Program tr3d

1 The program `tr3d`

The main command file `tr3d.m` uses a number of command files and function files.

A short explanation of the main file and the most important subfiles is given in this section. Because this program is used as a tool, we will not explain the various command and function files in detail. Attention is more given to making input and generating output. A number of example input files are available, including some for the models of the previous sections. The FE tool `tr3d` can be used in optimization procedures to determine the 'best' prestresses in the cables and other *design variable* values of truss/cable or tensegrity systems.

The program starts with checking input and providing default values for non-specified variables. Also, some things, like e.g. the number of nodes, can be calculated from input data, in this case the array with the nodal coordinates. All input data are stored in a data base to be readily available for later calculations. Finally, a lot of variables are initialized. Variable names ending with `0` have initial values. A variable name ending with `B` refers to a begin-increment value. A variable name ending with `C` indicates the current value, which may be in an iterative loop.

In the incremental loop, the load is incrementally changed, until the end of the analysis. Each increment starts with generating the incremental load with the command file `loadincr.m`, which has to be available and is usually written with Matlab `fprintf` commands. The external incremental force f_e (`fe`) is known at this point and the residual can be (re)calculated.

In the first increment, or when desired for other reasons, the system of equations is (re)generated, i.e. the stiffness matrix and the internal force are calculated in a loop over all elements. For each element the nodal coordinates `ec` are extracted from the coordinate matrix `crd`. In `tr3dgeom.m` matrices $\underline{M}_L$ (`ML`) and $\underline{M}_N$ (`MN`) and the column $\underset{\sim}{V}$ (`V`) are calculated using information from the database `eldaC`. The coefficients `CL` and `CN` are calculated in `tr2dmat.m` according to the material model which is specified for an element. Finally, in the assembling stage, the element matrix data are placed in the structural stiffness matrix as is the case for the internal forces.

We now enter the iterative loop and stay in it until convergence is reached for this increment. This is the case when the convergence norm `nrm` is smaller than the convergence criterion `ccr`. The loop is also finished when the number if iterations `it` exceeds a maximum value `mit`.

The iterative displacements can only be solved when the boundary conditions are taken into account properly. The system of equations is reorganized (*partitioning*) in `fbibpartit.m` and the unknown displacements are solved. Subsequently, the prescribed values are included and possible links relations are looked after. When all this is done we have the iterative displacements `Dp` and the incremental (`Ip`) and total displacements (`Tp`) can be updated.

To calculated element values like strains and stresses and also the internal loads, a new loop over all elements is started. In this loop the stiffness matrix is also calculated again. The internal load is used to calculate the convergence norm. Finally the iteration step conter is updated.

When convergence is reached, the end-increment data are stored and the next increment

can be started. Storage of desired data is done with a call to `savedata.m` which has to be available and is usually written with Matlab `fprintf` commands.

1.1 Examples

1.1.1 Example : a simple two-dimensional mast/cable stucture

In this first example we model and analyze a two-dimensional truss/cable stucture, which is shown in the figure below. In fact it is a three-dimensional model, with all displacements in z-direction suppressed. The structure is modelled with three elements : a truss between node 1 and 2, and two cable elements between node 2 and the nodes 3 and 4. The initial length of the truss is $\ell_0 = 4$ m and its cross-sectional area $A_0 = 100$ mm^2. The cables have a cross-sectional area $A_0 = 10$ mm^2. Nodes 1, 3 and 4 are fixed. The distance between these nodes – the footprint of the mast – is 2 m. In node 2 a force in x-direction is prescribed : $F = 20000$ N..

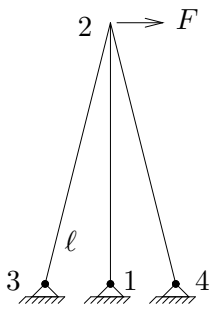

Fig. 1.1 : *Two-dimensional truss/cable structure*

1.1.2 Input

The input file for `tr3d` is `d2t1c2n4a.m` and must be made with an editor of some sort. We will discuss the input here in detail. The complete file is available for download. For later examples we will consider only the input files, with comments included.

Each input file starts with cleaning up the Matlab figures and the Matlab environment. Also some file have to be deleted.

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

The coordinates of the nodal points have to be placed in the array (matrix) `crd0`. Each row in this array stores the x-, y- and z-coordinate of a node.

```
crd0 = [ 0 0 0; 0 4 0; -1 0 0; 1 0 0 ];
```

The element nodes coincide with structural nodes, where they are connected to other elements or to the world. These connections are specified in the array `lok`. For each element e there is a corresponding row in `lok`. In subsequent four columns we find (write) : the element type (`ety`), the element group (`egr`) and finally, the two structural node numbers for element nodes 1 and 2. The element type number is 9 for a truss and 1 for a cable.

```
lok = [ 9 1 1 2; 1 2 3 2; 1 2 4 2 ];
```

Element data are given in the array `elda` for each element group in a separate row. In our case we have two element groups, with different properties, so `elda` has two rows. In each row we find in subsequent columns : the element type, the material number (`mnr`), the initial cross-sectional area A_0, a number 0 (which is not used here), the prestress σ_0, Young's modulus E and Poisson's ratio ν. The final three columns contain zero's. For all our examples the material behavior is linearly elastic, which is represented by material number `mnr = 11`. In our example the prestress is 10000 Pa.

```
elda = [ 9 11 100e-6   0 0 200e9 0.3 0 0 0
         1 11 10e-6  1e4 0 200e9 0.3 0 0 0 ];
```

Now we specify the boundary conditions. First the prescribed displacements in the array `pp` : for each prescribed displacement a row with : the node nmber, the coordinate direction (dof = degree of freedom) and its value.

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

Then the prescribed nodal forces in `pf`.

```
pf = [ 2 1 20000 ];
```

Because the analysis is nonlinear we have to provide some parameters for the incremental/iterative procedure. The number of increments `nic` is for our purposes either 1 or 2. In the first increment the prestress is applied. When we want to see its equilibrium state, we only do the first increment, so `nic = 1`. The actual loading of the prestressed structure is done in the second increment and if we want to analyse this we set `nic = 2`. When `nic` is set, we can run `tr3d`.

```
nic = 2; tr3d;
```

1.1.3 Output

The result of the analysis are available in various array's. The must importants ones are :

```
MTp = [ node dof displacement ]
Mfi = [ node dof internal-forces ]
eldaB(:,7) = element stresses
```

What we definitely need is a plot of the structure. This can be done with :

```
options = [magfac inimsh nodpnt elmnrs nodbcs 0 0 0 0]
plotmesh3(options,lok,crd0,crd,pp,eldaC,1);
view(0,90);
```

In the options array the various variables are :

```
magfac = magnification factor for displacements
inimsh = initial mesh (1 = yes; 0 = no)
nodpnt = nodal numbers (1 = yes; 0 = no)
elmnrs = element numbers (1 = yes; 0 = no)
nodbcs = boundary conditions (1 = yes; 0 = no)
```

In our example we use :

```
plotmesh3([1000000 1 0 0 0 0 0 0 0],lok,crd0,crd,pp,eldaC,1);
```

In the plot trusses are drawn in red color and cables are blue. When a cable slackens, which is the case when the stress becomes zero, the cable color becomes black. You can try this by reducing the prestress from 10^4 to 10^2. In this case the cable slacking does not lead to instability or failure, because the load is still carried by the other cable.

The complete input file is listed below and can also be downloaded.

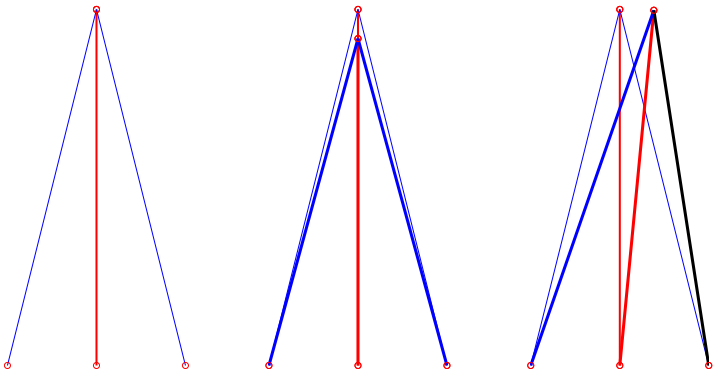

Fig. 1.2 : *Truss/cable mast: initial, prestressed and loaded geometry*

```
%=====================================================================
% d2t1c2n4a.m : two-dimensional mast
% units are 'meter'.

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

crd0 = [ 0 0 0; 0 4 0; -1 0 0; 1 0 0 ];
lok  = [ 9 1 1 2; 1 2 3 2; 1 2 4 2 ];
elda = [ 9 11 100e-6 0   0 200e9 0.3 0 0 0
         1 11 10e-6  1e6 0 200e9 0.3 0 0 0 ];

figure;plotmesh3([0 1 0 0 1 0 0 0 0],lok,crd0,crd0,[],[],1);
view(0,90);

pp = [ 1 1 0; 1 2 0; 1 3 0;
       2 3 0;
       3 1 0; 3 2 0; 3 3 0; 4 1 0; 4 2 0; 4 3 0 ];
pf = [ 2 1 1000 ];

nic=1; tr3d;

figure;plotmesh3([100000 1 0 0 0 0 0 0 0],lok,crd0,crd,pp,eldaC,1);
view(0,90);

nic=2; tr3d;

figure;plotmesh3([10 1 0 0 0 0 0 0 0],lok,crd0,crd,pp,eldaC,1);
view(0,90);
%=====================================================================
```

1.1.4 Example : a truss/cable triangular prism

This example explains the modelling and analysis of a three-dimensional strutt/cable prism. It is shown in the figure below in its undeformed state. The figure also shows the fixed displacements in the bottom plane. The sides of the bottom and top triangles have a length `LLL` = 200 mm. The height of the prism is `HHH` = 300 mm. Trusses have a cross-sectional area $A_0 = 10$ mm^2 and cables $A_0 = 1$ mm^2. Trusses and cables have the same material properties : $E = 100e3$ MPa and $\nu = 0.25$. Cables are given a prestress $\sigma_0 = 1000$ MPa. After prestressing, the structure is loaded with a vertical force $F = -1000$ N in the top-plane nodes.

Figures show the prestress state and the loaded state of the structure. The complete input file `d3t6c9n7a.m` is available for downloading and is listed below.

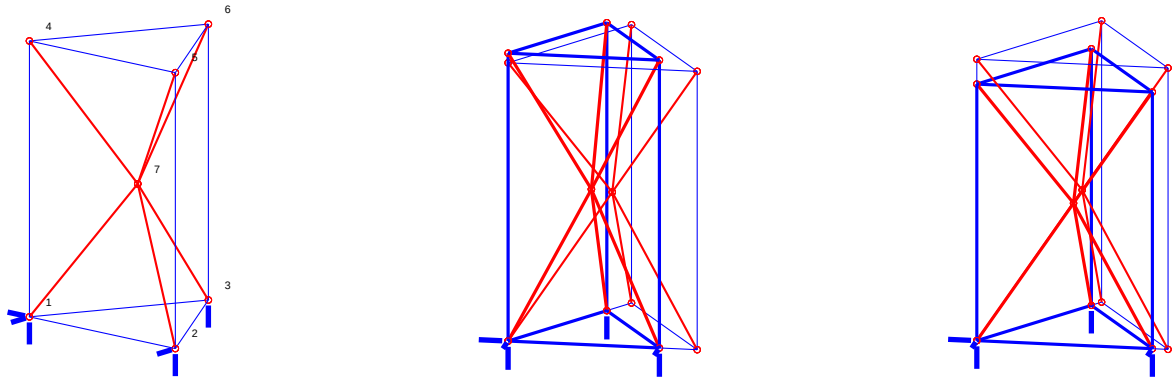

Fig. 1.3 : *Truss/cable triangular prism: initial, prestressed and loaded geometry*

```
%************************************************************************
% d3t6c9n7a.m : Cable-strut prism
% units are 'millimeter'

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

LLL=200; HHH=300; MAG=LLL/5;

crd0 = [ 0        0                  0;
         LLL      0                  0;
         LLL/2    LLL*(sqrt(3))/2    0;
         0        0                  HHH;
         LLL      0                  HHH;
         LLL/2    LLL*(sqrt(3))/2    HHH;
         LLL/2    LLL*(sqrt(3))/6    HHH/2    ];

lok = [
9 1 1 7; 9 1 2 7; 9 1 3 7;       % lower struts
9 1 4 7; 9 1 5 7; 9 1 6 7        % upper struts
1 2 1 2; 1 2 2 3; 1 2 3 1;       % lower cables
1 2 4 5; 1 2 5 6; 1 2 6 4;       % upper cables
1 3 1 4; 1 3 2 5; 1 3 3 6; ];    % vertical cables

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

figure;plotmesh3([0 1 1 0 1 0 0 0 0],lok,crd0,crd0,pp,[],1);
view(40,15);

elda = [ 9 11 10  0   0 100e3 0.25 0 0 0
         1 11  1  1e3 0 100e3 0.25 0 0 0
```

```
         1 11  1  1e3 0 100e3 0.25 0 0 0 ];

pf=[4 3 -1000; 5 3 -1000; 6 3 -1000];

nic=1; tr3d; magf = MAG/max(max(abs(MTp))),
figure;plotmesh3([magf 1 0 0 1 0 0 0 0],lok,crd0,crd,pp,eldaC,1);
view(10,15);

nic=2; tr3d; magf = MAG/max(max(abs(MTp))),
figure;plotmesh3([magf 1 0 0 1 0 0 0 0],lok,crd0,crd,pp,eldaC,1);
view(10,15);

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

1.1.5 Example : a two-dimensional tensegrity

This example model is a real tensegrity and is given here as a basis for the optimization approaches, which will be discussed later. The structure has 3 trusses and 7 cables. The distance between the fixations is 100 mm, the overall length is 150 mm, the height is 100 mm. Trusses have a cross-sectional area $A_0 = 10$ mm^2 and cables $A_0 = 1$ mm^2. Trusses and cables have the same elastic properties : $E = 100e3$ MPa and $\nu = 0.25$. It is not at all obvious how the prestresses in the cables must be chosen to get an equilibrium internal stress state and to prevent cable slacking after loading. For an averall cable prestress $\sigma_0 = 225$ MPa, the figures show the prestresses state (magnification 58) and the loaded state (magnification 9.8) respectively. Loading is done by a force $\vec{F} = 100\vec{e}_1 + 100\vec{e}_2$ in the upper-right corner node 6. The input file `d2t3c7n6a.m` is listed below the figure.

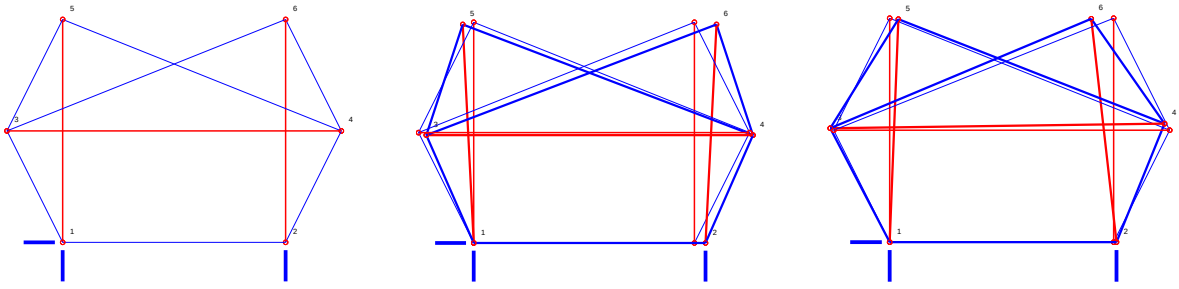

Fig. 1.4 : *Two-dimensional tensegrity: initial, prestressed and loaded geometry*

```
%*********************************************************************
% d2t3c7n6a.m : 2D-tensegrity; units are 'millimeter'
% Comment 'clear' and plotting when used in optimization

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

```
SSS=100;LLL=1.5*SSS;HHH=SSS;

crd0 = [ -SSS/2     0; SSS/2     0;
         -LLL/2 HHH/2; LLL/2 HHH/2;
         -SSS/2   HHH; SSS/2   HHH ];
crd0 = [ crd0 zeros(6,1) ];

lok = [ 9 1 1 5; 9 1 2 6; 9 1 3 4;              % struts
        1 2 1 2;                                % horizontal
        1 3 1 3; 1 3 2 4; 1 3 3 5; 1 3 4 6;     % sides
        1 4 3 6; 1 4 4 5 ];                     % diagonal

%figure;plotmesh3([0 1 1 0 0 0 0 0 0],lok,crd0,crd0,[],[],1);view(0,90);

pp = [ 1 3 0; 2 3 0; 3 3 0; 4 3 0; 5 3 0; 6 3 0 ;
       1 1 0; 1 2 0; 2 2 0; ];

%figure;plotmesh3([0 1 1 0 1 0 0 0 0],lok,crd0,crd0,pp,[],1);view(0,90);

if ~exist('Gs1'), Gs1 = 2.25e2; end;
if ~exist('Gs2'), Gs2 = 2.25e2; end;
if ~exist('Gs3'), Gs3 = 2.25e2; end;

elda = [ 9 11 10 0   0 100e3 0.25 0 0 0         % struts
         1 11 1  Gs1 0 100e3 0.25 0 0 0         % horizontal
         1 11 1  Gs2 0 100e3 0.25 0 0 0         % sides
         1 11 1  Gs3 0 100e3 0.25 0 0 0 ];      % diagonal

%nic=1; tr3d; magf = 10/max(max(abs(MTp)));
%figure;plotmesh3([magf 1 0 0 1 0 0 0 0],lok,crd0,crd,pp,eldaC,1);view(0,90);

pf = [ 6 1 -100; 6 2 100 ];

nic=2; tr3d; magf = 10/max(max(abs(MTp)));
%figure;plotmesh3([magf 1 0 0 1 0 0 0 0],lok,crd0,crd,pp,eldaC,1);view(0,90);

% For optimization we have to calculate some output variables.

sres1 = sqrt(Mfe(6,1).^2 + Mfe(6,2).^2 + Mfe(6,3).^2);
sres2 = sqrt(MTp(6,1).^2 + MTp(6,2).^2 + MTp(6,3).^2);
MSp = MTp - MIp;
sres3 = sqrt(MSp(6,1).^2 + MSp(6,2).^2 + MSp(6,3).^2);
cres1 = eldaC(:,7);

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