Program tr2dL

1 FE program `tr2dL`

The Matlab program `tr2dL` allows the modelling and analysis two-dimensional truss structures, where trusses are homogeneous and linear elastic. Deformation and rotations must be small, i.e. the behavior is geometrically linear.

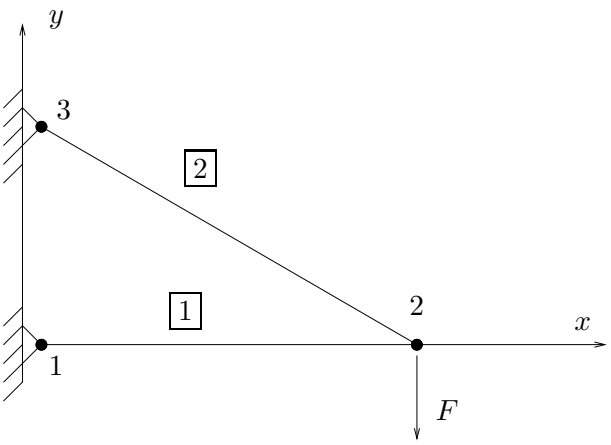
Model geometry, topology (connectivity), geometrical and material parameters, boundary conditions (prescribed displacements and point loads) and link relations (dependecies between degrees of freedom) must be available as input data.

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

In the following section an example input is presented, with explanatory comments. Finally the program source is listed and explained in more detail with included comment.

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, which are given in the table below.

truss		1	2	
cross-sectional area	A	10	20	[mm^2]
Young's modulus	E	200	150	[GPa]
Poisson's ratio	ν	0.3	0.3	[-]

Now let us see which Matlab commands do the job. Before starting, it might be wise to close all figures and clear the Matlab work space.

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

First we give the coordinates of the nodes in the array "crd0". Now we have to decide on the units and in this example we choose to model everything in mm.

```
crd0 = [ 0 0; 100 0; 0 100/sqrt(3) ];
```

The connectivity of the elements is defined in the array "lok". This array has a row for each element. The first column contains the element type, which is 9 for a truss. The second row is the element group number. Typically elements with the same properties are placed in one and the same group. Because our two elements indeed have different properties, they are placed in two different groups. The third and fourth column contain the first and second node of the element.

```
lok = [ 9 1 1 2 ; 9 2 2 3 ];
```

The geometrical and material properties are provided in the array "elda" (*el*ement *da*ta). For each element group we have a row in "elda". The first column contains a zero (0), which is not important for our use. The second column contains the material identification number. For linear elastic material, which we will use here, this number is 11 (eleven). The third column contains the cross-sectional area (in mm^2). The fourth and fifth column are not used for our problems and always contain a zero (0). The sixth and seventh column contain Young's modulus and Poisson's ratio. So for our example we have :

```
elda = [
0 11 10 0 0 200000 0.3
0 11 20 0 0 150000 0.3
];
```

Boundary conditions are prescribed nodal displacements and/or prescribed nodal forces. Prescribed nodal displacements are always needed to prevent rigid body motions.

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.

For our example we have :

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

The prescribed forces are given in the array "pf". Again each prescribed force component is placed on a row, with the node in the first, the direction in the second and the value in the third column.

For our example :

```
pf = [ 2 2 -100 ];
```

That is about all. The input is complete and the program can be executed to analyze the behavior.

```
tr2dL;
```

When the analysis is completed successfully, we want to see some results. First the nodal data, i.e. displacements and reaction forces. They are available in the array's "Mp" and "Mfi". Rows contain nodal data : displacement and forces in the first ($1 = x$ = horizontal) and second ($2 = y$ = vertical) directions. Just type the next commands in the Matlab shell.

```
Mp
Mfi
```

Element data, like stress and strain, are available in the data base. "eldaC". For element "e" we find the date in row "e" of "eldaC". Relevant date can be found at the following locations :

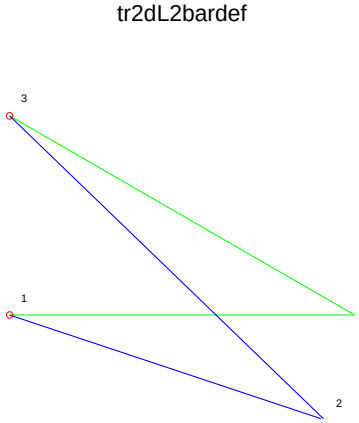
```
eldaC(e,1)  = sine of angle between axis and 1-direction
eldaC(e,2)  = cosine of angle between axis and 1-direction
eldaC(e,3)  = length
eldaC(e,4)  = cross-sectional area
eldaC(e,6)  = linear axial strain
eldaC(e,7)  = axial stress
eldaC(e,11) = axial stretch ratio
eldaC(e,12) = radial stretch ratio
eldaC(e,18) = axial force
```

To see some results for our example just type the following in the Matlab shell :

```
eldaC(1,6)
eldaC(1,7)
eldaC(1,18)
eldaC(2,6)
eldaC(2,7)
eldaC(2,18)
```

These values can ofcourse be stored and printed in various ways.

The above input commands can also be put in one single input file. The results are shown as plot of he deformed structure, where the deformation is enlarged.



3 Matlab program tr2dL

The program tr2dL.m is seeded with comments to explain variables and actions.

```
%*********************************************************************
% tr2dL : 2-dimensional linear truss element
%=====================================================================
%---------------------------------------------------------------------
% Calculate some parameters from the input data.
%  nndof   :  number of nodal degrees of freedom
%  nnod    :  number of nodes
%  ndof    :  number of system degrees of freedom
%  ne      :  number of elements
%  nenod   :  number of element nodes
%  nedof   :  number of element degrees of freedom
%  negr    :  number of element groups
%  lokvg   :  location of degrees of freedom of elements in structure
%---------------------------------------------------------------------

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

lokvg(1:ne,:) = ...
    [ nndof*(lok(1:ne,3)-1)+1 nndof*(lok(1:ne,3)-1)+2 ...
      nndof*(lok(1:ne,4)-1)+1 nndof*(lok(1:ne,4)-1)+2 ];

%---------------------------------------------------------------------
% Calculate transformation matrix 'Trm' for local dof's, if needed.
%---------------------------------------------------------------------

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;
```

```
%-------------------------------------------------------------------------
% Initialization of databases
%  elpa  : element parameters
%  elda0 : initiel values
%  eldaC : current values
%  ety : element type    ;  egr : element group  ;
%  mnr : material number ;  mcl : material class ;  mty : material type
%  l0  : initial element length
%  s0  : sine of axis angle
%  c0  : cosine of axis angle
%-------------------------------------------------------------------------

for e=1:ne
  ety = lok(e,1); egr = lok(e,2);
  mnr  = elda(egr,2);    A0 = elda(egr,3);
  E    = elda(egr,6);    Gn = elda(egr,7);
  mcl  = floor(mnr/10);    mty  = rem(mnr,10);
  k1   = lok(e,3); k2 = lok(e,4);
  x10  = crd0(k1,1); y10 = crd0(k1,2); x20 = crd0(k2,1); y20 = crd0(k2,2);
  l0   = sqrt((x20-x10)*(x20-x10)+(y20-y10)*(y20-y10));
  s0   = (y20-y10)/l0;
  c0   = (x20-x10)/l0;

  elpa(e,:)  = [ety egr nenod nndof nedof];
  elda0(e,:) = [ s0 c0 l0 A0 0 0 Gn E mcl mty ];
  eldaC(e,:) = [ s0 c0 l0 A0 0 0 Gn E 0   0 ];
end; % element loop 'e'

%-------------------------------------------------------------------------
% Boundary conditions are reorganized.
% Additional arrays for later partitioning are made.
%  npdof :  number of prescribed degrees of freedom
%  npfor :  number of prescribed nodal forces
%  nudof :  number of unknown degrees of freedom
%
%  Information for partitioning the system of equations associated
%  with prescribed boundary conditions is made available in the arrays
%  ppc, ppv, pfc  and  pfv.
%-------------------------------------------------------------------------

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);
  ec0 = crd0(lok(e,3:2+nenod),:); ec = ec0;
  em = zeros(nedof);

%-----------------------------------------------------------------------
% Element stiffness matrix

  s   = eldaC(e,1) ; c  = eldaC(e,2);
  ML  = [  c*c  c*s -c*c -c*s ;  c*s  s*s -c*s -s*s
          -c*c -c*s  c*c  c*s ; -c*s -s*s  c*s  s*s ];

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

  l0 = elda0(e,3); A0 = elda0(e,4); E0 = elda0(e,8);

  em = (A0/l0 * E0) * ML ;
  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);
  ec0 = crd0(lok(e,2+1:2+nenod),:);
  ec = crd(lok(e,2+1:2+nenod),:);
  ef = zeros(nedof,1);

%-------------------------------------------------------------------------
% Element internal forces

  s   = eldaC(e,1) ; c  = eldaC(e,2);
  V    = [ -c -s c s ]';

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

  l0 = elda0(e,3); A0 = elda0(e,4);
  E0 = elda0(e,8); Gn0 = elda0(e,7);
  x1 = ec(1,1); y1 = ec(1,2); x2 = ec(2,1); y2 = ec(2,2);
  l  = sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
  s  = (y2-y1)/l;     c  = (x2-x1)/l;
  Gl  = l/l0; Ge = Gl-1;
  Ged = -Gn0*Ge; Gm = Ged+1; A = Gm*Gm*A0; Gs = E0 * Ge; N = A * Gs;

  eldaC(e,1:7)   = [s c l A 0 Ge Gs];
```

```
  eldaC(e,11:18) = [Gl Gm Ge 0 0 Gs 0 N];

  ef = N * V;

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

if ntr>=1
MpT  = reshape(pT,nndof,nnod)';  MTpT = MpT;
MfiT = reshape(fiT,nndof,nnod)'; MfeT = reshape(feT,nndof,nnod)';
MrsT = reshape(rsT,nndof,nnod)';
end;
%-----------------------------------------------------------------------

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