MANUAL
MATLAB CODE FOR PLANAR FINITE ELEMENTS USING FRAME ELEMENTS

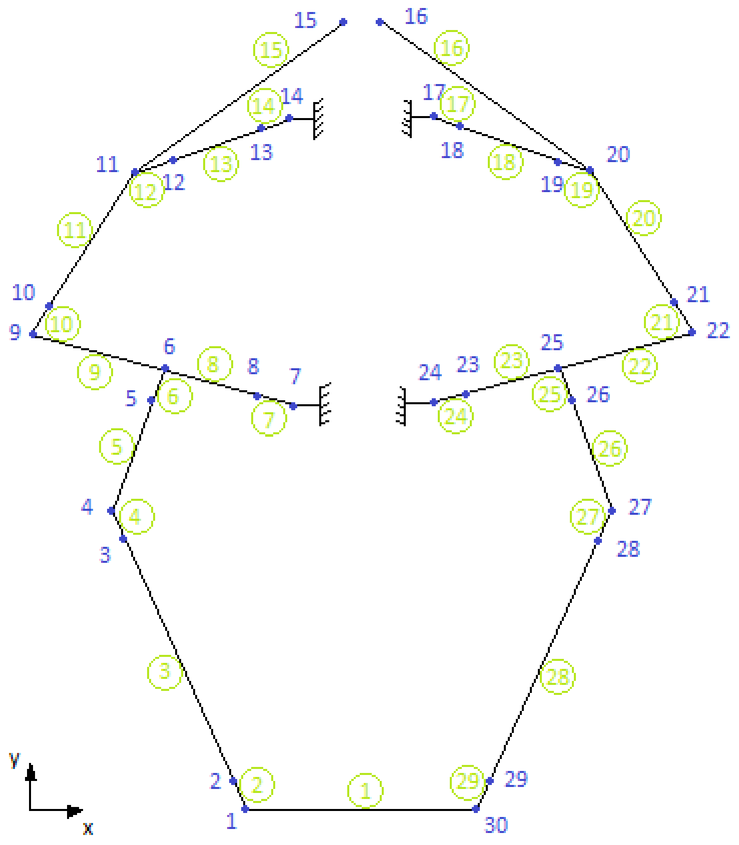
Paulina Correa Mejía
Isabel Cristina López Giraldo

Universidad EAFIT
Finite Element Method
November 2015

Setting a FE problem requires the definition of two items: The nodes and the elements. Nevertheless, the script is based on a node, element, displacement and force programming.

The pseudocode is at the end of this manual, so if it is required to understand the programming and the code in general, you can refer to it.

The frame to be exemplified is the following:



Once you have your frame defined, you create a matrix with the nodes' information, where each row represents each node and the columns represent the [index, x_coordinate, y_coordinate].

```
nodes = [1   48.64   0;
         2   45      29.63;
         3   20.37   230.20;
         4   16.56   261.25;
         5   25.04   274.15;
         6   29.24   279.52;
         7   55.21   271.09;
         8   47.71   273.52;
         9   0       289.03;
         10  0.37    293.27;
         11  5.38    357.57;
         12  29.26   364.87;
         13  54.58   372.61;
         14  70.89   377.6;
         15  64.79   447.29;
         16  102.56  447.29;
         17  94.46   377.6;
         18  112.77  372.71;
         19  138.1   364.87;
         20  161.98  357.57;
         21  166.98  293.27;
         22  167.35  289.03;
         23  119.64  273.52;
         24  112.14  271.09;
         25  139.51  279.52;
         26  142.31  274.15;
         27  150.79  261.25;
         28  146.98  230.2;
         29  122.35  29.63;
         30  118.71  0];

     nodes(:,2) = nodes(:,2) * 10^-3;
     nodes(:,3) = nodes(:,3) * 10^-3;
```

Create a matrix with the elements' information, where each row represents the element and the columns represent the [*index, node1_index, node2_index, area, inertia, young_modulus*].

Assuming a young modulus 172 MPa (PE high density) and the two areas and inertias shown below.

```
A1 = (1.5 * 10^-3) * (2 * 10^-3);
A2 = (1.5 * 10^-3) * (8 * 10^-3);

I1 = ((1.5 * 10^-3) * (2 * 10^-3)^3) / 12;
I2 = ((1.5 * 10^-3) * (8 * 10^-3)^3) / 12;

E = 172 * 10^6;
```

```
elements = [1   1   30  A2  I2  E;
            2   1   2   A1  I1  E;
            3   2   3   A2  I2  E;
            4   3   4   A1  I1  E;
            5   4   5   A2  I2  E;
            6   5   6   A1  I1  E;
            7   7   8   A1  I1  E;
            8   8   6   A2  I2  E;
            9   6   9   A2  I2  E;
            10  9   10  A1  I1  E;
            11  10  11  A2  I2  E;
            12  11  12  A1  I1  E;
            13  12  13  A2  I2  E;
            14  13  14  A1  I1  E;
            15  11  15  A2  I2  E;
            16  20  16  A2  I2  E;
            17  18  17  A1  I1  E;
            18  19  18  A2  I2  E;
            19  20  19  A1  I1  E;
            20  21  20  A2  I2  E;
            21  22  21  A1  I1  E;
            22  25  22  A2  I2  E;
            23  23  25  A2  I2  E;
            24  24  23  A1  I1  E;
            25  26  25  A1  I1  E;
            26  27  26  A2  I2  E;
            27  28  27  A1  I1  E;
            28  29  28  A2  I2  E;
            29  30  29  A1  I1  E];
```

Create a matrix with the boundary conditions' information, where each row represents each boundary condition and the columns represent the [*index, node_index, dof, disp_value*]. Dof is the direction in which the constraint is applied, thus 1 is a constraint in the x direction, 2 for y and 3 for rotations in z.

```
dispbc = [1   1   2  -(10 * 10^-3);
          2   7   1  0;
          3   7   2  0;
          4   7   3  0;
          5   14  1  0;
          6   14  2  0;
          7   14  3  0;
          8   17  1  0;
          9   17  2  0;
          10  17  3  0;
          11  24  1  0;
          12  24  2  0;
          13  24  3  0];
```

Create a matrix with the applied forces' information, where each row represents each force and the columns represent the [*index*, *node_index*, *dof*, *force_value*]. Dof is the direction in which the force is applied, thus 1 is a force in the x direction, 2 for y and 3 for rotations in z.

```
forces = [0 0  0  0];
```

Now we proceed to create a main script where you will copy the four matrices above as the entry parameters. And when it has been created, we can plot the structure.

```
plotNodes(nodes)
hold on
plotElements(elements,nodes)
axis equal
```

Once the frame is plotted, we continue to call the following functions getting the length of each element, the angle for each element, the global K, the global U, the global F; and the matrix of stresses that contains in its first column the index of the element, in its second the axial stress, in its third the shear stress, and it its four the bending stress.

```
[L, Theta] = GeomProp(elements,nodes);

[globalK] = CalculateK (elements,nodes,dispbc);

[globalU, globalF] = SolveMet(elements,nodes,dispbc,forces);

[stresses] = Stresses(nodes,elements,dispbc,forces);
```

Done this, you have gotten all the information needed to determine the Finite Element Analysis.

In each function's script there is a comment specifying its function.