The relative ranking of the error levels in the uniform mesh is given in Fig. 11.9.16. The SCP error estimating process yields a refinement parameter that can be used to select new element sizes that can be passed as input data into an automatic mesh generation program such as that given by Huang and Usmani [10] That process can be repeated to develop a series of solutions that approach the specified level of acceptable error in the energy norm. The number of equations involved in this series of meshes was 121, 223, 436, 757, and 1360, respectively. The reduction in the error norm is shown in Fig. 11.9.17 as a function of the number of equations solved. Note that to obtain the same error reduction with a uniform mesh refinement would have required about 4000 unknowns compared to the 1360 in the last mesh. For this problem all three choices for the type of patch definition gave the same error estimates and mesh refinements. The four such meshes that followed from the initial uniform mesh are shown in Fig. 11.9.18.

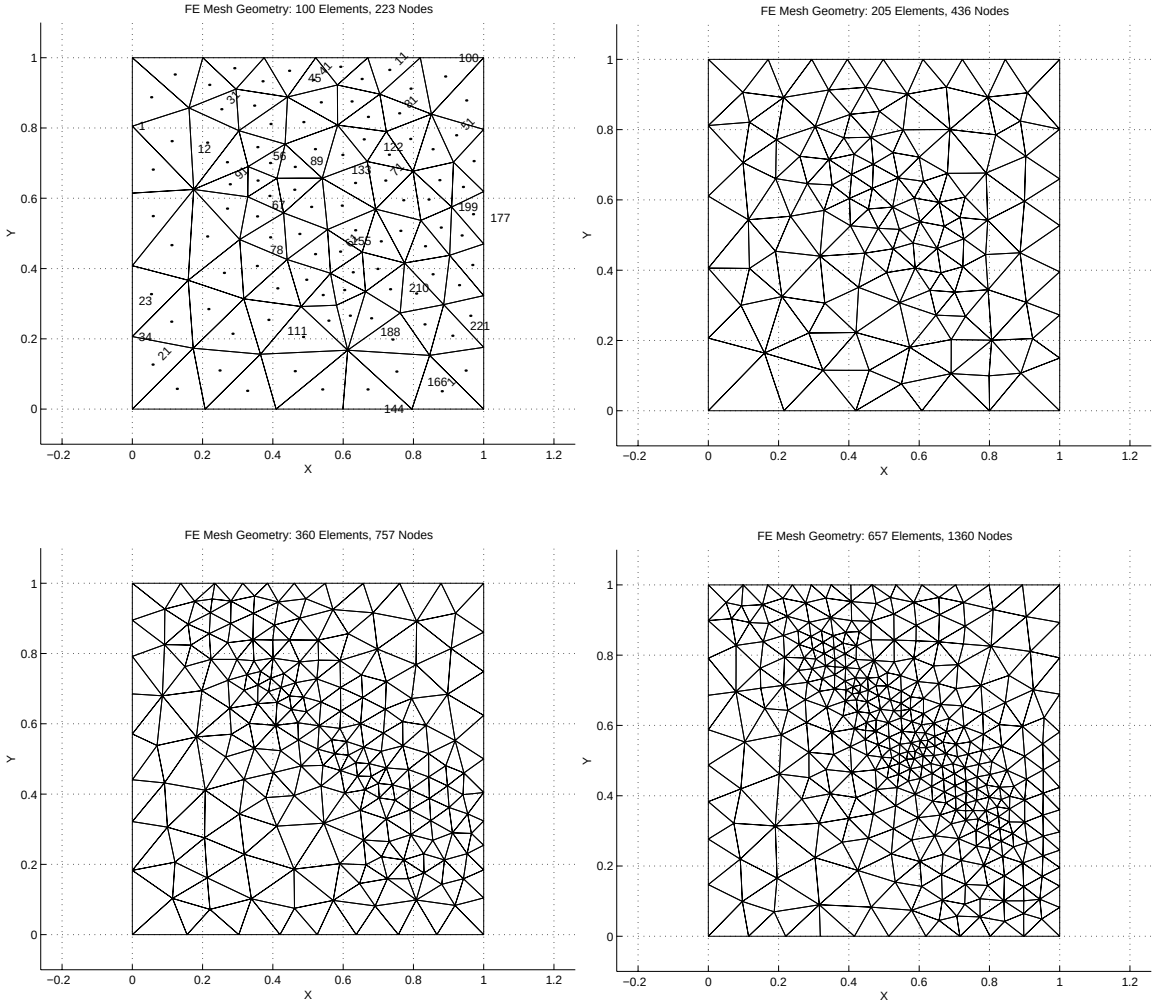

Figure 11.9.18 Strong diagonal example h-adaptive mesh stages

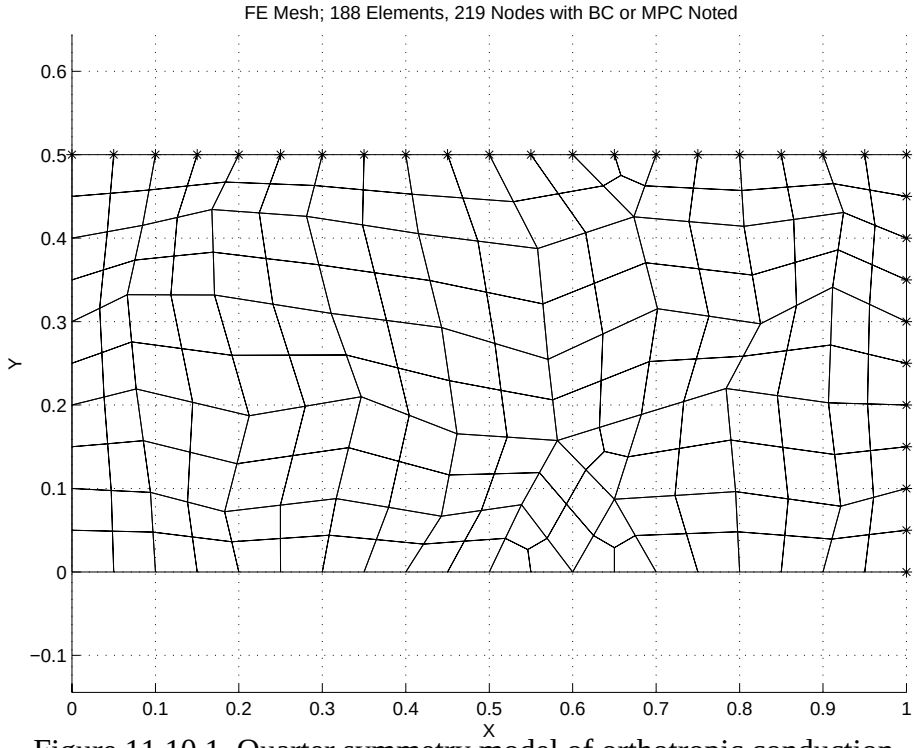

Figure 11.10.1 Quarter symmetry model of orthotropic conduction

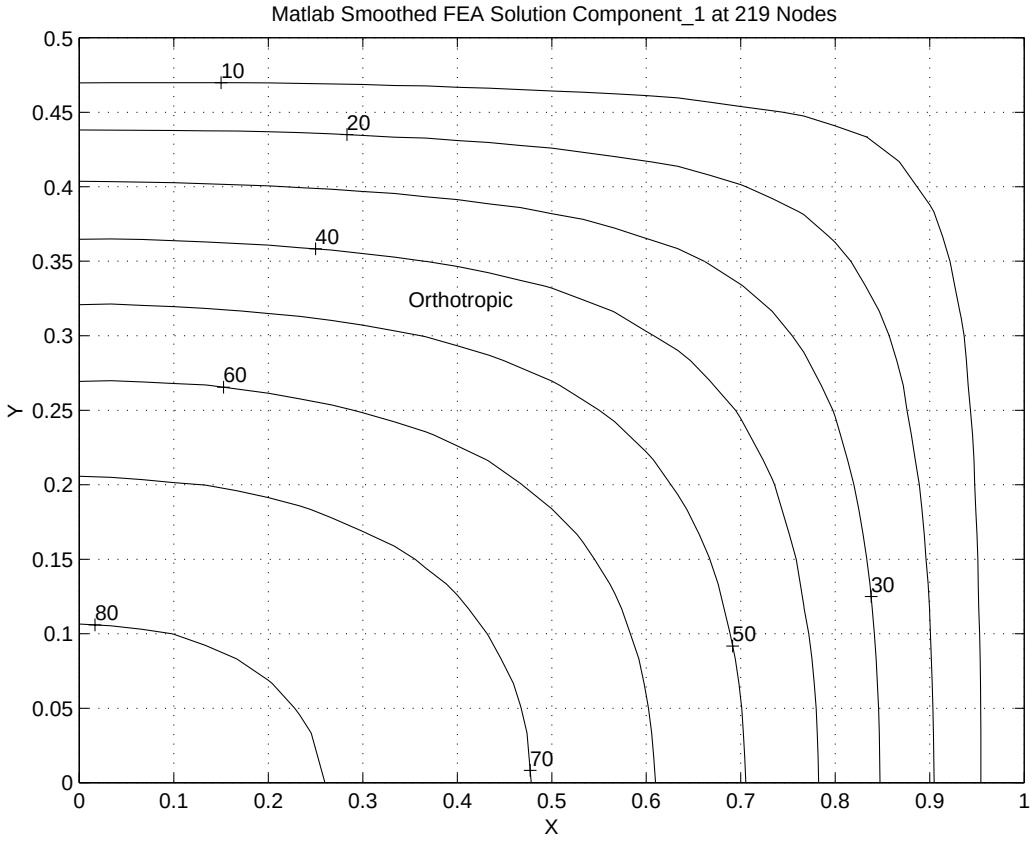

Figure 11.10.2 Temperature, $K_x = 2, K_y = 1.237, Q = 1000$

11.10 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 [5] 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. 11.10.1. 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.11.10.2-4.

The exact flux vectors for the isotropic material are shown in Fig. 11.10.5. 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. 11.10.6, 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. 11.10.7 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. 11.10.9 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. 11.10.11.

An interesting question is how well does 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. 11.10.12) and the peak error in the energy norm. Figures 11.10.13 and 14 show the exact and SCP estimated error as measured in the energy norm. The agreement in both the relative location of the peak error and the values of the local errors are unusually good.

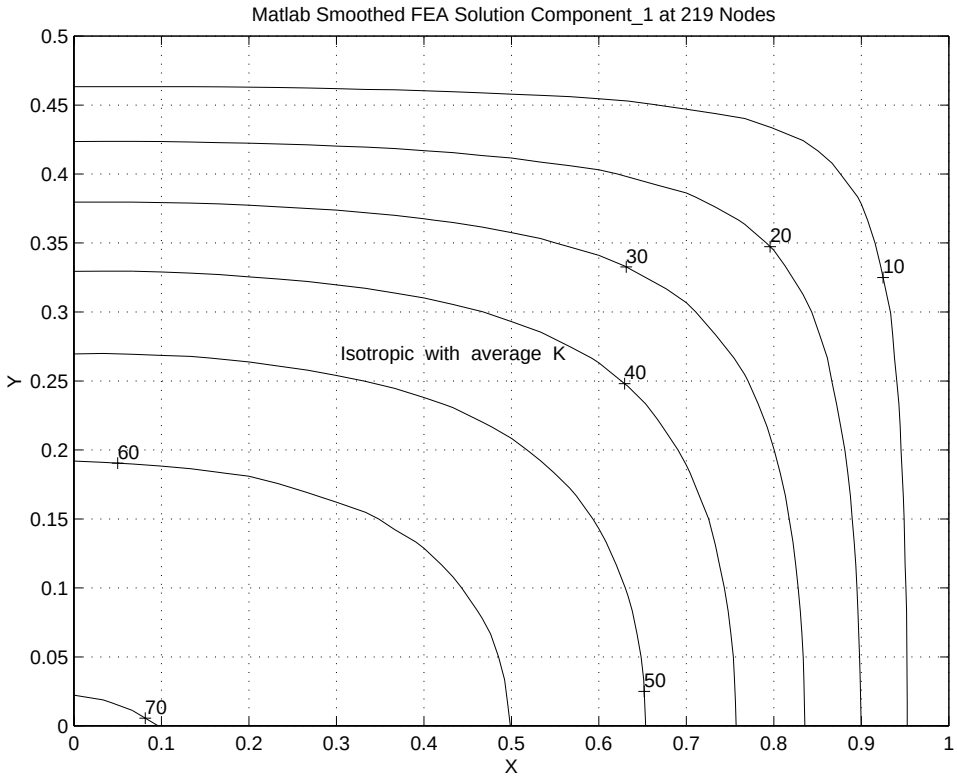

Figure 11.10.3 Temperature, $K_x = K_y = 1.617, Q = 1000$

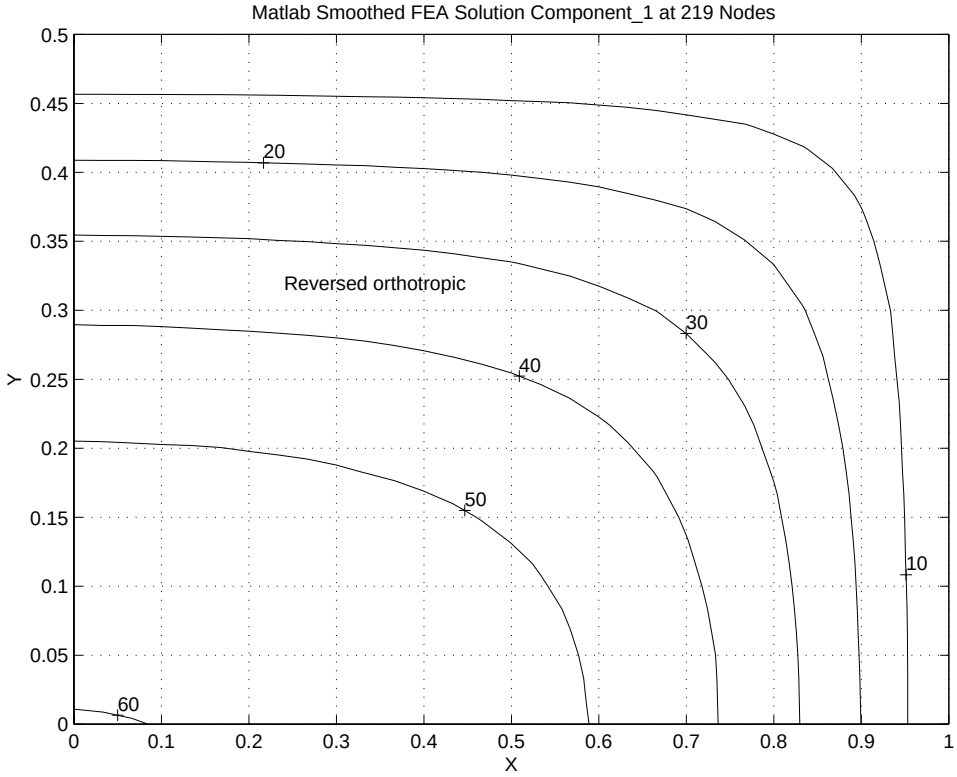

Figure 11.10.4 Temperature, $K_x = 1.2337, K_y = 2, Q = 1000$

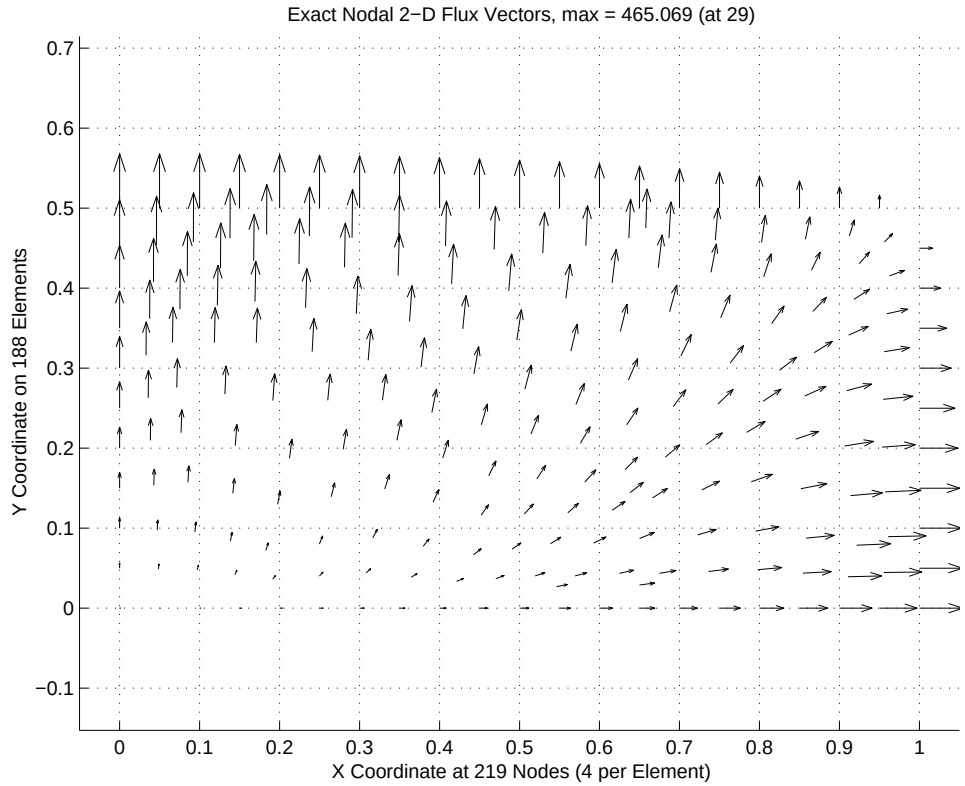

Figure 11.10.5 Exact isotropic heat flux vectors

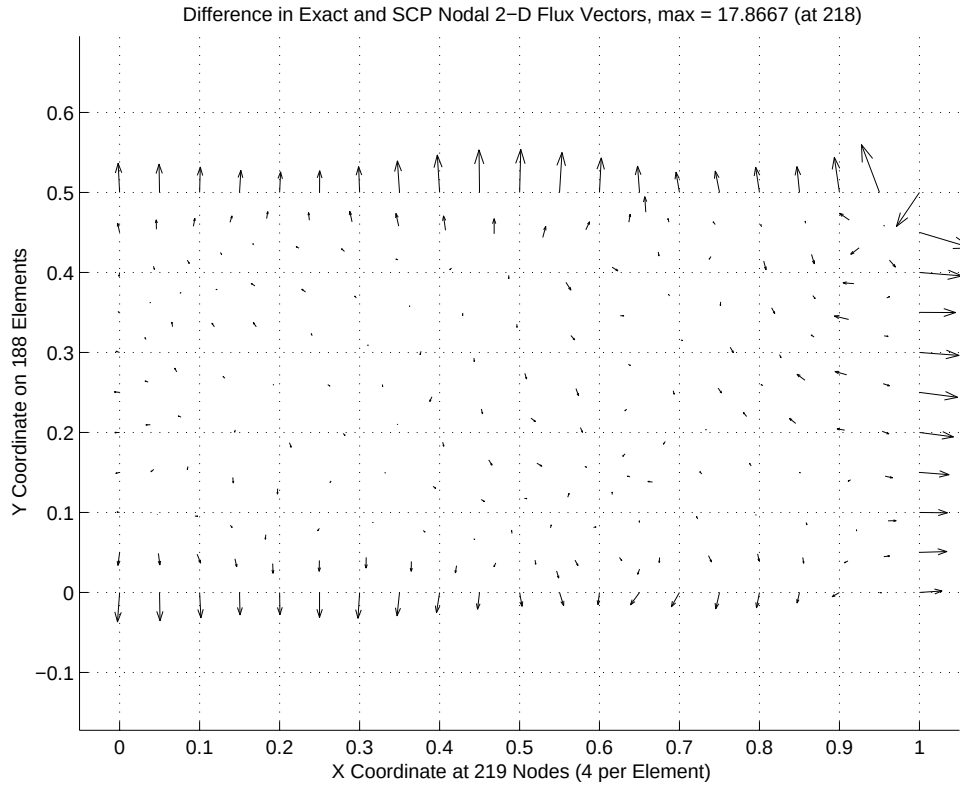

Figure 11.10.6 Differences in exact and SCP average flux vectors

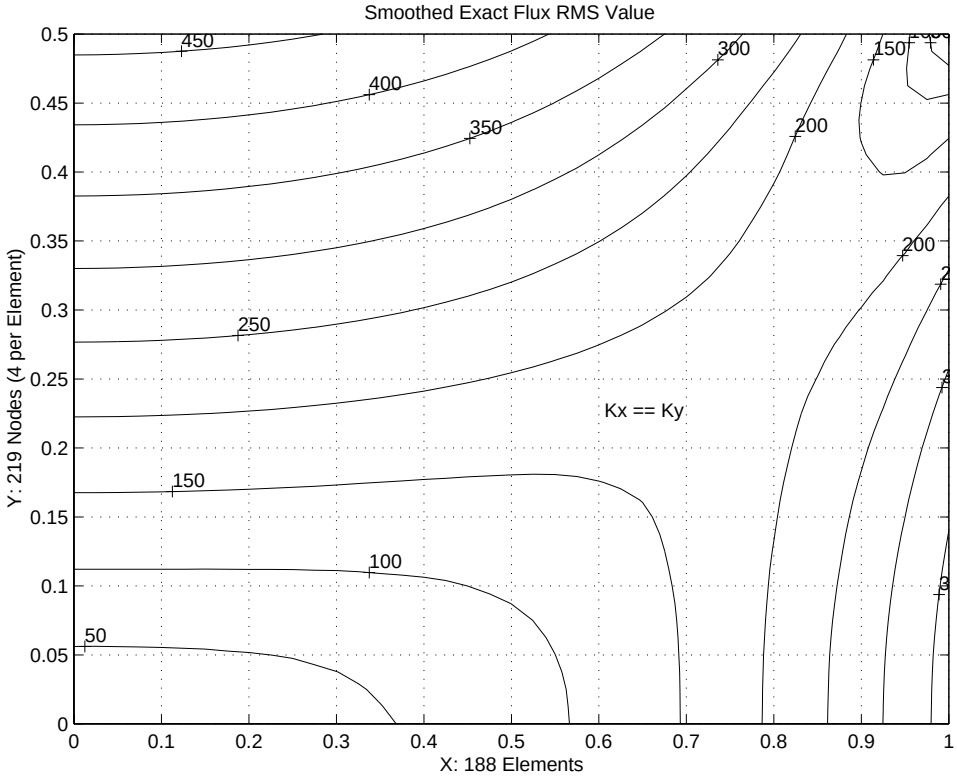

Figure 11.10.7 Exact isotropic heat flux value contours

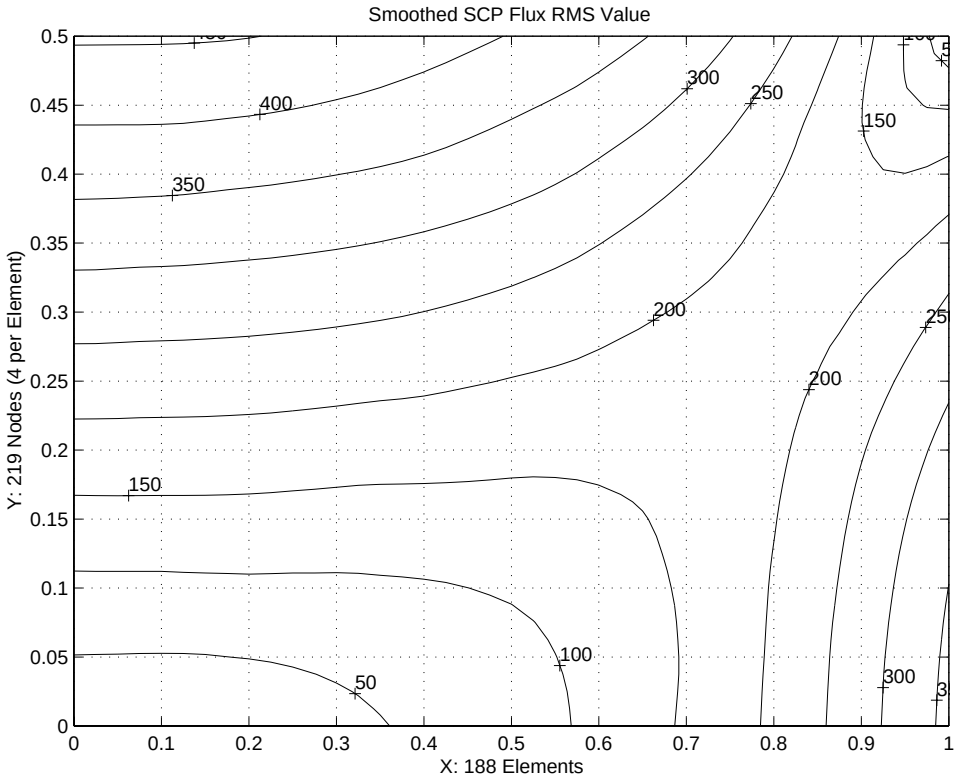

Figure 11.10.8 SCP averaged isotropic heat flux value contours

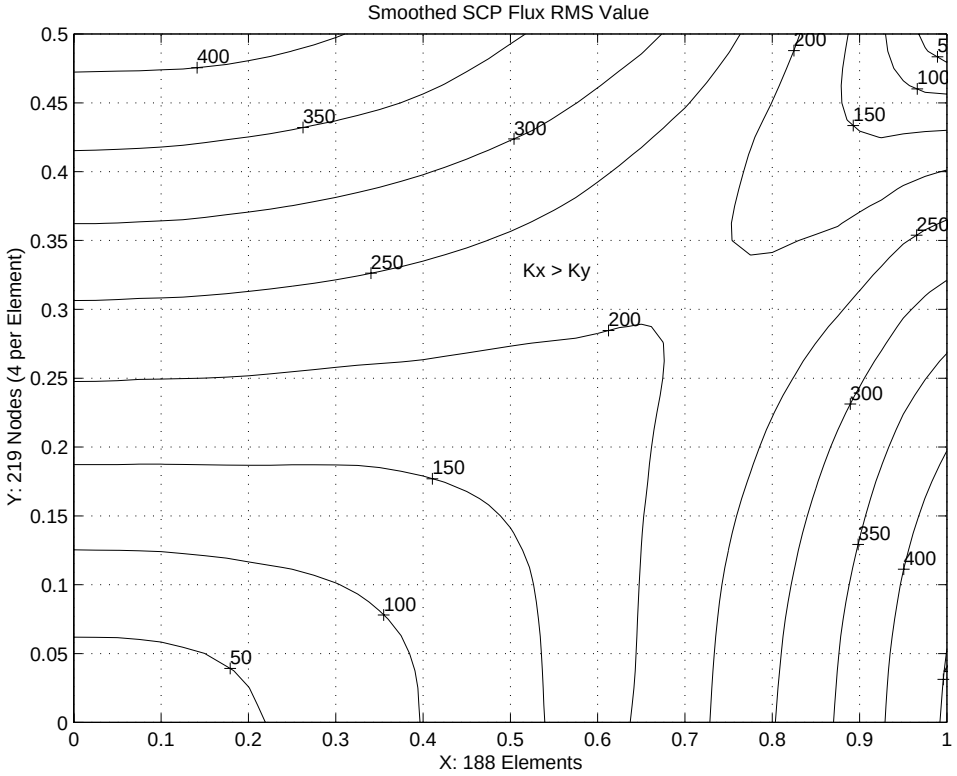

Figure 11.10.9 SCP averaged *Kx* > *Ky* heat flux value contours

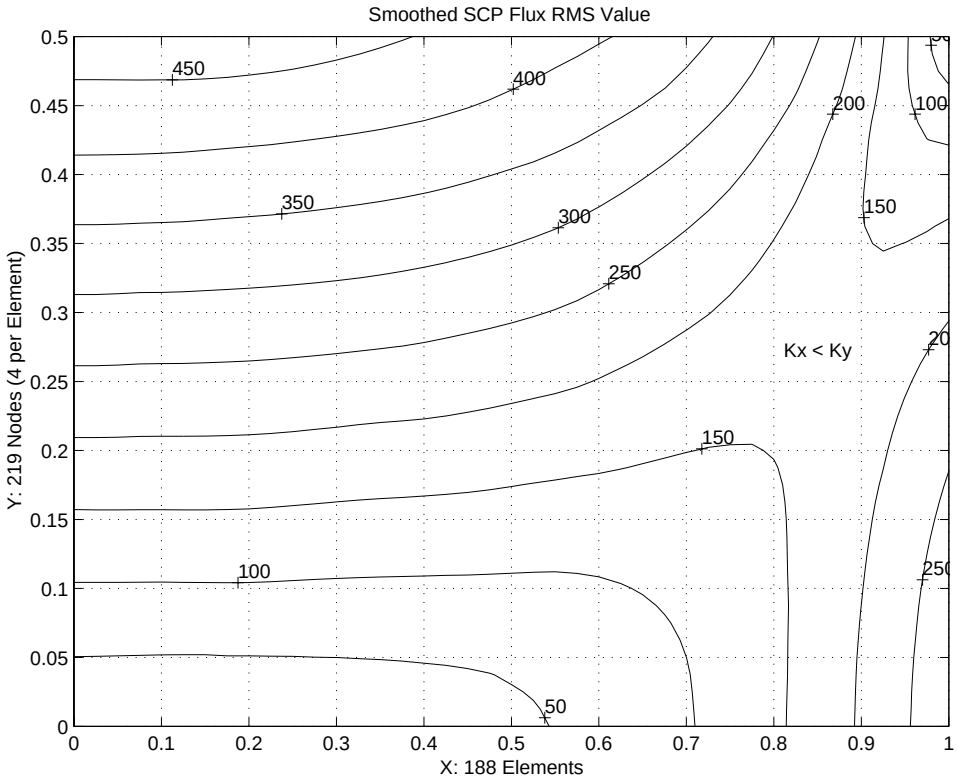

Figure 11.10.10 SCP averaged *Kx* < *Ky* heat flux value contours

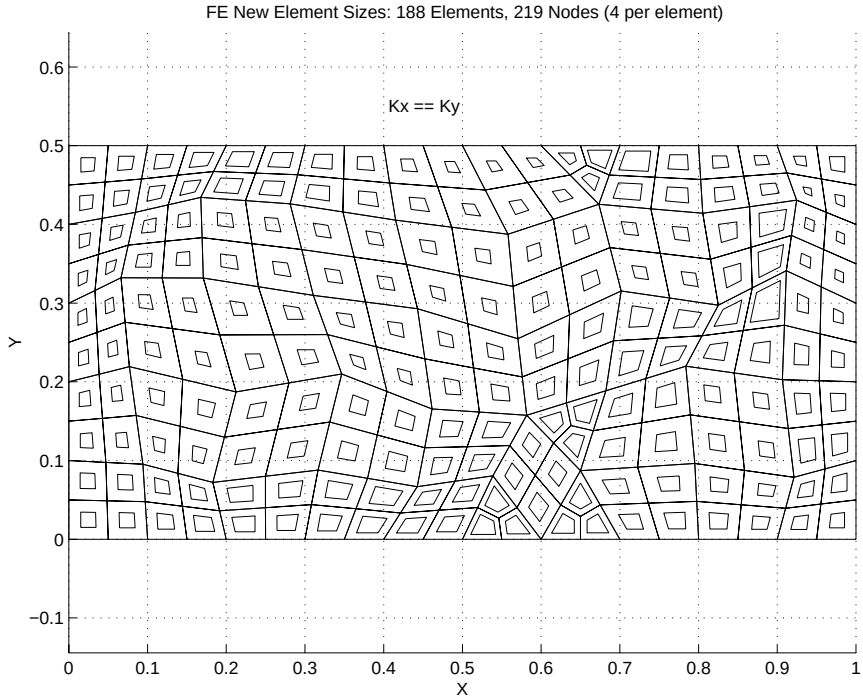

Figure 11.10.11 Suggested new mesh for isotropic material

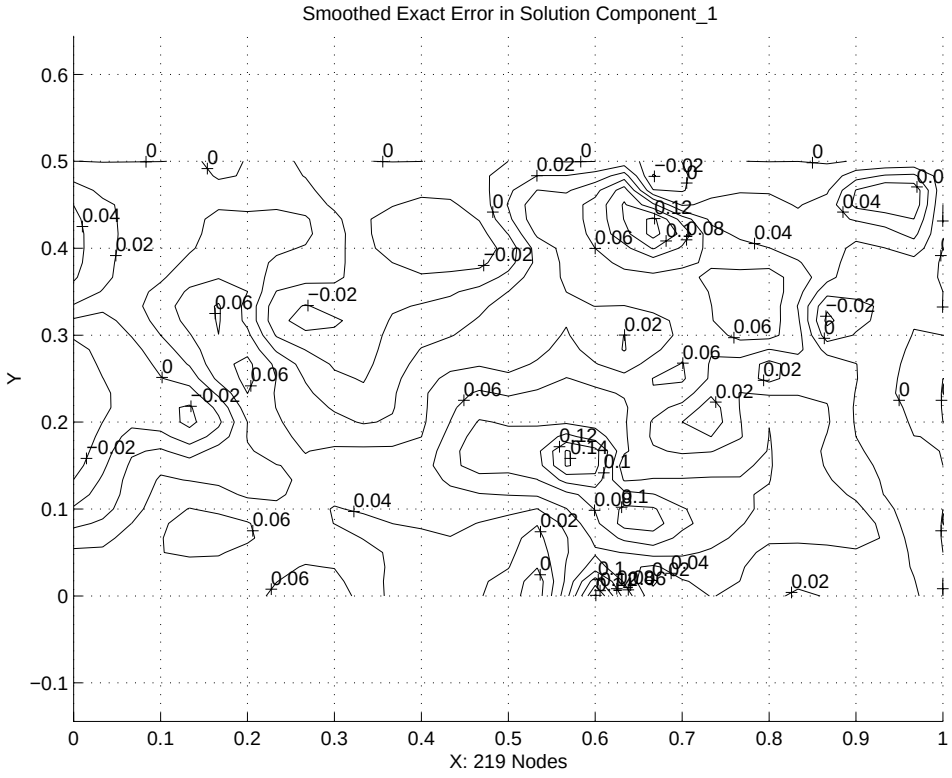

Figure 11.10.12 Exact error in the solution value, *Kx < Ky*

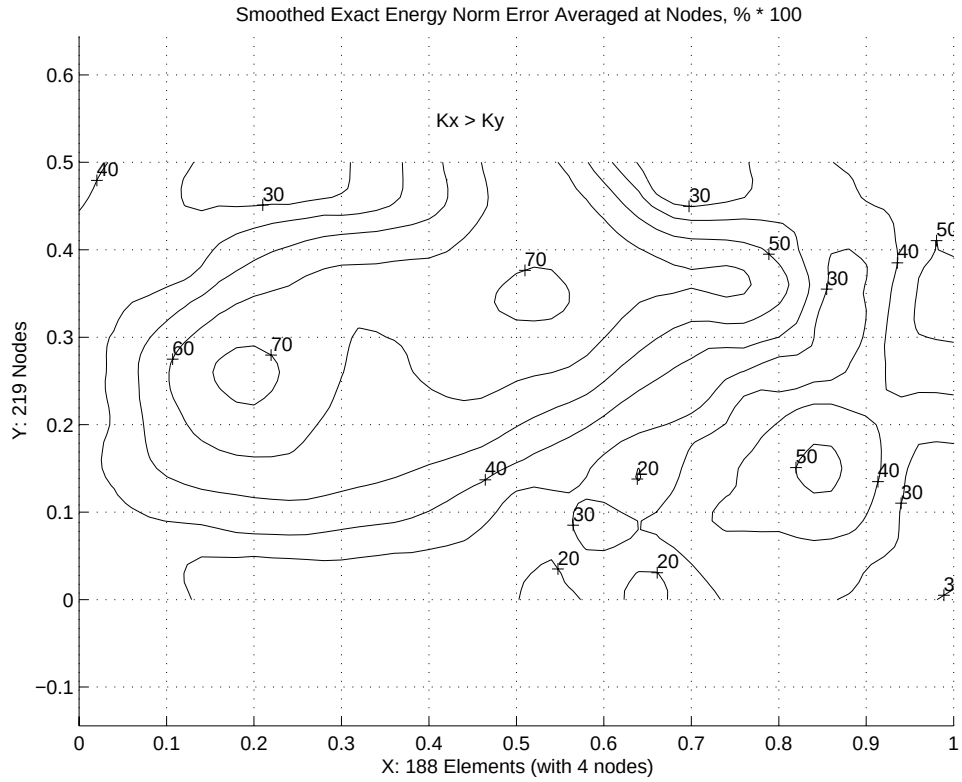

Figure 11.10.13 Exact error in the energy norm, *Kx > Ky*

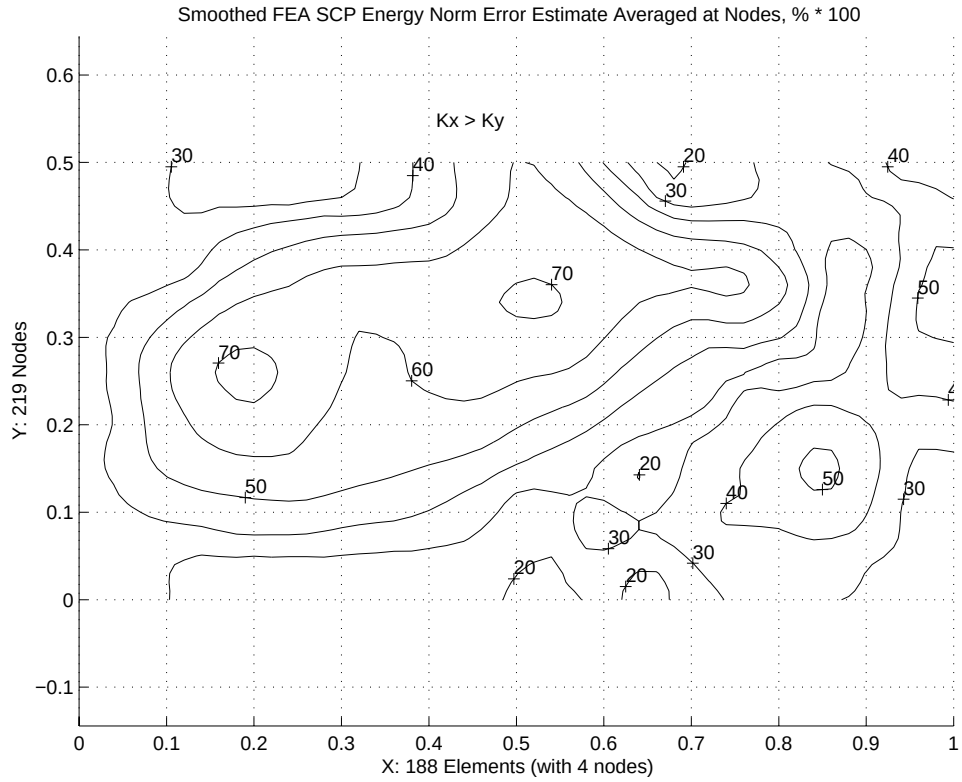

Figure 11.10.14 SCP estimated error in the energy norm, *Kx > Ky*

11.11 Axisymmetric Conductionns

For a general steady state orthotropic axisymmetric material we just recast the governing differential equation, Eq. 11.2, into cylindrical coordinates:

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

or re-arranging

$$\frac{\partial}{\partial r}\,(k_r r\, \frac{\partial \theta}{\partial r}) + \frac{\partial}{\partial z}\,(k_z r\, \frac{\partial \theta}{\partial z}) + Q\, r = 0 \tag{11.28}$$

Comparing this form to Eq. 11.2 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 11.10-11, except 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 the corresponding axisymmetric measures of $2\pi\, r\, da$ and $2\pi\, r\, ds$. Namely,

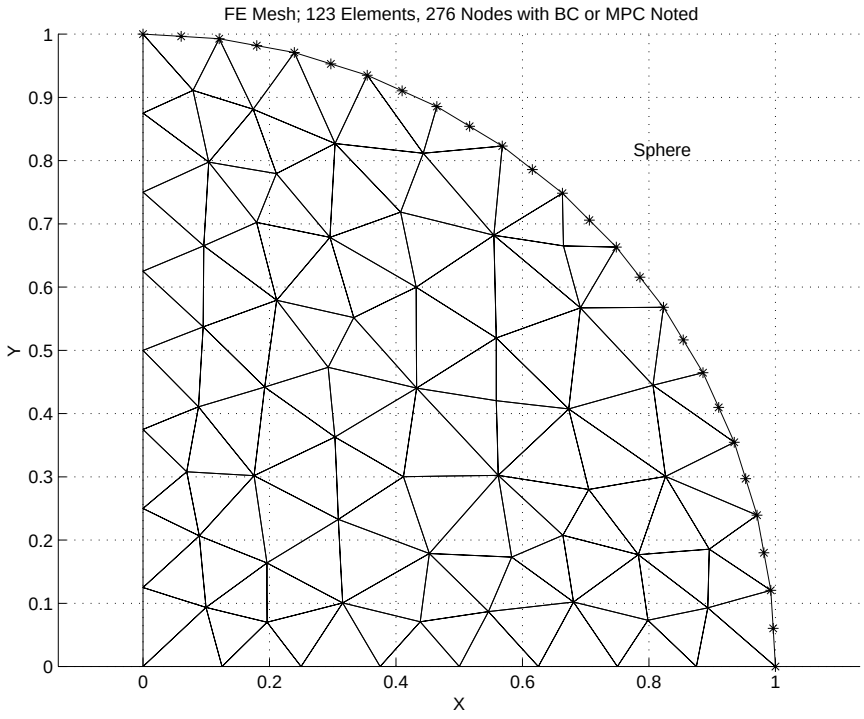

Figure 11.11.1 Half sphere model and boundary condition nodes

$$\mathbf{S}^e = \int_{A^e} (k_r^e \, \mathbf{H}_r^{e^T} \mathbf{H}_r^e + k_z^e \, \mathbf{H}_z^{e^T} \mathbf{H}_z^e) \, 2\pi \, r \, da \tag{11.29}$$

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

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

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

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 11.8.1-3 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 specified to be unity at the north and south poles and decreases to zero at the equator. We employ a half-symmetry model as shown in Fig. 11.11.1. 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, a cubic polynomial requires four points. Near the surface the Jacobian is not constant so a seven-point rule is used.

Portions of the input data are shown in Fig. 11.11.2 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. 11.11.3. 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. 11.11.4-5. 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. 11.11.6.


```
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
  4     0  0.00000E+00  9.37500E-01                                !25
  5     0  9.98188E-02  9.51896E-01                                !26
    ...                                                            !27
271     0  7.98446E-01  7.32400E-02 ! node bc_flag, R, Z           !28
272     0  8.75000E-01  0.00000E+00                                !29
273     0  8.36723E-01  3.66200E-02                                !30
274     0  7.50000E-01  0.00000E+00                                !31
275     0  7.74223E-01  3.66200E-02                                !32
276     0  8.12500E-01  0.00000E+00 ! last node                    !33
  1   272   259   262   261   260   264 ! elem, 6 nodes            !34
  2   274   272   271   276   273   275                            !35
  3   268   274   271   270   275   269                            !36
  4   259   246   262   248   247   260                            !37
  5   265   274   268   267   270   266                            !38
    ...                                                            !39
120    45    17    21    18    16    22 ! elem, 6 nodes           !40
121    37    45    21    36    22    23                           !41
122    33    21    10    20    11    12                           !42
123    21     6    10     7     5    11 ! last elem               !43
  2     1   1.00000E+00 ! node, dof, exact bc value               !44
  1     1   9.93937E-01                                           !45
  6     1   9.85471E-01                                           !46
  8     1   9.65196E-01                                           !47
 17     1   9.42728E-01                                           !48
 19     1   9.09384E-01                                           !49
 29     1   8.74255E-01                                           !50
    ...                                                            !51
201     1   8.94005E-02 ! node, dof, exact bc value               !52
222     1   5.72720E-02                                           !53
224     1   3.35888E-02                                           !54
246     1   1.45291E-02                                           !55
248     1   4.84775E-03                                           !56
259     1   1.00000E+00 ! last EBC                                !57
1 1. 1. 0. 0. ! el kr kz krz Q   (homogeneous elements)           !58
```

Figure 11.11.2 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 11.11.3 Selected sphere output results

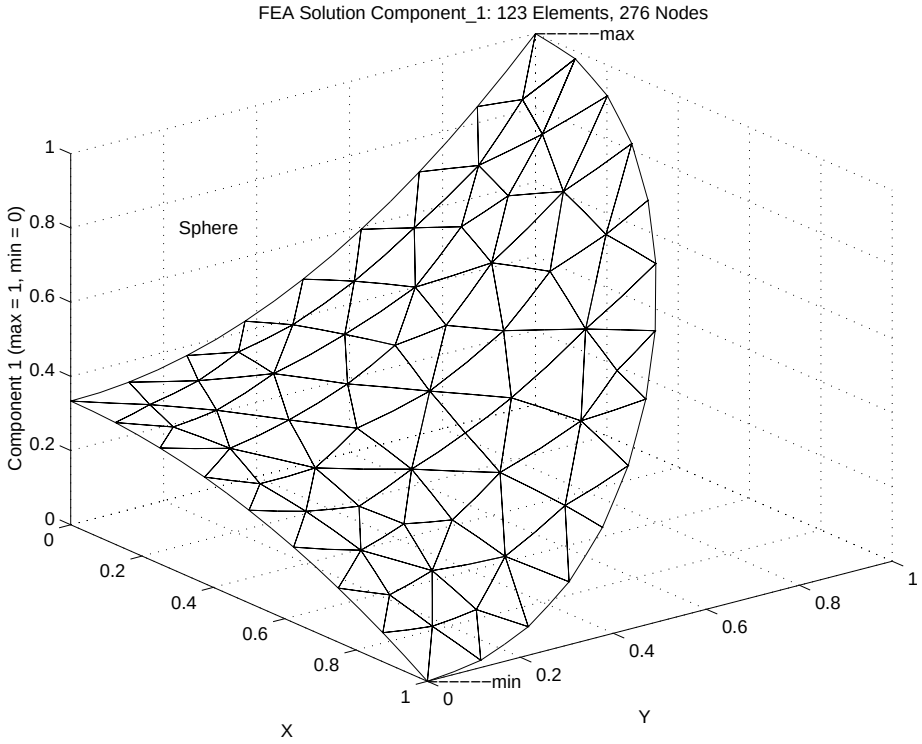

Figure 11.11.4 Carpet plot of finite element temperature

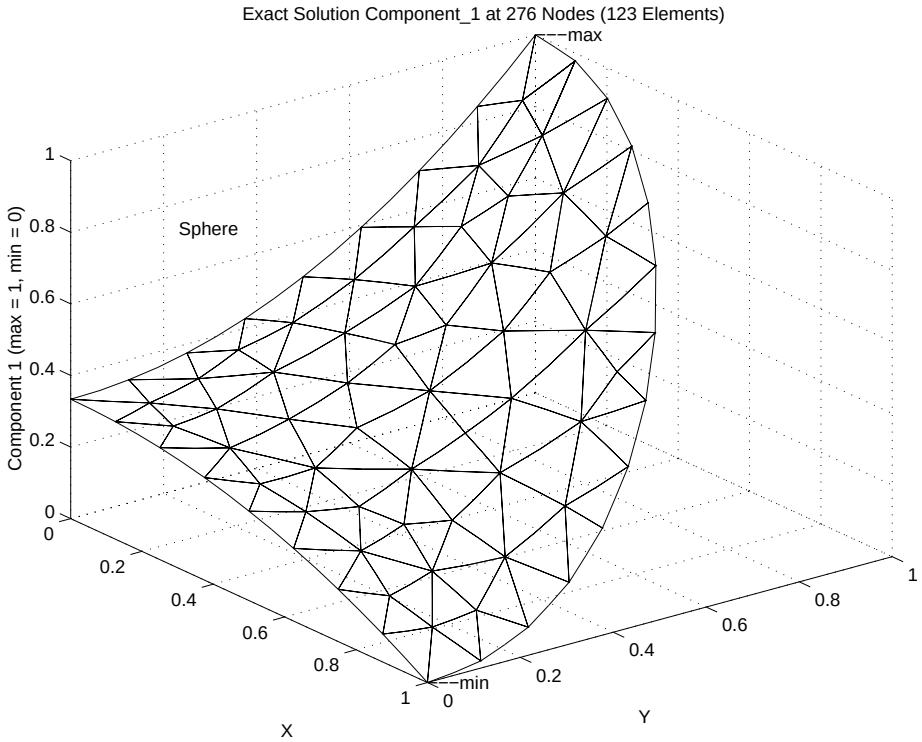

Figure 11.11.5 Carpet plot of exact temperature

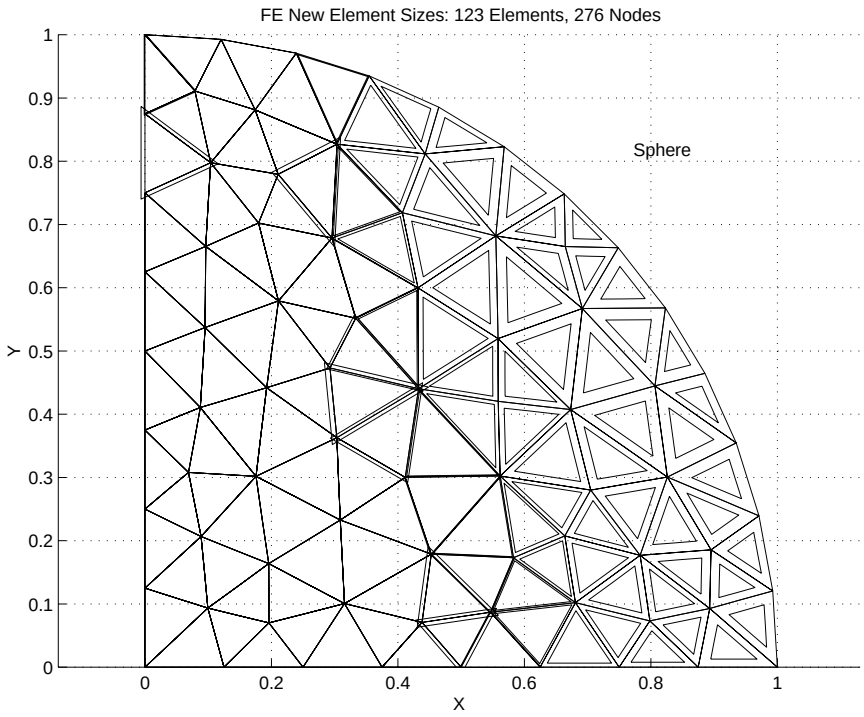

Figure 11.11.6 Element sizes for next mesh adaption

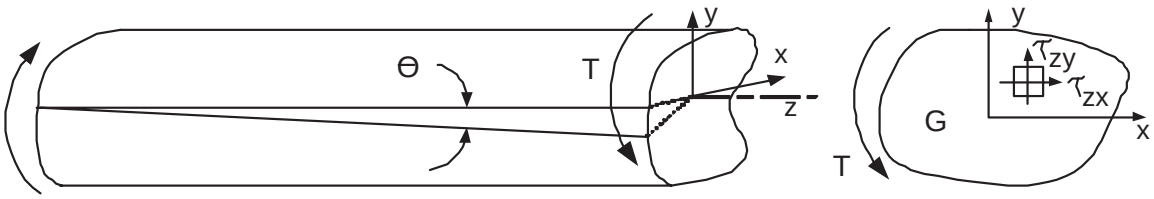

Figure 11.12.1 Torsion of a constant cross-section bar

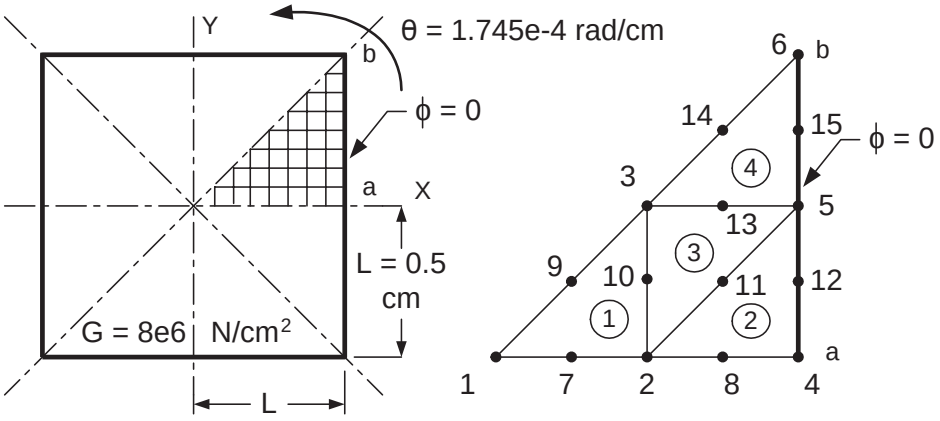

Figure 11.12.2 Torsion of a square shaft

11.12 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 approach illustrated here is only valid for singly connected domains (cross-sections without holes). If holes exist in the cross-section one must either use a different formulation or employ a numerical trick where the hole interiors are crudely meshed and assigned a material with a shear modulus, G, that is nearly zero. The case in point here is the torsion of a straight elastic bar of arbitrary cross-sectional area, shown in Fig. 11.12.1. 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:

$$\frac{1}{G}\frac{\partial^2\Phi}{\partial x^2} + \frac{1}{G}\frac{\partial^2\Phi}{\partial y^2} + 2\Theta = 0 \tag{11.30}$$

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

$$T = 2\int_{\Omega} \Phi d\Omega. \tag{11.31}$$

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 11.4.1 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 [21] presents a detailed solution of the torsion of a square bar shown in Fig. 11.12.2. 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. 11.12.3. 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 involving bars of more that one material. The computed stress function amplitude is shown in Fig. 11.12.4 and the corresponding average error estimate is in Fig. 11.12.5. 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. 11.12.6, while their nodal average values are given in Fig. 11.12.7. Our analogy and handbook equations cite the mid-point *a* as the point of maximum shear stress. The handbook stress value is $\tau_{\max} = T / (1.664 L^3)$ where T is the applied torque. The torque value is found by integrating the stress function in a post-processing step and this gives $T = 8(24.41\ N\ cm) = 195.3\ N\ cm$. Thus the estimated maximum shear stress is $938.9\ N/cm^2$ and the finite element prediction, at node 3, is $927.8\ N/cm^2$. 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}/T_{fea}, or $250./195.3 = 1.28$ to get the true maximum shear stress of $\tau_{\max} = 1,187.7\ N/cm^2$, and a true twist angle value of $0.0002234\ radians/cm$.

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. 11.12.8. 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. 11.12.8 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. 11.12.3 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


```
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_file1        !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 11.12.3 Data for the torsion model

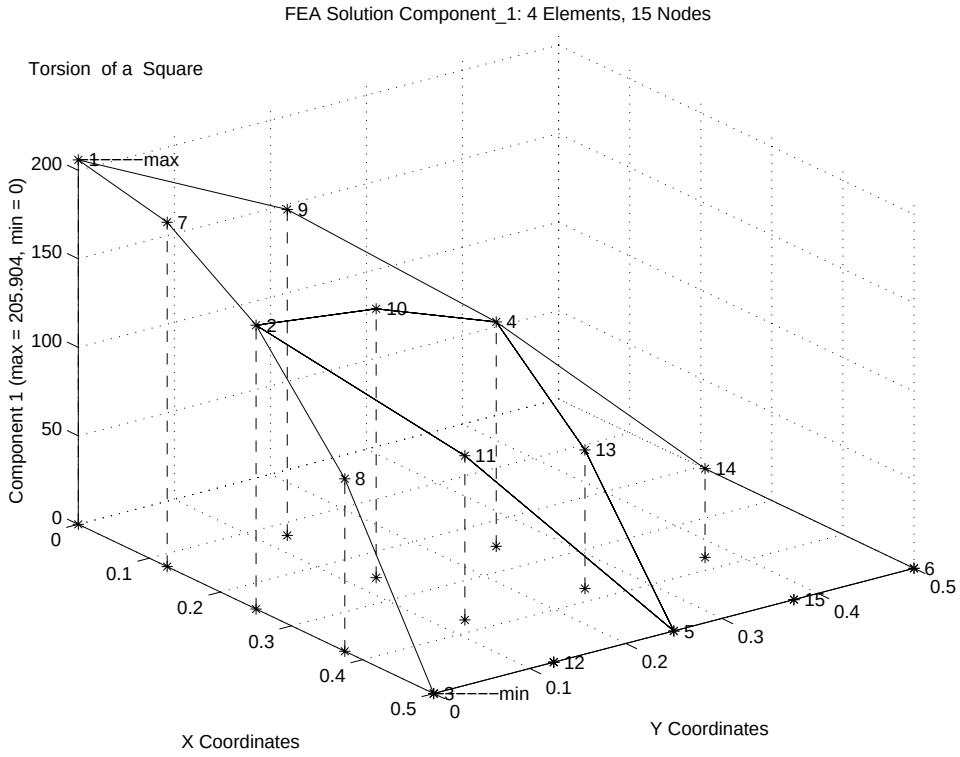

Figure 11.12.4 Stress function amplitude

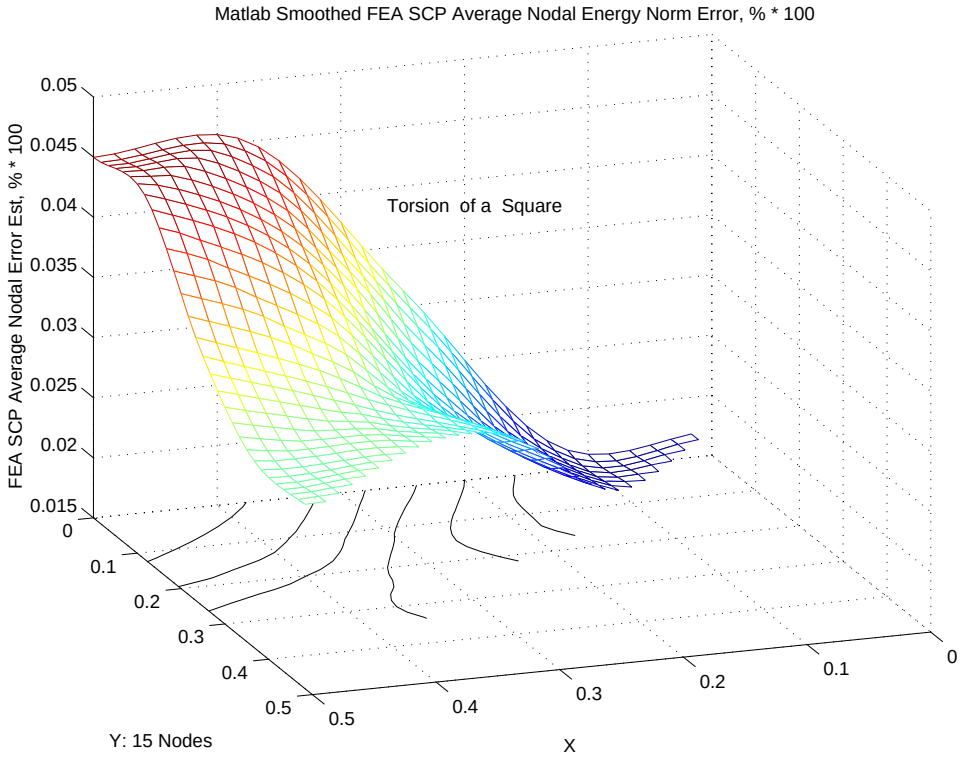

Figure 11.12.5 Estimated error in the solution

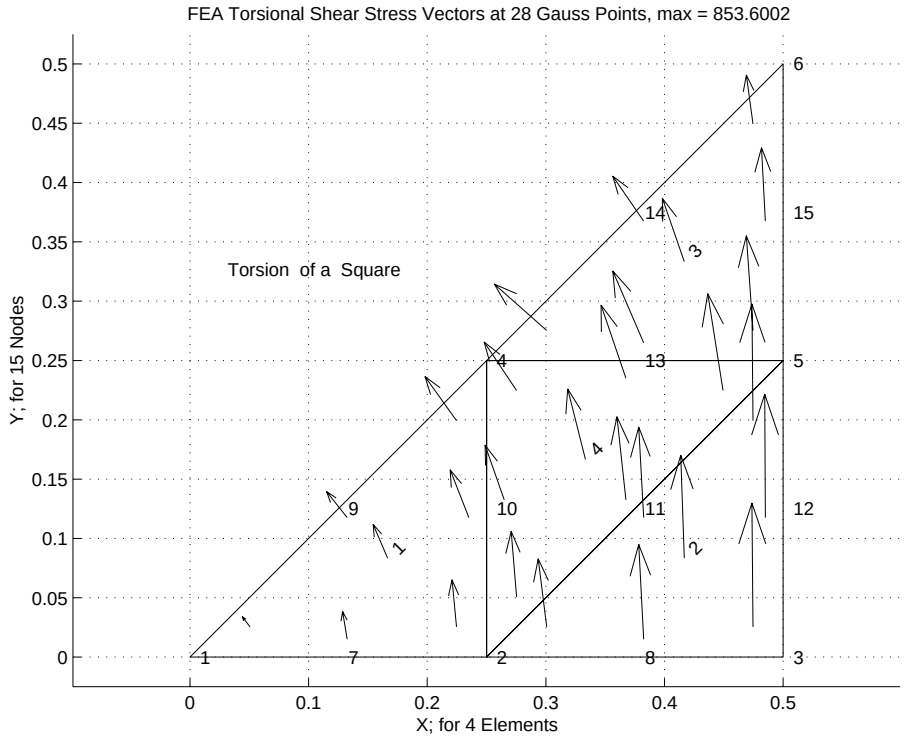

Figure 11.12.6 Shear stress vectors at the quadrature points

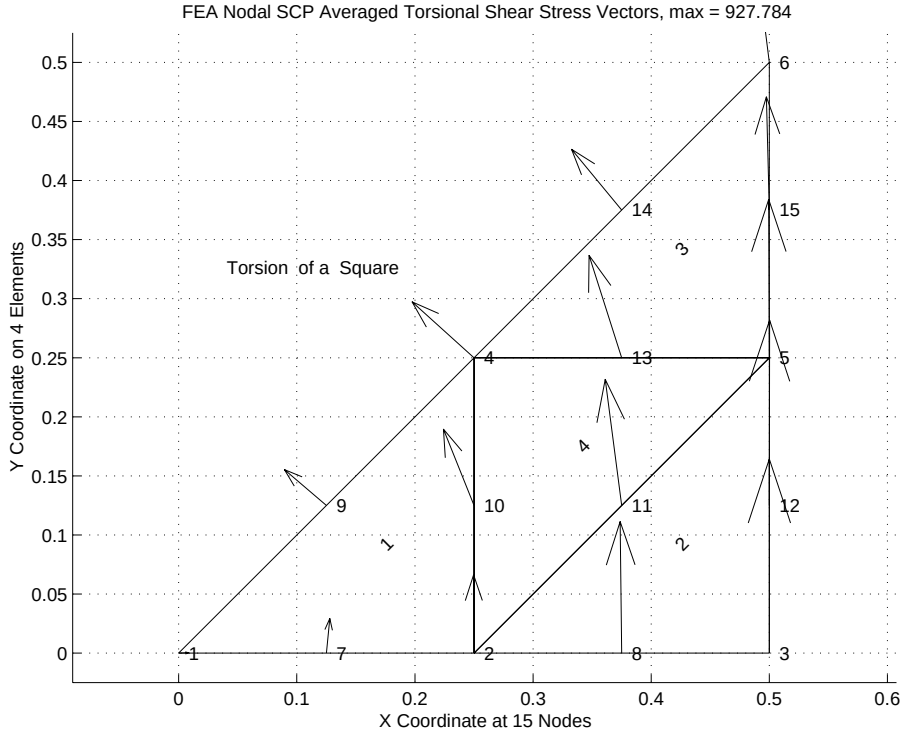

Figure 11.12.7 Shear stress vectors averaged at the nodes


```
! .......................................................    ! 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_FILE1 > 0 )  WRITE (N_FILE1)  H_INTG ! post_el keyword !46
                                                             !47
!    ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***  !48
```

Figure 11.12.8 Element matrices for torsion


```
! 205.my_post_el_inc                                                  ! 1
! ..................................................................  ! 2
!  ** POST_PROCESS_ELEM PROBLEM DEPENDENT STATEMENTS FOLLOW **       ! 3
!     Given: INTEGER,  INTENT(IN) :: N_FILE1, 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_FILE1, IE, EACH)                !12
                                                                     !13
Contains ! the local methods cited above                             !14
                                                                     !15
```

Figure 11.12.9 Encapsulating the element stress and torque recovery

programming style. The two methods are "encapsulated" within the post-processing "class" by placing them after the CONTAINS statement, as required by the language.

11.13 Introduction to Linear Flows

There are several classes of linear flow models that can be cast as a finite element model. Other flows are highly nonlinear and require much more advanced finite element methods and nonlinear equation solvers. Probably the most common example is the solution of the Navier- Stokes equations for fluid flow. There are several successful finite element formulations of computational fluid dynamics (CFD). Since they are nonlinear and require an iterative solution several implementations begin with a linear flow approximation that satisfies the continuity equation (mass conservation). Thus, there continues to be a need to have efficient linear flow models. Linear flow problems include a wide range of applications like potential flow, flow through porous media, lubrication flow, creeping viscous flows, all of which are elliptic in nature, and other classes like transonic potential flow that changes from elliptic to hyperbolic in nature as the Mach number increases. The subsonic flow of an ideal gas reduces to a nonlinear Poisson equation but it is often reduced to a linearized theory that gives a near singular Laplace equation usually known as the Prandl-Glauert equation. Here we will review some of the common examples of linear flows solved by finite element methods.

11.14 Potential Flow

A common class of problem which can be formulated in terms of the Poisson equation is that of potential flow of ideal fluids. That is, we wish to model an invisid, irrotational, incompressible, steady state flow. Potential flow can be formulated in terms of the velocity potential, ϕ, or the stream function, ψ. The latter sometimes yields simpler boundary conditions, but ϕ will be utilized here since it can be easily extended to three dimensions while the stream function is quite difficult to generalize to three dimensions. For the velocity potential formulation the diffusion coefficients, K_x and K_y, become the fluid mass density, which is assumed to be constant. The source term, G,


```
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 11.12.10 Element shear stress recovery option


```
SUBROUTINE LIST_ELEM_TORQUE_INTEGRAL (N_FILE, IE, EACH)   ! 78
! * * * * * * * * * * * * * * * * * * * * * * * * * * *   ! 79
!  LIST INTEGRAL OF TORQUE FROM H INTEGRAL, ON N_FILE     ! 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_FILE, 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_FILE  = 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_FILE, 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 11.12.11 Element torque recovery option

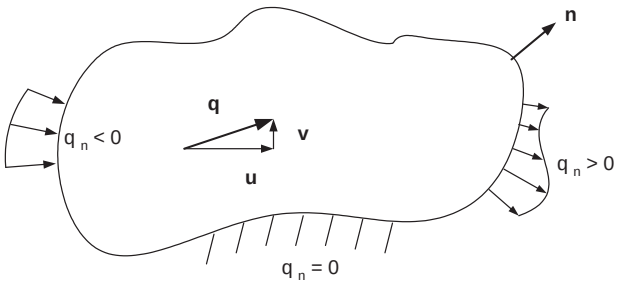

Figure 11.14.1 Defining potential flow terms

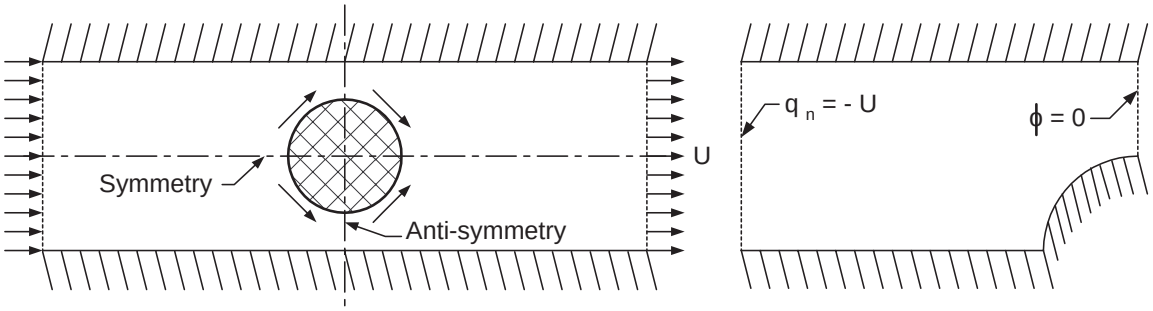

Figure 11.14.2 Typical boundary condition considerations

represents a source or sink term and is usually zero. In that common case the constant density term could be divided out so that the problem of potential flow is often presented as a solution of Laplace's equation rather than the Poisson equation. However, there are practical applications that merit retaining the Poisson form. The governing equation is

$$\rho\left(\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}\right) = Q, \tag{11.32}$$

where Q is a source or sink. The velocity potential, ϕ is usually of secondary interest and the analyst generally requires information on the velocity components. They are defined by the global derivatives of ϕ:

$$u \equiv \frac{\partial\phi}{\partial x}, \qquad v \equiv \frac{\partial\phi}{\partial y} \tag{11.33}$$

where u and v denote the x- and y-components of the velocity vector, **q**, in the plane of analysis, as shown in Fig. 11.14.1. Thus, although the program will yield the nodal values of ϕ one must also calculate the above global derivatives. This can be done economically since these derivative quantities must be generated at each quadrature point during construction of the element square matrix, **S**. Hence, one can simply store this derivative information, i.e., matrix **DGH**, and retrieve it later for calculating the global derivatives of ϕ. Another item of interest in Fig. 11.14.1 is the normal boundary flow into or out of the domain, $q_n = \partial\phi / \partial n$. The natural boundary condition is zero flux which represents an impervious wall. The use of these items to establish the boundary conditions in a simple flow domain is illustrated in Fig. 11.14.2. There on the vertical line of anti-symmetry we see that the vertical component of velocity, v, will be zero so $\partial\phi/\partial y = 0$ along the line of x constant, so we can set ϕ to an arbitrary constant.

Patch Test

We wish to test this Poisson equation solver by running a patch test. Originally based on engineering judgement, the *patch test* has been proven to be a mathematically valid convergence test [13, 14]. Consider a patch (or sub-assembly) of finite elements containing at least one internal node. An internal node is one completely surrounded by elements. Let the problem be formulated by an integral statement containing derivatives of order n. Assume an arbitrary function, $P(x)$, whose n^{th} order derivatives are constant. Use this function to prescribe the dependent variable on all external nodes of the patch

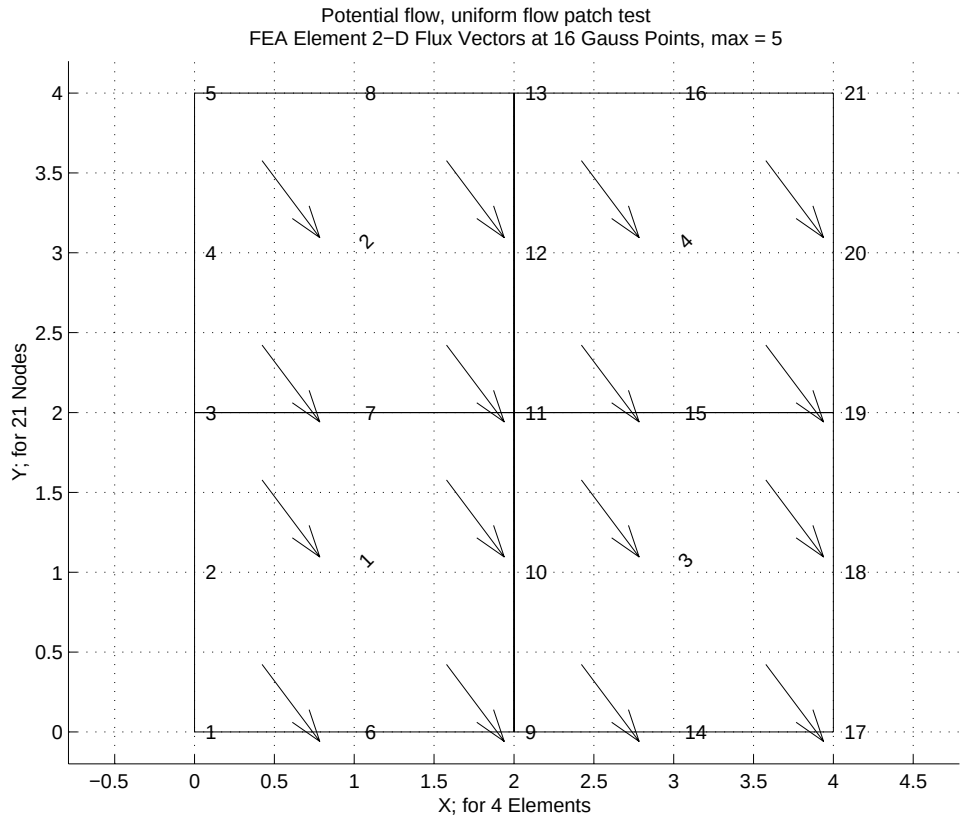

Figure 11.14.3 A uniform flow patch test

(i.e., $\phi_e = P(x_e)$). Solve for the internal nodal values of the dependent variable, ϕ_I, and its n^{th} order derivatives in each element. To be a convergent formulation:

1. The internal nodal values must agree with the assumed function evaluated at the internal points (i.e., $\phi_I = P(x_I)$?); and
2. The calculated n^{th} order derivatives *must* agree with the assumed constant values. In this application that means that all the velocity vectors are identical. Clearly that represents some inclined uniform flow state, as seen in Fig. 11.14.3.

It has been found that some non-conforming elements will yield convergent solutions for only one particular mesh pattern. The patch mesh should be completely arbitrary for a valid numerical test. The patch test is very important from the engineering point of view since it can be executed numerically. Thus, one obtains a numerical check of the entire program used to formulate the patch test.

The patch of elements shown in Fig. 11.14.3 was utilized. It was assumed that

$$\phi(x, y) \equiv 1 + 3x - 4y \tag{11.34}$$

such that the derivatives $\phi_{,x} = 3$, $\phi_{,y} = -4$ are constant everywhere. All 16 points on the exterior boundary were assigned values by substituting their coordinates into the

above relation. that is, the boundary conditions that $\phi_1 \equiv \phi(x_1, y_1)$, etc., were applied on the exterior boundary. Then the problem was solved numerically to determine the value of ϕ at the interior points (7, 10, 11, 12, 15) and the values of its global derivatives at each integration point. The output results of this patch test are shown in Fig. 11.14.4. The output shows clearly that the global derivatives at all integration points have the assumed values. It is also easily verified that all the interior nodal values of ϕ are consistent with the assumed form. Thus, the patch test is satisfied and the subroutines pass a necessary numerical test. It is also reassuring that the error estimator indicates that even this crude mesh does not need refinement.

Example — Flow Around a Cylinder

Martin and Carey [16] were among the first to publish a numerical example of a finite element potential flow analysis. This same example is also discussed by others [6, 8]. The problem considers the flow around a cylinder in a finite rectangular channel with a uniform inlet flow. The geometry is shown in Fig. 11.14.5. The present mesh is compared with that of Martin and Carey in Fig. 11.14.6. By using centerline symmetry and midstream antisymmetry it is possible to employ only one fourth of the flow field. The stream function boundary conditions are discussed by Martin and Carey [16] and Chung [6]. For the velocity potential one has four sets of Neumann (boundary flux) conditions and one set of Dirichlet (nodal parameter) conditions. The first involve zero normal flow, $q_n = \phi_{,n} \equiv 0$, along the centerline *ab* and the solid surfaces *bc* and *de*, and a uniform unit inflow, $q_n \equiv -1$, along *ad*. At the mid-section, *ce*, antisymmetry requires that $v = 0$. Thus, $\phi_{,y} = 0$ so that $\phi = \phi(x)$, but in this special case x is constant along that line so we can set ϕ to any desired constant, say zero, along *ce*. The output results are illustrated in Fig. 11.14.7. The velocity vector plots are shown in Fig. 11.14.8a. By changing the inlet flux conditions, other problems can be solved.

As before, the presence of a boundary flux, q_n, makes it necessary to evaluate the flux column matrix

$$\mathbf{C}^b = \int_{L^b} \mathbf{H}^{bT} \, q_n \, ds. \tag{11.35}$$

The variation of q_n along the boundary segment is assumed to be defined by the nodal (input) values and the segment interpolation equations, i.e.,

$$q_n(s) \equiv \mathbf{H}^b(s) \, \mathbf{q}_n^b.$$

Therefore, the segment column matrix (for a straight quadratic segment) becomes

$$\mathbf{C}^b = \int_{L^b} \mathbf{H}^{bT}\mathbf{H}^b \, ds \, \mathbf{q}_n^b = \frac{L^b}{30}\begin{bmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{bmatrix} \mathbf{q}_n^b. \tag{11.36}$$


```
TITLE: "Potential flow patch test, uniform flow"          ! 1
                                                          ! 2
***  REACTION RESULTANTS  ***                             ! 3
PARAMETER,    SUM          POSITIVE     NEGATIVE         ! 4
 DOF_1,      3.5527E-15   2.6000E+01  -2.6000E+01         ! 5
                                                          ! 6
***  OUTPUT OF RESULTS IN NODAL ORDER  ***                ! 7
    NODE,  X-Coord,    Y-Coord,    DOF_1,                ! 8
       1  0.0000E+00  0.0000E+00  1.0000E+00             ! 9
    . . .                                                !10
       7  1.0000E+00  2.0000E+00 -4.0000E+00             !11
       8  1.0000E+00  4.0000E+00 -1.2000E+01             !12
       9  2.0000E+00  0.0000E+00  7.0000E+00             !13
      10  2.0000E+00  1.0000E+00  3.0000E+00             !14
      11  2.0000E+00  2.0000E+00 -1.0000E+00             !15
      12  2.0000E+00  3.0000E+00 -5.0000E+00             !16
      13  2.0000E+00  4.0000E+00 -9.0000E+00             !17
      14  3.0000E+00  0.0000E+00  1.0000E+01             !18
      15  3.0000E+00  2.0000E+00  2.0000E+00             !19
    . . .                                                !20
      21  4.0000E+00  4.0000E+00 -3.0000E+00             !21
                                                         !22
*** FLUX COMPONENTS AT ELEMENT INTEGRATION POINTS ***    !23
ELEMENT, PT, X-Coord,   Y-Coord,    FLUX_1,    FLUX_2,   !24
      1  1  4.2265E-1  4.2265E-1  3.0000E+0 -4.0000E+0   !25
      1  2  1.5774E+0  4.2265E-1  3.0000E+0 -4.0000E+0   !26
      1  3  4.2265E-1  1.5774E+0  3.0000E+0 -4.0000E+0   !27
      1  4  1.5774E+0  1.5774E+0  3.0000E+0 -4.0000E+0   !28
    . . .                                                !29
      4  1  2.4226E+0  2.4226E+0  3.0000E+0 -4.0000E+0   !30
      4  2  3.5774E+0  2.4226E+0  3.0000E+0 -4.0000E+0   !31
      4  3  2.4226E+0  3.5774E+0  3.0000E+0 -4.0000E+0   !32
      4  4  3.5774E+0  3.5774E+0  3.0000E+0 -4.0000E+0   !33
                                                         !34
*** SUPER_CONVERGENT AVERAGED NODAL FLUXES ***           !35
NODE,   X-Coord,    Y-Coord,    FLUX_1,     FLUX_2,      !36
     1  0.0000E+00  0.0000E+00  3.0000E+00 -4.0000E+00   !37
     7  1.0000E+00  2.0000E+00  3.0000E+00 -4.0000E+00   !38
    . . .                                                !39
     8  1.0000E+00  4.0000E+00  3.0000E+00 -4.0000E+00   !40
     9  2.0000E+00  0.0000E+00  3.0000E+00 -4.0000E+00   !41
    10  2.0000E+00  1.0000E+00  3.0000E+00 -4.0000E+00   !42
    11  2.0000E+00  2.0000E+00  3.0000E+00 -4.0000E+00   !43
    12  2.0000E+00  3.0000E+00  3.0000E+00 -4.0000E+00   !44
    13  2.0000E+00  4.0000E+00  3.0000E+00 -4.0000E+00   !45
    14  3.0000E+00  0.0000E+00  3.0000E+00 -4.0000E+00   !46
    15  3.0000E+00  2.0000E+00  3.0000E+00 -4.0000E+00   !47
    . . .                                                !48
    21  4.0000E+00  4.0000E+00  3.0000E+00 -4.0000E+00   !49
                                                         !50
 ***  S_C_P  ENERGY  NORM  ERROR  ESTIMATE  DATA ***     !51
              ERROR IN        % ERROR IN      REFINEMENT !52
 ELEMENT,     ENERGY_NORM,    ENERGY_NORM,    PARAMETER  !53
       1      4.5626E-15      2.2813E-14      4.5626E-14 !54
       2      7.2788E-15      3.6394E-14      7.2788E-14 !55
       3      6.5704E-15      3.2852E-14      6.5704E-14 !56
       4      8.2159E-15      4.1080E-14      8.2159E-14 !57
```

Figure 11.14.4 Numerical results from standard patch test

11.15 Axisymmetric Plasma Equilibria *

Nuclear fusion is being developed as a future source of energy. The heart of the fusion reactors will be a device for confining the reacting plasma and heating it to thermonuclear temperatures. This confinement problem can be solved through the use of magnetic fields of the proper geometry which generate a so-called "magnetic bottle". The tokamak containment concept employs three magnetic field components to confine the plasma. An externally applied toroidal magnetic field, B_T, is obtained from coils through which the torus passes. A second field component is the polodial magnetic field, B_P, which is produced by a large current flowing in the plasma itself. This current is induced in the plasma by transformer action and assists in heating the plasma. Finally, a vertical (axial) field, B_V, is also applied. These typical fields are illustrated in Fig. 11.15.1. For many purposes a very good picture of the plasma behavior can be obtained by treating it as an ideal magnetohydrodynamic (MHD) media. The equations governing the steady state flow of an ideal MHD plasma are

$$grad\ \mathbf{B} = 0, \quad \nabla \mathrm{P} = \mathbf{J} \times \mathbf{B}, \quad \mathrm{curl}\ \mathbf{B} = \mu \mathbf{J}$$

where P is the pressure, **B** the magnetic flux density vector, **J** the current density vector, and μ a constant that depends on the system of units being employed. Consider an axisymmetric equilibria defined in cylindrical coordinates (r, z, θ) so that $\partial / \partial \theta = 0$. This implies the existence of a vector potential, **A**, such that curl **A** = **B**. Assuming that **A** = **A**(r, z) and $A_\theta = \psi / r$, where ψ is a stream function, we obtain

$$B_r = -\psi_{,z}/r, \quad B_z = \psi_{,r}/r, \quad B_\theta = A_{r,z} - A_{z,r} = B_T$$

Therefore the governing equation simplifies to

$$\frac{\partial^2 \psi}{\partial r^2} - \frac{1}{r}\frac{\partial \psi}{\partial r} + \frac{\partial^2 \psi}{\partial z^2} = -\mu r^2 P' - X X' = r J_\theta, \tag{11.37}$$

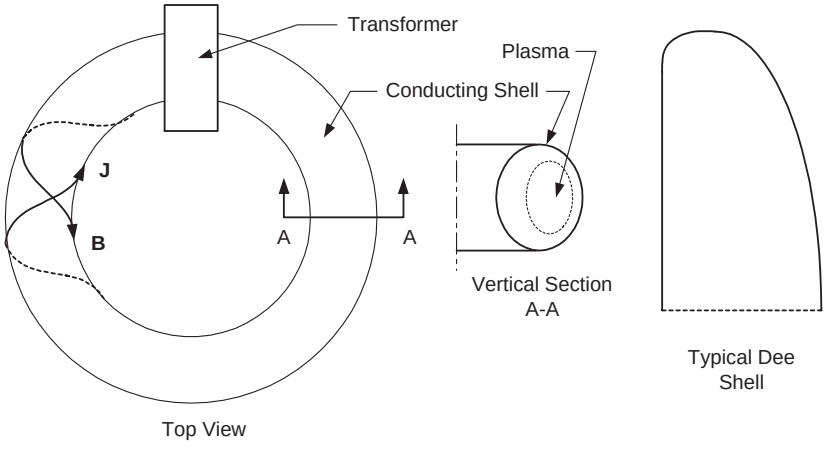

Figure 11.15.1 Schematic of tokamak fields and currents

where J_θ is the plasma current, P is the pressure, $X = r\,B_\theta$ and where P and X are functions of ψ alone. Both **J** and **B** are vectors that lie tangent to the surfaces of constant ψ. The above is the governing equation for the steady equilibrium flow of a plasma. For certain simple choices of P and B_θ, Eq. 11.37 will be linear but in general it is nonlinear. They are usually represented as a series in ψ as

$$P(\psi) = \alpha_0 + \alpha_1\psi + ... + \alpha_n\psi^n/n$$

$$X^2(\psi) = \beta_0 + \beta_1\psi + ... + \beta_n\psi^n/n$$

The essential boundary condition on the limiting surface, Γ_1, is

$$\psi = K + 1/2\, r^2\, B_V \quad on\ \Gamma_1$$

where K is a constant and B_V is a superimposed direct current vertical (z) field. On planes of symmetry one also has vanishing normal gradients of ψ, i.e.,

$$\frac{\partial\psi}{\partial n} = 0\ on\ \Gamma_2.$$

The right-hand side of Eq. 11.37 can often be written as

$$r\,J_\theta = p\psi + q \tag{11.38}$$

where, for the above special cases, $p = p(r, z)$ and $q = q(r, z)$, but where in general q is a nonlinear function of ψ, i.e., $q = q(r, z, \psi)$. Equations 11.37 and 11.38 are those for which we wish to establish the finite element model.

A finite element formulation of this problem has been presented by Akin and Wooten [1]. They recast Eq. 11.37 in a self-adjoint form, applied the Galerkin criterion, and integrated by parts. This defines the governing variational statement

$$I = \int\int_\Omega \left[\frac{1}{2}\left\{(\psi_{,r})^2 + (\psi_{,z})^2 + p\psi^2\right\} + q\psi\right]\frac{1}{r}\,dr\,dz \tag{11.39}$$

which, for the linear problem, yields Eq. 11.37 as the Euler equation when I is stationary, i.e., $\delta I = 0$. When $p = q = 0$, Eq. 11.39 also represents the case of axisymmetric inviscid fluid flow. Flow problems of this type were considered by Chung [6] using a similar procedure. For a typical element the element contributions for Eq. 11.38 are

$$\mathbf{S}^e = \int_{\Omega^e} [\mathbf{H}_{,r}^T\mathbf{H}_{,r} + \mathbf{H}_{,z}^T\mathbf{H}_{,z} + p(r, z)\mathbf{H}^T\mathbf{H}]\,\frac{1}{r}\,dr\,dz,$$

$$\mathbf{C}^e = \int_\Omega q(r, z)\mathbf{H}^T\,\frac{1}{r}\,dr\,dz. \tag{11.40}$$

Several other applications of this model to plasma equilibria are given by Akin and Wooten [1]. The major advantage of the finite element formulation over other methods such as finite differences is that it allows the plasma physicist to study arbitrary geometries. Some feel that the fabrication of the toroidal field coils may require the use of a circular plasma, while others recommend the use of dee-shaped plasmas. The current model has been applied to both of these geometries and the following figures


```
! .....................................................................   ! 1
!    ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***         ! 2
! .....................................................................   ! 3
!  FOR AXISYMMETRIC MHD PLASMA EQUILIBRIUM,    P,Q  /=  0  I.E.         ! 4
!  L(U) = (U,R*1/R),R + U,ZZ*1/R - P*U/R - Q/R = 0                       ! 5
!  U = STREAM FUNCTION                                                  ! 6
                                                                        ! 7
!  P = -C_1 * ALPHA_0 * R**2 - 0.5 * BETA_0, = 0 IF IDEAL PLASMA        ! 8
!  Q = -C_1 * ALPHA_1 * R**2 - 0.5 * BETA_1, = 0 IF IDEAL PLASMA        ! 9
                                                                        !10
! MISC PROPERTIES 1-5 ARE: C_1, ALPHA_0, ALPHA_1, BETA_0, BETA_1       !11
                                                                        !12
 REAL(DP), PARAMETER :: FOUR_PI = TWO_PI * 2.d0                         !13
 REAL(DP)            :: DET_WT, DET, P, Q, R                            !14
 REAL(DP), SAVE      :: ALPHA_0, ALPHA_1, BETA_0, BETA_1                !15
 INTEGER             :: IP, io_1                                        !16
                                                                        !17
!-->  DEFINE ELEMENT PROPERTIES (FIRST 2 TERMS IN POWERS OF U)          !18
  IF ( IE == 1 ) THEN ! first call                                      !19
    ALPHA_0 = GET_REAL_MISC (1)                                         !20
    ALPHA_1 = GET_REAL_MISC (2)                                         !21
    BETA_0  = GET_REAL_MISC (3)                                         !22
    BETA_1  = GET_REAL_MISC (4)                                         !23
  END IF                                                                !24
                                                                        !25
  CALL REAL_IDENTITY (N_R_B, E) !   DEFAULT TO IDENTITY MATRIX          !26
                                                                        !27
!      STORE NUMBER OF POINTS FOR FLUX CALCULATIONS                     !28
  CALL STORE_FLUX_POINT_COUNT ! Save LT_QP                              !29
                                                                        !30
!-->   NUMERICAL INTEGRATION LOOP                                       !31
   DO IP = 1, LT_QP                                                     !32
     H   = GET_H_AT_QP (IP)   ! EVALUATE INTERPOLATION FUNCTIONS        !33
     XYZ = MATMUL (H, COORD)  ! FIND GLOBAL COORD, (ISOPARAMETRIC)      !34
     R   = XYZ (1)            ! CHANGE NOTATION                        !35
     DLH = GET_DLH_AT_QP (IP) ! FIND LOCAL DERIVATIVES                  !36
     AJ  = MATMUL (DLH, COORD) ! FIND JACOBIAN AT THE POINT             !37
!         FORM INVERSE AND DETERMINATE OF JACOBIAN                     !38
     CALL INVERT_JACOBIAN (AJ, AJ_INV, DET, N_SPACE)                    !39
     DET_WT  = DET * WT(IP) / R                                         !40
                                                                        !41
!         EVALUATE GLOBAL DERIVATIVES, DGH == B                        !42
     DGH = MATMUL (AJ_INV, DLH)                                         !43
     B   = COPY_DGH_INTO_B_MATRIX (DGH) ! B = DGH                       !44
                                                                        !45
!      EVALUATE CONTRIBUTIONS TO SQUARE AND COLUMN MATRICES             !46
     P = - FOUR_PI * R * R * ALPHA_1 - 0.5d0 * BETA_1                   !47
     Q = - FOUR_PI * R * R * ALPHA_0 - 0.5d0 * BETA_0                   !48
                                                                        !49
     S = S + ( MATMUL ((MATMUL (TRANSPOSE (B), E)), B)   &              !50
             + P * OUTER_PRODUCT (H, H) ) * DET_WT                      !51
     C = C - Q * H * DET_WT                                             !52
                                                                        !53
!-->  SAVE COORDS, E AND DERIVATIVE MATRIX, FOR POST PROCESSING         !54
     CALL STORE_FLUX_POINT_DATA (XYZ, E, B)                             !55
   END DO                                                               !56
                                                                        !57
!    ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***           !58
```

Figure 11.15.2 Plasma element matrices evaluations

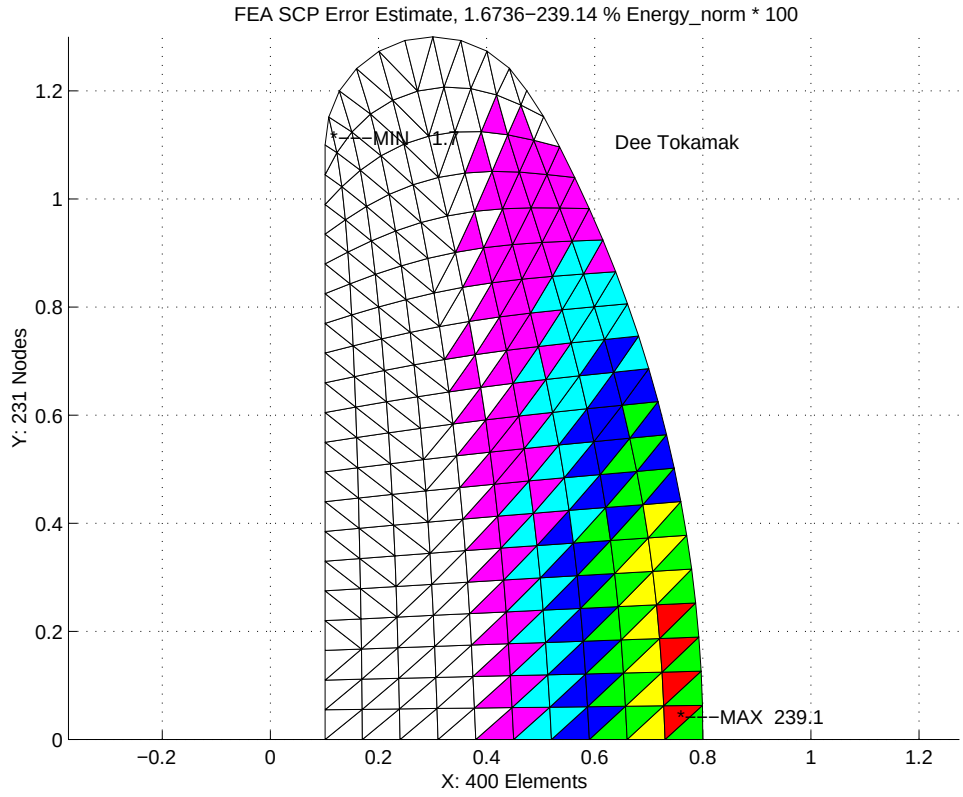

Figure 11.15.3 Initial Dee plasma mesh and error estimates

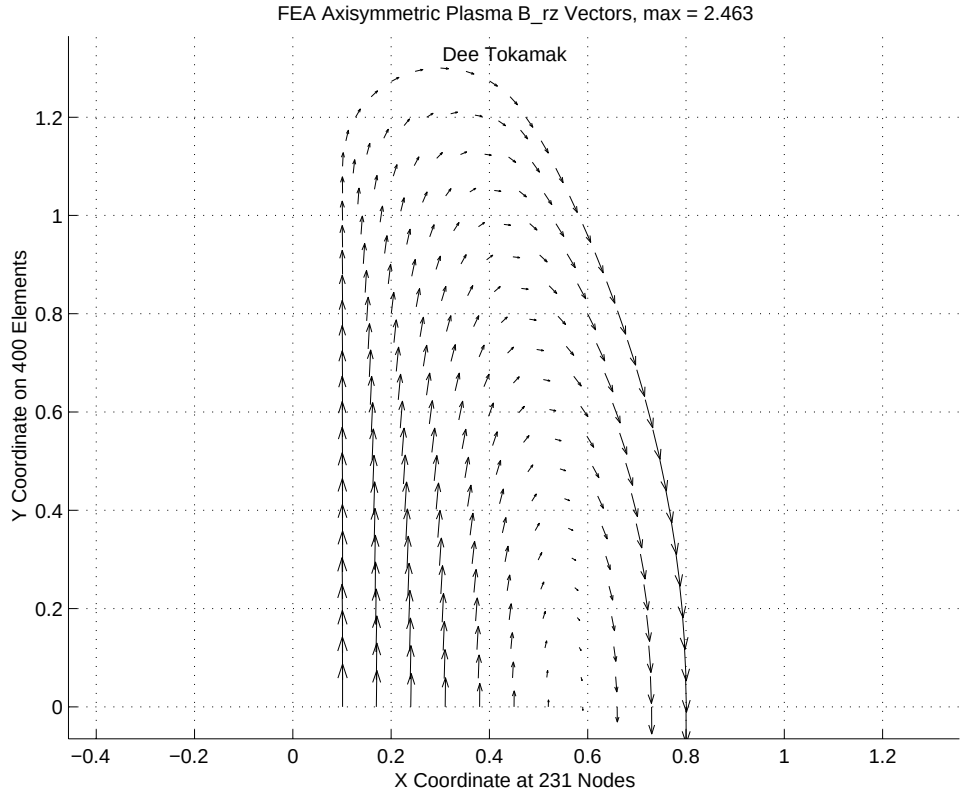

Figure 11.15.4 Initial Dee plasma planar **B** vectors

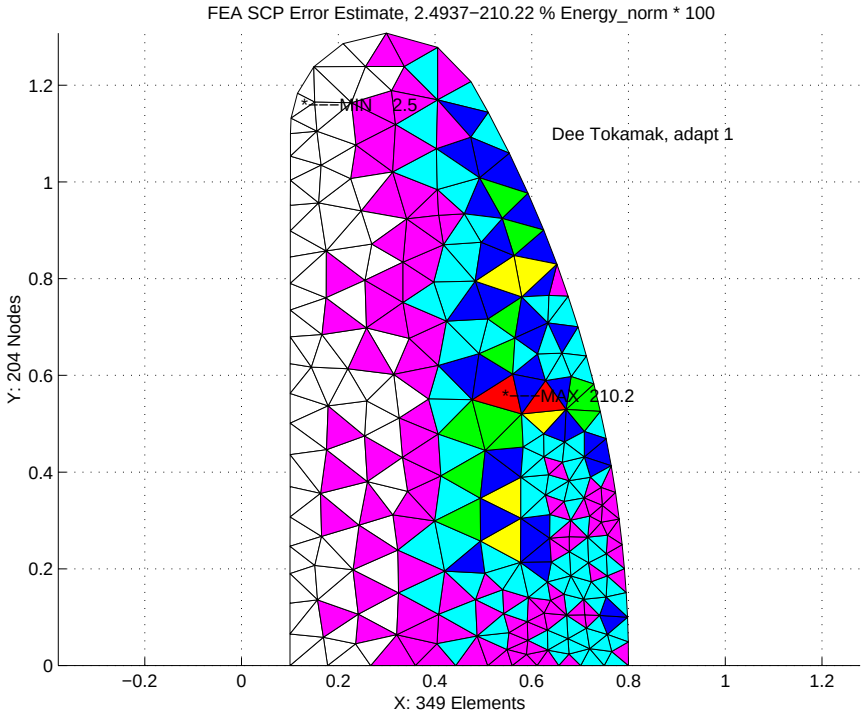

Figure 11.15.5 Plasma error estimates in first adaption

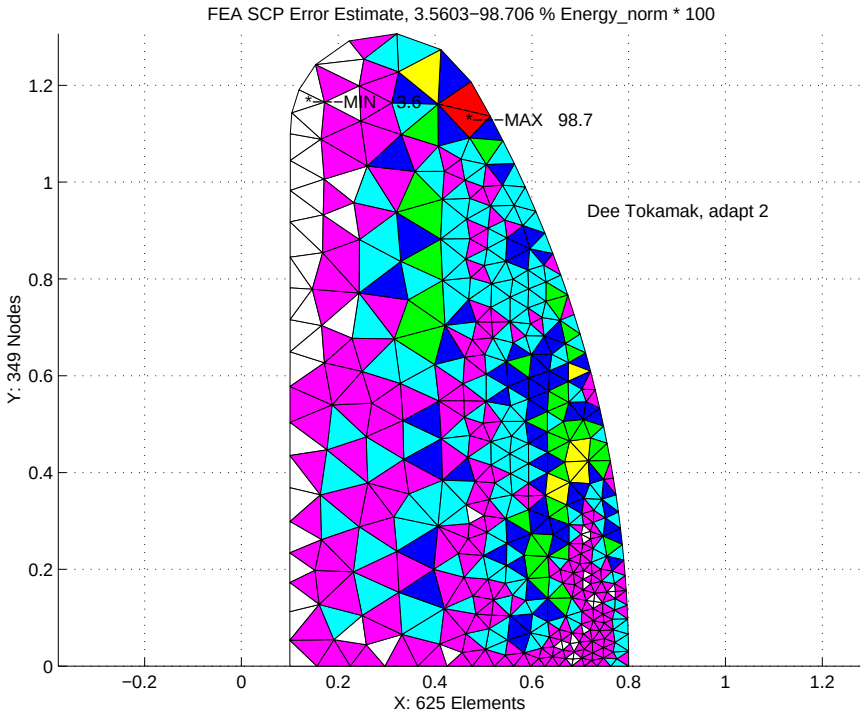

Figure 11.15.6 Plasma error estimates in second adaption

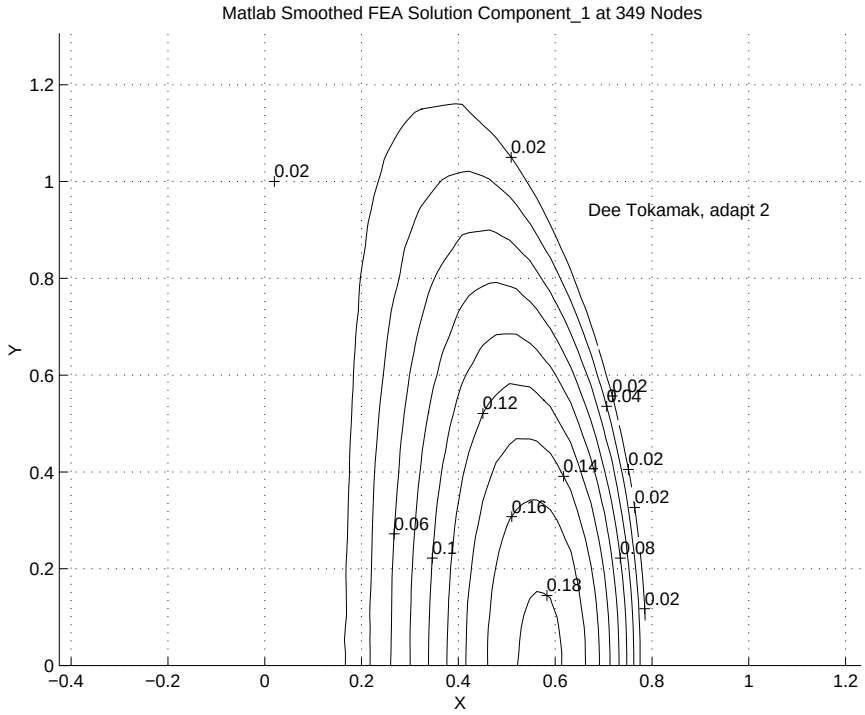

Figure 11.15.7 Stream function, ψ, values in second adaption

illustrate typical results for a dee-shape torus cross-section where the linear triangle element was employed. Biquadratic or bicubic elements would be better for some formulations which require post-solution calculations using the first and second derivatives of ψ. Figures 11.15.3, and 4 show the initial uniform mesh error estimates, and plasma **B** vectors, respectively. Note that the initial maximum error in the energy norm is about 2.4 % so adaptive refinements were taken to reduce the maximum error level to less than 1 %. The two adaptive meshes and error estimates are shown in Fig.s 11.15.5 and 6, while the final contours of ψ are in Fig 11.15.7. The initial and final surface plots of ψ are in Figs 11.15.8 and 9. The above results have assumed no external vertical **B** field so ψ was assigned values of zero on Γ_1.

11.16 Slider Bearing Lubrication

Several references are available on the application of the method to lubrication problems. These include the early work of Reddi [20], a detailed analysis and computer program for the three node triangle by Allan [4], and a presentation of higher order elements by Wada and Hayashi [23]. The most extensive discussion is probably found in the text by Huebner [11]. These formulations are based on the Reynolds equation of lubrication. For simplicity a one-dimensional formulation will be presented here. Consider the slider bearing shown in Fig. 11.16.1 which is assumed to extend to infinity out of the plane of the figure. It consists of a rigid bearing and a slider moving relative to the bearing with a velocity of U. The extremely thin gap between the bearing and the

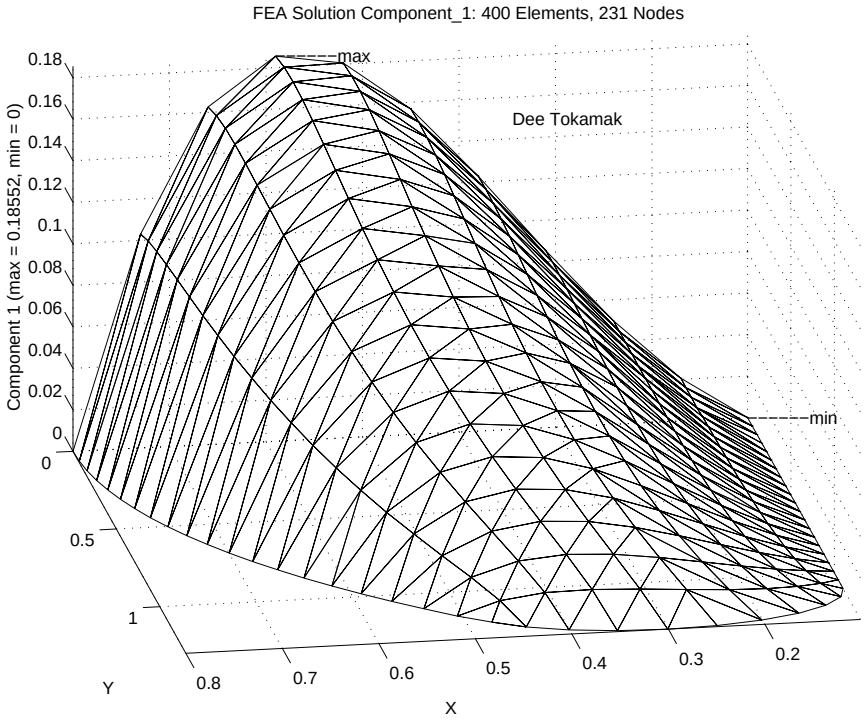

Figure 11.15.8 Initial ψ surface for half symmetry model

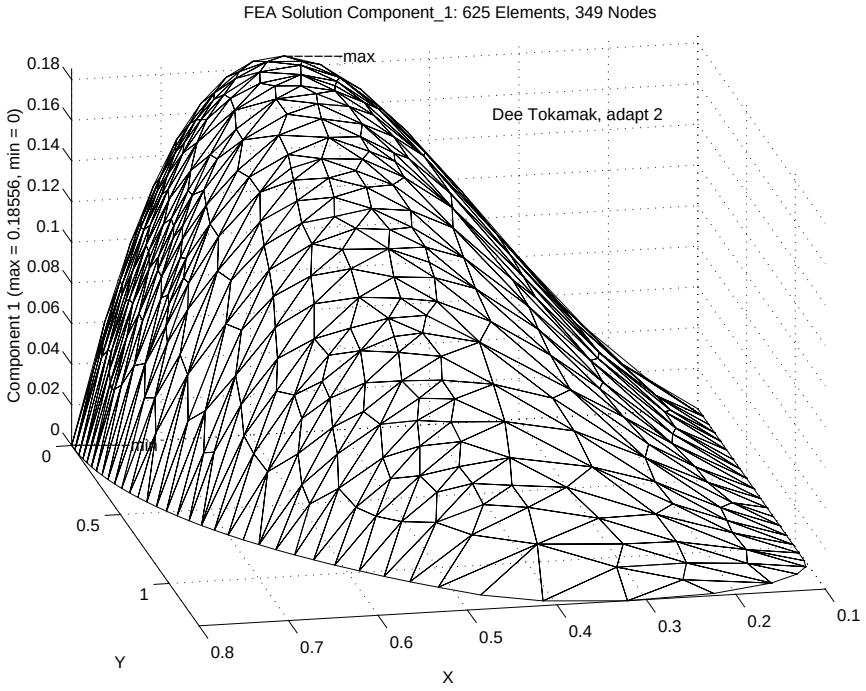

Figure 11.15.9 Final ψ surface for half symmetry model

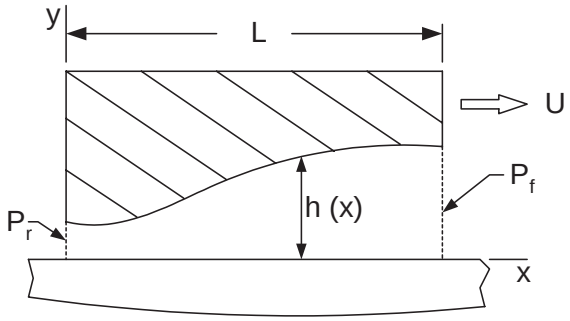

Figure 11.16.1 Thin film slider bearing notation

```
! ..................................................................... ! 1
!    ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW *** ! 2
! ..................................................................... ! 3
!     APPLICATION: LINEAR SLIDER BEARING,   Example 102          ! 4
! N_SPACE = 1,  NOD_PER_EL = 2, N_G_DOF = 1                      ! 5
! N_EL_FRE = 2, MISC_FL = 2                                      ! 6
! FLT_MISC (1) = VISCOSITY, FLT_MISC (2) = VELOCITY              ! 7
! EL_PROP (1) OR PRT_L_PT (K,1) = FILM THICKNESS                 ! 8
                                                                 ! 9
 INTEGER,   SAVE :: KALL = 1                                     !10
 REAL (DP), SAVE :: VIS, VEL, DL                                 !11
 REAL (DP)       :: THICK, CONST                                 !12
                                                                 !13
   IF ( KALL == 1 ) THEN ! GET GLOBAL REAL CONSTANTS             !14
     KALL = 0                                                    !15
     VIS = GET_REAL_MISC (1) ; VEL = GET_REAL_MISC (2)           !16
   END IF ! FIRST CALL                                           !17
                                                                 !18
!-->   DEFINE ELEMENT LENGTH AND ELEMENT THICKNESS               !19
   DL    = COORD (2, 1) - COORD (1, 1)                           !20
   THICK = 0.d0                                                  !21
   IF ( EL_REAL  > 0 ) THICK = GET_REAL_LP (1)                   !22
                                                                 !23
!      CHECK FOR ALTERNATE AVERAGE NODE THICKNESS                !24
   IF ( THICK == 0.d0 ) THEN  ! USE NODAL PROPERTY               !25
     IF ( N_NP_FLO > 0 ) THEN ! DATA EXISTS                      !26
       THICK = 0.5d0 * (PRT_L_PT (1, 1) + PRT_L_PT (2, 1) )      !27
     ELSE                                                        !28
       STOP 'NO SLIDER BEARING THICKNESS DATA'                   !29
     END IF                                                      !30
   END IF ! NODAL THICKNESS DATA                                 !31
                                                                 !32
!-->  GENERATE ELEMENT SQUARE MATRIX & COLUMN MATRIX             !33
   CONST = THICK**3 / (6.0_DP * VIS * DL)                        !34
   S (1, 1) =  CONST    ; S (2, 2) =  CONST                      !35
   S (1, 2) = -CONST    ; S (2, 1) = -CONST                      !36
   C (1) =  VEL * THICK ; C (2) = -VEL * THICK                   !37
                                                                 !38
!-->   GENERATE DATA FOR LOAD CALCULATIONS AND STORE            !39
   H_INTG (1) = 0.5_DP * DL ; H_INTG (2) = 0.5_DP * DL           !40
   IF ( N_FILE1 > 0 ) WRITE (N_FILE1) H_INTG                     !41
                                                                 !42
!    ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***    !43
```

Figure 11.16.2 Slider bearing square and load matrices

slider is filled with an incompressible lubricant having a viscosity of v. For the one-dimensional case the governing Reynolds equation reduces to

$$\frac{d}{dx}\left(\frac{h^3\,dP}{6v\,dx}\right) = \frac{d}{dx}\,(Uh)\,, \tag{11.41}$$

where $P(x)$ denotes the pressure and $h(x)$ denotes the thickness of the gap. The boundary conditions are that P must equal the known external pressures (usually zero) at the two ends of the bearing. It can be shown that the variational equivalent of the one-dimensional Reynolds equation requires the minimization of the functional

$$I = \int_0^L \left[\frac{h^3}{12v}\left(\frac{dP}{dx}\right)^2 + hU\left(\frac{dP}{dx}\right)\right] dx\,. \tag{11.42}$$

As a word of warning, it should be noted that, while the pressure P is continuous, the film thickness h is often discontinuous at one or more points on the bearing. Another related quantity of interest is the load capacity of the bearing. From statics one finds the resultant normal force per unit length in the z-direction, F_y, is

$$F_y = \int_0^L P\,dx\,. \tag{11.43}$$

This is a quantity which would be included in a typical set of post-solution calculations. Minimizing the above functional defines the element square and column matrices as

$$\mathbf{S}^e = \frac{1}{6v}\int_{x_i}^{x_j} h^3\mathbf{H}_{,x}^{eT}\,\mathbf{H}_{,x}^{e}\,dx\,,\quad \mathbf{C}^e = -U\int_{x_i}^{x_j} h\,\mathbf{H}_{,x}^{eT}\,dx\,. \tag{11.44}$$

As a specific example of a finite element formulation, consider a linear element with two nodes ($n_n = 2$) and one pressure per node ($n_g = 1$). Thus, $P(x) = \mathbf{H}^e(x)\mathbf{P}^e$ where as before $\mathbf{P}^{eT} = [\,P_i \quad P_j\,]$ and the interplation functions are $\mathbf{H}^e = [\,(x_j - x)\;(x - x_i)\,]/L$, where $L = (\,x_j - x_i\,)$ is the length of the element. For the element under consideration $\mathbf{H}^e$ is linear in x so that its first derivative will be constant. That is, $\mathbf{H}_{,x}^e = [-1 \quad 1\,]/L$ so that the element matrices simplify to

$$\mathbf{S}^e = \frac{1}{6vl^2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\int_{x_i}^{x_j} h^3\,dx\,,\quad \mathbf{C}^e = \frac{U}{l}\begin{Bmatrix} 1 \\ -1 \end{Bmatrix}\int_{x_i}^{x_j} h\,dx\,. \tag{11.45}$$

Thus, it is clear that the assumed thickness variation within the element has an important effect on the complexity of the element matrices. It should also be clear that the nodal points of the mesh must be located such that any discontinuity in h occurs at a node. The simplest assumption is that h is constant over the length of the element. In this case the latter two integrals reduce to $h^3 l$ and $h\,l$, respectively. One may wish to utilize this element to approximate a varying distribution of h by a series of constant steps. In this case, one could use an average thickness of $h = (\,h_i + h_j\,)/2$, where h_i and h_j denote the thickness at the nodal points of the element. Subroutine *ELEM_SQ_MATRIX* for this element is shown in Figs. 11.16.2.


```
! ..................................................................... ! 1
!  *** POST_PROCESS_ELEM PROBLEM DEPENDENT STATEMENTS FOLLOW *** ! 2
! ..................................................................... ! 3
!      APPLICATION: LINEAR SLIDER BEARING, Example 102                ! 4
! N_SPACE = 1,  NOD_PER_EL = 2, N_G_DOF = 1                           ! 5
! N_EL_FRE = 2, MISC_FL = 2                                           ! 6
                                                                      ! 7
 INTEGER,  SAVE :: KALL = 1                                           ! 8
 REAL(DP), SAVE :: FORCE, TOTAL = 0.0_DP                              ! 9
                                                                      !10
   IF ( KALL == 1 ) THEN   !    PRINT TITLES ON THE FIRST CALL        !11
     KALL = 0 ; WRITE (6, 5)                                          !12
     5 FORMAT (/, '***   E L E M E N T   L O A D S  ***',/, &         !13
                  'ELEMENT        LOAD            TOTAL')             !14
   END IF ! FIRST CALL                                                !15
                                                                      !16
!-->   CALCULATE LOADS ON THE ELEMENTS, F = H_INTG*D                  !17
   READ (N_FILE1) H_INTG            ! RECOVER INTEGRAL OF H           !18
   FORCE = DOT_PRODUCT (H_INTG, D)  ! INTEGRAL OF PRESSURE            !19
   TOTAL = TOTAL + FORCE            ! SYSTEM UPDATE                   !20
   WRITE (6, 10) IE, FORCE, TOTAL   ! LIST RESULTS                    !21
   10 FORMAT (I5, 1PE16.5, 3X, 1PE16.5)                               !22
                                                                      !23
!  *** END POST_PROCESS_ELEM PROBLEM DEPENDENT STATEMENTS ***         !24
```

Figure 11.16.3 Element bearing load calculations

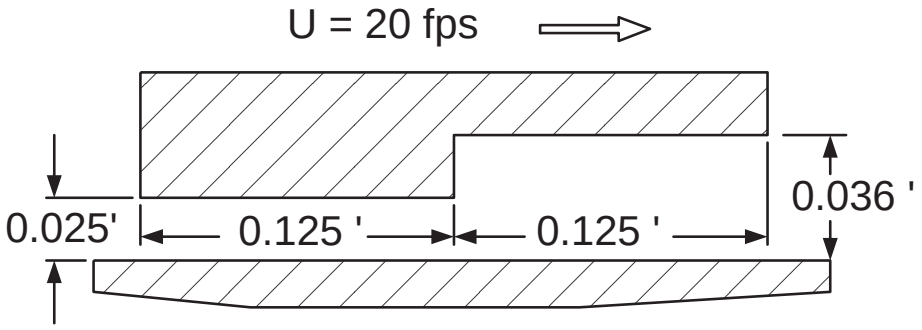

Figure 11.16.4 Step bearing example geometry


```
LINEAR SLIDER BEARING                                    ! 1
                                                         ! 2
NUMBER OF NODAL POINTS IN SYSTEM =.....    3 ! 3
NUMBER OF ELEMENTS IN SYSTEM =.........    2 ! 4
NUMBER OF NODES PER ELEMENT =..........    1 ! 5
NUMBER OF PARAMETERS PER NODE =........    2 ! 6
DIMENSION OF SPACE =...................    1 ! 7
NUMBER OF ITERATIONS TO BE RUN =.......    1 ! 8
NUMBER OF ROWS IN B MATRIX =...........    1 ! 9
ELEMENT SHAPE: LINE, TRI, QUAD, HEX, TET =...    1 !10
NUMBER OF DIFFERENT ELEMENT TYPES =.....    1 !11
STIFFNESS STORAGE MODE: SKY, BAND =.....    1 !12
NUMBER OF REAL PROPERTIES PER ELEMENT =.....    1 !13
NUMBER OF REAL MISCELLANEOUS  PROPERTIES =...    2 !14
OPTIONAL UNIT NUMBERS: N_FILE1 =  8                      !15
*** NODAL POINT DATA ***                                 !16
NODE, CONSTRAINT FLAG, 1 COORDINATES                     !17
         1         1    0.0000                          !18
         2         0    0.1250                          !19
         3         1    0.2500                          !20
*** ELEMENT CONNECTIVITY DATA ***                        !21
ELEMENT NO., 2 NODAL INCIDENCES.                         !22
    1    1    2                                         !23
    2    2    3                                         !24
***  NODAL PARAMETER CONSTRAINT LIST  ***                !25
TYPE      EQUATIONS                                      !26
   1         2                                          !27
*** CONSTRAINT EQUATION DATA ***                         !28
CONSTRAINT TYPE ONE                                      !29
EQ. NO.   NODE1   PAR1          A1                       !30
      1      1      1   0.00000E+00                      !31
      2      3      1   0.00000E+00                      !32
***  ELEMENT  PROPERTIES   ***                           !33
ELEMENT  PROPERTY      VALUE                             !34
    1         1    2.50000E-02                          !35
    2         1    3.60000E-02                          !36
***  MISCELLANEOUS SYSTEM PROPERTIES  ***                !37
PROPERTY     VALUE                                       !38
       1     2.00000E-03                                 !39
       2     2.00000E+01                                 !40
                                                         !41
***  OUTPUT OF RESULTS  ***                              !42
NODE, 1 COORDINATES, 1 PARAMETERS.                       !43
    1  0.00000E+00  0.00000E+00                          !44
    2  1.25000E-01  5.29857E+00                          !45
    3  2.50000E-01  0.00000E+00                          !46
                                                         !47
***   E L E M E N T   L O A D S  ***                     !48
ELEMENT        LOAD             TOTAL                    !49
    1     3.31160E-01        3.31160E-01                !50
    2     3.31160E-01        6.62321E-01                !51
                                                         !52
*** EXTREME VALUES OF THE NODAL PARAMETERS ***           !53
PARAMETER    MAXIMUM, NODE      MINIMUM, NODE            !54
      1   5.2986E+00,   2    0.0000E+00,   3            !55
```

Figure 11.16.5 Slider bearing results

Note that it allows for two methods of defining the film thickness, h, in each element. In the default option (keywords el_real 1 , pt_real 0) the value of h is input as a floating point element property, i.e., $H = GET\_REAL\_LP(1)$. In the second option (keywords el_real 0 , pt_real 1) the thickness is specific at each node as a floating point property. Note that for the latter option, el_real 0, causes $h = 0$. The program checks for this occurrence and then skips to the second definition of h.

The post-solution calculation function, *POST_PROCESS_ELEM*, is shown in Figs. 11.16.3. It evaluates the force, F_y^e, carried by each element. The load on a typical element is $F_y^e = \mathbf{Q}^e\mathbf{P}^e$ where $\mathbf{Q}^e = [l/2 \quad l/2]$. Subroutine *ELEM_SQ_MATRIX* also generates and stores $\mathbf{Q}^e$ for each element. Subroutine *POST_PROCESS_ELEM* carries out the multiplication once the nodal pressures, $\mathbf{P}^e$, are known. In addition, it sums the force on each element to obtain the total load capacity of the bearing. It prints the element number and its load and the total load on the bearing. With the addition of a few extra post-solution calculations, one could also output the location of the resultant bearing force. Both U and v are constant along the entire length of the bearing. They are simply defined as floating point miscellaneous system properties (keyword reals 2).

As a numerical example consider the step bearing shown in Fig. 11.16.6, which has two constants gaps of different thicknesses but equal lengths. Select a mesh with three nodes (nodes 3) and two elements (elems 2). Let $L_1 = L_2 = 0.125$ ft, $U = 20$ ft/s, $v = 0.002$ lb s/ft^2, $h_1 = 0.025$ ft, and $h_2 = 0.036$ ft. The two boundary conditions are $P_1 = P_3 = 0$ and we desire to calculate the pressure, P_2, at the step. The calculated pressure is $P_2 = 5.299$ psf, which is the exact value, and the total force on the bearing is $F_y = 0.66$ ppf. The accuracy is not surprising since the exact solution for this problem gives a linear pressure variation over each of the two segments of the bearing. The typical output data are shown in Fig. 11.16.5.

11.17 Transient Scalar Fields

All of the scalar field problems considered in this chapter can be easily extended to transient behaviour. As seen in Eq. 2.53 we just have to allow for a term involving the first partial derivative with respect to time. That also usually involves the input of one more material property and the definition of the symmetric consistent element mass matrix. Of course, we also need to describe the initial condition of the primary unknown, and possibly allow for the Dirichlet and/or Neumann and/or Robin conditions to become functions of time. Often the initial conditions are a global constant, which can be input via a control keyword. Otherwise one has to provide source code (in include file *my_iter_start_inc*) to define its spatial distribution. Below is a partial list of the more common keyword controls for semi-discrete transient integration methods:


```
TIME_CONTROL, VALUE ! REMARKS                                  [DEFAULT]
average_mass      ! Average consistent & diagonal mass matrices  [F]
crank_nicolson    ! Select transient one-step method 2           [F]
diagonal_mass     ! Use diagonalized mass matrix                 [F]
euler_forward     ! Select transient one-step method 1           [T]
galerkin_in_time  ! Select transient one-step method 3           [F]
gen_trapezoidal   ! Select transient one-step method 5           [F]
gen_trap_next .5  ! Weight of next time step, 0 to 1          [0.5]
history_dof  12   ! DOF number for time-history graph output    [0]
history_node  3   ! Node number for time-history graph outputs  [0]
inc_save      1   ! Save after steps inc, 2*inc, 3*inc, ...     [0]
initial_value 0.  ! Initial value of transient scalar everywhere [0]
least_sq_in_time  ! Select transient one-step method 4           [F]
save_1248         ! Save after steps 1, 2, 4, 8, 16 ...          [F]
start_time    0.  ! A time history starting time                 [0]
start_value   1.  ! Initial value of transient scalar everywhere [0]
time_groups   2   ! Number of groups of constant time_steps      [1]
time_method   1   ! 1-Euler, 2-Crank-Nicolson, 3-Galerkin, etc   [1]
time_step  1.d-3  ! Time step size for time dependent solution   [1]
time_steps   128  ! Number of time steps to employ               [5]
transient         ! Problem is first order in time               [F]
```

The classic finite element weighted residual formulation yields a full, symmetric consistent element mass matrix. This results in the assembled system mass matrix having the same sparsity as the system conduction square matrix. There are computational advantages in having a diagonal element, and system, mass matrix. They are often used to avoid physically impossible answers in the early stage of thermal shock applications. Good modelling practices, like having small element lengths in the direction normal to a thermal shock boundary, also avoid impossible answers but simply using diagonal mass matrices may be quicker. Hughes [12] compares several problems solved with both consistent and diagonal mass matrices. He also illustrates that the average of the consistent and diagonal mass matrices often gives a higher order of accuracy, especially for linear elements.

The storage for the element mass matrix (*EL_M*) is automatically dynamically allocated and its actual definition (in *ELEM_SQ_MATRIX*) is usually less than ten new line of code. Mainly one needs to allow for a small group of new control keywords in the data stream to activate and control the additional matrix manipulations, and additional output for time history related plots. The additional minor programming steps, at the element level, are shown in Fig. 11.17.1 for the general anisotropic Poisson equation. Basically one only need recover the extra material time coefficient, simply called ρ in that listing, (as done in line 23 or 26 or 28 or defaulted in line 9) and compute the integral of the element mass matrix by numerical integration (in line 52).

As a typical example let us returne to the mesh shown in Fig. 11.5.4 where we considered the steady state temperature of a square having a constant internal heat source. The only input changes (if the density and thickness are defaulted) are shown in Fig. 11.17.2. If we assume that the body was initially at a constant temperature of 5 (that matches the previous essential boundary condition) and pick $rhp\, c_p$ and a step size Δt so it reaches steady state in about 70 steps we compute the time history of the center point (node 1) as shown in Fig. 11.17.3. Likewise if we display the temperature surface, over the one-eighth symmetry model, at time steps 1, 4, 16, and 64 we get a good feel for how


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

Figure 11.17.1 Including the mass matric for transient applications

the temperatures change with time. These are shown in Fig. 11.17.4.

A similar problem is the study of the cooling of a cylinder formed by revolving a 1:2 rectangle about the z-axis, Assume that it is everywhere at a constant uniform temperature of 100 when its outer cylindrical surface and two end are suddenly changed to zero temperature. As you would expect, the center point will remain at its initial temperature for a while, but will eventually cool to zero. Such a time history for the center node is shown in Fig. 11.17.5 as obtained from a uniform mesh of 8 by 8 axisymmetric T3 elements.

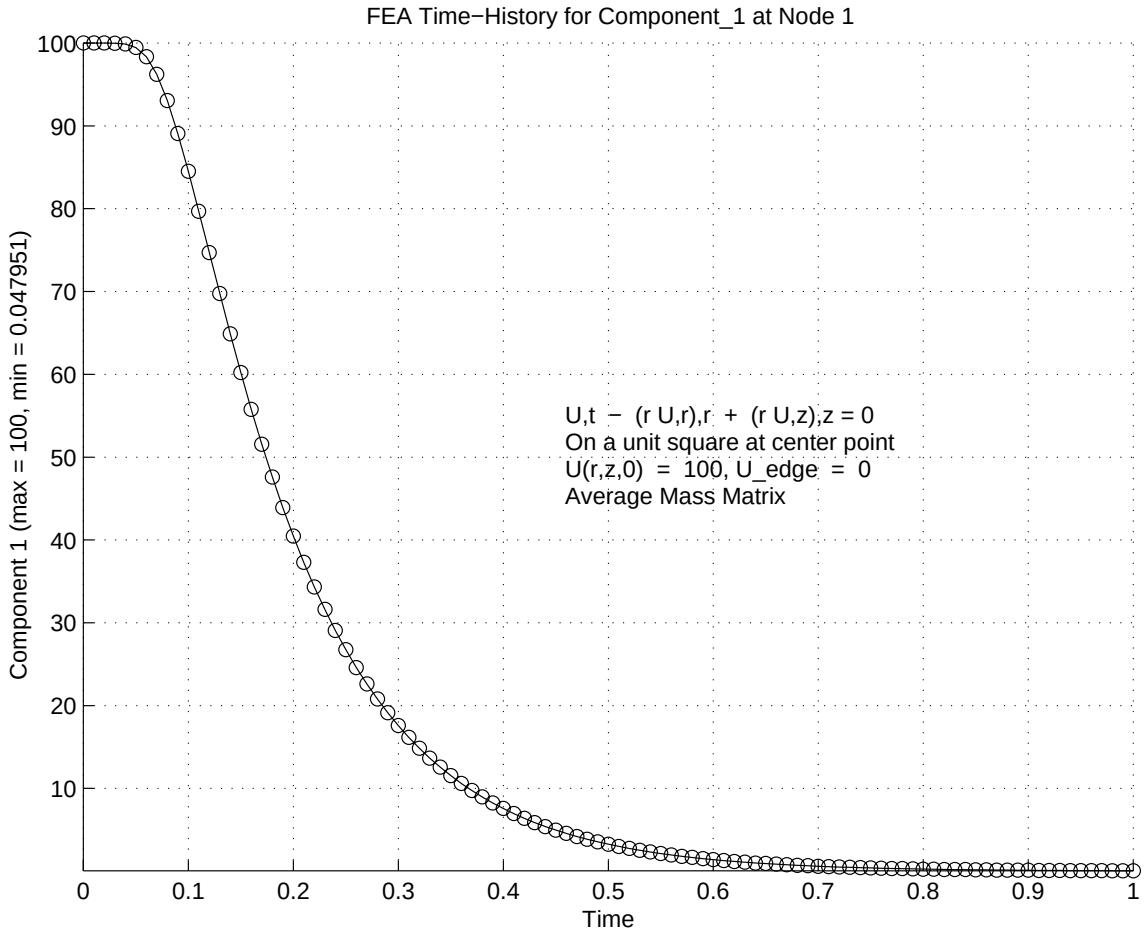

Figure 11.17.5 Cooling of the center point inside a cylinder

11.18 Exercises

1. The two-dimensional Laplace equation $\partial^2 u/\partial x^2 + \partial^2 u/\partial y^2 = 0$ is satisfied by the cubic $u(x, y) = -x^3 - y^3 + 3xy^2 + 3x^2 y$. It can be used to define the exact essential (Dirichlet) boundary conditions on the edges of ant two-dimensional shape. For example, for a unit square domain with a corner at the origin the boundary conditions are


```
title "Transient Square Conduction 1/8 Symmetry T3"              ! 1
transient         ! Problem is first order in time               ! 2
save_1248         ! Save after steps 1, 2, 4, 8, 16 ...          ! 3
average_mass      ! Average consistent & diagonal mass matrices  ! 4
history_node  1   ! Node number for time-history graph output    ! 5
start_time    0.  ! A time history starting time                 ! 6
time_method   2   ! 1-Euler, 2-Crank-Nicolson, 3-Galerkin, etc   ! 7
time_step 4.0d-2  ! Time step size for time dependent solution   ! 8
time_steps    64  ! Number of time steps in each group           ! 9
start_value   5.  ! Initial value of transient scalar everywhere !10
```

Figure 11.17.2 Modifying Fig. 11.5.4 for a transient study

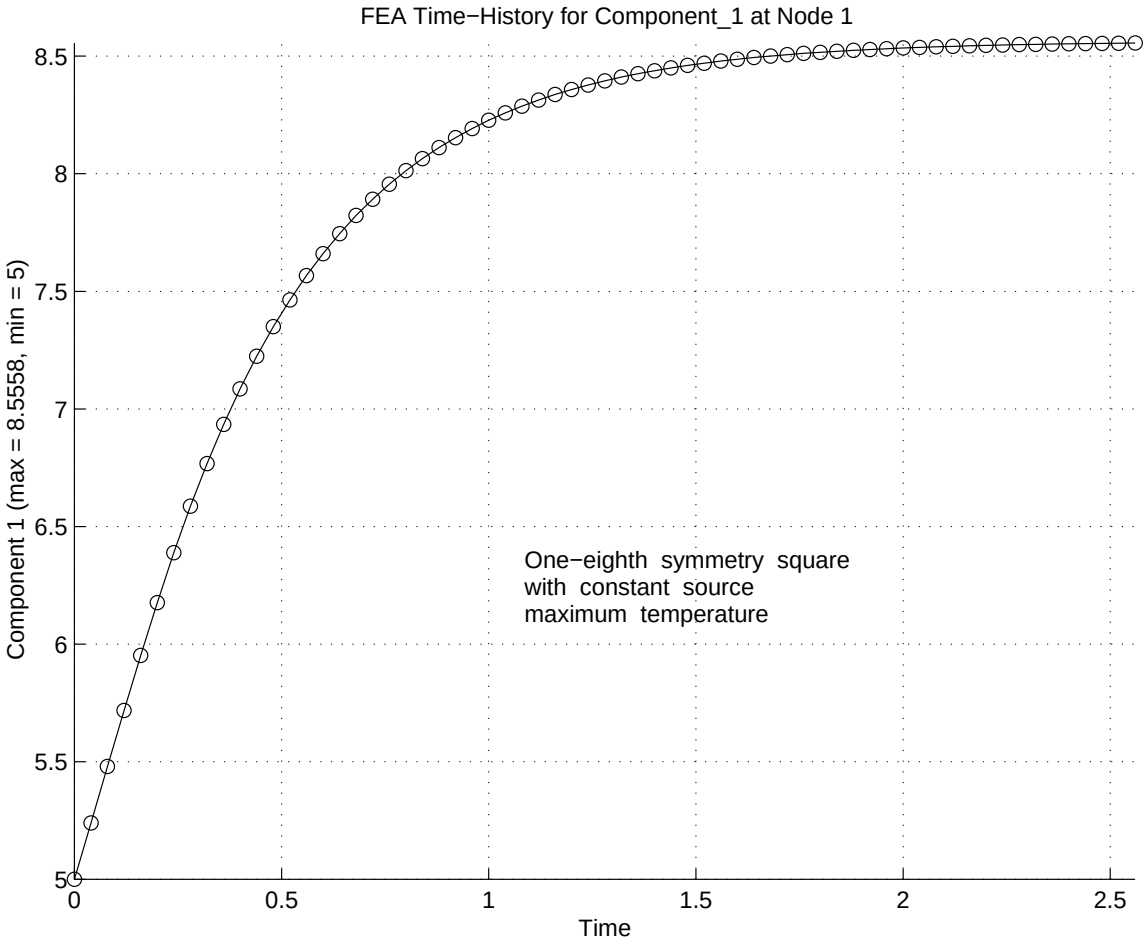

Figure 11.17.3 Center point transient, constant source square

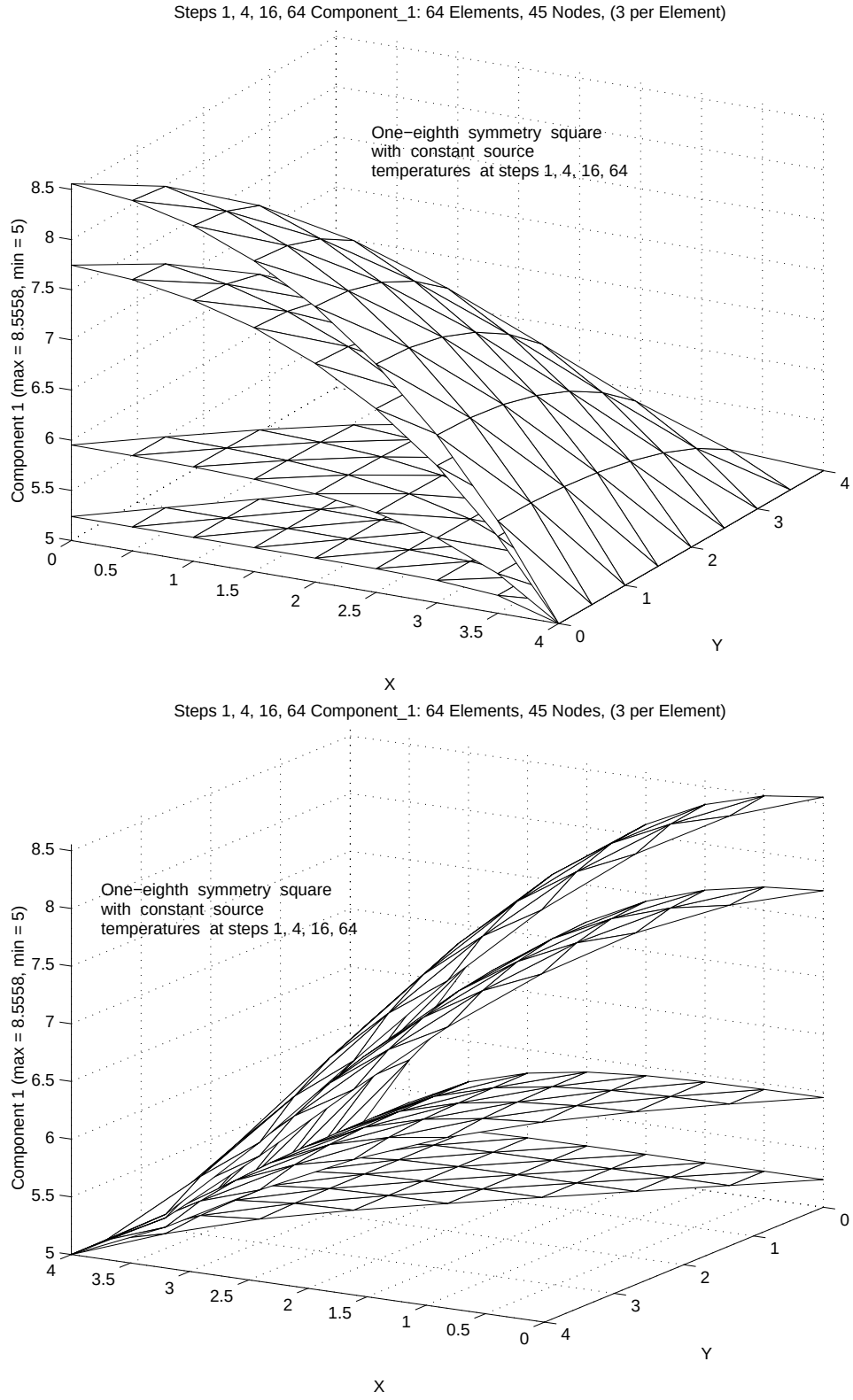

Figure 11.17.4 Constant source square, temperature surfaces over time

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

Obtain a finite element solution and sketch it along with the exact values along lines of constant x or y, or compare to *exact_case* 20 in the MODEL code. Solve using a domain of: a) the above unit square, or b) a rectangle over $0 \le x \le 3$ and $0 \le y \le 2$, or c) a quarter circle of radius 2 centered at the origin.

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

$$u(0,y) = -y^3, \quad u(1,y) = -1 - y^3 + 3y^2 + 3y\,,$$
$$\frac{\partial u}{\partial y}(x,0) = 3x^2, \quad \frac{\partial u}{\partial y}(x,1) = -3 + 6x + 3x^2\,.$$

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

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

and edge Dirichlet and Neumann boundary conditions

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

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

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 $\boldsymbol{v}\cdot\nabla\phi = \begin{Bmatrix} v_x \\ v_y \end{Bmatrix}\begin{bmatrix} \dfrac{\partial\phi}{\partial x} & \dfrac{\partial\phi}{\partial y} \end{bmatrix}$. 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:

$$\underset{1\ x\ n_s}{\mathbf{v}} = \underset{1\ x\ n_s}{[v_x \quad v_y]} = \underset{1\ x\ n_n}{\mathbf{H}^e}\ \underset{n_n\ x\ n_s}{[V_x^e \quad V_y^e]}.$$

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.

5. Sketch the boundary conditions for the ideal fluid flow around a cylinder given in Fig. 11.14.2 for an approach based on the use of a) the velocity potential, b) the stream function.

6. Solve the fin example of Fig. 11.5.14 by hand a) with insulated edges, b) with edge

convection.

7. For the solid conducting bar of Fig. 11.2.2 assume a length $L = 8$ cm, a width $2W = 4$ cm, and a thickness of $2H = 1$ cm. The material has a thermal conductivity of $k = 3\,W/cm\,C$, is surrounded by a fluid at a temperature of $\Theta_\infty = 20\,C$, and has a surface convection coefficient of $h = 0.1\,W/cm^2\,C$. Employ two equal length conduction elements and associated face, line, or point convection elements to obtain a solution for the temperature distribution and the convection heat loss. Use conduction elements that are: a) linear line elements, b) bilinear rectangular elements, c) trilinear brick elements.

8. How would you generalize the post-processing related to Eq. 11.43 to also locate the x position of the resultant force?

11.19 Bibliography

[1] Akin, J.E. and Wooten, J.W., "Tokamak Plasma Equilibria by Finite Elements," in *Finite Elements in Fluids III*, Chapter 21, ed. R.H. Gallagher, New York: John Wiley (1978).

[2] Akin, J.E., *Object-Oriented Programming Via Fortran 90/95*, Cambridge: Cambridge University Press (2003).

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

[4] Allan, T., "The Application of Finite Element Analysis to Hydrodynamic and Externally Pressurized Pocket Bearings," *Wear*, **19**, pp. 169−206 (1972).

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

[6] Chung, T.J., *Finite Element Analysis in Fluid Dynamics*, New York: McGraw-Hill (1978).

[7] 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).

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

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

[10] Huang, H.C. and Usmani, A.S., in *Finite Element Analysis for Heat Transfer*, London: Springer-Verlag (1994).

[11] Huebner, K.H., Thornton, E.A., and Byrom, T.G., *Finite Element Method for Engineers*, New York: John Wiley (1994).

[12] Hughes, T.J.R., *The Finite Element Method*, Mineola: Dover Publications (2003).

[13] Irons, B.M. and Razzaque, A., "Experience with the Patch Test for Convergence of the Finite Element Method," pp. 557−587 in *Mathematical Foundation of the Finite Element Method*, ed. A.R. Aziz, New York: Academic Press (1972).

[14] Irons, B.M. and Ahmad, S., *Techniques of Finite Elements*, New York: John Wiley (1980).
[15] Kwon, Y.W. and Bang, H., *The Finite Element Method using Matlab*, Boca Raton: CRC Press (1997).
[16] Martin, H.C. and Carey, G.F., *Introduction to Finite Element Analysis*, New York: McGraw-Hill (1974).
[17] Meek, J.L., "Field Problems Solutions by Finite Element Methods," *Civil Eng. Trans., Inst. Eng. Aust.*, , pp. 173−180 (Oct. 1968).
[18] Myers, G.E., *Analytical Methods in Conduction Heat Transfer*, New York: McGraw-Hill (1971).
[19] Oden, J.T., Demkowicz, L., Strouboulis, T., and Devloo, P., "Adaptive Methods for Problems in Solid and Fluid Mechanics," pp. 249−280 in *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, ed. I. Babuska, O.C. Zienkiewicz, J. Gago, and E.R. de A. Oliveira, Chichester: John Wiley (1986).
[20] Reddi, M.M., "Finite Element Solution of the Incompressible Lubrication Problem," *J. Lubrication Technology*, **53**(3), pp. 524−532 (July 1969).
[21] Segerlind, L.J., *Applied Finite Element Analysis*, New York: John Wiley (1984).
[22] Silvester, P.P. and Ferrari, R.L., *Finite Elements for Electrical Engineers*, Cambridge: Cambridge University Press (1983).
[23] Wada, S. and Hayashi, H., "Application of Finite Element Method to Hydrodynamic Lubrication Problems," *Bulletin of Japanese Soc. Mech. Eng.*, **14**(77), pp. 1222−1244 (1971).
[24] Zienkiewicz, O.C. and Zhu, J.Z., "Superconvergent Patch Recovery Techniques and Adaptive Finite Element Refinement," *Comp. Meth. Appl. Mech. Eng.*, **101**, pp. 207−224 (1992).
[25] Zienkiewicz, O.C. and Taylor, R.L., *The Finite Element Method*, 5th Edition, London: Butterworth-Heinemann (2000).

