

1. Description of the **AmFEM** program

A dedicated computer program was developed in the MATLAB environment and used for all example computations presented in this work. The program can solve four types of problems:

- using flat shell elements only (3D model),
- using thin-walled beam elements only (1D model),
- using spatial joint elements (mixed 1D/3D model),
- using transition elements (mixed 1D/3D model).

Moreover, one of five analysis types can be applied to each problem type:

- linear analysis,
- buckling analysis,
- $p - \Delta$ analysis for small nonlinearities,
- nonlinear analysis tracing the fundamental equilibrium path, with load or displacement control,
- nonlinear analysis with detection of critical points by the bisection method.

The program also provides basic postprocessing:

- manipulation of the structure view (zooming in, zooming out, panning), display of node and element numbers,
- display of the structure in the initial and current configuration,
- marking of loaded nodes,
- marking of nodes with boundary conditions,
- marking of nodes shared by the shell and beam parts of the model,
- display of buckling modes,
- plotting of fundamental equilibrium paths,
- display of internal force diagrams in thin-walled beam elements.

The program is started by issuing the command

```
amfem
```

Listing 1.1. Starting the **AmFEM** program

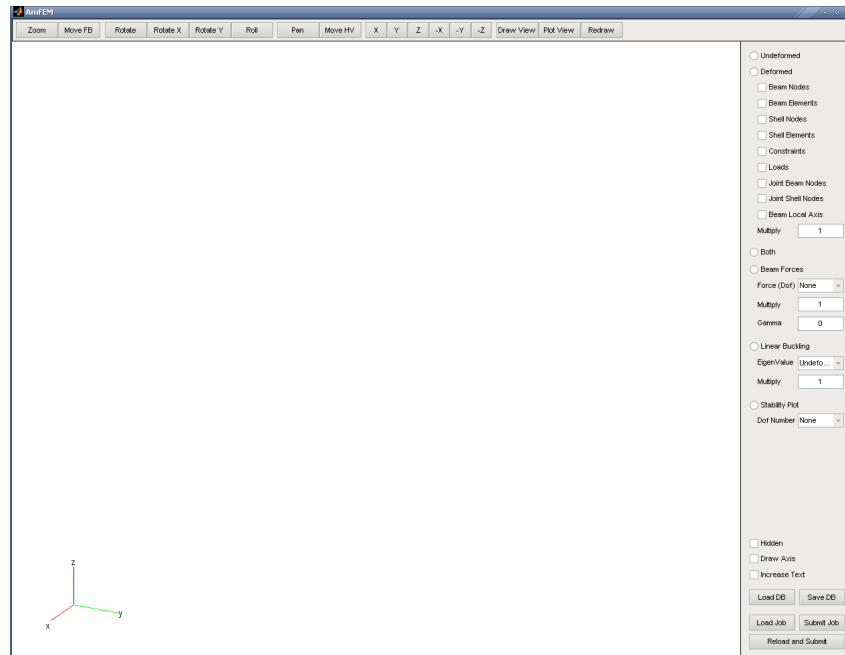
at the MATLAB command line. Once the program is running (Fig. 1.1), a job file should be loaded and the analysis started.

A job file is a MATLAB script (m-file) that initialises the appropriate variable structures. Below is an example job script (split into parts and commented) for a nonlinear analysis using transition elements.

Definition of the job type:

```
Job.Type = 'Transient NonLinear';
```

Listing 1.2. Definition of the job type



Rys. 1.1. The AmFEM program window

The `Type` variable of the `Job` structure consists of two space-separated parts defining the model and the analysis type. The model type is specified by one of the keywords:

- Beam – thin-walled elements only,
- Shell – shell elements only,
- Joint – problem using the spatial joint element,
- Transient – problem using transition elements.

The analysis type is specified by one of the keywords:

- Linear – linear analysis,
- Buckle – buckling analysis,
- Delta – $p - \Delta$ analysis,
- NonLinear – nonlinear analysis,
- Critical – critical point search.

The material definition is placed in the `Material` structure:

```
Material(1).E = 205e9;
Material(1).Er = 225.27e9;
Material(1).Ni = 0.3;
Material(1).G = 78.85e9;
```

Listing 1.3. Definition of material constants

It consists of four fields:

- E – Young's modulus,
- Er – reduced Young's modulus,
- Ni – Poisson's ratio,
- G – shear (Kirchhoff) modulus.

Multiple materials can be defined in a job file by adding further `Material` structures (their consecutive numbers are given in parentheses).

The thickness of a shell element is defined by assigning a value to the `Thickness` field of the `ShellSection` structure:

```
ShellSection(1).Thickness = 0.01;
```

Listing 1.4. Definition of shell element thickness

Multiple thicknesses can be defined in a job file by adding further numbered `ShellSection` structures.

The `BeamSection` structure describes the geometry of the thin-walled cross-section. Its fields define the cross-sectional area, moments of inertia, shear centre location and higher-order moments:

```
BeamSection(1).Area = 6.799999999999963e-003;
BeamSection(1).Jy = 1.024266666666667e-004;
BeamSection(1).Jz = 1.335666666666670e-005;
...
BeamSection(1).Ys = 4.021042851316333e-014;
BeamSection(1).Zs = 1.540261776800476e-014;
...
BeamSection(1).Myooz = 1.468713529473182e-020;
BeamSection(1).Mzooy = -8.762127773378741e-020;
BeamSection(1).Mzooz = -2.782079843262899e-007;
```

Listing 1.5. Definition of a thin-walled element cross-section

Multiple thin-walled cross-sections can be defined in a job file by adding further numbered `BeamSection` structures.

Node coordinates and the connectivity table of shell elements are placed in the `ShellInstance` structure. The `ShellInstance.Nodes` array holds the node coordinates:

```
ShellInstance(1).Nodes = [
    1  4.15 -0.1  4.15
    2  4.15  0    4.15
    3  3.85  0    3.85
    ...
   349 4.15 -0.1  3.6
];
```

Listing 1.6. Definition of shell element node coordinates

and the `ShellInstance.Elements` array holds the finite element connectivity:

```
ShellInstance(1).Elements = [
    1  1  2  19  24  124  125  126  127  1  1
    2  24  19  20  23  126  128  129  130  1  1
    3  23  20  21  22  129  131  132  133  1  1
    ...
   104 32  7  18  89  198  348  349  346  1  1
];
```

Listing 1.7. Definition of shell element connectivity

Apart from connectivity, this array also stores the material and cross-section of each element. These are given in its last two columns, which contain the number of the material structure and

the number of the structure describing the shell thickness. If the model contains more than one shell part, the parts are distinguished by consecutive numbers of the `ShellInstance` structure.

Boundary conditions and loads are defined by the `Constraint` and `Load` structures. The `Constraint` structure has two fields: `Nodes`, containing the numbers of the constrained nodes, and `Dof`, defining the numbers of the degrees of freedom at a node. The example shows the rotational degrees of freedom being fixed at nodes 1, 2 and 3:

```
ShellInstance(1).Constraint(1).Nodes = [1 2 3];
ShellInstance(1).Constraint(1).Dof = [4 5 6];
```

Listing 1.8. Definition of boundary conditions in the shell part

The `Load` structure consists of three fields: `Magnitude`, `Nodes` and `Dof`, containing the load values, node numbers and the numbers of the degrees of freedom at which the load is applied. The example shows a bending moment of 1000 acting at node 69:

```
ShellInstance(1).Load(1).Magnitude = 1000;
ShellInstance(1).Load(1).Nodes = [69];
ShellInstance(1).Load(1).Dof = [4];
```

Listing 1.9. Definition of loads in the shell part

Any number of structures describing boundary conditions and loads can be defined in a job file.

Problems with a shell part require the definition of characteristic nodes belonging to the wall on the beam-element and shell-element sides. This is done with the `BeamNodes` and `Wall` fields of the `ShellInstance` structure. The `BeamNodes` array defines the numbers of the beam nodes associated with the walls.

```
ShellInstance(1).BeamNodes = [37 38];
```

Listing 1.10. Definition of wall beam nodes

The `Wall` structure consists of the fields:

- `Nodes` – numbers of the shell nodes belonging to the wall,
- `TtNodes` – numbers of the nodes defining the direction normal to the wall,
- `BeamSection` – cross-section number,
- `YZOmega` – values of the y , z coordinates and of the warping function ω at the shell nodes, in the principal centroidal axes of the cross-section.

```
ShellInstance(1).Wall(1).Nodes = [17 336 10 ... 13];
ShellInstance(1).Wall(2).Nodes = [15 304 7 ... 11];

ShellInstance(1).Wall(1).TtNodes = [35 36];
ShellInstance(1).Wall(2).TtNodes = [39 40];

ShellInstance(1).Wall(1).BeamSection = 1;
ShellInstance(1).Wall(2).BeamSection = 1;

ShellInstance(1).Wall(1).YZOmega = [
    0.1    0.15    0.015
    0.05    0.15    0.0075
    0      0.15    0
    ...
```

```

    0.1   -0.15  -0.015
];

ShellInstance(1).Wall(2).YZOmega = [
    0.1   0.15   0.015
    0.05  0.15   0.0075
    0     0.15   0
    ...
    0.1   -0.15  -0.015];

```

Listing 1.11. Definition of wall parameters

Beam node coordinates and the connectivity table are defined analogously to shell elements, using the `Nodes` and `Elements` fields of the `BeamInstance` structure:

```

BeamInstance.Nodes = [
    1   0   0   4
    2  0.1  0   4
    ...
    74  4   0   0
];

BeamInstance.Elements = [
    1   1   2   1   1
    2   2   3   1   1
    ...
    72  73  74   1   1
];

```

Listing 1.12. Definition of node coordinates and connectivity of the beam part

Only one beam part can be defined in a job file.

The `Constraint` and `Load` fields define boundary conditions and loads analogously to shell elements. Below, a clamped support fixing all degrees of freedom at nodes 1 and 74 is shown:

```

BeamInstance.Constraint(1).Nodes = [1 74];
BeamInstance.Constraint(1).Dof = [1:7];

```

Listing 1.13. Definition of boundary conditions in the beam part

together with a torsional moment of -2000 applied at node 21:

```

BeamInstance.Load(1).Magnitude = -2000;
BeamInstance.Load(1).Nodes = [21];
BeamInstance.Load(1).Dof = [4];

```

Listing 1.14. Definition of loads in the beam part

The `BeamInstance.ForceElements` field defines the numbers of the elements whose internal force values will be shown in the plot.

```

BeamInstance.ForceElements = [1 74];

```

Listing 1.15. Definition of elements for which internal force values will be displayed

The `Settings` structure defines various parameters of the computation and postprocessing. It can be used to set the scale factors of the plotted finite element models:

```
Settings.DisplacementMultiply = 200;
Settings.ForceMultiply = 0.001;
```

Listing 1.16. Setting the initial scale factors

to include finite rotation effects:

```
Settings.SecondRotation = 'yes';
```

Listing 1.17. Including finite rotation effects in beam elements

or to set the penalty parameter for transition elements:

```
Settings.ConstraintsPenalty = Material(1).E;
```

Listing 1.18. Setting the penalty parameter

This structure also defines the control scheme of the Newton–Raphson method. The `StabilityControl` field selects the control method: load control (value 'force') or displacement control ('displacement'). The `StabilityIncrement` field defines the load or displacement increments in consecutive computation steps. The `StabilityJoint`, `StabilityElement`, `StabilityNodes` and `StabilityDof` fields define the elements, nodes and degrees of freedom for which equilibrium path plots are to be produced:

```
Settings.StabilityControl = 'force';
Settings.StabilityIncrement = [0.1:0.1:1];
Settings.StabilityJoint = 1;
Settings.StabilityElement = 'beam';
Settings.StabilityNodes = [54 54 54 54 54 54];
Settings.StabilityDof = [1 2 3 4 5 6];
```

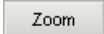
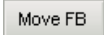
Listing 1.19. Definition of control parameters for the Newton–Raphson method

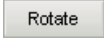
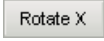
Fig. 1.2 shows the summary after solving a nonlinear problem with transition elements.

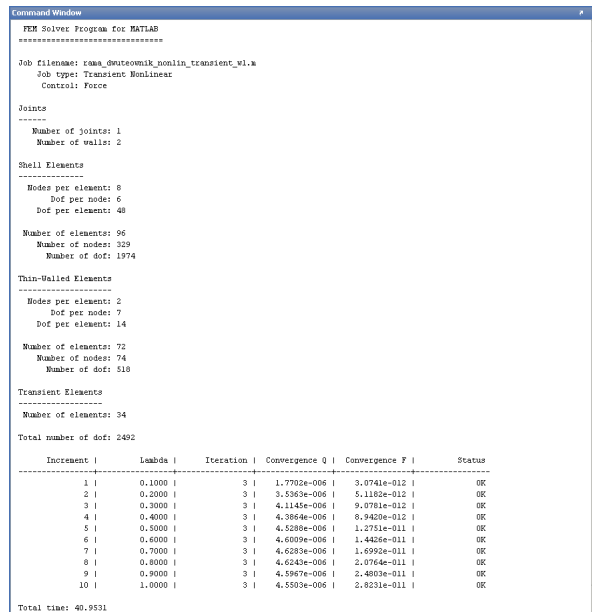
Figs. 1.3 and 1.4 show the **AmFEM** program window with the deformed finite element model and with a plotted equilibrium path, respectively. Fig. 1.5 shows the program window with a buckling mode.

For all problem types, the `BeamOut` and `ShellOut` structures contain a field named `Q` holding the generalised displacements at beam and shell nodes. For buckling problems, in addition to the linear analysis results, further results are stored in the `BeamBuckle` and `ShellBuckle` structures. For each critical load value, a `Mode` structure is created in them, holding the critical load multiplier `EigenValue` and the buckling mode vector `EigenVector`. For nonlinear and $p - \Delta$ problems, apart from the generalised displacements, an output structure `Stability` is also available, containing the equilibrium path results for each node defined in the `Settings` record. Among others, it contains the fields `Inc` and `Q`, which store the load parameter values and the corresponding displacements.

The top toolbar contains buttons for changing the plot view:

- | | |
|---|--|
|  | – zooming the structure view in and out (by changing the camera settings), |
|  | – moving the structure sketch forwards and backwards, |

-  – free rotation of the structure sketch (by changing the camera settings),
-  – rotation of the structure sketch in the screen plane about the horizontal axis (by changing the camera settings),
-  – rotation of the structure sketch in the screen plane about the vertical axis (by changing the camera settings),



Rys. 1.2. Job summary in the MATLAB Command Window

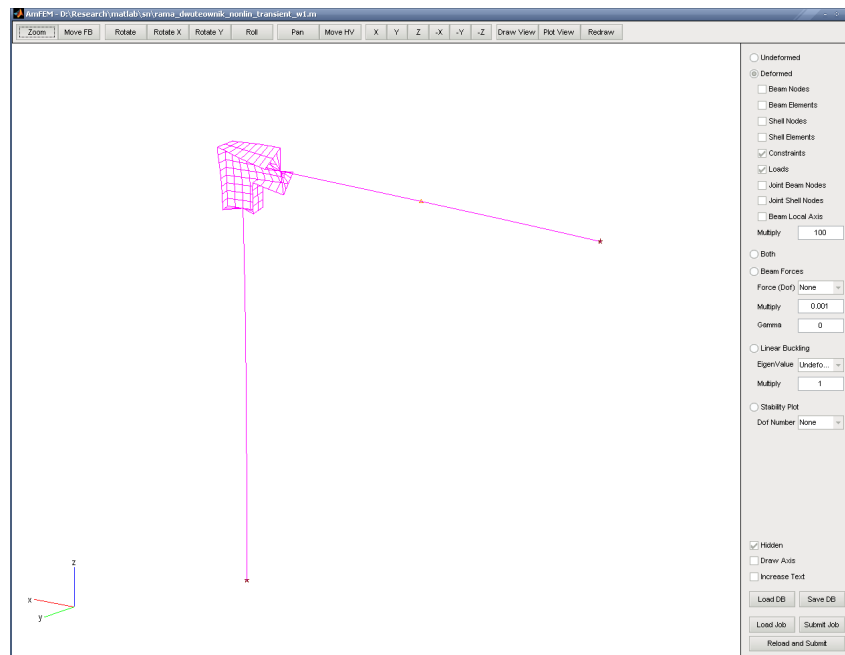
- panning the structure sketch in the screen plane (by changing the camera settings),
- moving the structure sketch horizontally and vertically,
- view along the X or $-X$ axis,
- view along the Y or $-Y$ axis,
- view along the Z or $-Z$ axis,
- default view of the structure sketch,
- default view of the equilibrium path plot,
- screen refresh.

Tools controlling what is drawn are located on the side toolbar, in the following groups:

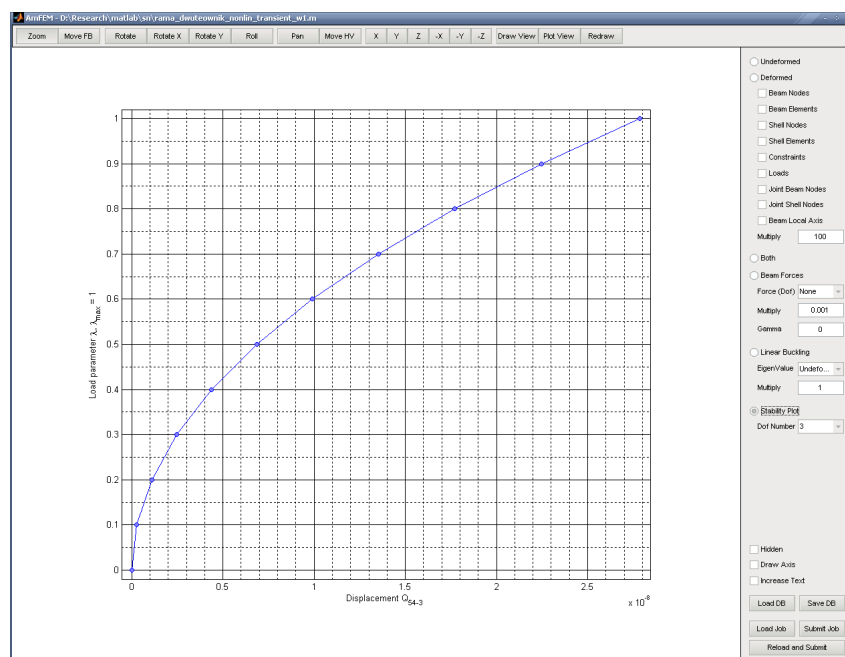
- model in the initial and current configuration:

- ☐ Undeformed – drawing the structure in the initial configuration,
- ☐ Deformed – drawing the structure in the current configuration,
- ☐ Beam Nodes – display of beam node numbers,
- ☐ Beam Elements – display of beam element numbers,
- ☐ Shell Nodes – display of shell node numbers,
- ☐ Shell Elements – display of shell element numbers,
- ☐ Constraints – marking of nodes with boundary conditions,
- ☐ Loads – marking of loaded nodes,
- ☐ Joint Beam Nodes – display of wall beam nodes,
- ☐ Joint Shell Nodes – display of wall shell nodes,
- ☐ Beam Local Axis – display of local coordinate systems of beam elements,
- Multiply – scale factor for the current configuration,
- ☐ Both – drawing the undeformed and deformed structure,

- internal force diagrams in beam elements:



Rys. 1.3. The AmFEM program window with the deformed model

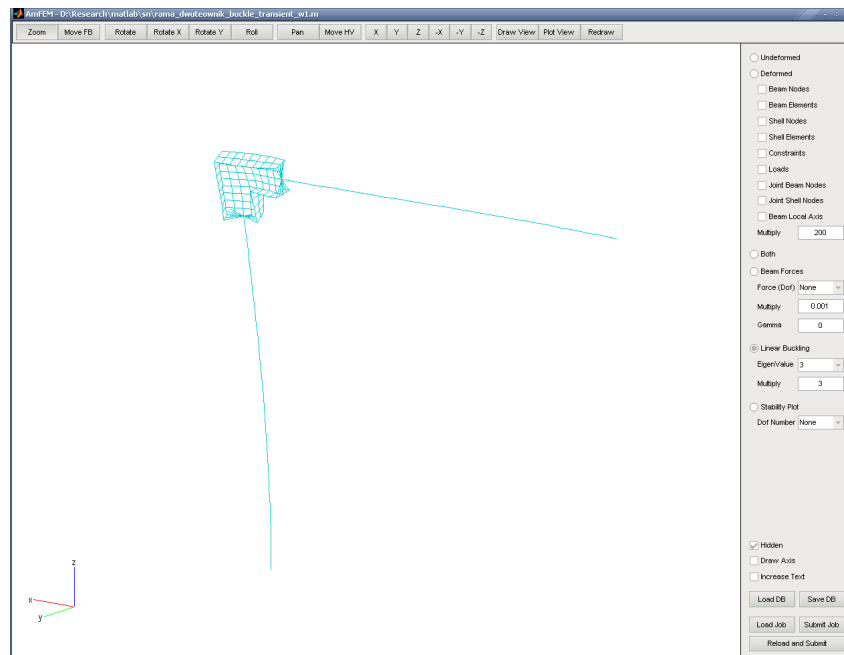


Rys. 1.4. The AmFEM program window with a plotted equilibrium path

— equilibrium path:

- ☐ Stability Plot — enabling the equilibrium path plot,
- Dof Number — number of the degree of freedom shown in the equilibrium path plot,

— in each of the above cases, the following can also be selected:



Rys. 1.5. The AmFEM program window with a buckling mode

- | | | |
|--|---|---|
| ☐ Hidden | – | hiding invisible edges, |
| ☐ Draw Axis | – | sketch of the global coordinate system, |
| ☐ Increase Text | – | enlarging the font used for node and element numbers. |

Job files can be loaded, and analysis results loaded and saved, using the buttons:

- | | | | |
|--|---|---|---|
| <input type="button" value="Load DB"/> | <input type="button" value="Save DB"/> | – | loading and saving analysis results, |
| <input type="button" value="Load Job"/> | <input type="button" value="Submit Job"/> | – | loading a job and running the analysis, |
| <input type="button" value="Reload and Submit"/> | | – | reloading the job and rerunning the analysis. |