Matlab Code

- This code has two parts: (1) beam and (ii) pate.
- To run a beam problem copy the BEAM.m function and run in Matlab. The required functions to run BEAM.m are: aplybcdyn.m, FEAPLYC2.m, FEAPLYCS.m, FEASMBL1.m, FEASMBL2.m, feasmblf.m, FEBEM.m, FELDOF.m, FEELDOF.m, feload.m, felresp.m, yprime_beam.m, optiactuator.m.
- To run a palte problem copy the Plate_D_F.m function and run in Matlab. The required functions are: aplybcdyn.m, BOUNRY.m, ELKMFR_D_F.m, ELSTF.m, FEAPLYC2.m, FEAPLYCS.m, FEASMBL1.m, FEASMBL2.m, FELDOF.m, femesh.m, MSH2DR.m, Piezo_resultant.m, PLSTF.m, QBARSTF.m, QBARPIEZO.m, SHPRCT.m, STRESS.m, yprime_plate.m.

PART1 PLATE

Run plate_D_F.m, 2. Give an input file (example circle.inp) give an output file (example circle.out).

1. All the displacements and stress values will be stored in filename.out
2. Plots will be displayed
3. Read through the programs to change the working sequence
4. Read through the programs and input template to use the automated mesh code

Plate_D_F.m

```
%-----------------------------------------------------------------------
% PLATE FINITE ELEMENT PROGRAM
% Dynamic Analysis of composite plates
% CLPT, FSDT CAPABLITIES
% RECTANGULAR ISO PARAMETRIC ELEMENTS
%-----------------------------------------------------------------------
clear all
close all
FILE1 = input('Input Data File Name ','s');
LINP  = fopen(FILE1,'r');
FILE2 = input('Output Data File Name ','s');
stsrt_time=cputime;
LOUT  = fopen(FILE2,'w');
global LINP LOUT                                                % Files
global E1 E2 G12 G13 G23 ANU12 RHO ALFA1 ALFA2 THETA THKNS MATYP %
Material
global ELF ELK ELM ELXY ELU ELV ELA A1 A2 A3 A4 A5 Q0 QX QY    % Stiff1
global A B D E F G H A44 A45 A55 D44 D45 D55                    % Stiff2
global Q QBAR ALFA Z                                          % Elastic
global IPDF IPDR NIPF NIPR                                      % Point
global P0 P1 P2 P3 P4 P5 P6                                     % Integ
global T0 T1 TN TM TL TK                                        % Therm
global SF GDSF SFH GDSFH GDDSFH                                 % Shape
global PIEZO_E PM PN                                            % Piezo

%-----------------------------------------------------------------------
%       DATA INPUT
%-----------------------------------------------------------------------
```

```
INTIAL=0;
TITLE = fgets(LINP);                %READ TITLE
TMP = str2num(fgets(LINP));         %Read THEORY  and  ANALYSIS  type
[ITYPE,IDYN,NTHERM,NEIGN,PIEZO] =
deal(TMP(1),TMP(2),TMP(3),TMP(4),TMP(5));
if IDYN==0 NEIGN=0; end
TMP = str2num(fgets(LINP));         %READ LAYERS,NTHKNS,NUMAT
[LAYERS,NTHKNS,NUMAT] = deal(TMP(1),TMP(2),TMP(3));
TMP = str2num(fgets(LINP));         %READ THETA for all LAYERS
for I=1:LAYERS THETA(I)=TMP(I); end
TMP = str2num(fgets(LINP));         %READ THKNS for all LAYERS
for I=1:LAYERS THKNS(I)=TMP(I); end
TMP = str2num(fgets(LINP));         %READ MATerialTYPe for all LAYERS
for I=1:LAYERS MATYP(I)=TMP(I); end

if PIEZO>0
TMP = str2num(fgets(LINP));         %Read piezo values
[NPLY,p_tk,p_theta,AV] = deal(TMP(1),TMP(2),TMP(3),TMP(4));
end

%READ Material Properties for NUMAT
for N=1:NUMAT
    TMP = str2num(fgets(LINP));
    [E1(N),E2(N),G12(N),G13(N),G23(N),ANU12(N)] =
deal(TMP(1),TMP(2),TMP(3), TMP(4),TMP(5),TMP(6));
    if IDYN>0  TMP = str2num(fgets(LINP)); RHO(N)=deal(TMP(1)); end
    if NTHERM>0
        TMP=str2num(fgets(LINP));
        [ALFA1(N),ALFA2(N)]=deal(TMP(1),TMP(2));
    end
end
if PIEZO>0
      TMP = str2num(fgets(LINP));
      [EC13,EC23,EC33,EC41,EC42,EC51,EC52] = deal(TMP(1),TMP(2),TMP(3),
TMP(4),TMP(5),TMP(6),TMP(7));
end
                                    %READ AK
if ITYPE>1 AK=fscanf(LINP,'%e\n',1); else AK=0; end   %else condition
added
TMP = str2num(fgets(LINP));         %READ  XL,YL
[XL,YL] = deal(TMP(1),TMP(2));
TMP = str2num(fgets(LINP));         %READ DH for NTHKNS
for I=1:NTHKNS DH(I)=TMP(I); end
if NEIGN==0
TMP=str2num(fgets(LINP));
[LOAD,LPT]=deal(TMP(1),TMP(2));
TMP=str2num(fgets(LINP));           %READ Q0,QX,QY
[Q0,QX,QY]=deal(TMP(1),TMP(2),TMP(3));
end
if NTHERM>0
TMP=str2num(fgets(LINP));
[T0,T1]=deal(TMP(1),TMP2);         %READ T0,T1
end

% Read finite element mesh information:
TMP = fscanf(LINP,'%e\n',5);
```

```
[IELTYP,NPE,MESH,IGRAD,NPRNT] =
deal(TMP(1),TMP(2),TMP(3),TMP(4),TMP(5));
TMP = fscanf(LINP,'%e\n',3);
[IPDF,IPDR,ISTR] = deal(TMP(1),TMP(2),TMP(3));

  if MESH==0
      TMP=fscanf(LINP,'%d',2);    %READ NEM,NNM
      [NEM,NNM]=deal(TMP(1),TMP(2));
      for N=1:NEM                  %READ NOD for NPE
          TMP = fscanf(LINP,'%d',4);
          for I=1:NPE NOD(N,I)=TMP(I); end
      end
      for I=1:NNM
          TMP=fscanf(LINP,'%f',2);
          for J=1:2  GLXY(I,J)=TMP(J);   end
      end
  else
      TMP = fscanf(LINP,'%e\n',3);
      [NX,NY,EQSP] = deal(TMP(1),TMP(2),TMP(3));
      if EQSP==1  %EQSP=1 -- equaliy spaced mesh with NDIVX divisions
         TMP = fscanf(LINP,'%e\n',2);
         [X0,Y0]= deal(TMP(1),TMP(2));
         for I=1:NX DX(I)=XL/NX; end
         for I=1:NY DY(I)=YL/NY; end
      else %EQSP=0 -- unequaliy spaced mesh with DX()
          TMP = str2num(fgets(LINP));
          X0=TMP(1);
          for I=1:NX DX(I)=TMP(I+1); end
          TMP = str2num(fgets(LINP));
          Y0=TMP(1);
          for I=1:NY DY(I)=TMP(I+1); end
      end
  end
if NPE<=4
    IEL=1;
else
    IEL=2;
end

      if MESH>=1
          [NOD,GLXY,NNM,NEM]=MSH2DR(IEL,NX,NY,NPE,DX,DY,X0,Y0);
      end

      NDF=5;
      if ITYPE==1 NDF=6; end
      NEQ=NNM*NDF;
      NN=NPE*NDF;
%bc=0 -> provide boundary conditions
%bc=1 -> auto genrates boundary conditions- provide edge conditions
%bc=2 -> special boundary conditions
bc=fscanf(LINP,'%d',1);
%Generating Boundary Conditions
%Edge conditions:: SS=1; FIXED=2
if bc==1
    TMP=fscanf(LINP,'%d',4);
    [bcx0,bcxl,bcy0,bcyl] = deal(TMP(1),TMP(2),TMP(3),TMP(4));
```

```
    [NSPV,ISPV,VSPV]=bcgen(NOD,GLXY,NNM,NEM,bcx0,bcxl,bcy0,bcyl,XL,YL);
%     Eigen Boundary conditions
    if NEIGN==0         NSSV=fscanf(LINP,'%e',1);         end
    if NSSV==0 ISSV=0; VSSV=0; end % this line is added to declare
ISSV, VSSV
elseif bc==0
%     Read specified primary and secondary degrees of freedom: node
%     number, local degree of freedom number, and specified value.
    NSPV=fscanf(LINP,'%d',1);
      if NSPV~=0
          for I=1:NSPV
              TMP=fscanf(LINP,'%d',2);
              for J=1:2  ISPV(I,J)=TMP(J);   end
          end
          if NEIGN==0
              for I=1:NSPV
                  VSPV(I)=fscanf(LINP,'%e',1);
              end
          end
      end

      if NEIGN==0         NSSV=fscanf(LINP,'%e',1);         end
      if NSSV~=0
          for I=1:NSPV
              TMP=fscanf(LINP,'%d',2);
              for J=1:2  ISPV(I,J)=TMP(J);   end
          end
          for I=1:NSSV VSSV(I)=fscanf(LINP,'%e',1); end
      end
      if NSSV==0 ISSV=0; VSSV=0; end % this line is added to declare
ISSV, VSSV
end %BC input selection ends
for I=1:NSPV
    BCDOF(I)=ISPV(I,1)*5+ISPV(I,2);
end
   if PIEZO>0
           I=1:NEM;  PIEZO_E_in(I)=0; %assumes no element has piezo
initially
%            for I=1:NEM PIEZO_E_in(I)=fscanf(LINP,'%d',1); end
%           PIEZO_E(I)=fscanf(LINP,'%d',1);  %puts '1' for the elements
have piezo
%       else
% %           I=1:NEM;  PIEZO_E(I)=0;
            N_PIEZO_EL=fscanf(LINP,'%d',1);
            for I=1:N_PIEZO_EL  PIEZO_EL(I)=fscanf(LINP,'%d',1); end

            for J=1:N_PIEZO_EL
                for I=1:NEM
                    if I==PIEZO_EL(J);
                       PIEZO_E_in(I)=1;
                    end
                end
            end
        end
% %
% % Dynamic Input commented
```

```
% %       if IDYN~=0
% %           IDYN=2;
% %           if NEIGN==0
% %               TMP = str2num(fgets(LINP));
% %               [NTIME,NSTP,INTVL,INTIAL] =
deal(TMP(1),TMP(2),TMP(3),TMP(4));
% %               if INTVL<0 INTVL=1; end
% %               TMP = str2num(fgets(LINP));
% %               [DT,BETA,GAMA] = deal(TMP(1),TMP(2),TMP(3));
% %               if INTIAL~=0
% %                   TMP = str2num(fgets(LINP));
% %                   for I=1:NEQ GLU(I)=TMP(I); end
% %                   TMP = str2num(fgets(LINP));
% %                   for I=1:NEQ GLV(I)=TMP(I); end
% %               end
% %           else
% %               NVCTR=fscanf(LINP,'%e',1);
% %           end
% %       end

%-------------------------------------------------------------
% OUTPUT : Writes the input data in to the output file
%-------------------------------------------------------------------------
fprintf(LOUT,'\n \t \t Output for Input Data from file %s\n',FILE1);
fprintf(LOUT,'\n \t \t P. RAMESH \n\n');
if ITYPE<1
    fprintf(LOUT,'\t******* CLASSICAL  LAMINATE PLATE THEORY
*******\n');
    if ITYPE==0 fprintf(LOUT,'\t***** Non-conforming plate element is
used *****\n'); end
    if ITYPE==1 fprintf(LOUT,'\t******* Conforming plate element is
used *******\n'); end
else
    fprintf(LOUT, '\t***** FIRST-ORDER SHEAR DEFORMATION THEORY
*****\n');
end
if IDYN==0
    fprintf(LOUT,'\t****** A STEADY-STATE PROBLEM is analyzed
******\n');
else
    if NEIGN>0
        fprintf(LOUT,'\t****** An EIGENVALUE PROBLEM  is analyzed
******\n');
    else
        fprintf(LOUT, '\t******* A TRANSIENT PROBLEM  is analyzed
*******\n');
    end
end
fprintf(LOUT,'\n \n%s\n',TITLE);
fprintf(LOUT,'Number of layers in the laminate .......= %4d
\n',LAYERS);
fprintf(LOUT,'The layer orientations and thicknesses are:(identical
material layers) \n ');
for I=1:LAYERS     fprintf(LOUT,'%4d,%4.4e\n',THETA(I),THKNS(I)); end
for M=1:NUMAT
    fprintf(LOUT,'\n P R O P E R T I E S  of material no. = %d \n\n
',M);
```

```
    fprintf(LOUT,'Youngs modulus,   E1 ................... = %e
\n',E1(M));
    fprintf(LOUT,'Youngs modulus,   E2 ................... = %e
\n',E2(M));
    fprintf(LOUT,'Shear  modulus,  G12 ................... = %e
\n',G12(M));
    fprintf(LOUT,'Shear  modulus,  G13 ................... = %e
\n',G13(M));
    fprintf(LOUT,'Shear  modulus,  G23 ................... = %e
\n',G23(M));
    fprintf(LOUT,'Poissons ratio, NU12 ................... = %e
\n',ANU12(M));
end
if IDYN>0
    fprintf(LOUT,'Density of the material, RHO ........... = %e
\n',RHO(M));
end
if NTHERM>0
    fprintf(LOUT,'Coefficient of thermal expansion, ALFA1  = %e
\n',ALFA1(M));
    fprintf(LOUT,'Coefficient of thermal expansion, ALFA2  = %e
\n',ALFA2(M));
end
if NTHERM>0
    fprintf(LOUT,'Uniform temperature thru thickness,  T0  = %e
\n',T0);
    fprintf(LOUT,'Temperature gradient thru thickness, T1  = %e
\n',T1);
end
if NEIGN==0
    fprintf(LOUT, 'Intensity of transverse load, Q0 ....... = %e
\n',Q0);
    fprintf(LOUT,'Linear part of transverse load, QX ..... = %e
\n',QX);
    fprintf(LOUT,'Quadratic part of transverse load, QY .. = %e \n',
QY);
    if LOAD==2
        fprintf(LOUT,'Sinusoidal distribution of the load is used \n');
        fprintf(LOUT,'(1 = quarter plate model; 2 = full plate = %e
\n',LPT);
    end
end

% % Dynamic load
% if IDYN~=0
%     if NEIGN==0
%         if INTVL<=0 INTVL=1; end
%         fprintf(LOUT,'Time increment used, DT ................ = %e
\n', DT);
%         fprintf(LOUT,'Parameter BETA in the Newmark scheme ... = %e
\n', BETA);
%         fprintf(LOUT,'Parameter GAMA in the Newmark scheme ... = %e
\n', GAMA);
%         fprintf(LOUT,'Number of time steps used, NTIME ....... = %e
\n', NTIME);
%         fprintf(LOUT,'Time step at which load is removed, NSTP = %e
\n', NSTP);
```

```
%         fprintf(LOUT,'Time interval at which soln. is printed  = %e
\n', INTVL);
%         DT2=DT*DT;
%         A1=(1.0-BETA)*DT;
%         A2=BETA*DT;
%         A3=2.0/GAMA/DT2;
%         A4=A3*DT;
%         A5=1.0/GAMA-1.0;
%         if INTIAL>0
%             for I=1:NEQ GLA(I)=0.0; end
%         else
%             for I=1:NEQ
%                 GLU(I)=0.0;
%                 GLV(I)=0.0;
%                 GLA(I)=0.0;
%             end
%         end
%     end
% end

if ITYPE<=1
    NSHR=0;
else
    NSHR=1;
end
if IELTYP==0
    fprintf(LOUT,   '\n \t *** A mesh of    TRIANGLES   is chosen by
user *** \n');
else
    fprintf(LOUT,  '\n \t*** A mesh of QUADRILATERALS is chosen by user
*** \n');
end
%     Compute the half bandwidth of the global coefficient matrix
if NEIGN==0
    NHBW=0;
    for N=1:NEM
        for I=1:NPE
            for J=1:NPE
                NW=(abs(NOD(N,I)-NOD(N,J))+1)*NDF;
                if NHBW<NW NHBW=NW; end
            end
        end
    end
else
    NHBW=NEQ;
end

fprintf(LOUT,'\n \t FINITE ELEMENT MESH INFORMATION:\n \n');
fprintf(LOUT,'Element type: 0 = Triangle; >0 = Quad.)  = %d
\n',IELTYP);
fprintf(LOUT,'Number of nodes per element, NPE ..... = %d \n',NPE);
fprintf(LOUT,'No. of primary deg. of freedom/node, NDF = %d \n',NDF);
fprintf(LOUT,'No. of deg. of freedom per element, NN   = %d \n',NN);
fprintf(LOUT,'Number of elements in the mesh, NEM .... = %d \n',NEM);
fprintf(LOUT,'Number of nodes in the mesh, NNM ..... = %d \n',NNM);
fprintf(LOUT,'Number of equations to be solved, NEQ .. = %d \n',NEQ);
```

```
fprintf(LOUT,'Half bandwidth of the matrix GLK, NHBW   = %d \n',NHBW);
if MESH==1
    fprintf(LOUT,'Mesh subdivisions, NX and NY ..... =%d \t %d
\n',NX,NY);
end
fprintf(LOUT,'No. of specified generalized displ., NSPV= %d \n', NSPV);
if NSSV~=0
    fprintf(LOUT, 'No. of specified generalized forces, NSSV= %d
\n',NSSV);
    fprintf(LOUT,  'Node  DOF     Value\n');
    for IB=1:NSSV
        for JB=1:2
            fprintf(LOUT,'%d',ISSV(IB,JB));
        end
        fprintf(LOUT,'%d',VSSV(IB));
    end
end
fprintf(LOUT,'_____________________________________________________________
_____________________________\n');
fprintf(LOUT,'\t Node   x-coord.    y-coord. \t\t Specified
displacements and forces \n \t\t\t\t(0, unspecified; >0, specified)
Displ. DOF   Force DOF\n');
fprintf(LOUT,'_____________________________________________________________
_____________________________\n');
for IM=1:NNM       %140
    for K=1:NDF
        IBP(K)=0;
        IBS(K)=0;
    end
    if NSPV~=0
        for JP=1:NSPV
            NODE=ISPV(JP,1);
            NDOF=ISPV(JP,2);
            if NODE==IM
                IBP(NDOF)=NDOF;
            end
        end
    end
    if NSSV~=0
        for JS=1:NSSV
            NODE=ISSV(JS,1);
            NDOF=ISSV(JS,2);
            if NODE.EQ.IM
                IBS(NDOF)=NDOF;
            end
        end
    end
    if NDF==5
        fprintf(LOUT,'\n \t %d ',IM);
        for J=1:2 fprintf(LOUT,'\t %e',GLXY(IM,J)); end
        for K=1:NDF fprintf(LOUT,'\t %d',IBP(K)); end
        for K=1:NDF fprintf(LOUT,'\t %d',IBS(K)); end
    else
        fprintf(LOUT,'\n \t %d ',IM);
        for J=1:2 fprintf(LOUT,'\t %e',GLXY(IM,J)); end
        for K=1:NDF fprintf(LOUT,'\t %d',IBP(K)); end
        for K=1:NDF fprintf(LOUT,'\t %d',IBS(K)); end
```

```
    end
end
fprintf(LOUT,'\n_____________________________________________________________
_____________________________________\n');
%     Define the polynomial degree and number of integration points
%     (based on the CONSTANT laminate stiffnesses, [A],[B],[D], etc.)
fprintf(LOUT,'\n \n \t NUMERICAL INTEGRATION DATA:\n\n');
fprintf(LOUT,'Full quadrature (IPDF x IPDF) rule, IPDF = %d \n',IPDF);
fprintf(LOUT,'Reduced quadrature (IPDR x IPDR),   IPDR = %d \n',IPDR);
fprintf(LOUT,'Quadrature rule used in postproc.,  ISTR = %d \n',ISTR);
if NPRNT==1
    fprintf(LOUT,'\nConnectivity Matrix, [NOD] \n');
    for  I=1:NEM
        fprintf(LOUT,'\n \t %d ',I);
        for J=1:NPE fprintf(LOUT,'\t %d' ,NOD(I,J)); end
    end
end



% %Deformation Ploting -3D
% plotting
figure
for i=1:NEM
    for j=1:NPE
        xel(j)=GLXY(NOD(i,j),1);
        yel(j)=GLXY(NOD(i,j),2);
        zel=zeros(size(xel));
    end
%     plot3(xel,yel,zel);
    fill3(xel,yel,zel,'m');
    hold on
end

%-----------------------------------------------------------------------
%     *                   P R O C E S S O R   U N I T                 *
%-----------------------------------------------------------------------
% optloc=1;
% for P=1:NEM+1 %forms Bact
%     I=1:NEM;  PIEZO_E(I)=0;
%     PIEZO_E(P)=1;  %sets Piezo=1 for element-1 or location-1
% %     if P>NEM PIEZO_E(optloc)=1; end
% if P>NEM PIEZO_E=PIEZO_E_in; end
PIEZO_E=PIEZO_E_in;
for NTK=1:NTHKNS   %350
    HSCALE=DH(NTK);
    HT=0.0;
    for I=1:LAYERS
        HT=HT+THKNS(I)*HSCALE;
    end
    HT2=HT*HT;
    HT3=HT2*HT;
    YL2=YL*YL;
    YL3=YL*YL2;
    YL4=YL2*YL2;
%     SCALEU=100.0*E2(1)*HT2/Q0/YL3;
%     SCALEW=100.0*E2(1)*HT3/Q0/YL4;
```

```
    fprintf(LOUT,'\n\nTotal thickness of the laminate, HT = %e \n',HT);

    % Calculate laminate stifnesses A,B,D and write them in to output
file
    PLTSTF(LAYERS,NSHR,IDYN,NTHERM,NPRNT,HSCALE);
    if PIEZO>0
        [SP]=piezo_resultant(p_tk, p_theta, EC13, EC23, EC33, EC41,
EC42, EC51, EC52, AV, NPLY,ITYPE);
    end

%     if IDYN>0 & NEIGN==0  TIME=0.0;    end
    NT = 0;
     NCOUNT=0;
    %loop=1;
    % while loop~=0
%     NCOUNT=NCOUNT+1; %160
%     if IDYN~=0 & NEIGN==0
%         if NCOUNT>=NSTP
%             Q0=0.0;
%             QX=0.0;
%             QY=0.0;
%         end
%     end
%     %     Initialize the global coefficient matrices and vectors
GLF=zeros(1,NEQ);
%     GLK=zeros(NEQ,NHBW);
%     if NEIGN>0 GLM=zeros(NEQ,NHBW); end

    %     Initialize the global coefficient matrices and vectors
    GLFN=zeros(1,NEQ);
    GLKN=zeros(NEQ,NEQ);
    GLMN=zeros(NEQ,NEQ);

    %Do-loop on the number of ELEMENTS to compute element matrices
    %and their assembly begins here
%     disp(sprintf('E n t e r i n g..element loop'));
    for N=1:NEM  % 250
        for I=1:NPE
            NI=NOD(N,I);
            ELXY(I,1)=GLXY(NI,1);
            ELXY(I,2)=GLXY(NI,2);
%             if NEIGN==0
%                 if IDYN>0
%                     LI=(NI-1)*NDF;
%                     L = (I-1)*NDF;
%                     for J=1:NDF
%                         LI=LI+1;
%                         L=L+1;
%                         % Initializing the displacement, velocity,
%                         % acceleration for dynamic analysis
%                         ELU(L)=GLU(LI);
%                         ELV(L)=GLV(LI);
%                         ELA(L)=GLA(LI);
%                     end
%                 end
%             end
```

```
        end


ELKMFR_D_F(AK,XL,YL,LPT,NEIGN,NPE,NDF,ITYPE,IDYN,LOAD,NTHERM,INTIAL,NCO
UNT,N);

%         %Print element matrices and vectors (only when NPRNT=1 or
NPRNT=3)
%         if NCOUNT==1
%             if NPRNT==1 | NPRNT==3
%                 if N==1 %for element no 1 only
%                     fprintf(LOUT, '\n\n \t Element stiffness matrix:
%d Element \n',N);
%                     for I=1:NN
%                         fprintf(LOUT,'\n');
%                         for J=1:NN
%                             fprintf(LOUT,'\t %e',ELK(I,J));
%                         end
%                     end
%                     if NEIGN==0
%                         fprintf(LOUT,'\n\n \t Element force
vector:\n')
%                         fprintf(LOUT,'\n');
%                         for J=1:NN
%                             fprintf(LOUT,'\t %e',ELF(J));
%                         end
%                     else
%                         fprintf(LOUT, '\n\n Element mass matrix:
\n');
%                         for I=1:NN
%                             fprintf(LOUT,'\n');
%                             for J=1:NN
%                                 fprintf(LOUT,'\t %e',ELM(I,J));
%                             end
%                         end
%                     end
%                 end
%          end

         [index]=feeldof(NOD,NPE,NDF,N);
         [GLKN,GLFN]=feasmbl2(GLKN,GLFN,index);
         [GLMN]=feasmbl1(GLMN,index);

          for I=1:NPE
%             NR=(NOD(N,I)-1)*NDF;
%             for II=1:NDF
%                 NR=NR+1;
%                 L=(I-1)*NDF+II;
%                 if NEIGN==0
%                     GLF(NR)=GLF(NR)+ELF(L);
%                 end
%                 for J=1:NPE
%                     if NEIGN==0
%                         NCL=(NOD(N,J)-1)*NDF;
%                     else
```

```
%                         NC=(NOD(N,J)-1)*NDF;
%                     end
%                     for JJ=1:NDF
%                         M=(J-1)*NDF+JJ;
%                         if NEIGN==0
%                             NC=NCL+JJ+1-NR;
%                             if NC>0
%                                 GLK(NR,NC)=GLK(NR,NC)+ELK(L,M);
%                             end
%                         else
%                             NC=NC+1;
%                             GLK(NR,NC)=GLK(NR,NC)+ELK(L,M);
%                             GLM(NR,NC)=GLM(NR,NC)+ELM(L,M);
%                         end
%                     end %240
%                 end %240
%             end%240
%         end%240

    end % 250 (ELEMENT LOOP ends)

%     %Print global matrices when NPRNT > 2
%     if NCOUNT<=1
%         if NPRNT>=2
%             fprintf(LOUT, '\n Global stiffness matrix (upper band):
\n');
%             for I=1:NEQ
%                 fprintf(LOUT,'\n');
%                 for J=1:NHBW
%                     fprintf(LOUT,'\t %e', GLK(I,J));
%                 end
%             end
%             if NEIGN==0
%                 fprintf(LOUT, '\n Global force vector:\n');
%                 for I=1:NHBW
%                     fprintf(LOUT,'\t %e', GLF(I));
%                 end
%             else
%                 fprintf(LOUT, '\n  Global mass matrix (full form):\n
');
%                 for I=1:NEQ
%                     for J=1:NHBW
%                         fprintf(LOUT,'\t %e', GLM(I,J));
%                     end
%                 end
%             end
%         end
%     end
   % if NEIGN>0   commneted for static analysis..
    EGNBOU(GLK,GLM,IBDY,ISPV,MAXSPV,NDF,NEQ,NEQR,NSPV,NRMAX)
    % AXLBX;  %  CALL AXLBX
(NEQR,GLK,GLM,EGNVAL,EGNVEC,JVEC,NROT,NRMAX)
    % fprintf(LOUT,  '\n S O L U T I O N :\n ');
    % fprintf(LOUT,'\n Number of Jacobi iterations ..... NROT =
%d\n',NROT);
    % CONST=(YL*YL/HT)*DSQRT(RHO(1)/E2(1));
```

```
    % for  I=1:NEQR
    % if IDYN>0 & NEIGN==1
    % VALUE = DSQRT(EGNVAL(I))*CONST;
    %    fprintf(LOUT, '\n Eigenvalue(%d) =%e \t\t Frequency =
%e\n',I,EGNVAL(I),VALUE);
    %      else
    %               fprintf(LOUT, 'Buckling load(%d)
=%e\n',I,EGNVAL(I));
    %           end
    %           if NVCTR~=0
    %            fprintf(LOUT, '\n Eigenvector:\n');
    %            for J=1:NEQR fprintf(LOUT,'\t\t %e\n', EGNVEC(J,I));
end
%          end
    %       end
    %      STOP


    %else %NEIGN CONDITION :(
%     [GLKE,GLME]=feaplycs(GLKN,GLMN,ISPV,NSPV,NEQ);
%     fsol=eig(GLKE,GLME);
%     fsol=sqrt(fsol);
%     num=1:1:NEQ;
%     frequency=[num' fsol]

    % Static Analysis
    GLK1=GLKN; %before applying BC
    GLM=GLMN; %before applying BC
 [GLKN,GLFN]=feaplyc2(GLKN,GLFN,ISPV,ISSV,NDF,NEQ,NSPV,NSSV,VSPV,VSSV);
    [GLKE,GLME]=feaplycs(GLK1,GLMN,ISPV,NSPV,NEQ);
    IRES=0;
    FORCE=GLFN;
    SP_GLKN=sparse(GLKN);
    GLKN_UP=chol(SP_GLKN);
    GLFN=GLKN_UP\(GLKN_UP'\GLFN');


        %   Dynamic Analysis
    %   if IDYN>0
    %
    %     For nonzero initial conditions, GLF in the very first
solution
    %     is the acceleration, {A}=[MINV]({F}-[K]{U})
    %
    %            if NCOUNT==1 & INTIAL~=0
    %               if IDYN>0
    %                  for I=1:NEQ
    %                  GLA(I)=GLF(I);
    %                  end
    %                  fprintf(LOUT, '\n *TIME* = %e (Initial
acceleration vector:)\n');
    %                  for I=1:NEQ fprintf(LOUT, '\t %e',GLA(I)); end
    %
    %
    %
```

```
    %               else
    %                   NT = NT + 1;
    %                  TIME=TIME+DT;
    %              end
    %          else
    %                NT = NT + 1;
    %                TIME=TIME+DT;
    %            end
        %            for I=1:NEQ
    %              GLU(I)=A3*(GLF(I)-GLU(I))-A4*GLV(I)-A5*GLA(I);
    %              GLV(I)=GLV(I)+A1*GLA(I)+A2*GLU(I);
    %              GLA(I)=GLU(I);
    %             GLU(I)=GLF(I);
    %         end
    %          INTGR=(NT/INTVL)*INTVL;
    %           if INTGR==NT
    %                NFLAG=1;
    %            else
    %                NFLAG=0;
    %            end
    %   else%IDYN>O condition
%      for I=1:NEQ
%          GLU(I)=GLF(I);
%      end
    %end %IDYN>O condition
    %     Print the solution (i.e., nodal values of the primary
variables)
    %       if IDYN>0
    %            fprintf(LOUT,'\n*TIME* = %e\t  Time Step Number
=%d',TIME,NT);
%        end
%     disp(sprintf('E n t e r i n g... soln..'));

%     if P==NEM
%         NFLAG=1;
%     else
%         NFLAG=0;
%     end

NFLAG=1;

    if NFLAG>0
        fprintf(LOUT,  '\n\n\n\t\tS O L U T I O N :\n');
        %             displacements
%
fprintf(LOUT,'\n_____________________________________________________________
_____________________________\n');
%         if NDF<=5 fprintf(LOUT, 'Node   x-coord.    y-coord.
displ. u displ. v    deflec. w   rot. dw/dx  rot. dw/dy\n'); end
%         if NDF>5 fprintf(LOUT,  'Node   x-coord.    y-coord.
displ. u displ. v    deflec. w   rot. phi-x  rot. phi-y\n'); end
%
fprintf(LOUT,'\n_____________________________________________________________
_____________________________\n');
%         NDF1=5;
%         for I=1:NNM   % 320
```

```
%              II=NDF*(I-1)+1;
%              JJ=II+NDF1-1;
%              GLF(II)=GLF(II)*SCALEU;
%              GLF(II+1)=GLF(II+1)*SCALEU;
%              GLF(II+2)=GLF(II+2)*SCALEW;
%              fprintf(LOUT,'\n %d \t',I);
%              for J=1:2 fprintf(LOUT,'\t %e\t',GLXY(I,J)); end
%              for J=II:JJ  fprintf(LOUT,'%e\t',GLF(J));  end
%          end % 320 ends...
%
fprintf(LOUT,'\n_____________________________________________________________
_____________________________________\n');
%


         fprintf(LOUT,  '\n\n\n\t\t NEW S O L U T I O N :\n');
         %             displacements

fprintf(LOUT,'\n_____________________________________________________________
_____________________________________\n');
         if NDF<=5 fprintf(LOUT, 'Node   x-coord.    y-coord.    displ.
u displ. v    deflec. w   rot. dw/dx  rot. dw/dy\n'); end
         if NDF>5 fprintf(LOUT,  'Node   x-coord.    y-coord.    displ.
u displ. v    deflec. w   rot. phi-x  rot. phi-y\n'); end

fprintf(LOUT,'\n_____________________________________________________________
_____________________________________\n');
         NDF1=5;
         for I=1:NNM   % 320
             II=NDF*(I-1)+1;
             JJ=II+NDF1-1;
%              GLFN(II)=GLFN(II)*SCALEU;
%              GLFN(II+1)=GLFN(II+1)*SCALEU;
%              GLFN(II+2)=GLFN(II+2)*SCALEW;
             GLFN(II)=GLFN(II);
             GLFN(II+1)=GLFN(II+1);
             GLFN(II+2)=GLFN(II+2);
             fprintf(LOUT,'\n %d \t',I);
             for J=1:2 fprintf(LOUT,'\t %e\t',GLXY(I,J)); end
             for J=II:JJ  fprintf(LOUT,'%e\t',GLFN(J));  end
         end % 320 ends...

fprintf(LOUT,'\n_____________________________________________________________
_____________________________________\n');
         %Newly added ends


         if NDF>NDF1
             fprintf(LOUT, '\n Nodal values of W,xy for conforming plate
element:\n ');
         end
%          if IGRAD~=0
%              % *  P O S T P R O C E S S O R   U N I T            *
%              fprintf(LOUT, '\n\n STRESSES AT top (T) and bottom (B) of
each layer are printed below: \n');
```

```
%
fprintf(LOUT,'\n\t\t_____________________________________________________________
_____________________________________\n');
%             fprintf(LOUT,  '\n\t Ply  sgmxx-T  sgmxx-B sggmyy-T
sgmyy-B sgmxy-T sgmxy-B  sgmxz-T  sgmyz-B\n');
%
fprintf(LOUT,'\n\t\t_____________________________________________________________
_____________________________________\n');
%             %     Compute the stresses
%             ICOUNT=1;
%             for N=1:NEM
%                 for I=1:NPE
%                     NI=NOD(N,I);
%                     ELXY(I,1)=GLXY(NI,1);
%                     ELXY(I,2)=GLXY(NI,2);
%                     LI=(NI-1)*NDF;
%                     L=(I-1)*NDF;
%                     for J=1:NDF
%                         LI=LI+1;
%                         L=L+1;
%                         ELU(L)=GLU(LI);
%                     end
%                 end
%                 disp(sprintf('\t\tcalling  STRESS for elemnt
%d..',N));
%                 STRESS; %   CALL
STRESS(ELU,ELXY,HT,Q0,YL,ISTR,ITYPE,LAYERS,NDF,NPE,NTHERM,ICOUNT,MAXELM
,MAXGPT,RNXX,RNYY,RNXY,RMXX,RMYY,RMXY,RQX,RQY)
%                 ICOUNT=ICOUNT+1 ;
%             end  %element loop for STRESS ends...
%             fprintf(LOUT,'\n_____________________________________________ :)
THE END :) _____________________________________________________\n');
%         end % IGRAD condition...
%
        %check for IDYN>0 is modified to avoid GOTO and is modified
with WHILE
        % new control variable "loop"has been used..
        %   if IDYN>0
        %       if NT>=NTIME
        %          loop=0;
        %      else
        %          loop=1;
        %      end
        %  end
        %      end
        %   end

        %  end%while loop ends here.....
    end % NFLAG condition
end % NTK loop ends...

% % % Forming Bact
% if P<=NEM
% N=NEQ;
% F=FORCE';
% [V,D]=eig(K,M);
```

```
% FF=V'*F;
%
% F_vec = [zeros(size(FF))
%          inv(M)*FF];
% Bact(:,P)=[F_vec];
% end

% % % First order System
% if P==NEM
% K=GLKN;
% M=GLME;
% [V,D]=eig(K,M);
% KK=V'*K*V;
% MM=V'*M*V;
%
% C=0.2*MM+0.005*KK;  % Alpha=0.2, Beta=0.005
% A_mat = [zeros(size(KK)) eye(size(KK))
%          -inv(MM)*KK     -inv(MM)*C];
% C_mat=[eye(size(A_mat)/2), zeros(size(A_mat)/2)];
% end

% Optimal location calculation
%  if P>NEM
%  [optloc]=optactuator(A_mat,Bact,C,10,0.001,100,1,0);
%optactuator(A_mat,Bact,C,optact0,tol,itmax,type,cov);
% % % %Hand calculation
% % Q=C_mat'*C_mat;
% % R=1;
% % for j=1:NEM
% %     B_mat=Bact(:,j);
% %     [f,K]=lqr(A_mat,B_mat,Q,R);
% % [Vs,es]=eig(K);
% % sigmahand(j)=.5*max(diag(es));
% % end;
% % opthand=min(sigmahand);
% % optacthand=find(sigmahand<opthand+.00001)
%  end

% disp(sprintf(' loaction %d',P))

% end % Bact loop ends
disp(sprintf('Static analysis compleeted...'));
fclose('all');
%%%%%%%%%%%%%%%%  end of main program %%%%%%%%%%%%%%%%%%%%%%%%%%




% %--------------------------------------
% % Dynamic Analysis
% %--------------------------------------
% t0 = 0;
% t1 = 2.5;
% nsteps1 = round((t1-t0)*100);
% hsteps1 = (t1-t0)/nsteps1;
```

```
% N=NEQ;
% K=GLKN;
% M=GLME;
% F=FORCE';
% %--------------------------------------
% %Eigen value analysis
% % %--------------------------------------
% % freq=eig(K,M);
% % freq=sqrt(freq)
% % %--------------------------------------
%
% C=0.2*M+0.005*K;  % Alpha=0.2, Beta=0.005
% A_mat = [zeros(size(K)) eye(size(K))
%          -inv(M)*K     -inv(M)*C];
% % A_mat = [zeros(size(K)) eye(size(K))
% %           -inv(M)*K     eye(size(K))];
% F_vec = [zeros(size(F))
%          inv(M)*F];
% %
% %  Integrate system using the stiff ODE solver ode15s.m.
% %
% y0 = zeros(2*N,1);
% tspan = t0:hsteps1:t1;
% options =
odeset('JConstant','on','Jacobian','on','BDF','on','RelTol',1e-4,
'AbsTol',1e-4);
% [t,y] = ode23s('yprime_plate',tspan,y0,options,A_mat,F_vec);
% yt1 = y(:,1:N);
% % ysolution = Bvn*yt1';
% ysolution = yt1';
%
% figure(1)
% plot(t,ysolution(((NNM-1)/2)*5+3,:))
% xlabel('Time (s)')
% ylabel('Displacement ( )')
% %

%Deformation Ploting -3D
J=1;
for I=3:5:NNM*5
    Z(J)=GLFN(I);
    J=J+1;
end
% [XP,YP]=meshgrid(GLXY(:,1),GLXY(:,2));
% ZP=griddata(GLXY(:,1),GLXY(:,2),Z,XP,YP);
% figure
% surf(XP,YP,ZP);

% % plotting
figure
for i=1:NEM
    for j=1:NPE
        xel(j)=GLXY(NOD(i,j),1);
        yel(j)=GLXY(NOD(i,j),2);
        zel(j)=Z(NOD(i,j));
    end
```

```
%     plot3(xel,yel,zel);
    fill3(xel,yel,zel,'g');
    hold on
end

elapsed_time=cputime-stsrt_time

% %Linear System simulation
% t = 0:1/200:10;
% sys = ss(A_mat, Bact(:,20), C_mat,0);
% [y,t] = lsim(sys,60*sin(t),t);
% figure(2)
% plot(t,y(:,end-1),'r-.');
% xlabel('Time [s]');
% ylabel('Amplitude [m]');

% %Step response
% figure(3)
% y=step(A_mat,Bact(:,20),C_mat,0,1,t);
% plot(t,y(:,1))
```

aplybcdyn.m

```
function [kkk,mmm,fff]=aplybcdyn(bcdof,kk,mm,ff)
%Apply BC for dynamic analysis - general way
kkk=kk;
mmm=mm;
fff=ff;
% Row
for i = 1:length(bcdof)
    if bcdof(i)-i+1 == 1
        tmpk = [kkk(bcdof(i)-i+2:end,:)];
        tmpm = [mmm(bcdof(i)-i+2:end,:)];
    elseif bcdof(i)-i+1 == size(kkk,1)
        tmpk = [kkk(1:bcdof(i)-i,:)];
        tmpm = [mmm(1:bcdof(i)-i,:)];
    else
        tmpk = [kkk(1:bcdof(i)-i,:); kkk(bcdof(i)-i+2:end,:)];
        tmpm = [mmm(1:bcdof(i)-i,:); mmm(bcdof(i)-i+2:end,:)];
    end
    kkk = tmpk;
    mmm = tmpm;
end
% Col
for i = 1:length(bcdof)
    if bcdof(i)-i+1 == 1
      tmpk = [kkk(:,bcdof(i)-i+2:end)];
      tmpm = [mmm(:,bcdof(i)-i+2:end)];
    elseif bcdof(i)-i+1 == size(kkk,2)
       tmpk = [kkk(:,1:bcdof(i)-i)];
       tmpm = [mmm(:,1:bcdof(i)-i)];
    else
       tmpk = [kkk(:,1:bcdof(i)-i) kkk(:,bcdof(i)-i+2:end)];
       tmpm = [mmm(:,1:bcdof(i)-i) mmm(:,bcdof(i)-i+2:end)];
    end
    kkk = tmpk;
```

```
    mmm = tmpm;
end
% F
for i = 1:length(bcdof)
    if bcdof(i)-i+1 == 1
        tmpf = [fff(bcdof(i)-i+2:end)];
    elseif bcdof(i)-i+1 == length(fff)
        tmpf = [fff(1:bcdof(i)-i)];
    else
        tmpf = [fff(1:bcdof(i)-i); fff(bcdof(i)-i+2:end)];
    end
    fff = tmpf;
end
fff=fff';
```

bcgen.m

```
function
[NSPV,ISPV,VSPV]=bcgen(NOD,GLXY,NNM,NEM,bcx0,bcxl,bcy0,bcyl,XL,YL)
nx0=0; ny0=0; nxl=0; nyl=0;
% BGLXY=GLXY*1000; BXL=XL*1000; BYL=YL*1000; % Dummy multipliers
err=10e-12;
for i=1:NNM
    if GLXY(i,1)<=err
        nx0=nx0+1;
        edgx0(nx0)=i;
    end
    if GLXY(i,2)<=err
        ny0=ny0+1;
        edgy0(ny0)=i;
    end
    if (XL-GLXY(i,1))<=err
        nxl=nxl+1;
        edgxl(nxl)=i;
    end
    if (YL-GLXY(i,2))<=err
        nyl=nyl+1;
        edgyl(nyl)=i;
    end
end
iNSPV=0;
for i=1:NNM
    inx0=(edgx0==i); inxl=(edgxl==i);
    iny0=(edgy0==i); inyl=(edgyl==i);
    %if the node is at the corner
    if (sum(inx0)>0 & sum(iny0)>0) | (sum(inx0)>0 & sum(inyl)>0) |
(sum(inxl)>0 & sum(iny0)>0) | (sum(inxl)>0 & sum(inyl)>0)
        ISPV(iNSPV+1,1)=i; ISPV(iNSPV+2,1)=i; ISPV(iNSPV+3,1)=i;
ISPV(iNSPV+4,1)=i; ISPV(iNSPV+5,1)=i;
        if bcx0==1 %SS
            ISPV(iNSPV+1,2)=1; ISPV(iNSPV+2,2)=2; ISPV(iNSPV+3,2)=3;
ISPV(iNSPV+4,2)=4; ISPV(iNSPV+5,2)=5;
        end
         iNSPV=iNSPV+5;
    elseif sum(inx0)>0 |sum(inxl)>0 %if the node is @x=0 OR x=L edges
        ISPV(iNSPV+1,1)=i; ISPV(iNSPV+2,1)=i; ISPV(iNSPV+3,1)=i;
        if bcx0==1 %SS
            ISPV(iNSPV+1,2)=2; ISPV(iNSPV+2,2)=3; ISPV(iNSPV+3,2)=5;
```

```
        end
        iNSPV=iNSPV+3;
    elseif sum(iny0)>0 |sum(inyl)>0 %if the node is @y=0 OR y=L edges
        ISPV(iNSPV+1,1)=i; ISPV(iNSPV+2,1)=i; ISPV(iNSPV+3,1)=i;
        if bcy0==1 %SS
            ISPV(iNSPV+1,2)=1; ISPV(iNSPV+2,2)=3; ISPV(iNSPV+3,2)=4;
        end
        iNSPV=iNSPV+3;
    end
end
NSPV=length(ISPV);
VSPV=zeros(NSPV,1); %zero bc values
```

BOUNRY.m

```
 function
[GLK,GLF]=BOUNRY(ISPV,ISSV,NDF,NEQ,NHBW,NSPV,NSSV,GLK,GLF,VSPV,VSSV,NCO
UNT,INTIAL)
%_____________________________________________________________________
%     The subroutine implements  specified  values of the primary and
%     secondary variables by modifying the coefficient matrix [S] and
%     (banded and symmetric) and the right-hand side vector {SL}.
%_____________________________________________________________________

S=GLK;
SL=GLF;
      if NSSV~=0
         if INTIAL==0 | NCOUNT~=1
%
%     Implement specified values of the SECONDARY ARIABLES:_____________
%
            for I=1:NSSV
            II=(ISSV(I,1)-1)*NDF+ISSV(I,2);
            SL(II)=SL(II)+VSSV(I);
        end
    end
end
%
%     Implement specified values of the PRIMARY VARIABLES
      if NSPV~=0
         for  NB=1:NSPV   % loop 50 ..
         IE=(ISPV(NB,1)-1)*NDF+ISPV(NB,2);
         VALUE=VSPV(NB);
         IT=NHBW-1;
         I=IE-NHBW;
        for II=1:IT
         I=I+1;
         if I>1
            J=IE-I+1;
            SL(I)=SL(I)-S(I,J)*VALUE;
            S(I,J)=0.0;
        end
    end
         S(IE,1)=1.0;
         SL(IE)=VALUE;
         I=IE;
         for  II=2:NHBW  % loop 40..
```

```
          I=I+1;
           if I<=NEQ
             SL(I)=SL(I)-S(IE,II)*VALUE;
             S(IE,II)=0.0;
          end
    end
end
end

GLK=S;
GLF=SL;
```

ELKMFR_D_F.m

```
function
[]=ELKMFR_D_F(AK,XL,YL,LPT,NEIGN,NPE,NDF,ITYPE,IDYN,LOAD,NTHERM,INTIAL,
NCOUNT,N)
global ELF ELK ELM ELXY ELU ELV ELA A1 A2 A3 A4 A5 Q0 QX QY      %
Stiff1
global IPDF IPDR NIPF NIPR                                       %
Point
global P0 P1 P2 P3 P4 P5 P6                                      %Integ
global T0 T1 TN TM TL TK                                         %Therm
global SF GDSF SFH GDSFH GDDSFH                                  %Shape
global A B D E F G H A44 A45 A55 D44 D45 D55                      %
Stiff2
global PIEZO_E PM PN

    GAUSPT=[0.0 0.0 0.0 0.0 0.0;
            -0.57735027 0.57735027 0.0 0.0 0.0;
            -0.77459667 0.0 0.77459667 0.0 0.0;
            -0.86113631 -0.33998104 0.33998104 0.86113631 0.0;
            -0.90617984 -0.53846931 0.0 0.53846931 0.9061798];
     GAUSPT=GAUSPT';
%


     GAUSWT=[2.0 0.0 0.0 0.0 0.0;
            1.0 1.0 0.0 0.0 0.0;
            0.55555555 0.88888888 0.55555555 0.0 0.0;
            0.34785485 0.65214515 0.65214515 0.34785485 0.0;
            0.23692688 0.47862867 0.56888888 0.47862867 0.23692688];
     GAUSWT=GAUSWT';

%
NN=NDF*NPE;
if ITYPE==0
    NET = 3;
else
    NET = 4;
end

    ELK=zeros(45);
    ELF=zeros(45,1);
    ELM=zeros(45);
%     Do-loops on numerical (Gauss) integration begin here. Subroutine
%     SHPRCT (SHaPe functions for ReCTangular elements) is called here
```

```
      for  NI = 1:IPDF
      for  NJ = 1:IPDF
      XI  = GAUSPT(NI,IPDF);
      ETA = GAUSPT(NJ,IPDF);
%
    [DET]=SHPRCT(NPE,XI,ETA,ELXY,ITYPE);
%
      CNST = DET*GAUSWT(NI,IPDF)*GAUSWT(NJ,IPDF);
      X=0.0;
      Y=0.0;
      for I=1:NPE
      X=X+ELXY(I,1)*SF(I);
      Y=Y+ELXY(I,2)*SF(I);
      end
%
      if NEIGN==0
         if LOAD==2
            if LPT==1
               QL=Q0*cos(pi*X/XL)*cos(pi*Y/YL);
           else
               QL=Q0*sin(pi*X/XL)*sin(pi*Y/YL);
           end
       else
            QL=Q0+QX*X+QY*Y;
        end
    end
%
      II=1;
      for I=1:NPE
      II1=II+1;
      II2=II+2;
      II3=II+3;
      II4=II+4;
      JJ=1;
      for J=1:NPE
      JJ1=JJ+1;
      JJ2=JJ+2;
      JJ3=JJ+3;
      JJ4=JJ+4;
      SXX=GDSF(1,I)*GDSF(1,J);
      SXY=GDSF(1,I)*GDSF(2,J);
      SYX=GDSF(2,I)*GDSF(1,J);
      SYY=GDSF(2,I)*GDSF(2,J);
%
%     All theories; stiffness coefficients for inplane
DOF:_____________
%
      ELK(II,JJ)  =ELK(II,JJ)  +CNST*(A(1,1)*SXX+A(1,3)*(SXY+SYX)
+A(3,3)*SYY);
      ELK(II,JJ1) =ELK(II,JJ1)
+CNST*(A(1,2)*SXY+A(1,3)*SXX+A(2,3)*SYY+A(3,3)*SYX);
      ELK(II1,JJ) =ELK(II1,JJ)
+CNST*(A(1,2)*SYX+A(1,3)*SXX+A(2,3)*SYY+A(3,3)*SXY);

ELK(II1,JJ1)=ELK(II1,JJ1)+CNST*(A(3,3)*SXX+A(2,3)*(SXY+SYX)+A(2,2)*SYY)
;
      if IDYN>0
```

```
        S00=SF(I)*SF(J)*CNST;
        ELM(II,JJ)  =ELM(II,JJ)  +P0*S00;
        ELM(II1,JJ1)=ELM(II1,JJ1)+P0*S00;
    end
%
%     First-order theory; stiffness coefficients for bending
DOF:_______
%
      if ITYPE>1
      ELK(II,JJ3) =ELK(II,JJ3)
+CNST*(B(1,1)*SXX+B(1,3)*(SXY+SYX)+B(3,3)*SYY);
      ELK(II3,JJ) =ELK(II3,JJ)
+CNST*(B(1,1)*SXX+B(1,3)*(SXY+SYX)+B(3,3)*SYY);
      ELK(II,JJ4) =ELK(II,JJ4)
+CNST*(B(1,2)*SXY+B(1,3)*SXX+B(2,3)*SYY+B(3,3)*SYX);
      ELK(II4,JJ) =ELK(II4,JJ)
+CNST*(B(1,2)*SYX+B(1,3)*SXX+B(2,3)*SYY+B(3,3)*SXY);

ELK(II1,JJ3)=ELK(II1,JJ3)+CNST*(B(1,3)*SXX+B(3,3)*SXY+B(1,2)*SYX+B(2,3)
*SYY);

ELK(II3,JJ1)=ELK(II3,JJ1)+CNST*(B(1,3)*SXX+B(3,3)*SYX+B(1,2)*SXY+B(2,3)
*SYY);

ELK(II1,JJ4)=ELK(II1,JJ4)+CNST*(B(3,3)*SXX+B(2,3)*(SXY+SYX)+B(2,2)*SYY)
;

ELK(II4,JJ1)=ELK(II4,JJ1)+CNST*(B(3,3)*SXX+B(2,3)*(SXY+SYX)+B(2,2)*SYY)
;

ELK(II3,JJ3)=ELK(II3,JJ3)+CNST*(D(1,1)*SXX+D(1,3)*(SXY+SYX)+D(3,3)*SYY)
;

ELK(II3,JJ4)=ELK(II3,JJ4)+CNST*(D(1,2)*SXY+D(3,3)*SYX+D(1,3)*SXX+D(2,3)
*SYY);

ELK(II4,JJ3)=ELK(II4,JJ3)+CNST*(D(1,2)*SYX+D(3,3)*SXY+D(1,3)*SXX+D(2,3)
*SYY);

ELK(II4,JJ4)=ELK(II4,JJ4)+CNST*(D(3,3)*SXX+D(2,3)*(SXY+SYX)+D(2,2)*SYY)
;
         if IDYN>0
            ELM(II,JJ3) =ELM(II,JJ3) +P1*S00;
            ELM(II3,JJ) =ELM(II3,JJ) +P1*S00;
            ELM(II1,JJ4)=ELM(II1,JJ4)+P1*S00;
            ELM(II4,JJ1)=ELM(II4,JJ1)+P1*S00;
            ELM(II2,JJ2)=ELM(II2,JJ2)+P0*S00;
            ELM(II3,JJ3)=ELM(II3,JJ3)+P2*S00;
            ELM(II4,JJ4)=ELM(II4,JJ4)+P2*S00;
         end

    else

         for K=1:NET
         K1=(J-1)*NET+K;
         JK2=JJ2+K-1;
```

```
          RXXX=GDSF(1,I)*GDDSFH(1,K1);
          RXXY=GDSF(1,I)*GDDSFH(3,K1);
          RXYY=GDSF(1,I)*GDDSFH(2,K1);
          RYXX=GDSF(2,I)*GDDSFH(1,K1);
          RYXY=GDSF(2,I)*GDDSFH(3,K1);
          RYYY=GDSF(2,I)*GDDSFH(2,K1);
       ELK(II,JK2) =ELK(II,JK2) -
CNST*(B(1,1)*RXXX+B(1,2)*RXYY+2.0*B(2,3)*RXXY+B(1,3)*RYXX+B(2,3)*RYYY+2
.0*B(3,3)*RYXY);
       ELK(II1,JK2)=ELK(II1,JK2)-
CNST*(B(1,3)*RXXX+B(2,3)*RXYY+2.0*B(3,3)*RXXY+B(1,2)*RYXX+B(2,2)*RYYY+2
.0*B(2,3)*RYXY);
          if IDYN>0
             SX0=GDSF(1,I)*SFH(K1);
             SY0=GDSF(2,I)*SFH(K1);
             ELM(II,JK2)  =ELM(II,JK2)  +P1*SX0*CNST;
             ELM(II1,JK2) =ELM(II1,JK2) +P1*SY0*CNST;
         end

    end

          for K=1:NET
          K1=(I-1)*NET+K;
          IK2=II2+K-1;
          RXXX=GDSF(1,J)*GDDSFH(1,K1);
          RXXY=GDSF(1,J)*GDDSFH(3,K1);
          RXYY=GDSF(1,J)*GDDSFH(2,K1);
          RYXX=GDSF(2,J)*GDDSFH(1,K1);
          RYXY=GDSF(2,J)*GDDSFH(3,K1);
          RYYY=GDSF(2,J)*GDDSFH(2,K1);
       ELK(IK2,JJ) =ELK(IK2,JJ) -
CNST*(B(1,1)*RXXX+B(1,2)*RXYY+2.0*B(2,3)*RXXY+B(1,3)*RYXX+B(2,3)*RYYY+2
.0*B(3,3)*RYXY);
       ELK(IK2,JJ1)=ELK(IK2,JJ1)-
CNST*(B(1,3)*RXXX+B(2,3)*RXYY+2.0*B(3,3)*RXXY+B(1,2)*RYXX+B(2,2)*RYYY+2
.0*B(2,3)*RYXY);
          if IDYN>0
             SX0=GDSF(1,J)*SFH(K1);
             SY0=GDSF(2,J)*SFH(K1);
             ELM(IK2,JJ)  =ELM(IK2,JJ)  +P1*SX0*CNST;
             ELM(IK2,JJ1) =ELM(IK2,JJ1) +P1*SY0*CNST;
         end

    end

          for  K=1:NET
          K1=(I-1)*NET+K;
          IK2=II2+K-1;
          for L=1:NET
          L1=(J-1)*NET+L;
          JL2=JJ2+L-1;
          TXXXX=GDDSFH(1,K1)*GDDSFH(1,L1);
          TXXYY=GDDSFH(1,K1)*GDDSFH(2,L1);
          TXXXY=GDDSFH(1,K1)*GDDSFH(3,L1);
          TXYXX=GDDSFH(3,K1)*GDDSFH(1,L1);
          TYYXX=GDDSFH(2,K1)*GDDSFH(1,L1);
```

```
          TYYXY=GDDSFH(2,K1)*GDDSFH(3,L1);
          TXYYY=GDDSFH(3,K1)*GDDSFH(2,L1);
          TYYYY=GDDSFH(2,K1)*GDDSFH(2,L1);
          TXYXY=GDDSFH(3,K1)*GDDSFH(3,L1);

ELK(IK2,JL2)=ELK(IK2,JL2)+CNST*(D(1,1)*TXXXX+D(1,2)*(TXXYY+TYYXX)+2.0*D
(1,3)*(TXXXY+TXYXX)+2.0*D(2,3)*(TXYYY+TYYXY)+4.0*D(3,3)*TXYXY+D(2,2)*TY
YYY);
          if IDYN>0
             S00=SFH(K1)*SFH(L1);
             SXY=GDSFH(1,K1)*GDSFH(1,L1)+GDSFH(2,K1)*GDSFH(2,L1);
             ELM(IK2,JL2)  =ELM(IK2,JL2)+CNST*(P0*S00+P2*SXY);
         end
     end
end
end
       JJ = NDF*J+1;
  end

if ITYPE>1
          ELF(II2)=ELF(II2)+CNST*SF(I)*QL;
   if NTHERM~=0
      ELF(II) =ELF(II) -CNST*(GDSF(1,I)*TN(1)+GDSF(2,I)*TN(3));
      ELF(II1)=ELF(II1)-CNST*(GDSF(1,I)*TN(3)+GDSF(2,I)*TN(2));
      ELF(II3)=ELF(II3)-CNST*(GDSF(1,I)*TM(1)+GDSF(2,I)*TM(3));
      ELF(II4)=ELF(II4)-CNST*(GDSF(1,I)*TM(3)+GDSF(2,I)*TM(2));
   end
  if PIEZO_E(N)~=0
      ELF(II) =ELF(II) -CNST*(GDSF(1,I)*PN(1)+GDSF(2,I)*PN(3));
      ELF(II1)=ELF(II1)-CNST*(GDSF(1,I)*PN(3)+GDSF(2,I)*PN(2));
      ELF(II3)=ELF(II3)-CNST*(GDSF(1,I)*PM(1)+GDSF(2,I)*PM(3));
      ELF(II4)=ELF(II4)-CNST*(GDSF(1,I)*PM(3)+GDSF(2,I)*PM(2));
  end
else
          for  K=1:NET
          IK2=II2+K-1;
          K1=(I-1)*NET+K;
          ELF(IK2)=ELF(IK2)+CNST*SFH(K1)*QL;
          if NTHERM~=0

ELF(IK2)=ELF(IK2)+CNST*(GDDSFH(1,K1)*TM(1)+2.0*GDDSFH(3,K1)*TM(3)+GDDSF
H(2,K1)*TM(2));
          end
          if PIEZO_E(N)~=0

ELF(IK2)=ELF(IK2)+CNST*(GDDSFH(1,K1)*PM(1)+2.0*GDDSFH(3,K1)*PM(3)+GDDSF
H(2,K1)*PM(2));
          end
          end
end
   II = NDF*I+1;
end
end
end
%
%     First-order theory; stiffness coefficients for transverse shear.
%     Use reduced integration to evaluate coefficients associated with
```

```
%     TRANSVERSE SHEAR
terms:_____________________________________________
%
      if ITYPE>1
         for  NI=1:IPDR
         for  NJ=1:IPDR
         XI  = GAUSPT(NI,IPDR);
         ETA = GAUSPT(NJ,IPDR);
%
         [DET]=SHPRCT(NPE,XI,ETA,ELXY,ITYPE);
%
         CNST=DET*GAUSWT(NI,IPDR)*GAUSWT(NJ,IPDR)*AK;
%
         II=1;
         for  I=1:NPE
         II2=II+2;
         II3=II+3;
         II4=II+4;
         JJ = 1;
         for J=1:NPE
         JJ2=JJ+2;
         JJ3=JJ+3;
         JJ4=JJ+4;
         S00=SF(I)*SF(J);
         SX0=GDSF(1,I)*SF(J);
         S0X=SF(I)*GDSF(1,J);
         SY0=GDSF(2,I)*SF(J);
         S0Y=SF(I)*GDSF(2,J);
         SXX=GDSF(1,I)*GDSF(1,J);
         SXY=GDSF(1,I)*GDSF(2,J);
         SYX=GDSF(2,I)*GDSF(1,J);
         SYY=GDSF(2,I)*GDSF(2,J);

ELK(II2,JJ2)=ELK(II2,JJ2)+CNST*(A55*SXX+A44*SYY+A45*(SXY+SYX));
         ELK(II2,JJ3)=ELK(II2,JJ3)+CNST*(A55*SX0+A45*SY0);
         ELK(II3,JJ2)=ELK(II3,JJ2)+CNST*(A55*S0X+A45*S0Y);
         ELK(II2,JJ4)=ELK(II2,JJ4)+CNST*(A45*SX0+A44*SY0);
         ELK(II4,JJ2)=ELK(II4,JJ2)+CNST*(A45*S0X+A44*S0Y);
         ELK(II3,JJ3)=ELK(II3,JJ3)+CNST*A55*S00;
         ELK(II3,JJ4)=ELK(II3,JJ4)+CNST*A45*S00;
         ELK(II4,JJ3)=ELK(II4,JJ3)+CNST*A45*S00;
         ELK(II4,JJ4)=ELK(II4,JJ4)+CNST*A44*S00;
       JJ=NDF*J+1;
end
     II=NDF*I+1;
 end
end
end
end
%       if IDYN>0 & NEIGN==0
% %
% %     The Newmark intergration scheme for hyperbolic equations
% %
%          if NCOUNT==1 & INTIAL~=0
%             for I = 1:NN
%                 ELF(I)  = 0.0;
%                 for  J = 1:NN
```

```
%                     ELF(I)   = ELF(I)-ELK(I,J)*ELU(J);
%                     ELK(I,J)= ELM(I,J);
%                 end
%             end
%         else
%             for  I = 1:NN
%                 SUM     = 0.0;
%                 for  J = 1:NN
%                     SUM     =
SUM+ELM(I,J)*(A3*ELU(J)+A4*ELV(J)+A5*ELA(J));
%                     ELK(I,J)= ELK(I,J)+A3*ELM(I,J);
%                  end
%                 ELF(I)   = ELF(I)+SUM;
%             end
%          end
%       end
```

ELSTIF.m

```
function []=ELSTIF(M)
global E1 E2 G12 G13 G23 ANU12 RHO ALFA1 ALFA2 THETA THKNS MATYP
%Material
global Q QBAR ALFA Z
%Elastic
ANU21=ANU12(M)*E2(M)/E1(M);
DENOM=(1.0-ANU12(M)*ANU21);

Q(1,1)=E1(M)/DENOM;
Q(1,2)=ANU12(M)*E2(M)/DENOM;
Q(2,1)=Q(1,2);
Q(2,2)=Q(1,1)*E2(M)/E1(M);
Q(3,3)=G12(M);
Q(4,4)=G23(M);
Q(5,5)=G13(M);
```

feaplyc2.m

```
function [kk,ff]=feaplyc2(kk,ff,ISPV,ISSV,NDF,NEQ,NSPV,NSSV,VSPV,VSSV)
for i=1:NSPV
    id=(ISPV(i,1)-1)*NDF+ISPV(i,2);
%     for j=1:NEQ
%        kk(c,j)=0;
%     end
%
%     kk(c,c)=1;
%     ff(c)=VSPV(i);
    value=VSPV(i);
    for j=1:NEQ
        ff(j)=ff(j)-value*kk(j,id);
        kk(id,j)=0;
        kk(j,id)=0;
    end
    kk(id,id)=1;
    ff(id)=value;
 end
```

feaplycs.m

```
function [kk,mm]=feaplycs(kk,mm,ISPV,NSPV,NEQ)
```

```
for i=1:NSPV
    c=ISPV(i);
    for j=1:NEQ
       kk(c,j)=0;
       kk(j,c)=0;
       mm(c,j)=0;
       mm(j,c)=0;
    end
   mm(c,c)=1;
 end
```

feasmbl1.m

```
function [kk]=feasmbl1(kk,index)
global  ELM                                    % Stiff1
k=ELM;
for i=1:length(index)
   ii=index(i);
     for j=1:length(index)
       jj=index(j);
         kk(ii,jj)=kk(ii,jj)+k(i,j);
     end
 end
```

feasmbl2.m

```
function [kk,ff]=feasmbl2(kk,ff,index)
global ELF ELK ELM                                  % Stiff1
 k=ELK;
 f=ELF;
 for i=1:length(index)
   ii=index(i);
     ff(ii)=ff(ii)+f(i);
     for j=1:length(index)
         jj=index(j);
         kk(ii,jj)=kk(ii,jj)+k(i,j);
     end
 end
```

feeldof.m

```
function [index]=feeldof(NOD,NPE,NDF,N)
NN = NPE*NDF;
   k=0;
   for i=1:NPE
     start = (NOD(N,i)-1)*NDF;
       for j=1:NDF
         k=k+1;
         index(k)=start+j;
       end
   end
```

femesh.m

```
function femesh();
for I=1:NNM
    II=NDF*(I-1)+1;
    JJ=II+NDF1-1;
    GLFN(II)=GLFN(II);
    GLFN(II+1)=GLFN(II+1);
```

```
    GLFN(II+2)=GLFN(II+2);
    fprintf(LOUT,'\n %d \t',I);
    X(I)=GLXY(I,1);
    Y(I)=GLXY(I,2);
    for kk=1:5
        for J=II:JJ
%         fprintf(LOUT,'%e\t',GLFN(J));
        disp(kk)=GLFN(J);
        end
    end
    XN(I)=X(I)+disp(1);
    YN(I)=Y(I)+disp(2);
    ZN(I)=disp(3);
end
[XX,YY,ZZ]=meshgrid(XN,YN,ZN);
surf(XX,YY,ZZ)
```

MSH2DR.m

```
function [NOD,GLXY,NNM,NEM]=MSH2DR(IEL,NX,NY,NPE,DX,DY,X0,Y0)
% IMPLICIT REAL*8 (A-H,O-Z)
%      DIMENSION NOD(MAXNEM,9),GLXY(MAXNNM,2),DX(MAXNX),DY(MAXNY)
%      COMMON/IO/IN,ITT
%
IELTYP=1;%always quadratic element
      NEX1 = NX+1;
      NEY1 = NY+1;
      NXX  = IEL*NX;
      NYY  = IEL*NY;
      NXX1 = NXX + 1;
      NYY1 = NYY + 1;
      NEM  = NX*NY;
      if IELTYP==0 NEM=2*NX*NY;  end
      NNM=NXX1*NYY1;
      if NPE==8 NNM = NXX1*NYY1 - NX*NY; end
%
      if IELTYP==0
%     Generate the array [NOD] and global coordinates [GLXY] for
%     TRIANGULAR ELEMENTS
%
         NX2=2*NX;
         NY2=2*NY;
         NOD(1,1) = 1;
         NOD(1,2) = IEL+1;
         NOD(1,3) = IEL*NXX1+IEL+1;
         if NPE>3
             NOD(1,4) = 2;
             NOD(1,5) = NXX1 + 3;
             NOD(1,6) = NXX1 + 2;
         end
%
         NOD(2,1) = 1;
         NOD(2,2) = NOD(1,3);
         NOD(2,3) = IEL*NXX1+1;
         if NPE>3
            NOD(2,4) = NOD(1,6);
            NOD(2,5) = NOD(1,3) - 1;
            NOD(2,6) = NOD(2,4) - 1;
```

```
        end
%
        K=3;
        for IY=1:NY
            L=IY*NX2;
            M=(IY-1)*NX2;
            if NX>1
              for N=K:2:L
              for I=1:NPE
                NOD(N,I)   = NOD(N-2,I)+IEL;
                NOD(N+1,I)= NOD(N-1,I)+IEL;
              end
             end
            end
 %
             if IY<NY
              for I=1:NPE
                NOD(L+1,I)=NOD(M+1,I)+IEL*NXX1;
                NOD(L+2,I)=NOD(M+2,I)+IEL*NXX1;
              end
            end
           K=L+3
         end
%
%
    else
%
%       Generate the array [NOD] and global coordinates [GLXY] for
%       RECTANGULAR ELEMENTS
%
        K0 = 0;
        if NPE==9  K0=1; end
        NOD(1,1) = 1;
        NOD(1,2) = IEL+1;
        NOD(1,3) = NXX1+(IEL-1)*NEX1+IEL+1;
        if NPE==9  NOD(1,3)=4*NX+5; end
        NOD(1,4) = NOD(1,3) - IEL;
        if NPE>4
           NOD(1,5) = 2;
           NOD(1,6) = NXX1 + (NPE-6);
           NOD(1,7) = NOD(1,3) - 1;
           NOD(1,8) = NXX1+1;
           if NPE==9
               NOD(1,9)=NXX1+2;
           end
        end
%
        if NY>1
           M = 1;
           for N = 2:NY
           L = (N-1)*NX + 1;
           for I = 1:NPE
             NOD(L,I) = NOD(M,I)+NXX1+(IEL-1)*NEX1+K0*NX;
           end
            M=L;
           end
         end
```

```
%
        if NX>1
         for NI=2:NX
           for I = 1:NPE
               K1 = IEL;
               if I==6 | I==8   K1=1+K0; end
               NOD(NI,I) = NOD(NI-1,I)+K1;
           end
           M = NI;
           for NJ = 2:NY
             L = (NJ-1)*NX+NI;
             for J = 1:NPE
               NOD(L,J) = NOD(M,J)+NXX1+(IEL-1)*NEX1+K0*NX;
             end
           M = L;
           end
         end
        end
    end          %END IF FOR SELECTION OF RECT OR TRI


%          Generate the global coordinates of the nodes, [GLXY]
     DX(NEX1)=0.0;
     DY(NEY1)=0.0;
     XC=X0;
     YC=Y0;
     if NPE==8
        for NI = 1:NEY1
        I = (NXX1+NEX1)*(NI-1)+1;
        J = 2*NI-1;
        GLXY(I,1) = XC;
        GLXY(I,2) = YC;
            for NJ = 1:NX
            DELX=0.5*DX(NJ);
            I=I+1;
            GLXY(I,1) = GLXY(I-1,1)+DELX;
            GLXY(I,2) = YC;
            I=I+1;
            GLXY(I,1) = GLXY(I-1,1)+DELX;
            GLXY(I,2) = YC;
            end
         if NI<NY
           I = I+1;
           YC= YC+0.5*DY(NI);
           GLXY(I,1) = XC;
           GLXY(I,2) = YC;
             for II = 1:NX
             I = I+1;
             GLXY(I,1) = GLXY(I-1,1)+DX(II);
             GLXY(I,2) = YC;
             end
         end

        YC = YC+0.5*DY(NI);
        end
%
```

```
     else
%
         YC=Y0;
         for NI = 1:NEY1
         XC = X0;
         I = NXX1*IEL*(NI-1);
         for NJ = 1:NEX1
         I=I+1;
         GLXY(I,1) = XC;
         GLXY(I,2) = YC;
         if NJ<NEX1
            if IEL==2
               I=I+1;
               XC = XC + 0.5*DX(NJ);
               GLXY(I,1) = XC;
               GLXY(I,2) = YC;
            end
         end
         XC = XC + DX(NJ)/IEL;
         end
         XC = X0;
         if IEL==2
            YC = YC + 0.5*DY(NI);
            for NJ = 1:NEX1
            I=I+1;
            GLXY(I,1) = XC;
            GLXY(I,2) = YC;
            if NJ<NEX1
               I=I+1;
               XC = XC + 0.5*DX(NJ);
               GLXY(I,1) = XC;
               GLXY(I,2) = YC;
            end
            XC = XC + 0.5*DX(NJ);
            end
          end
         YC = YC + DY(NI)/IEL;
         end     %for loop
      end
```

piezo_resultant.m

```
function [SP]=piezo_resultant(p_tk, p_theta, EC31, EC32, EC36, EC14,
EC24, EC15, EC25, AV, NPLY,ITYPE)
%-----------------------------------------------------------------------
% Calculates the piezo resutants PN, PM
%-----------------------------------------------------------------------
% global P_tk p_theta EC13 EC23 EC33 AV NPLY
%Piezo
global PM PN
PN=zeros(3,1); PM=zeros(3,1);
for K=1:NPLY
    EP=[0; 0; -AV/p_tk];
    %Form EC matrix
    EC=zeros(3,3); % EC=zeros(3,6);
    ECBAR=zeros(3,3); %ECBAR=zeros(3,6);
    EC(3,1)=EC31; EC(3,2)=EC32;% EC(3,6)=EC36;
```

```
    %Find ECBAR (Transformed Pieoelectric coefficients)
    CN=cos(p_theta);
    SN=sin(p_theta);
    SN2=SN*SN;
    CN2=CN*CN;

    ECBAR(3,1)=EC(3,1)*CN2+EC(3,2)*SN2;
    ECBAR(3,2)=EC(3,1)*SN2+EC(3,2)*CN2;
    ECBAR(3,3)=EC(3,3);
   % ECBAR(3,6)=(EC(3,1)-EC(3,2))*SN*CN;

%     if ITYPE>0
%         ECBAR(1,4)=(EC15-EC24)*SN*CN;
%         ECBAR(2,4)=EC24*CN2+EC15*SN2;
%         ECBAR(1,5)=EC15*CN2+EC24*SN2;
%         ECBAR(2,5)=(EC15-EC24)*SN*CN;
%     end
%
    [QBARP]=QBARPIEZO(2,p_theta); %2 is the materials number later have
to be changed
    ECBAR=ECBAR*QBARP;
    ECBAR=ECBAR';
    %Resultants
    %for CLPT
    for I=1:3
        for J=1:3
            PN(I)=PN(I)+ECBAR(I,J)*EP(J)*p_tk;
            PM(I)=PM(I)+ECBAR(I,J)*EP(J)*0.5*p_tk^2;
        end
    end
%     PM=2*PM; PN=PN*2;
    %for FSDT
    if ITYPE>0
     SP=ECBAR(4:5,:)*EP*p_tk;
     else
        SP=zeros(2,1);
    end
end
```

PLTSTF.m

```
function []=PLTSTF(LAYERS,NSHR,IDYN,NTHERM,NPRNT,HSCALE)
% _____________________________________________________________________
% A,B,D,E,F,G,H: Laminate stiffnesses (integrals of Q(I,J)
% multiplied with 1,Z,Z2,Z3,Z4,Z5,Z6, respectively)
% ALFA(I): Thermal coefficients of expansion in the global coordinates
% (x,y,z).
% HT............. Total thickness of the laminate
% _____________________________________________________________________

global LINP LOUT                                                %Files
global E1 E2 G12 G13 G23 ANU12 RHO ALFA1 ALFA2 THETA THKNS MATYP
%Material
global A B D E F G H A44 A45 A55 D44 D45 D55
%Stiff2
global Q QBAR ALFA Z
%Elastic
```

```
global P0 P1 P2 P3 P4 P5 P6                                          %Integ

IDYN=1;
RHO=[7000 7000];


      IT=6;
      HT=0.0;
      for I=1:LAYERS  HT=HT+THKNS(I)*HSCALE;  end
      LL = LAYERS+1;
      Z(1) = -0.5*HT;
      for  I=2:LL Z(I) = Z(I-1) + THKNS(I-1)*HSCALE;  end

% In computing the laminate stiffnesses it is assumed that the
% midplane of the laminate is the xy-plane

for I=1:3
       if NTHERM>0
         TN(I)=0.0;
         TM(I)=0.0;
        end
     for J=1:3
      A(I,J)=0.0;
      B(I,J)=0.0;
      D(I,J)=0.0;
  end
end
      if NSHR==1
         A44=0.0;
         A45=0.0;
         A55=0.0;
     end
     if IDYN>0
         P0=0.0;
         P1=0.0;
         P2=0.0;
     end
     if NSHR>1
         for  I=1:3
            if NTHERM>0
                TL(I)=0.0;
                TK(I)=0.0;
            end
         for J=1:3
             E(I,J)=0.0;
             F(I,J)=0.0;
             G(I,J)=0.0;
             H(I,J)=0.0;
         end
     end

         D44=0.0;
         D45=0.0;
         D55=0.0;
         F44=0.0;
         F45=0.0;
```

```
         F55=0.0;
         if IDYN>0
            P3=0.0;
            P4=0.0;
            P5=0.0;
            P6=0.0;
        end
     end
for K=1:LAYERS
      ANGLE = pi*THETA(K)/180.0;
      QBARSTF(K,ANGLE,NTHERM);
      if NPRNT~=0
         if K<=2
            if K==1
               fprintf(LOUT,'\n\n\t\tTRANSFORMED STIFFNESSES FOR THE
FIRST LAYER:\n\n');
           else
               fprintf(LOUT, '\n\n\n\t\t TRANSFORMED STIFFNESSES FOR
THE SECOND LAYER:\n\n');           end
            fprintf(LOUT,'\n Elastic stiffness coefficients, [Q] \n');
for I=1:3
                fprintf(LOUT,'\n');
                for J=1:3 fprintf(LOUT,'\t %e', Q(I,J)); end
end
            fprintf(LOUT,'\n\tLamina stiffnesses, QBAR(I,J): \n');
for I=1:3
                fprintf(LOUT,'\n');
                for J=1:3 fprintf(LOUT,'\t %e',QBAR(I,J)); end
end
            if NSHR>=1
               fprintf(LOUT, '\n\tLamina stiffnesses, QBAR44, QBAR45,
QBAR55: \n');               for I=4:5
                    fprintf(LOUT,'\n');
                    for J=4:5 fprintf(LOUT,'\t %e',QBAR(I,J)); end
end
            end
            if NTHERM>0
               for I=1:3 fprintf(LOUT,'\t %e',ALFA(I)); end
           end
       end
      end
%     ..............................................................
%                    COMPUTE THE LAMINATE STIFFNESSES
%     ..............................................................
%     Laminate stiffnesses for the classical and first-order theories
      ZTB=Z(K+1)-Z(K);
      ZT2=Z(K+1)*Z(K+1);
      ZB2=Z(K)*Z(K);
      ZT3=ZT2*Z(K+1);
      ZB3=ZB2*Z(K);
      for I=1:3
            for J=1:3
                if NTHERM>0
                    TN(I)=TN(I)+QBAR(I,J)*ALFA(J)*(T0*ZTB+0.5*T1*(ZT2-
ZB2));
                    TM(I)=TM(I)+QBAR(I,J)*ALFA(J)*(0.5*T0*(ZT2-
ZB2)+T1*(ZT3-ZB3)/3.0);
```

```
                end
                 A(I,J)=A(I,J)+QBAR(I,J)*ZTB;
                 B(I,J)=B(I,J)+QBAR(I,J)*(ZT2-ZB2)/2.0;
                 D(I,J)=D(I,J)+QBAR(I,J)*(ZT3-ZB3)/3.0;
             end
          end

       if IDYN>0
          M=MATYP(K);
          P0=P0+RHO(M)*ZTB;
          P1=P1+RHO(M)*(ZT2-ZB2)/2.0;
          P2=P2+RHO(M)*(ZT3-ZB3)/3.0;
      end
       if NSHR>0
          A44=A44+QBAR(4,4)*ZTB;
          A45=A45+QBAR(4,5)*ZTB;
          A55=A55+QBAR(5,5)*ZTB;
     S(1,1)=A44;
     S(1,2)=A45;
     S(2,1)=A45;
     S(2,2)=A55;
end
%
%     Laminate stiffnesses for the third-order theory
%
       if NSHR>1
          ZT4=ZT2*ZT2;
          ZB4=ZB2*ZB2;
          ZT5=ZT3*ZT2;
          ZB5=ZB3*ZB2;
          ZB6=ZB3*ZB3;
          ZT6=ZT3*ZT3;
          ZT7=ZT5*ZT2;
          ZB7=ZB5*ZB2;
       for I=1:3
          for J=1:3
             if NTHERM>0
                  TL(I)=TL(I)+QBAR(I,J)*ALFA(J)*(T0*(ZT3-
ZB3)/3.0+T1*(ZT4-ZB4)/4.0);
                  TK(I)=TK(I)+QBAR(I,J)*ALFA(J)*(T0*(ZT4-
ZB4)/4.0+T1*(ZT5-ZB5)/5.0);
              end
             E(I,J)=E(I,J)+QBAR(I,J)*(ZT4-ZB4)/4.0;
              F(I,J)=F(I,J)+QBAR(I,J)*(ZT5-ZB5)/5.0;
              G(I,J)=G(I,J)+QBAR(I,J)*(ZT6-ZB6)/6.0;
             H(I,J)=H(I,J)+QBAR(I,J)*(ZT7-ZB7)/7.0;
         end
     end
          D44=D44+QBAR(4,4)*(ZT3-ZB3)/3.0;
          D45=D45+QBAR(4,5)*(ZT3-ZB3)/3.0;
          D55=D55+QBAR(5,5)*(ZT3-ZB3)/3.0;
          F44=F44+QBAR(4,4)*(ZT5-ZB5)/5.0;
          F45=F45+QBAR(4,5)*(ZT5-ZB5)/5.0;
          F55=F55+QBAR(5,5)*(ZT5-ZB5)/5.0;
          if IDYN>0
             P3=P3+RHO(M)*(ZT4-ZB4)/4.0;
             P4=P4+RHO(M)*(ZT5-ZB5)/5.0;
```

```
            P5=P5+RHO(M)*(ZT6-ZB6)/6.0;
            P6=P6+RHO(M)*(ZT7-ZB7)/7.0;
        end
    end
end

if NPRNT~=0
         fprintf(LOUT,'\n Laminate stiffnesses, [A]:\n' );
        for I=1:3
            fprintf(LOUT,'\n');
            for J=1:3 fprintf(LOUT, '\t %e', A(I,J)); end
        end
        fprintf(LOUT, '\n Laminate stiffnesses, [B]:\n ');
        for I=1:3
            fprintf(LOUT,'\n');
            for J=1:3 fprintf(LOUT, '\t %e', B(I,J)); end
        end
         fprintf(LOUT,'\n Laminate stiffnesses, [D]:\n');
        for I=1:3
            fprintf(LOUT,'\n');
            for J=1:3 fprintf(LOUT, '\t %e', D(I,J)); end
        end
         if IDYN>0
                    fprintf(LOUT, '\n Laminate mass inertias
(A0,A1,A2):\n ');
                    fprintf(LOUT,  '\t %e\t%e\t%e \n',P0,P1,P2);
end

         if NSHR>0
            fprintf(LOUT,'\n\tLaminate shear stiffnesses
(A44,A45,A55):\n');

fprintf(LOUT,'\t %e\t%e\t%e \n',A44,A45,A55);

end

if NTHERM>0
            fprintf(LOUT, '\nThermal force resultants, NTXX, NTYY,
NTXY: ');

for I=1:3 fprintf(LOUT, '\t %e',TN(I)); end
            fprintf(LOUT, '\n Thermal moment resultants, MTXX, MTYY,
MTXY: ');

 for I=1:3 fprintf(LOUT, '\t %e',TM(I)); end         end

         if NSHR>1
            fprintf(LOUT, '\n Laminate stiffnesses, [E]:\n');
for I=1:3
                fprintf(LOUT,'\n');
                for J=1:3 fprintf(LOUT, '\t %e',E(I,J)); end
            end
            fprintf(LOUT, '\n Laminate stiffnesses, [F]:\n');
for I=1:3
                fprintf(LOUT,'\n');
                for J=1:3 fprintf(LOUT, '\t %e',F(I,J)); end
```

```
            end
            fprintf(LOUT, '\nLaminate stiffnesses, [G]:\n');
for I=1:3
                fprintf(LOUT,'\n');
                for J=1:3 fprintf(LOUT, '\t %e',G(I,J)); end
            end
            fprintf(LOUT, '\n Laminate stiffnesses, _H]:\n ');
for I=1:3
                fprintf(LOUT,'\n');
                for J=1:3 fprintf(LOUT, '\t %e',H(I,J)); end
            end
            fprintf(LOUT,'\n Laminate shear stiffnesses (D44,D45,D55):
\n');
            fprintf(LOUT,'\t %e\t%e\t%e \n', D44,D45,D55);
            fprintf(LOUT,'\n Laminate shear stiffnesses
(F44,F45,F55):\n');
            fprintf(LOUT,'\t %e\t%e\t%e \n',F44,F45,F55);
            if IDYN>0
               fprintf(LOUT,'Laminate mass inertias
(A3,A4,A5,A6):\n');)
               fprintf(LOUT,'\t %e\t%e\t%e\t%e\n', P3,P4,P5,P6);
          end%if
           if NTHERM>0
               fprintf(LOUT, 'Higher order thermal resultants, LTXX,
LTYY, LTXY:\t');
               for I=1:3 fprintf(LOUT, '\t %e',TL(I)); end
               fprintf(LOUT,'\n Higher order thermal resultants, KTXX,
KTYY, KTXY:\t');
               for I=1:3 fprintf(LOUT, '\t %e',TK(I)); end
           end
       end
   end
```

QBARPIEZO.m

```
function [QBARP]=QBARPIEZO(M,p_theta)
global E1 E2 G12 G13 G23 ANU12 RHO ALFA1 ALFA2 THETA THKNS MATYP
%Material

      ANU21=ANU12(M)*E2(M)/E1(M);
      DENOM=(1.0-ANU12(M)*ANU21);
      QP(1,1)=E1(M)/DENOM;
      QP(1,2)=ANU12(M)*E2(M)/DENOM;
      QP(2,1)=QP(1,2);
      QP(2,2)=QP(1,1)*E2(M)/E1(M);
      QP(3,3)=G12(M);

      ANGLE = pi*p_theta/180.0;

      CN=cos(ANGLE);
      SN=sin(ANGLE);
      SN2=SN*SN;
      SN3=SN2*SN;
      SN4=SN2*SN2;
      CN2=CN*CN;
      CN3=CN2*CN;
      CN4=CN2*CN2;
```

```
%

QBARP(1,1)=QP(1,1)*CN4+2.0*(QP(1,2)+2.0*QP(3,3))*SN2*CN2+QP(2,2)*SN4;
      QBARP(1,2)=(QP(1,1)+QP(2,2)-
4.0*QP(3,3))*SN2*CN2+QP(1,2)*(SN4+CN4);
      QBARP(1,3)=(QP(1,1)-QP(1,2)-2.0*QP(3,3))*SN*CN3+(QP(1,2)-
QP(2,2)+2.0*QP(3,3))*SN3*CN;
      QBARP(2,1)=(QP(1,1)+QP(2,2)-
4.0*QP(3,3))*SN2*CN2+QP(1,2)*(SN4+CN4);

QBARP(2,2)=QP(1,1)*SN4+2.0*(QP(1,2)+2.0*QP(3,3))*SN2*CN2+QP(2,2)*CN4;
      QBARP(2,3)=(QP(1,1)-QP(1,2)-2.0*QP(3,3))*SN3*CN+(QP(1,2)-
QP(2,2)+2.0*QP(3,3))*SN*CN3;
      QBARP(3,1)=(QP(1,1)-QP(1,2)-2.0*QP(3,3))*SN*CN3+(QP(1,2)-
QP(2,2)+2.0*QP(3,3))*SN3*CN;
      QBARP(3,2)=(QP(1,1)-QP(1,2)-2.0*QP(3,3))*SN3*CN+(QP(1,2)-
QP(2,2)+2.0*QP(3,3))*SN*CN3;
      QBARP(3,3)=(QP(1,1)+QP(2,2)-2.0*QP(1,2)-
2.0*QP(3,3))*SN2*CN2+QP(3,3)*(SN4+CN4);
```

QBARSTF.m

```
function []=QBARSTF(K,ANGLE,NTHERM)
global E1 E2 G12 G13 G23 ANU12 RHO ALFA1 ALFA2 THETA THKNS MATYP
global Q QBAR ALFA Z
%Elastic

      M=MATYP(K);
      ELSTIF(M);
      CN=cos(ANGLE);
      SN=sin(ANGLE);
      SN2=SN*SN;
      SN3=SN2*SN;
      SN4=SN2*SN2;
      CN2=CN*CN;
      CN3=CN2*CN;
      CN4=CN2*CN2;
%
      QBAR(1,1)=Q(1,1)*CN4+2.0*(Q(1,2)+2.0*Q(3,3))*SN2*CN2+Q(2,2)*SN4;
      QBAR(1,2)=(Q(1,1)+Q(2,2)-4.0*Q(3,3))*SN2*CN2+Q(1,2)*(SN4+CN4);
      QBAR(1,3)=(Q(1,1)-Q(1,2)-2.0*Q(3,3))*SN*CN3+(Q(1,2)-
Q(2,2)+2.0*Q(3,3))*SN3*CN;
      QBAR(2,1)=(Q(1,1)+Q(2,2)-4.0*Q(3,3))*SN2*CN2+Q(1,2)*(SN4+CN4);
      QBAR(2,2)=Q(1,1)*SN4+2.0*(Q(1,2)+2.0*Q(3,3))*SN2*CN2+Q(2,2)*CN4;
      QBAR(2,3)=(Q(1,1)-Q(1,2)-2.0*Q(3,3))*SN3*CN+(Q(1,2)-
Q(2,2)+2.0*Q(3,3))*SN*CN3;
      QBAR(3,1)=(Q(1,1)-Q(1,2)-2.0*Q(3,3))*SN*CN3+(Q(1,2)-
Q(2,2)+2.0*Q(3,3))*SN3*CN;
      QBAR(3,2)=(Q(1,1)-Q(1,2)-2.0*Q(3,3))*SN3*CN+(Q(1,2)-
Q(2,2)+2.0*Q(3,3))*SN*CN3;
      QBAR(3,3)=(Q(1,1)+Q(2,2)-2.0*Q(1,2)-
2.0*Q(3,3))*SN2*CN2+Q(3,3)*(SN4+CN4);
      QBAR(4,4)=Q(4,4)*CN2+Q(5,5)*SN2;
      QBAR(4,5)=(Q(5,5)-Q(4,4))*CN*SN;
      QBAR(5,4)=(Q(5,5)-Q(4,4))*CN*SN;
      QBAR(5,5)=Q(5,5)*CN2+Q(4,4)*SN2;
```

```
      if NTHERM>0
%     Transformation of the thermal coefficients of expansion
         ALFA(1)=ALFA1(M)*CN2+ALFA2(M)*SN2;
         ALFA(2)=ALFA1(M)*SN2+ALFA2(M)*CN2;
         ALFA(3)=2.0*(ALFA1(M)-ALFA2(M))*SN*CN;
      end
```

SHPRCT.m

```
function [DET]=SHPRCT(NPE,XI,ETA,ELXY,ITYPE)
global SF GDSF SFH GDSFH GDDSFH                                    %Shape
global P0 P1 P2 P3 P4 P5 P6                                        %Integ

XNODE =[-1.0 1.0 1.0 -1.0 0.0 1.0 0.0 -1.0 0.0 ; -1.0 -1.0 1.0 1.0 -1.0
0.0 1.0 0.0 0.0];
XNODE=XNODE';
NP=[1 2 3 4 5 7 6 8 9];
%FNC = A*B;
if NPE==4
    %     LINEAR Lagrange interpolation functions for FOUR-NODE element
          for  I = 1:NPE
            XP  = XNODE(I,1);
            YP  = XNODE(I,2);
            XI0 = 1.0+XI*XP;
            ETA0=1.0+ETA*YP;
            SF(I)   = 0.25*XI0*ETA0;
            DSF(1,I)= 0.25*XP*ETA0;
            DSF(2,I)= 0.25*YP*XI0;
         end
     else

         if NPE==8
%     QUADRATIC Lagrange interpol. funct. for EIGHT-NODE element
(NPE=8)
%
           for I = 1:NPE
               NI   = NP(I);
               XP   = XNODE(NI,1);
               YP   = XNODE(NI,2);
               XI0  = 1.0+XI*XP;
               ETA0 = 1.0+ETA*YP;
               XI1  = 1.0-XI*XI;
               ETA1 = 1.0-ETA*ETA;
               if I<=4
                  SF(NI)    = 0.25*XI0*ETA0*(XI*XP+ETA*YP-1.0);
                  DSF(1,NI) = 0.25*ETA0*XP*(2.0*XI*XP+ETA*YP);
                  DSF(2,NI) = 0.25*XI0*YP*(2.0*ETA*YP+XI*XP);
              else
                  if I<=6
                     SF(NI)    = 0.5*XI1*ETA0;
                     DSF(1,NI) = -XI*ETA0;
                     DSF(2,NI) = 0.5*YP*XI1;
                 else
                     SF(NI)    = 0.5*ETA1*XI0;
                     DSF(1,NI) = 0.5*XP*ETA1;
                     DSF(2,NI) = -ETA*XI0;
```

```
                end
            end
        end
  else
%
%     QUADRATIC Lagrange interpol. funct. for NINE-NODE element (NPE=9)
%
          for I=1:NPE
              NI   = NP(I);
              XP   = XNODE(NI,1);
              YP   = XNODE(NI,2);
              XI0  = 1.0+XI*XP;
              ETA0 = 1.0+ETA*YP;
              XI1  = 1.0-XI*XI;
             ETA1 = 1.0-ETA*ETA;
             XI2  = XP*XI;
             ETA2 = YP*ETA;
             if I<=4
                SF(NI)   = 0.25*XI0*ETA0*XI2*ETA2;
                DSF(1,NI)= 0.25*XP*ETA2*ETA0*(1.0+2.0*XI2);
                DSF(2,NI)= 0.25*YP*XI2*XI0*(1.0+2.0*ETA2);
            else
                if I <=  6
                   SF(NI)    = 0.5*XI1*ETA0*ETA2;
                   DSF(1,NI) = -XI*ETA2*ETA0;
                   DSF(2,NI) = 0.5*XI1*YP*(1.0+2.0*ETA2);
                else
                    if I <= 8
                       SF(NI)    = 0.5*ETA1*XI0*XI2;
                       DSF(2,NI) = -ETA*XI2*XI0;
                       DSF(1,NI) = 0.5*ETA1*XP*(1.0+2.0*XI2);
                   else
                       SF(NI)    = XI1*ETA1;
                       DSF(1,NI) = -2.0*XI*ETA1;
                       DSF(2,NI) = -2.0*ETA*XI1;
                   end
                end
            end
        end %for
 end
end
%
%     Compute the Jacobian matrix [GJ] and its inverse [GJINV]
%
      for I = 1:2
      for J = 1:2
      GJ(I,J)  = 0.0;
      for K = 1:NPE
      GJ(I,J)  = GJ(I,J) + DSF(I,K)*ELXY(K,J);
end
end
end
%
      DET = GJ(1,1)*GJ(2,2)-GJ(1,2)*GJ(2,1);
      GJINV(1,1) = GJ(2,2)/DET;
      GJINV(2,2) = GJ(1,1)/DET;
      GJINV(1,2) = -GJ(1,2)/DET;
```

```
      GJINV(2,1) = -GJ(2,1)/DET;
%
%     Compute the derivatives of the interpolation functions with
%     respect to the global coordinates (x,y): [GDSF]
%
      for I  = 1:2
      for J  = 1:NPE
      GDSF(I,J) = 0.0;
      for K  = 1:2
      GDSF(I,J) = GDSF(I,J) + GJINV(I,K)*DSF(K,J);
  end
end
end

    if ITYPE<=1
         if ITYPE==1
            NET=4*NPE;
            II = 1;
            for I = 1:NPE
            XP   = XNODE(I,1);
            YP   = XNODE(I,2);
            XI1  = XI*XP-1.0;
            XI2  = XI1-1.0;
            ETA1 = ETA*YP-1.0;
            ETA2 = ETA1-1.0;
            XI0  = (XI+XP)^2;
            ETA0 = (ETA+YP)^2;
            XIP0 = XI+XP;
            XIP1 = 3.0*XI*XP+XP*XP;
            XIP2 = 3.0*XI*XP+2.0*XP*XP;
            YIP0 = ETA+YP;
            YIP1 = 3.0*ETA*YP+YP*YP;
            YIP2 = 3.0*ETA*YP+2.0*YP*YP;
%
            SFH(II)       = 0.0625*ETA0*ETA2*XI0*XI2;
            DSFH(1,II)    = 0.0625*ETA0*ETA2*XIP0*(XIP1-4.0);
            DSFH(2,II)    = 0.0625*XI0*XI2*YIP0*(YIP1-4.0);
            DDSFH(1,II)   = 0.125*ETA0*ETA2*(XIP2-2.0);
            DDSFH(2,II)   = 0.125*XI0*XI2*(YIP2-2.0);
            DDSFH(3,II)   = 0.0625*(XIP1-4.0)*(YIP1-4.0)*XIP0*YIP0;
%
            SFH(II+1)     = -0.0625*XP*XI0*XI1*ETA0*ETA2;
            DSFH(1,II+1)  = -0.0625*ETA0*ETA2*XP*XIP0*(XIP1-2.0);
            DSFH(2,II+1)  = -0.0625*XI0*XI1*XP*YIP0*(YIP1-4.);
            DDSFH(1,II+1) = -0.125*ETA0*ETA2*XP*(XIP2-1.0);
            DDSFH(2,II+1) = -0.125*XI0*XI1*(YIP2-2.0)*XP;
            DDSFH(3,II+1) = -0.0625*XP*XIP0*(XIP1-2.)*(YIP1-4.)*YIP0;
%
            SFH(II+2)     = -0.0625*YP*XI0*XI2*(ETA0*ETA1);
            DSFH(1,II+2)  = -0.0625*ETA0*ETA1*YP*XIP0*(XIP1-4.);
            DSFH(2,II+2)  = -0.0625*XI0*XI2*YP*YIP0*(YIP1-2.);
            DDSFH(1,II+2) = -0.125*ETA0*ETA1*YP*(XIP2-2.);
            DDSFH(2,II+2) = -0.125*XI0*XI2*YP*(YIP2-1.0);
            DDSFH(3,II+2) = -0.0625*YP*YIP0*(YIP1-2.)*(XIP1-4.0)*XIP0;
%
            SFH(II+3)     = 0.0625*XP*YP*XI0*XI1*(ETA0*ETA1);
            DSFH(1,II+3)  = 0.0625*ETA0*ETA1*XP*YP*(XIP1-2.)*XIP0;
```

```
            DSFH(2,II+3)  = 0.0625*XI0*XI1*XP*YP*(YIP1-2.)*YIP0;
            DDSFH(1,II+3) = 0.125*ETA0*ETA1*XP*YP*(XIP2-1.);
            DDSFH(2,II+3) = 0.125*XI0*XI1*XP*YP*(YIP2-1.0);
            DDSFH(3,II+3) = 0.0625*XP*YP*YIP0*XIP0*(YIP1-2.)*(XIP1-2.);
          II = I*4 + 1;
      end
  else
          NET=3*NPE;
           II = 1;
           for  I = 1:NPE
           XP    = XNODE(I,1);
           YP    = XNODE(I,2);
           XI0   = XI*XP;
           ETA0  = ETA*YP;
           XIP1  = XI0+1;
           ETAP1 = ETA0+1;
           XIM1  = XI0-1;
           ETAM1 = ETA0-1;
           XID   = 3.0+2.0*XI0+ETA0-3.0*XI*XI-ETA*ETA-2.0*XI/XP;
           ETAD  = 3.0+XI0+2.0*ETA0-XI*XI-3.0*ETA*ETA-2.0*ETA/YP;
           ETAXI = 4.0+2.0*(XI0+ETA0)-3.0*(XI*XI+ETA*ETA)-
2.0*(ETA/YP+XI/XP);
%
           SFH(II) = 0.125*XIP1*ETAP1*(2.0+XI0+ETA0-XI*XI-ETA*ETA);
           DSFH(1,II)    = 0.125*XP*ETAP1*XID;
           DSFH(2,II)    = 0.125*YP*XIP1*ETAD;
           DDSFH(1,II)   = 0.250*XP*ETAP1*(XP-3.0*XI-1.0/XP);
           DDSFH(2,II)   = 0.250*YP*XIP1*(YP-3.0*ETA-1.0/YP);
           DDSFH(3,II)   = 0.125*XP*YP*ETAXI;
%
           SFH(II+1)     = 0.125*XP*XIP1*XIP1*XIM1*ETAP1;
           DSFH(1,II+1)  = 0.125*XP*XP*ETAP1*(3.0*XI0-1.0)*XIP1;
           DSFH(2,II+1)  = 0.125*XP*YP*XIP1*XIP1*XIM1;
           DDSFH(1,II+1) = 0.250*XP*XP*XP*ETAP1*(3.0*XI0+1.0);
           DDSFH(2,II+1) = 0.0;
           DDSFH(3,II+1) = 0.125*XP*XP*YP*(3.0*XI0-1.0)*XIP1;
%
           SFH(II+2)     = 0.125*YP*XIP1*ETAP1*ETAP1*ETAM1;
           DSFH(1,II+2)  = 0.125*XP*YP*ETAP1*ETAP1*ETAM1;
           DSFH(2,II+2)  = 0.125*YP*YP*XIP1*(3.0*ETA0-1.0)*ETAP1;
           DDSFH(1,II+2) = 0.0;
           DDSFH(2,II+2) = 0.250*YP*YP*YP*XIP1*(3.0*ETA0+1.0);
           DDSFH(3,II+2) = 0.125*XP*YP*YP*(3.0*ETA0-1.0)*ETAP1;
         II = I*3 + 1;
end
end

        DDSF(1,1) =   0.0;
        DDSF(2,1) =   0.0;
        DDSF(3,1) =   0.250;
        DDSF(1,2) =   0.0;
        DDSF(2,2) =   0.0;
        DDSF(3,2) = - 0.250;
        DDSF(1,3) =   0.0;
        DDSF(2,3) =   0.0;
        DDSF(3,3) =   0.250;
        DDSF(1,4) =   0.0;
```

```
          DDSF(2,4) =   0.0;
          DDSF(3,4) = - 0.250;

          for  I = 1:2
          for J = 1:NET
          SUM = 0.0;
          for  K = 1:2
        SUM = SUM + GJINV(I,K)*DSFH(K,J);
    end
          GDSFH(I,J) = SUM;
      end
 end
          for I = 1:3
          for J = 1:2
          SUM = 0.0D0;
          for K = 1:NPE
          SUM = SUM + DDSF(I,K)*ELXY(K,J);
      end
          DJCB(I,J) = SUM;
      end
 end
%
          for  K = 1:3
          for J = 1:NET
          SUM = 0.0;
          for L = 1:2
          SUM = SUM + DJCB(K,L)*GDSFH(L,J);
      end
          DDSJ(K,J) = SUM;
      end
 end
%
%     Compute the jacobian of the transformation
%
          GGJ(1,1)=GJ(1,1)*GJ(1,1);
          GGJ(1,2)=GJ(1,2)*GJ(1,2);
          GGJ(1,3)=2.0*GJ(1,1)*GJ(1,2);
          GGJ(2,1)=GJ(2,1)*GJ(2,1);
          GGJ(2,2)=GJ(2,2)*GJ(2,2);
          GGJ(2,3)=2.0*GJ(2,1)*GJ(2,2);
          GGJ(3,1)=GJ(2,1)*GJ(1,1);
          GGJ(3,2)=GJ(2,2)*GJ(1,2);
          GGJ(3,3)=GJ(2,1)*GJ(1,2)+GJ(1,1)*GJ(2,2);

          GGINV=inv(GGJ); % CALL INVRSE(GGJ,GGINV);
%
          for I = 1:3
          for J = 1:NET
          SUM = 0.0;
         for K = 1:3
          SUM = SUM + GGINV(I,K)*(DDSFH(K,J)-DDSJ(K,J));
      end
          GDDSFH(I,J) = SUM;
      end
 end
end
```

```
function
[]=STRESS(ELD,ELXY,HT,Q0,YL,ISTR,ITYPE,LAYERS,NDF,NPE,NTHERM,ICOUNT,MAX
ELM,MAXGPT,RNXX,RNYY,RNXY,RMXX,RMYY,RMXY,RQX,RQY)
%     ------------------------------------------------------------------
%
%     Laminate stresses are computed in this subroutine
%     ------------------------------------------------------------------
%
RNXX=zeros(NEM,9);
RNYY=zeros(NEM,9);
RNXY=zeros(NEM,9);
RMXX=zeros(NEM,9);
RMYY=zeros(NEM,9);
RMXY=zeros(NEM,9);
RQX=zeros(NEM,9);
RQY=zeros(NEM,9);
ELD=ELU;
%
GAUSPT=[0.0 0.0 0.0 0.0 0.0;
             -0.57735027 0.57735027 0.0 0.0 0.0;
             -0.77459667 0.0 0.77459667 0.0 0.0;
             -0.86113631 -0.33998104 0.33998104 0.86113631 0.0;
             -0.90617984 -0.53846931 0.0 0.53846931 0.9061798];
GAUSPT=GAUSPT';

%


      %pi = 22/7;
      if ITYPE==0
         NET=3;
     else
         NET=4;
     end
      HT2=HT*HT;
      YL2=YL*YL;
      SCALE1=HT2/YL2/Q0;
      SCALE2=HT/YL/Q0;
%
    IC=1;
    for  IGP=1:ISTR  %700
%
    for JGP=1:ISTR   %600
%
    XI  = GAUSPT(IGP,ISTR);
      ETA = GAUSPT(JGP,ISTR);
SHPRCT; % CALL SHPRCT (NPE,XI,ETA,DET,ELXY,ITYPE)
      XG=0.0;
      YG=0.0;
%C      DO 10 I=1,NPE
%C      XG=XG+ELXY(I,1)*SF(I)
%C10    YG=YG+ELXY(I,2)*SF(I)
      EXX0=0.0;
      EYY0=0.0;
      EXY0=0.0;
```

```
      EXX1=0.0;
      EYY1=0.0;
      EXY1=0.0;
      EXZ0=0.0;
      EYZ0=0.0;
%
%     Compute the membrane strains
   for  I=1:NPE
      L=(I-1)*NDF+1;
      EXX0=EXX0+GDSF(1,I)*ELD(L);
      EYY0=EYY0+GDSF(2,I)*ELD(L+1);
      EXY0=EXY0+((GDSF(2,I)*ELD(L))+(GDSF(1,I)*ELD(L+1)));
end
%
%     Compute bending strains
%
      M=0;
      for I=1:NPE
      J=(I-1)*NDF+1;
    if ITYPE<=1
         for K=1:NET
         L=J+K+1;
         M=M+1;
         EXX1=EXX1-GDDSFH(1,M)*ELD(L);
         EYY1=EYY1-GDDSFH(2,M)*ELD(L);
         EXY1=EXY1-2.0*GDDSFH(3,M)*ELD(L);
     end
 else
         EXX1=EXX1+GDSF(1,I)*ELD(J+3);
         EYY1=EYY1+GDSF(2,I)*ELD(J+4);
         EXY1=EXY1+GDSF(2,I)*ELD(J+3)+GDSF(1,I)*ELD(J+4);
         EXZ0=EXZ0+SF(I)*ELD(J+3)+GDSF(1,I)*ELD(J+2);
         EYZ0=EYZ0+SF(I)*ELD(J+4)+GDSF(2,I)*ELD(J+2);
     end %if
 end
%
 for L=1:LAYERS
      K=LAYERS-L+1;
      ANGLE = pi*THETA(K)/180.0;
QBARSTF;  % CALL QBARSTF(K,ANGLE,NTHERM)
      ZK1=Z(K);
      ZP1=Z(K+1);
%

SGMXXB=(QBAR(1,1)*(EXX0+EXX1*ZK1)+QBAR(1,2)*(EYY0+EYY1*ZK1)+QBAR(1,3)*(
EXY0+EXY1*ZK1))*SCALE1;

SGMXXT=(QBAR(1,1)*(EXX0+EXX1*ZP1)+QBAR(1,2)*(EYY0+EYY1*ZP1)+QBAR(1,3)*(
EXY0+EXY1*ZP1))*SCALE1;

SGMYYB=(QBAR(1,2)*(EXX0+EXX1*ZK1)+QBAR(2,2)*(EYY0+EYY1*ZK1)+QBAR(2,3)*(
EXY0+EXY1*ZK1))*SCALE1;

SGMYYT=(QBAR(1,2)*(EXX0+EXX1*ZP1)+QBAR(2,2)*(EYY0+EYY1*ZP1)+QBAR(2,3)*(
EXY0+EXY1*ZP1))*SCALE1;
```

```
SGMXYB=(QBAR(1,3)*(EXX0+EXX1*ZK1)+QBAR(2,3)*(EYY0+EYY1*ZK1)+QBAR(3,3)*(
EXY0+EXY1*ZK1))*SCALE1;

SGMXYT=(QBAR(1,3)*(EXX0+EXX1*ZP1)+QBAR(2,3)*(EYY0+EYY1*ZP1)+QBAR(3,3)*(
EXY0+EXY1*ZP1))*SCALE1;
%     Transverse shear stresses for shear deformation theories
%     First-Order theory
       if ITYPE>1
         SGMYZT=(QBAR(4,4)*EYZ0+QBAR(4,5)*EXZ0)*SCALE2;
         SGMXZT=(QBAR(4,5)*EYZ0+QBAR(5,5)*EXZ0)*SCALE2;
     else
         SGMYZT=0;
         SGMXZT=0;
     end
%*****COMPUTATION OF RESULTANT FORCES AND MOMENTS FOR
LAMINATE**************
%
%   COMPUTATION OF Z
    ZP12=ZP1^2;
    ZK12=ZK1^2;
    ZP13=ZP1^3;
    ZK13=ZK1^3;
    ZINT1=ZP1-ZK1;
    ZINT2=0.5*(ZP12-ZK12);
    ZINT3=(ZP13-ZK13)/3.0;
%
%   COMPUTATION OF FORCE RESULTANTS
%

RNXX(ICOUNT,IC)=RNXX(ICOUNT,IC)+(QBAR(1,1)*(EXX0*ZINT1+EXX1*ZINT2)+QBAR
(1,2)*(EYY0*ZINT1+EYY1*ZINT2)+QBAR(1,3)*(EXY0*ZINT1+EXY1*ZINT2))*SCALE1
;
%

RNYY(ICOUNT,IC)=RNYY(ICOUNT,IC)+(QBAR(1,2)*(EXX0*ZINT1+EXX1*ZINT2)+QBAR
(2,2)*(EYY0*ZINT1+EYY1*ZINT2)+QBAR(2,3)*(EXY0*ZINT1+EXY1*ZINT2))*SCALE1
;
%

RNXY(ICOUNT,IC)=RNXY(ICOUNT,IC)+(QBAR(1,3)*(EXX0*ZINT1+EXX1*ZINT2)+QBAR
(2,3)*(EYY0*ZINT1+EYY1*ZINT2)+QBAR(3,3)*(EXY0*ZINT1+EXY1*ZINT2))*SCALE1
;
%
%    COMPUTATION OF MOMENT RESULTANTS
%


RMXX(ICOUNT,IC)=RMXX(ICOUNT,IC)+(QBAR(1,1)*(EXX0*ZINT2+EXX1*ZINT3)+QBAR
(1,2)*(EYY0*ZINT2+EYY1*ZINT3)+QBAR(1,3)*(EXY0*ZINT2+EXY1*ZINT3))*SCALE1
;
%

RMYY(ICOUNT,IC)=RMYY(ICOUNT,IC)+(QBAR(1,2)*(EXX0*ZINT2+EXX1*ZINT3)+QBAR
(2,2)*(EYY0*ZINT2+EYY1*ZINT3)+QBAR(2,3)*(EXY0*ZINT2+EXY1*ZINT3))*SCALE1
;
%
```

```
RMXY(ICOUNT,IC)=RMXY(ICOUNT,IC)+(QBAR(1,3)*(EXX0*ZINT2+EXX1*ZINT3)+QBAR
(2,3)*(EYY0*ZINT2+EYY1*ZINT3)+QBAR(3,3)*(EXY0*ZINT2+EXY1*ZINT3))*SCALE1
;
%
%   COMPUTATION OF RESULTANTS FOR THE FIRST ORDER THEORY
    if ITYPE>1

RQY(ICOUNT,IC)=RQY(ICOUNT,IC)+(QBAR(4,4)*EYZ0*ZINT1+QBAR(4,5)*EXZ0*ZINT
1)*SCALE2;
%

RQX(ICOUNT,IC)=RQX(ICOUNT,IC)+(QBAR(4,5)*EYZ0*ZINT1+QBAR(5,5)*EXZ0*ZINT
1)*SCALE2;
  end %if
%
% CHANGE HAS BEEN MADE JUST PRINT XX AND YY AT TOP OF A LAYER TO FILE
%
    if N==1 | N ==2
    fprintf(LOUT,'\n\t %d\t%e\t%e\t%e\t%e\t%e\t%e\t%e\t%e
',K,SGMXXT,SGMXXB,SGMYYT,SGMYYB,SGMXYT,SGMXYB,SGMXZT,SGMYZT);
end
    end
    IC=IC+1;
end

end

if N==1 | N ==2
fprintf(LOUT,'\n_____________________________________________________________
_____________________________________\n'
end
```

yprime_plate.m

```
function ydot = yprime_plate(t,y,flag,A_mat,F_vec)
switch flag
case ''
    force = F_vec*sin(10*pi*t);
    ydot = A_mat*y + force;

case 'jacobian'
    ydot = A_mat;
end
```

Example input file circle.inp

```
Analysis of Circular plate with piezo
0  1  0   0 1
1  1  2
0
0.75e-3
1
1 0.26e-3 0.0 45.0
150e9  150e9  7.1e9 7.1e9 2.5e9 0.3
1600
63e9  63e9  24.8e9 24.8e9 24.8e9 0.3
```

```
7600
-254e-12 -254e-12 0 0 0 0 0
500e-3  500e-3
1
1  1
0.0  0.0  0.0
1  4  0  1  1
3  3  2
180 216
1   2   8   7
2   3   9   8
3   4   10  9
4   5   11  10
5   6   12  11
7   8   14  13
8   9   15  14
9   10  16  15
10  11  17  16
11  12  18  17
13  14  20  19
14  15  21  20
15  16  22  21
16  17  23  22
17  18  24  23
19  20  26  25
20  21  27  26
21  22  28  27
22  23  29  28
23  24  30  29
25  26  32  31
26  27  33  32
27  28  34  33
28  29  35  34
29  30  36  35
31  32  38  37
32  33  39  38
33  34  40  39
34  35  41  40
35  36  42  41
37  38  44  43
38  39  45  44
39  40  46  45
40  41  47  46
41  42  48  47
43  44  50  49
44  45  51  50
45  46  52  51
46  47  53  52
47  48  54  53
49  50  56  55
50  51  57  56
51  52  58  57
52  53  59  58
53  54  60  59
55  56  62  61
56  57  63  62
57  58  64  63
```

58	59	65	64
59	60	66	65
61	62	68	67
62	63	69	68
63	64	70	69
64	65	71	70
65	66	72	71
67	68	74	73
68	69	75	74
69	70	76	75
70	71	77	76
71	72	78	77
73	74	80	79
74	75	81	80
75	76	82	81
76	77	83	82
77	78	84	83
79	80	86	85
80	81	87	86
81	82	88	87
82	83	89	88
83	84	90	89
85	86	92	91
86	87	93	92
87	88	94	93
88	89	95	94
89	90	96	95
91	92	98	97
92	93	99	98
93	94	100	99
94	95	101	100
95	96	102	101
97	98	104	103
98	99	105	104
99	100	106	105
100	101	107	106
101	102	108	107
103	104	110	109
104	105	111	110
105	106	112	111
106	107	113	112
107	108	114	113
109	110	116	115
110	111	117	116
111	112	118	117
112	113	119	118
113	114	120	119
115	116	122	121
116	117	123	122
117	118	124	123
118	119	125	124
119	120	126	125
121	122	128	127
122	123	129	128
123	124	130	129
124	125	131	130
125	126	132	131

```
127 128 134 133
128 129 135 134
129 130 136 135
130 131 137 136
131 132 138 137
133 134 140 139
134 135 141 140
135 136 142 141
136 137 143 142
137 138 144 143
139 140 146 145
140 141 147 146
141 142 148 147
142 143 149 148
143 144 150 149
145 146 152 151
146 147 153 152
147 148 154 153
148 149 155 154
149 150 156 155
151 152 158 157
152 153 159 158
153 154 160 159
154 155 161 160
155 156 162 161
157 158 164 163
158 159 165 164
159 160 166 165
160 161 167 166
161 162 168 167
163 164 170 169
164 165 171 170
165 166 172 171
166 167 173 172
167 168 174 173
169 170 176 175
170 171 177 176
171 172 178 177
172 173 179 178
173 174 180 179
175 176 182 181
176 177 183 182
177 178 184 183
178 179 185 184
179 180 186 185
181 182 188 187
182 183 189 188
183 184 190 189
184 185 191 190
185 186 192 191
187 188 194 193
188 189 195 194
189 190 196 195
190 191 197 196
191 192 198 197
193 194 200 199
194 195 201 200
```

```
195 196 202 201
196 197 203 202
197 198 204 203
199 200 206 205
200 201 207 206
201 202 208 207
202 203 209 208
203 204 210 209
205 206 212 211
206 207 213 212
207 208 214 213
208 209 215 214
209 210 216 215
211 212 2   1
212 213 3   2
213 214 4   3
214 215 5   4
215 216 6   5
0.5 0.449999988
0.5 0.360000014
0.5 0.270000011
0.5 0.180000007
0.5 0.090000004
0.5 0
0.50868243  0.450759619
0.524310768 0.362126917
0.539939106 0.273494214
0.555567443 0.184861526
0.571195781 0.096228823
0.586824059 0.007596124
0.51710099  0.453015357
0.547882795 0.368443042
0.57866466  0.283870697
0.609446466 0.199298367
0.640228271 0.114726022
0.671010077 0.03015369
0.524999976 0.456698716
0.569999993 0.378756434
0.61500001  0.300814152
0.660000026 0.22287187
0.704999983 0.144929588
0.75    0.066987298
0.532139361 0.461697787
0.589990258 0.39275378
0.647841156 0.323809773
0.705692053 0.254865766
0.763542891 0.185921773
0.821393788 0.116977781
0.538302243 0.467860609
0.60724622  0.410009742
0.676190197 0.352158844
0.745134234 0.294307977
0.814078212 0.23645708
0.883022249 0.178606197
0.543301284 0.474999994
0.621243536 0.430000007
0.699185848 0.38499999
```

```
0.7771281   0.340000004
0.855070412 0.294999987
0.933012724 0.25
0.546984613 0.48289898
0.631556988 0.452117175
0.716129303 0.421335369
0.800701618 0.390553564
0.885273993 0.359771729
0.969846308 0.328989923
0.54924041  0.4913176
0.637873113 0.475689262
0.726505756 0.460060924
0.815138459 0.444432586
0.903771162 0.428804249
0.992403865 0.413175911
0.550000012 0.5
0.639999986 0.5
0.730000019 0.5
0.819999993 0.5
0.910000026 0.5
1   0.5
0.54924041  0.50868243
0.637873113 0.524310768
0.726505756 0.539939106
0.815138459 0.555567443
0.903771162 0.571195781
0.992403865 0.586824059
0.546984613 0.51710099
0.631556988 0.547882795
0.716129303 0.57866466
0.800701618 0.609446466
0.885273993 0.640228271
0.969846308 0.671010077
0.543301284 0.524999976
0.621243536 0.569999993
0.699185848 0.61500001
0.7771281   0.660000026
0.855070412 0.704999983
0.933012724 0.75
0.538302243 0.532139361
0.60724622  0.589990258
0.676190197 0.647841156
0.745134234 0.705692053
0.814078212 0.763542891
0.883022249 0.821393788
0.532139361 0.538302243
0.589990258 0.60724622
0.647841156 0.676190197
0.705692053 0.745134234
0.763542891 0.814078212
0.821393788 0.883022249
0.524999976 0.543301284
0.569999993 0.621243536
0.61500001  0.699185848
0.660000026 0.7771281
0.704999983 0.855070412
0.75    0.933012724
```

```
0.51710099  0.546984613
0.547882795 0.631556988
0.57866466  0.716129303
0.609446466 0.800701618
0.640228271 0.885273993
0.671010077 0.969846308
0.50868243  0.54924041
0.524310768 0.637873113
0.539939106 0.726505756
0.555567443 0.815138459
0.571195781 0.903771162
0.586824059 0.992403865
0.5 0.550000012
0.5 0.639999986
0.5 0.730000019
0.5 0.819999993
0.5 0.910000026
0.5 1
0.4913176   0.54924041
0.475689262 0.637873113
0.460060924 0.726505756
0.444432586 0.815138459
0.428804249 0.903771162
0.413175911 0.992403865
0.48289898  0.546984613
0.452117175 0.631556988
0.421335369 0.716129303
0.390553564 0.800701618
0.359771729 0.885273993
0.328989923 0.969846308
0.474999994 0.543301284
0.430000007 0.621243536
0.38499999  0.699185848
0.340000004 0.7771281
0.294999987 0.855070412
0.25    0.933012724
0.467860609 0.538302243
0.410009742 0.60724622
0.352158844 0.676190197
0.294307977 0.745134234
0.23645708  0.814078212
0.178606197 0.883022249
0.461697787 0.532139361
0.39275378  0.589990258
0.323809773 0.647841156
0.254865766 0.705692053
0.185921773 0.763542891
0.116977781 0.821393788
0.456698716 0.524999976
0.378756434 0.569999993
0.300814152 0.61500001
0.22287187  0.660000026
0.144929588 0.704999983
0.066987298 0.75
0.453015357 0.51710099
0.368443042 0.547882795
0.283870697 0.57866466
```

```
0.199298367 0.609446466
0.114726022 0.640228271
0.03015369  0.671010077
0.450759619 0.50868243
0.362126917 0.524310768
0.273494214 0.539939106
0.184861526 0.555567443
0.096228823 0.571195781
0.007596124 0.586824059
0.449999988 0.5
0.360000014 0.5
0.270000011 0.5
0.180000007 0.5
0.090000004 0.5
0   0.5
0.450759619 0.4913176
0.362126917 0.475689262
0.273494214 0.460060924
0.184861526 0.444432586
0.096228823 0.428804249
0.007596124 0.413175911
0.453015357 0.48289898
0.368443042 0.452117175
0.283870697 0.421335369
0.199298367 0.390553564
0.114726022 0.359771729
0.03015369  0.328989923
0.456698716 0.474999994
0.378756434 0.430000007
0.300814152 0.38499999
0.22287187  0.340000004
0.144929588 0.294999987
0.066987298 0.25
0.461697787 0.467860609
0.39275378  0.410009742
0.323809773 0.352158844
0.254865766 0.294307977
0.185921773 0.23645708
0.116977781 0.178606197
0.467860609 0.461697787
0.410009742 0.39275378
0.352158844 0.323809773
0.294307977 0.254865766
0.23645708  0.185921773
0.178606197 0.116977781
0.474999994 0.456698716
0.430000007 0.378756434
0.38499999  0.300814152
0.340000004 0.22287187
0.294999987 0.144929588
0.25    0.066987298
0.48289898  0.453015357
0.452117175 0.368443042
0.421335369 0.283870697
0.390553564 0.199298367
0.359771729 0.114726022
0.328989923 0.03015369
```

```
0.4913176   0.450759619
0.475689262 0.362126917
0.460060924 0.273494214
0.444432586 0.184861526
0.428804249 0.096228823
0.413175911 0.007596124
0
180
1 1
1 2
1 3
1 4
1 5
7 1
7 2
7 3
7 4
7 5
13 1
13 2
13 3
13 4
13 5
19 1
19 2
19 3
19 4
19 5
25 1
25 2
25 3
25 4
25 5
31 1
31 2
31 3
31 4
31 5
37 1
37 2
37 3
37 4
37 5
43 1
43 2
43 3
43 4
43 5
49 1
49 2
49 3
49 4
49 5
55 1
55 2
55 3
55 4
```

```
55 5
61 1
61 2
61 3
61 4
61 5
67 1
67 2
67 3
67 4
67 5
73 1
73 2
73 3
73 4
73 5
79 1
79 2
79 3
79 4
79 5
85 1
85 2
85 3
85 4
85 5
91 1
91 2
91 3
91 4
91 5
97 1
97 2
97 3
97 4
97 5
103 1
103 2
103 3
103 4
103 5
109 1
109 2
109 3
109 4
109 5
115 1
115 2
115 3
115 4
115 5
121 1
121 2
121 3
121 4
121 5
127 1
```

127 2
127 3
127 4
127 5
133 1
133 2
133 3
133 4
133 5
139 1
139 2
139 3
139 4
139 5
145 1
145 2
145 3
145 4
145 5
151 1
151 2
151 3
151 4
151 5
157 1
157 2
157 3
157 4
157 5
163 1
163 2
163 3
163 4
163 5
169 1
169 2
169 3
169 4
169 5
175 1
175 2
175 3
175 4
175 5
181 1
181 2
181 3
181 4
181 5
187 1
187 2
187 3
187 4
187 5
193 1
193 2
193 3

```
193 4
193 5
199 1
199 2
199 3
199 4
199 5
205 1
205 2
205 3
205 4
205 5
211 1
211 2
211 3
211 4
211 5
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
16
2 3 42 43 47 48 87 88 92 93 132 133 137 138 177 178
```

```
Analysis of Circular plate with piezo
0  1  0   0 1
1  1  2
0
0.75e-3
1
1 0.26e-3 0.0 45.0
150e9  150e9  7.1e9 7.1e9 2.5e9 0.3
1600
63e9  63e9  24.8e9 24.8e9 24.8e9 0.3
7600
-254e-12 -254e-12 0 0 0 0 0
500e-3  500e-3
1
1  1
0.0  0.0  0.0
1  4  0  1  1
3  3  2
180 216
1   2   8   7
2   3   9   8
3   4   10  9
4   5   11  10
5   6   12  11
7   8   14  13
8   9   15  14
9   10  16  15
10  11  17  16
```

11	12	18	17
13	14	20	19
14	15	21	20
15	16	22	21
16	17	23	22
17	18	24	23
19	20	26	25
20	21	27	26
21	22	28	27
22	23	29	28
23	24	30	29
25	26	32	31
26	27	33	32
27	28	34	33
28	29	35	34
29	30	36	35
31	32	38	37
32	33	39	38
33	34	40	39
34	35	41	40
35	36	42	41
37	38	44	43
38	39	45	44
39	40	46	45
40	41	47	46
41	42	48	47
43	44	50	49
44	45	51	50
45	46	52	51
46	47	53	52
47	48	54	53
49	50	56	55
50	51	57	56
51	52	58	57
52	53	59	58
53	54	60	59
55	56	62	61
56	57	63	62
57	58	64	63
58	59	65	64
59	60	66	65
61	62	68	67
62	63	69	68
63	64	70	69
64	65	71	70
65	66	72	71
67	68	74	73
68	69	75	74
69	70	76	75
70	71	77	76
71	72	78	77
73	74	80	79
74	75	81	80
75	76	82	81
76	77	83	82
77	78	84	83
79	80	86	85

```
80  81  87  86
81  82  88  87
82  83  89  88
83  84  90  89
85  86  92  91
86  87  93  92
87  88  94  93
88  89  95  94
89  90  96  95
91  92  98  97
92  93  99  98
93  94  100 99
94  95  101 100
95  96  102 101
97  98  104 103
98  99  105 104
99  100 106 105
100 101 107 106
101 102 108 107
103 104 110 109
104 105 111 110
105 106 112 111
106 107 113 112
107 108 114 113
109 110 116 115
110 111 117 116
111 112 118 117
112 113 119 118
113 114 120 119
115 116 122 121
116 117 123 122
117 118 124 123
118 119 125 124
119 120 126 125
121 122 128 127
122 123 129 128
123 124 130 129
124 125 131 130
125 126 132 131
127 128 134 133
128 129 135 134
129 130 136 135
130 131 137 136
131 132 138 137
133 134 140 139
134 135 141 140
135 136 142 141
136 137 143 142
137 138 144 143
139 140 146 145
140 141 147 146
141 142 148 147
142 143 149 148
143 144 150 149
145 146 152 151
146 147 153 152
147 148 154 153
```

```
148 149 155 154
149 150 156 155
151 152 158 157
152 153 159 158
153 154 160 159
154 155 161 160
155 156 162 161
157 158 164 163
158 159 165 164
159 160 166 165
160 161 167 166
161 162 168 167
163 164 170 169
164 165 171 170
165 166 172 171
166 167 173 172
167 168 174 173
169 170 176 175
170 171 177 176
171 172 178 177
172 173 179 178
173 174 180 179
175 176 182 181
176 177 183 182
177 178 184 183
178 179 185 184
179 180 186 185
181 182 188 187
182 183 189 188
183 184 190 189
184 185 191 190
185 186 192 191
187 188 194 193
188 189 195 194
189 190 196 195
190 191 197 196
191 192 198 197
193 194 200 199
194 195 201 200
195 196 202 201
196 197 203 202
197 198 204 203
199 200 206 205
200 201 207 206
201 202 208 207
202 203 209 208
203 204 210 209
205 206 212 211
206 207 213 212
207 208 214 213
208 209 215 214
209 210 216 215
211 212 2   1
212 213 3   2
213 214 4   3
214 215 5   4
215 216 6   5
```

```
0.5 0.449999988
0.5 0.360000014
0.5 0.270000011
0.5 0.180000007
0.5 0.090000004
0.5 0
0.50868243  0.450759619
0.524310768 0.362126917
0.539939106 0.273494214
0.555567443 0.184861526
0.571195781 0.096228823
0.586824059 0.007596124
0.51710099  0.453015357
0.547882795 0.368443042
0.57866466  0.283870697
0.609446466 0.199298367
0.640228271 0.114726022
0.671010077 0.03015369
0.524999976 0.456698716
0.569999993 0.378756434
0.61500001  0.300814152
0.660000026 0.22287187
0.704999983 0.144929588
0.75    0.066987298
0.532139361 0.461697787
0.589990258 0.39275378
0.647841156 0.323809773
0.705692053 0.254865766
0.763542891 0.185921773
0.821393788 0.116977781
0.538302243 0.467860609
0.60724622  0.410009742
0.676190197 0.352158844
0.745134234 0.294307977
0.814078212 0.23645708
0.883022249 0.178606197
0.543301284 0.474999994
0.621243536 0.430000007
0.699185848 0.38499999
0.7771281   0.340000004
0.855070412 0.294999987
0.933012724 0.25
0.546984613 0.48289898
0.631556988 0.452117175
0.716129303 0.421335369
0.800701618 0.390553564
0.885273993 0.359771729
0.969846308 0.328989923
0.54924041  0.4913176
0.637873113 0.475689262
0.726505756 0.460060924
0.815138459 0.444432586
0.903771162 0.428804249
0.992403865 0.413175911
0.550000012 0.5
0.639999986 0.5
0.730000019 0.5
```

```
0.819999993 0.5
0.910000026 0.5
1   0.5
0.54924041  0.50868243
0.637873113 0.524310768
0.726505756 0.539939106
0.815138459 0.555567443
0.903771162 0.571195781
0.992403865 0.586824059
0.546984613 0.51710099
0.631556988 0.547882795
0.716129303 0.57866466
0.800701618 0.609446466
0.885273993 0.640228271
0.969846308 0.671010077
0.543301284 0.524999976
0.621243536 0.569999993
0.699185848 0.61500001
0.7771281   0.660000026
0.855070412 0.704999983
0.933012724 0.75
0.538302243 0.532139361
0.60724622  0.589990258
0.676190197 0.647841156
0.745134234 0.705692053
0.814078212 0.763542891
0.883022249 0.821393788
0.532139361 0.538302243
0.589990258 0.60724622
0.647841156 0.676190197
0.705692053 0.745134234
0.763542891 0.814078212
0.821393788 0.883022249
0.524999976 0.543301284
0.569999993 0.621243536
0.61500001  0.699185848
0.660000026 0.7771281
0.704999983 0.855070412
0.75    0.933012724
0.51710099  0.546984613
0.547882795 0.631556988
0.57866466  0.716129303
0.609446466 0.800701618
0.640228271 0.885273993
0.671010077 0.969846308
0.50868243  0.54924041
0.524310768 0.637873113
0.539939106 0.726505756
0.555567443 0.815138459
0.571195781 0.903771162
0.586824059 0.992403865
0.5 0.550000012
0.5 0.639999986
0.5 0.730000019
0.5 0.819999993
0.5 0.910000026
0.5 1
```

```
0.4913176   0.54924041
0.475689262 0.637873113
0.460060924 0.726505756
0.444432586 0.815138459
0.428804249 0.903771162
0.413175911 0.992403865
0.48289898  0.546984613
0.452117175 0.631556988
0.421335369 0.716129303
0.390553564 0.800701618
0.359771729 0.885273993
0.328989923 0.969846308
0.474999994 0.543301284
0.430000007 0.621243536
0.38499999  0.699185848
0.340000004 0.7771281
0.294999987 0.855070412
0.25    0.933012724
0.467860609 0.538302243
0.410009742 0.60724622
0.352158844 0.676190197
0.294307977 0.745134234
0.23645708  0.814078212
0.178606197 0.883022249
0.461697787 0.532139361
0.39275378  0.589990258
0.323809773 0.647841156
0.254865766 0.705692053
0.185921773 0.763542891
0.116977781 0.821393788
0.456698716 0.524999976
0.378756434 0.569999993
0.300814152 0.61500001
0.22287187  0.660000026
0.144929588 0.704999983
0.066987298 0.75
0.453015357 0.51710099
0.368443042 0.547882795
0.283870697 0.57866466
0.199298367 0.609446466
0.114726022 0.640228271
0.03015369  0.671010077
0.450759619 0.50868243
0.362126917 0.524310768
0.273494214 0.539939106
0.184861526 0.555567443
0.096228823 0.571195781
0.007596124 0.586824059
0.449999988 0.5
0.360000014 0.5
0.270000011 0.5
0.180000007 0.5
0.090000004 0.5
0   0.5
0.450759619 0.4913176
0.362126917 0.475689262
0.273494214 0.460060924
```

```
0.184861526 0.444432586
0.096228823 0.428804249
0.007596124 0.413175911
0.453015357 0.48289898
0.368443042 0.452117175
0.283870697 0.421335369
0.199298367 0.390553564
0.114726022 0.359771729
0.03015369  0.328989923
0.456698716 0.474999994
0.378756434 0.430000007
0.300814152 0.38499999
0.22287187  0.340000004
0.144929588 0.294999987
0.066987298 0.25
0.461697787 0.467860609
0.39275378  0.410009742
0.323809773 0.352158844
0.254865766 0.294307977
0.185921773 0.23645708
0.116977781 0.178606197
0.467860609 0.461697787
0.410009742 0.39275378
0.352158844 0.323809773
0.294307977 0.254865766
0.23645708  0.185921773
0.178606197 0.116977781
0.474999994 0.456698716
0.430000007 0.378756434
0.38499999  0.300814152
0.340000004 0.22287187
0.294999987 0.144929588
0.25    0.066987298
0.48289898  0.453015357
0.452117175 0.368443042
0.421335369 0.283870697
0.390553564 0.199298367
0.359771729 0.114726022
0.328989923 0.03015369
0.4913176   0.450759619
0.475689262 0.362126917
0.460060924 0.273494214
0.444432586 0.184861526
0.428804249 0.096228823
0.413175911 0.007596124
0
180
1 1
1 2
1 3
1 4
1 5
7 1
7 2
7 3
7 4
7 5
```

```
13 1
13 2
13 3
13 4
13 5
19 1
19 2
19 3
19 4
19 5
25 1
25 2
25 3
25 4
25 5
31 1
31 2
31 3
31 4
31 5
37 1
37 2
37 3
37 4
37 5
43 1
43 2
43 3
43 4
43 5
49 1
49 2
49 3
49 4
49 5
55 1
55 2
55 3
55 4
55 5
61 1
61 2
61 3
61 4
61 5
67 1
67 2
67 3
67 4
67 5
73 1
73 2
73 3
73 4
73 5
79 1
79 2
```

79 3
79 4
79 5
85 1
85 2
85 3
85 4
85 5
91 1
91 2
91 3
91 4
91 5
97 1
97 2
97 3
97 4
97 5
103 1
103 2
103 3
103 4
103 5
109 1
109 2
109 3
109 4
109 5
115 1
115 2
115 3
115 4
115 5
121 1
121 2
121 3
121 4
121 5
127 1
127 2
127 3
127 4
127 5
133 1
133 2
133 3
133 4
133 5
139 1
139 2
139 3
139 4
139 5
145 1
145 2
145 3
145 4

145 5
151 1
151 2
151 3
151 4
151 5
157 1
157 2
157 3
157 4
157 5
163 1
163 2
163 3
163 4
163 5
169 1
169 2
169 3
169 4
169 5
175 1
175 2
175 3
175 4
175 5
181 1
181 2
181 3
181 4
181 5
187 1
187 2
187 3
187 4
187 5
193 1
193 2
193 3
193 4
193 5
199 1
199 2
199 3
199 4
199 5
205 1
205 2
205 3
205 4
205 5
211 1
211 2
211 3
211 4
211 5

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
16
2 3 42 43 47 48 87 88 92 93 132 133 137 138 177 178
```

PART2 BEAM

1. Run the BEAM.m
2. all the required input information in in the same file (change the values for different problems)
3. Uncomment the required sections, by default the code finds the optimal location of one actuator on a cantilever and plot the static deformations

BEAM.m

```
% ------------------------------------------------------------%
% Beam with piezoelctric actuators
% bc=1 -Simply supported
% bc=2 -fixed-free
%-------------------------------------------------------------%
clc
clear all
nel=10;            % number of elements
nnel=2;           % number of nodes per element
ndof=2;           % number of dofs per node
nnode=(nnel-1)*nel+1;   % total number of nodes in system
sdof=nnode*ndof; % total system dofs
bc=2;              %bc=1 Simply supported beam  % bc=2%Fixed-free beam

el=79e9;          % elastic modulus
tleng=1;         % length of the beam
b=0.05; h=0.01;            % bearth and thickness of the beam
xi=(b*h^3)/12;          % moment of inertia of cross-section
leng=tleng/nel;      % element length of equal size
area=b*h;           % cross-sectional area of the beam
rho=2500;            % mass density
ipt=1;            % option for mass matrix
q0=0;             % distributed load
% m0=0;             % moment
% pl=0;             % point load
vol=0;            %Voltage
b=0.05; hp=0.01; tp=2e-4; d31=-254e-12;
Ep=63e9;           % elastic modulus of piezo

% el=70e9;          % elastic modulus
% xi=1.25e-12;          % moment of inertia of cross-section
% tleng=0.2;         % length of a half of the beam
% leng=tleng/nel;      % element length of equal size
% area=1.5e-5;           % cross-sectional area of the beam
% rho=1;            % mass density
% ipt=1;            % option for mass matrix
% q0=2;             % distributed load
% % m0=0;             % moment
% % pl=0;             % point load
% vol=-80;            %Voltage
% b=0.015; hp=1e-3; tp=0.2e-3; d31=-254e-12;
% Ep=63e9;           % elastic modulus of piezo

% el=100e9;          % elastic modulus
% xi=1/12;          % moment of inertia of cross-section
% tleng=1;         % length of a half of the beam
```

```
% b=0.02; h=0.02;           % bearth and thickness of the beam
% xi=(b*h^3)/12;         % moment of inertia of cross-section
% leng=tleng/nel;     % element length of equal size
% area=b*h;          % cross-sectional area of the beam
% rho=1000;           % mass density
% ipt=1;           % option for mass matrix
% q0=0;            % distributed load
% % m0=0;            % moment
% % pl=0;            % point load
% vol=-80;           %Voltage
% b=0.02; hp=0.01; tp=0.001; d31=-254e-12;
% Ep=63e9;          % elastic modulus of piezo

bcdof(1)=1;      % first dof (deflection at left end) is constrained
bcval(1)=0;      % whose described value is 0
if bc==1
bcdof(2)=sdof-1;     % deflection at right end  is constrained (Simply
Supported)
elseif bc==2;
bcdof(2)=2;     % Rotation at left end  is constrained (Fixed-free)
end
bcval(2)=0;      % whose described value is 0

i=1:nel; piezo(i)=0;
%Piezoelectric Moment due to the voltage applied
% S Yang and B Ngoi, AIAA, Vol 38, No 12, pp 2292-2298
% mp=-d31*b*Ep*(hp+tp)*vol;
mp=-d31*b*Ep*(hp+tp);

ff=zeros(sdof,1);     % initialization of system force vector
kk=zeros(sdof,sdof);  % initialization of system matrix
mm=zeros(sdof,sdof);  % initialization of system matrix
index=zeros(nnel*ndof,1);  % initialization of index vector

%Forming FE matrices
for iel=1:nel    % loop for the total number of elements
index=feeldof1(iel,nnel,ndof);  % extract system dofs associated with
element
[k,m]=febeam1(el,xi,leng,area,rho,ipt); % compute element stiffness
matrix
f=feload(leng,q0,'l'); % compute element stiffness matrix - for
distributed

% % loading
% if q0>0
%     f=feload(leng,q0,'l');
% else
%     f=feload(leng,0,'l');
% end
% if piezo(iel)==1
%     fp=feload(leng,mp,'p');
% else
%     fp=feload(leng,0,'p');
% end
% f=f+fp;              %element force matrix

% assemble each element matrix into system
```

```
[kk,ff]=feasmbl2(kk,ff,k,f,index);
mm=feasmbl1(mm,m,index);
end

% apply the boundary conditions
[kks,ffs]=feaplyc2(kk,ff,bcdof,bcval);  %static
[kkd,mmd]=feaplycs(kk,mm,bcdof);        %eigen
[kkk,mmm,fff]=aplybcdyn(bcdof,kk,mm,ff);%dyn

%Eigen  values
[V1,D1]=eig(kkk,mmm);
nfreq=sqrt(D1);

%First order system
C=0.2*mmm+0.005*kkk;  % Alpha=0.2, Beta=0.005
% C=0;
inv_m=inv(mmm);
A_mat = [zeros(size(kkk)) eye(size(kkk))
         -inv_m*kkk     -inv_m*C];
C_mat=[eye(size(kkk)), zeros(size(kkk))];
F_mat= [zeros(length(kkk),1)
inv_m*fff'];
Dsys=eig(A_mat);

% Modal Analysis
[Omega,Phi,ModF]=femodal(mmm,kkk,fff');
%Model Coordinate
kkk_m=Phi'*kkk*Phi;
mmm_m=Phi'*mmm*Phi;
%Eigen  values
[V2,D2]=eig(kkk_m,mmm_m);
nfreq_m=sqrt(D2);

%First order system
% C=0.2*mmm_m+0.0004*kkk_m;  % Alpha=0.2, Beta=0.005
C=0;
inv_m=inv(mmm_m);
A_mat_m = [zeros(size(kkk_m)) eye(size(kkk_m))
         -inv_m*kkk_m     -inv_m*C];
% C_mat_m=[eye(size(kkk_m)), zeros(size(kkk_m))];
fff=Phi'*fff';  %Modal Coordinate
F_mat_m= [zeros(size(kkk_m),1)
inv_m*fff];
Dsys_m=eig(A_mat_m);

C_mat_m=[eye(size(Phi))*Phi'*Phi', zeros(size(Phi))];

% %Forming Bact
for ielp=1:nel
fp=zeros(sdof,1);     % initialization of system force vector
ffp=zeros(sdof-2,1);     % initialization of system force vector
index=feeldof1(ielp,nnel,ndof);  % extract system dofs associated with
element
f=feload(leng,mp,'p');
fp=feasmblf(fp,f,index);
%applying bc
[kkp,ffp]=feaplyc2(kk,ff,bcdof,bcval);  %static
```

```
[kkk,mmm,ffp]=aplybcdyn(bcdof,kk,mm,fp);%dyn
% for i=1:sdof-2
%         if bc==2
%             ffp(i)=fp(i+2);
%         end
%         if bc==1
%             if i<sdof-1
%                 ffp(i)=fp(i+1);
%             else
%                 ffp(i)=fp(sdof);
%             end
%         end
% end
ffp=Phi'*ffp';  %Modal Coordinate
% ffp=ffp';  %Modal Coordinate
B_vec = [zeros(length(ffp),1)
inv_m*ffp];
Bact(:,ielp)=[B_vec];
end


% ttt=0:1/1947:1;
% sss=ss(A_mat_m,Bact(:,2),C_mat_m,0);
% figure(1)
% ires=impulse(sss);
% plot(ires(:,16))
% plot(ires(:,2),'g')
%  plot(ttt,ires(:,1))
% hold on
% plot(ttt,ires(:,2),'g')
% hold off
%

% % % Swaping the column numbers
% % j=1;
% % for i=iel:-1:1
% %     Bactn(:,j)=Bact(:,i);
% %     j=j+1;
% % end
% % Bact=Bactn;

% %Call the optimization function
% %optactuator(A_mat,Bact,C,optact0,tol,itmax,type,cov);
%
[sigma,ub,lb,numit,optact,f,K,k]=optactuator(Phi,A_mat_m,Bact,C_mat_m,1
,0.000001,100,1,0);
% %
[sigma,ub,lb,numit,optact,f,K,k]=optactuator(A_mat,Bact,C_mat,5,0.00000
1,100,1,0);
% optlocation=optact(k);

% % %Hand calculation
% Q=C_mat_m'*C_mat_m;
% % Q=C_mat'*C_mat;
% R=1;
% for j=1:nel
%     B_mat=Bact(:,j);
```

```
% %     [f,K]=lqr(A_mat_m,B_mat,Q,R);
%     [f,K]=lqr(A_mat,B_mat,Q,R);
% [Vs,es]=eig(K);
% sigmahand(j)=.5*max(diag(es));
% end;
% opthand=min(sigmahand);
% optacthand=find(sigmahand<opthand+.00001)

% % % Hand calculation for controlability measure
% D=0;
% for j=1:nel
%     sys=ss(A_mat,Bact(:,j),C_mat,D);
%     WC=gram(sys,'c');
%     wc_tr(j)=trace(WC);
%     wc_det(j)=det(WC);
% end;

ploc=[3];
for pl=1:length(ploc)
%
vol=[100];
for vloop=1:length(vol)
% Optimal configuration
ff=zeros(sdof,1);     % initialization of system force vector
kk=zeros(sdof,sdof);  % initialization of system matrix
mm=zeros(sdof,sdof);  % initialization of system matrix
index=zeros(nnel*ndof,1);  % initialization of index vector
% i=1:nel; piezo(i)=1;

% piezo(1)=1;  piezo(4)=1;  piezo(6)=1;
piezo(ploc(pl))=1;
for iel=1:nel    % loop for the total number of elements
index=feeldof1(iel,nnel,ndof);  % extract system dofs associated with
element
[k,m]=febeam1(el,xi,leng,area,rho,ipt); % compute element stiffness
matrix
% loading
if q0>0
    f=feload(leng,q0,'l');
else
    f=feload(leng,0,'l');
end
if piezo(iel)==1
%         fp=feload(leng,mp,'p');
    fp=feload(leng,mp*vol(vloop),'p');
else
    fp=feload(leng,0,'p');
end
f=f+fp;
kk=feasmbl1(kk,k,index); % assemble each element matrix into system
matrix
ff(nnode)=0;
[kk,ff]=feasmbl2(kk,ff,k,f,index);
mm=feasmbl1(mm,m,index); % assemble each element matrix into system
end
[kks,ffs]=feaplyc2(kk,ff,bcdof,bcval);  % apply the boundary conditions
[kkd,mmd]=feaplycs(kk,mm,bcdof);  % apply the boundary conditions
```

```
%Static analysis
fsol=kks\ffs;   % Solve the matrix equation
% plotting the deformation
i=1:2:sdof-1;
j=1:nnode;
x=0:leng:tleng;
y(j)=fsol(i);
yv(:,vloop)=y(j);
end
hold on
figure(1)
plot(x,yv)

hold on
end

% % %Static analysis with one piezo at a time
for ielp=1:nel
i=1:nel; piezo(i)=0; piezo(ielp)=1;
ff=zeros(sdof,1);     % initialization of system force vector
index=feeldof1(ielp,nnel,ndof);  % extract system dofs associated with
element
load=mp; type='p';
f=feload(leng,load,type);
ff=feasmblf(ff,f,index);
% [kks,ffs]=feaplyc2(kk,ff,bcdof,bcval);  % apply the boundary
conditions
fsols(:,ielp)=kks\ff;   % Solve the matrix equation

if(ielp>1)
% plotting the deformation
i=1:2:sdof-1;
j=1:nnode;
x=0:leng:tleng;
y(j)=fsols(i,ielp);
figure(2)
hold on
plot(x,y)
end
end

% Dynamic analysis
%
% % % NOTE TO SELF
% A_mat_m = A_mat_m;
% B_mat_m = Bact(:,10);
% B_mat = Bact(:,6);
% C_mat_m = zeros(1,40);
% C_mat_m(20) = 1;
% D_mat_m = zeros(1,1);

t0 = 0;
t1 = 10;
nsteps1 = round((t1-t0)*200);
hsteps1 = (t1-t0)/nsteps1;
N=sdof-2;
```

```
% kk=kkk;
% mm=mmm;
% for i=1:sdof-2
%     fff(i)=ff(i+2);
% end
% ff=fff';
% % First order representation
% C=0.2*mm+0.005*kk;  % Alpha=0.2, Beta=0.005
% % C=0;
% A_mat1 = [zeros(size(kk)) eye(size(kk))
%          -inv(mm)*kk     -inv(mm)*C];
% C_mat1=[eye(N), zeros(N)];
% B_mat1 = [zeros(size(ff))
%             inv(mm)*ff];
% D1=0;
% CO=ctrb(A_mat1,B_mat1);
% rank(CO)


B_mat=[Bact(:,6) F_mat_m];
% B_mat=[Bact(:,5) F_mat];
u=100;
d=1;
y0 = zeros(2*N,1);
tspan = t0:hsteps1:t1;
options =
odeset('JConstant','on','Jacobian','on','BDF','on','RelTol',1e-4,
'AbsTol',1e-4);
[t,y] = ode23s('yprime_beam',tspan,y0,options,A_mat_m,B_mat,u,d);
% [t,y] = ode23s('yprime_beam',tspan,y0,options,A_mat,B_mat,u,d);
yt1 = y(:,1:N);
figure(3)
% plot(t,yt1(:,end-1))  %uncontrolled ode23s
t=tspan;
u_new = [u*sin(t); d*sin(t)];
% sys = ss(A_mat, B_mat, C_mat,0);
sys = ss(A_mat_m, B_mat, C_mat_m,0);
% WC=gram(sys,'c');
[y,t] = lsim(sys,u_new,t);
% figure(4);
hold on;
plot(t,y(:,end-1),'r');  %uncontrolled lsim
% hold off;
xlabel('Time [s]');
ylabel('Amplitude [m]');


% optimal control
% Q=C_mat'*C_mat;
Q=C_mat_m'*C_mat_m;
R=[1 0; 0 1];
% [f,K]=lqr(A_mat,B_mat,Q,R);
% A_new=(A_mat-Bact(:,6)*ones(1,length(B_mat))*K);
% syso = ss(A_new, F_mat, C_mat,0);
[f,K]=lqr(A_mat_m,B_mat,Q,R);
A_new=(A_mat_m-Bact(:,6)*ones(1,length(B_mat))*400000000*K); %
syso = ss(A_new, F_mat, C_mat_m,0);
```

```
[y,t] = lsim(syso,d*sin(t),t);
% figure(6);
% hold on;
plot(t,y(:,end-1),'g');

hold on;
% optimal control2
% Q=C_mat'*C_mat;
Q=C_mat_m'*C_mat_m;
R=[1 0; 0 1];
% [f,K]=lqr(A_mat,B_mat,Q,R);
% A_new=(A_mat-Bact(:,6)*ones(1,length(B_mat))*K);
% syso = ss(A_new, F_mat, C_mat,0);
% B_mat=[Bact(:,6) F_mat_m];
[f,K]=lqr(A_mat_m,B_mat,Q,R);
A_new=(A_mat_m-Bact(:,6)*ones(1,length(B_mat))*400000000*K);
syso = ss(A_new, F_mat, C_mat_m,0);
[y,t] = lsim(syso,2*sin(t),t);
% figure(6);
% hold on;
plot(t,y(:,end-1),'b');

hold off;


% Controllability tests
% CO=ctrb(A_mat_m,F_mat_m);
sys=ss(A_mat_m,B_mat,C_mat_m,0);
[sysb,g]=balreal(sys);
rsysb=balred(sysb,5);
CO=ctrb(rsysb.A,rsysb.B);
rank(CO);



% figure(4)
% plot(tspan,60*sin(tspan));
% figure(5)
% i=1;
% for ip=1:2:nel*2
%     yp(:,i)=y(:,ip);
%     i=i+1;
% end
% mesh(yp)
%
% figure(6)
% i=1;
% for ip=1:2:nel*2
%     ypt(:,i)=yt1(:,ip);
%     i=i+1;
% end
% mesh(ypt)

% x=1:10;
% y=t;
% z=yp(:,end-1);
% [xp,yp]=meshgrid(x,y);
```

```
% zp=griddata(x,y,z,xp,yp);
% surf(xp,yp,zp);
%
% % Q=C_mat_m'*C_mat_m;
% % R=1;
% % [f,K]=lqr(A_mat_m,B_mat_m,Q,R);
% % sysc = ss((A_mat_m-B_mat_m*K), B_mat_m, C_mat_m,0);
% % [y,t] = lsim(sys,60*sin(t),t);
% % figure(6)
% % plot(t,y(:,end-1),'g');

% % % Simulink
% tend = tspan(end);
% Ts = hsteps1;
% u = [tspan' 60*sin(tspan')];
% sim('beam_sim.mdl');
% plot(tspan, y(:,end-1));
% %
 %Step response
% figure(6)
% y=impulse(syso);
% plot(y(:,2));
% figure(7)
% plot(y(:,3));
%plot(t,y(:,20))
```

aplybcdyn.m

```
function [kkk,mmm,fff]=aplybcdyn(bcdof,kk,mm,ff)
%Apply BC for dynamic analysis - general way
kkk=kk;
mmm=mm;
fff=ff;
% Row
for i = 1:length(bcdof)
    if bcdof(i)-i+1 == 1
        tmpk = [kkk(bcdof(i)-i+2:end,:)];
        tmpm = [mmm(bcdof(i)-i+2:end,:)];
    elseif bcdof(i)-i+1 == size(kkk,1)
        tmpk = [kkk(1:bcdof(i)-i,:)];
        tmpm = [mmm(1:bcdof(i)-i,:)];
    else
        tmpk = [kkk(1:bcdof(i)-i,:); kkk(bcdof(i)-i+2:end,:)];
        tmpm = [mmm(1:bcdof(i)-i,:); mmm(bcdof(i)-i+2:end,:)];
    end
    kkk = tmpk;
    mmm = tmpm;
end
% Col
for i = 1:length(bcdof)
    if bcdof(i)-i+1 == 1
      tmpk = [kkk(:,bcdof(i)-i+2:end)];
      tmpm = [mmm(:,bcdof(i)-i+2:end)];
    elseif bcdof(i)-i+1 == size(kkk,2)
       tmpk = [kkk(:,1:bcdof(i)-i)];
       tmpm = [mmm(:,1:bcdof(i)-i)];
    else
```

```
        tmpk = [kkk(:,1:bcdof(i)-i) kkk(:,bcdof(i)-i+2:end)];
        tmpm = [mmm(:,1:bcdof(i)-i) mmm(:,bcdof(i)-i+2:end)];
    end
    kkk = tmpk;
    mmm = tmpm;
end
% F
for i = 1:length(bcdof)
    if bcdof(i)-i+1 == 1
        tmpf = [fff(bcdof(i)-i+2:end)];
    elseif bcdof(i)-i+1 == length(fff)
        tmpf = [fff(1:bcdof(i)-i)];
    else
        tmpf = [fff(1:bcdof(i)-i); fff(bcdof(i)-i+2:end)];
    end
    fff = tmpf;
end
fff=fff';
```

feaplyc2.m

```
function [kk,ff]=feaplyc2(kk,ff,bcdof,bcval)
n=length(bcdof);
 sdof=size(kk);

 for i=1:n
    c=bcdof(i);
    for j=1:sdof
       kk(c,j)=0;
    end

    kk(c,c)=1;
    ff(c)=bcval(i);
 end
```

feaplycs.m

```
function [kk,mm]=feaplycs(kk,mm,bcdof)
n=length(bcdof);
 sdof=size(kk);

 for i=1:n
    c=bcdof(i);
    for j=1:sdof
       kk(c,j)=0;
       kk(j,c)=0;
       mm(c,j)=0;
       mm(j,c)=0;
    end

    mm(c,c)=1;
 end
```

feasmbl1.m

```
function [kk]=feasmbl1(kk,k,index)
edof = length(index);
 for i=1:edof
   ii=index(i);
```

```
    for j=1:edof
      jj=index(j);
        kk(ii,jj)=kk(ii,jj)+k(i,j);
    end
 end
```

feasmbl2

```
function [kk,ff]=feasmbl2(kk,ff,k,f,index)
 edof = length(index);
 for i=1:edof
   ii=index(i);
     ff(ii)=ff(ii)+f(i);
     for j=1:edof
       jj=index(j);
         kk(ii,jj)=kk(ii,jj)+k(i,j);
     end
 end
```

feasmblf.m

```
function [ff]=feasmblf(ff,f,index)
edof = length(index);
 for i=1:edof
   ii=index(i);
     ff(ii)=ff(ii)+f(i);
 end
```

feload.m

```
function [f]=feload(leng,load,type);
% This function forms the load vector
if type=='l'
    c=load/12;
    f=c*[6*leng; leng^2; 6*leng; -1*leng^2];
end
if type=='p'
    f=[0; load/2; 0; -load/2];
end
end
```

febeam1.m

```
function [k,m]=febeam1(el,xi,leng,area,rho,ipt)
% stiffness matrix
 c=el*xi/(leng^3);
 k=c*[12      6*leng   -12       6*leng;...
      6*leng  4*leng^2 -6*leng   2*leng^2;...
     -12     -6*leng    12      -6*leng;...
      6*leng  2*leng^2 -6*leng   4*leng^2];

% consistent mass matrix

 if ipt==1

    mm=rho*area*leng/420;
    m=mm*[156      22*leng   54       -13*leng;...
          22*leng  4*leng^2  13*leng  -3*leng^2;...
```

```
          54       13*leng   156      -22*leng;...
         -13*leng -3*leng^2 -22*leng   4*leng^2];

% lumped mass matrix

 elseif ipt==2

    m=zeros(4,4);
    mass=rho*area*leng;
    m=diag([mass/2  0  mass/2  0]);

% diagonal mass matrix

 else

    m=zeros(4,4);
    mass=rho*area*leng;
    m=mass*diag([1/2  leng^2/78  1/2  leng^2/78]);

end
```

feeldof1.m

```
function [index]=feeldof1(iel,nnel,ndof)
edof = nnel*ndof;
 start = (iel-1)*(nnel-1)*ndof;

   for i=1:edof
      index(i)=start+i;
   end
```

yprime_beam.m

```
function ydot = yprime_beam(t,y,flag,A_mat,B_mat,u,d)
switch flag
case ''
%     force = F_vec*1*sin(t);
    u_new = [u*sin(t'); d*sin(t')];
    force = B_mat*u_new;
%     force = B_mat(:,1)*u*sin(t) + B_mat(:,2)*d*sin(t);
%     force = F_vec*60;
    ydot = A_mat*y + force;

case 'jacobian'
    ydot = A_mat;
end
```

optactuator.m

```
function
[sigma,ub,lb,numit,optact,f,K,k]=optactuator(Phi,A,Bact,C,optact0,tol,i
tmax,type,cov);
% First pass at file to calculate optimal actuator locations
% 1 actuator. Columns of Bact are possible control operators.
% Pi0 is vector with entries 1 or 0 to indicate initial actuator
locations.
% Direct solution of lyapunov & ARE equations
% type=1, max eigenvalue; type =0 , trace
```

```
% % optactuator(A,Bact,C,1,0.001,100,1,0);
[N,M]=size(Bact);
R=eye(M);
% C=Phi'*C*Phi;
Q=C'*C;
%
% Step 1
% Initialization
%
k=1;
Pi=zeros(M,1);
Pi(optact0)=1;
optact(1)=optact0;
B=Bact(:,optact(1));

[f,K]=lqr(full(A),full(B),full(Q),1);
if(type==1)
[V,es]=eig(K);
zeta=V(:,N);
sigma(k)=.5*max(eig(K));
else
zeta=cov;
sigma(k)=.5*trace(K*cov);
end;
ub(k)=sigma(k);
lb(k)=0;
err=(ub(k)-lb(k))/ub(k);

relcon=zeros(M,1);

%
% Optimization loop
%
while(err>tol & k<itmax)
    % Step 2 solve master problem
    ZZ=lyap(A-B*f,zeta*zeta');
    S=-1/2*K*ZZ*K;
    for j=1:M
    grad(j)=trace(Bact(:,j)*(Bact(:,j)'*S));
    end;
    d=sigma(k)-grad*Pi;
  %
  % alg. for 1 actuator using (65) in Geromel
  % first find max over all previous calc., calc. at 1st location
  for j=1:M;
      relcon(j)=max(relcon(j),d(:)+grad(:,j));
  end;
  lb(k+1)=min(relcon);
  index=find(relcon<lb(k+1)+tol*.001);
  optact(k+1)=index(1);
  %
% Now have current best location, and lower bound on cost
%
% Step 3
% Update cost with current best location
Pi=zeros(M,1);
```

```
Pi(optact(k+1))=1;
B=sparse(Bact(:,optact(k+1)));
% Using 0 as initial estimate
%[f,cP,Ricounter,Lycounter,TL]=sparsenk2(A,B,C,sparse(zeros(1,6)),nktol
*.01
[f,K]=lqr(full(A),full(B),full(Q),1);
if(type==1)  % use lambda max

[V,es]=eig(K);
zeta=V(:,N);
sigma(k+1)=.5*max(eig(K));
else
    sigma(k+1)=.5*trace(K*cov);
end;
if(ub>sigma(k+1))
    ub=sigma(k+1);
end;
err=(sigma(k+1)-lb(k+1))/sigma(k+1);
k=k+1;
% Repeat if upper bound currentbd and lower bound theta not close.
end;
if(k==itmax)
    disp('WARNING: maximum number of iterations for actuator location
reached.')
end;
numit=k;
disp('Optimal location is')
disp(optact(k))
%
```