2020

Calculations on the Bridge Over the Pauzeński Canal

Doğu İlmak / Computer Methods

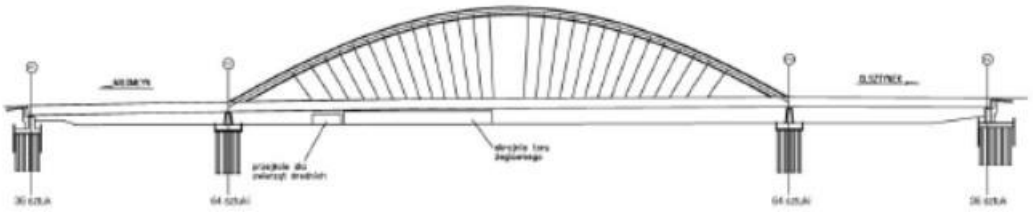
University of Warmia and Mazury in Olsztyn

14.06.2020

Content:

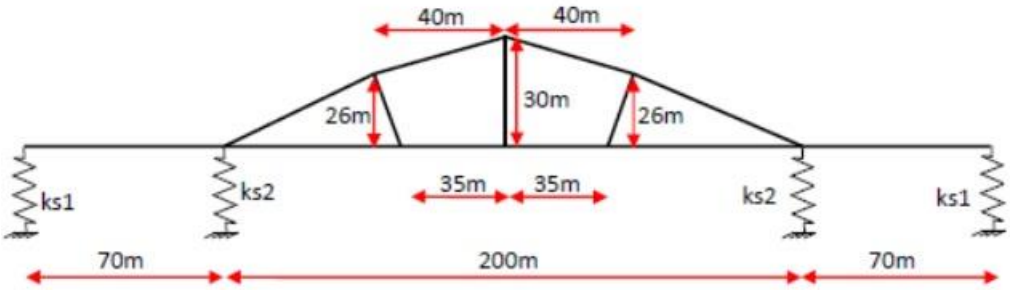
Description

(Fig.1 – Bridge over the Pauzenski Canal)



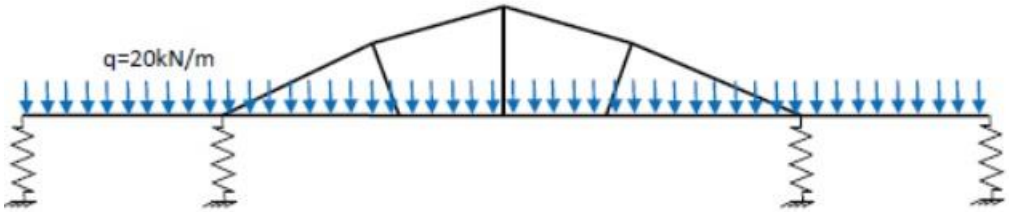
In this work, we are going to determine internal forces(M, N, T) and reactions of supports in the bridge structure using spring and plane frame elements.

On the below you can see some spesific features about the bridge:



(Fig.2 – Lengths of elements)

We can determine the distances between nodes (lenghts of the elements) in Fig.2.



(Fig.3 – Force on bridge over the Pauzenski Canal)

We can see the force on the bridge as kN/m. As you can see from the Fig.3 our force is 20kN/m.

What are internal forces?

Internal forces are produced from the external forces acting on structure members such as poled, beamd or columnd. Generally, we have three types of internal forces: axial, dhear and moment. Axial force, sometimes called 'normal force,' is a compression or tension force acting aligned with the extension of a structure member. Shear force is a force acting in a direction perpendicular to the alignment of the member.

Moment force, lastly, is a turning result of a force multiplied by the distance from its acting location to the turning point. The number of these components varies in one-dimensional, two-dimensional and three-dimensional cases. Now the questions are what each of these components do and how to calculate these internal forces.

What is spring element?

The spring element is a one-dimensional finite element where the local and global coordinates coincide. It should be noted that the spring element is the simplest finite element available.

Necessery functions for calculading spring elements written on Octave:

1. SpringElementStiffness

```
function y = SpringElementStiffness(k)
%SpringElementStiffness   This function returns the element stiffness
%                         matrix for a spring with stiffness k.
%                         The size of the element stiffness matrix
%                         is 2 x 2.
y = [k -k ; -k k];
```

(Fig.4 - SpringElementStiffness on Octave)

2. SpringAssemble

```
function y = SpringAssemble(K,k,i,j)
%SpringAssemble   This function assembles the element stiffness
%                 matrix k of the spring with nodes i and j into the
%                 global stiffness matrix K.
%                 This function returns the global stiffness matrix K
%                 after the element stiffness matrix k is assembled.
K(i,i) = K(i,i) + k(1,1);
K(i,j) = K(i,j) + k(1,2);
K(j,i) = K(j,i) + k(2,1);
K(j,j) = K(j,j) + k(2,2);
y = K;
```

(Fig.5 - SpringAssemble on Octave)

What is plane frame element?

The plane frame element is a two-dimensional finite element with both local and global coordinates. The plane frame element has modulus of elasticity E, moment of inertia I, cross-sectional area A, and length L. Each plane frame element has two nodes and is inclined with an angle θ measured counterclockwise from the positive global X axis.

Necessery functions for calculading plane frame elements on Octave:

1. PlaneFrameElementLenght

```
function y = PlaneFrameElementLength(x1,y1,x2,y2)
%PlaneFrameElementLength   This function returns the length of the
%                          plane frame element whose first node has
%                          coordinates (x1,y1) and second node has
%                          coordinates (x2,y2).
y = sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1));

```

(Fig.6 SpringElementStiffness on Octave)

2. PlaneFrameElementStiffness

(Fig.7 - PlaneFrameElementStiffness on Octave)

```
function y = PlaneFrameElementStiffness(E,A,I,L,theta)
%PlaneFrameElementStiffness   This function returns the element
%                             stiffness matrix for a plane frame
%                             element with modulus of elasticity E,
%                             cross-sectional area A, moment of
%                             inertia I, length L, and angle
%                             theta (in degrees).
%                             The size of the element stiffness
%                             matrix is 6 x 6.
x = theta*pi/180;
C = cos(x);
S = sin(x);
w1 = A*C*C + 12*I*S*S/(L*L);
w2 = A*S*S + 12*I*C*C/(L*L);
w3 = (A-12*I/(L*L))*C*S;
w4 = 6*I*S/L;
w5 = 6*I*C/L;
y = E/L*[w1 w3 -w4 -w1 -w3 -w4 ; w3 w2 w5 -w3 -w2 w5 ;
   -w4 w5 4*I w4 -w5 2*I ; -w1 -w3 w4 w1 w3 w4 ;
   -w3 -w2 -w5 w3 w2 -w5 ; -w4 w5 2*I w4 -w5 4*I];

```

3. PlaneFrameAssemble

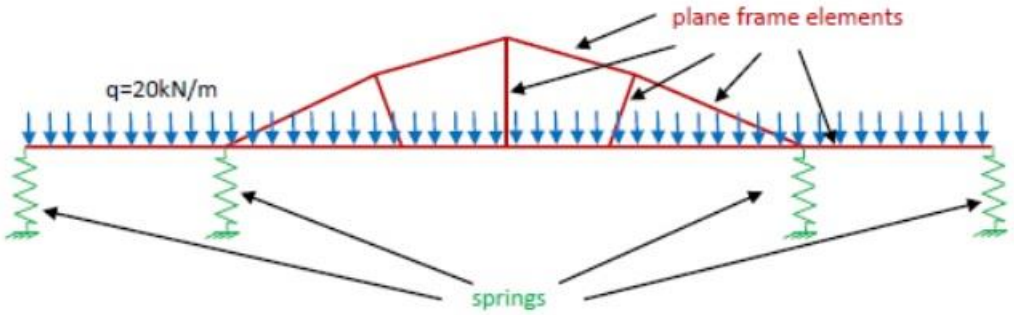
```
function y = PlaneFrameAssemble(K,k,i,j)
%PlaneFrameAssemble   This function assembles the element stiffness
%                     matrix k of the plane frame element with nodes
%                     i and j into the global stiffness matrix K.
%                     This function returns the global stiffness
%                     matrix K after the element stiffness matrix
%                     k is assembled.
K(3*i-2,3*i-2) = K(3*i-2,3*i-2) + k(1,1);
K(3*i-2,3*i-1) = K(3*i-2,3*i-1) + k(1,2);
K(3*i-2,3*i) = K(3*i-2,3*i) + k(1,3);
K(3*i-2,3*j-2) = K(3*i-2,3*j-2) + k(1,4);
K(3*i-2,3*j-1) = K(3*i-2,3*j-1) + k(1,5);
K(3*i-2,3*j) = K(3*i-2,3*j) + k(1,6);
K(3*i-1,3*i-2) = K(3*i-1,3*i-2) + k(2,1);
K(3*i-1,3*i-1) = K(3*i-1,3*i-1) + k(2,2);
K(3*i-1,3*i) = K(3*i-1,3*i) + k(2,3);
K(3*i-1,3*j-2) = K(3*i-1,3*j-2) + k(2,4);
K(3*i-1,3*j-1) = K(3*i-1,3*j-1) + k(2,5);
K(3*i-1,3*j) = K(3*i-1,3*j) + k(2,6);
K(3*i,3*i-2) = K(3*i,3*i-2) + k(3,1);
K(3*i,3*i-1) = K(3*i,3*i-1) + k(3,2);
K(3*i,3*i) = K(3*i,3*i) + k(3,3);
K(3*i,3*j-2) = K(3*i,3*j-2) + k(3,4);
K(3*i,3*j-1) = K(3*i,3*j-1) + k(3,5);
K(3*i,3*j) = K(3*i,3*j) + k(3,6);
K(3*j-2,3*i-2) = K(3*j-2,3*i-2) + k(4,1);
K(3*j-2,3*i-1) = K(3*j-2,3*i-1) + k(4,2);
K(3*j-2,3*i) = K(3*j-2,3*i) + k(4,3);
K(3*j-2,3*j-2) = K(3*j-2,3*j-2) + k(4,4);
K(3*j-2,3*j-1) = K(3*j-2,3*j-1) + k(4,5);
K(3*j-2,3*j) = K(3*j-2,3*j) + k(4,6);
K(3*j-1,3*i-2) = K(3*j-1,3*i-2) + k(5,1);
K(3*j-1,3*i-1) = K(3*j-1,3*i-1) + k(5,2);
K(3*j-1,3*i) = K(3*j-1,3*i) + k(5,3);
K(3*j-1,3*j-2) = K(3*j-1,3*j-2) + k(5,4);
K(3*j-1,3*j-1) = K(3*j-1,3*j-1) + k(5,5);
K(3*j-1,3*j) = K(3*j-1,3*j) + k(5,6);
K(3*j,3*i-2) = K(3*j,3*i-2) + k(6,1);
K(3*j,3*i-1) = K(3*j,3*i-1) + k(6,2);
K(3*j,3*i) = K(3*j,3*i) + k(6,3);
K(3*j,3*j-2) = K(3*j,3*j-2) + k(6,4);
K(3*j,3*j-1) = K(3*j,3*j-1) + k(6,5);
K(3*j,3*j) = K(3*j,3*j) + k(6,6);
y = K;
```

(Fig.8 - PlaneFrameAssemble on Octave)

Coding for Calculations on Octave

First of all, we made all the necessery functions on Octave such as SpringElementStiffness, SpringAssemble, PlaneFrameElementLength, PlaneFrameElementStiffness and PlaneFrameAssemble. We are going to use them in code and we are going to place convenient values into them.

Step 1 – Define Spring Elements and Plane Frame Elements



Spring elements	Plane frame elements
$ks1 = 10^5$ kN/m $ks2 = 2 \cdot 10^5$ kN/m	$E = 210 \cdot 10^6$ kPa $A = 0.75$ m^2 $I = 0.141$ m^4

(Fig.9 – Showing where the plane frame elements work and spring elements and also values of the spring elements and plane frame elements)

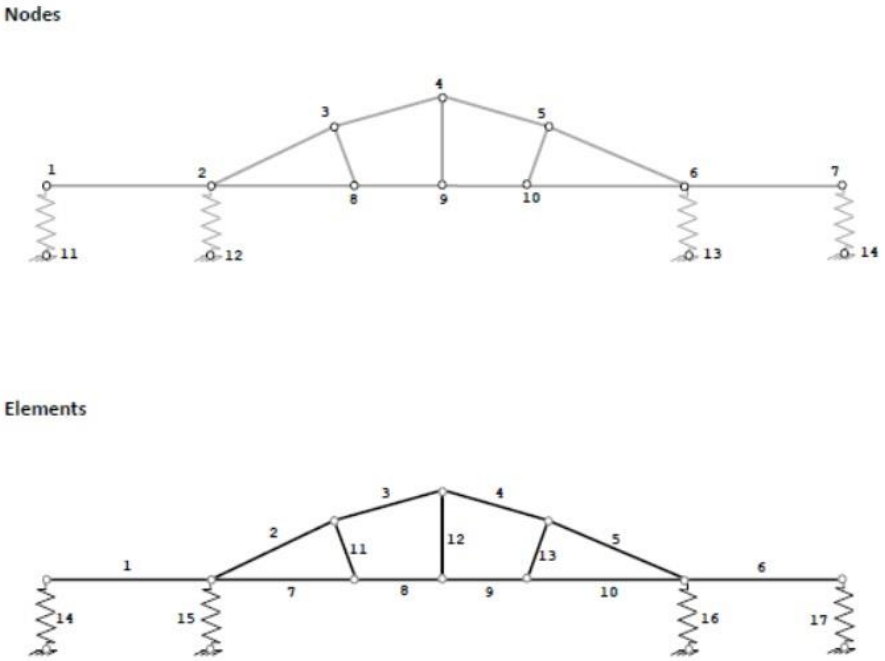
As you can see from the Fig.9 we can see the values for our calculations and informations about springs and plane frame elements. You can see them on Fig.10 on the below:

```
clc
clear

E = 210e6 % kPa
A = 0.75 % m²
I = 0.141 % m4
q = 20 % kN/m
rad2deg = 180 / pi; % degree

ks1 = SpringElementStiffness(1e5) % kN/m
ks2 = SpringElementStiffness(2e5) % kN/m
```

(Fig.10 –Element values on Octave)

Step 2 – Writing the Stiffness Matrices

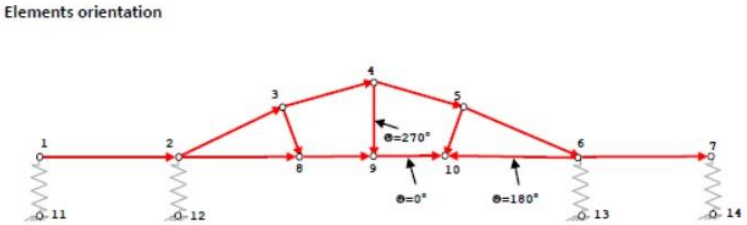
In first step, we have to discretize the domain.

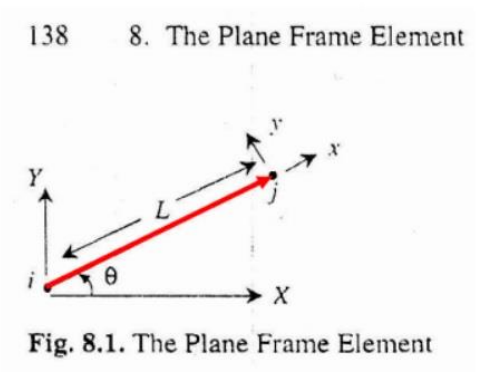


(Fig.11 – Showing nodes and elements with numbers)

In Fig.11 we can clearly see the number of the nodes and elements. We are going to use them as reference. We need to determine the lenghts with using PlaneFrameElementLength and degrees.

Degrees are defined from the node with the lower number to the node with the higher number, and the angle theta is measured counterclockwise. (Fig.12 – Elements orientation and theta angel definition)





```
%Lengths
L1 = 70 % m
L2 = PlaneFrameElementLength(0, 0, 60, 26) % m
L3 = PlaneFrameElementLength(0, 0, 40, 4) % m
L4 = PlaneFrameElementLength(0, 0, 40, -4) % m
L5 = PlaneFrameElementLength(0, 0, 60, -26) % m
L6 = 70 % m
L7 = 65 % m
L8 = 35 % m
L9 = 35 % m
L10 = 65 % m
L11 = PlaneFrameElementLength(0, 0, -5, 26) % m
L12 = 30 % m
L13 = PlaneFrameElementLength(0, 0, 5, 26) % m

%Degrees
theta1 = atan(26 / 60) * rad2deg;
theta2 = atan(4 / 40) * rad2deg;
theta3 = 360 - theta2;
theta4 = 360 - theta1;
theta5 = 270 + atan(5 / 26) * rad2deg;
theta6 = 270 - atan(5 / 26) * rad2deg;
```

(Fig.13 – Lenghts and theta angels as degree on Octave)

In the Fig.13, we can clearly see what we done about our degrees and lengths. rad2deg is equal to 180/pi. We can see it on Fig.10. We need to change the degrees radian to degrees and this function is helping us about transformation process.

After coding without any mistakes, we can create our global stiffness matrix fort he structure. We are going to use the functions from Fig.10 and from Fig.13.

(Fig.14 – Building structure of global stiffness matrix on Octave)

```
% Building global stiffness matrix for the structure.
k1 = PlaneFrameElementStiffness(E, A, I, L1, 0);
k2 = PlaneFrameElementStiffness(E, A, I, L2, theta1);
k3 = PlaneFrameElementStiffness(E, A, I, L3, theta2);
k4 = PlaneFrameElementStiffness(E, A, I, L4, theta3);
k5 = PlaneFrameElementStiffness(E, A, I, L5, theta4);
k6 = PlaneFrameElementStiffness(E, A, I, L6, 0);
k7 = PlaneFrameElementStiffness(E, A, I, L7, 0);
k8 = PlaneFrameElementStiffness(E, A, I, L8, 0);
k9 = PlaneFrameElementStiffness(E, A, I, L9, 0);
k10 = PlaneFrameElementStiffness(E, A, I, L10, 180);
k11 = PlaneFrameElementStiffness(E, A, I, L11, theta5);
k12 = PlaneFrameElementStiffness(E, A, I, L12, 270);
k13 = PlaneFrameElementStiffness(E, A, I, L13, theta6);
```

Step 3 – Assembling the Global Stiffness Matrix

First of all we obtain K we first set up a zero matrix of size 34x34, then make 13 calls for the plane elements and 3 calls for the spring elements with the MATLAB function PlaneFrameAssemble and SpringAssemble since we have 13 plane frame elements in the structure. Each call to the function will assemble one element which is 'K'.

The number of degrees of freedom (DoF) is calculated for plane frame elements only. The springs have also DoF: nodes 11, 12, 13 and 14 - every node has one DoF, so total number of DoFs = 10 * 3 + 4 * 1 = **34**

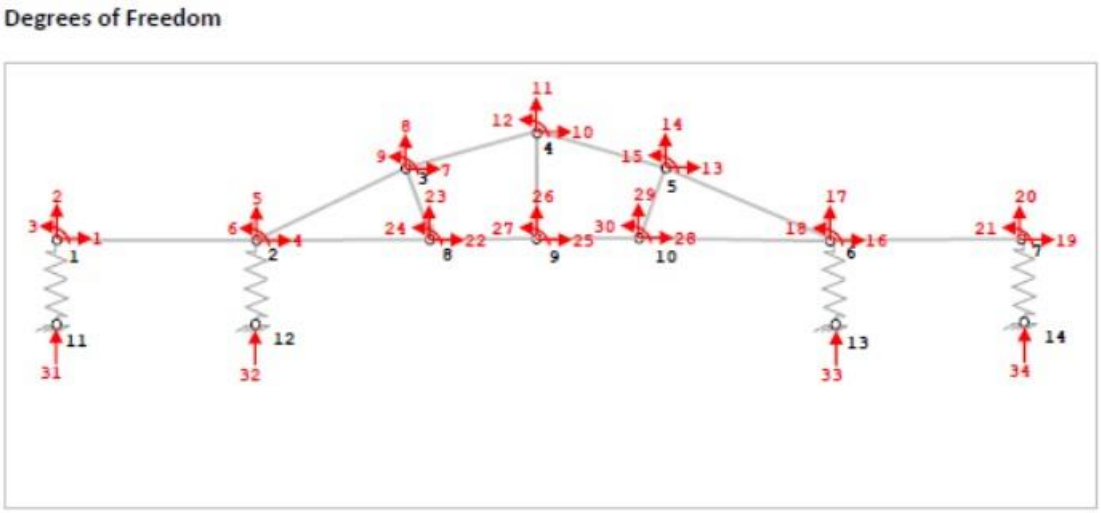
(Fig.15 - Building zero matrix and calling elements with functions we made on Octave)

```
57  K = zeros(34, 34);
58
59  K = PlaneFrameAssemble(K, k1, 1, 2);
60  K = PlaneFrameAssemble(K, k2, 2, 3);
61  K = PlaneFrameAssemble(K, k3, 3, 4);
62  K = PlaneFrameAssemble(K, k4, 4, 5);
63  K = PlaneFrameAssemble(K, k5, 5, 6);
64  K = PlaneFrameAssemble(K, k6, 6, 7);
65  K = PlaneFrameAssemble(K, k7, 2, 8);
66  K = PlaneFrameAssemble(K, k8, 8, 9);
67  K = PlaneFrameAssemble(K, k9, 9, 10);
68  K = PlaneFrameAssemble(K, k10, 6, 10);
69  K = PlaneFrameAssemble(K, k11, 3, 8);
70  K = PlaneFrameAssemble(K, k12, 4, 9);
71  K = PlaneFrameAssemble(K, k13, 5, 10);
72
73  K = SpringAssemble(K, ks1, 2, 31);
74  K = SpringAssemble(K, ks2, 5, 32);
75  K = SpringAssemble(K, ks2, 17, 33);
76  K = SpringAssemble(K, ks1, 20, 34);
```

Step 4 – Applying the Boundary Conditions

In step 3 we are going to apply the boundary conditions.

(Fig.16 - Degrees of freedom)



The last arguments of the SpringAssemble() function are node numbers only for systems that contain only spring elements (no other element types). Since spring nodes have only one degree of freedom, you need to provide freedom degree numbers instead of node numbers to properly assemble the springs.

Now we need to extract 30 columns and 30 rows from the global stiffness matrix.

Hint: displacements in spring supports are known and equal to zero - we don't need columns and rows that are the equations of these supports.

(Fig.17 – Calculating rotations at nodes(as rad) on Octave)

```
k = K(1:30,1:30);
f = [0; -700; -8167; 0; -1350; 1125; 0; 0; 0; 0; 0; 0; 0; 0; 0; -1350; -1125; 0; -700; 8167; 0; -1000; 5000; 0; -700; 0; 0; -1000; -5000];
u = k\f;
```

```
u =

  -5.2105e-03
  -5.4265e-03
  -5.4405e-03
  -5.2105e-03
  -1.4287e-02
   8.4755e-04
  -2.7513e-03
  -2.3252e-02
  -2.7885e-04
  -3.3538e-03
  -2.4986e-02
  -4.2031e-17
  -3.9563e-03
  -2.3252e-02
   2.7885e-04
  -1.4971e-03
  -1.4287e-02
  -8.4755e-04
  -1.4971e-03
  -5.4265e-03
   5.4405e-03
  -4.0232e-03
  -2.3670e-02
   4.0436e-04
  -3.3538e-03
  -2.5093e-02
  -6.6507e-17
  -2.6845e-03
  -2.3670e-02
  -4.0436e-04
```

(Fig.18 - Results of rotations at nodes(as rad) on command window)

Note: The backslash operator "\" is used for Gaussian elimination. We can clearly see our displaments at nodes. Rotations at nodes(as rad).

Positive numbers: *Counterclock-wise*

Negative numbers: C*lockwise.*

Now we can setup our global nodal displacement vector U. The calculated displacements for nodes 1-10 should be supplemented with four displacements in the spring supports.

(i.e. nodes: 11 12 13 14, one displacement per node, which actually means one zero per node.)

Step 5 – Post-processing

```
96  U = [u;0;0;0;0];
97  % Calculading the global nodal force vector F.
98  F = K * U % Reactions in 1st, 2nd, 6th, 7th, 8th, 9th and 10th nodes.
```

(Fig.19 – Calculating the global nodal force vector on Octave)

```
F =

    -0.00000
  -700.00000
 -8167.00000
     0.00000
 -1350.00000
  1125.00000
    -0.00000
     0.00000
    -0.00000
    -0.00000
    -0.00000
    -0.00000
    -0.00000
     0.00000
    -0.00000
    -0.00000
 -1350.00000
 -1125.00000
     0.00000
  -700.00000
  8167.00000
    -0.00000
 -1000.00000
  5000.00000
     0.00000
  -700.00000
    -0.00000
     0.00000
 -1000.00000
 -5000.00000
   542.65236
  2857.34764
  2857.34764
   542.65236
```

In this step, finally we obtain the reactions at nodes 1st, 2nd, 6th, 7th, 8th, 9th and 10th forces (axial forces, shears and moments) in each plane frame element using MATLAB as follows. First we set up the global nodal displacement vector ‘U’ , then we calculate the global nodal force vector ‘F’(as kN). You can see it on Fig.19.

(Fig.20 – Results of the global nodal force vector on command window)

Reactions in each nodes.

Positive numbers: upward

Negative numbers: downward

We have just calculated internal forces(M, N, T) and reactions of supports in the bridge structure using spring and plane frame elements.

END