Chapter 16

SELECTED SOLUTIONS.

16.1 Introduction

Here we will present selected source codes, and/or data files and corresponding outputs that are associated with the exercises at the end of the various chapters. They are listed in chapter order.

16.2 Problems from Chapter 2

Problem 2.1-A

Write a program (or spread-sheet) to plot the exact solution and the approximations on the same scale.

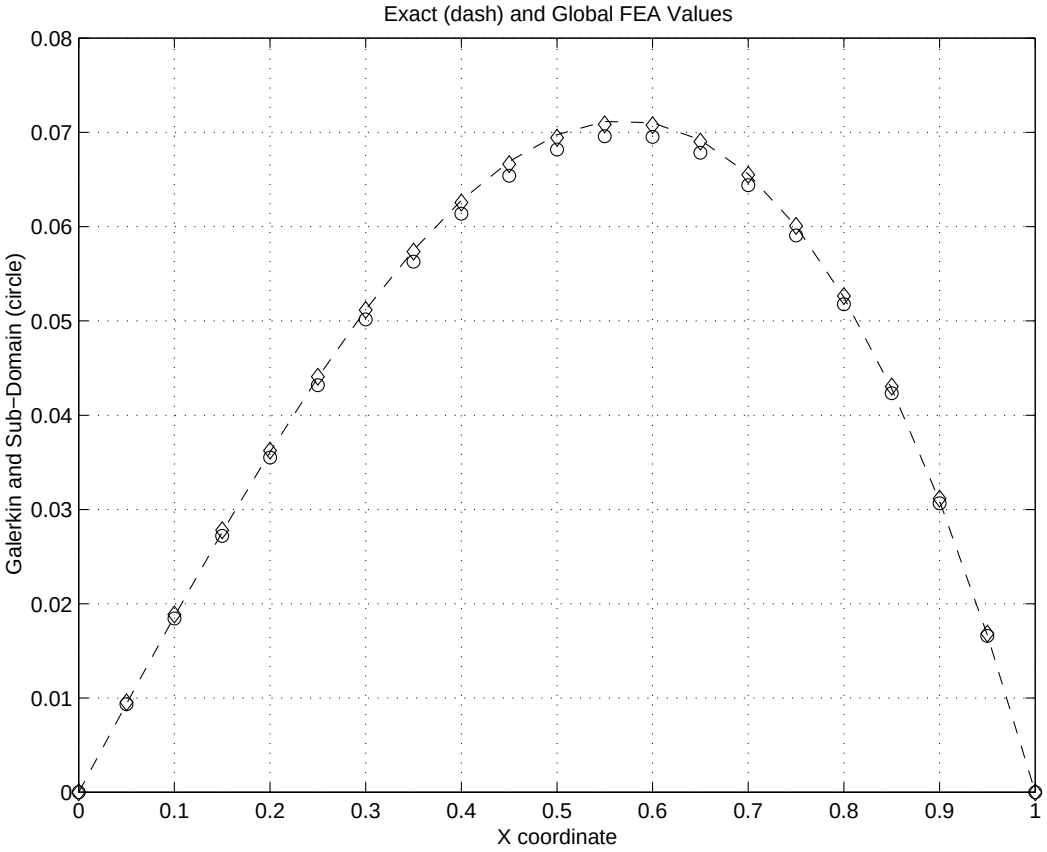
Solution: This problem is most reasonably solved by using a spread-sheet or a writing a simple code in a plotting environment like Matlab or Maple. Using Matlab a minimal script would be like that of figure P2.1Aa which produces output in P2.1Ab.

However, we will want to do this many times for various meshes and analytic results. Thus we use a much larger code that accepts data files containing the coordinates, connectivity, and result values and produces plots like Fig. 2.10.6. Those files can be created manually, or be given as output from the MODEL code. In addition, we can foresee problems with more that one unknown per node, and analytic solutions to be selected from a given library of solutions. Thus the example plot script requires the user to declare the degree of freedom to be plotted and the *exact_case* number that identifies the solution (here and in MODEL). It is presented in figure P2.1Ac and d. Note that in lines 16-18 and 26-28 it cites and loads the three ASCII character files that give the required data to plot. It would be much more efficient on large problems to use binary files but the character files are retained for educational use.

A group of more that 150 Matlab plot scripts, including this one, are provided in the public source library. Many are very similar to each other. They provide for 1-D graphs, mesh plots, boundary condition flags, code, or values, 2-D contour plots, shaded displays, 3-D wireframe, color filled, or hidden surface displays, etc. They are available for data


```
function P2_1A_plot   % for Matlab                              ! 1
% Global (single element) solution comparisons                 ! 2
  clf                                % clear frame              ! 3
  x  = [0:0.05:1.];                 % 21 points                 ! 4
  ar = sin(x)/sin(1) - x ;           % analytic result          ! 5
  plot (x, ar, 'k--')                % dash lines               ! 6
  hold ; grid                        % wait for more data       ! 7
  c1 = 0.1880 ; c2 = 0.1695 ;        % sub-domain               ! 8
  y  = x .* (1.0 - x) .* ( c1 + c2 .* x) ; % global fe          ! 9
  plot (x, y, 'ko')                  % circles                  !10
  c1 = 0.1924 ; c2 = 0.1707 ;        % Galerkin                 !11
  y  = x .* (1.0 - x) .* ( c1 + c2 .* x) ; % global fe          !12
  plot (x, y, 'kd')                  % diamonds                 !13
  title ('Exact (dash) and Global FEA Values')                  !14
  ylabel ('Galerkin and Sub-Domain (circle)')                   !15
  xlabel ('X coordinate')                                       !16
% end function P2_1A_plot                                      !17
```

Problem P2.1Aa An elementary Matlab plot script



Problem P2.1Ab Example Galerkin and sub-domain results


```
function  true_result_1d_graph (i_p, Exact)                 !  1
%  ------------------------------------------------------   !  2
%  Graph FEA & Exact_Case i_p-th component values           !  3
%   If i_p = 0, show RMS value                              !  4
%  ------------------------------------------------------   !  5
                                                            !  6
% nod_per_el = Nodes per element                            !  7
% np         = Number of Points                             !  8
% nr         = Number of results per node                   !  9
% nt         = Number of elements                           ! 10
% t_x        = x coordinates of nod_per_el corners          ! 11
% t_nodes    = nodes on an element                          ! 12
% x          = all x-coordinates     ; ax for analytic      ! 13
% y          = result component i_p  ; ar for analytic      ! 14
                                                            ! 15
% msh_typ_nodes = connectivity list, nt x nod_per_el        ! 16
% msh_bc_xyz    = nodal coordinates, np x 1                 ! 17
% node_results  = nodal result values, np x nr              ! 18
                                                            ! 19
  if ( nargin < 2 )                                         ! 20
    error ('No solution given for Exact_Case number')       ! 21
  end % if no arguments                                     ! 22
  pre_e  = 0 ;  % items before connectivity list            ! 23
  pre_p  = 1 ;  % items before coordinates (BC flag)        ! 24
                                                            ! 25
%    Read coordinate, connectivity and result files         ! 26
  load msh_bc_xyz.tmp    ;  load  msh_typ_nodes.tmp ;       ! 27
  load node_results.tmp  ;                                  ! 28
                                                            ! 29
%    Set control data:                                      ! 30
  np = size (msh_bc_xyz,1) ;       % number of nodal points ! 31
  nr = size (node_results, 1) ;    % dof per node           ! 32
  nt = size (msh_typ_nodes,1) ;    % number of elements     ! 33
  nod_per_el = size (msh_typ_nodes,2) - pre_e -1 ; % nodes  ! 34
  max_p = size (node_results, 2) ; % number of results      ! 35
                                                            ! 36
% Optional pre-allocation of arrays, get x coords           ! 37
  x (np) = 0. ; y (np) = 0. ; t_nodes (nod_per_el) = 0 ;    ! 38
  t_x (nod_per_el) = 0 ; t_y (nod_per_el) = 0 ;             ! 39
  x    = msh_bc_xyz (1:np, (pre_p+1)) ; % extract x column  ! 40
  xmax = max (x)  ; xmin = min (x) ;    % x range to plot   ! 41
                                                            ! 42
%  add analytic points (12 per element)                     ! 43
  a_inc = (xmax-xmin)/(10*nt) ; ax = [xmin:a_inc:xmax] ;    ! 44
  [ar] = analytic_1_d_result (i_p, ax, Exact) % result      ! 45
  maxa = max (ar)      ; mina = min (ar) ;    % range       ! 46
                                                            ! 47
  if ( i_p >= 1 )            %  get FEA nodal results       ! 48
    y = node_results(:, i_p) ;                              ! 49
  else % i_p = 0, get root mean square value                ! 50
    for k = 1:np                                            ! 51
      y (k) = sqrt ( sum (node_results (k, 1:max_p).^2)) ;  ! 52
    end % for k                                             ! 53
  end % if get RMS value                                    ! 54
  maxy = max (y)      ; miny = min (y) ;   % range         ! 55
```

Problem P2.1Ac Gather data for a FEA and analytic graph


```
                                                                        ! 56
%    Cite max, min FEA values and locations                             ! 57
  [V_X, L_X] = max (y) ; [V_N, L_N] = min (y) ;                         ! 58
  fprintf ('Max value is %g at node %g \n', V_X, L_X)                   ! 59
  fprintf ('Min value is %g at node %g \n', V_N, L_N)                   ! 60
  null (1:np) = V_N ; % to locate labels                                ! 61
                                                                        ! 62
% finalize axes                                                        ! 63
  ymax = max ([maxy, maxa])      ; ymin = min ([miny, mina]) ;          ! 64
  clf ; hold on ;                  % clear graphics, hold              ! 65
  axis ([xmin, xmax, ymin, ymax])   % set axes                         ! 66
  xlabel ('X, Node number at 45 deg, Element number at 90 deg')         ! 67
                                                                        ! 68
  if ( i_p >= 1 )                                                       ! 69
    title(['Exact (dash) & FEA Solution Component\_', ...               ! 70
      int2str(i_p),':  ', int2str(nt),' Elements, ', ...                ! 71
      int2str(np),' Nodes'])                                            ! 72
    ylabel (['Component ', int2str(i_p), ' (max = ', ...                ! 73
    num2str(V_X), ', min = ', num2str(V_N), ')'])                       ! 74
                                                                        ! 75
  else % i_p = 0, get root mean sq                                      ! 76
    title(['Exact (dash) & FEA RMS\_value:  ', int2str(nt), ...         ! 77
           ' Elements, ', int2str(np),' Nodes'])                        ! 78
    ylabel (['Solution RMS Value (max = ', ...                          ! 79
    num2str(V_X), ', min = ', num2str(V_N), ')'])                       ! 80
  end % if get RMS value                                                ! 81
                                                                        ! 82
  plot (ax, ar, 'r--')    %      add analytic plot first                ! 83
                                                                        ! 84
  for it = 1:nt  ;  %    Loop over all elements                         ! 85
                                                                        ! 86
%      Extract element connectivity & coordinates                       ! 87
   t_nodes = msh_typ_nodes(it,(pre_e+2):(nod_per_el+pre_e+1));          ! 88
   t_x  = x (t_nodes) ; t_y  = y (t_nodes) ; % coordinates              ! 89
                                                                        ! 90
%      Plot the element number, graph element result                    ! 91
   x_bar  = sum (t_x' )/nod_per_el ;     % centroid                    ! 92
   t_text = sprintf ('   (%g)', it);     % offset # from pt            ! 93
   text (x_bar, V_N, t_text, 'Rotation', 90) % incline                  ! 94
   plot (x_bar, V_N, 'k+')               % element number              ! 95
   plot (t_x, t_y)                       % element lines               ! 96
  end % for over all elements                                           ! 97
                                                                        ! 98
  for i = 1:np                % plot node points on axis                 ! 99
    t_text = sprintf ('   %g', i);      % offset # from pt              !100
    text (x(i), null(i), t_text, 'Rotation', 45) % incline              !101
  end % for all node numbers, Add * at nodes                            !102
  plot (x, null, 'k*') ; grid ; hold off                                !103
% end of true_result_1d_graph                                          !104
```

Problem P2.1Ad Plotting a FEA and analytic graph

checks, FEA results and fluxes, exact results and fluxes, FEA error estimates, exact errors and error norms, etc. They all access sequential character files such as the three above. The necessary files are produced (or not) by MODEL. Below is a list of the most common files and a list of associated keywords. time integration.

el_error_est	msh_bc_xyz
el_qp_xyz_fluxes	msh_bc_values
el_qp_xyz_grads	msh_new_el_scale
el_qp_xyz_sources	msh_typ_nodes
el_qp_xyz_vel	node_results
el_tau_value	pt_ave_error_est
exact_node_solution	pt_ave_ex_error
exact_node_flux	scp_node_ave_fluxes

List of data files for optional plotting

Keywords	File Created
list_exact_flux	exact_node_flux
sav_exact	exact_node_solution
sav_exact	pt_ave_ex_error
turn_on_scp	pt_ave_error_est
turn_on_scp	scp_node_ave_fluxes
turn_on_scp	pt_ave_ex_error

List of optional keyword control of data files

```
! ..............................................................  ! 1
!  ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***   ! 2
! ..............................................................  ! 3
!     APPLICATION:  LEAST SQUARES SOLUTION OF Y' + A * Y = F     ! 4
! Exact solution: y(x) = (1 - e (-Ax)) * F / A, with y(0) = 0    ! 5
!           N_SPACE = 1, NOD_PER_EL = 2, N_G_DOF = 1             ! 6
                                                                 ! 7
 REAL(DP), SAVE :: A, F, DL  ! GLOBAL DATA, ELEMENT LENGTH       ! 8
                                                                 ! 9
!  RECOVER GLOBAL PROBLEM COEFFICIENTS, A AND F (ON FIRST CALL) !10
  IF ( THIS_EL== 1 ) THEN ! Get coefficients                     !11
    A = GET_REAL_MISC (1) ; F = GET_REAL_MISC (2)                !12
  END IF ! First call                                            !13
                                                                 !14
  DL = COORD (2, 1) - COORD (1, 1)  ! ELEMENT LENGTH             !15
                                                                 !16
!    EXACT INTEGRATION SQUARE MATRIX AND SOURCE VECTOR           !17
  S (1, 1) = (3.d0 - 3.d0 * A * DL + A * A * DL * DL) / 3 / DL   !18
  S (2, 2) = (3.d0 + 3.d0 * A * DL + A * A * DL * DL) / 3 / DL   !19
  S (1, 2) = (A * A * DL * DL - 6.d0) / 6.d0 / DL                !20
  S (2, 1) = S (1, 2)                                            !21
                                                                 !22
  C (1) = 0.5d0 * F * (A * DL - 2.d0)                            !23
  C (2) = 0.5d0 * F * (A * DL + 2.d0)                            !24
! ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***       !25
```

Problem P2.6Aa Element matrices from exact integration

Problem 2.6-A

Formulate the first order equation $dy/dx + Ay = F$ by A. least squares, B. Galerkin method. Use analytic integration for the linear line element (L2) to form the two element matrices. Compute a solution for $y(0) = 0$ with $A = 2, F = 10$ for 5 uniform elements

over $x \le 0.5$.

Solution: Since this problem involves only the first derivative in the weak form both a Galerkin and a least squares form can utilize C^0 continuity elements. That is, only the solution must be continuous between elements, so we select the simple L2 line element for an analytically integrated set of matrices. For the least square choice we get the matrices shown in P2.6Aa which for the specified data gives the results in P2.6Ab.

Note that one could consider x here to represent time so this becomes an initial value problem, and the boundary condition $y(0) = 0$ is actually the initial condition. That is, the example also corresponds to a finite element least squares integration in time. Likewise, a Galerkin implementation can be thought of as a finite element Galerkin integration in time. The process is easily extended to matrix coefficients where y is a vector, A is a square matrix, and the leading coefficient is the identity matrix. For large matrices we may not have enough memory to take 5 step in time in one solution but we could solve one step in time repeatedly and use the answer for the first step (here $y(0.1)$) as the initial condition for the next step. Taking more than one step in time in a single solution, when feasible, reduces the error in the time integration.

16.3 Problems from Chapter 3

Problem 3.8

Obtain a Galerkin solution of

$$y'' - 2y'x/g + 2y/g = g,\ g = (x^2 + 1), \quad 0 \le x \le 1$$

with the boundary conditions $y(0) = 2$, $y(1) = 5/3$.

Solution: This requires only a C^0 approximation since the weak form involves only the first derivative of y. Since the variable coefficients are rational polynomials it would be complicated to try to evaluate the matrices by exact integration so we use numerical integration. For the same reason, Gaussian quadratures will not be exact so we select a moderate number of quadrature points as a balance between accuracy and computational cost. This implementation supports all one-dimension elements in the library and is displayed in P3.8A.

Note, in lines 25-26, that the variable coefficients have been hard coded rather than using an include file or user function. The variable *SOURCE* is a global item that can sometimes be assigned constant values through keyword data controls. It was not actually needed here since f_3 is defined. To test the solution we select a crude mesh of quadratic L3 line elements. The reader should try a solution with linear or cubic elements. The typical input test data are given in P3.8B along with corresponding selected outputs in P3.8C. The true solution is compared to the Galerkin FEA result in P3.8D. There straight lines are plotted between nodal values (rather than actual parabolic curves) so the two plots appear different.


```
TITLE "LEAST SQUARES SOLUTION OF Y'+2Y=10, Y(0) = 0" ! 1
                                                     ! 2
 THE NEXT  3  LINES ARE USER SUPPLIED                ! 3
 1 LEAST SQUARES SOLUTION OF Y'+2Y=10,  Y(0) = 0     ! 4
 2 Exact solution = 5(1 - e^(-2x))                   ! 5
 3 This is example 103 in the source library         ! 6
                                                     ! 7
*** NODAL POINT DATA ***                             ! 8
   NODE, BC_FLAG, X-Coord,                           ! 9
      1  1    0.0000                                 !10
      2  0    0.1000                                 !11
      3  0    0.2000                                 !12
      4  0    0.3000                                 !13
      5  0    0.4000                                 !14
      6  0    0.5000                                 !15
                                                     !16
*** ELEMENT CONNECTIVITY DATA ***                    !17
ELEMENT,  2 NODAL INCIDENCES.                        !18
      1     1     2                                  !19
      2     2     3                                  !20
      3     3     4                                  !21
      4     4     5                                  !22
      5     5     6                                  !23
                                                     !24
*** CONSTRAINT EQUATION DATA ***                     !25
CONSTRAINT TYPE_ONE: (PAR_1 @ NODE_1) =  A_1.        !26
EQ. NO.   NODE_1   PAR_1          A_1                !27
      1      1      1   0.00000E+00                  !28
                                                     !29
***  MISCELLANEOUS SYSTEM PROPERTIES  ***            !30
PROPERTY    REAL_VALUE                               !31
       1     2.00000E+00                             !32
       2     1.00000E+01                             !33
                                                     !34
***  OUTPUT OF RESULTS IN NODAL ORDER  ***           !35
NODE,  X-Coord,    DOF_1,                            !36
   1  0.0000E+00  0.0000E+00                         !37
   2  1.0000E-01  9.0749E-01                         !38
   3  2.0000E-01  1.6502E+00                         !39
   4  3.0000E-01  2.2580E+00                         !40
   5  4.0000E-01  2.7554E+00                         !41
   6  5.0000E-01  3.1624E+00    (exact 3.1606)       !42
```

Problem P2.6Ab Results for 5 L2 elements


```
! ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***    ! 1
! ..................................................................! 2
!     APPLICATION DEPENDENT Galerkin MWR FOR EXAMPLE 106          ! 3
!        Y'' - 2XY'/(X^2+1) + 2Y/(X^2+1) = (X^2+1), or            ! 4
!      E Y'' + f_1 Y'       + f_2 Y      = f_3                    ! 5
! Y(0)=2, Y(1)=5/3, Y=X^4/6 - 3X^2/2 + X + 2 (is EXACT_CASE 15)   ! 6
                                                                  ! 7
  REAL(DP)  :: DL, DX_DR, CONST           ! Length, Jacobian     ! 8
  REAL(DP)  :: f_1, f_2, f_3              ! Coefficients         ! 9
  INTEGER   :: IQ                         ! Loops                !10
                                                                 !11
  DL    = COORD (LT_N, 1) - COORD (1, 1)  ! LENGTH               !12
  DX_DR = DL / 2.                         ! CONSTANT JACOBIAN    !13
  E (1, 1) = 1                            ! identity matrix      !14
  CALL STORE_FLUX_POINT_COUNT             ! Save LT_QP           !15
                                                                 !16
  DO IQ = 1, LT_QP           ! LOOP OVER QUADRATURES             !17
    CONST = DX_DR * WT (IQ)  ! NET WEIGHT                        !18
                                                                 !19
!        GET INTERPOLATION FUNCTIONS, AND X-COORD                !20
    H    = GET_H_AT_QP (IQ)   ! INTERPOLATIONS                   !21
    XYZ  = MATMUL (H, COORD)  ! ISOPARAMETRIC POINT              !22
                                                                 !23
!        DEFINE VARIABLE COEFFICIENTS                            !24
    f_3 = 1.d0 + XYZ (1) **2  ; SOURCE = -f_3  ! global source   !25
    f_2 = 2.d0 / f_3          ; f_1 = -XYZ (1) * f_2             !26
                                                                 !27
!        LOCAL AND GLOBAL DERIVATIVES                            !28
    DLH   = GET_DLH_AT_QP (IQ) ; DGH   = DLH / DX_DR             !29
                                                                 !30
!         SQUARE MATRIX                                          !31
    S = S + MATMUL (TRANSPOSE(DGH), DGH) * CONST       &         !32
          - f_1 * OUTER_PRODUCT (H, DGH(1, :)) * CONST &         !33
          - f_2 * OUTER_PRODUCT (H, H) * CONST                   !34
                                                                 !35
    C = C + SOURCE * H * CONST ! RESULTANT SOURCE VECTOR         !36
                                                                 !37
    CALL STORE_FLUX_POINT_DATA (XYZ, E, DGH) ! for SCP           !38
  END DO  ! QUADRATURE                                           !39
!    ***  END ELEM_SQ_EX_106 PROBLEM DEPENDENT STATEMENTS ***    !40
```

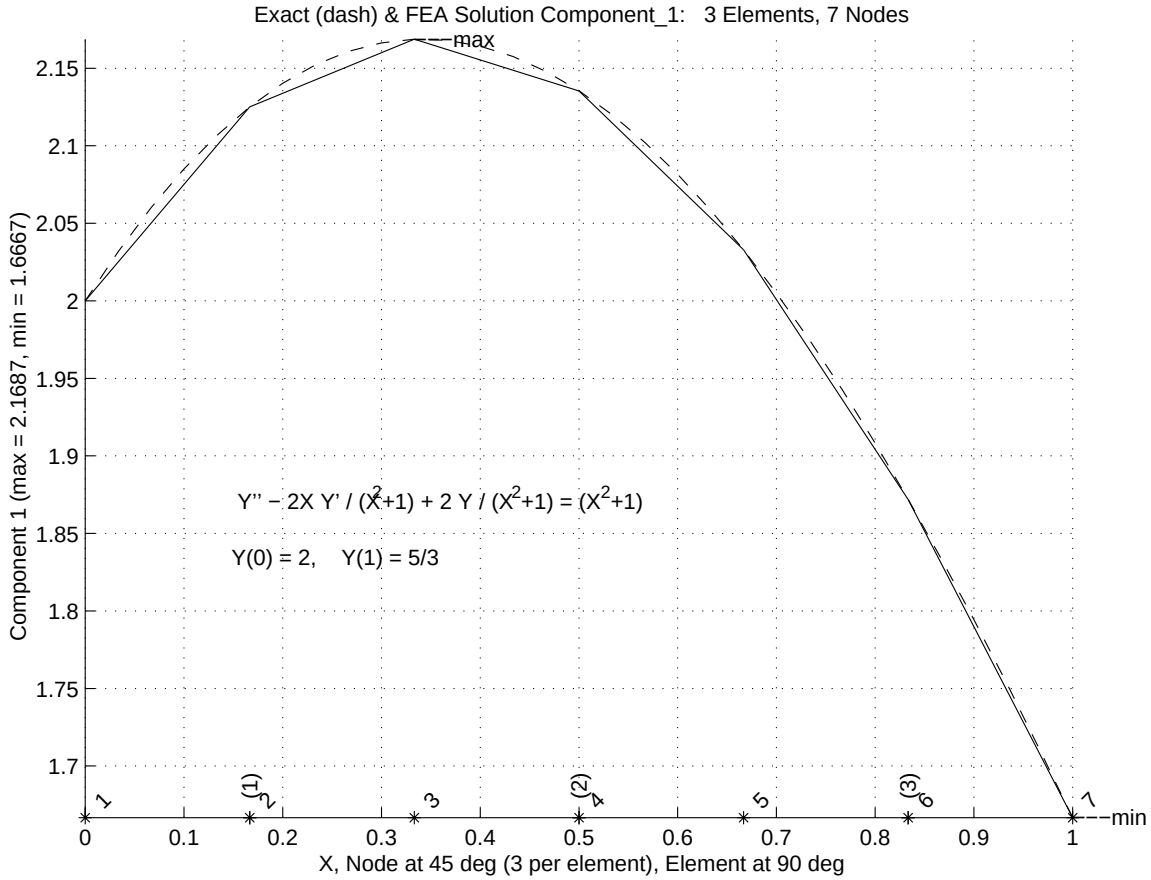
Problem P3.8A Galerkin variable coefficient test


```
title "Y'' - 2XY'/(X^2+1) + 2Y/(X^2+1) = (X^2+1)"       ! 1
nodes         7 ! Number of nodes in the mesh           ! 2
elems         3 ! Number of elements in the system      ! 3
dof           1 ! Number of unknowns per node           ! 4
el_nodes      3 ! Maximum number of nodes per element   ! 5
space         1 ! Solution space dimension              ! 6
b_rows        1 ! Number of rows in B (operator) matrix ! 7
el_react        ! Compute & list element reactions      ! 8
gauss         4 ! Maximum number of quadrature points   ! 9
exact_case   15 ! Exact analytic solution               !10
scp_neigh_el    ! Default SCP patch group type          !11
unsymmetric     ! Unsymmetric skyline storage is used   !12
list_exact      ! List exact answers at nodes           !13
list_exact_flux ! List exact fluxes at nodes            !14
bar_chart       ! print-plot result                     !15
remarks       4 ! Number of user remarks                !16
quit ! keyword input, remarks follow                    !17
Galerkin: Y'' - 2XY'/(X^2+1) + 2Y/(X^2+1) = (X^2+1)     !18
with Y(0)=2, Y(1)=5/3, Y(X) = X^4/6 - 3X^2/2 + X + 2    !19
Fausett p. 482, EXACT_CASE = 15                         !20
Here we use three quadratic (L3) line elements.         !21
 1 1 0.          ! node bc_flag x                       !22
 2 0 0.166666667                                        !23
 3 0 0.333333333                                        !24
 4 0 0.5                                                !25
 5 0 0.666666667                                        !26
 6 0 0.833333334                                        !27
 7 1 1.00                                               !28
 1    1    2    3  ! elem j, k, l                       !29
 2    3    4    5                                      !30
 3    5    6    7                                      !31
 1 1 2.           ! Essential BC: node dof value        !32
 7 1 1.66666667   ! Essential BC: node dof value        !33
```

Problem P3.8b Galerkin L2 model test data


```
Y"-2XY'/(X^2+1)+2Y/(X^2+1)=(X^2+1), Y(0)=2, Y(1)=5/3   ! 1
                                                       ! 2
** OUTPUT OF RESULTS & EXACT VALUES IN NODAL ORDER **  ! 3
NODE,  X-Coord,    DOF_1,      EXACT1,                 ! 4
   1  0.0000E+00  2.0000E+00  2.0000E+00                ! 5
   2  1.6667E-01  2.1251E+00  2.1251E+00                ! 6
   3  3.3333E-01  2.1687E+00  2.1687E+00                ! 7
   4  5.0000E-01  2.1353E+00  2.1354E+00                ! 8
   5  6.6667E-01  2.0329E+00  2.0329E+00                ! 9
   6  8.3333E-01  1.8719E+00  1.8720E+00                !10
   7  1.0000E+00  1.6667E+00  1.6667E+00                !11
                                                       !12
** FE & EXACT FLUX COMPONENTS AT INTEGRATION POINTS ** !13
ELEMENT, PT,  X-Coord,   FX_1,     EX_1,               !14
       1  1  2.314E-02  9.270E-01  9.306E-01           !15
       1  2  1.100E-01  6.723E-01  6.709E-01           !16
       1  3  2.233E-01  3.399E-01  3.374E-01           !17
       1  4  3.102E-01  8.517E-02  8.933E-02           !18
ELEMENT, PT,  X-Coord,   FX_1,     EX_1,               !19
       2  1  3.565E-01 -5.063E-02 -3.923E-02           !20
       2  2  4.433E-01 -2.665E-01 -2.719E-01           !21
       2  3  5.567E-01 -5.482E-01 -5.550E-01           !22
       2  4  6.435E-01 -7.641E-01 -7.529E-01           !23
ELEMENT, PT,  X-Coord,   FX_1,     EX_1,               !24
       3  1  6.898E-01 -8.700E-01 -8.506E-01           !25
       3  2  7.767E-01 -1.008E+00 -1.018E+00           !26
       3  3  8.900E-01 -1.189E+00 -1.200E+00           !27
       3  4  9.769E-01 -1.327E+00 -1.309E+00           !28
                                                       !29
** SUPER_CONVERGENT AVERAGED NODAL & EXACT FLUXES **   !30
NODE,   X-Coord,    FLUX_1,     EXACT1,                !31
     1  0.000E+00  1.016E+00  1.000E+00                !32
     2  1.667E-01  4.939E-01  5.031E-01                !33
     3  3.333E-01  1.817E-02  2.469E-02                !34
     4  5.000E-01 -4.187E-01 -4.167E-01                !35
     5  6.667E-01 -8.007E-01 -8.025E-01                !36
     6  8.333E-01 -1.110E+00 -1.114E+00                !37
     7  1.000E+00 -1.360E+00 -1.333E+00                !38
                                                       !39
ELEMENT ERROR ESTIMATE GRAPH                           !40
EL  VALUE +---+---+---+---+---+---+---+---+---+---+    !41
1 5.89E-3 XXXXXXXXXXXXXXXX                      +      !42
2 5.14E-3 X                                     +      !43
3 6.95E-3 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX    !44
EL  VALUE +---+---+---+---+---+---+---+---+---+---+    !45
                                                       !46
NODAL SOLUTION GRAPHS                                  !47
PT  VALUE +---+---+---+---+---+---+---+---+---+---+    !48
1 2.00E+0 XXXXXXXXXXXXXXXXXXXXXXXXXX            +      !49
2 2.13E+0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  +      !50
3 2.17E+0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX    !51
4 2.14E+0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  +    !52
5 2.03E+0 XXXXXXXXXXXXXXXXXXXXXXXXXXXX          +      !53
6 1.87E+0 XXXXXXXXXXXXXXXX                      +      !54
7 1.67E+0 X                                     +      !55
PT  VALUE +---+---+---+---+---+---+---+---+---+---+    !56
```

Problem P3.8c Galerkin variable coefficient results and exact values



Problem P3.8d True solution and Galerkin nodal values

Problem 3.9

For the differential equation in Problem 3.8 if we have one essential boundary condition of $y(0) = 1$ and one Neumann flux boundary condition of $dy/dx(1) = -4/3$ the solution is unchanged. Obtain a Galerkin finite element solution and compare it to the exact result.

Solution: This problem only requires a change in the data file that reduces the number of essential boundary conditions to one and adds a known flux term to the rhs source vector. These changes are denoted in P3.9a. External source vector components are initialized to zero. The keyword *loads* requires the non-zero entries to be input (after the EBC and MPC) by giving the node number, degree of freedom number, and corresponding known flux. The data stream terminates with the last dof flux value (which is usually zero). The output is unchanged except for echoing the above lines and listing the initial external source vector terms, in P3.9b.


```
loads           ! An initial source vector is input    !11
Galerkin: Y'' - 2XY'/(X^2+1) + 2Y/(X^2+1) = (X^2+1)    !18
with Y(0)=2, Y'(1)=-4/3, Y(X) = X^4/6 - 3X^2/2 + X + 2 !19
 7 0 1.00        ! removed EBC flag                     !28
 7 1 -1.33333 ! Enter only (last) non-zero source term !33
```

Problem P3.9a Galerkin L2 data changes for Neumann condition

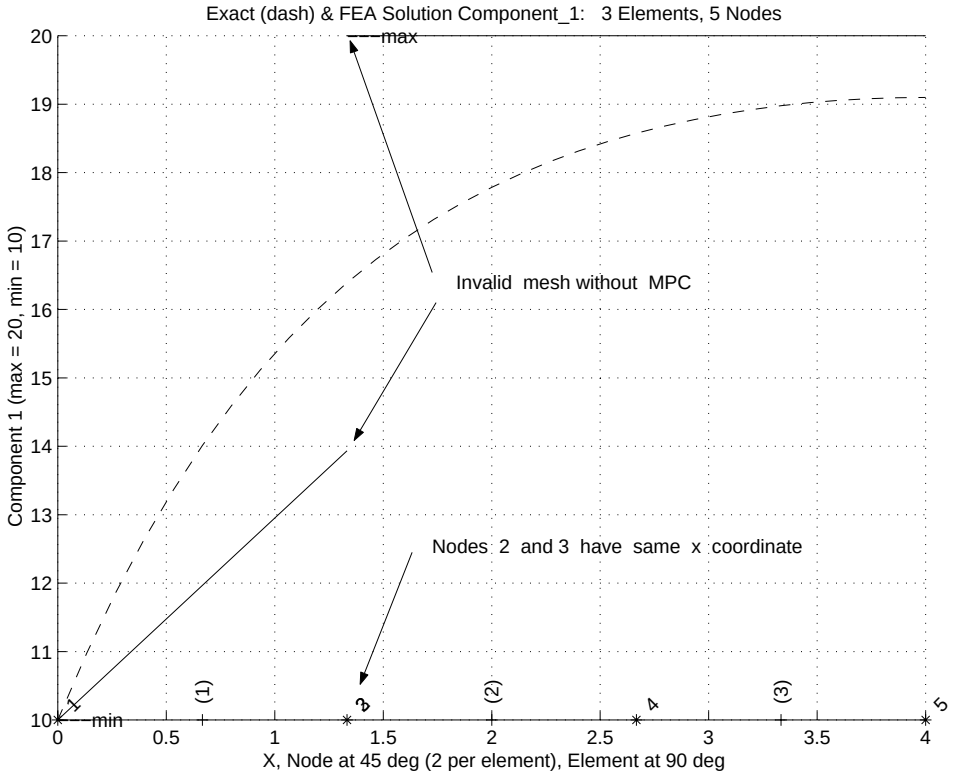
```
    . . .                                               !
*** INITIAL FORCING VECTOR DATA ***                     !
     NODE   PARAMETER     VALUE    EQUATION            !
        7           1  -1.33333E+00       7            !
*** RESULTANTS ***                                      !
COMPONENT        SUM         POSITIVE      NEGATIVE    !
 IN_1,       -1.3333E+00    0.0000E+00   -1.3333E+00  !
```

Problem P3.9b Galerkin L2 output changes for Neumann condition

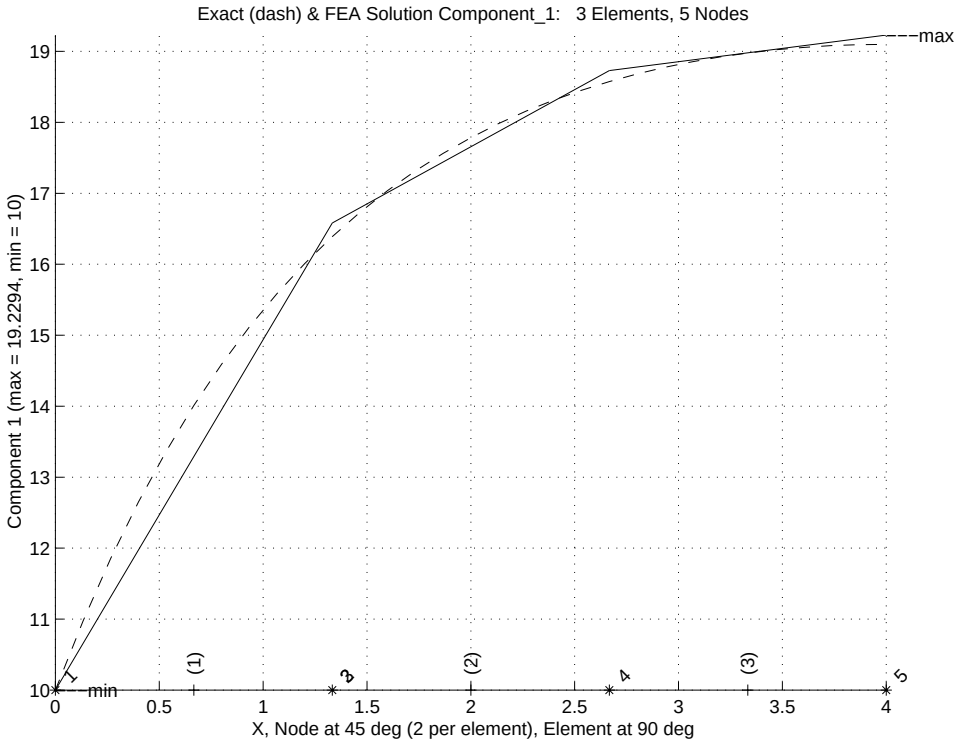
Problem 3.12

Resolve the heat transfer problem in Fig. 3.7.1 with the external reference temperature increased from 0 to 20 F ($U\_ext = 20$). Employ 3 equal length L2 linear elements, and 5 (not 4) nodes with nodes 2 and 3 both at $x = 4/3$. Connect element 1 to nodes 1 and 2 and (incorrectly) connect element 2 to nodes 3 and 4. (a) Solve the incorrect model, sketch the FEA and true solutions, and discuss why the left and right ($x \le 4/3$ and $x \ge 4/3$) domain FEA solutions behave as they do. (b) Enforce the correct solution by imposing a "multiple point constraint" (MPC) that requires $1 \times t_2 - 1 \times t_3 = 0$. Solution: (a) This only requires changing the data in Fig.3.7.3, as described above, and applying the matrix definitions in Fig. 3.7.2, or the simplified L2 version thereof. The invalid (discontinuous) FEA solution is given in P3.12a.

Solution: (b) One must either correct the mesh to have the proper connectivity, or apply a MPC to enforce continuity. Here the latter is employed to reinforce the concept of MPC's which are often needed in real applications to correct data errors (as here) or for other analysis reasons. The corrected solution is in P3.12b and is reasonable for a 3 element solution. The data for this solution are given in P3.12c. In the MODEL code a MPC between 2 degrees of freedom is referred to as a "Type 2" constraint. In lines 45 and 46 of P3.12c the normally zero boundary constraint flag has been set to 2 for each of the constrained degrees of freedom. For each such pair of flags there must be one "Type 2" constraint equation. In general its form would be $a \times t_j + b \times t_k = c$. Here we want 2 *dof* to be identical so we set $a = 1, b = -1, c = 0$. In addition we must provide the equation numbers, j and k. Here j is for node 2, parameter 1, while k is for node 3, and also parameter 1. Those data are give after the essential boundary conditions (the "Type 1" conditions) and appear at line 53 of the data set. The selected output in P3.12d echos these constraint data again and gives the FEA and exact nodal values.



Problem P3.12a Invalid (discontinuous) L2 convection solution results



Problem P3.12b Corrected (via MPC) L2 convection solution results


```
title "1-D heat transfer example, MPC mesh fix, 3 L2"  ! 1
nodes    5 ! Number of nodes in the mesh               ! 2
elems    3 ! Number of elements in the system          ! 3
dof      1 ! Number of unknowns per node               ! 4
el_nodes 2 ! Maximum number of nodes per element       ! 5
space    1 ! Solution space dimension                  ! 6
b_rows   1 ! Number of rows in B (operator) matrix     ! 7
shape    1 ! Shape, 1=line, 2=tri, 3=quad, 4=hex       ! 8
gauss    2 ! Maximum number of quadrature point        ! 9
el_real  4 ! Number of real properties per element     !10
reals    3 ! Number of miscellaneous real properties   !11
el_homo    ! Element properties are homogeneous        !12
el_list    ! List results at each node of each element !13
post_1     ! Require post-processing, create n_tape1   !14
post_2     ! Require post-processing, create n_tape2   !15
example  101 ! Source library example number           !16
data_set   3 ! Data set for example (this file)        !17
exact_case 1 ! Analytic solution for list_exact, etc   !18
scp_neigh_el    ! Default SCP patch group type         !19
list_exact      ! List given exact answers at nodes    !20
list_exact_flux ! List given exact fluxes at nodes     !21
remarks     20  ! Number of user remarks               !22
quit ! keyword input, remarks follow                   !23
Combined heat conduction and convection from a bar:    !24
K*A*U,XX - h*P*(U-U_ext) = 0,  U(0)=U_0, dU/dx(L)=0     !25
For globally constant data the analytic solution is:   !26
U(x) = U_ext + (U_0-U_ext)*Cosh [m*(L-x)]/Cosh [mL]     !27
m^2 = h_e * P_e/(K_e * A_e), dimensionless             !28
Miscellaneous FE data:                                 !29
U_ext = GET_REAL_MISC (1)  external reference temp, F  !30
Miscellaneous data used ONLY for analytic solution:    !31
L     = GET_REAL_MISC (2)  exact length, ft            !32
U_0   = GET_REAL_MISC (3)  essential bc at x = 0, F    !33
Real FE problem properties are:                        !34
K_e = GET_REAL_LP (1) conductivity, BTU/ hr ft F       !35
A_e = GET_REAL_LP (2) area of bar, ft^2                !36
h_e = GET_REAL_LP (3) convection, BTU/ hr ft^2 F       !37
P_e = GET_REAL_LP (4) perimeter of area A_e, ft        !38
Elements 1 & 2 are NOT connected.  A correct solution  !39
can not be obtained without a mesh correction, or      !40
a "Multi-Point Constraint" that enforces the same      !41
answer at nodes 2 & 3. It is: 1*U_2 - 1*U_3 = 0.       !42
Flagged by setting the bc code to 2 at those nodes.    !43
 1   1 0.00        ! begin nodes                       !44
 2   2 1.33333333  ! correcting duplicate node         !45
 3   2 1.33333333  ! correcting duplicate node         !46
 4   0 2.66666667                                     !47
 5   0 4.00                                           !48
  1    1    2   ! begin elements                      !49
  2    3    4   ! not connected to element 1         !50
  3    4    5                                        !51
1 1 10.           ! essential BC                       !52
2 1 3 1 1. -1. 0. ! MPC data: 1*U_2 - 1*U_3 = 0.      !53
1 120. 0.01389 2. 0.5 ! elem: K_e, A_e, h_e, P_e      !54
20.0  4.0 10.0 ! misc: U_ext (L, U_0 for exact)       !55
```

Problem P3.12c Invalid mesh corrected by MPC


```
title "1-D heat transfer example, MPC mesh fix, Three L2"   ! 1
                                                            ! 2
*** CONSTRAINT EQUATION DATA ***                            ! 3
CONSTRAINT TYPE_1: (PAR_1 @ NODE_1) =  A_1.                 ! 4
EQ. NO.   NODE_1   PAR_1          A_1                       ! 5
     1      1      1          1.00000E+01                   ! 6
TYPE_2: A_1*(PAR_1 @ NODE_1) + A_2*(PAR_2 @ NODE_2) = A_3   ! 7
EQ. NODE_1  PAR_1  NODE_2  PAR_2    A_1       A_2       A_3 ! 8
 2    2     1      3       1    1.00E+0  -1.00E+0   0.00E+0 ! 9
                                                           !10
**  OUTPUT OF RESULTS AND EXACT VALUES IN NODAL ORDER  **   !11
    NODE,  X-Coord,    DOF_1,      EXACT1,                  !12
       1  0.0000E+00  1.0000E+01  1.0000E+01                !13
       2  1.3333E+00  1.6582E+01  1.6390E+01                !14
       3  1.3333E+00  1.6582E+01  1.6390E+01                !15
       4  2.6667E+00  1.8730E+01  1.8575E+01                !16
       5  4.0000E+00  1.9229E+01  1.9099E+01                !17
```

Problem P3.12d Selected MPC data and temperature results

Problem 3.13

Implement the exact integral matrices for the linear line element version of Eq. 3.34 and the matrix for recovering the convection heat loss, Eq. 3.38, in the post processing. Apply a small model to the problem in Fig. 3.7.1, but replace the Dirichlet boundary condition at $x = 0$ with the corresponding exact flux $q(0) = 12.86$. Compare the temperature solution with that in Fig. 3.7.1. Why do we still get a solution when we no longer have a Dirichlet boundary condition?

Solution: The definitions of the matrices are given in the text for the L2 element. Here we have simply multiplied the constant element coefficients times arrays defined with the F90 RESHAPE intrinsic to convert the vector data, stored by columns, into a matrix of the desired shape. One additional change is that we follow the usual practice here and assume that the external reference temperature can be different for each element. It is leement real property number 5 in this case. The element matrices are in P3.13a for both the element generation and post-processing phases.

A set of sample data for this element and a flux boundary condition is given in P3.13b. The assigned flux (computed from the exact flux at $x = 0$) is noted in lines 10 and 56. The remarks, lines 38 to 40, address why we can get a unique solution without a Dirichlet boundary condition. Selected output from these data are in P3.13c. It is very unusual for a problem not to have at least one Dirichlet boundary condition so a warning is observed on line 3. The input flux data is echoed in line 7. As expected its value is equal and opposite to the convection heat loss listed in lines 50 to 57. The computed temperatures are reasonably accurate. Note that the value at $x = 0$ is in error by one percent compared to the Dirichlet boundary condition value used to compute the exact solution. That is mainly due to weak form implementation of the insulated condition at the right end of the bar ($x = 4$).

In lines 23 to 48 of P3.13c we see the element level reactions, which represent heat flows. These data were requested in line 9 of P3.13b. Since those reactions are not equal and opposite to each other there must have been an internal heat source per unit length

and/or a surface convection heat loss. In this case their differences, cited in the rightmost SUMS column, should equal the heat flow lost from each element by convection. That is confirmed again bu looking at lines 50 to 57. Normally we do not look at such element reactions in detail but they can be important in applications like heat transfer and structural mechanics.

16.4 Problems from Chapter 4

Problem 4.1,2

1. For a one-dimensional quadratic element use the unit coordinate interpolation functions in Fig. 4.4.1 to evaluate the matrices:

$$a)\quad \mathbf{C}^e = \int_{L^e} \mathbf{H}^T\, dx,\ b)\quad \mathbf{M}^e = \int_{L^e} \mathbf{H}^T\, \mathbf{H}\, dx\,,$$

$$c)\quad \mathbf{S}^e = \int_{L^e} \frac{d\mathbf{H}^T}{dx}\,\frac{d\mathbf{H}}{dx}\, dx\,,\ d)\quad \mathbf{U}^e = \int_{L^e} \mathbf{H}^T\,\frac{d\mathbf{H}}{dx}\, dx\,.$$

2. Solve the above problem by using the natural coordinate version, $-1 \le n \le 1$, from Fig, 4.4.1.

Solution: Either local coordinate system must lead to the same final result, if one treats the Jacobian correctly. The exact integrals are listed in subroutine form in Problem P4.1. Note that the last square matrix is unsymmetrical.

Problem P4.3

Solve Problem P3.8 using the least squares finite element method instead. Solution: The approach here is quite different from the Galerkin solution. The good news is that the equation system is always symmetric so we can use a more efficient storage and solver mode. However, we can not use integration by parts and thus the second derivative term will remain in the weak form and that in turn requires a C^1 interpolation function. That is, the solution and its slope (first derivative) must be between elements. Thus we are forced to use a cubic Hermite polynomial with two degrees of freedom per node (value and slope). We have not considered such generalized (or vector) nodal degrees of freedom before. The MODEL program is designed to simplify the use of the most common C^0 elements by gathering the quadrature data, evaluating the interpolation functions and their local derivatives there, etc. For C^1 or C^2 type elements more specific programming details must be supplied to form the element matrices.

Referring to P4.3a we will point out some of the extra details that are not usually needed. In lines 8 and 10 we see that while storage is always allocated for the generalized interpolation functions and their gradient one must specifically assign storage for their second (or higher) derivatives, if needed. The least squares approach is usually most simply programmed by introducing a work space vector that holds the result of the differential operator acting on the generalized interpolations. That is, for every appearance of Y in the differential equation there is a corresponding action on V (the


```
!   ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***   ! 1
!.............................................................   ! 2
! Combined heat conduction through, convection from a bar:       ! 3
!   K*A*U,xx - h*P*(U-U_ext) = 0, hard coded L2 element          ! 4
!  Real element properties are:                                  ! 5
!  1) K_e   = conductivity, BTU/ hr ft F                         ! 6
!  2) A_e   = area of bar, ft^2                                  ! 7
!  3) h_e   = convection, BTU/ hr ft^2 F                         ! 8
!  4) P_e   = perimeter of area A_e, ft                          ! 9
!  5) Q_e   = source per unit length, BTU/ hr ft                 !10
!  6) U_ext = external reference temperature, F                  !11
 REAL(DP) :: DL                               ! Length          !12
 REAL(DP) :: K_e, A_e, h_e, P_e, Q_e, U_ext  ! properties       !13
                                                                !14
 DL    = COORD (LT_N, 1) - COORD (1, 1) ! Length                !15
 K_e   = GET_REAL_LP (1) ! thermal conductivity                 !16
 A_e   = GET_REAL_LP (2) ! area of bar                          !17
 h_e   = GET_REAL_LP (3) ! convection coefficient on perimeter  !18
 P_e   = GET_REAL_LP (4) ! perimeter of area A_e                !19
 Q_e   = GET_REAL_LP (5) ! source per unit length, BTU/ hr ft   !20
 U_ext = GET_REAL_LP (6) ! external temperature                 !21
                                                                !22
!         INTERNAL & CONVECTION SOURCES                        !23
 C = (Q_e + h_e * P_e * U_ext) * DL * (/ 1, 1 /) * 0.5d0       !24
                                                                !25
!         SQUARE MATRIX, CONDUCTION & CONVECTION               !26
 S = K_e * A_e / DL * RESHAPE ((/ 1, -1, -1, 1 /), (/2,2/))    &  !27
   + h_e * P_e * DL * RESHAPE ((/ 2,  1,  1, 2 /), (/2,2/))/6.d0  !28
!***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***       !29
                                                                !30
! *** POST_PROCESS_ELEM PROBLEM DEPENDENT STATEMENTS FOLLOW ***  !31
!H_INTG (LT_N) Integral of interpolation functions, H, available  !32
!.............................................................   !33
!   K*A*U,xx - h*P*(U-U_ext) = 0, hard coded L2 element          !34
 REAL(DP) :: DL, h_e, P_e, U_ext    ! Length, properties        !35
 REAL(DP), SAVE :: Q_LOSS, TOTAL    ! Face and total heat loss  !36
 LOGICAL,  SAVE :: FIRST = .TRUE.   ! printing                  !37
                                                                !38
  IF ( FIRST ) THEN                              ! first call   !39
    TOTAL = 0 ; FIRST = .FALSE. ; WRITE (6, 5) ! print headings  !40
    5 FORMAT ('*** CONVECTION HEAT LOSS ***', /, &              !41
    &         'ELEMENT    HEAT_LOST') ; END IF ! first call     !42
                                                                !43
!   GET LENGTH & CONSTANT PROPERTIES                           !44
  DL    = COORD (LT_N, 1) - COORD (1, 1) ! Length of bar       !45
  h_e   = GET_REAL_LP (3)   ! convection coefficient on perimeter !46
  P_e   = GET_REAL_LP (4)   ! perimeter of area                !47
  U_ext = GET_REAL_LP (5)   ! external temperature             !48
                                                               !49
!  HEAT LOST ON THIS FACE: Integral over face of h * (T - T_inf) !50
  H_INTG (1:2) = DL / 2              ! Integral of H array     !51
  D (1:2) = D(1:2) - U_ext           ! Temp difference at nodes  !52
  Q_LOSS  = h_e * P_e * DOT_PRODUCT (H_INTG, D)  ! Face loss   !53
  TOTAL   = TOTAL + Q_LOSS           ! Running total           !54
                                                               !55
  PRINT '(I6, ES15.5)', THIS_EL, Q_LOSS                        !56
  IF ( THIS_EL == N_ELEMS ) PRINT *, 'TOTAL = ', TOTAL         !57
! *** END POST_PROCESS_ELEM PROBLEM DEPENDENT STATEMENTS ***   !58
```

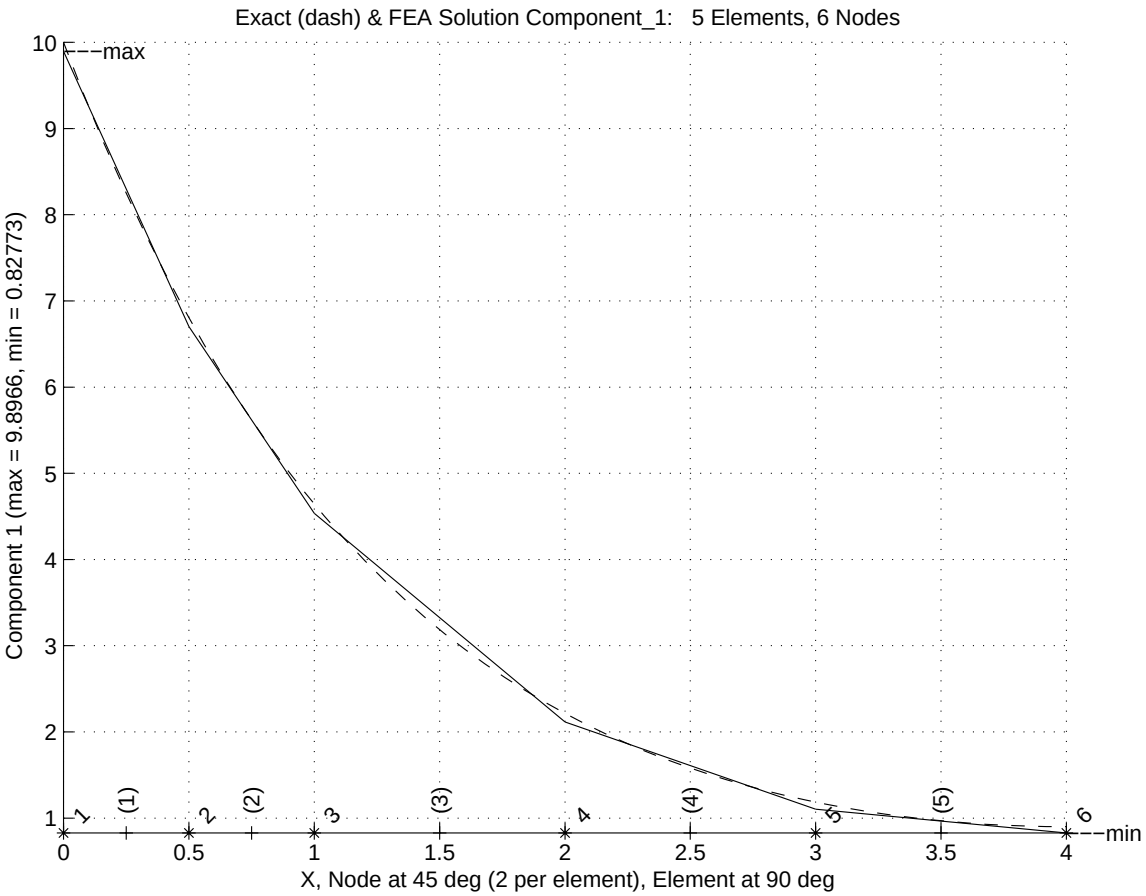
Problem P3.13a Exact element and post-processing matrices for L2


```
title "1-D heat transfer, fluxes only, with 5 L2"       ! 1
nodes      6 ! Number of nodes in the mesh              ! 2
elems      5 ! Number of elements in the system         ! 3
dof        1 ! Number of unknowns per node              ! 4
el_nodes   2 ! Maximum number of nodes per element      ! 5
space      1 ! Solution space dimension                 ! 6
b_rows     1 ! Number of rows in B (operator) matrix    ! 7
shape      1 ! Shape, 1=line, 2=tri, 3=quad, 4=hex      ! 8
el_react     ! Compute & list element reactions         ! 9
loads        ! An initial source vector is input        !10
gauss      0 ! Maximum number of quadrature point       !11
el_real    6 ! Number of real properties per element    !12
reals      3 ! Number of miscellaneous real properties  !13
el_homo      ! Element properties are homogeneous       !14
post_el      ! Require post-processing, create n_tape1  !15
exact_case 1 ! Exact analytic solution                  !16
list_exact      ! List given exact answers at nodes     !17
list_exact_flux ! List given exact fluxes at nodes      !18
remarks    22 ! Number of user remarks                  !19
quit ! keyword input, remarks follow                    !20
Combined heat conduction, convection from a bar:        !21
K*A*U,xx - h*P*(U-U_ext) + Q_e = 0,  dU/dx(L)=0,        !22
K*A*dU/dx(0)= q = 12.86 (reaction for U(0) = 10)        !23
For globally constant data the analytic solution is:    !24
U(x) = U_ext - (U_0-U_ext)*Cosh [m*(L-x)]/Cosh [mL]     !25
m^2 = h_e * P_e/(K_e * A_e), dimensionless              !26
Real FE problem properties are:                         !27
K_e   = GET_REAL_LP (1) conductivity, BTU/ hr ft F      !28
A_e   = GET_REAL_LP (2) area of bar, ft^2               !29
h_e   = GET_REAL_LP (3) convection, BTU/ hr ft^2 F      !30
P_e   = GET_REAL_LP (4) perimeter of area A_e, ft       !31
Q_e   = GET_REAL_LP (5) source per unit len, BTU/ hr ft !32
U_ext = GET_REAL_LP (6) external reference temp, F      !33
Miscellaneous data used ONLY for analytic solution:     !34
U_ext = GET_REAL_MISC (1) external ref temperature, F   !35
L     = GET_REAL_MISC (2) exact length, ft              !36
U_0   = GET_REAL_MISC (3) essential bc at x = 0, F      !37
Note: Flux loadings only. Convection allows a unique    !38
solution (prevents "rigid body motion"), otherwise      !39
solution would be known within an arbitrary constant.   !40
Note: Input flux, 12.86 BTU (exact), offsets the sum of !41
      element convection losses, 12.86 BTU (phys chk)   !42
         1         0 0.00       ! begin nodes           !43
         2         0 0.5                                !44
         3         0 1.00                               !45
         4         0 2.00                               !46
         5         0 3.00                               !47
         6         0 4.00                               !48
    1    1    2    ! begin elements                     !49
    2    2    3                                         !50
    3    3    4                                         !51
    4    4    5                                         !52
    5    5    6                                         !53
1 120. 0.01389 2. 0.5 0. 0. ! homo el, K A h P Q U_ext  !54
0.0  4.0 10.0   ! misc properties for exact_case 1      !55
 1  1 12.86     ! point flux as input source (loads)    !56
 6  1 0.0       ! terminate source with last dof        !57
```

Problem 3.13b Data for input flux condition version


```
TITLE 1-D heat transfer example, flux only, with 5 L2 ! 1
                                                      ! 2
WARNING: PARAMETER 1 HAS NO BOUNDARY CONDITION        ! 3
                                                      ! 4
 *** INITIAL FORCING VECTOR DATA ***                  ! 5
      NODE   PARAMETER     VALUE    EQUATION          ! 6
          1         1   1.28600E+01       1          ! 7
          6         1   0.00000E+00       6          ! 8
                                                      ! 9
 *** EXTREME VALUES OF THE NODAL PARAMETERS ***       !10
 PARAMETER      MAXIMUM,   NODE      MINIMUM,   NODE  !11
  DOF_1,     9.8966E+00,      1   8.2773E-01,      6  !12
                                                      !13
** OUTPUT OF RESULTS & EXACT VALUES IN NODAL ORDER ** !14
     NODE,  X-Coord,    DOF_1,      EXACT1,          !15
        1  0.0000E+00  9.8966E+00  1.0000E+01         !16
        2  5.0000E-01  6.7012E+00  6.8051E+00         !17
        3  1.0000E+00  4.5367E+00  4.6438E+00         !18
        4  2.0000E+00  2.1152E+00  2.2157E+00         !19
        5  3.0000E+00  1.1036E+00  1.1847E+00         !20
        6  4.0000E+00  8.2773E-01  9.0072E-01         !21
                                                      !22
** ELEMENT REACTION, INTERNAL SOURCES AND SUMS **     !23
                ELEMENT  1                            !24
  NODE  DOF   REACTION      ELEM_SOURCE    SUMS       !25
    1    1    1.28600E+1    0.00000E+0                !26
    2    1   -8.71053E+0    0.00000E+0                !27
 SUM:    1    4.14947E+0    0.00000E+0    4.14947E+0  !28
                ELEMENT  2                            !29
  NODE  DOF   REACTION      ELEM_SOURCE    SUMS       !30
    2    1    8.71053E+0    0.00000E+0                !31
    3    1   -5.90103E+0    0.00000E+0                !32
 SUM:    1    2.80950E+0    0.00000E+0    2.80950E+0  !33
                ELEMENT  3                            !34
  NODE  DOF   REACTION      ELEM_SOURCE    SUMS       !35
    3    1    5.90103E+0    0.00000E+0                !36
    4    1   -2.57507E+0    0.00000E+0                !37
 SUM:    1    3.32596E+0    0.00000E+0    3.32596E+0  !38
                ELEMENT  4                            !39
  NODE  DOF   REACTION      ELEM_SOURCE    SUMS       !40
    4    1    2.57507E+0    0.00000E+0                !41
    5    1   -9.65674E-1    0.00000E+0                !42
 SUM:    1    1.60940E+0    0.00000E+0    1.60940E+0  !43
                ELEMENT  5                            !44
  NODE  DOF   REACTION      ELEM_SOURCE    SUMS       !45
    5    1    9.65674E-1    0.00000E+0                !46
    6    1    0.00000E+0    0.00000E+0                !47
 SUM:    1    9.65674E-1    0.00000E+0    9.65674E-1  !48
                                                      !49
 *** CONVECTION HEAT LOSS ***                         !50
 ELEMENT    HEAT_LOST                                 !51
      1    4.14947E+00                                !52
      2    2.80950E+00                                !53
      3    3.32596E+00                                !54
      4    1.60940E+00                                !55
      5    9.65674E-01                                !56
  TOTAL =  12.859999994                               !57
```

Problem 3.13c Selected results from flux condition version



Problem P3.13d Exact and L2 result for flux boundary condition


```
SUBROUTINE INTEGRAL_H_ON_L3 (LENGTH, C_E)                              ! 1
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-*              ! 2
! INTEGRATE SHAPE FUNCTIONS OF A 3 NODE LINE ELEMENT                   ! 3
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-*              ! 4
Use Precision_Module  ! for DP                                         ! 5
 IMPLICIT NONE                                                         ! 6
 REAL(DP), INTENT(IN)  :: LENGTH   ! PHYSICAL LENGTH                   ! 7
 REAL(DP), INTENT(OUT) :: C_E (3)  ! INTEGRAL OF H                     ! 8
! LOCAL NODE COORD. ARE -1,0,+1    1-----2-----3                       ! 9
                                                                       !10
  C_E = LENGTH * (/ 1.d0, 4.d0, 1.d0 /) / 6.d0                         !11
END SUBROUTINE INTEGRAL_H_ON_L3                                        !12
                                                                       !13
SUBROUTINE INTEGRAL_HT_H_ON_L3 (LENGTH, MASS_E)                        !14
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-*              !15
! INTEGRATE H-TRANSPOSE H ON A 3 NODE LINE ELEMENT                     !16
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-*              !17
Use Precision_Module  ! for DP                                         !18
 IMPLICIT NONE                                                         !19
 REAL(DP), INTENT(IN)  :: LENGTH         ! PHYSICAL LENGTH             !20
 REAL(DP), INTENT(OUT) :: MASS_E (3, 3)  ! INTEGRAL OF H' H            !21
 REAL(DP), PARAMETER   :: MASS (3, 3) = RESHAPE (       &              !22
   (/ 4, 2, -1, 2, 16, 2, -1, 2, 4 /), (/3, 3/) ) / 30.d0              !23
! LOCAL NODE COORD. ARE -1,0,+1    1-----2-----3                       !24
                                                                       !25
  MASS_E = LENGTH * MASS                                               !26
END SUBROUTINE INTEGRAL_HT_H_ON_L3                                     !27
                                                                       !28
SUBROUTINE INTEGRAL_DHDXT_DHDX_ON_L3 (LENGTH, STIFF_E)                 !29
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-*              !30
! INTEGRATE DHDX-TRANSPOSE DHDX ON A 3 NODE LINE ELEMENT               !31
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-*              !32
Use Precision_Module  ! for DP                                         !33
 IMPLICIT NONE                                                         !34
 REAL(DP), INTENT(IN)  :: LENGTH         ! PHYSICAL LENGTH             !35
 REAL(DP), INTENT(OUT) :: STIFF_E (3, 3) ! INTEGRAL DHDX' DHDX         !36
 REAL(DP), PARAMETER   :: STIFF (3, 3) = RESHAPE (       &             !37
   (/ 7, -8, 1, -8, 16, -8, 1, -8, 7/), (/3, 3/) ) / 3.d0              !38
! LOCAL NODE COORD. ARE -1,0,+1    1-----2-----3                       !39
                                                                       !40
  STIFF_E = STIFF / LENGTH                                             !41
END SUBROUTINE INTEGRAL_DHDXT_DHDX_ON_L3                               !42
                                                                       !43
SUBROUTINE INTEGRAL_HT_DHDX_ON_L3 (U_E)                                !44
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-*              !45
! INTEGRATE H-TRANSPOSE DHDX ON A 3 NODE LINE ELEMENT                  !46
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-*              !47
Use Precision_Module  ! for DP                                         !48
 IMPLICIT NONE                                                         !49
 REAL(DP), INTENT(OUT) :: U_E (3, 3) ! INTEGRAL H' DHDX                !50
 REAL(DP), PARAMETER   :: U (3, 3) = RESHAPE (       &                 !51
   (/ -3, -4, 1, 4, 0, -4, -1, 4, 3/), (/3, 3/) ) / 6.d0               !52
! LOCAL NODE COORD. ARE -1,0,+1    1-----2-----3                       !53
                                                                       !54
  U_E = U                                                              !55
END SUBROUTINE INTEGRAL_HT_DHDX_ON_L3                                  !56
```

Problem P4.1 Typical quadratic C^0 element integrals

generalized interpolation array) in the work vector. In line 11 we name the workspace *OP_V* (for OPerator acting on V) and assign it the same storage allotment as *V*. Note

that at any point, the work vector times the element degrees of freedom plus any source term is the residual error in the differential equation at that point. In other words $OP\_V = \partial R / \partial \mathbf{\Delta}$ which was defined in Eq. 2.25 as the weighting term in a least squares formulation. Using this work vector the element square matrix and source vector are always defined as:

XXX

For this specific application the terms required in the work vector are seen by comparing the comments in lines 3 and 41.

From Fig. 4.5.1 we see that the Hermite family of elements are provided here in unit coordinates, not in the natural coordinates used to store the Gaussian quadrature data (for lines). Therefore it is necessary to either convert the quadrature data or re-program all the Hermite interpolations and derivatives. Lines 12 and 30-31 declare and carry out the quadrature conversion. Line 13 does the linear unit coordinate interpolation for the physical X position needed to evaluate the variable coefficients and source term. Lines 37-39 evaluate the Hermite polynomials and the first and second physical derivatives (by using the element length, *DL*). One only needs lines 22, 24, 50-52 if SCP or other post-processing is desired. For the cubic Hermite used here, that already has accurate continuous nodal slopes, the SCP averaging process would only give useful estimates of the nodal second derivatives (item *FLUX*_2 column in output lines 54-58 of P4.3c). The SCP estimated nodal slopes (*FLUX*_1 lines 54-58 of P4.3c) will usually be less accurate that the computed nodal slopes (item *DOF*_2 in lines 25-30 of P4.3c). The SCP process and associated error estimator could be improved by adding specific options for Hermite elements but such coding is not provided here.

In closing, it should also be pointed out that the essential boundary condition flags have also generalized to account for multiple unknowns per node. In general it is a packed integer flag with as many digits as unknowns per node. The unknowns are counted from left to right. Thus the boundary condition flags (in lines 22-26 in P4.3b and lines 7-12 of P4.3c) indicate that only the first unknown at nodes 1 and 4 have known essential boundary conditions applied. Thus only the first *dof* (*PAR*_1 in lines 21-23 of P4.3c) are cited in the data. In a different example the slope (second *dof*) might be specified instead.

For example, we have the same exact solution if we change the boundary condition at $x = 1$ to a slope condition, $dY/dx(1) = -4/3$. For the least squares C^1 form that is an essential boundary condition (but it is a Neumann condition in the Galerkin implementation). Thus, simply changing the data to have a different essential boundary condition give essentially the same results as before, as summarized in P4.3d.


```
!  ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***  ! 1
! ..................................................................! 2
!  LEAST SQ. SOL. OF: Y'' - 2XY'/(X^2+1) + 2Y/(X^2+1) = (X^2+1) ! 3
!         USING 3-RD ORDER HERMITE IN UNIT COORDINATES           ! 4
!             CONVERTED FROM NATURAL COORDINATES                 ! 5
!     >>> NOTE NOT SCALAR INTERPOLATION, C_1 ELEMENTS <<<        ! 6
!         ---------------------------------------                ! 7
! Allocated V (LT_FREE) and DGV (N_SPACE, LT_FREE) by MODEL      ! 8
                                                                 ! 9
 REAL(DP) :: D2GV (LT_FREE)       ! v'', 2nd derivative          !10
 REAL(DP) :: OP_V (LT_FREE)       ! ODE on H array               !11
 REAL(DP) :: PT_UNIT, WT_UNIT     ! unit coord pt, weight        !12
 REAL(DP) :: DL, X_SQ, X_G        ! physical length, pt          !13
 INTEGER  :: IP                   ! loops                        !14
                                                                 !15
! V    = GENERALIZED (VECTOR) INTERPOLATION FUNCTIONS            !16
! DGV  = DERIVATIVE OF GENERALIZED (VECTOR) INTERPOLATION        !17
! D2GV = 2nd DERIVATIVE OF GENERALIZED (VECTOR) INTERPOLATION    !18
                                                                 !19
  IF ( DEBUG_EL_SQ ) WRITE (N_BUG, *) 'Enter my_sq_matrix_inc'   !20
                                                                 !21
  E  = GET_REAL_IDENTITY (N_R_B)     ! DUMMY CONSTITUTIVE        !22
  DL = COORD (2, 1) - COORD (1, 1)   ! GET THE LENGTH            !23
  CALL STORE_FLUX_POINT_COUNT        ! Save LT_QP                !24
                                                                 !25
!      NUMERICAL INTEGRATION LOOP                               !26
  DO IP = 1, LT_QP                                               !27
                                                                 !28
!-->     FIND UNIT COORDINATES AND WEIGHT FOR INTEGRATION        !29
    PT_UNIT = (1.d0 + PT(1,IP)) / 2.d0    ! Convert pt          !30
    WT_UNIT = WT (IP) * 2.0d0             ! Convert wt          !31
    XYZ (1) = COORD (1, 1) + PT_UNIT * DL ! Physical pt         !32
    X_SQ    = XYZ (1) * XYZ (1)           ! x ^ 2               !33
    X_G     = X_SQ + 1.d0                 ! scalar g(x)         !34
                                                                 !35
!-->     EVALUATE HERMITE SHAPE FUNCTIONS AND DERIVATIVES        !36
    CALL SHAPE_C1_L  (PT_UNIT, DL, V)     ! V (NOT H)           !37
    CALL DERIV_C1_L  (PT_UNIT, DL, DGV)   ! DV / DX             !38
    CALL DERIV2_C1_L (PT_UNIT, DL, D2GV)  ! D^2 V / DX^2        !39
                                                                 !40
!     WORK VECTOR, OP_V =  V'' - 2X V'/(X^2+1) + 2 V/(X^2+1)    !41
   OP_V = D2GV - 2.d0 * XYZ (1) * DGV(1,:) / X_G  &              !42
        + 2.d0 * V / X_G                                         !43
                                                                 !44
!     COMPLETE THE SQUARE MATRIX AND SOURCE VECTOR              !45
!        S_IJ = S_IJ + WT_UNIT * OP_V_I * OP_V_J * DL           !46
    S = S + WT_UNIT * OUTER_PRODUCT (OP_V, OP_V) * DL            !47
    C = C + WT_UNIT * X_G * OP_V * DL                            !48
                                                                 !49
!      STORE DATA FOR SCP OR POST PROCESSING                     !50
    B (1, :) = DGV (1, :) ; B (2, :) = D2GV (:)                  !51
    CALL STORE_FLUX_POINT_DATA (XYZ, E, B)     ! for SCP         !52
  END DO                                                         !53
! ***  END ELEM_SQ_EX_107 PROBLEM DEPENDENT STATEMENTS ***       !54
```

Problem P4.3a A C^1 line element least squares version


```
title "Y'' - 2XY'/(X^2+1) + 2Y/(X^2+1) = (X^2+1)"     ! 1
nodes       4 ! Number of nodes in the mesh            ! 2
elems       3 ! Number of elements in the system       ! 3
dof         2 ! Number of unknowns per node            ! 4
el_nodes    2 ! Maximum number of nodes per element    ! 5
space       1 ! Solution space dimension               ! 6
b_rows      2 ! Number of rows in B (operator) matrix  ! 7
shape       1 ! Shape, 1=line, 2=tri, 3=quad, 4=hex    ! 8
gauss       5 ! Maximum number of quadrature points    ! 9
example   107 ! Source library example number          !10
data_set    1 ! Data set for example (this file)       !11
exact_case 15 ! Exact analytic solution                !12
scp_neigh_el  ! Default SCP patch group type           !13
list_exact    ! List exact answers at nodes            !14
list_exact_flux ! List exact fluxes at nodes           !15
remarks    4 ! Number of user remarks                  !16
quit ! keyword input                                   !17
Least Sq: Y'' - 2XY'/(X^2+1) + 2Y/(X^2+1) = (X^2+1)    !18
with Y(0)=2, Y(1)=5/3, Y(X) = X^4/6 - 3X^2/2 + X + 2   !19
Fausett p. 482, EXACT_CASE = 15                        !20
Here we use 3 cubic Hermite (C1_L) line elements.      !21
 1 10 0.          ! node, 2_binary_ bc_flags, x        !22
 2 00 0.333333333                                      !23
 3 00 0.666666667                                      !24
 4 10 1.00                                             !25
 1    1    2    ! elem j, k                           !26
 2    2    3                                          !27
 3    3    4                                          !28
 1 1 2.         ! Essential BC: node dof value        !29
 4 1 1.66666667 ! Essential BC: node dof value        !30
```

Problem P4.3b Three element least square C^1 data set

```
 THE NEXT 4 LINES ARE USER REMARKS                                  ! 1
 1 Least Sq: Y'' - 2XY'/(X^2+1) + 2Y/(X^2+1) = (X^2+1)              ! 2
 2 with Y(0)=2, Y(1)=5/3, Y(X) = X^4/6 - 3X^2/2 + X + 2             ! 3
 3 Fausett p. 482, EXACT_CASE = 15                                  ! 4
 4 Here we use 3 cubic Hermite (C1_L) line elements.                ! 5
                                                                    ! 6
*** NODAL POINT DATA ***                                            ! 7
NODE, BC_FLAG, X-Coord,                                             ! 8
   1  10    0.0000  ! note binary BC_FLAG                          ! 9
   2  00    0.3333  ! first is value, second slope                 !10
   3  00    0.6667                                                 !11
   4  10    1.0000                                                 !12
                                                                    !13
*** ELEMENT CONNECTIVITY DATA ***                                   !14
ELEMENT,  2 NODAL INCIDENCES.                                       !15
      1     1     2                                                !16
      2     2     3                                                !17
      3     3     4                                                !18
                                                                    !19
*** CONSTRAINT EQUATION DATA ***                                    !20
EQ. NO.   NODE_1   PAR_1          A_1                               !21
     1      1      1        2.00000E+00                            !22
     2      4      1        1.66667E+00                            !23
                                                                    !24
**  OUTPUT OF RESULTS AND EXACT VALUES IN NODAL ORDER  **          !25
PT,  X-Coord,    DOF_1,     DOF_2,     EXACT1,    EXACT2          !26
 1  0.0000E+0  2.0000E+0  9.9993E-1  2.0000E+0  1.0000E+0          !27
 2  3.3333E-1  2.1687E+0  2.4669E-2  2.1687E+0  2.4691E-2          !28
 3  6.6667E-1  2.0329E+0 -8.0245E-1  2.0329E+0 -8.0247E-1          !29
 4  1.0000E+0  1.6667E+0 -1.3333E+0  1.6667E+0 -1.3333E+0          !30
                                                                    !31
** FE AND EXACT FLUX COMPONENTS AT INTEGRATION POINTS **           !32
ELEM, PT,  X-Coord,   FX_1,     FX_2,     EX_1,     EX_2          !33
   1  1  1.564E-2  9.525E-1 -3.026E+0  9.531E-1 -3.000E+0          !34
   1  2  7.692E-2  7.683E-1 -2.986E+0  7.695E-1 -2.988E+0          !35
   1  3  1.667E-1  5.030E-1 -2.926E+0  5.031E-1 -2.944E+0          !36
   1  4  2.564E-1  2.432E-1 -2.866E+0  2.420E-1 -2.869E+0          !37
   1  5  3.177E-1  6.876E-2 -2.825E+0  6.829E-2 -2.798E+0          !38
ELEM, PT,  X-Coord,   FX_1,     FX_2,     EX_1,     EX_2          !39
   2  1  3.490E-1 -1.910E-2 -2.783E+0 -1.858E-2 -2.756E+0          !40
   2  2  4.103E-1 -1.859E-1 -2.661E+0 -1.847E-1 -2.663E+0          !41
   2  3  5.000E-1 -4.167E-1 -2.481E+0 -4.167E-1 -2.500E+0          !42
   2  4  5.897E-1 -6.313E-1 -2.302E+0 -6.325E-1 -2.304E+0          !43
   2  5  6.510E-1 -7.686E-1 -2.179E+0 -7.691E-1 -2.152E+0          !44
ELEM, PT,  X-Coord,   FX_1,     FX_2,     EX_1,     EX_2          !45
   3  1  6.823E-1 -8.356E-1 -2.096E+0 -8.352E-1 -2.069E+0          !46
   3  2  7.436E-1 -9.578E-1 -1.892E+0 -9.567E-1 -1.894E+0          !47
   3  3  8.333E-1 -1.114E+0 -1.592E+0 -1.114E+0 -1.611E+0          !48
   3  4  9.231E-1 -1.244E+0 -1.293E+0 -1.245E+0 -1.296E+0          !49
   3  5  9.844E-1 -1.317E+0 -1.089E+0 -1.317E+0 -1.062E+0          !50
LARGEST FLUX RMS_VALUE = 3.1728 AT ELEM = 1, POINT = 1             !51
                                                                    !52
** SUPER_CONVERGENT AVERAGED NODAL & EXACT FLUXES **               !53
PT, X-Coord,    FLUX_1,  FLUX_2,    EXACT1,   EXACT2               !54
 1  0.000E+0  9.126E-1 -3.241E+0  1.000E+0 -3.000E+0               !55
 2  3.333E-1  1.896E-2 -2.765E+0  2.469E-2 -2.778E+0               !56
 3  6.667E-1 -7.782E-1 -2.099E+0 -8.025E-1 -2.111E+0               !57
 4  1.000E+0 -1.480E+0 -1.241E+0 -1.333E+0 -1.000E+0               !58
```

Problem P4.3c Three element least square C^1 results and exact values


```
Least Sq: Y'' - 2XY'/(X^2+1) + 2Y/(X^2+1) = (X^2+1)        !18
Y(0)=2, Y'(1)=-4/3, so  Y(X) = X^4/6 - 3X^2/2 + X + 2     !19
 4 01 1.00      ! flag slope as known                      !25
 4 2 -1.3333333 ! Essential BC: node dof slope_value       !30
                                                           !
                                                           !
**  OUTPUT OF RESULTS AND EXACT VALUES IN NODAL ORDER  ** !25
PT,  X-Coord,    DOF_1,     DOF_2,     EXACT1,    EXACT2  !26
 1  0.0000E+0  2.0000E+0  9.9986E-1  2.0000E+0  1.0000E+0 !27
 2  3.3333E-1  2.1687E+0  2.4600E-2  2.1687E+0  2.4691E-2 !28
 3  6.6667E-1  2.0329E+0 -8.0251E-1  2.0329E+0 -8.0247E-1 !29
 4  1.0000E+0  1.6666E+0 -1.3333E+0  1.6667E+0 -1.3333E+0 !30
```

Problem P4.3d Least square C^1 slope condition changes

16.5 Problems from Chapter 5

Problem 5.3 should yield the same results as presented in problem 4.1.

16.6 Problems from Chapter 6

Problem 6.1

A hollow coaxial cable is made from a hollow conducting core and an insulating outer layer with $\varepsilon_1 = 0.5$, $\varepsilon_2 = 2.0$ and charge densities of $\zeta_1 = 100$ and $\zeta_2 = 0$, respectively. The inner, interface, and outer radii are 5, 10, and 25 mm. The corresponding inner and outer potentials (boundary conditions) are $\phi = 500$ and $\phi = 0$, respectively. Use the element formulation in Fig. 6.2.1 to compute the interface potential.

Solution: The element definition in Fig. 6.3.2 will work for all C^0 line elements in the library. We just need a new data file since the differenrial equation is the same. Data and results for 2 cubic line elements are in P6.1

16.7 Problems from Chapter 7

Problem 7.3

The truss in Fig. P7.1 has two $L = 10$ *inch* bays with members made of steel, $E = 30,000$ *ksi*, and constant cross-sections of $A = 1$ *in*^{2}. The bottom center load is $P_3 = 10,000$ *lb*, $P_5 = 0$, and the left and right corners are supported with a pin and roller, respectively. Compute the deflections and the reactions. Solution : Basically one need only new data such as those listed in P7.3a to produce the selected outputs in P7.3b that correspond to the deformed plot shown in P7.3c. The latter figure also includes the member stresses at their center (positive is tension). No member line loads are present so there is no local bending and they are pure axial loads of a typical truss.

Problem 7.8

Implement Eqs. 7.4.10 and 11 as an explicit formulation of the stiffness and loads on a two-dimensional truss. Solution is given in P7.8.


```
title "Coaxial cable radial electrical potential"    ! 1
axisymmetric ! Problem is axisymmetric, x radius     ! 2
nodes      5 ! Number of nodes in the mesh           ! 3
elems      2 ! Number of elements in the system      ! 4
dof        1 ! Number of unknowns per node           ! 5
el_nodes   3 ! Maximum number of nodes per element   ! 6
b_rows     1 ! Number of rows in B (operator) matrix ! 7
gauss      3 ! Maximum number of quadrature point    ! 8
el_real    2 ! Number of real properties per element ! 9
bar_long     ! Use distance between bar chart nodes  !10
remarks    8 ! Number of user remarks                !11
quit ! keyword input, remarks follow                 !12
! 1/R * d[R K_RR dT/dR]/dR + Q = 0, Example 109      !13
! Buchanan Example 2.19, electrical cable           !14
! Real FE problem properties are:                   !15
! K_RR = GET_REAL_LP (1) permittivity: 0.5 and 2.0   !16
! Q   = GET_REAL_LP (2) charge density: 100. and 0.  !17
! [material]          [1]      [2]                  !18
! Mesh T=500, R=5 *----*----*----*----* R= 25, T=0   !19
! Nodes, (Elem)   1    2(1) 3    4(2) 5             !20
1  1   5.     ! begin nodes, flag, x                 !21
2  0   7.5                                          !22
3  0   10.    ! T_analytic = 918.29, FEA 910.76     !23
4  0   17.5                                         !24
5  1   25.0                                         !25
1 1 2 3 ! begin elements                             !26
2 3 4 5                                              !27
1 1 500.     ! essential bc (electrical potential)   !28
5 1 0.                                               !29
 1 0.5 100.  ! el, permittivity, charge density      !30
 2 2.0 0.    ! el, permittivity, charge density      !31
                                                     !32
           Selected Output                           !33
***  OUTPUT OF RESULTS IN NODAL ORDER  ***           !34
NODE,  Radius r,   DOF_1,                            !35
   1  5.0000E+00  5.0000E+02                         !36
   2  7.5000E+00  1.3646E+03                         !37
   3  1.0000E+01  9.1076E+02 ! T_analytic = 918.29   !38
   4  1.7500E+01  3.5780E+02                         !39
   5  2.5000E+01  0.0000E+00                         !40
                                                     !41
DOF_1, EVALUATED AT 5 NODE POINTS                    !42
RANGE ON GRAPH IS  0.00000E+00 TO  1.36461E+03       !43
PT  VALUE +---+---+---+---+---+---+---+---+---+---+  !44
1 5.00E+2 XXXXXXXXXXXXXXX    +                   +  !45
          +                  +                   +  !46
2 1.37E+3 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  !47
          +                  +                   +  !48
3 9.11E+2 XXXXXXXXXXXXXXXXXXXXXXXXXX            +  !49
          +                  +                   +  !50
          +                  +                   +  !51
          +                  +                   +  !52
4 3.58E+2 XXXXXXXXXXXX       +                   +  !53
          +                  +                   +  !54
          +                  +                   +  !55
          +                  +                   +  !56
5 0.00E+0 X                  +                   +  !57
PT  VALUE +---+---+---+---+---+---+---+---+---+---+  !58
```

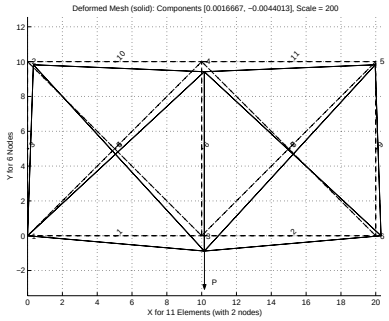
Problem P6.1 Coaxial cable data and results


```
title "Sample Meek 2-D truss"                          ! 1
nodes     6 ! Number of nodes in the mesh              ! 2
elems    11 ! Number of elements in the system         ! 3
el_homo     ! Element properties are homogeneous       ! 4
el_real   7 ! Number of real properties per element    ! 5
dof       2 ! Number of unknowns per node              ! 6
el_nodes  2 ! Maximum number of nodes per element      ! 7
space     2 ! Solution space dimension                 ! 8
b_rows    1 ! Number of rows in B (operator) matrix    ! 9
shape     1 ! Element shape, 1=line, 2=tri, 3=quad     !10
loads       ! An initial source vector is input        !11
el_react    ! Compute & list element reactions         !12
post_el     ! Require post-processing                  !13
example 206 ! Source library example number            !14
data_set  1 ! Data set for example (this file)         !15
remarks  15 ! Number of user remarks                   !16
quit ! keyword input, begin remarks                    !17
   2       4        5   Meek's Example 7.2 truss      !18
   *--(10)-*--(11)-*    E = 30,000 ksi,  A = 1 in^2   !19
   |\(4)  /|\(7)  /|    Two 10 inch bays              !20
   |  \ /  |  \ /  |    Vertical deflection at 3      !21
 (3)   X  (6)  X  (9)   is -4.4013E-03 inches          !22
   |  / \  |  / \  |                                   !23
   |/(5)  \|/(8)  \|    Reactions 5K each at 1, 6     !24
 1 #--(1)--*--(2)--o 6                                 !25
 Pin      3|       Roller                              !26
           v P=10K                                     !27
ELEMENT REAL PROPERTIES:                               !28
(1) = AREA,      (2) = MODULUS OF ELASTICITY,          !29
(3) = TEMP RISE, (4) = COEF THERMAL EXPANSION,         !30
(5) = LINE LOAD, (6) = MOMENT OF INERTIA,              !31
(7) = HALF DEPTH OF BAR                                !32
 1    11  0.0     0.0   ! node, bc flag, x, y         !33
 2    00  0.0     10.0                                !34
 3    00  10.0    0.0                                 !35
 4    00  10.0    10.0                                !36
 5    00  20.0    10.0                                !37
 6    01  20.0    0.0                                 !38
    1    1    3  ! element, two nodes                 !39
    2    3    6                                       !40
    3    1    2                                       !41
    4    2    3                                       !42
    5    1    4                                       !43
    6    3    4                                       !44
    7    4    6                                       !45
    8    3    5                                       !46
    9    5    6                                       !47
   10    2    4                                       !48
   11    4    5                                       !49
  1    1  0.0  ! node, direction, displacement        !50
  1    2  0.0                                         !51
  6    2  0.0                                         !52
1 1. 30000. 0 0 0 0 0 ! elem, A, E, null properties   !53
  3    2  -10.   ! node, direction, load              !54
  6    2  0.0    ! terminate with last                !55
```

Problem P7.3b Typical data for Meek's truss example


```
title "Sample Meek 2-D truss"                                  ! 1
                                                               ! 2
*** REACTION RECOVERY ***                                      ! 3
NODE, PARAMETER,     REACTION, EQUATION                        ! 4
   1,  DOF_1,       4.4409E-16     1                          ! 5
   1,  DOF_2,       5.0000E+00     2                          ! 6
   6,  DOF_2,       5.0000E+00    12                          ! 7
REACTION RESULTANTS                                            ! 8
PARAMETER,    SUM          POSITIVE     NEGATIVE             ! 9
 DOF_1,      4.4409E-16   4.4409E-16   0.0000E+00            !10
 DOF_2,      1.0000E+01   1.0000E+01   0.0000E+00            !11
                                                               !12
*** EXTREME VALUES OF THE NODAL PARAMETERS ***                 !13
PARAMETER      MAXIMUM,   NODE      MINIMUM,   NODE           !14
 DOF_1,     1.6667E-03,      2  -1.5639E-04,      5          !15
 DOF_2,     0.0000E+00,      1  -4.4013E-03,      3          !16
                                                               !17
***  OUTPUT OF RESULTS IN NODAL ORDER  ***                     !18
NODE,  X-Coord,    Y-Coord,    DOF_1,      DOF_2,            !19
  1  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00           !20
  2  0.0000E+00  1.0000E+01  1.6667E-03 -9.1153E-04           !21
  3  1.0000E+01  0.0000E+00  7.5514E-04 -4.4013E-03           !22
  4  1.0000E+01  1.0000E+01  7.5514E-04 -2.8910E-03           !23
  5  2.0000E+01  1.0000E+01 -1.5639E-04 -9.1153E-04           !24
  6  2.0000E+01  0.0000E+00  1.5103E-03  0.0000E+00           !25
                                                               !26
** BEGIN ELEMENT APPLICATION POST PROCESSING **               !27
        E L E M E N T    S T R E S S E S                       !28
 ELEMENT      MID SECTION (BENDING) STRESS AT:                !29
 NUMBER          RIGHT                LEFT                   !30
    1        0.2265409E+01       0.2265409E+01               !31
    2        0.2265409E+01       0.2265409E+01               !32
    3       -0.2734591E+01      -0.2734591E+01               !33
    4        0.3867295E+01       0.3867295E+01               !34
    5       -0.3203772E+01      -0.3203772E+01               !35
    6        0.4530818E+01       0.4530818E+01               !36
    7       -0.3203772E+01      -0.3203772E+01               !37
    8        0.3867295E+01       0.3867295E+01               !38
    9       -0.2734591E+01      -0.2734591E+01               !39
   10       -0.2734591E+01      -0.2734591E+01               !40
   11       -0.2734591E+01      -0.2734591E+01               !41
```

Problem P7.3b Selected output for the Meek truss



Problem P7.3c Meek's deformed planar truss example


```
! ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***   ! 1
! .....................................................        ! 2
                                                               ! 3
! APPLICATION: TWO-DIMENSIONAL TRUSS. ELEMENT REAL PROPERTIES:  ! 4
!  (1) = AREA,      (2) = MODULUS OF ELASTICITY,                ! 5
!  (3) = TEMP RISE, (4) = COEF THERMAL EXPANSION                ! 6
                                                               ! 7
 REAL(DP) :: X_I, X_J, Y_I, Y_J         ! coordinates          ! 8
 REAL(DP) :: D_X, D_Y, BAR_L            ! lengths              ! 9
 REAL(DP) :: DELTA_T, ALPHA             ! temp rise, coeff     !10
 REAL(DP) :: AREA                       ! area                 !11
 REAL(DP) :: M_E                        ! modulus of elasticity !12
 REAL(DP) :: C_X, C_Y, C_XX, C_XY, C_YY ! cosines & products   !13
 REAL(DP) :: F, STIF                    ! forces, stiffness    !14
 REAL(DP) :: THERMAL                    ! expansion            !15
                                                               !16
!  Get geometry                                                !17
   X_I = COORD (1, 1) ;        X_J = COORD (2, 1)              !18
   Y_I = COORD (1, 2) ;        Y_J = COORD (2, 2)              !19
                                                               !20
! Get properties for this element                             !21
   AREA    = GET_REAL_LP (1)  ! area                           !22
   M_E     = GET_REAL_LP (2)  ! elastic modulus                !23
   DELTA_T = GET_REAL_LP (3)  ! temperature rise               !24
   ALPHA   = GET_REAL_LP (4)  ! coeff thermal expansion        !25
                                                               !26
!-->  FIND BAR LENGTH AND DIRECTION COSINES                    !27
   D_X = X_J - X_I  ; D_Y = Y_J - Y_I    ! lengths            !28
   BAR_L = SQRT (D_X * D_X + D_Y * D_Y)  ! total length       !29
   C_X = D_X / BAR_L ; C_Y = D_Y / BAR_L ! cosines            !30
                                                               !31
   STIF = M_E * AREA / BAR_L  ! AXIAL STIFFNESS, K=E*A/L       !32
                                                               !33
!-->  TRANSFORM TO 2-D STIFFNESS (closed form)                 !34
   C_XX = C_X * C_X ; C_XY = C_X * C_Y ; C_YY = C_Y * C_Y      !35
                                                               !36
   S (1, 1) =   STIF * C_XX ; S (2, 1) =   STIF * C_XY         !37
   S (3, 1) = - STIF * C_XX ; S (4, 1) = - STIF * C_XY         !38
   S (1, 2) =   STIF * C_XY ; S (2, 2) =   STIF * C_YY         !39
   S (3, 2) = - STIF * C_XY ; S (4, 2) = - STIF * C_YY         !40
   S (1, 3) = - STIF * C_XX ; S (2, 3) = - STIF * C_XY         !41
   S (3, 3) =   STIF * C_XX ; S (4, 3) =   STIF * C_XY         !42
   S (1, 4) = - STIF * C_XY ; S (2, 4) = - STIF * C_YY         !43
   S (3, 4) =   STIF * C_XY ; S (4, 4) =   STIF * C_YY         !44
                                                               !45
!       Form any local loads                                  !46
   C = 0.d0 ; THERMAL = 0.d0                ! initialize       !47
                                                               !48
   IF ( DELTA_T /= 0.d0 ) THEN  ! THERMAL STRAIN EFFECTS       !49
     THERMAL = ALPHA * DELTA_T              ! thermal strain   !50
     F       = M_E * THERMAL * AREA         ! thermal force    !51
     C (1) = - C_X * F ; C (2) = - C_Y * F  ! components       !52
     C (3) =   C_X * F ; C (4) =   C_Y * F  ! components       !53
   END IF ! thermal                                            !54
                                                               !55
!  End of application dependent code                          !56
```

Problem P7.8 Truss member with thermal load

16.8 Problems from Chapter 8

Problem 8.1

Use the subroutines in Fig. 4.5.2 to form similar functions for a C^1 rectangular element by taking a tensor product of the one-dimensional Hermite interpolation relations. This will be a 16 degree of freedom element since each node will have $u, \partial u/\partial x, \partial u/\partial y,$ and $\partial^2 u/\partial x \partial y$ as nodal unknowns. Solution: Taking the products of the 1-D functions yields the interpolations and first derivatives shown in P8.1a and P8.1b. Such and element would usually be needed for a fourth order differential equation which would have the second derivatives in the integral form. Thus a similar subroutine for the second derivatives would be needed too.

```
SUBROUTINE SHAPE_16_R (R, S, A, B, H)                          ! 1
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-*              ! 2
! C1 RECTANGULAR ELEMENT IN UNIT COORDINATES                   ! 3
! USING TENSOR PRODUCTS OF 1D HERMITE BASIS                    ! 4
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-*              ! 5
Use Precision_Module                                           ! 6
 IMPLICIT NONE                                                 ! 7
 REAL(DP), INTENT(IN)  :: R, S, A, B                           ! 8
 REAL(DP), INTENT(OUT) :: H (16)                               ! 9
 REAL(DP)              :: HR (4), HS (4)                       !10
                                                               !11
! DOF ARE W W,X W,Y W,XY AT EACH NODE (N_G_DOF=4)              !12
! X // R, Y // S.               S                              !13
! A = PHYSICAL LENGTH IN X      4 -------- 3                   !14
! B = PHYSICAL LENGTH IN Y      I          I                   !15
! R,S = LOCAL UNIT COORDS       I          I                   !16
! 1 at (0,0), 3 at (1,1)        1 -------- 2 ->R              !17
                                                               !18
! Evaluate the 1D Hermite interpolations                       !19
 CALL SHAPE_C1_L (R, A, HR) ; CALL SHAPE_C1_L (S, B, HS)       !20
                                                               !21
! Form tensor products                                         !22
  H (1)  = HR (1) * HS (1) ; H (2)  = HR (2) * HS (1)          !23
  H (3)  = HR (1) * HS (2) ; H (4)  = HR (2) * HS (2)          !24
  H (5)  = HR (3) * HS (1) ; H (6)  = HR (4) * HS (1)          !25
  H (7)  = HR (3) * HS (2) ; H (8)  = HR (4) * HS (2)          !26
  H (9)  = HR (3) * HS (3) ; H (10) = HR (4) * HS (3)          !27
  H (11) = HR (3) * HS (4) ; H (12) = HR (4) * HS (4)          !28
  H (13) = HR (1) * HS (3) ; H (14) = HR (2) * HS (3)          !29
  H (15) = HR (1) * HS (4) ; H (16) = HR (2) * HS (4)          !30
END SUBROUTINE SHAPE_16_R                                      !31
```

Problem 8.1a Shape functions from 1-D C^1 products


```
SUBROUTINE DERIV_16_R (R, S, A, B, DH)                                  ! 1
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *            ! 2
! FIRST DERIVATIVES OF A C1 RECTANGULAR ELEMENT IN                      ! 3
! UNIT COORDINATES USING TENSOR PRODUCTS OF 1D BASIS                    ! 4
! *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *-* *            ! 5
Use Precision_Module                                                    ! 6
 IMPLICIT NONE                                                          ! 7
 REAL(DP), INTENT(IN)  :: R, S, A, B                                    ! 8
 REAL(DP), INTENT(OUT) :: DH (2, 16)                                    ! 9
 REAL(DP)              :: HR (4), DR (4), HS (4), DS (4)                !10
                                                                        !11
! DOF ARE W W,X W,Y W,XY AT EACH NODE (N_G_DOF=4)                       !12
! X // R, Y // S.              S                                        !13
! A = PHYSICAL LENGTH IN X     4 -------- 3                             !14
! B = PHYSICAL LENGTH IN Y     I          I                             !15
! R,S = LOCAL UNIT COORDS      I          I                             !16
! 1 at (0,0), 3 at (1,1)       1 -------- 2 ->R                         !17
                                                                        !18
! Evaluate the 1D Hermite interpolations                                !19
  CALL SHAPE_C1_L (R, A, HR) ; CALL SHAPE_C1_L (S, B, HS)               !20
  CALL DERIV_C1_L (R, A, DR) ; CALL DERIV_C1_L (S, B, DS)               !21
                                                                        !22
! Form tensor products for R direction                                  !23
  DH (1, 1)  = DR (1) * HS (1) ; DH (1, 2)  = DR (2) * HS (1)          !24
  DH (1, 3)  = DR (1) * HS (2) ; DH (1, 4)  = DR (2) * HS (2)          !25
  DH (1, 5)  = DR (3) * HS (1) ; DH (1, 6)  = DR (4) * HS (1)          !26
  DH (1, 7)  = DR (3) * HS (2) ; DH (1, 8)  = DR (4) * HS (2)          !27
  DH (1, 9)  = DR (3) * HS (3) ; DH (1, 10) = DR (4) * HS (3)          !28
  DH (1, 11) = DR (3) * HS (4) ; DH (1, 12) = DR (4) * HS (4)          !29
  DH (1, 13) = DR (1) * HS (3) ; DH (1, 14) = DR (2) * HS (3)          !30
  DH (1, 15) = DR (1) * HS (4) ; DH (1, 16) = DR (2) * HS (4)          !31
                                                                        !32
! Form tensor products for S direction                                  !33
  DH (2, 1)  = HR (1) * DS (1) ; DH (2, 2)  = HR (2) * DS (1)          !34
  DH (2, 3)  = HR (1) * DS (2) ; DH (2, 4)  = HR (2) * DS (2)          !35
  DH (2, 5)  = HR (3) * DS (1) ; DH (2, 6)  = HR (4) * DS (1)          !36
  DH (2, 7)  = HR (3) * DS (2) ; DH (2, 8)  = HR (4) * DS (2)          !37
  DH (2, 9)  = HR (3) * DS (3) ; DH (2, 10) = HR (4) * DS (3)          !38
  DH (2, 11) = HR (3) * DS (4) ; DH (2, 12) = HR (4) * DS (4)          !39
  DH (2, 13) = HR (1) * DS (3) ; DH (2, 14) = HR (2) * DS (3)          !40
  DH (2, 15) = HR (1) * DS (4) ; DH (2, 16) = HR (2) * DS (4)          !41
END SUBROUTINE DERIV_16_R                                               !42
```

Problem 8.1b Gradients from 1-D C^1 products

16.9 Problems from Chapter 9

XXX

16.10 Problems from Chapter 10

XXX

16.11 Problems from Chapter 11

1. Implement the isotropic $\mathbf{E}^e$ matrix for the plane strain assumption. Assume the stress components are in the XX, YY, XY order. Solution:

```
SUBROUTINE E_PLANE_STRAIN (E)                                          ! 1
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     ! 2
!     PLANE_STRAIN CONSTITUTIVE MATRIX DEFINITION                      ! 3
! STRESS & STRAIN COMPONENT ORDER: XX, YY, XY, SO N_R_B = 3           ! 4
! PROPERTY ORDER: 1-YOUNG'S MODULUS, 2-POISSON'S RATIO                ! 5
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     ! 6
Use System_Constants    ! for DP, N_R_B                                ! 7
Use Sys_Properties_Data ! for function GET_REAL_LP                     ! 8
 IMPLICIT NONE                                                         ! 9
 REAL(DP), INTENT(OUT) :: E (N_R_B, N_R_B)                             !10
 REAL(DP) :: C_1, C_2, P_R, S_M, Y_M                                   !11
                                                                       !12
! E     = CONSTITUTIVE MATRIX                                         !13
! N_R_B = NUMBER OF ROWS IN B AND E MATRICES                          !14
! P_R   = POISSON'S RATIO                                             !15
! S_M   = SHEAR MODULUS                                               !16
! Y_M   = YOUNG'S MODULUS OF ELASTICITY                               !17
                                                                       !18
  IF ( EL_REAL < 2 ) STOP 'KEYWORD el_real<2 IN E_PLANE_STRAIN'        !19
  Y_M = GET_REAL_LP (1) ; P_R = GET_REAL_LP (2)                        !20
  IF (P_R == 0.5d0) STOP "BAD POISSON'S RATIO, E_PLANE_STRAIN"         !21
                                                                       !22
  C_1 = Y_M * (1 - P_R) / ((1 + P_R)*(1 - P_R - P_R))                  !23
  C_2 = Y_M *  P_R / ((1 + P_R)*(1 - P_R - P_R))                       !24
  S_M = 0.5d0 * Y_M / ( 1 + P_R)                                       !25
                                                                       !26
  E (1, 1) = C_1     ; E (2, 1) = C_2     ; E (3, 1) = 0.d0            !27
  E (1, 2) = C_2     ; E (2, 2) = C_1     ; E (3, 2) = 0.d0            !28
  E (1, 3) = 0.d0    ; E (2, 3) = 0.d0    ; E (3, 3) = S_M             !29
END SUBROUTINE E_PLANE_STRAIN                                          !30
```

Problem P11.1 Plane strain assumption isotropic constitutive law

2. Implement the isotropic $\mathbf{E}^e$ matrix for the general solid. Assume the stress components are in the XX, YY, XY, ZZ, XZ, YZ order. Solution:


```
SUBROUTINE E_SOLID_STRESS (E)                                          ! 1
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     ! 2
!     SOLID_STRESS CONSTITUTIVE MATRIX DEFINITION                      ! 3
! STRESS COMPONENT ORDER: XX, YY, XY, ZZ, XZ, YZ, SO N_R_B = 6         ! 4
! PROPERTY ORDER: 1-YOUNG'S MODULUS, 2-POISSON'S RATIO                 ! 5
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *     ! 6
Use System_Constants    ! for DP, N_R_B                                ! 7
Use Sys_Properties_Data ! for function GET_REAL_LP                     ! 8
 IMPLICIT NONE                                                         ! 9
 REAL(DP), INTENT(OUT) :: E (N_R_B, N_R_B)                             !10
 REAL(DP) :: C_1, C_2, P_R, S_M, Y_M                                   !11
                                                                       !12
! E     = CONSTITUTIVE MATRIX                                         !13
! N_R_B = NUMBER OF ROWS IN B AND E MATRICES                          !14
! P_R   = POISSON'S RATIO                                             !15
! S_M   = SHEAR MODULUS                                               !16
! Y_M   = YOUNG'S MODULUS OF ELASTICITY                               !17
                                                                       !18
  IF ( N_R_B  /= 6 )  STOP 'INVALID N_R_B IN E_SOLID_STRESS'           !19
  IF (EL_REAL < 2) STOP 'KEY el_real < 2 IN E_SOLID_STRESS'            !20
  Y_M = GET_REAL_LP (1) ; P_R = GET_REAL_LP (2)                        !21
                                                                       !22
  C_1  = Y_M * (1 - P_R) / (1 + P_R) / (1 - P_R - P_R)                 !23
  C_2  = C_1 * P_R / (1 - P_R); S_M  = 0.5d0 * Y_M / (1 + P_R)         !24
                                                                       !25
  E = 0.d0 ! Mostly                                                    !26
  E (1, 1) = C_1 ; E (2, 1) = C_2 ; E (4, 1) = C_2  ! normals          !27
  E (1, 2) = C_2 ; E (2, 2) = C_1 ; E (4, 2) = C_2  ! normals          !28
  E (1, 4) = C_2 ; E (2, 4) = C_2 ; E (4, 4) = C_1  ! normals          !29
  E (3, 3) = S_M ; E (5, 5) = S_M ; E (6, 6) = S_M  ! shears           !30
END SUBROUTINE E_SOLID_STRESS                                          !31
```

Problem P11.2 Isotropic solid constitutive matrix

3. Give an implementation of the $\mathbf{B}^e$ matrix for a general solid. Assume the strain components are in the XX, YY, XY, ZZ, XZ, YZ order. Solution:


```
SUBROUTINE ELASTIC_B_SOLID (DGH, B)                                  ! 1
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   ! 2
! 3-D ELASTICITY STRAIN-DISPLACEMENT RELATIONS (B)                   ! 3
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   ! 4
Use System_Constants ! for DP, N_R_B, N_G_DOF, N_SPACE             ! 5
Use Elem_Type_Data   ! for LT_FREE, LT_N                           ! 6
 IMPLICIT NONE                                                     ! 7
 REAL(DP), INTENT(IN)  :: DGH (N_SPACE, LT_N)                      ! 8
 REAL(DP), INTENT(OUT) :: B   (N_R_B, LT_N * N_G_DOF)              ! 9
 INTEGER  :: J, K, L, M                                           !10
                                                                  !11
! B       = STRAIN-DISPLACEMENT MATRIX (RETURNED)                 !12
! DGH     = GLOBAL DERIVATIVES OF ELEM INTERPOLATIONS              !13
! LT_N    = NUMBER OF NODES PER ELEMENT                           !14
! N_G_DOF = NUMBER OF PARAMETERS PER NODE                         !15
! N_R_B   = NUMBER OF STRAINS (IN B) XX, YY, XY, ZZ, XZ, YZ       !16
! N_SPACE = DIMENSION OF SPACE                                    !17
                                                                  !18
  B = 0.d0                                                        !19
  DO J = 1, LT_N               ! ROW NUMBER                       !20
    K = N_G_DOF * (J - 1) + 1 ! FIRST COLUMN,  U TERMS            !21
    L = K + 1                  ! SECOND COLUMN, V TERMS           !22
    M = L + 1                  ! THIRD COLUMN,  W TERMS           !23
    B (1, K) = DGH (1, J)      ! DU/DX  FOR XX NORMAL             !24
    B (3, K) = DGH (2, J)      ! DU/DY  FOR XY SHEAR              !25
    B (5, K) = DGH (3, J)      ! DU/DZ  FOR XZ SHEAR              !26
    B (2, L) = DGH (2, J)      ! DV/DY  FOR YY NORMAL             !27
    B (3, L) = DGH (1, J)      ! DV/DX  FOR XY SHEAR              !28
    B (6, L) = DGH (3, J)      ! DV/DZ  FOR YZ SHEAR              !29
    B (4, M) = DGH (3, J)      ! DW/DZ  FOR ZZ NORMAL             !30
    B (5, M) = DGH (1, J)      ! DW/DX  FOR XZ SHEAR              !31
    B (6, M) = DGH (2, J)      ! DW/DY  FOR YZ SHEAR              !32
  END DO                                                          !33
END SUBROUTINE ELASTIC_B_SOLID                                    !34
```

Problem P11.3 Strain-displacement matrix for a general solid

4. Review the scalar implementation for the T3 element, given in Fig. 10.4.1, and extend it to the case of a two-dimensional stress model. Solution:


```
! ..............................................................  ! 1
!    ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***  ! 2
! ..............................................................  ! 3
! Note: BODY (N_SPACE) is available through the interface         ! 4
!         Linear Triangle, T3, Plane stress only                  ! 5
! STRESS AND STRAIN COMPONENT ORDER: XX, YY, XY, SO N_R_B = 3     ! 6
 REAL(DP):: X_I, X_J, X_K, Y_I, Y_J, Y_K     ! Global coordinates ! 7
 REAL(DP):: A_I, A_J, A_K, B_I, B_J, B_K     ! Standard geometry  ! 8
 REAL(DP):: C_I, C_J, C_K, X_CG, Y_CG, TWO_A ! Standard geometry  ! 9
 REAL(DP):: THICK                            ! Element thickness  !10
                                                                  !11
!   DEFINE NODAL COORDINATES, CCW: I, J, K                       !12
   X_I = COORD (1,1) ;  X_J = COORD (2,1) ;  X_K = COORD (3,1)    !13
   Y_I = COORD (1,2) ;  Y_J = COORD (2,2) ;  Y_K = COORD (3,2)    !14
                                                                  !15
!   DEFINE CENTROID COORDINATES (QUADRATURE POINT)               !16
   X_CG = (X_I + X_J + X_K)/3.d0 ; Y_CG = (Y_I + Y_J + Y_K)/3.d0  !17
                                                                  !18
!   GEOMETRIC PARAMETERS H_I (X,Y) = (A_I + B_I*X + C_I*Y)/TWO_A  !19
   A_I = X_J * Y_K - X_K * Y_J; B_I = Y_J - Y_K; C_I = X_K - X_J  !20
   A_J = X_K * Y_I - X_I * Y_K; B_J = Y_K - Y_I; C_J = X_I - X_K  !21
   A_K = X_I * Y_J - X_J * Y_I; B_K = Y_I - Y_J; C_K = X_J - X_I  !22
                                                                  !23
!   CALCULATE TWICE ELEMENT AREA                                 !24
   TWO_A = A_I + A_J + A_K  ! = B_J*C_K - B_K*C_J also           !25
                                                                  !26
!   DEFINE 3 BY 6 STRAIN-DISPLACEMENT MATRIX, B                  !27
   B (1, 1:6) = (/ B_I,  0.d0, B_J,  0.d0, B_K,  0.d0 /)/TWO_A    !28
   B (2, 1:6) = (/ 0.d0, C_I,  0.d0, C_J,  0.d0, C_K  /)/TWO_A    !29
   B (3, 1:6) = (/ C_I,  B_I,  C_J,  B_J,  C_K,  B_K  /)/TWO_A    !30
                                                                  !31
   CALL E_PLANE_STRESS (E)        ! THE CONSTITUTIVE MATRIX       !32
   THICK = 1 ; IF ( AREA_THICK /= 1.d0 ) THICK = AREA_THICK       !33
   TWO_A = TWO_A * THICK                      ! true volume       !34
                                                                  !35
!   STIFFNESS MATRIX, WITH CONSTANT JACOBIAN                     !36
   S = MATMUL ( TRANSPOSE (B), MATMUL (E, B) ) * TWO_A * 0.5d0    !37
                                                                  !38
!   BODY FORCE PER UNIT VOLUME                                   !39
   IF ( BODY_FORCE ) THEN ! Check keyword values in Body_F array  !40
     C (1:5:2) = BODY_F (1) * TWO_A / 6.d0  ! X component        !41
     C (2:6:2) = BODY_F (2) * TWO_A / 6.d0  ! Y component        !42
   END IF ! or set up properties for body force                   !43
                                                                  !44
!  SAVE ONE POINT RULE TO AVERAGING, OR ERROR ESTIMATOR          !45
   LT_QP = 1 ; CALL STORE_FLUX_POINT_COUNT ! Save LT_QP          !46
   CALL STORE_FLUX_POINT_DATA ( (/ X_CG, Y_CG /), E, B )         !47
                                                                  !48
   if ( DEBUG_EL_SQ ) then                                        !49
     print *,'S matrix:' ; call rprint (S, LT_FREE, LT_FREE, 1)   !50
     print *,'C matrix:' ; call rprint (C,       1, LT_FREE, 1)   !51
   end if                                                         !52
!  End of application dependent code                             !53
!    ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***     !54
```

Problem P11.4 Stiffness for isotropic plane stress CST

5. Implement a general evaluation of the elasticity $\mathbf{E}^e$ matrix that will handle all the 1-, 2-, and 3-dimensional cases. Solution:


```
SUBROUTINE ELASTICITY_E_MATRIX (E)                                  ! 1
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ! 2
!      ELASTICITY CONSTITUTIVE MATRIX DEFINITION                     ! 3
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ! 4
Use System_Constants    ! for DP, N_R_B, N_SPACE,                   ! 5
                        ! AXISYMMETRIC, PLANE_STRAIN                ! 6
 IMPLICIT NONE                                                      ! 7
 REAL(DP), INTENT(OUT) :: E (N_R_B, N_R_B)  ! CONSTITUTIVE          ! 8
! N_R_B = NUMBER OF ROWS IN B AND E MATRICES                        ! 9
! E     = SYMMETRIC CONSTITUTIVE MATRIX                            !10
                                                                    !11
IF ( ANISOTROPIC ) THEN                                             !12
  STOP 'ELASTICITY_E_MATRIX: ANISOTROPIC E UNAVAILABLE'             !13
! SELECT CASE (N_SPACE)                                             !14
!   CASE (1) ! 1-D or cylinder                                      !15
!     IF ( AXISYMMETRIC ) THEN      ! Components RR & TT            !16
!       CALL ANISOTROPIC_CYL_STRESS (E)                           !17
!     ELSE                          ! Component XX                  !18
!       CALL E_BAR_STRESS (E)       ; END IF                        !19
!   CASE (2) ! 2-D or axisymmetric                                  !20
!     IF ( AXISYMMETRIC ) THEN     ! Components RR ZZ RZ TT         !21
!       CALL ANISOTROPIC_AXI_STRESS (E)                           !22
!     ELSE IF ( PLANE_STRAIN ) THEN ! Components XX YY XY          !23
!       CALL ANISOTROPIC_2D_STRAIN (E)                            !24
!     ELSE ! plane stress            Components XX YY XY          !25
!       CALL ANISOTROPIC_2D_STRESS (E)      ; END IF              !26
!   CASE (3) ! 3-D                                                 !27
!     CALL ANISOTROPIC_3D_STRESS (E) ! XX YY XY ZZ ZX ZY          !28
! END SELECT                                                      !29
ELSE ! Isotropic                                                    !30
                                                                    !31
  SELECT CASE (N_SPACE)                                             !32
    CASE (1) ! 1-D or cylinder                                      !33
      IF ( AXISYMMETRIC ) THEN      ! Components RR & TT            !34
      ! CALL E_CYLINDER_STRESS (E)                                 !35
      ELSE                          ! Component XX                  !36
      ! CALL E_BAR_STRESS (E)                                      !37
      END IF                                                        !38
    CASE (2) ! 2-D or axisymmetric                                  !39
      IF ( AXISYMMETRIC ) THEN     ! Components RR ZZ RZ TT         !40
        CALL E_AXISYMMETRIC_STRESS (E)                              !41
      ELSE IF ( PLANE_STRAIN ) THEN ! Components XX YY XY           !42
        CALL E_PLANE_STRAIN (E)                                     !43
      ELSE ! plane stress            Components XX YY XY          !44
        CALL E_PLANE_STRESS (E)      ; END IF                       !45
    CASE (3) ! 3-D                                                  !46
      CALL E_SOLID_STRESS (E)        ! XX YY XY ZZ ZX ZY          !47
  END SELECT                                                        !48
END IF ! anisotropic material                                       !49
END SUBROUTINE ELASTICITY_E_MATRIX                                  !50
```

Problem P11.5 A general elasticity constitutive matrix control

6. Implement a general evaluation of the elasticity $\mathbf{B}^e$ matrix that will handle all the 1-, 2-, and 3-dimensional cases. Solution:


```
SUBROUTINE ELASTICITY_B_MATRIX (DGH, XYZ, B)                    ! 1
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ! 2
!   GENERAL ELASTICITY STRAIN-DISPLACEMENT RELATIONS  (B)       ! 3
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ! 4
Use System_Constants ! for DP, N_R_B, N_G_DOF, N_SPACE,         ! 5
                     ! AXISYMMETRIC                             ! 6
Use Elem_Type_Data   ! for LT_FREE, LT_N, H (LT_N)              ! 7
 IMPLICIT NONE                                                  ! 8
 REAL(DP), INTENT(IN)  :: DGH (N_SPACE, LT_N) ! Gradients       ! 9
 REAL(DP), INTENT(IN)  :: XYZ (N_SPACE)       ! Coordinates     !10
 REAL(DP), INTENT(OUT) :: B   (N_R_B, LT_N * N_G_DOF)           !11
                                                                !12
! B       = STRAIN-DISPLACEMENT MATRIX (RETURNED)               !13
! DGH     = GLOBAL DERIVATIVES OF H                             !14
! H       = ELEMENT INTERPOLATION FUNCTIONS                     !15
! LT_N    = NUMBER OF NODES PER ELEMENT                         !16
! N_G_DOF = NUMBER OF PARAMETERS PER NODE                       !17
! N_R_B   = NUMBER OF STRAINS (ROWS IN B)                       !18
! N_SPACE = DIMENSION OF SPACE                                  !19
! XYZ     = CURRENT POINT COORDINATES, RADIUS = XYZ (1)         !20
                                                                !21
  SELECT CASE (N_SPACE)                                         !22
    CASE (1) ! 1-D or cylinder                                  !23
      IF ( AXISYMMETRIC ) THEN      ! Components RR & TT        !24
        CALL ELASTIC_B_CYLINDER (DGH, XYZ (1), B)               !25
      ELSE                          ! Component XX              !26
        CALL ELASTIC_B_BAR (DGH, B) ; END IF                    !27
    CASE (2) ! 2-D or axisymmetric                              !28
      IF ( AXISYMMETRIC ) THEN     ! Components RR ZZ RZ TT     !29
        CALL ELASTIC_B_AXISYMMETRIC (DGH, XYZ (1), B)           !30
      ELSE ! plane stress or strain. Components XX YY XY        !31
        CALL ELASTIC_B_PLANAR (DGH, B); END IF                  !32
    CASE (3) ! 3-D                                              !33
      CALL ELASTIC_B_SOLID (DGH, B)  ! XX YY XY ZZ ZX ZY        !34
  END SELECT                                                    !35
END SUBROUTINE ELASTICITY_B_MATRIX                              !36
```

Problem P11.6 A general elasticity strain-displacement matrix

Problems from Chapter 12

XXX

Problems from Chapter 13

XXX

Problems from Chapter 14

XXX

Problems from Chapter 15

XXX

