Jlinpro command glossary

Enes Šiljak

Contents

1 Set up jlinpro project

Instructions about set up of the project *jlinpro* in Eclipse IDE can be found on my web page http://www.siljak.ba

2 Input file

Input file is text file with extension `.lscr` containing commands that are interpreted by *jlinpro*. All files with extension `.lscr` in directory `mydata` which is in project root directory, are listed in main window combobox. By selecting file in this combobox program opens it, so it will be fast way to open file. For that reason, all examples will be put in folder `mydata`. If you edit file inside *jlinpro*, click on button **Save** to save changes. Input files can be edited with external text editor as well.

Commands are interpreted in class `ui.Interpreter.java` where `ui` is package name. On one line only one command is placed. Line is commented with # sign at the beginning.

```
#commented line
```

Node is defined with `n` followed with x and y coordinates, for example node at $(2, 3)$ is defined as follows:

NODES AND BEAMS

```
n,2,3
```

Cross section is defined with `cs` followed with modulus of elasticity, area and moment of inertia. Optionally, thermal coefficient alpha can be defined at the end.

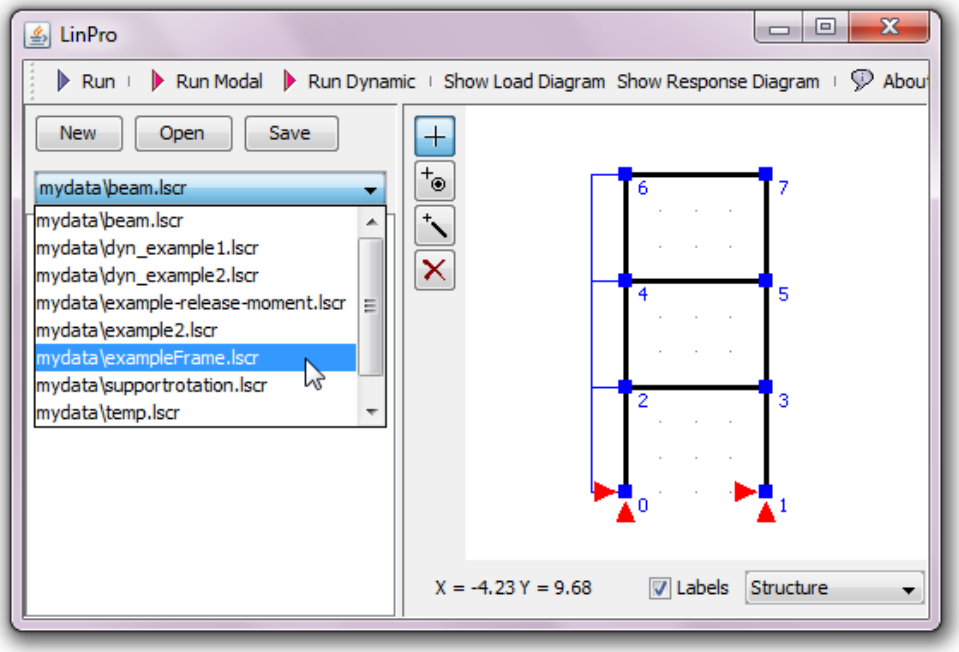


Figure 1: Jlinpro main window. In order to open file given in listing (1) select `mydata\exampleFrame.lscr` from combo box in main window.

```
cs,2E8,0.01,0.0001,1.2E-5
```

Beam is defined with `beam2D` followed with first and second node of the beam. The most simple case is definition of beam with only two nodes. Beam takes first defined cross section (Note that there must be at least one cross section defined):

```
beam2D,1,2
```

If you want to define beam with some other cross section, add previously defined cross section index after second node. (Note that all arrays are zero based). For example, if you have defined two cross sections

```
cs,2E8,0.01,0.0001
cs,3E7,0.021,0.00018,1.2E-5
```

Indices of these cross sections are 0 and 1 respectively. So, if you want to define beam from node 2 to 3, with the second cross section you would write:

```
beam2D,2,3,1
```

If you want to define beam with releases at the ends add word `release` after nodes or cross section, followed with one or more of the following: `Ni, Qi, Mi, Nj, Qj, Mj`. For example beam with released moments on both ends:

```
beam2D,5,3,release,Mi,Mj
```

LOADS

Point force is defined with

```
f,FX,FY,MOM,local[global],A,relative[absolute]
```

where `f` is key word, `FX, FY, M` are force components, `local` or `global` coordinate system in which `FX, FY` act, `A` length from i node which can be `relative` or `absolute`. For example

Until now, I have implemented concentrated force, uniform load in local y direction, general distributed load in local y direction and temperature load.

```
f,0,-100,0,global,0.5,relative
```

would be definition of vertical force of -100 acting in the middle of the beam.

Uniform load is defined with `py` followed with its intensity. For example

```
py,-10
```

defines uniform load of -10.

General distributed load in local y direction is defined with

```
gpy,a,b,Absolute[Relative],pa,pb
```

where `a` is distance of the beginning of the load from node i, `b` is distance of the end from node i, argument `Absolute` or alternatively `Relative` relates to distances and arguments `pa` and `pb` are load intensity at the beginning and at the load end. For example load from relative 0.2 distance to relative 0.6 of the intensity -5 at the beginning and -10 at the end would be defined as

```
gpy,0.2,0.6,Relative,-5,-10
```

Temperature is defined with

```
temp,dT0,dT/h
```

where `dT0` is value of uniformly distributed temperature change along cross section, and `dT/h` is linearly distributed temperature change.

LOAD APPLICATION

After definition, loads must be applied to elements. Application of the load is done with command `al` followed by load index and element index. For example first defined load is applied on the third defined beam with

```
al,0,2
```

since arrays are zero based. Application of concentrated force on node is done with

```
aln,LOAD,NODES
```

SUPPORTS

Supports are modelled with `spring` elements.

```
spring,NODE,LOCALDOF,VALUE
```

where `spring` is keyword, `NODE` is node index on which we want to apply support, `LOCALDOF` is local degree of freedom in which we want to apply support which can be $0, 1$ or 2, for x, y and rotation respectively, and `VALUE` is stiffness value, for fixed support we can use some very high value, for example $1E10$. For example, if we want to apply vertical support on node 3 command would be:

```
spring,3,1,1E10
```

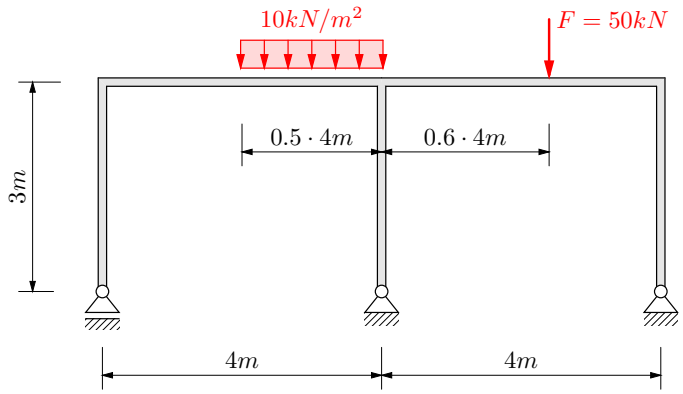
Rotation of node is applied with command `csrotation` followed by node index and angle of rotation in degrees.

```
csrotation,5,30
```

3 Example 1

Finally, input file for simple frame shown in figure (2) is given in listing (1)

This and some other examples can be found in directory `mydata`



$E = 2 \cdot 10^8 kN/m^2$
$A = 1 \cdot 10^{-2} m^2$
$I = 1 \cdot 10^{-4} m^4$

Figure 2: Example 1

Listing 1: Input file for structure example 1.

```
1  dimension,3
2  #nodes
3  #n,X,Y
4  n,0,0
5  n,4,0
6  n,8,0
7  n,0,3
8  n,4,3
9  n,8,3
10 #
11 #cross sections
12 #cs,E,A,I[,ALPHA]
13 cs,2E8,1E-2,1E-4,1.2E-5
14 #
15 #elements
```

```
#beam2D,NODEI,NODEJ[,CS,release,Ni,Qi,Mi,Nj,Qj,Mj]
beam2D,0,3
beam2D,1,4
beam2D,2,5
beam2D,3,4
beam2D,4,5
#
#supports
#spring,NODE,LOCDOF,VAL
#
spring,0,1,1E10
#
spring,1,0,1E10
spring,1,1,1E10
#
spring,2,0,1E10
spring,2,1,1E10
#
#loads
#py,VAL
#gpy,A,B,relative[absolute],PA,PB
#f,FX,FY,MOM,local[global],A,relative[absolute]
#temp,dT0,dT/h
gpy,0.5,1,relative,-10,-10
f,0,-50,0,local,0.6,relative
#
#load application
#al,LOAD,BEAM
al,0,3
al,1,4
```

Resulting moment diagram for this structure is given on figure (3)

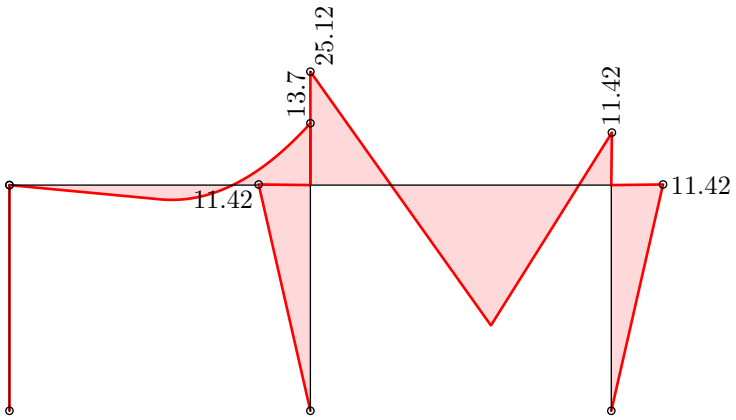


Figure 3: Moment diagram for example 1

4 Results

Diagrams of cross section forces, reactions and deflection line are drawn selecting option at the bottom of the main window, see picture (4). Numerical values are printed on the system output.

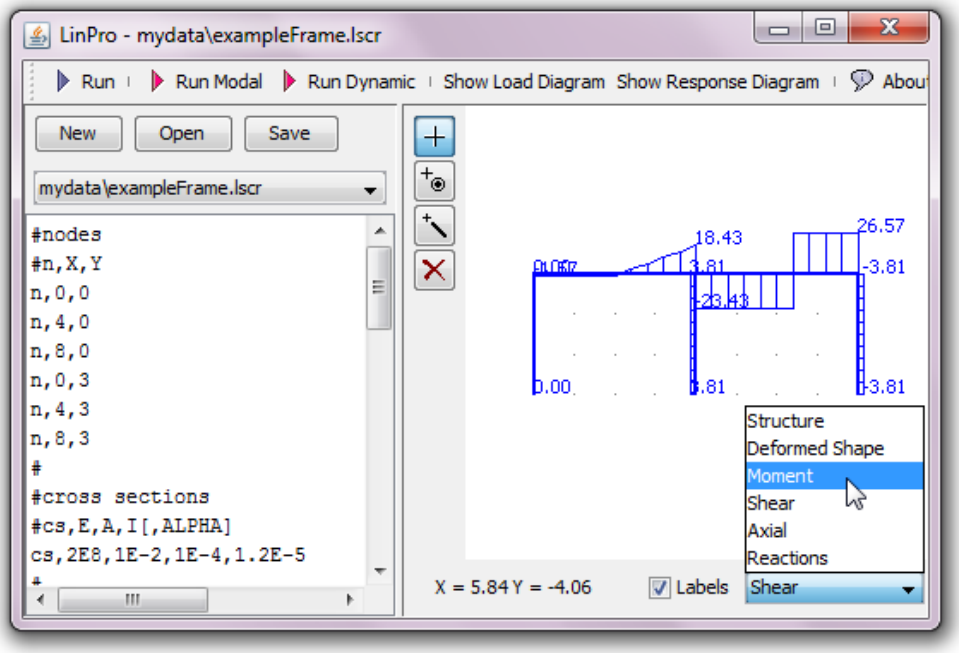


Figure 4: Drawing diagram of cross section forces.

5 Make change in input file

Now let's make some changes in input file. For example we will make pinned support on node 0, and release moment at node 4 on the beam between nodes 4 and 5. Find vertical support definition `spring,0,1,1E10` and add horizontal support at node 0:

```
spring,0,0,1E10
```

Now node 0 is pinned. Find beam definition `beam2D,4,5` and add `,release,Mi` so that it is defined as

```
beam2D,4,5,release,Mi
```

Click on button **Save**. Click on button **Run**.

Moment diagram after this change is given in figure (5)

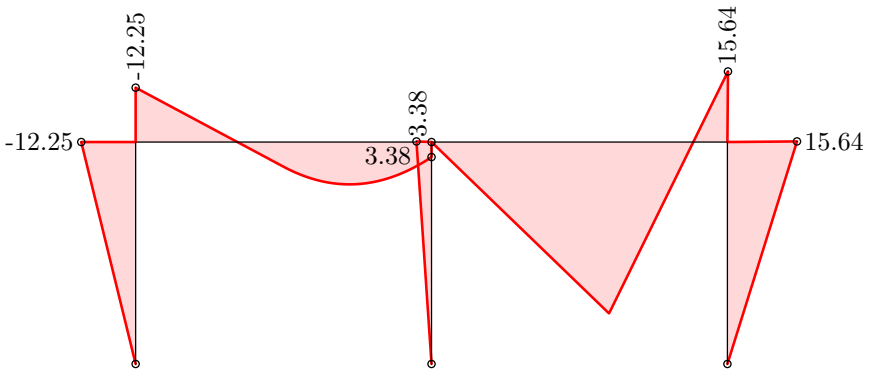


Figure 5: Example 2

6 Print internal calculation results

Now we will print stiffness matrix of each beam element. In Eclipse Package manager open source file `Structure.java` in package `structure` and find method `solveStatic()`.

In method `solveStatic()` find part of code that assembles global stiffness matrix. That code is given in listing (2). On the line 47 element stiffness matrix and load vector are formed. Immediately after that we will add line 48 to print element stiffness matrix.

Listing 2: Assembling global stiffness matrix, in class `Structure`

```
46 for(int k = 0; k < elements_.size(); k++){
47  elementSystem = elements_.get(k).getElementSystem();
48  Mat.printMatrix("Element " + k, elementSystem.stiffnessMatrix);
49  dofs = elements_.get(k).getDOF();
50  for (int i = 0; i < dofs.length; i++){
51   for (int j = 0; j < dofs.length; j++)
52    solver.add(dofs[i], dofs[j], elementSystem.stiffnessMatrix[i][j]);
53
54  RHS[dofs[i]] += elementSystem.loadVector[i];
55  }
56 }
```

Now run calculation and in Eclipse Console view stiffness matrices are printed.