02  7.9601E-01             !45
    ...                                                          !46
 273  8.3672E-01  3.6620E-02 -2.9326E+00  2.5670E-01             !47
 274  7.5000E-01  0.0000E+00 -2.3562E+00  2.4802E-08             !48
 275  7.7422E-01  3.6620E-02 -2.5109E+00  2.3752E-01             !49
 276  8.1250E-01  0.0000E+00 -2.7653E+00  4.2908E-09             !50
                                                                 !51
MAXIMUM ELEMENT ENERGY ERROR OF 7.01E-01 OCCURS IN ELEM 8        !52
MAXIMUM ENERGY ERROR DENSITY OF 3.69E+00 OCCURS IN ELEM 63       !53
```

Figure 10.10.3 Selected sphere output results

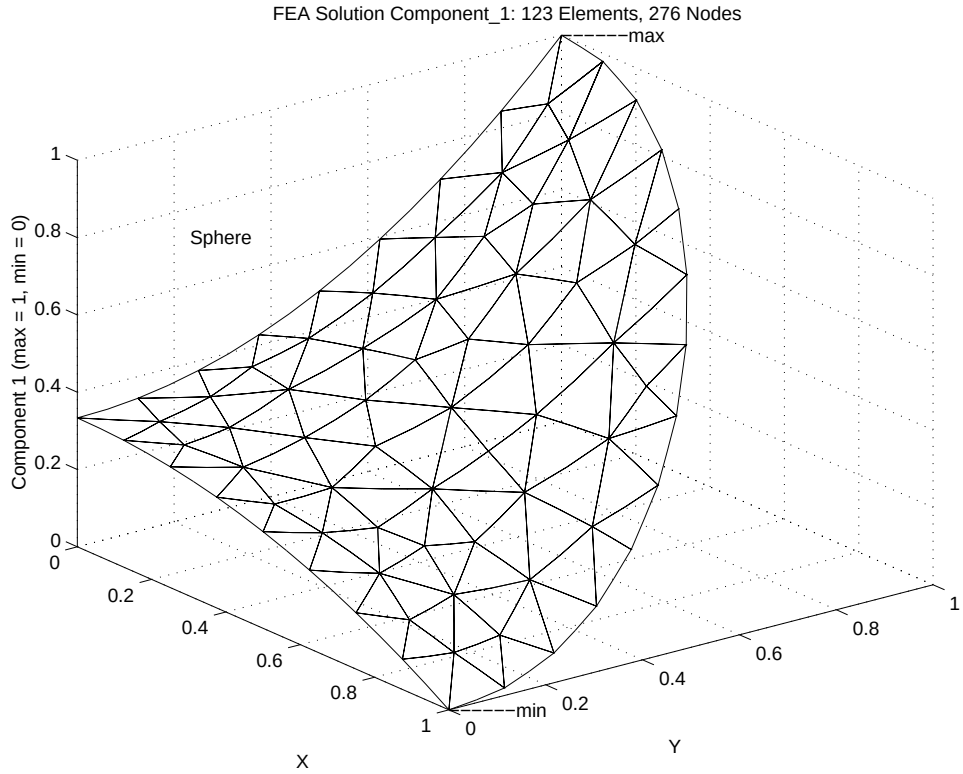


Figure 10.10.4 Carpet plot of finite element temperature

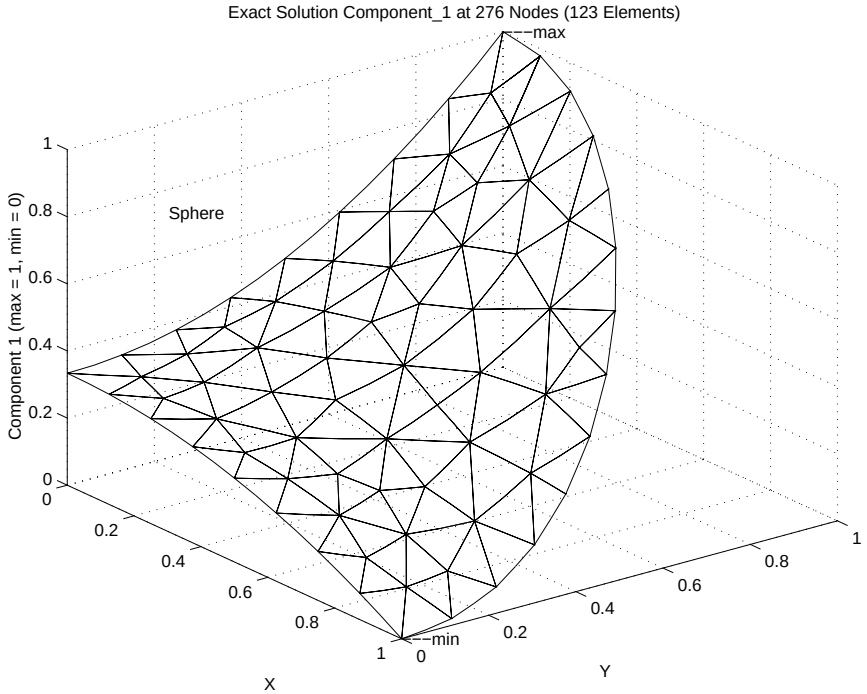


Figure 10.10.5 Carpet plot of exact temperature

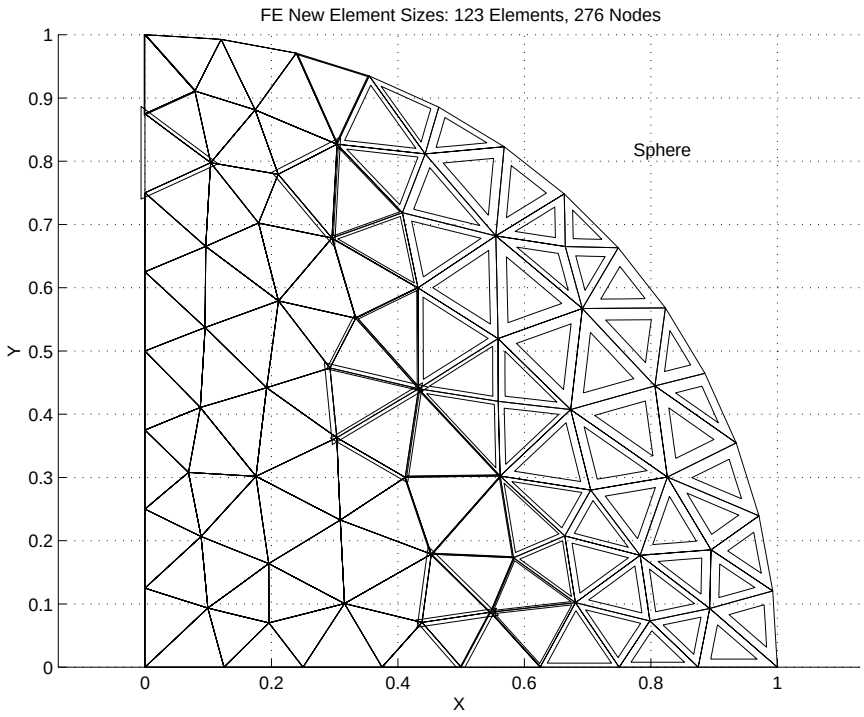


Figure 10.10.6 Element sizes for next mesh adaption

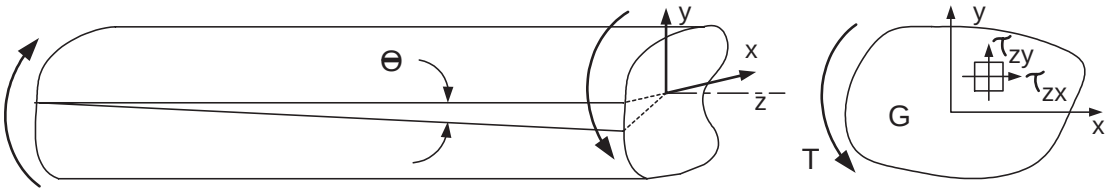


Figure 10.11.1 Torsion of a constant cross-section bar

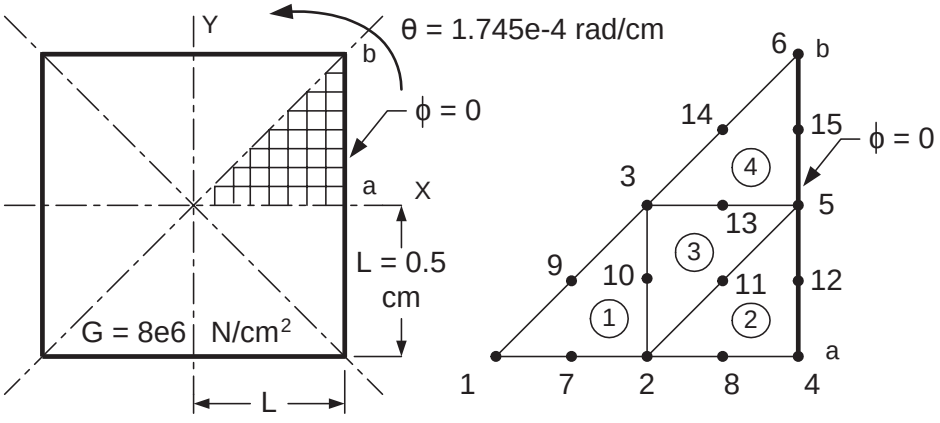


Figure 10.11.2 Torsion of a square shaft

10.11 Torsion

Most finite element formulations of stress analysis problems employ an energy formulation rather than beginning with the differential equations of equilibrium and applying the Galerkin method. It has been shown that both approaches yield identically the same element matrices. However, there are a few cases where a different approach is also useful. One such method is to employ a stress function, $\Phi(x, y)$, whose derivatives define the mechanical stresses that appear in the equation of equilibrium. This lets us recast those equations into another set of differential equations, and boundary conditions whose solution is known, or more easily computed. The case in point here is the torsion of a straight elastic bar of arbitrary cross-sectional area, shown in Fig. 10.11.1. The bar is subjected to a twisting moment, or torque, T, that causes an angle of twist per unit length of Θ. The twist per unit length is assumed to be small compared to its value squared. Assuming the stress function defined the shear stress components, in the cross-sectional plane, by $\tau_{zx} = \partial\Phi/\partial y$ and $\tau_{zy} = \partial\Phi/\partial x$. Then the governing differential equation is:

$$\frac{1}{G}\frac{\partial^2\Phi}{\partial x^2} + \frac{1}{G}\frac{\partial^2\Phi}{\partial y^2} + 2\Theta = 0$$

in the domain of the cross-section, Ω with the essential boundary condition that $\Phi = 0$ on its boundary. In the above, G denotes the elastic shear modulus of the material. For a homogeneous material we could usually divide by G but we wish to allow for non-homogeneous bars so we must keep it with the differential operator. Based on the previous chapter we recognize this as the Poisson equation. We previously defined how to implement its solution by the finite element method. Here the terms just take on new meaning and the post-processing will change. Before, the gradient vector components were directly proportional to the heat flux vector. But here different signs are associated with each component of the recovered gradient. Also the x component of stress is related to the y component of the gradient, and visa versa. Another new post-processing aspect is that once the solution Φ is known its integral, over the cross-section is related to the applied torque causing the twist by

$$T = 2\int_{\Omega} \Phi d\Omega.$$

It turns out that these equations are related to another problem known as the "soap film" analogy. There we visualized a thin soap film inflated over the cross-sectional shape by a constant pressure. Then the height of the soap film is proportional to the value of Φ. Also, the slope of the soap film is proportional to the shear stress at the same point, but the shear stress acts in a direction perpendicular to that slope. Finally, the volume under the membrane is proportional to the applied torque. This lets us visualize the expected results for the two common shapes of a circular and rectangular cross-section. For a circular bar the shear stress is zero at the center and maximum and constant along its circumference.. The distribution of shear stress is more complicated for a rectangular shape. It is also zero at the center point, but the maximum shear stress occurs at the two midpoints of the shortest sides of the rectangle. If we want to consider the torsion of a square bar then we could use the previous study of heat transfer of a square area with a constant internal source. As noted in figure 10.4.1 one can use a one-eighth symmetry

model. From the above analogy we will expect the maximum shear stress to occur at point *a*.

Segerlind [10] presents a detailed solution of the torsion of a square bar shown in Fig. 10.11.2. He used two linear triangles combined with one bilinear square element. Here we will employ a slightly better mesh by employing four quadratic (six node) triangles over the one-eighth symmetry region. The MODEL data file is shown in Fig. 10.11.3. There you will note that the angle of twist per unit length is given in the last line because it is a global (miscellaneous) property that applies to all elements. The shear modulus, *G*, is given for each element to allow for applications in volving bars of more that one material. The computed stress function amplitude is shown in Fig. 10.11.4 and the corresponding average error estimate is in Fig. 10.11.5. While the stress function may not be directly useful the shear stress vectors in the plane can be obtained from the physical derivatives and are shown at the quadrature points in Fig. 10.11.6, while their nodal average values are given in Fig. 10.11.7. Our analogy and handbook equations cite the mid-point *a* as the point of maximum shear stress. The handbook value is $\tau_{\max} = T \big/ (1.664 L^3)$ where T is the applied torque. The torque value is found by integrating the stress function in a post-processing step and gives $T = 8(24.41\ N\ cm) = 195.3\ N\ cm$. Thus the estimated maximum shear stress is $938.9\ N \big/ cm^2$ and the finite element prediction, at node 3, is $927.8\ N \big/ cm^2$. The error of about one-percent in the maximum stress for this crude mesh is quite reasonable (the three linear element model gave 11 % error). In practice however, we generally know the applied torque, T, and not the twist. Thus we have to scale all these linear results to match the actual torque. For example, if the actual torque was $250\ N - cm$ we scale stress and twist angle by the ratio of $T_{true} \big/ T_{fea}$, or $250./195.3 = 1.28$ to get the true maximum shear stress of $\tau_{\max} = 1,187.7\ N \big/ cm^2$, and a true twist value of $0.0002234\ radians \big/ cm$.

To be able to use any element in the interpolation library the solution has been implemented by numerical integration and the square matrix form is given in Fig. 10.11.8. To have the option to recover the physical gradients for stress calculation those data were saved to auxilary storage in lines 20 and 42. Likewise, it is not unusal to need the integral of the solution so the MODEL system sets aside storage for the integral of the element interpolation functions, **H**, and call it array H_INT. Line 46 of Fig. 10.11.8 allows for a user option to save those data for a later recovery of the torque, T. Keyword "post_el" in line 16 of Fig. 10.11.3 activated that storage as well as its recovery later on. The calculation of H_INT is quite cheap since **H** is already evaluated (at line 24) in the quadrature loop. In the next chapter we will see that most stress analysis problems are based on vector fields. Since data have been saved for post-processing we have supplied an INCLUDE file, my_post_el_inc, that is accessed when that keyword is present. Since two different recovery processes could be used we have included two special routines for this torsion problem. One to get only the shear stresses and one for the torque. They could be in a single code but it is best to use a modular programming style. The two methods are "encapsulated" within the post-processing "class" by placing them after the CONTAINS statement, as required by the language


```
title "TORSION OF A SQUARE BAR, 1/8 SYMMETRY, (4 ELEMS)"      ! 1
nodes    15 ! Number of nodes in the mesh                     ! 2
elems     4 ! Number of elements in the system                ! 3
dof       1 ! Number of unknowns per node                     ! 4
el_nodes  6 ! Maximum number of nodes per element             ! 5
space     2 ! Solution space dimension                        ! 6
b_rows    2 ! Number of rows in the B (operator) matrix       ! 7
shape     2 ! Element shape, 1=line, 2=tri, 3=quad, 4=hex     ! 8
remarks  12 ! Number of user remarks                          ! 9
gauss     7 ! Maximum number of quadrature point              !10
el_types  1 ! Number of different types of elements           !11
el_real   1 ! Number of real properties per element           !12
reals     1 ! Number of miscellaneous real properties         !13
el_homo     ! Element properties are homogeneous             !14
pt_list     ! List the answers at each node point            !15
post_el     ! Require post-processing, create n_tape1        !16
quit ! keyword input, remarks follow                          !17
1 SEGERLIND, 2ND ED. EXAMPLE P. 102, U1=217, U2=159, U4=125,  !18
2 AND T=21.9 VIA LINEAR ELEMENTS (T THEORY = 24.5). Keyword   !19
3 post_el turns on the torque recovery and stress listing.    !20
4                     / 6   Torque, T, twice the solution     !21
5 Mesh:             /    :   integral is reported after the   !22
6                 14     15  integral reported after the      !23
7 C/L           /    (3)  :   Here T=24.41 N-cm, and          !24
8 |           4 -- 13 --  5   U1=205.9, U2=160.3, U4=126.7 cm  !25
9 |         /  : (4)   /   :                                  !26
0 v    9    10    11     12  Max shear stress = 853.6 N/cm^2  !27
1   / (1)    : /  (2)    :   x=0.47 & y=0.025 cm (theory max  !28
2 1--  7 --- 2 --- 8 --- 3 <--- C/L value = 942 N/cm^2, @ 3)  !29
   1    0 0.        0.       ! node, bc_flag, x, y  (cm)      !30
   2    0 0.25      0.                                        !31
   3    1 0.5       0.                                        !32
   4    0 0.25      0.25                                      !33
   5    1 0.5       0.25                                      !34
   6    1 0.5       0.5                                       !35
   7    0 0.125     0.                                        !36
   8    0 0.375     0.                                        !37
   9    0 0.125     0.125                                     !38
  10    0 0.25      0.125                                     !39
  11    0 0.375     0.125                                     !40
  12    1 0.5       0.125                                     !41
  13    0 0.375     0.25                                      !42
  14    0 0.375     0.375                                     !43
  15    1 0.5       0.375                                     !44
     1  1    2    4    7   10    9  ! elem, six nodes         !45
     2  2    3    5    8   12   11                           !46
     3  4    5    6   13   15   14                           !47
     4  2    5    4   11   13   10                           !48
  3    1 0.      ! node, dof, essential bc value             !49
  5    1 0.                                                  !50
  6    1 0.                                                  !51
 12    1 0.                                                  !52
 15    1 0.                                                  !53
  1 8.e6       ! elem, shear_modulus (homogeneous) N/cm^2    !54
 1.745e-4      ! angle of twist (global) radians/cm          !55
```

Figure 10.11.3 Data for the torsion model

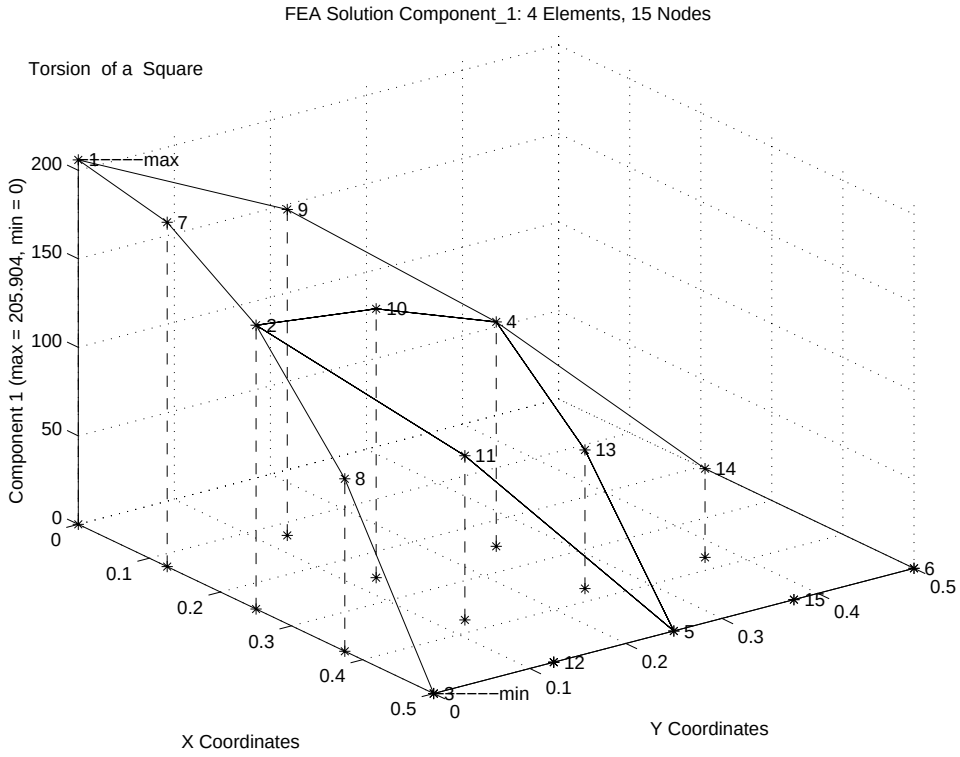


Figure 10.11.4 Stress function amplitude

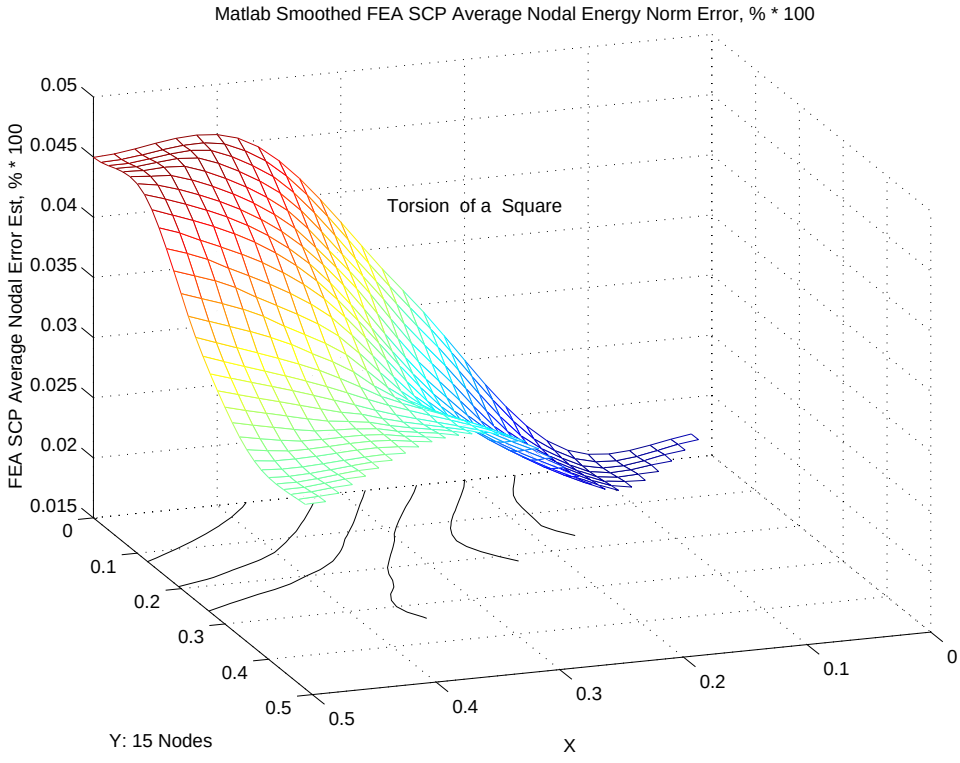


Figure 10.11.5 Estimated error in the solution

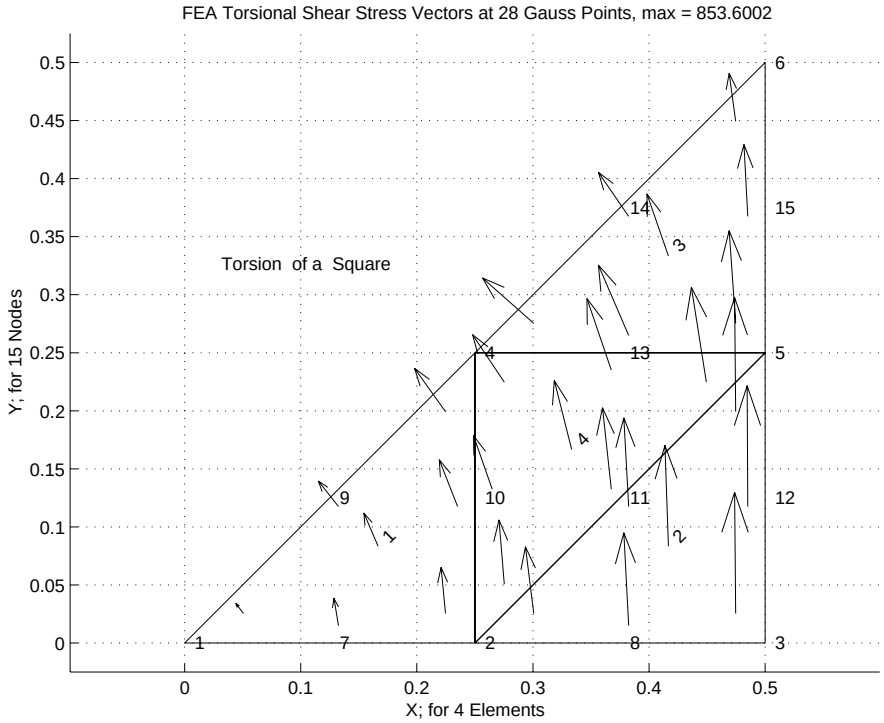


Figure 10.11.6 Shear stress vectors at the quadrature points

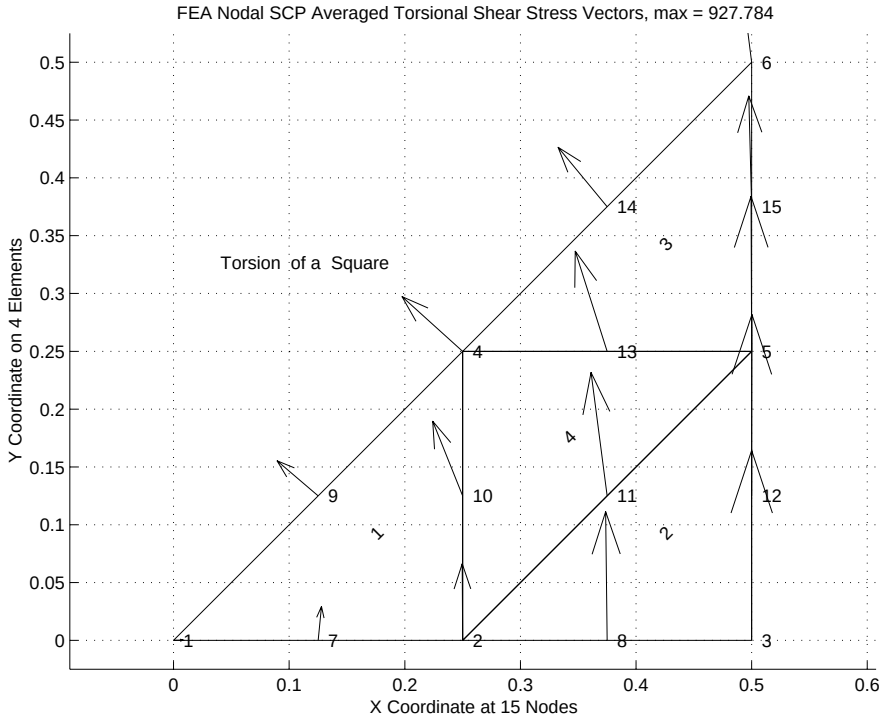


Figure 10.11.7 Shear stress vectors averaged at the nodes


```
! .......................................................          ! 1
!  **  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW **        ! 2
! .......................................................          ! 3
! TORSION (POISSON EQUATION) OF TWO-DIMENSIONAL SHAPE              ! 4
                                                                   ! 5
 REAL(DP), PARAMETER :: ZERO = 2 * TINY (1.d0)                     ! 6
 REAL(DP)            :: CONST, DET, SOURCE                         ! 7
 INTEGER             :: IP                                         ! 8
                                                                   ! 9
! 1/G *(U,xx + U,yy) + Q = 0, Example 205, Q = 2*Twist_Angle       !10
                                                                   !11
! Shear modulus  = el real property 1   = GET_REAL_LP (1)          !12
! Angle of twist = misc real property 1 = GET_REAL_MISC (1)        !13
                                                                   !14
!-->  DEFINE ELEMENT PROPERTIES                                    !15
  CALL APPLICATION_E_MATRIX (IE, XYZ, E)  ! diagonal 1/G           !16
  SOURCE = 2.d0 * GET_REAL_MISC (1)        ! twice twist           !17
                                                                   !18
!      STORE NUMBER OF POINTS FOR STRESS OR ERROR EST              !19
  CALL STORE_FLUX_POINT_COUNT ! Save LT_QP                         !20
                                                                   !21
!-->   NUMERICAL INTEGRATION LOOP                                  !22
  DO IP = 1, LT_QP                                                 !23
    H   = GET_H_AT_QP (IP)   ! INTERPOLATION FUNCTIONS             !24
    XYZ = MATMUL (H, COORD)  ! FIND POINT, ISOPARAMETRIC           !25
    DLH = GET_DLH_AT_QP (IP) ! FIND LOCAL DERIVATIVES              !26
    AJ  = MATMUL (DLH, COORD) ! FIND JACOBIAN AT THE PT            !27
!     FORM INVERSE AND DETERMINATE OF JACOBIAN                     !28
    CALL INVERT_JACOBIAN (AJ, AJ_INV, DET, N_SPACE)                !29
    CONST  = DET * WT(IP)                                          !30
                                                                   !31
!     EVALUATE GLOBAL DERIVATIVES, B == DGH                        !32
    DGH = MATMUL (AJ_INV, DLH)                                     !33
    CALL APPLICATION_B_MATRIX (DGH, XYZ, B) ! for error est        !34
                                                                   !35
!     ELEMENT MATRICES: Stiffness, Source, Result integral         !36
    S = S + CONST * MATMUL ((MATMUL (TRANSPOSE (B), E)),B)         !37
    C = C + CONST * SOURCE * H   ! source                          !38
    H_INTG = H_INTG + H * CONST  ! for solution integral           !39
                                                                   !40
!-->  SAVE PT., E AND DERIVATIVES, FOR POST PROCESSING             !41
    CALL STORE_FLUX_POINT_DATA (XYZ, E, B)                         !42
  END DO                                                           !43
                                                                   !44
!-->     SAVE INTEGRAL OF INTERPOLATION FUNCTIONS                  !45
  IF ( N_TAPE1 > 0 )  WRITE (N_TAPE1)  H_INTG ! post_el keyword    !46
                                                                   !47
!    ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***      !48
```

Figure 10.11.8 Element matrices for torsion


```
! 205.my_post_el_inc                                                  ! 1
! ..................................................................  ! 2
!  ** POST_PROCESS_ELEM PROBLEM DEPENDENT STATEMENTS FOLLOW **       ! 3
!     Given: INTEGER,  INTENT(IN) :: N_TAPE1, IE                     ! 4
!  REAL(DP), INTENT(IN) :: COORD (LT_N, N_SPACE), D (LT_FREE)        ! 5
! ..................................................................  ! 6
!     TORSION (POISSON EQUATION) OF TWO-DIMENSIONAL SHAPE            ! 7
! SHEAR STRESSES AND TORQUE, if keyword post_el is true              ! 8
 LOGICAL, SAVE :: EACH = .false. ! list each or total ?              ! 9
                                                                     !10
   CALL LIST_ELEM_TORSION_STRESS  (IE)                               !11
   CALL LIST_ELEM_TORQUE_INTEGRAL (N_TAPE1, IE, EACH)                !12
                                                                     !13
Contains ! the local methods cited above                             !14
                                                                     !15
```

Figure 10.11.9 Encapsulating the element stress and torque recovery

standard.

10.12 Exercises

1. The Laplace equation on a square

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad 0 < x < 1, \quad 0 < y < 1$$

with essential (Dirichlet) boundary conditions on the four edges of

$$u(x = 0) = -y^3, \quad u(x = 1) = -1 - y^3 + 3y^2 + 3y$$

$$u(y = 0) = -x^3, \quad u(y = 1) = -1 - x^3 + 3x^2 + 3x$$

has an exact solution given by the cubic $u(x, y) = -x^3 - y^3 + 3xy^2 + 3x^2 y$. Obtain a finite element solution and sketch it along with the exact value along the center lines of $x = 1/2$ and $y = 1/2$, or compare to *exact_case* 25 in the MODEL code.

2. Modify the above problem to have normal flux (Neumann) boundary conditions on the two edges where y is constant (corresponding to the same exact solution):

$$u(x = 0) = -y^3, \qquad \frac{\partial u}{\partial y}(y = 0) = 3x^2,$$

$$u(x = 1) = -1 - y^3 + 3y^2 + 3y, \qquad \frac{\partial u}{\partial y}(y = 1) = -3 + 6x + 3x^2.$$

3. Consider a two-dimensional Poisson equation on a unit square $0 \le x \le 1$, $0 \le y \le 1$ that contains a local high gradient peak on its interior centered at $x = beta$, $y = \beta$. The amplitude of the peak is set by a parameter α. Let $A = (x - \beta)/\alpha$, $B = (y - \beta)/\alpha$, $C = (1 - \beta)/\alpha$, and $D = \beta/\alpha$. Then for the source, Q, defined by


```
SUBROUTINE LIST_ELEM_TORSION_STRESS (IE)                              !16
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   !17
!     LIST ELEMENT SHEAR STRESS AT QUADRATURE POINTS                  !18
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   !19
Use System_Constants ! for DP, N_R_B, N_SPACE, E, XYZ                 !20
Use Elem_Type_Data   ! for LT_FREE, LT_N, D, DGH                      !21
 IMPLICIT NONE                                                        !22
 INTEGER,  INTENT(IN) :: IE                                           !23
!         Global Arrays                                              !24
 REAL(DP) :: DGH (N_SPACE, LT_FREE), STRESS (N_R_B + 2), &            !25
             XYZ (N_SPACE), E (N_R_B, N_R_B)                          !26
                                                                      !27
 INTEGER,  SAVE :: TEST_E, TEST_P, J, N_IP ! for max value            !28
 REAL(DP), SAVE :: DERIV_MAX = -HUGE(1.d0) ! for max value            !29
                                                                      !30
!                 VARIABLES:                                         !31
! D       = NODAL PARAMETERS ASSOCIATED WITH AN ELEMENT               !32
! E       = CONSTITUTIVE MATRIX                                      !33
! DGH     = GLOBAL DERIVATIVES INTERPOLATION FUNCTIONS               !34
! IE      = CURRENT ELEMENT NUMBER                                   !35
! LT_N    = NUMBER OF NODES PER ELEMENT                              !36
! LT_FREE = NUMBER OF DEGREES OF FREEDOM PER ELEMENT                 !37
! N_ELEMS = TOTAL NUMBER OF ELEMENTS                                 !38
! N_R_B   = NUMBER OF ROWS IN B AND E MATRICES                       !39
! N_SPACE = DIMENSION OF SPACE                                       !40
! STRESS  = STRESS OR GRADIENT VECTOR                                !41
! XYZ     = SPACE COORDINATES AT A POINT                             !42
                                                                      !43
!-->     PRINT TITLES ON THE FIRST CALL                               !44
  IF ( IE == 1 ) THEN ; WRITE (N_PRT, 5)                              !45
    5 FORMAT (/,'*** TORSIONAL SHEAR STRESSES ***',/,     &           !46
    ' ELEMENT, POINT,        X           Y ', /,          &           !47
    ' ELEMENT,         TAU_ZX        TAU_ZY        TAU', /)           !48
  END IF                                                              !49
                                                                      !50
  CALL READ_FLUX_POINT_COUNT (N_IP) ! NUMBER OF POINTS                !51
                                                                      !52
  DO J = 1, N_IP                            ! QUADRATURE LOOP         !53
    CALL READ_FLUX_POINT_DATA (XYZ, E, B) ! RECOVER DATA              !54
                                                                      !55
!       CALCULATE SHEAR STRESSES,  STRESS = E*DGH*D                   !56
    STRESS (1:N_R_B) = MATMUL (DGH, D)                                !57
    STRESS (N_R_B+1) = SQRT ( SUM (STRESS(1:N_R_B)**2) )              !58
                                                                      !59
!-->     PRINT COORDINATES AND GRADIENT AT THE POINT                  !60
    WRITE (N_PRT, 20) IE, J, XYZ                                      !61
    20 FORMAT ( I7, I6, 3(1PE13.5))                                   !62
    WRITE (N_PRT, 30) IE, STRESS(2), -STRESS(1), STRESS(3)            !63
    30 FORMAT ( I7, 6X, 4(1PE13.5) )                                  !64
    IF ( STRESS (N_R_B+1) > DERIV_MAX ) THEN                          !65
      DERIV_MAX = STRESS (N_R_B+1)   ! maximum value                  !66
      TEST_E   = IE ; TEST_P = J     ! maximum point                  !67
    END IF                                                            !68
  END DO ! integration                                                !69
!-->   ARE CALCULATIONS COMPLETE FOR ALL ELEMENTS                     !70
  IF ( IE == N_ELEMS ) THEN ; WRITE (N_PRT,         &                 !71
    "('LARGEST SHEAR STRESS = ', 1PE13.5)") DERIV_MAX                 !72
    WRITE (N_PRT, "('ELEM =', I6, ', POINT = ', I2)") &               !73
  TEST_E, TEST_P ; END IF ! LAST ELEMENT                              !74
END SUBROUTINE LIST_ELEM_TORSION_STRESS                               !75
```

Figure 10.11.10 Element shear stress recovery option


```
SUBROUTINE LIST_ELEM_TORQUE_INTEGRAL (N_TAPE, IE, EACH)   ! 78
! * * * * * * * * * * * * * * * * * * * * * * * * * * *   ! 79
!  LIST INTEGRAL OF TORQUE FROM H INTEGRAL, ON N_TAPE     ! 80
! * * * * * * * * * * * * * * * * * * * * * * * * * * *   ! 81
Use System_Constants  ! for DP                            ! 82
Use Elem_Type_Data    ! for LT_FREE, LT_N, D, H_INTG      ! 83
 IMPLICIT NONE                                            ! 84
 INTEGER,  INTENT(IN) :: N_TAPE, IE  ! source of data     ! 85
 LOGICAL,  INTENT(IN) :: EACH        ! list each ?        ! 86
                                                          ! 87
 REAL(DP), SAVE :: VALUE, TOTAL      ! integrals          ! 88
 REAL(DP)       :: H_INTG (LT_FREE)                       ! 89
 INTEGER        :: EOF               ! End_Of_File        ! 90
                                                          ! 91
!                 VARIABLES:                              ! 92
! D       = NODAL PARAMETERS ASSOCIATED WITH ELEMENT      ! 93
! EACH    = TRUE IF ALL LISTED, ELSE JUST TOTAL           ! 94
! H       = SOLUTION INTERPOLATION FUNCTIONS              ! 95
! H_INTG  = INTEGRAL OF INTERPOLATION FUNCTIONS           ! 96
! IE      = CURRENT ELEMENT NUMBER                        ! 97
! LT_FREE = NUMBER OF DEGREES OF FREEDOM PER ELEMENT      ! 98
! N_TAPE  = UNIT FOR POST SOLUTION MATRICES STORAGE       ! 99
                                                         !100
!-->     PRINT TITLES ON THE FIRST CALL AND INITIALIZE   !101
  IF ( IE == 1 ) THEN ; TOTAL = 0.d0                     !102
    IF ( EACH ) WRITE (N_PRT, 5) ; 5 FORMAT       &      !103
      (/,'** TORQUE INTEGRAL CONTRIBUTIONS **',/, &      !104
        'ELEMENT      TORQUE')                           !105
  END IF                                                 !106
  IF ( IE <= N_ELEMS ) THEN ! ELEMENT RESULTS            !107
    READ (N_TAPE, IOSTAT = EOF) H_INTG   ! GET INTEGRAL  !108
    IF ( EOF /= 0 ) THEN ; PRINT *,                &     !109
      'LIST_ELEM_TORQUE_INTEGRAL EOF AT ELEMENT ', IE    !110
      STOP 'ERROR, EOF IN LIST_ELEM_TORQUE_INTEGRAL'     !111
    END IF ! MISSING DATA                                !112
                                                         !113
!-->   CALCULATE ELEMENT CONTRIBUTION, VALUE = H_INTG*D  !114
    VALUE = DOT_PRODUCT (H_INTG, D) * 2.d0               !115
    TOTAL = TOTAL + VALUE                                !116
    IF ( EACH ) WRITE (N_PRT, '(I7, 1PE18.6)') IE, VALUE !117
    IF ( IE == N_ELEMS ) WRITE (N_PRT, &                 !118
      "('TOTAL TORQUE INTEGRAL = ', 1PE16.6, /)")  TOTAL !119
  END IF                                                 !120
END SUBROUTINE LIST_ELEM_TORQUE_INTEGRAL                 !121
! ..................................................... !122
! * END POST_PROCESS_ELEM PROBLEM DEPENDENT STATEMENTS *  !123
! ..................................................... !124
```

Figure 10.11.11 Element torque recovery option

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = Q = -6x - 6y - \frac{4}{\alpha^2}\left[1 + A^2 + B^2\right]\exp\left[-A^2 - B^2\right]$$

and edge Dirichlet and Neumann boundary conditions

$$u(x=0) = -y^3 + \exp\left[-D^2 - B^2\right], \quad u_{,y}(y=0) = \frac{2\beta}{\alpha^2}\exp\left[-D^2 - A^2\right]$$

$$u(x=1) = -1 - y^3 + \exp\left[-C^2 - B^2\right], \quad u_{,y}(y=1) = -3 - 2\frac{C}{\alpha}\exp\left[-A^2 - C^2\right]$$

the exact solution is $u = -x^3 - y^3 + \exp[-A^2 - B^2]$. Obtain a finite element solution and compare it to the exact values along the center lines $x = 1/2$, and $y = 1/2$. Assume $\beta = 0.5$ and $\alpha = 0.05$.

4. Consider the partial differential equation $\nabla\cdot(\mathbf{E}\nabla\phi) + \boldsymbol{v}\cdot\nabla\phi + F\phi + Q = 0$ *in* Ω. The second term is new. In 2-D it is $\boldsymbol{v}\cdot\nabla\phi = \begin{Bmatrix} v_x \\ v_y \end{Bmatrix}\left[\frac{\partial\phi}{\partial x} \quad \frac{\partial\phi}{\partial y}\right]$. Apply the Galerkin method to get the additional matrix, say $\mathbf{U}_v^e$, that appears because of this term. State the result in matrix integral form. Is the result symmetric? The data $\boldsymbol{v}$ are often given at the nodes and we employ standard interpolation for those data:

$$\underset{1\,x\,s}{\mathbf{v}_t} = \underset{1\,x\,s}{[v_x \quad v_y]} = \underset{1\,x\,n}{\mathbf{H}^e}\,\underset{n\,x\,s}{[V_x^e \quad V_y^e]}.$$

Outline how you would numerically integrate $\mathbf{U}_v^e$ in that case. Give the linear line element matrix for a 1-D problem with v_x given constant data.

10.13 References

[1] Allaire, P.E., *Basics of the Finite Element Method*, Dubuque: Wm. C. Brown Pub. (1985).

[2] Carslaw, H.S. and Jaeger, J.C., *Conduction of Heat in Solids*, Oxford: Oxford Press (1959).

[3] Cook, R.D., Malkus, D.S., Plesha, N.E., and Witt, R.J., *Concepts and Applications of Finite Element Analysis*, New York: John Wiley (2002).

[4] Desai, C.S. and , T. Kundu, *Introduction to the Finite Element Method*, Boca Raton, FL: CRC Press (2001).

[5] Hinton, E. and Owen, D.R.J., *Finite Element Programming*, London: Academic Press (1977).

[6] Hughes, T.J.R., *The Finite Element Method*, Englewood Cliffs: Prentice-Hall (1987).

[7] Kwon, Y.W. and Bang, H., *The Finite Element Method using Matlab*, Boca Raton: CRC Press (1997).

[8] Meek, J.L., "Field Problems Solutions by Finite Element Methods," *Civil Eng. Trans., Inst. Eng. Aust.*, , pp. 173−180 (Oct. 1968).

[9] Myers, G.E., *Analytical Methods in Conduction Heat Transfer*, New York: McGraw-Hill (1971).

[10] Segerlind, L.J., *Applied Finite Element Analysis*, New York: John Wiley (1984).

[11] Silvester, P.P. and Ferrari, R.L., *Finite Elements for Electrical Engineers*, Cambridge: Cambridge University Press (1983).

[12] Zienkiewicz, O.C. and Taylor, R.L., *The Finite Element Method*, 5th Edition, London: Arnold (2000).

