Program tr2d

1 FE program `tr2d`

The Matlab program `tr2d` allows to model and analyze two-dimensional truss structures, where trusses are homogeneous and can behave nonlinear. Deformation and rotations can be large, i.e. the behavior is geometrically nonlinear.

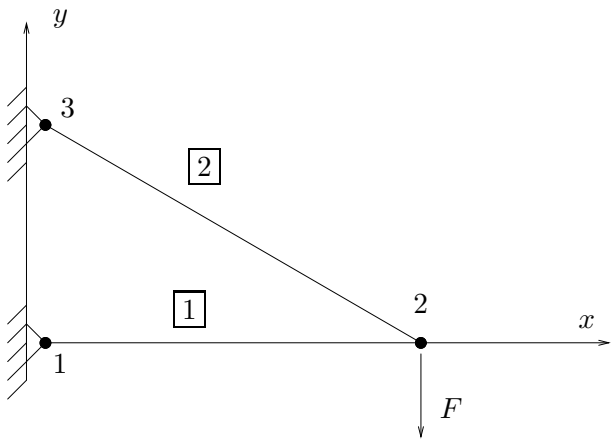
Model geometry, topology (connectivity), geometrical and material parameters, boundary conditions (prescribed displacements and point loads) and link relations (dependecies between degrees of freedom) must be available as input data. Also the history of the prescribed boundary conditions must be specified.

When the analysis is finished, output date are stored in the data base and various other data arrays.

In the following section an example input is presented, with explanatory comments. Finally the program itself is explained in more detail.

2 Example input file

As an example, the two-bar truss structure, shown in the figure below will be modelled, loaded and analyzed.



Both trusses have different geometrical and material properties, The material of truss 1 behaves elastcally, according to a linear relation between the axial stress σ and the linear strain $\varepsilon = \varepsilon_l = \lambda - 1$, specified by the modulus E and the Poisson's ratio ν. Truss 2 behaves linearly elastic upto the yield stress σ_{y0} after which isotropic linear hardening occurs with hardening parameter $H = E/20$. The vertical force in node 4 is increased from 0 to -50000 N and decreases back to zero again.

The Matlab workspace is cleared and figures are closed. The file 'loadincr.m', which describes the time history of the boundary conditions, is deleted. The file 'savedata.m', which describes, which calculated values have to be saved for postprocessing, is also deleted.

```
clear all; close all;
delete('loadincr.m'); delete('savedata.m');
```

Coordinates of the nodal points and the connectivity is given in the matrices "crd0" and "lok". In the latter array, the first column contains the element type and the second column the element group.

```
crd0 = [ 0 0; 100 0; 0 100/sqrt(3) ];
lok = [ 9 1 1 2 ; 9 2 2 3 ];
```

element data are given in the array "elda", which has one row for each property group. The second column contains the material number, which has two digits. The first digit is the material class ("mcl") and the second digit is the material type ("mty"). The next classes and types are currently implemented :

mcl = 1 : elastic material
mcl = 2 : elastomeric material
mcl = 3 : elastoplastic material
mcl = 4 : linear viscoelastic material
mcl = 5 : viscoplastic material (Perzyna)
mcl = 6 : nonlinear viscoelastic material (Leonov)
mcl = 7 : elastoviscous material (creep)

For the element type you should look in the Matlab source file.

```
elda = [ 0 11 10 0 0 200000 0.3 0 0 0 ;
         0 31 20 0 0 200000 0.3 250 10000 0 ];
```

We also have to indicate which hardening law ("hm") is used and which stress update procedure ("pr", explicit or implicit).

```
hm = 'li'; pr = 'ex';
```

The boundary conditions are prescribed, first the incremental displacements, then the incremental forces.

```
pp = [ 1 1 0; 1 2 0; 3 1 0; 3 2 0; ];
pf = [ 2 2 -100 ];
```

The load history is prescribed with a call to the function file "mloin.m". In its source file, it is explained how it must be used.

```
[St,Sft,nic,GDt,tend] = mloin(0,0,400,1,'pol',[0 0 200 50 400 0]);
```

Analysis data for postprocessing must be saved. This is done by making a file "savedata.m", which is called by the program `tr2d` at the end of each increment.

```
sada=fopen('savedata.m','w');
fprintf(sada,'Sf2x(ic)=Mfi(2,1);Sf2y(ic)=Mfi(2,2); \n');
fprintf(sada,'Su2x(ic)=MTp(2,1);Su2y(ic)=MTp(2,2); \n');
fprintf(sada,'SGe1(ic)=eldaB(1,6);SGe2(ic)=eldaB(2,6); \n');
fprintf(sada,'SGs1(ic)=eldaB(1,7);SGs2(ic)=eldaB(2,7); \n');
fclose(sada);
```

Now the input is complete and the program can be called for the analysis.

```
tr2d;
```

The result is shown in the figures below.

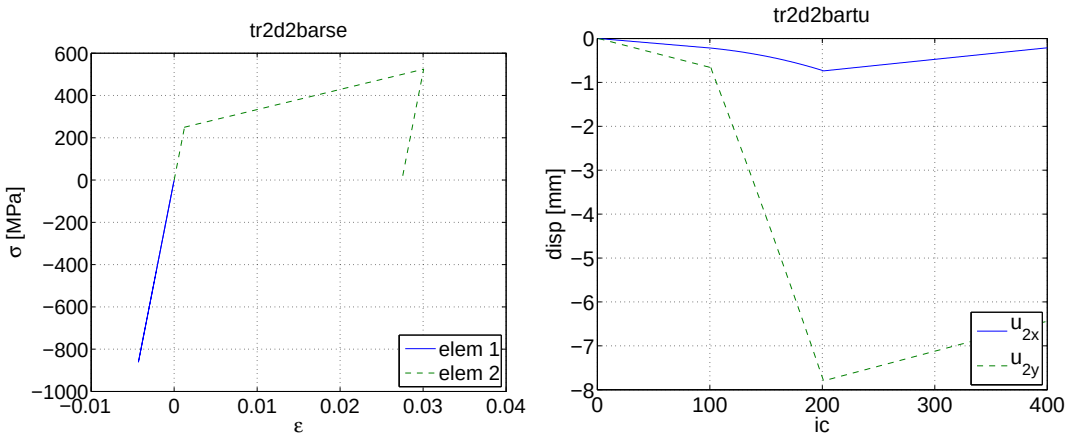


Fig. 2.1 : *Element stress-strain relations and displacements of node 4*

3 Matlab program `tr2d`

The program `tr2d.m` calls a collection of command and function files, which can and should be inspected for through understanding of the procedure. The program structure is however clearly shown in the listing.

```
%*********************************************************************
% tr2d : 2-dimensional truss element
%=====================================================================
tr2dchkinp;                                                % tr2dchkinp.m
fbiblcase;                                                  % fbiblcase.m
[Trm] = fbibtransbc(tr,ndof,nndof);                       % fbibtransbc.m
[elda0,eldaB,eldaC,mcl4] = tr2dinidat(ne,elgr,elda,neip,ts,lok,crd0,mm);
                                                           % tr2dinidat.m
tr2dinizer;                                                % tr2dinizer.m

save([matf num2str(0)]);
crdB = crd0; crd = crd0;
%=====================================================================
```

```
% Incremental calculation
%=========================================================================
ic = 1; ti = 0; it = 0; slow = 1;

while ic<=nic
%-------------------------------------------------------------------------
fbibcutback;                                                % fbibcutback.m
ti = ti + ts; it = 0; loadincr;                               % loadincr.m
pe = peC./slow; fe = feC;
rs = fe - fi;
Dp = zeros(ndof,1); Ip = zeros(ndof,1); IpT = zeros(ndof,1);
%=========================================================================
% System matrix is assembled from element matrices
% System matrix is transformed for local nodal coord.sys.
%=========================================================================
if (ic==1 | nl==1)
%-------------------------------------------------------------------------
sm=zeros(ndof); mcl4=0; mcl42=0; mcl9=0;

for e=1:ne
  ec = crd(lok(e,3:nenod+2),:);
  [ML,MN,V,eldaC] = tr2dgeom(e,ec,eldaC);                   % tr2dgeom.m
  tr2dmat;                                                   % tr2dmat.m

  em = CL * ML + CN * MN ;                     % element stiffness matrix
  ef = (CI-CV) * V;                         % element internal load column

  sm(lokvg(e,:),lokvg(e,:)) = sm(lokvg(e,:),lokvg(e,:)) + em;
end;

sm = Trm'*sm *Trm; if ic==1, sm00=sm; end; sm0=sm;
%-------------------------------------------------------------------------
end;
%=========================================================================
% Iterative calculation
%=========================================================================
nrm = 1000; it = 1; sm0 = sm;

while (nrm>ccr) & (it<=mit)
%-------------------------------------------------------------------------
%=========================================================================
% Links and boundary conditions are taken into account
% Unknown nodal point values are solved
% Prescribed nodal values are inserted in the solution vector
%=========================================================================
%sm = sm00;                   % only used to test modified Newton-Raphson
```

```
if npl>0, rs = rs - sm(:,plc)*lif'; end;
[sm,rs] = fbibpartit(it,sm,rs,ndof,pa,ppc,plc,prc,pe,lim);% fbibpartit.m

sol = inv(sm)*rs;

p = zeros(ndof,1); p(pu) = sol;
if it==1, p(ppc) = pe(ppc); end;
if npl>0, p(plc) = lim*p(prc) + lif'; end;

Dp = p; Ip = Ip + Dp; Tp = Tp + Dp;
%=====================================================================
% Transformation dof's from local to global nodal coordinate systems
%=====================================================================
DpT = Trm * Dp; IpT = IpT + DpT; TpT = TpT + DpT;
crd = crd0 + reshape(TpT,nndof,nnod)';
%=====================================================================
% Calculate stresses and strains.
% Make system matrix and internal force vector for next step.
%=====================================================================
sm=zeros(ndof); fi=zeros(ndof,1); mcl4=0; mcl42=0; mcl9=0;

for e=1:ne
  ec = crd(lok(e,3:nenod+2),:);             % element nodal coordinates
  [ML,MN,V,eldaC] = tr2dgeom(e,ec,eldaC);                  % tr2dgeom.m
  tr2dmat;                                                  % tr2dmat.m

  em = CL * ML + CN * MN ;                   % element stiffness matrix
  ef = (CI-CV) * V;                        % element internal load column

  sm(lokvg(e,:),lokvg(e,:)) = sm(lokvg(e,:),lokvg(e,:)) + em;
  fi(lokvg(e,:)) = fi(lokvg(e,:)) + ef;
end;

sm=Trm'*sm*Trm; fi=Trm'*fi;
%=====================================================================
% Calculate residual force and convergence norm
%=====================================================================
rs = fe - fi;
nrm = fbibcnvnrm(cnm,pu,ppc,prs,Dp,Ip,rs,fi);           % fbibcnvnrm.m

it = it + 1;                     % increment the iteration step counter

fbibwr2scr;                                               % fbibwr2scr.m
%---------------------------------------------------------------------
end; %it
%=====================================================================
% Transformation nodal forces from local to global nodal coord.sys.
```

```
%=========================================================================
fiT = fi; feT = fe; rsT = rs;
fiT = Trm * fi; feT = Trm * fe; rsT = Trm * rs;

%=========================================================================
% Update and store values
%=========================================================================
fbibcol2mat1;                                          % fbibcol2mat1.m
crdB = crd; feB = fe; eldaB = eldaC; HGsB = HGsC;
savefile = [matf num2str(ic)]; savedata;                   % savedata.m

ic = ic + 1;                          % increment the increment counter
save([matf '00'],'ic');                              % save date to 'matf'

%-------------------------------------------------------------------------
end; %ic

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