Lehigh University

Advanced Technology for Large Structural Systems

CEE 466 – Advanced Finite Element Methods

User Manual – Finite Element Analysis Program

By: Soheil Sadeghi

Instructor: Dr. Paolo Bocchini

October 2017

The finite element code is organized as a main file and several subroutines that are called through the main file. The main file is Main_file.m in the package. The operator needs to enter all inputs in four different section which are introduced in the following part. The manual is designed in a way to scan over all parts of the program using the fifth verifying example as the illustrative. The problem definition of this example is shown below:

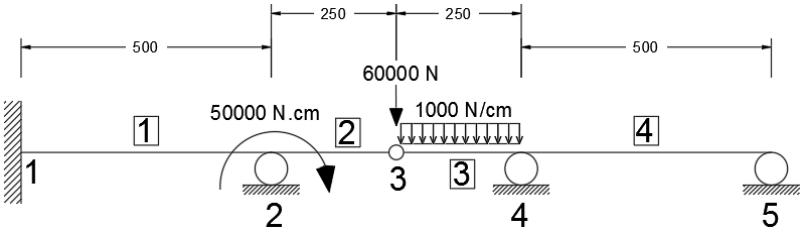

Figure 1. Example five layout

The first INPUT file that has to be completed beforehand is the nodal_coordinate.m file. In this file the coordinates for all nodes should be entered. In addition, if any nodal loads are applied, these should be introduced as a nodal load matrix p. Each row in p consists three values, indicating nodal forces in x, y and rotational direction. Note that the operator should be consistent in units they use. As a presumed unit set, N and centimeters are used for all the verifying examples. For the illustrative example, the input is as follows:

```
x = [0     500     750     1000    1500];
y = [0     0       0       0       0   ];

p = [     0    0          0
          0    0          50000
          0    -60000     0
          0    0          0
          0    0          0     ];
```

The next step to input data is to assign elements properties. The program is set in a way that the operator just needs to call labels associated to the desired material or section and the program automatically call all parameters needed from the designated libraries. The material library is material.m that has all material definitions needed for the verifying examples. No need to emphasize that if a new material is needed, the operator should add it to the library and then call. The same approach is used for section properties. Note that to assign materials, element types and sections, the operator just needs to fill three parameters in the Main_file.m file. For the demonstrating example here, the following materials and sections are called:

```
material_ID = [3,3,3,3];                          % Enter IDs using labels in
'material.m' (material library)
element_type = [2,2,2,2];                         % Enter element types (1:
BAR, 2: TIMOSHENKO, 3: BERNOULLI, 21: FEM-TIMO)
element_prop_ID = [2,2,2,2];                      % Enter prop_ID labels from
sec_prop.m (mechanical props library)
```

In the material.m file, a material with label 3 is defined as follows:

```
        case 3
             MAT(i,1) = 21000000;        % Elastic Modulus (N/cm2)
             MAT(i,2) =0.29;             % Poisson's ratio
```

In which MAT(:,1) is the elastic modulus and MAT(:,2) is the poisson's ratio.

In addition, the section 2 which is used in this example is defined in the sec_prop.m library as follows:

```
        case 2                  % for W sections
            % w16x31
            I_ele(i) = 15608.678;
            area_ele(i) = 58.8386;
            areaS_ele(i) = 28.1741;
```

I_ele, area_ele and areaS_ele are respectively the moment of inertia, gross area and shear area of the section. As the operator proceeds on the Main_file.m script, there is the second INPUT file that has to be filled, which is conn.m. This .m file gets inputs of element connectivities, nodal constraints (supports) and also nodal displacement. For the illustrative example, this file has filled up as follows:

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

supp =  [1   1   1   1
         2   0   1   0
         3   0   0   0
         4   0   1   0
         5   0   1   0];

nodal_disp = [1   0   0   0
              2   0   0   0
              3   0   0   0
              4   0   0   0
              5   0   0   0];
```

The conn matrix contains the connectivity information for each elements in a row wise manner. For example, the second row is [2 1 1 1 1 1 0]. First value is the node ID and the first three values after ID are the connectivity switch in each of three dimension (x, y, rotation) for the starting tip of the element. Accordingly, the last three values are the connectivity switch for the ending tip of the element. As the example definition offers, the second element is fully connected on its left side to the node 2 and is translationally connected to the node 3 on its right. So, rotation is not connected on the right side and that is the reason for the last zero in the connectivity data of the second element.

The sup matrix defines nodal external constraints. For instance, [4 0 1 0] which is the fourth row, tells that the node 4 is free to move in x and rotate, but it is constrained to move vertically (in y). It complies with what the example statement offers.

The nodal_disp defines nodal applied displacements if there is any. In this example there is no externally imposed displacements, therefore the matrix consists all zeros.

The last INPUT file that has to be completed by the operator is ele_nodes.m. In this file the operator defines elements positions and directions, by defining starting and ending nodes for each element. For the example here, the following script works:

```
st_node =  [1   2   3   4];          % starting nodes for elements
end_node = [2   3   4   5];          % ending nodes for elements

DL = [   0   0       0
```

```
          0   0       0
          0   -1000   0
          0   0       0      ];      % Distributed Load for elements - row per
element (parallel, transverse, moment)
```

The st_node and end_node vectors include all starting points and ending points for elements in the model respectively. For example, the third element is placed between node 3 and 4 as the figure shows.

The DL matrix contains all distributed loads over the elements. This matrix has the same number of rows as the elements in a model. Here there are four elements, so the matrix has four rows. The three values at each node indicate distributed axial, transverse and moment loads on each element. As an example, the third row tells that there is a downward distributed load on the third element. Note that the element order is consistent with the order of starting and ending nodes.

The extra features that I have implemented in my program are: 1) Imposed nodal displacements, 2) Automeshing and 3) Internal disconnections.

For the automeshing feature, the operator needs to define the maximum size of mesh. This parameter can be entered in the beginning part of the ele_nodes.m file. It is obvious that the parameter is trivial for all other type of elements than Finite Element Timoshenko element.

In addition to this brief manual, it has been also tried to explain all parts of the main file and all the supporting subroutines by local comments throughout the scripts.