Program plaxL.m

1 FE program `plaxL`

The program `plaxL` can be used to model and analyze linear elastic deformation of planar and axisymmetric structures. The planar problems can be either plane strain or plane stress. For axisymmetric problems, the rotation axis is the global y-axis and the radial axis is the global x-axis. The half cross-section must be modelled for $x \geq 0$. Both linear 4-node quadrilateral and quadratic 8-node quadrilateral elements can be used.

2 Example input file

As an example, a tensile test is modelled and analyzed, both in plane stress and axisymmetricly. Because the deformation is homogeneous, only one element is used.

Figures are closed and the Matlab work space is cleared.

```
  close all; clear all;
```

The coordinates of the nodes are given in the matrix "crd0". Units are chosen to be mm.

```
 crd0 = [ 0 0; 100 0; 100 100; 0 100 ];
```

The location array "lok" contains information about the element type (first column), element group (second column) and connectivity (last four columns). The element type can be : 3 for plane stress, 11 for plane strain and 10 for axisymmetry. As we have only one element, there is only one element group. The element node numbers are 1, 2, 3 and 4.

```
 lok = [ 3 1 1 2 3 4 ];
```

For each group we have to supply geometric and material data in the array "elda". In the subsequent columns we provide :

- 1 : random
- 2 : material number : 11 is isotropic linear material
- 3 : thickness
- 4 : unused
- 5 : unused
- 6 : Young's modulus
- 7 : Poisson's ratio

```
  elda = [ 0 11 1 0 0 200000 0.3 ];
```

Prescribed displacements are provided in the array "pp". For each prescribed displacement component we have one row. The first column contains the node, the second column contains the direction (either 1 (= x = horizontal) or 2 (= y = vertical). The third column contains the value.

When we want to prescribe the elongation of 1 mm, we have :

```
pp = [ 1 1 0; 1 2 0; 2 2 0; 4 1 0 ];
pp = [ pp; 3 2 1; 4 2 1 ];
```

We could also have prescribed a tensile force of 200000 N.

```
pp = [ 1 1 0; 1 2 0; 2 2 0; 4 1 0 ];
pf = [ 3 2 100000; 4 2 100000 ];
```

The input is complete and `plaxL` can be executed to analyze the behavior.

```
plaxL;
```

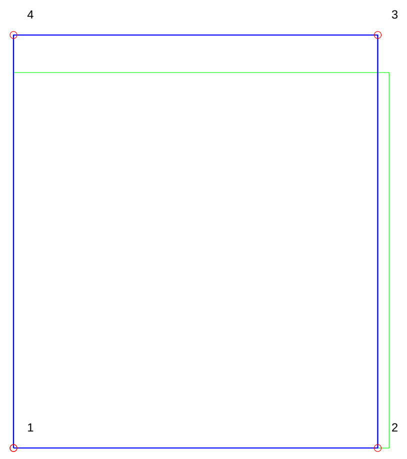
After the analysis the nodal displcacements are available in the array "Mp". Element data can be found in the database "eidaC".

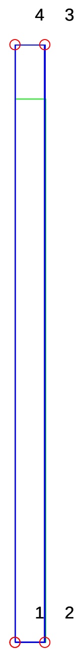
When an axisymmetric model is used, the axis is along the global y-axis, so vertical. The coordinate and location array are then :

```
crd0 = [ 0 0; sqrt(100/pi) 0; sqrt(100/pi) 100; 0 100 ];
lok = [ 10 1 1 2 3 4 ];
```

When a tensile force is applied it will be different in the node on the central axis then in the node on the outer edge.

The figure below shows the deformation in both cases.





3 Matlab program `plaxL`

The program `plaxL.m` is listed below and is seeded with comments to explain variables and actions.

```
%**********************************************************************
% plaxL : 2-dimensional planar/axisym.; linear
%======================================================================

nndof = 2;
nnod  = size(crd0,1);
ndof  = nnod * nndof;
ne    = size(lok,1);
negr  = size(elda,1);

Trm = eye(ndof);
if exist('tr'),
  ntr = size(tr,1);
  for itr=1:ntr
    trp = round(tr(itr,1));     tra = tr(itr,2);
    trc = cos((pi/180)*tra);  trs = sin((pi/180)*tra);
    k1  = nndof*(trp-1)+1;    k2  = nndof*(trp-1)+2;
    trm = [trc -trs ; trs trc];
    Trm([k1 k2],[k1 k2]) = trm;
  end;
else, ntr = 0; tr = []; end;

for e=1:ne
  ety = lok(e,1); egr = lok(e,2);
  if     ety==3,  nenod=4; neip=4; vrs=2;
  elseif ety==11, nenod=4; neip=4; vrs=1;
  elseif ety==10, nenod=4; neip=4; vrs=3; end;
  nedof = nndof * nenod;

  k=1;
  for n=1:nenod
    for v=1:nndof
      lokvg(e,k) = nndof*(lok(e,2+n)-1)+v;
      k=k+1;
    end;
  end;

  elpa(e,1:7) = [ety egr nenod nndof nedof neip vrs];

  mat = elda(egr,2);         thk = elda(egr,3);
  gf2 = elda(egr,4);         gf3 = elda(egr,5);
%  mcl = floor(mnr/10);       mty = rem(mnr,10);
  if mat==1
    E = elda(egr,6);   Gn = elda(egr,7);
  end;

  for ip=1:neip
```

```
    gip = neip*(e-1) + ip;
    eida0(gip,:) = [ ety mat thk gf2 gf3 E Gn ];
    eidaC(gip,:) = [ ety mat thk gf2 gf3 E Gn ];
  end;

end;

%-----------------------------------------------------------------------

if ~exist('pp'), pp = []; ppc = []; ppv = []; end;
if ~exist('pf'), pf = []; pfc = []; pfv = []; end;

npdof = size(pp,1);
npfor = size(pf,1);
nudof = ndof - npdof;

if npdof>0
  ppc = [nndof*(round(pp(:,1))-1)+round(pp(:,2))];
  ppv = pp(:,nndof+1);
end;
if npfor>0,
  pfc = [nndof*(round(pf(:,1))-1)+round(pf(:,2))];
  pfv = pf(:,nndof+1);
end;

%  Information for partitioning the system of equations associated
%  with linked degrees of freedom is made available in the arrays
%  plc  and  prc.

if ~exist('pl'),  pl  = []; plc = []; end;
if ~exist('pr'),  pr  = []; prc = []; end;
if ~exist('lim'), lim = []; end;

npl = size(pl,1);
npr = size(pr,1);

if ~exist('lif'), lif = zeros(1,npl); end;

if npl>0
  plc = [nndof*(round(pl(:,1))-1)+round(pl(:,2))];
  prc = [nndof*(round(pr(:,1))-1)+round(pr(:,2))];
end;

%  Some extra arrays are made for later use.

pa = 1:ndof; pu = 1:ndof; prs = 1:ndof;
pu([ppc' plc']) = [];
```

```
prs([ppc' pfc' plc']) = [];

%  pe0   :  column with prescribed initial displacements
%  fe0   :  array with prescribed initial forces

pe0 = zeros(ndof,1); pe0(ppc(1:npdof)) = ppv(1:npdof);
fe0 = zeros(ndof,1); fe0(pfc(1:npfor)) = pfv(1:npfor);

%-----------------------------------------------------------------------
% Initialization to zero
%  pe    :  column with nodal displacements
%  p     :  column with nodal displacements
%  fe    :  column with external (applied) nodal forces
%  fi    :  column with internal (resulting) nodal forces
%  #T    :  column with transformed components
%-----------------------------------------------------------------------

pe  = zeros(ndof,1); p   = zeros(ndof,1); pT  = zeros(ndof,1);
fe  = zeros(ndof,1); fi  = zeros(ndof,1);
feT = zeros(ndof,1); fiT = zeros(ndof,1);

%-----------------------------------------------------------------------
% Loop over all elements to generate element stiffness matrix 'em'
% Assemble 'em' into structural stiffness matrix 'sm'
%  ec0  : initial coordinates of element nodes
%  ec   : current coordinates of element nodes
%-----------------------------------------------------------------------
sm=zeros(ndof);

for e=1:ne
  ety = elpa(e,1); egr = elpa(e,2);
  nenod = elpa(e,3); nedof = elpa(e,5); neip = elpa(e,6);
  ec0 = crd0(lok(e,3:2+nenod),:); ec = ec0;
  em = zeros(nedof);

    [ksi,psi,psidksi,ipwf] = fbibfe2dq48(e,elpa(e,:));  % fbibfe2dq48.m
    vrs = elpa(e,7);

    for ip=1:neip
      gip = neip*(e-1) + ip;
%-----------------------------------------------------------------------
  if vrs==3
     r0   = psi(ip,:)*ec0(:,1);     r    = psi(ip,:)*ec(:,1);
     thk0 = r0*2*pi;                thk  = r*2*pi;
     mF33 = r/r0;                   ax   = 1;
  else
     r0   = 1;                      r    = 1;
     thk0 = eida0(gip,3);           thk  = thk0;
     mF33 = 1;                      ax   = 0;
  end;

  dpsi(:,1) = psidksi(:,2*ip-1);    dpsi(:,2) = psidksi(:,2*ip);
  jc0  = dpsi' * ec0;   jci0 = inv(jc0);
  jc   = dpsi' * ec;      jci  = inv(jc);
  dt   = det(jc);         dt0  = det(jc0);
  dfie0   = zeros(5,2*nenod);
  dpsixy0 = dpsi * jci0' ;
  dfie0(1,2*(1:nenod)-1) = dpsixy0(1:nenod,1)';
  dfie0(2,2*(1:nenod))   = dpsixy0(1:nenod,2)';
  dfie0(3,2*(1:nenod)-1) = ax.*psi(ip,1:nenod)/r0;
  dfie0(4,2*(1:nenod))   = dpsixy0(1:nenod,1)';
  dfie0(5,2*(1:nenod)-1) = dpsixy0(1:nenod,2)';
  mF  = eye(3); mF(1:2,1:2)  = jci0*jc;  mF(3,3)  = mF33;  mF  = mF';
%------------------------------------------------------------------------
      du = zeros(5,1);
      [mmM,ccGs,ccGe,vm] = plaxelas1(eida0(gip,6:7),vrs,du);
                                                            % plaxelas1.m

      em = em + dfie0' * mmM * dfie0 * thk0 * dt0 * ipwf(ip);
    end;

  sm(lokvg(e,1:nedof),lokvg(e,1:nedof)) = ...
    sm(lokvg(e,1:nedof),lokvg(e,1:nedof)) + em;
end; % element loop 'e'

%------------------------------------------------------------------------
% Transformation for local nodal coordinate systems
%------------------------------------------------------------------------
sm = Trm'*sm *Trm;
%------------------------------------------------------------------------
% Boundary conditions and links
%------------------------------------------------------------------------
pe = pe0; fe = fe0; rs = fe;

if npl>0, rs = rs - sm(:,plc)*lif'; end;

%------------------------------------------------------------------------
% Partitioning is done in the function                      fbibpartit.m

[sm,rs] = fbibpartit(1,sm,rs,ndof,pa,ppc,plc,prc,pe,lim);

%------------------------------------------------------------------------
% Solving the system of equations and take prescribed displacements
% and links into account.
% Update nodal point coordinates 'crd'.

sol = inv(sm)*rs; % sol = sm\rs;

pe(pu) = sol;
if npl>0, pe(plc) = lim*pe(prc) + lif'; end;

p = pe; pT = Trm * p;
crd = crd0 + reshape(pT,nndof,nnod)';

%----------------------------------------------------------------------
% Calculate stresses and strains and the internal forces 'ef'.
% Internal forces 'ef' are assembled into 'fi', the structural
% internal forces, representing the reaction forces.
%----------------------------------------------------------------------
fi = zeros(ndof,1);

for e=1:ne
  ety = elpa(e,1); egr = elpa(e,2);
  nenod = elpa(e,3); nedof = elpa(e,5); neip = elpa(e,6);
  ec0 = crd0(lok(e,2+1:2+nenod),:);
  ec = crd(lok(e,2+1:2+nenod),:);
  Tpe  = p(lokvg(e,1:nedof));
  em = zeros(nedof); ef = zeros(nedof,1);

    [ksi,psi,psidksi,ipwf] = fbibfe2dq48(e,elpa(e,:));  % fbibfe2dq48.m
    vrs = elpa(e,7);

    for ip=1:neip
      gip = neip*(e-1) + ip;
%----------------------------------------------------------------------

  if vrs==3
     r0   = psi(ip,:)*ec0(:,1);     r    = psi(ip,:)*ec(:,1);
     thk0 = r0*2*pi;                thk  = r*2*pi;
     mF33 = r/r0;                   ax   = 1;
  else
     r0   = 1;                      r    = 1;
     thk0 = eida0(gip,3);           thk  = thk0;
     mF33 = 1;                      ax   = 0;
  end;

  dpsi(:,1) = psidksi(:,2*ip-1);    dpsi(:,2) = psidksi(:,2*ip);
  jc0  = dpsi' * ec0;   jci0 = inv(jc0);
  jc   = dpsi' * ec;      jci  = inv(jc);
  dt   = det(jc);         dt0  = det(jc0);
  dfie   = zeros(5,2*nenod);
  dpsixy = dpsi * jci' ;
  dfie(1,2*(1:nenod)-1) = dpsixy(1:nenod,1)';
  dfie(2,2*(1:nenod)) = dpsixy(1:nenod,2)';
  dfie(3,2*(1:nenod)-1) = ax.*psi(ip,1:nenod)/r;
  dfie(4,2*(1:nenod)) = dpsixy(1:nenod,1)';
  dfie(5,2*(1:nenod)-1) = dpsixy(1:nenod,2)';
  dfie0   = zeros(5,2*nenod);
  dpsixy0 = dpsi * jci0' ;
  dfie0(1,2*(1:nenod)-1) = dpsixy0(1:nenod,1)';
  dfie0(2,2*(1:nenod))   = dpsixy0(1:nenod,2)';
  dfie0(3,2*(1:nenod)-1) = ax.*psi(ip,1:nenod)/r0;
  dfie0(4,2*(1:nenod))   = dpsixy0(1:nenod,1)';
  dfie0(5,2*(1:nenod)-1) = dpsixy0(1:nenod,2)';
  mF  = eye(3); mF(1:2,1:2)  = jci0*jc;  mF(3,3)  = mF33;  mF  = mF';
%-----------------------------------------------------------------------

      du = dfie0*Tpe;
      [mmM,ccGs,ccGe,vm] = plaxelas1(eidaC(gip,6:7),vrs,du);
      ccGs = ccGs'; ccGe = ccGe';
      thk = thk0 + thk0*ccGe(3);

      eidaC(gip,3)     = thk;
      eidaC(gip,17:20) = ccGe(1:4);
      eidaC(gip,21:24) = ccGs(1:4);
      eidaC(gip,25)    = dt;
      eidaC(gip,90)    = r0;
      eidaC(gip,91)    = r;

      ef = ef + dfie' * ccGs' * thk * dt * ipwf(ip);
    end;

  fi(lokvg(e,1:nedof))                 = fi(lokvg(e,1:nedof)) + ef;
end;
rs = fe - fi;
fi=Trm'*fi; fiT = fi; fiT = Trm * fi; rsT = feT - fiT;

%-----------------------------------------------------------------------
% Reshaping columns into matrices

Mp  = reshape(p,nndof,nnod)';
Mfi = reshape(fi,nndof,nnod)';   Mfe = reshape(fe,nndof,nnod)';
Mrs = reshape(rs,nndof,nnod)';

if ntr>=1
MpT = reshape(TpT,nndof,nnod)';
MfiT = reshape(fiT,nndof,nnod)'; MfeT = reshape(feT,nndof,nnod)';
MrsT = reshape(rsT,nndof,nnod)';
```

```
end;
%-------------------------------------------------------------------------

%*************************************************************************
```