Relatorio EP3 - PMR3401

Rafael Sobral Augusto 10337193
Laurent Rouvinez 11488026

São Paulo
Escola Politecnica da USP
2020

Table des matières

1 Introduction

This exercise comprises of two parts. The first part analyses the dynamics of a mechanical structure simulating a wind turbine under a load. We will heavily rely on *Ansys* to do various modal, transient and harmonic analyses. Furthermore, we will do the same analysis using a specifically developed program in *Matlab* to compare and support our results.

The second part of this exercise is a static analysis of a given piece under high load. Here again, we will use *Ansys* to extract the deformations and internal stresses.

2 Part 1

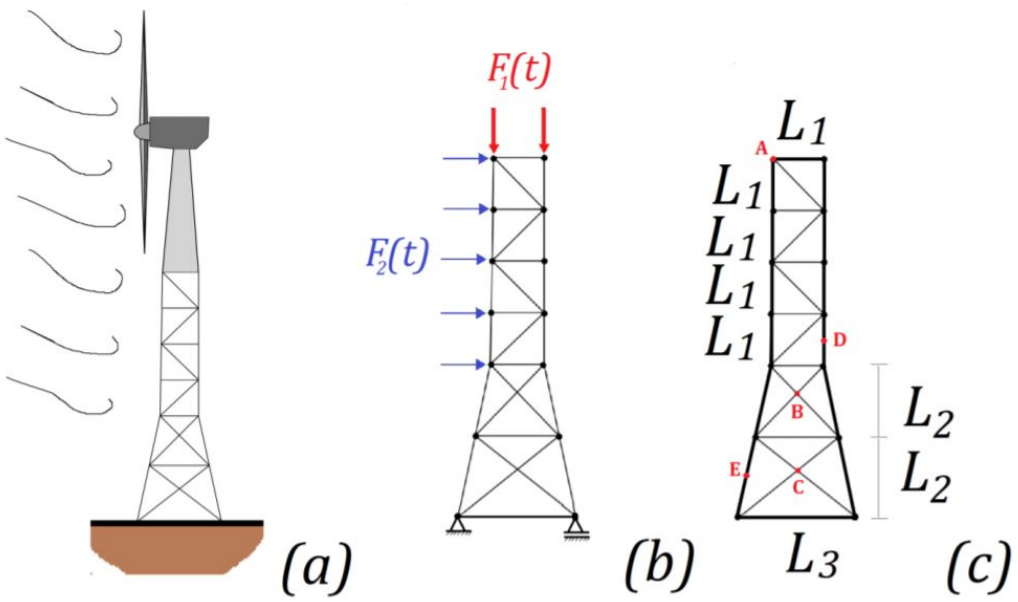


FIGURE 1 – Structure Diagram

Illustrated here is a diagram representing the simplified mechanical structure of the wind turbine an the load it is subjected to. The forces and physical parameters are display below.

$$\overrightarrow{F_1(t)} = 2.F.\sin(2\pi t)\,\vec{j}$$

$$\overrightarrow{F_2(t)} = \begin{cases} 5.D\vec{i} & se\ t_1 \le t \le t_2 \\ 0 & se\ t < t_1\ ou\ t > t_2 \end{cases}$$

FIGURE 2 – Forces

F	D	t_1	t_2	L_1	L_2	v
8000	2000	2,0	8,0	2,0	3,0	0,29
d_{1i}	d_{1e}	d_{2i}	d_{2e}	L_3	E	ρ
0,072	0,080	0,090	0,100	4,0	210.10^9	7650

FIGURE 3 – Parameters

2.1 a) *Ansys*

Using *Ansys* we can model the structure of the turbine. The program easily allows us to create beam of different cross-sections and to impose the necessary degrees of freedom to the base. In this first analysis the individual beams have beam have been meshed into 10 individual elements.

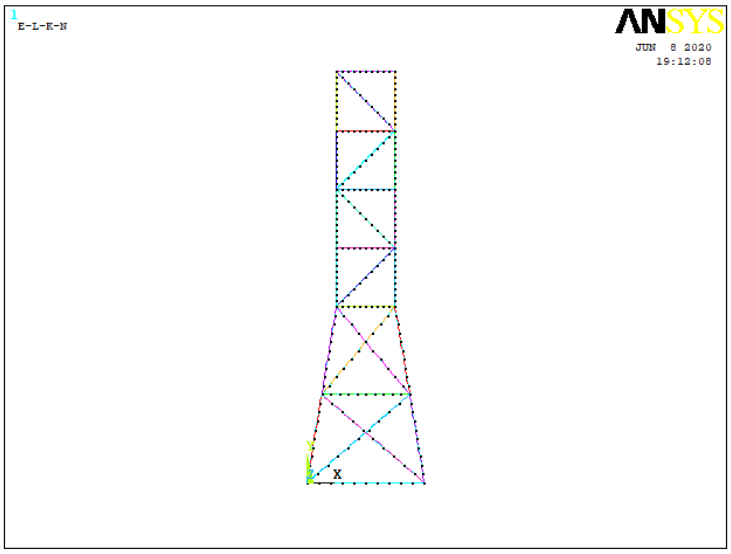


FIGURE 4 – Nodes of Structure

2.1.1 a.1) Modal Analysis

We have to plot the first 6 vibration modes and extract their corresponding resonance frequencies. Using modal analysis option of *Ansys* makes this task rather straightforward :

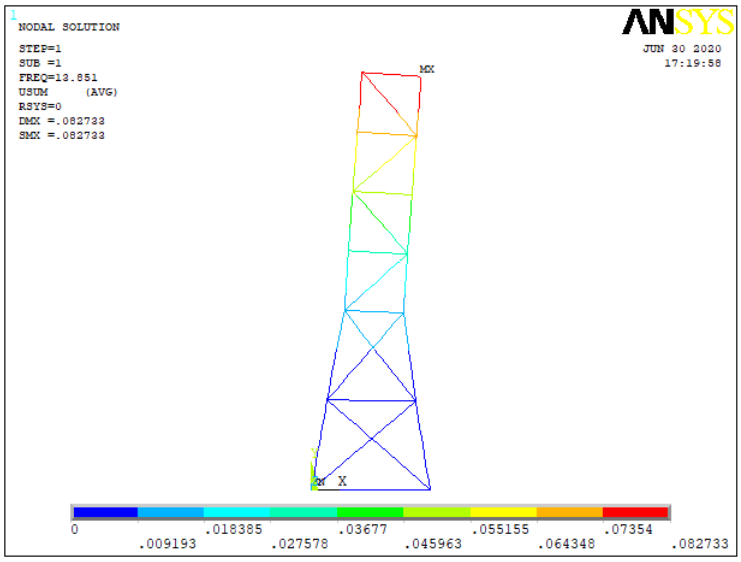


FIGURE 5 – Mode 1

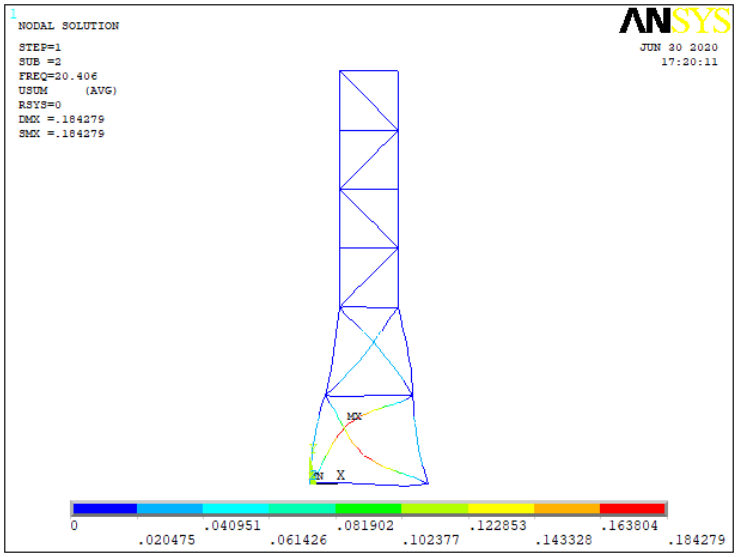


FIGURE 6 – Mode 2

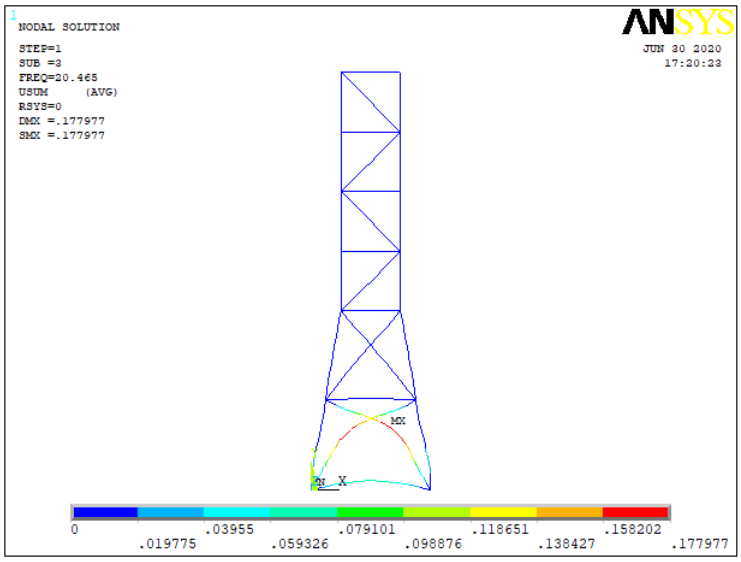


FIGURE 7 – Mode 3

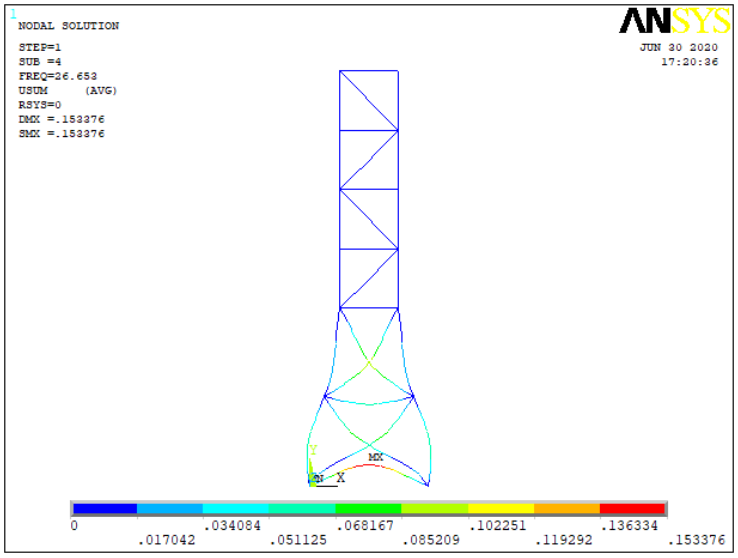


FIGURE 8 – Mode 4

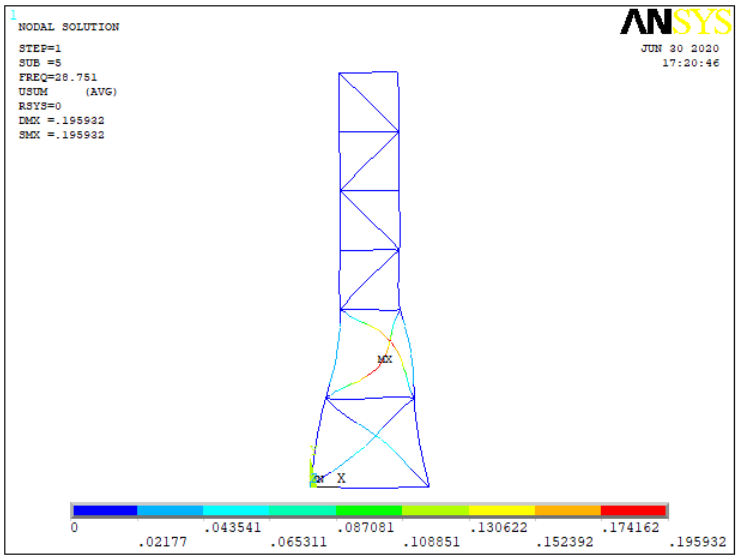


FIGURE 9 – Mode 5

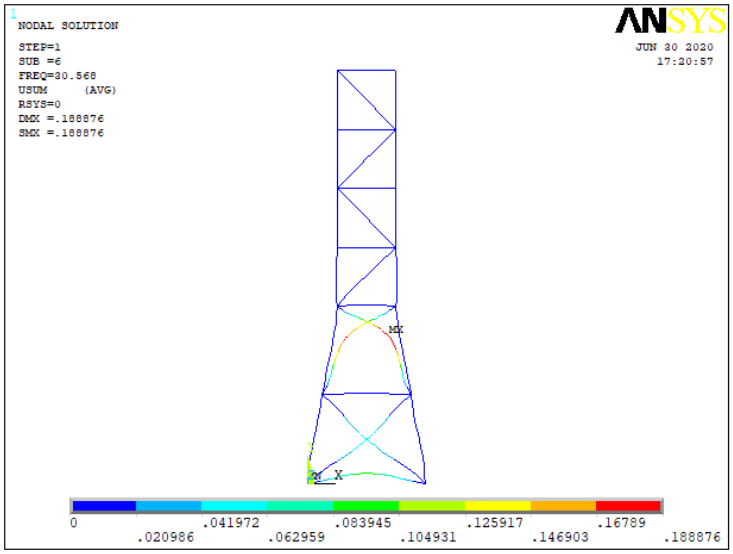


FIGURE 10 – Mode 6

Extracting the resonance frequencies, we can complete the following table :

Mode	Frequency [Hz]
1	13.851
2	20.406
3	20.465
4	26.653
5	28.751
6	30.568

TABLE 1 – Resonance Frequencies

2.1.2 a.2) Transient Analysis

For this second analysis, we have to take into account both forces *F1(t)* and *F2(t)*. Since *F2(t)* is only modeled as a temporarily acting force, we will concentrate our analysis around its active time. Hence, we will simulate the system between *0s* and *10s*. Additionally, we have to choose the amount of substeps, which will determine discretisation of time. Note that in **a.4)** we will vary this parameter to analyse its influence on the simulation. For now, the analysis will have 100 substeps, meaning there is one step every *0.1s*.

Firstly, we have to plot the mechanical tension at the points A and B over time :

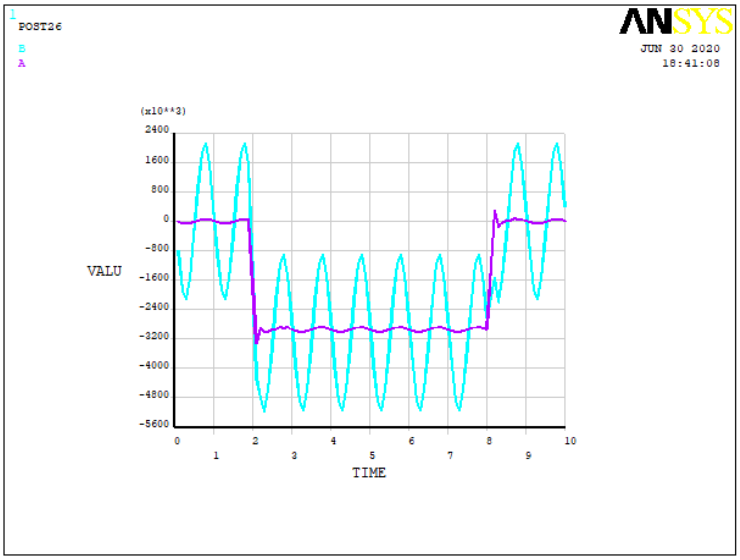


FIGURE 11 – Mechanical Tensions at Points A and B

Now, we can plot the displacements of points D and E over time. The following graphs show the points' displacements over x, y, and their modulus.

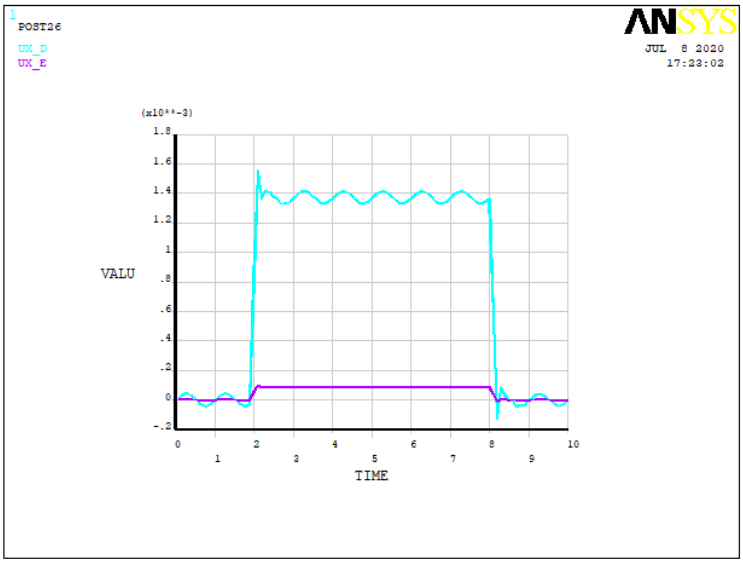


FIGURE 12 – Displacements Along X of Points D and E

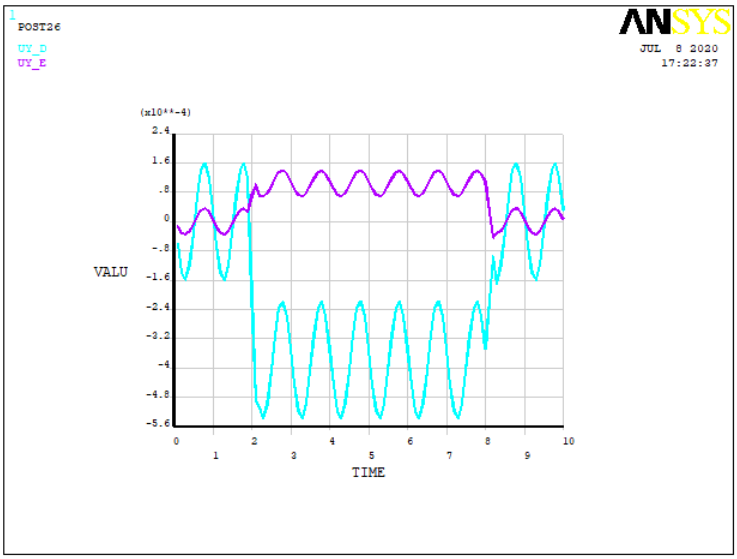


FIGURE 13 – Displacements Along Y of Points D and E

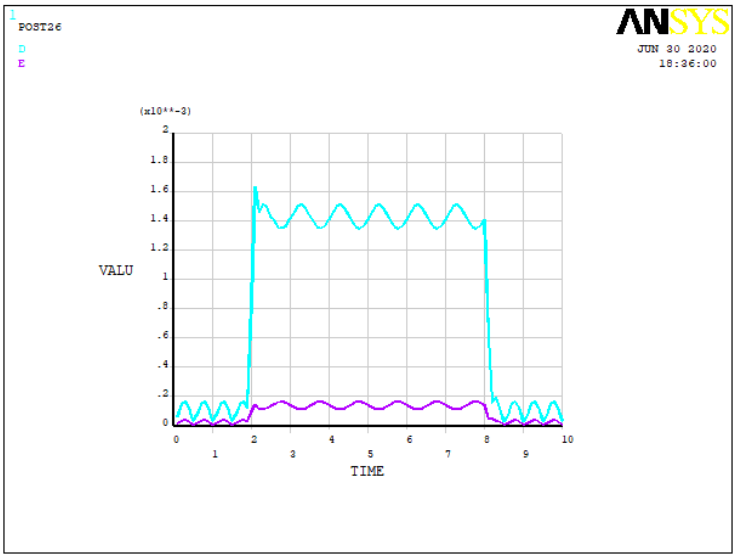


FIGURE 14 – Displacement Modulus of Points D and E

2.1.3 a.3) Harmonic Analysis

This section of the exercise focuses on the influence of the frequency of the load on the structure. By varying the applied frequency and plotting the absolute displacement of select points, we should be able to pick out the structure's resonance frequencies. Here again, the spectrum of frequencies and number of substeps will define the discretisation of the frequency dimension. We want to include all 6 resonance frequencies found before (Table 1) and choose a reasonable number of substeps. The following simulation analyses the frequencies 10Hz-35Hz with 200 substeps. We can then plot the amplitudes of displacement of points B and C :

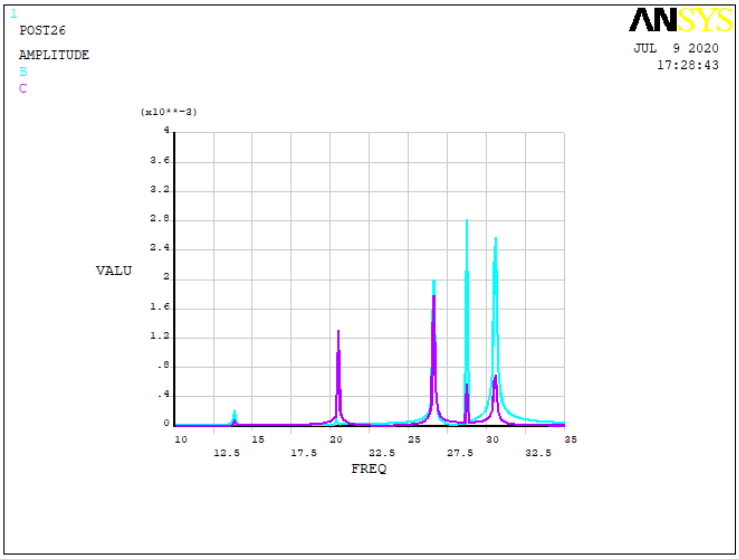


FIGURE 15 – Displacement Modulus of Points D and E

We can clearly identify every frequency by a peak in amplitude, except the second and third mode, since their values are too close to be distinguishable. The amplitude generally seems to be higher for point B, which is probably due to the fact that the point is higher up in the structure, where the nodes are further away from the fixed base.

2.1.4 a.4) Influence of discretisation

We now have to discuss the influence of different discretisations on our simulation. Firstly, we will evaluate the effect of the division of beams on the resoncance frequencies. The frequencies obtained with 10 elements per beam were the following :

Mode	Frequency [Hz]
1	13.851
2	20.406
3	20.465
4	26.653
5	28.751
6	30.568

TABLE 2 – Resonance Frequencies with 10 elements/beam

Now varying the amount of elements per beam :

Mode	Frequency [Hz]
1	13.851
2	20.406
3	20.464
4	26.652
5	28.750
6	30.567

TABLE 3 – Resonance Frequencies with 18 elements/beam

Mode	Frequency [Hz]
1	13.852
2	20.413
3	20.472
4	26.660
5	28.762
6	30.581

TABLE 4 – Resonance Frequencies with 5 elements/beam

Mode	Frequency [Hz]
1	13.901
2	29.534
3	32.225
4	39.805
5	48.193
6	57.838

TABLE 5 – Resonance Frequencies with 1 element/beam

Unfortunately, the given student version of *Ansys* does not allow for any more than 18 divisions per element. We can however still deduct that the frequencies do not vary significantly. Only by using but one element to represent each beam, we obtain an almost completely different set of frequencies. The reason for this is that, except for the first mode, all the following modes of vibration were oscillation of the fin internal beams. By reducing each beam to one element, we exempt this from happening. Hence, all the modes are oscillations of the structure as a whole, as opposed to individual beams.

Finally, we wish to analyse the discretisation of time and its influence on the displacement of the point A. The transient analysis before had 100 substeps, we can now vary this amount and see if we obtain different results :

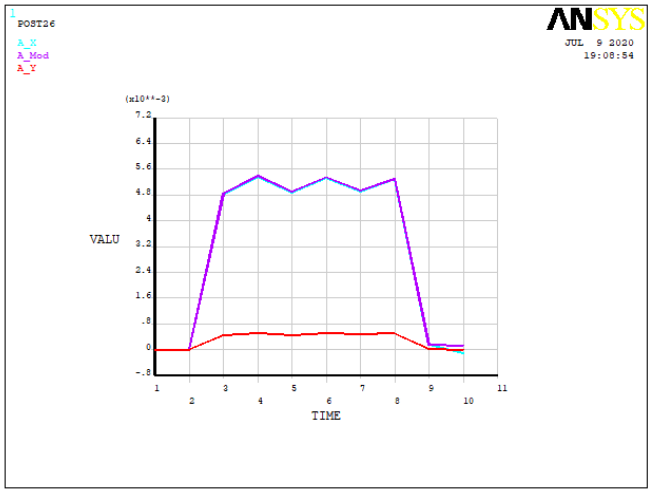


FIGURE 16 – Displacement of Point A using 10 substeps.

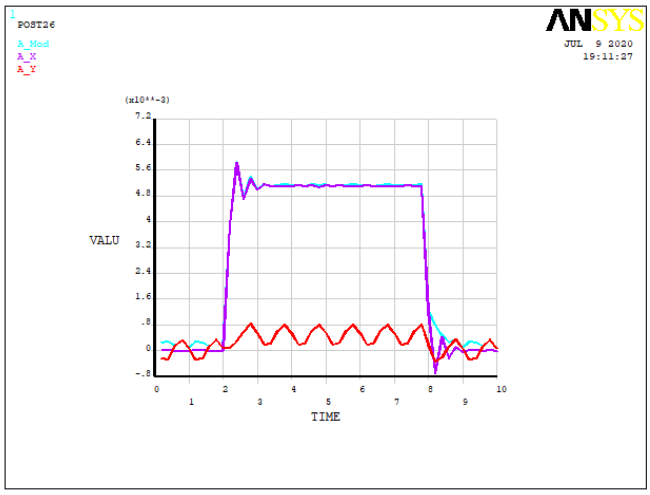


FIGURE 17 – Displacement of Point A using 50 substeps.

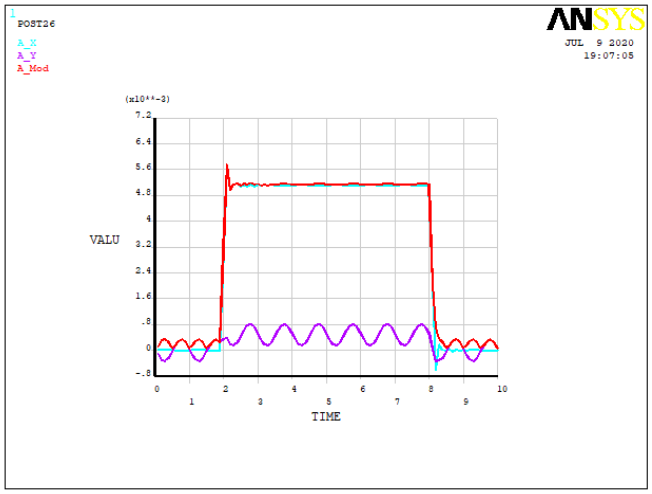


FIGURE 18 – Displacement of Point A using 100 substeps.

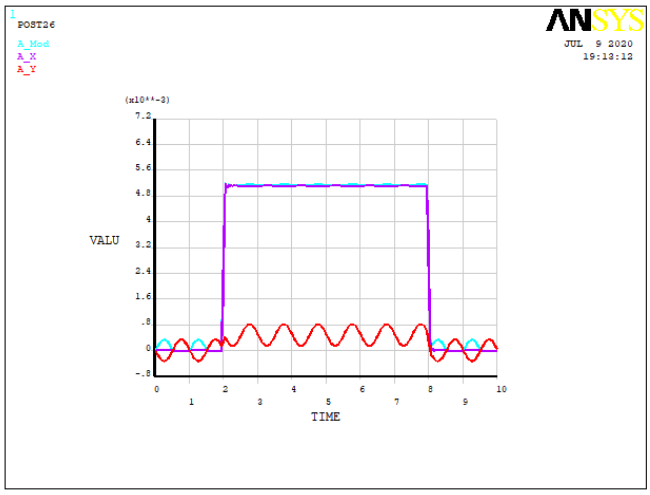


FIGURE 19 – Displacement of Point A using 250 substeps.

As we can see, the gross amplitude of displacement during the steady states of the analysis are similar no matter the number of substeps. Obviously, using less substeps will result in a graph which seems more like a series of points, rather than a line. Using many substeps we can however observe an interesting behaviour : comparing figure 18 and 19, we see that the simulation with 250 substeps has less of an overshoot after the application of the second force (around $2s$). There might be a few reasons for this. One reason could be a loss of accuracy due to a rounding error considering the minuscule size of the timestep. Another reason could be due to the way the dynamics of the system are computed by *Ansys*. Either way, by comparing different options, we can gain a better understanding of the behaviour of the system.

2.2 b) *Matlab*

2.2.1 Pre-processing

After having finished all the analysis using *Ansys*, we were able to develop a program in *Matlab* in order to simulate the same situation and compare the results between the 2 distinct *softwares*.

At first, we defined the original nodes, which are

Nodes	X (m)	Y (m)
1	0	0
2	4	0
3	0.5	3
4	3.5	3
5	1	6
6	3	6
7	1	8
8	3	8
9	1	10
10	3	10
11	1	12
12	3	12
13	1	14
14	3	14

TABLE 6 – Table with the original nodes and its corresponding X and Y coordinates.

After having settled the original nodes coordinates and the parameters of the problem as enunciated, we were able to use the function *criabarras.m* - which is part of the *Appendix* section - to create the bars that connect the nodes with each other in the same way as the proposed geometry.

In order to confirm that we had defined the bars correctly, we used the function *plotatorre.m* - also part of the *Appendix* section - to plot the geometry that we had so far, and this was the result :

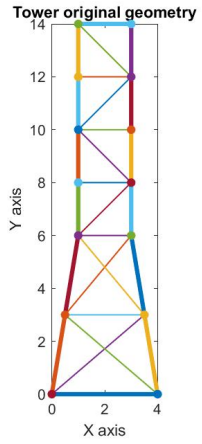
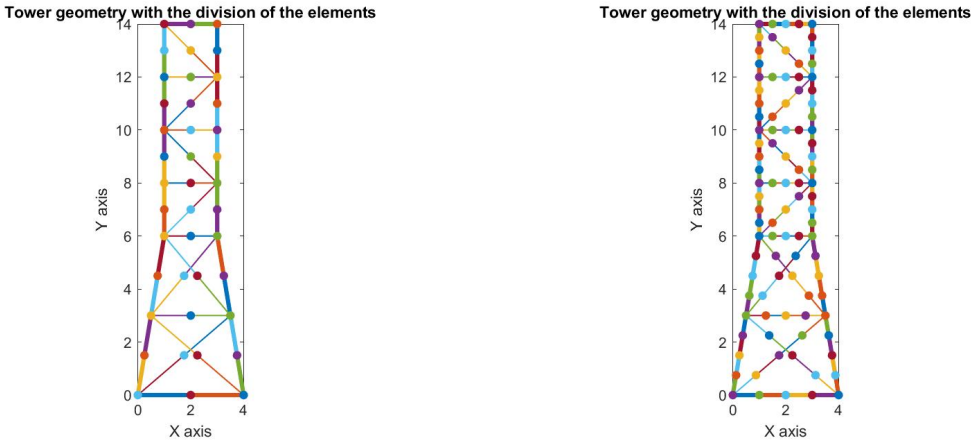


FIGURE 20 – Reconstruction of the proposed geometry with *Matlab*.

Secondly, we divided each bar in elements. The program allows you to change the number of elements desired so the analysis can be refined. The routine responsible for doing so creates two new lists, *nos_elemento* and *elemento*, which contants, respectively, the coordinates of all the nodes - the original ones and the newly created - and the nodes which the elements are linked to.

The entire routine to do this discretization is contained in the function *criaelementos.m*, part of the *Appendix*. Just to exemplify, we were able to plot all the nodes and elements after we had done this discretization. The plots of the tower using 2 and 4 elements to divide each bar were the ones shown below :



(a) (b)

FIGURE 21 – a) Tower with each bar divided into 2 elements. b) Tower with each bar divided into 4 elements.

Once we had the coordinates of each node and element, we were able to use the local stiffness and mass matrices to build the global one. Since we are using portico elements, the local stiffness, mass and the lumped mass matrices - usually used to lower the complexity of the program - were created with anonymous functions inside the function *criamatrizes.m*, and they are as shown below :

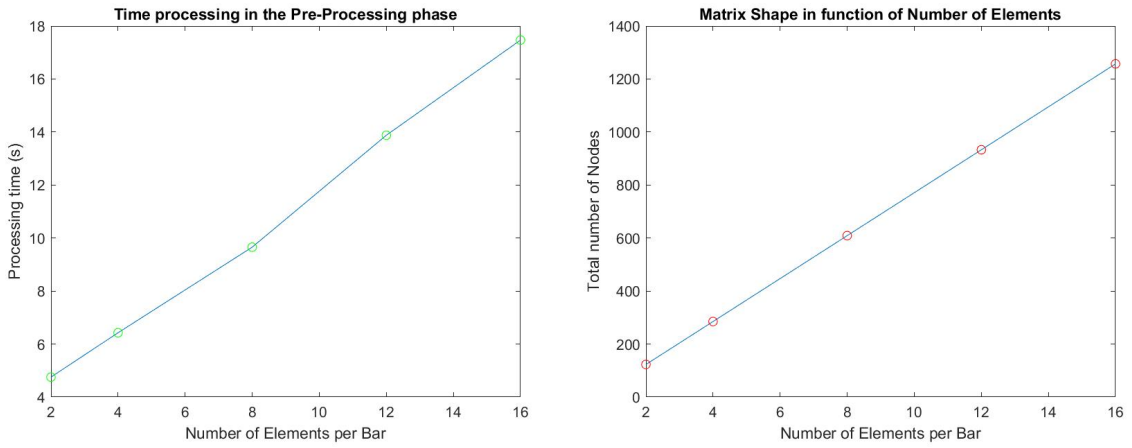
$$[K] = \frac{EA}{L}\begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \frac{EI}{L^3}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 12 & 6L & 0 & -12 & 6L \\ 0 & 6L & 4L^2 & 0 & -6L & 2L^2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -12 & -6L & 0 & 12 & -6L \\ 0 & 6L & 2L^2 & 0 & -6L & 4L^2 \end{bmatrix}$$

$$[M] = \frac{\rho AL}{420}\begin{bmatrix} 140 & 0 & 0 & 70 & 0 & 0 \\ 0 & 156 & 22L & 0 & 54 & -13L \\ 0 & 22L & 4L^2 & 0 & 13L & -3L^2 \\ 70 & 0 & 0 & 140 & 0 & 0 \\ 0 & 54 & 13L & 0 & 156 & -22L \\ 0 & -13L & -3L^2 & 0 & -22L & 4L^2 \end{bmatrix}$$

$$[M_{lumped}] = \frac{\rho AL}{78}\begin{bmatrix} 39 & 0 & 0 & 0 & 0 & 0 \\ 0 & 39 & 0 & 0 & 0 & 0 \\ 0 & 0 & L^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 39 & 0 & 0 \\ 0 & 0 & 0 & 0 & 39 & 0 \\ 0 & 0 & 0 & 0 & 0 & L^2 \end{bmatrix}$$

After this, the combination of each element matrices into the global ones was implemented in the function *matrizesglobais.m*. To do so, we used the coordinates of the two nodes which every element was linked to in order to position their contribution to the global matrix where it should be. Hence, we were ready to begin with the dynamic analysis.

As an addition, we compared the time it took to complete the pre-processing of everything using different numbers of elements/bar and the shape of the matrices generated.



(a) (b)

FIGURE 22 – a) Time processing in the Pre-Processing phase. b) Matrices Shape in function of the number of elements per bar.

As we can see above, the time processing - which was measured five times and then we calculated the mean time processing of each case - to complete the entire pre-processing is approximately linear, which we found curious. But, since we calculate all the matrices in one single loop, that depends only on the number of elements - which vary linearly, as the shape of the matrices - it made sense and was very reasonable.

2.2.2 a.1) Modal Analysis

In order to do the **Modal Analysis**, we first established the contour conditions and reshaped the matrix without the rows and columns that represented the movement in the directions that are not allowed by the constraints. Thus, we removed the rows and columns that represented the linear movement - in X and Y axis - regarding to the original node 1 and the part of the matrix that represented the movement along the Y axis for the original node number 2.

After this, we obtained the *eigenvalues* and the respective *eigenvectors* of the matrix

$$[K - \omega^2 M]$$

Which represent, respectively, the resonance frequencies and the vibration modes of the structure. Both were obtained using the command *eigs* from *Matlab*.

We varied the number of elements used and were able to compare the resonance frequencies obtained by the method with this difference in the discretization and between both the complete and the lumped mass matrices. Thus, the 6 first resonance frequencies obtained in different cases were :

Number of Elements/Bar	Resonance frequencies (ω)					
	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6
1	13.9008	29.5458	32.2380	39.8241	48.2002	57.8671
2	13.8549	20.6623	20.7137	26.9410	29.1335	31.0112
4	13.8519	20.4289	20.4870	26.6821	28.7867	30.6109
8	13.8517	20.4111	20.4698	26.6639	28.7598	30.5786
12	13.8516	20.4102	20.4689	26.6629	28.7584	30.5768

TABLE 7 – Resonance frequencies obtained using the complete mass matrix.

Number of Elements/Bar	Resonance frequencies (ω)					
	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6
1	13.7085	28.1508	31.7933	36.4917	40.3268	49.1837
2	13.8092	18.5432	18.6321	24.9427	26.0636	27.3298
4	13.8407	20.1906	20.2508	26.3889	28.4673	30.2297
8	13.8489	20.3627	20.4216	26.5987	28.6958	30.5022
12	13.8504	20.3894	20.4482	26.6345	28.7309	30.5441

TABLE 8 – Resonance frequencies obtained using the lumped mass matrix.

As we can see from the data above, both the complete and the lumped mass matrices were able to converge to the same resonance frequencies obtained in our simulations with *Ansys*. As we can see, the number of elements per bar also wasn't a big factor, since after 2 elements/bar the resonance frequencies obtained were very close. The only dissonant result was obtained with only 1 element/bar, and this also happened with *Ansys*, showing us that this wasn't a *Matlab* limitation, but that this discretization wasn't enough to find the correct resonance frequencies.

Regarding the modes of vibration, we were able to plot them using the *eigenvectors* obtained using the command *eigs* in *Matlab*. Once we had this values, we used the **Lagrange Polynomials** to interpolate the longitudinal deformation of the elements and the **Hermite Shape Functions** to interpolate the deflection regarding the elements' relatives Y-axis and the rotational deformation. The function *plotavibsmodes.m* in the *Appendix* contains the entire routine used to do so, and the vibration modes found - using 8 elements per bar - were :

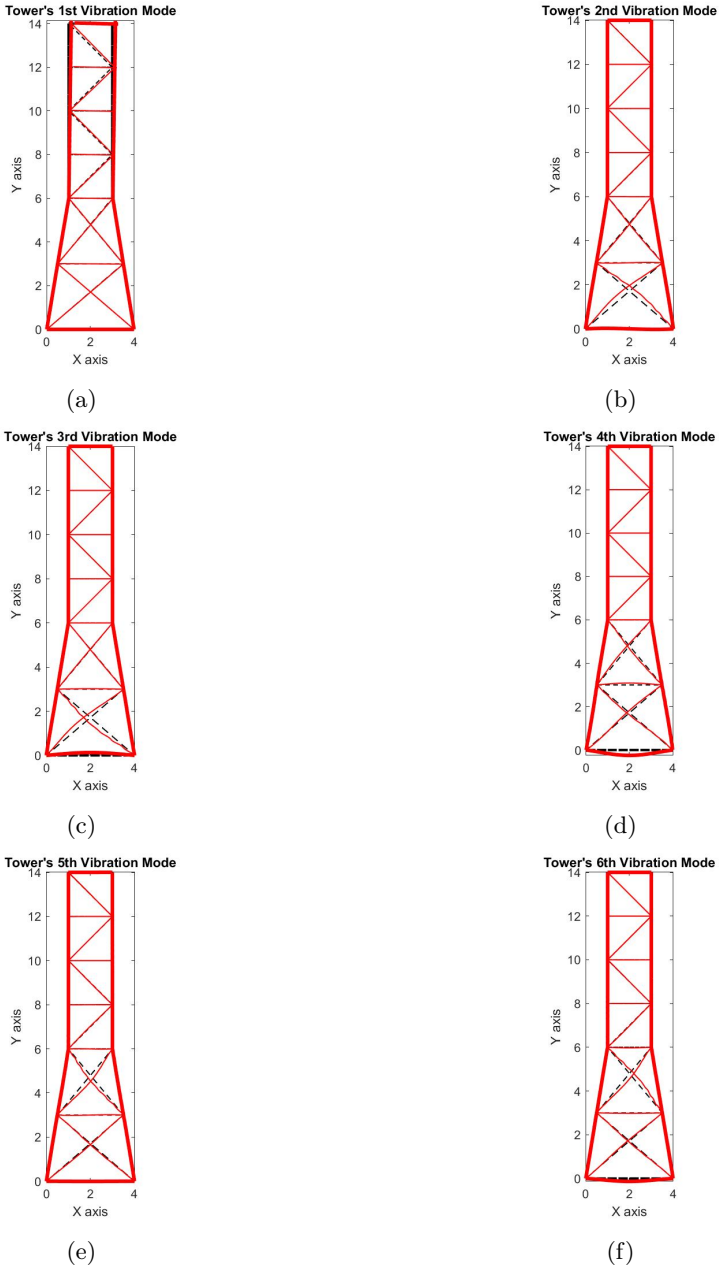
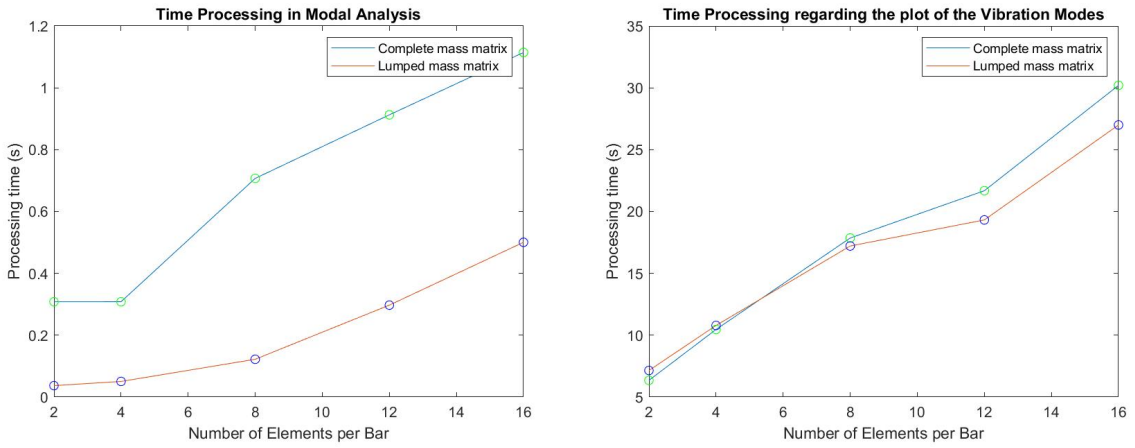


FIGURE 23 – a) 1^{st} vibration mode. b) 2^{nd} vibration mode. c) 3^{rd} vibration mode. d) 4^{th} vibration mode. e) 5^{th} vibration mode. f) 6^{th} vibration mode.

Again, after comparing the results to the ones obtained using *Ansys*, we could confirm that both results match - both regarding the resonance frequencies and their respective vibration modes.

As an addition, we analysed how the processing time was affected by the different number of elements used to process the several **Modal Analysis** to study how the resonance frequencies change with it. Therefore, we measured the time it took to compute the 6 first resonance frequencies - eigenvalues - and its respective vibration modes - eigenvectors - and the time to plot it.



(a) (b)

FIGURE 24 – a) Time processing in **Modal Analysis**. b) Time processing regarding the plot of the Vibration Modes.

In order to gather the data for those plots, we ran our code five times per number of elements/bar, calculated the mean processing time and then plotted the graphs. As we can see, the time to plot the results don't change much between the both matrices, what is expected, since the loops are entirely depending only on the number of elements and nodes, which are the same for both.

However, somehow, even though the matrices have the same dimensions, the time it takes to obtain the inverse matrix and the eigenvalues and Eigenvectors are different. This values are calculated by the commands *inv* and *eigs*, built-in commands in *Matlab*, that somehow can identify that the lumped mass matrix is only diagonal and use it to fasten up the program, cutting the processing time in half.

This is a small time for the matrices in the dimensions we are using, but if we had to process million-elements matrices, it would result in a significant time saved.

2.2.3 a.2) Transient Analysis

In order to perform the **Transient Analysis**, we first approximated the damping matrix using the *Rayleigh Method*

$$[C] = \alpha[M] + \beta[K]$$

And considering $\alpha = 0.3$ and $\beta = 0.03$. After this, we performed the **Transient Analysis** using the *Newmark Method*, and considering the *constant acceleration method*, which defines the values $\gamma = 0.5$ and $\beta_2 = 0.25$ for this method. This method consists in, once we have the dynamic matrices, we solve a linear system for each step on time to define the acceleration in the next instant :

$$[\bar{M}]\{a_{n+1}\} = \{\bar{f}_{n+1}\}$$

Where the matrices used are such as

$$[\bar{M}] = [M] + \Delta t\gamma[C] + \Delta t^2\beta_2[K]$$

$$[\bar{M}] = (1 + \Delta t\gamma\alpha)[M] + (\Delta t^2\beta_2 + \Delta t\beta\gamma)[K]$$

$$\{\bar{f}_{n+1}\} = \{f_{n+1}\} - [C](\{v_n\} + \Delta t(1-\gamma)\{a_n\}) - [K]\left(\{d_n\} + \Delta t\{v_n\} + \tfrac{\Delta t^2}{2}(1-2\beta_2)\{a_n\}\right)$$

Once we've calculated the matrices with the last step in time and solved the linear system to obtain the values of the acceleration in the next step $\{a_{n+1}\}$, we could use this value to calculate the velocity and the displacement of the nodes in the next step in time as well, with the formulas :

$$\{v_{n+1}\} = \{v_n\} + \Delta t((1-\gamma)\{a_n\} + \gamma\{a_{n+1}\})$$

$$\{d_{n+1}\} = \{d_n\} + \Delta t\{v_n\} + \frac{\Delta t^2}{2}((1-2\beta_2)\{a_n\}) + 2\beta_2\{a_{n+1}\}$$

Once we had all this settled, we were able to establish the loop to calculate the values of these variables over time.

It was asked that we plotted the displacement in the nodes D and E, and we calculated it between 0 and $10s$ of simulation, with different time steps to analyse their influence on the solution. Therefore, using 8 elements per bar and step intervals $\Delta t = 0.5, 0.1, 0.05, 0.01$ we were able to obtain the following graphics

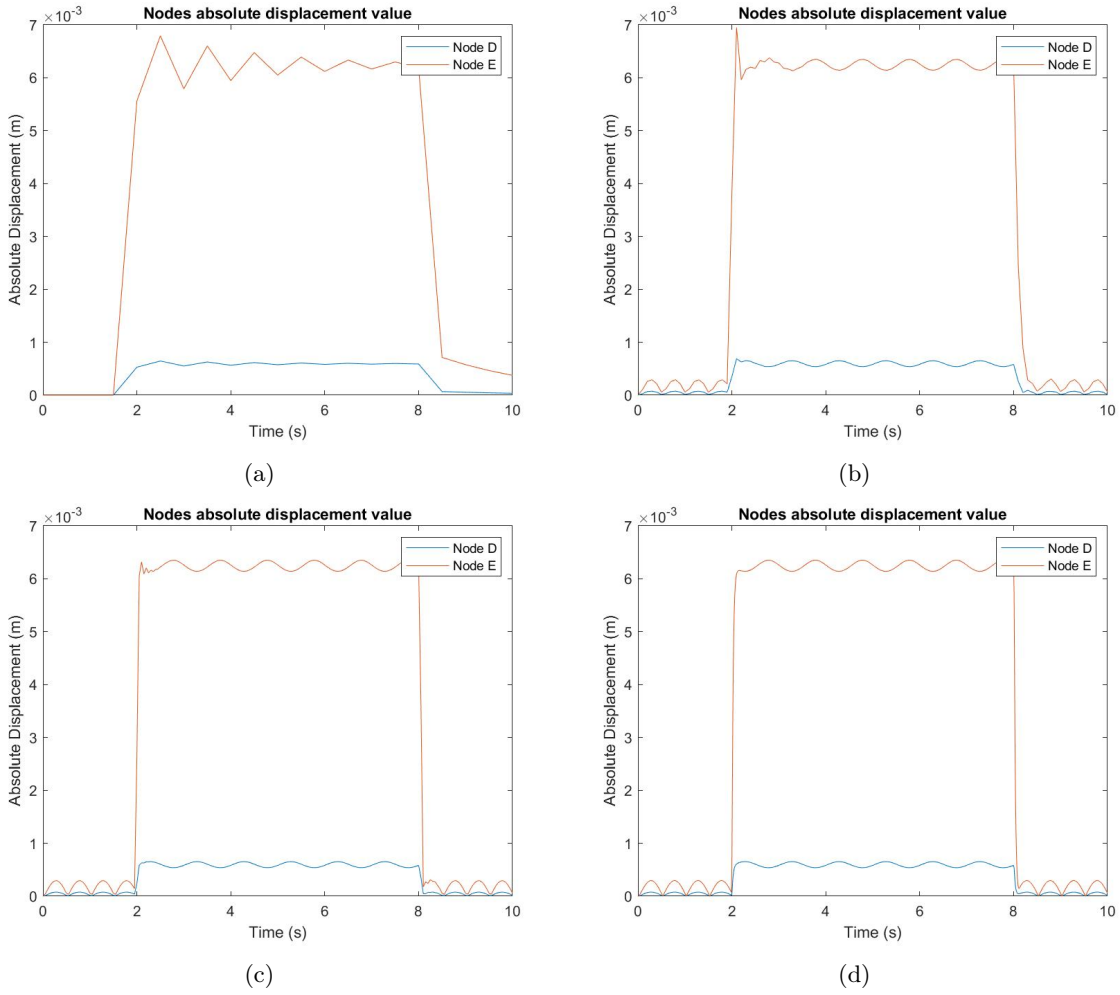


FIGURE 25 – a) Absolute displacements using $\Delta t = 0.5s$. b) Absolute displacements using $\Delta t = 0.1s$. c) Absolute displacements using $\Delta t = 0.05s$. d) Absolute displacements using $\Delta t = 0.01s$.

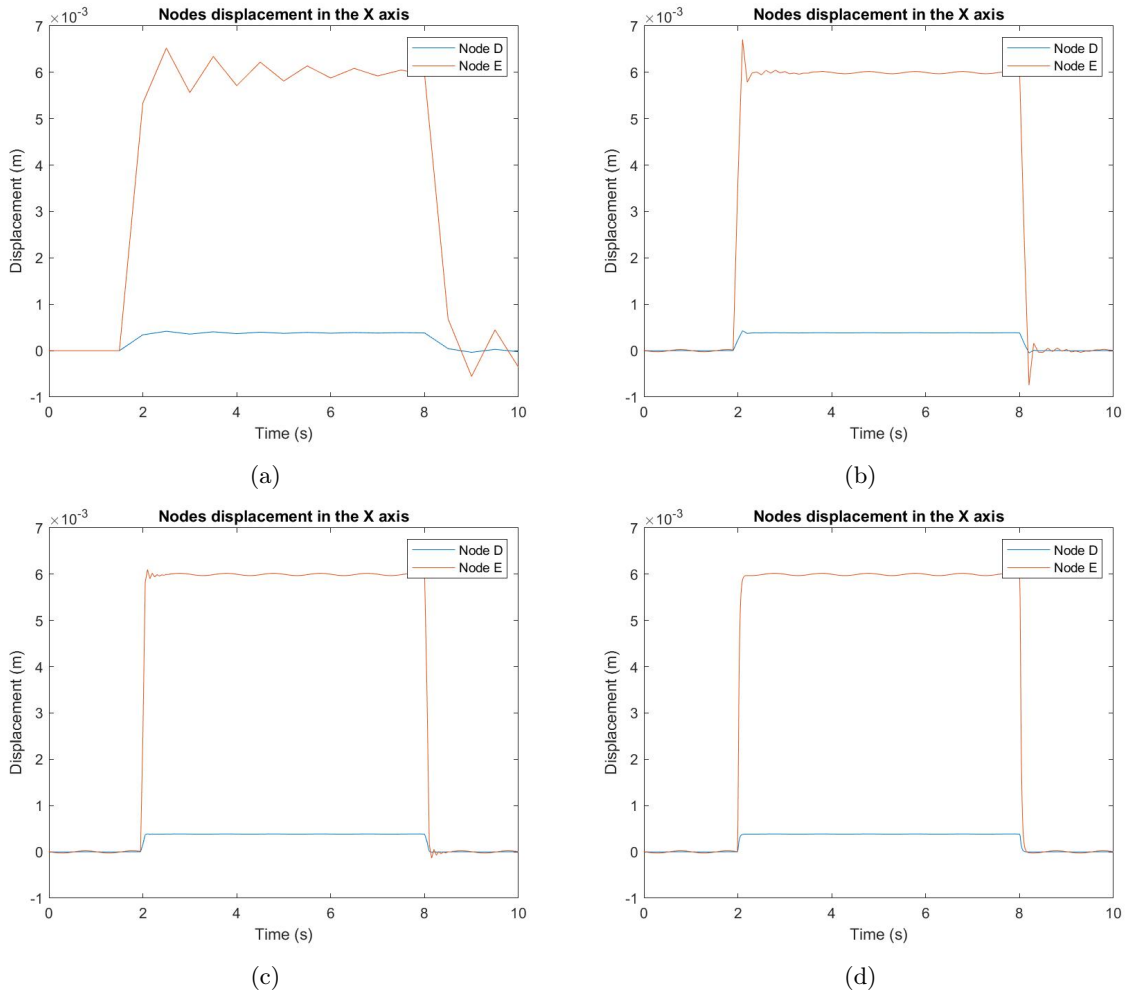


FIGURE 26 – a) Horizontal displacements using $\Delta t = 0.5s$. b) Horizontal displacements using $\Delta t = 0.1s$. c) Horizontal displacements using $\Delta t = 0.05s$. d) Horizontal displacements using $\Delta t = 0.01s$.

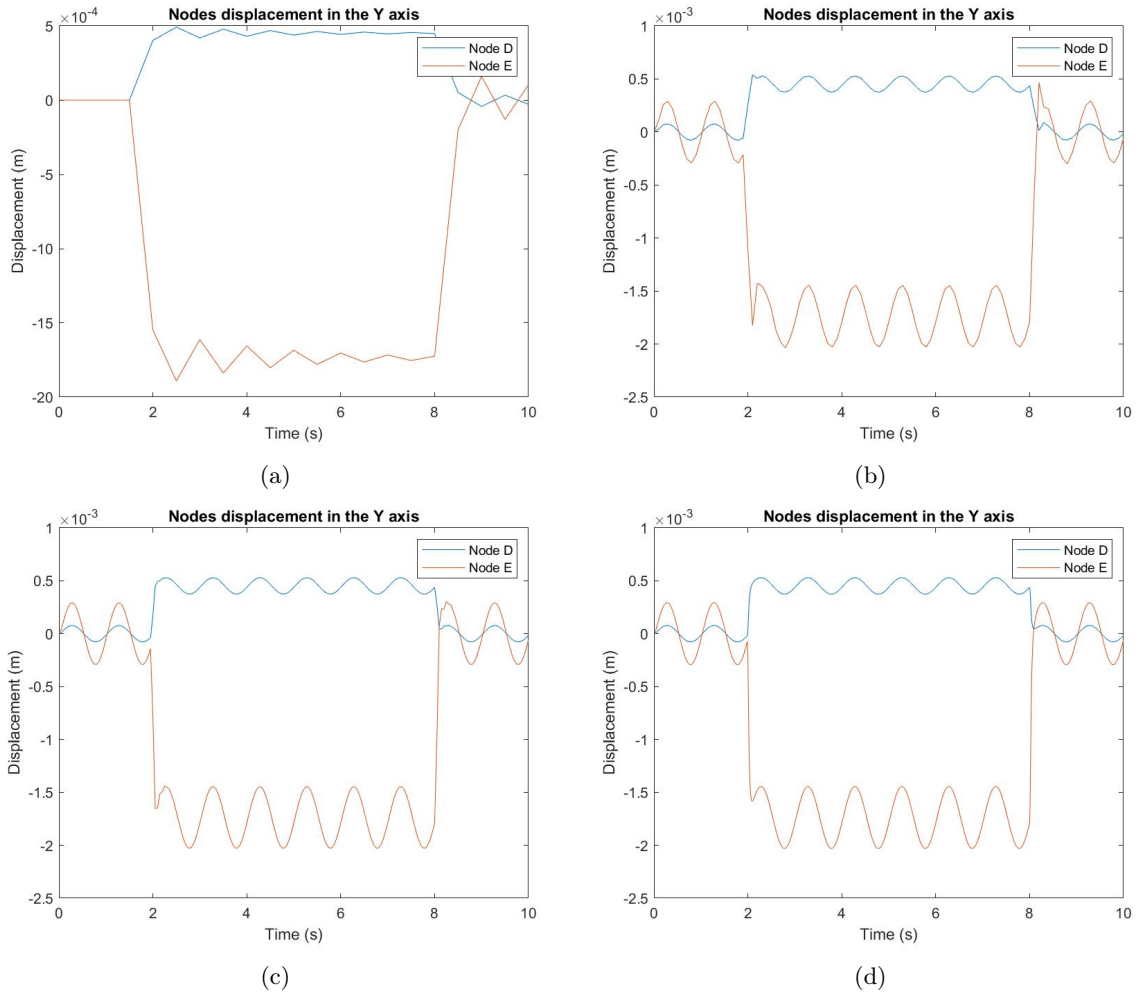


FIGURE 27 – a) Vertical displacements using $\Delta t = 0.5s$. b) Vertical displacements using $\Delta t = 0.1s$. c) Vertical displacements using $\Delta t = 0.05s$. d) Vertical displacements using $\Delta t = 0.01s$.

As we can see above, the use of $\Delta t = 0.05s$ and $\Delta t = 0.01s$ found consistent values of nodal displacements when compared to *Ansys*' results. For bigger steps in time, the shape of the graphics seem to be correct, but the precision is much lower, which is expected since we are calculating the dynamic of the system with less steps.

This makes the code much faster, but as a downsize, doesn't represent correctly the dynamic. But a step of $\Delta t = 0.001s$ didn't appear to be necessary, since with $\Delta t = 0.005s$ the results were satisfying.

We were also able to analyse the tension in the nodes A and B with the same values of Δt. The results found are the images below.

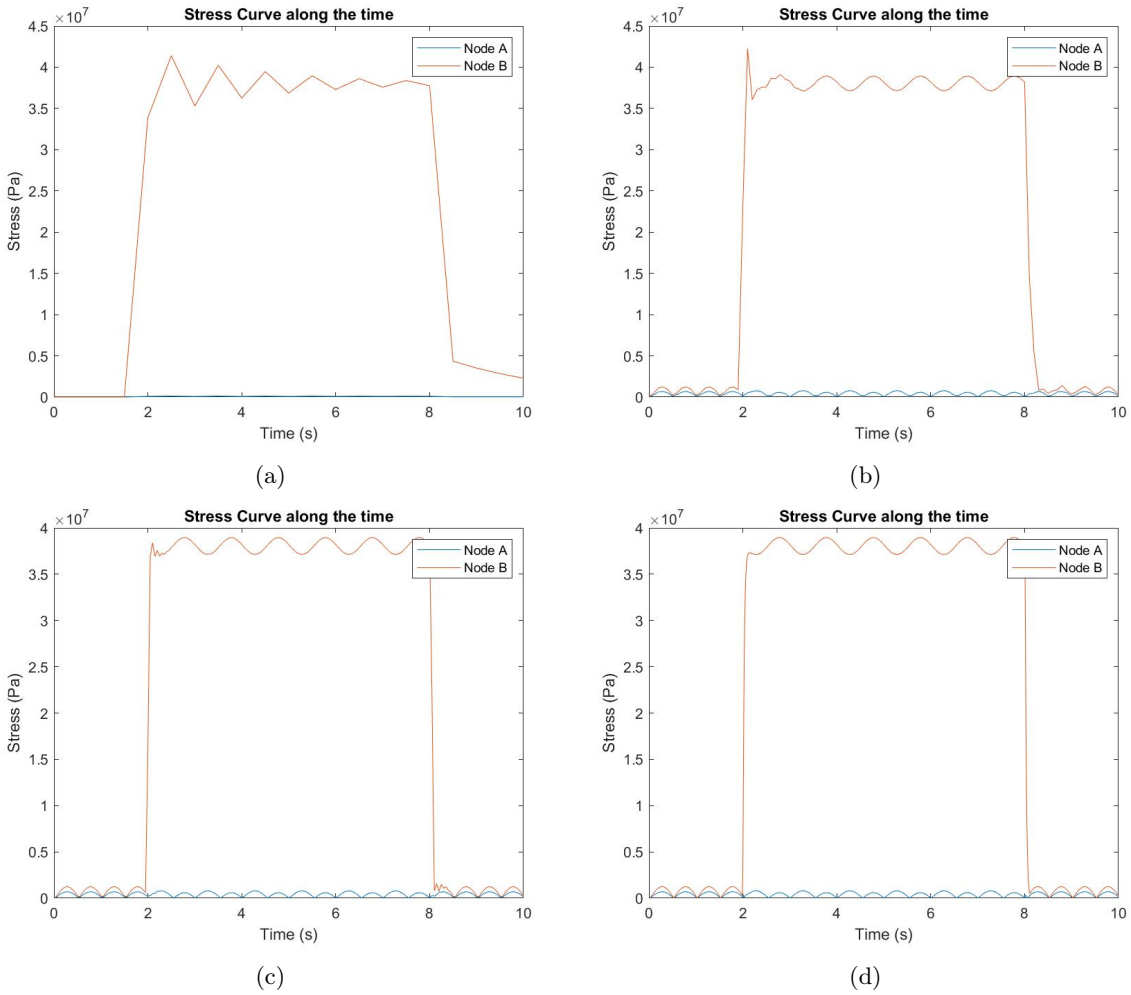


FIGURE 28 – a) Stress curve using $\Delta t = 0.5s$. b) Stress curve using $\Delta t = 0.1s$. c) Stress curve using $\Delta t = 0.05s$. d) Stress curve using $\Delta t = 0.01s$.

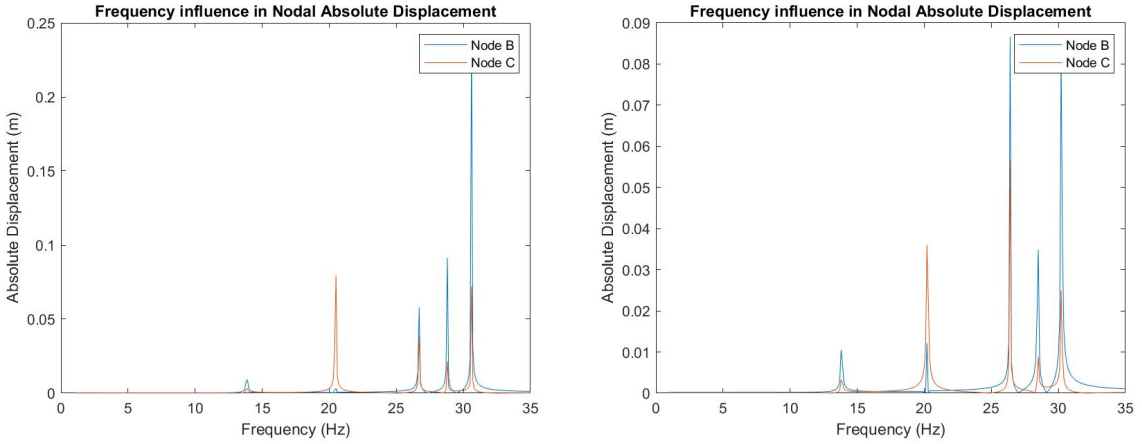
Again, we can confirm that for big time intervals such as $\Delta t = 0.5s$ and $\Delta t = 0.1s$, the measured stress is much more imprecise than using smaller steps such as $\Delta t = 0.05$ and $\Delta t = 0.01s$. This time, the values encountered didn't match the ones found in *Ansys*, although having used the same formulas and the displacement found matched perfectly. We couldn't confirm why such phenomenon occurred, but it might have to do with the precision of the calculus in *Ansys* and in *Matlab* and the number of elements used.

2.2.4 a.3) Harmonic Analysis

In order to proceed with the **Modal Analysis**, we set an *array* of frequencies, varying within the range of all the resonance frequencies observed in **Item a.1)** to observe their influence in the absolute movement of the points B and C of the structures.

As the force, we only considered the influence of the sinusoidal disturb since the other one didn't have a frequency involved, due to being a constant force in time. Therefore, varing the frequency ω from $1Hz$ to $35Hz$ with a step of 0.1 and considering 4 elements/bar, we first analysed the differences obtained between the complete and the lumped mass matrices once again. In the end, we were able to obtain the following range of absolute displacement $||u||$ of the Nodes B and C in function of the

excitation frequencies :



(a) (b)

FIGURE 29 – a) Absolute displacement using the complete mass matrix. b) Absolute displacement using the lumped mass matrix.

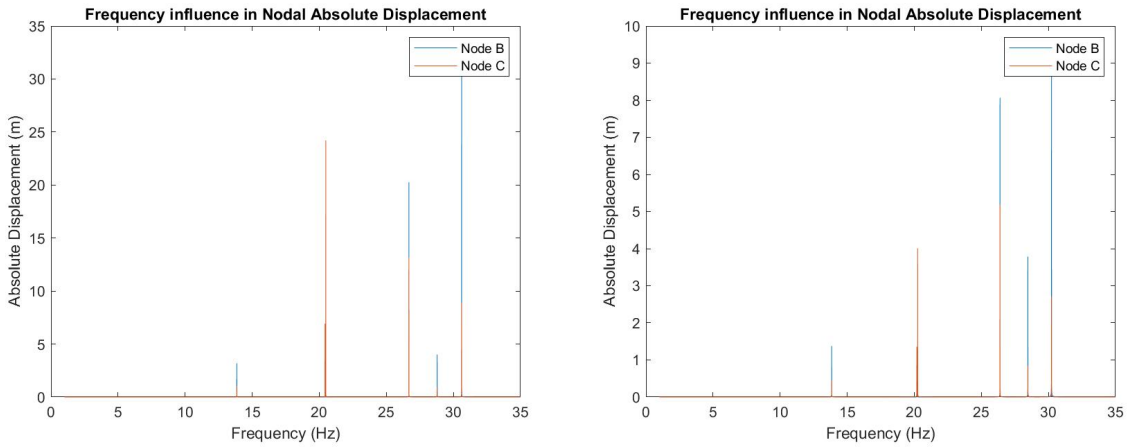
As we can see above, the peaks of absolute displacement in both nodes are in the resonance frequencies encountered in **Item a.1)**, as expected. The only difference notable is that we are only able to see 5 peaks, but that's because the second and the third resonance frequencies are too close to be distinguished with the step of 0.1 in the frequency that we used.

We can analyse from the images that the absolute displacement followed what was observed with the vibration modes - a small movement in the first vibration mode, in which the upper part of the structure has a bigger range of movement, followed by a bigger movement of the internal node C in the next 2 vibration modes, in which the lower internal bars have a bigger displacement. This was followed, in the next 3 resonance frequencies, by an increase in the node B displacement, which matched as well with the vibration modes found.

Although the absolute value of the absolute displacement varied a little between the two mass matrices used, the peaks were in the same frequencies, therefore we can conclude that both were able to show how the absolute movement vary when changing the frequency of the force applied and the influence of the resonance frequencies.

We can also conclude, by the images, that both *Matlab* and *Ansys* were able to obtain the same expected results.

We also tried to change the interval between frequencies from 0.1 to 0.001 in order to try to see if we could observe the two different peaks regarding the second and third resonance frequencies. Hence, the plots extracted are the shown below :



(a) (b)

FIGURE 30 – a) Absolute displacement using the complete mass matrix. b) Absolute displacement using the lumped mass matrix.

Now, it was easier to spot the difference between the two resonance frequencies, and with a small zoom, we were able to see the two different peaks :

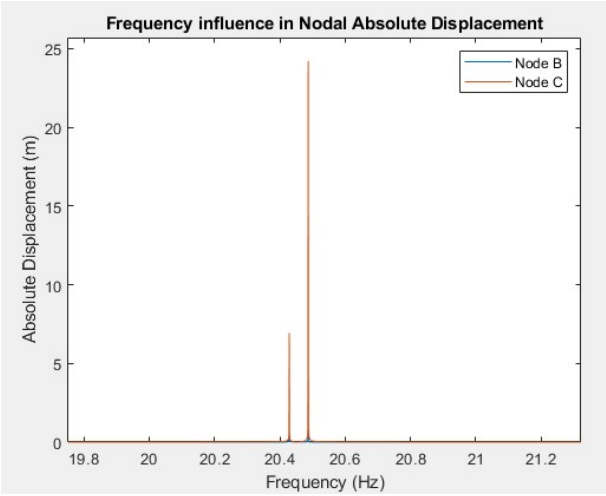


FIGURE 31 – Absolute displacement using the complete mass matrix with *zoom*.

With this experiment, we could see that bigger the refinement of the step, bigger were the absolute displacements encountered. An absolute displacement of 35 meters makes no sense for a structure of 14 meters, but this illustrates perfectly how the resonance frequencies influence the structures and make them vibrate until they break down, and the closer we get to this exact frequencies, the bigger will be the displacement.

3 Part 2

The second part of this exercise consists of a static analysis of a thin piece of material under various loads. The piece and forces are represented below :

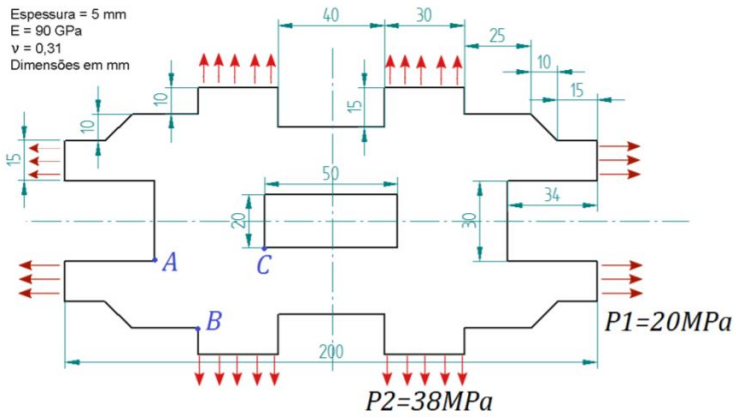


FIGURE 32 – Piece

Using *Ansys*, we can model this structure in two dimensions. Furthermore, since the piece and the pressures are symmetric in two axes, we will only have to model one quarter of the piece. Here again, the discretisation of the structure is of great importance. The following simulations were done using an element size of *0.005m*. We will later vary this parameter and inspect its influence on the results.

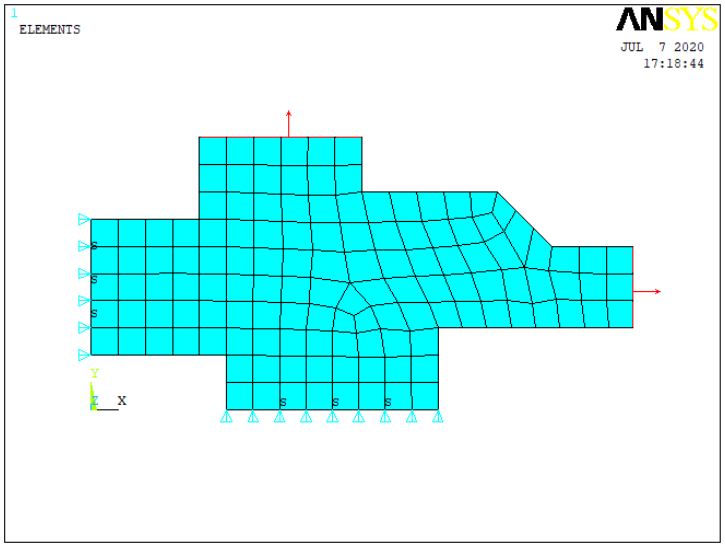


FIGURE 33 – Meshed Piece

3.1 a) Deformation

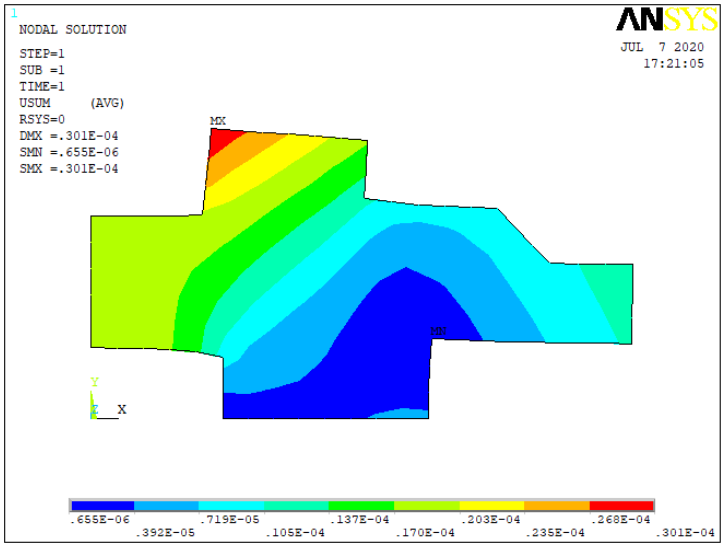


FIGURE 34 – Deformed Piece Under Load

The above figure depicts the geometry of the piece under load. The node where the displacement is maximum is denoted by *MX*, whose value of the modulus of displacement is *.301E-4m*.

3.2 b) Von Mises stress

We can furthermore inspect the internal stresses of the structure under load. Using *Ansys*' post processing function we can now plot the von Mises stresses of the whole piece and extract their values around the points A, B, and C :

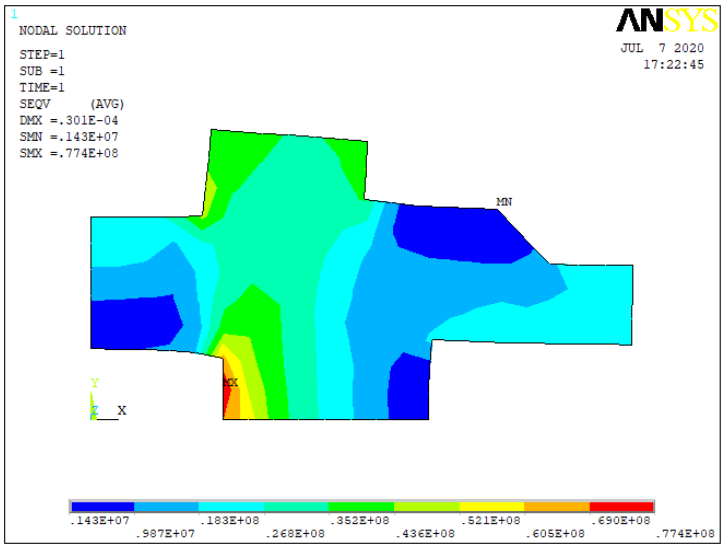


FIGURE 35 – Von Mises stresses with element size *0.005m*.

Points	Internal Stress [MPa]
A	19.8837
B	36.1909
C	63.2742

TABLE 9 – Von Mises stresses at point A, B, and C, with element size 0.005m.

Additionally, we can now vary the element size and inspect the influece it has on the results :

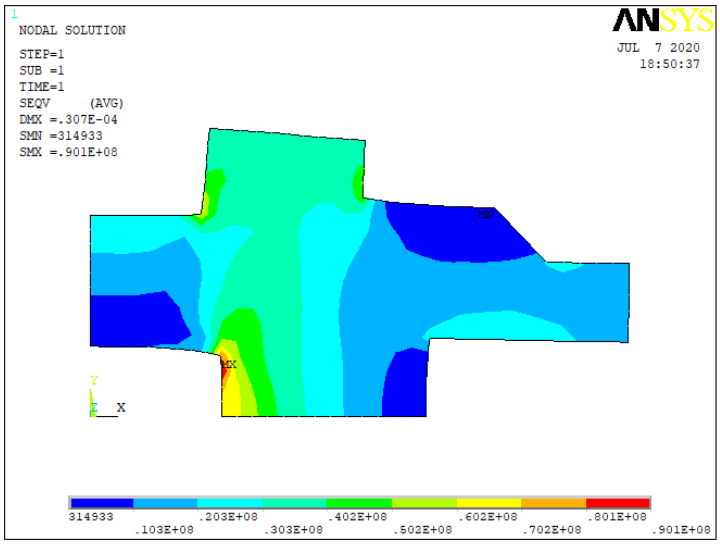


FIGURE 36 – Von Mises stresses with element size *0.003m*.

Points	Internal Stress [MPa]
A	24.0763
B	47.2686
C	78.7373

TABLE 10 – Von Mises stresses at point A, B, and C, with element size 0.003m.

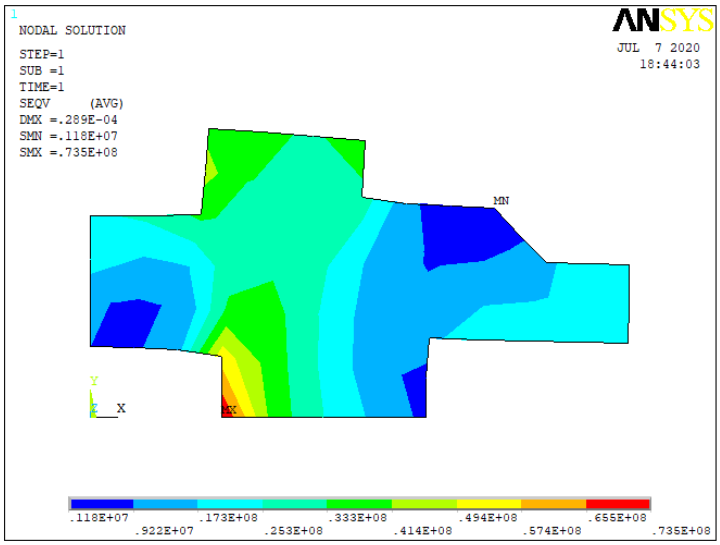


FIGURE 37 – Von Mises stresses with element size *0.01m*.

Points	Internal Stress [MPa]
A	14.9612
B	29.4009
C	53.7560

TABLE 11 – Von Mises stresses at point A, B, and C, with element size 0.01m.

We can conclude that, the smaller the element size is, the more concentrated and higher the loads are. Unfortunately, this student version of *Ansys* only allows for a limited amount of elements. We thus cannot force the element size too far down without running into this problem. The given results however already show a certain trend.

3.3 c) Maximum Stress

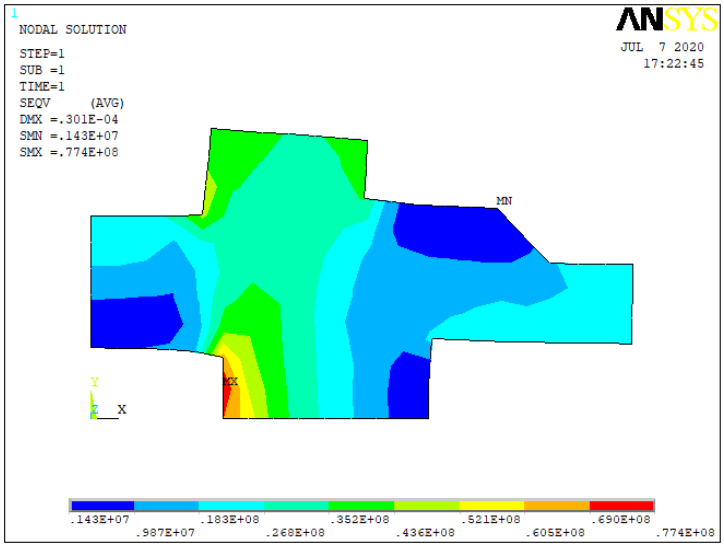


FIGURE 38 – Von Mises stresses with element size *0.005m*.

The point of maximum internal stress is denoted by *MX* and is located near the bottom left, close to a corner. Its value of stress is *77.4 MPa* for an element size of *0.005m*. Note, that this value changes as well with the element size. If we analyse figure 36, representing the von Mises plot for element size *0.003m*, we can see that the higher tensions are concentrated near the corners of the structure. We can probably lessen these tensions by virtue of using a chamfer or a fillet on these corners to reduce the angle.

4 Conclusion

First of all, this was a good exercise to understand a lot more of how and why we apply the *Finite Elements Method* theory learnt in class, using it to solve and understand a lot more of a real problem we'd have to face as Engineers.

It was also highly fruitful to be able to use a *software* used commercially such as *Ansys* and compare the differences and similarities between it and something we are more used to work with, such as *Matlab*. Although we were able to get the same results using both *software*, they were much easier and faster to find using *Ansys* once we had figured out how to use it. However, it was very hard to discover how to work with it alone, and we think one or two classes to teach future students how to use it would be of great help.

A *APDL* Scripts

A.1 Modal Analysis Script

```
finish
/clear
/PREP7
/COM,  Structural

!Parameters
F = 8000
D = 2000
L1 = 2
L2 = 3
L3 = 4
v = 0.29
E = 210E9
rho = 7650
d1i = 0.072
d1e = 0.08
d2i = 0.09
d2e = 0.1

pi = 3.141592653
A1 = pi*( (d1e/2)**2 - (d1i/2)**2 )
A2 = pi*( (d2e/2)**2 - (d2i/2)**2 )
Izz1 = pi*( d1e**4 - d1i**4 )/64
Izz2 = pi*( d2e**4 - d2i**4 )/64

!Beam
ET,1,BEAM3

!Material Properties
MPTEMP,1,0
MPDATA,DENS,1,,rho
MPDATA,EX,1,,E
MPDATA,PRXY,1,,v

!Sections
SECTYPE,   1, BEAM, CTUBE, smallBeam, 0
SECOFFSET, CENT
SECDATA,d1i/2,d1e/2,20,0,0,0,0,0,0,0
SECTYPE,   2, BEAM, CTUBE, bigBeam, 0
SECOFFSET, CENT
SECDATA,d2i/2,d2e/2,20,0,0,0,0,0,0,0

!Real constants
R,1,A1,Izz1,d1e,0,0,0,
R,2,A2,Izz2,d2e,0,0,0,

!Keypoints
K,1,0,0,,
K,2,L3,0,,
K,3,(L3-L1)/4,L2,,
K,4,L3-(L3-L1)/4,L2,,
K,5,(L3-L1)/2,2*L2,,
K,6,L3-(L3-L1)/2,2*L2,,
K,7,(L3-L1)/2,2*L2+L1,,
K,8,L3-(L3-L1)/2,2*L2+L1,,
K,9,(L3-L1)/2,2*L2+L1*2,,
```

```
K,10,L3-(L3-L1)/2,2*L2+L1*2,,
K,11,(L3-L1)/2,2*L2+L1*3,,
K,12,L3-(L3-L1)/2,2*L2+L1*3,,
K,13,(L3-L1)/2,2*L2+L1*4,,
K,14,L3-(L3-L1)/2,2*L2+L1*4,,

!Lines
LSTR,1,2    !Outer Lines
LSTR,2,4
LSTR,4,6
LSTR,6,8
LSTR,8,10
LSTR,10,12
LSTR,12,14
LSTR,13,14
LSTR,11,13
LSTR,9,11
LSTR,7,9
LSTR,5,7
LSTR,3,5
LSTR,1,3

LSTR,1,4    !Inner Lines
LSTR,2,3
LSTR,3,4
LSTR,3,6
LSTR,4,5
LSTR,5,6
LSTR,5,8
LSTR,7,8
LSTR,8,9
LSTR,9,10
LSTR,9,12
LSTR,11,12
LSTR,12,13

!Meshing
LESIZE,ALL, , ,10, ,1, , ,1,

TYPE,1      !Beam style BIG
MAT,1
SECNUM,2
REAL,2
LMESH,1,14,1

TYPE,1      !Beam style SMALL
MAT,1
SECNUM,1
REAL,1
LMESH,15,27,1

!Degees of freedom
/GO
DK,1, , , ,0,UX,UY, , , , ,
DK,2, , , ,0,UY, , , , ,

!Modal analysis
/SOLU
ANTYPE,2
MODOPT,LANB,6
EQSLV,SPAR
MXPAND,0, , ,0
LUMPM,0
PSTRES,0
MODOPT,LANB,6,0,0, ,OFF
```

```
SOLVE


finish
```

A.2 Transient Analysis Script

```
finish
/clear
/PREP7
/COM,  Structural

!Parameters
F = 8000
D = 2000
L1 = 2
L2 = 3
L3 = 4
v = 0.29
E = 210E9
rho = 7650
d1i = 0.072
d1e = 0.08
d2i = 0.09
d2e = 0.1

pi = 3.141592653
A1 = pi*( (d1e/2)**2 - (d1i/2)**2 )
A2 = pi*( (d2e/2)**2 - (d2i/2)**2 )
Izz1 = pi*( d1e**4 - d1i**4 )/64
Izz2 = pi*( d2e**4 - d2i**4 )/64

!Beam
ET,1,BEAM3

!Material Properties
MPTEMP,1,0
MPDATA,DENS,1,,rho
MPDATA,EX,1,,E
MPDATA,PRXY,1,,v

!Sections
SECTYPE,   1, BEAM, CTUBE, smallBeam, 0
SECOFFSET, CENT
SECDATA,d1i/2,d1e/2,20,0,0,0,0,0,0,0
SECTYPE,   2, BEAM, CTUBE, bigBeam, 0
SECOFFSET, CENT
SECDATA,d2i/2,d2e/2,20,0,0,0,0,0,0,0

!Real constants
R,1,A1,Izz1,d1e,0,0,0,
R,2,A2,Izz2,d2e,0,0,0,

!Keypoints
K,1,0,0,,
K,2,L3,0,,
K,3,(L3-L1)/4,L2,,
K,4,L3-(L3-L1)/4,L2,,
K,5,(L3-L1)/2,2*L2,,
K,6,L3-(L3-L1)/2,2*L2,,
K,7,(L3-L1)/2,2*L2+L1,,
K,8,L3-(L3-L1)/2,2*L2+L1,,
```

```
K,9,(L3—L1)/2,2*L2+L1*2,,
K,10,L3—(L3—L1)/2,2*L2+L1*2,,
K,11,(L3—L1)/2,2*L2+L1*3,,
K,12,L3—(L3—L1)/2,2*L2+L1*3,,
K,13,(L3—L1)/2,2*L2+L1*4,,
K,14,L3—(L3—L1)/2,2*L2+L1*4,,

!Lines
LSTR,1,2    !Outer Lines
LSTR,2,4
LSTR,4,6
LSTR,6,8
LSTR,8,10
LSTR,10,12
LSTR,12,14
LSTR,13,14
LSTR,11,13
LSTR,9,11
LSTR,7,9
LSTR,5,7
LSTR,3,5
LSTR,1,3

LSTR,1,4    !Inner Lines
LSTR,2,3
LSTR,3,4
LSTR,3,6
LSTR,4,5
LSTR,5,6
LSTR,5,8
LSTR,7,8
LSTR,8,9
LSTR,9,10
LSTR,9,12
LSTR,11,12
LSTR,12,13

!Meshing
LESIZE,ALL, , ,10, ,1, , ,1,

TYPE,1      !Beam style BIG
MAT,1
SECNUM,2
REAL,2
LMESH,1,14,1

TYPE,1      !Beam style SMALL
MAT,1
SECNUM,1
REAL,1
LMESH,15,27,1

!Degees of freedom
/GO
DK,1, , , ,0,UX,UY, , , , ,
DK,2, , , ,0,UY, , , , ,

!Transient analysis
/SOLU
ANTYPE,4

!Analysis param
NSUBST,100,0,0
ALPHAD,0.3
BETAD,0.03
```

```
TIME,10
OUTRES,ALL,ALL

!F1
*DEL,_FNCNAME
*DEL,_FNCMTID
*SET,_FNCNAME,'f1'
*DIM,%_FNCNAME%,TABLE,6,8,1,,,,%_FNCCSYS%
! Begin of equation: -8000*sin(2*{PI}*{TIME})
*SET,%_FNCNAME%(0,0,1), 0.0, -999
*SET,%_FNCNAME%(2,0,1), 0.0
*SET,%_FNCNAME%(3,0,1), 0.0
*SET,%_FNCNAME%(4,0,1), 0.0
*SET,%_FNCNAME%(5,0,1), 0.0
*SET,%_FNCNAME%(6,0,1), 0.0
*SET,%_FNCNAME%(0,1,1), 1.0, -1, 0, 2, 0, 0, 0
*SET,%_FNCNAME%(0,2,1), 0.0, -2, 0, 3.14159265358979312, 0, 0, -1
*SET,%_FNCNAME%(0,3,1),   0, -3, 0, 1, -1, 3, -2
*SET,%_FNCNAME%(0,4,1), 0.0, -1, 0, 1, -3, 3, 1
*SET,%_FNCNAME%(0,5,1), 0.0, -1, 9, 1, -1, 0, 0
*SET,%_FNCNAME%(0,6,1), 0.0, -2, 0, -8000, 0, 0, -1
*SET,%_FNCNAME%(0,7,1), 0.0, -3, 0, 1, -2, 3, -1
*SET,%_FNCNAME%(0,8,1), 0.0, 99, 0, 1, -3, 0, 0
! End of equation: -8000*sin(2*{PI}*{TIME})

!F2
*DEL,_FNCNAME
*DEL,_FNCMTID
*SET,_FNCNAME,'f2'
*DIM,%_FNCNAME%,TABLE,6,3,4,,,,%_FNCCSYS%
! Begin of equation: {TIME}
*SET,%_FNCNAME%(0,0,1), 0, -999
*SET,%_FNCNAME%(2,0,1), 0.0
*SET,%_FNCNAME%(3,0,1), 0.0
*SET,%_FNCNAME%(4,0,1), 0.0
*SET,%_FNCNAME%(5,0,1), 0.0
*SET,%_FNCNAME%(6,0,1), 0.0
*SET,%_FNCNAME%(0,1,1), 1.0, 99, 0, 1, 1, 0, 0
*SET,%_FNCNAME%(0,2,1),   0
*SET,%_FNCNAME%(0,3,1),   0
! End of equation: {TIME}
!
! Begin of equation: 0
*SET,%_FNCNAME%(0,0,2), 2, -999
*SET,%_FNCNAME%(2,0,2), 0.0
*SET,%_FNCNAME%(3,0,2), 0.0
*SET,%_FNCNAME%(4,0,2), 0.0
*SET,%_FNCNAME%(5,0,2), 0.0
*SET,%_FNCNAME%(6,0,2), 0.0
*SET,%_FNCNAME%(0,1,2), 1.0, 99, 0, 0, 0, 0, 0
*SET,%_FNCNAME%(0,2,2),   0
*SET,%_FNCNAME%(0,3,2),   0
! End of equation: 0
!
! Begin of equation: 2000
*SET,%_FNCNAME%(0,0,3), 8, -999
*SET,%_FNCNAME%(2,0,3), 0.0
*SET,%_FNCNAME%(3,0,3), 0.0
*SET,%_FNCNAME%(4,0,3), 0.0
*SET,%_FNCNAME%(5,0,3), 0.0
*SET,%_FNCNAME%(6,0,3), 0.0
*SET,%_FNCNAME%(0,1,3), 1.0, 99, 0, 2000, 0, 0, 0
*SET,%_FNCNAME%(0,2,3),   0
*SET,%_FNCNAME%(0,3,3),   0
! End of equation: 2000
```

```
!
! Begin of equation: 0
*SET,%_FNCNAME%(0,0,4), 60, -999
*SET,%_FNCNAME%(2,0,4), 0.0
*SET,%_FNCNAME%(3,0,4), 0.0
*SET,%_FNCNAME%(4,0,4), 0.0
*SET,%_FNCNAME%(5,0,4), 0.0
*SET,%_FNCNAME%(6,0,4), 0.0
*SET,%_FNCNAME%(0,1,4), 1.0, 99, 0, 0, 0, 0, 0
*SET,%_FNCNAME%(0,2,4),   0
*SET,%_FNCNAME%(0,3,4),   0
! End of equation: 0


FK,13,FY, %F1%
FK,14,FY, %F1%

FK,5,FX, %F2%
FK,7,FX, %F2%
FK,9,FX, %F2%
FK,11,FX, %F2%
FK,13,FX, %F2%

SOlVE
FINISH
```

A.3 Harmonic Analysis Script

```
finish
/clear
/PREP7
/COM,  Structural

!Parameters
F = 8000
D = 2000
L1 = 2
L2 = 3
L3 = 4
v = 0.29
E = 210E9
rho = 7650
d1i = 0.072
d1e = 0.08
d2i = 0.09
d2e = 0.1

pi = 3.141592653
A1 = pi*( (d1e/2)**2 - (d1i/2)**2 )
A2 = pi*( (d2e/2)**2 - (d2i/2)**2 )
Izz1 = pi*( d1e**4 - d1i**4 )/64
Izz2 = pi*( d2e**4 - d2i**4 )/64

!Beam
ET,1,BEAM3

!Material Properties
MPTEMP,1,0
MPDATA,DENS,1,,rho
MPDATA,EX,1,,E
MPDATA,PRXY,1,,v

```

```
!Sections
SECTYPE,   1, BEAM, CTUBE, smallBeam, 0
SECOFFSET, CENT
SECDATA,d1i/2,d1e/2,20,0,0,0,0,0,0,0
SECTYPE,   2, BEAM, CTUBE, bigBeam, 0
SECOFFSET, CENT
SECDATA,d2i/2,d2e/2,20,0,0,0,0,0,0,0

!Real constants
R,1,A1,Izz1,d1e,0,0,0,
R,2,A2,Izz2,d2e,0,0,0,

!Keypoints
K,1,0,0,,
K,2,L3,0,,
K,3,(L3-L1)/4,L2,,
K,4,L3-(L3-L1)/4,L2,,
K,5,(L3-L1)/2,2*L2,,
K,6,L3-(L3-L1)/2,2*L2,,
K,7,(L3-L1)/2,2*L2+L1,,
K,8,L3-(L3-L1)/2,2*L2+L1,,
K,9,(L3-L1)/2,2*L2+L1*2,,
K,10,L3-(L3-L1)/2,2*L2+L1*2,,
K,11,(L3-L1)/2,2*L2+L1*3,,
K,12,L3-(L3-L1)/2,2*L2+L1*3,,
K,13,(L3-L1)/2,2*L2+L1*4,,
K,14,L3-(L3-L1)/2,2*L2+L1*4,,

!Lines
LSTR,1,2    !Outer Lines
LSTR,2,4
LSTR,4,6
LSTR,6,8
LSTR,8,10
LSTR,10,12
LSTR,12,14
LSTR,13,14
LSTR,11,13
LSTR,9,11
LSTR,7,9
LSTR,5,7
LSTR,3,5
LSTR,1,3

LSTR,1,4    !Inner Lines
LSTR,2,3
LSTR,3,4
LSTR,3,6
LSTR,4,5
LSTR,5,6
LSTR,5,8
LSTR,7,8
LSTR,8,9
LSTR,9,10
LSTR,9,12
LSTR,11,12
LSTR,12,13

!Meshing
LESIZE,ALL, , ,10, ,1, , ,1,

TYPE,1      !Beam style BIG
MAT,1
SECNUM,2
REAL,2
```

```
LMESH,1,14,1

TYPE,1      !Beam style SMALL
MAT,1
SECNUM,1
REAL,1
LMESH,15,27,1

!Degees of freedom
/GO
DK,1, , , ,0,UX,UY, , , , ,
DK,2, , , ,0,UY, , , , ,


!Harmonic analysis
f_0 = 10
f_f = 35
steps = 200

/SOLU
ANTYPE,3

HARFRQ,f_0,f_f,
NSUBST,steps,
KBC,1

FK,13,FY,-1000,
FK,14,FY,-1000,

!OUTRES,ALL,ALL

/STATUS,SOLU
SOLVE
FINISH

!sqrt(nsol(173 ,U,X)^2+nsol(173,U,Y)^2)
!sqrt(nsol(155,U,X)^2+nsol(155,U,Y)^2)
```

A.4 Part 2 Script

```
finish
/CLEAR

!Parameters
thickness = 5e-3
E = 90e9
v = 0.31
P1 = 20e6
P2 = 38e6
elem_size = 5e-3


/PREP7

!Plane
ET,1,PLANE182

!Material Properties
MPTEMP,1,0
MPDATA,EX,1,,E
MPDATA,PRXY,1,,v

```

```
R,1,thickness,

!Keypoints
K,1,25e-3,10e-3
K,2,25e-3,0
K,3,64e-3,0
K,4,64e-3,15e-3
K,5,100e-3,15e-3
K,6,100e-3,30e-3
K,7,85e-3,30e-3
K,8,75e-3,40e-3
K,9,50e-3,40e-3
K,10,50e-3,50e-3
K,11,20e-3,50e-3
K,12,20e-3,35e-3
K,13,0,35e-3
K,14,0,10e-3

! criar as linhas
L,1,2
L,2,3
L,3,4
L,4,5
L,5,6
L,6,7
L,7,8
L,8,9
L,9,10
L,10,11
L,11,12
L,12,13
L,13,14
L,14,1

lsel,s,line,,1,14
al,all
alls

! MALHAGEM
TYPE,1
MAT,1
ESIZE,elem_size,0,

AMESH,1

FINISH
/SOL

!Static Analysis
ANTYPE,0

!Symmetry conditions
DL,2, ,SYMM
DL,13, ,SYMM

SFL,5,PRES,-P1,
SFL,10,PRES,-P2,

SOLVE
```

B *Matlab* Scripts

B.1 Main Script

```
close all

% Definição das variáveis
global F D t1 t2 L1 L2 L3 v d1i d1e d2i d2e E ro Elementos
global alfa beta gama beta2
F=8000;
D=2000;
t1=2;
t2=8;
L1=2;
L2=3;
L3=4;
v=0.29;
d1i=0.072;
d1e=0.08;
d2i=0.09;
d2e=0.1;
E=210e9;
ro=7650;
Elementos=4;

alfa=0.3;
beta=0.03;
gama=1/2;
beta2=1/4;


tic
%Pré-processamento - Definição de n s e barras
nos_originais=[0 0;4 0;0.5 3;3.5 3;1 6;3 6;1 8;3 8;1 10;3 10;1 12;3 12;1 14;3 14];
barras=criabarras(nos_originais);

%Cria matrizes por elemento
[K,M,Malternativa]=criamatrizes;

%Cria os elementos e atualiza os n s
[nos_elementos,elementos]=criaelementos(barras);

%Cria as matrizes globais
[Kglobs,Mglobs,Mglobs2,Lele]=matrizesglobais(elementos,nos_elementos,K,M,Malternativa);

toc

%Análise modal
tic
[A,w1,CC]=modal(nos_elementos,elementos,Kglobs,Mglobs,Lele);
toc
%Análise modal com a matriz 'lumped'
tic
[B,w2,CC]=modal(nos_elementos,elementos,Kglobs,Mglobs2,Lele);
toc

%Análise transiente
tic
[MCC,KCC,nos_F1,nos_F2] = transiente(nos_elementos,CC,Mglobs,Kglobs);
toc
```

```
%Análise transiente com a matriz 'lumped'
tic
[MCC2,KCC,nos_F1,nos_F2] = transiente(nos_elementos,CC,Mglobs2,Kglobs);
toc

%Análise harm nica
tic
harmonica(nos_elementos,MCC,KCC,nos_F1,nos_F2)
toc
%Análise harm nica com a matriz 'lumped'
tic
harmonica(nos_elementos,MCC2,KCC,nos_F1,nos_F2)
toc
```

B.2 Pre-Processing Functions

```
function barras = criabarras(nos)
    %Função responsável por criar as matrizes que armazenam as barras da
    %torre, posteriormente discretizadas em elementos
    %A organização dentro da matriz é de cada linha ser uma barra e os
    %valores contidos dentro serem
    %barras=[x1 y1 x2 y2 di de]
    global d1i d1e d2i d2e
    barras=zeros(27,6);
    y=1;
    direita=1;
    for i=1:2:length(nos)-2
        if nos(i,1)<nos(i+2,1)
            if nos(i,1)==0
                barras(y,:)=...
                    [nos(i,1),nos(i,2),nos(i+1,1),nos(i+1,2),d2i,d2e];
            else
                barras(y,:)=...
                    [nos(i,1),nos(i,2),nos(i+1,1),nos(i+1,2),d1i,d1e];
            end
            barras(y+1,:)=...
                [nos(i,1),nos(i,2),nos(i+2,1),nos(i+2,2),d2i,d2e];
            barras(y+2,:)=...
                [nos(i+1,1),nos(i+1,2),nos(i+3,1),nos(i+3,2),d2i,d2e];
            barras(y+3,:)=...
                [nos(i,1),nos(i,2),nos(i+3,1),nos(i+3,2),d1i,d1e];
            barras(y+4,:)=...
                [nos(i+1,1),nos(i+1,2),nos(i+2,1),nos(i+2,2),d1i,d1e];
            y=y+5;
        else
            barras(y,:)=[nos(i,1),nos(i,2),nos(i+1,1),nos(i+1,2),d1i,d1e];
            barras(y+1,:)=...
                [nos(i,1),nos(i,2),nos(i+2,1),nos(i+2,2),d2i,d2e];
            barras(y+2,:)=...
                [nos(i+1,1),nos(i+1,2),nos(i+3,1),nos(i+3,2),d2i,d2e];
            if direita==1
                barras(y+3,:)=...
                    [nos(i,1),nos(i,2),nos(i+3,1),nos(i+3,2),d1i,d1e];
                direita=0;
            else
                barras(y+3,:)=...
                    [nos(i+1,1),nos(i+1,2),nos(i+2,1),nos(i+2,2),d1i,d1e];
                direita=1;
            end
            y=y+4;
        end
    end
```

```
    barras(y,:)=[nos(length(nos)-1,1),...
      nos(length(nos)-1,2),nos(length(nos),1),nos(length(nos),2),d2i,d2e];
    plotatorre(nos,barras,1,0);
end
```

```
function [nos_elementos,elementos] = criaelementos(barras)
    %Função responsável por fazer a discretização desejada, dividindo as
    %barras em elementos e armazenando tanto os elementos que surgem quanto
    %os n s.
    %Será uma matriz gigantesca dos elementos em que cada elemento será
    %salvo como
    %elementos=[x1,y1,x2,y2,di,de]
    global Elementos
    nos_elementos=[];
    elementos=[];

    for i=1:length(barras)
        for j=1:Elementos
            if j==1
                x1=barras(i,1);
                y1=barras(i,2);
                if length(nos_elementos)~=0
                    if length(find(nos_elementos(:,1)==x1))==0 ||...
                            length(find(nos_elementos(:,2)==y1))==0
                            nos_elementos(end+1,:)=barras(i,1:2);
                    else
                        a=(find(nos_elementos(:,1)==x1));
                        b=(find(nos_elementos(:,2)==y1));
                        jatem=0;
                        for m=1:length(a)
                            for n=1:length(b)
                                if a(m)==b(n)
                                    jatem=1;
                                end
                            end
                        end
                        if jatem==0
                            nos_elementos(end+1,:)=barras(i,1:2);
                        end
                    end
                else
                    nos_elementos(end+1,:)=barras(i,1:2);
                end
                x2=barras(i,1)+(j/Elementos)*(barras(i,3)-barras(i,1));
                y2=barras(i,2)+(j/Elementos)*(barras(i,4)-barras(i,2));
                if Elementos~=1 || i==length(barras)
                    nos_elementos(end+1,:)=[x2,y2];
                end
                elementos(end+1,:)=[barras(i,1:2),x2,y2,barras(i,5:6)];
            elseif (j==Elementos && j~=1)
                x1=x2;
                y1=y2;
                x2=barras(i,3);
                y2=barras(i,4);
                if length(find(nos_elementos(:,1)==x2))==0 ||...
                        length(find(nos_elementos(:,2)==y2))==0
                        nos_elementos(end+1,:)=barras(i,3:4);
                else
                    a=(find(nos_elementos(:,1)==x2));
                    b=(find(nos_elementos(:,2)==y2));
                    jatem=0;
                    for m=1:length(a)
```

```
                        for n=1:length(b)
                            if a(m)==b(n)
                                jatem=1;
                            end
                        end
                    end
                    if jatem==0
                        nos_elementos(end+1,:)=barras(i,3:4);
                    end
                end
                elementos(end+1,:)=[x1,y1,barras(i,3:6)];
            else
                x1=x2;
                y1=y2;
                x2=barras(i,1)+(j/Elementos)*(barras(i,3)-barras(i,1));
                y2=barras(i,2)+(j/Elementos)*(barras(i,4)-barras(i,2));
                nos_elementos(end+1,:)=[x2,y2];
                elementos(end+1,:)=[x1 y1 x2 y2 barras(i,5:6)];
            end
        end
    end

    plotatorre(nos_elementos,elementos,2,0);
end
```

```
function plotatorre(nos, barras,titulo,modo)
    %Função para plotar a torre sem deformações
    global d2e
    if titulo~=3
       figure()

        for i = 1:length(barras)
            if barras(i,6) == d2e
                plot([barras(i,1) barras(i,3)],...
                    [barras(i,2) barras(i,4)],'LineWidth',3);
            else
                plot([barras(i,1) barras(i,3)],...
                    [barras(i,2) barras(i,4)],'LineWidth',1);
            end
            hold on
        end
        for i = 1:length(nos)
           scatter(nos(i,1),nos(i,2),'filled');
           hold on
        end

        xlabel('X axis');
        ylabel('Y axis');
        if titulo==1
            title('Tower original geometry');
        else
            title('Tower geometry with the division of the elements')
        end
        axis image
        hold off
        if titulo==1
        saveas(gcf,'imgbarras.jpg')
        else
            saveas(gcf,'imgelementos.jpg')
        end
    else
        for i = 1:length(barras)
```

```
            if barras(i,6) == d2e
                plot([barras(i,1) barras(i,3)],...
                    [barras(i,2) barras(i,4)],'--k','LineWidth',2);
            else
                plot([barras(i,1) barras(i,3)],...
                    [barras(i,2) barras(i,4)],'--k','LineWidth',1);
            end
            hold on
        end
    end
end
```

```
function [K,M,Malternas] = criamatrizes
%Função para criar as
%Matrizes de rigidez e de massa por elemento
%Matrizes ordenadas por [u1 w1 phi1 u2 w2 phi2]

global ro E

%Matriz de rigidez por elemento
Coord1=@(L,A) E*A/L;
Coord2=@(L,I) E*I/L^3;
Coord3=@(L,I) E*I/L^2;
Coord4=@(L,I) E*I/L;
K=@(L,A,I) ...
[Coord1(L,A) 0 0 -Coord1(L,A) 0 0; ...
0 12*Coord2(L,I) 6*Coord3(L,I) 0 -12*Coord2(L,I) 6*Coord3(L,I);...
0 6*Coord3(L,I) 4*Coord4(L,I) 0 -6*Coord3(L,I) 2*Coord4(L,I);...
-Coord1(L,A) 0 0 Coord1(L,A) 0 0; ...
0 -12*Coord2(L,I) -6*Coord3(L,I) 0 12*Coord2(L,I) -6*Coord3(L,I);...
0 6*Coord3(L,I) 2*Coord4(L,I) 0 -6*Coord3(L,I) 4*Coord4(L,I)];

%Matriz de massa por elemento
Coord5=@(L,A) ro*A*L/420;
Coord6=@(L,A) ro*A*L^2/420;
Coord7=@(L,A) ro*A*L^3/420;
M=@(L,A) [140*Coord5(L,A) 0 0 70*Coord5(L,A) 0 0; ...
0 156*Coord5(L,A) 22*Coord6(L,A) 0 54*Coord5(L,A) -13*Coord6(L,A);...
0 22*Coord6(L,A) 4*Coord7(L,A) 0 13*Coord6(L,A) -3*Coord7(L,A);...
70*Coord5(L,A) 0 0 140*Coord5(L,A) 0 0;...
0 54*Coord5(L,A) 13*Coord6(L,A) 0 156*Coord5(L,A) -22*Coord6(L,A);...
0 -13*Coord6(L,A) -3*Coord7(L,A) 0 -22*Coord6(L,A) 4*Coord7(L,A)];

%Matriz de massa 'lumped' por elemento
Coord8=@(L,A) ro*A*L/78;
Coord9=@(L,A) ro*A*L^3/78;
Malternas=@(L,A)[39*Coord8(L,A) 0 0 0 0 0; 0 39*Coord8(L,A) 0 0 0 0;...
0 0 Coord9(L,A) 0 0 0; 0 0 0 39*Coord8(L,A) 0 0; ...
0 0 0 0 39*Coord8(L,A) 0; 0 0 0 0 0 Coord9(L,A)];
end
```

```
function [Kglobs,Mglobs,Mglobs2,Lele]=...
    matrizesglobais(elementos,nos_elementos,K,M,Malternas)

%Função para, uma vez criar as matrizes globais necessárias
Mglobs=zeros(3*length(nos_elementos));
Kglobs=zeros(3*length(nos_elementos));
Mglobs2=zeros(3*length(nos_elementos));

```

```
Lele=zeros(1,length(elementos));
Iele=zeros(1,length(elementos));
Aele=zeros(1,length(elementos));

for i=1:length(elementos)

    Lele(i)=sqrt((elementos(i,1)-elementos(i,3))^2+...
        (elementos(i,2)-elementos(i,4))^2);
    Iele(i)=pi*(elementos(i,6)^4-elementos(i,5)^4)/64;
    Aele(i)=pi*((elementos(i,6)/2)^2-(elementos(i,5)/2)^2);
    cosseno=(elementos(i,3)-elementos(i,1))/Lele(i);
    seno=(elementos(i,4)-elementos(i,2))/Lele(i);
    T=[cosseno seno 0 0 0 0; -seno cosseno 0 0 0 0; 0 0 1 0 0 0;...
        0 0 0 cosseno seno 0; 0 0 0 -seno cosseno 0; 0 0 0 0 0 1];
    Mele=transpose(T)*M(Lele(i),Aele(i))*(T);

    Mele2=transpose(T)*Malternas(Lele(i),Aele(i))*(T);

    Kele=transpose(T)*K(Lele(i),Aele(i),Iele(i))*(T);
    achax1=find(nos_elementos(:,1)==elementos(i,1));
    achay1=find(nos_elementos(achax1,2)==elementos(i,2));
    achax2=find(nos_elementos(:,1)==elementos(i,3));
    achay2=find(nos_elementos(achax2,2)==elementos(i,4));
    a=3*achax1(achay1);
    b=3*achax2(achay2);
    Mglobs(a-2:a,a-2:a)=Mglobs(a-2:a,a-2:a)+Mele(1:3,1:3);
    Mglobs(b-2:b,b-2:b)=Mglobs(b-2:b,b-2:b)+Mele(4:6,4:6);
    Mglobs(a-2:a,b-2:b)=Mglobs(a-2:a,b-2:b)+Mele(1:3,4:6);
    Mglobs(b-2:b,a-2:a)=Mglobs(b-2:b,a-2:a)+Mele(4:6,1:3);

    Mglobs2(a-2:a,a-2:a)=Mglobs2(a-2:a,a-2:a)+Mele2(1:3,1:3);
    Mglobs2(b-2:b,b-2:b)=Mglobs2(b-2:b,b-2:b)+Mele2(4:6,4:6);
    Mglobs2(a-2:a,b-2:b)=Mglobs2(a-2:a,b-2:b)+Mele2(1:3,4:6);
    Mglobs2(b-2:b,a-2:a)=Mglobs2(b-2:b,a-2:a)+Mele2(4:6,1:3);

    Kglobs(a-2:a,a-2:a)=Kglobs(a-2:a,a-2:a)+Kele(1:3,1:3);
    Kglobs(b-2:b,b-2:b)=Kglobs(b-2:b,b-2:b)+Kele(4:6,4:6);
    Kglobs(a-2:a,b-2:b)=Kglobs(a-2:a,b-2:b)+Kele(1:3,4:6);
    Kglobs(b-2:b,a-2:a)=Kglobs(b-2:b,a-2:a)+Kele(4:6,1:3);

end
end
```

B.3 Modal Analysis Script

```
function [A,w1,CC]=modal(nos_elementos,elementos,Kglobs,Mglobs,Lele)

%Função que realiza a análise modal

tic
CC=3*find(nos_elementos(:,2)==0 & (nos_elementos(:,1)==0 | nos_elementos(:,1)==4));

MCC=Mglobs;
KCC=Kglobs;
KCC(CC(2)-1,:)=[];
KCC(:,CC(2)-1)=[];
KCC(CC(1)-2:CC(1)-1,:)=[];
KCC(:,CC(1)-2:CC(1)-1)=[];

MCC(CC(2)-1,:)=[];
MCC(:,CC(2)-1)=[];
MCC(CC(1)-2:CC(1)-1,:)=[];
```

```
MCC(:,CC(1)-2:CC(1)-1)=[];

[A,V]=eigs(inv(MCC)*KCC,6,'smallestabs');

w1=sqrt(abs(V))/(2*pi);

toc
plotavibsmodes(A,nos_elementos,elementos,Lele)

end
```

```
function plotavibsmodes(A,nos_elementos,elementos,Lele)
    %Função para realizar a plotagem dos modos de vibração
    global d2e

    A=[zeros(1,6); zeros(1,6); A];
    posicao=find(nos_elementos(:,1)==4 & nos_elementos(:,2)==0);
    A=-[A(1:3*posicao-2,:); zeros(1,6); A(3*posicao-1:end,:)];

    xvet=zeros(length(elementos),21);
    yvet=zeros(length(elementos),21);

    nosnovos=nos_elementos;
    for i=1:length(A(1,:))
        ipx=20; %Numero de pontos dentro do elemento usados para interpolar
        figure()
        plotatorre(nos_elementos,elementos,3,i)
        for j=1:length(elementos)
            %Percorre todos os elementos pra encontrar seus autovetores
            xf=[];
            %Vetor para armazenar os pontos dentro do elemento
            %que serão aproximados por Hermite/Lagrange
            Df=[]; %Vetor para armazenar os
            %autovetores associados s  aquele elemento
            achax1=find(nos_elementos(:,1)==elementos(j,1));
            achay1=find(nos_elementos(achax1,2)==elementos(j,2));
            achax2=find(nos_elementos(:,1)==elementos(j,3));
            achay2=find(nos_elementos(achax2,2)==elementos(j,4));
            a=3*achax1(achay1);
            b=3*achax2(achay2);
            Autovet(1:3)=A(a-2:a,i);
            %Salva as coordenadas associadas a esses pontos pra esse modo
            Autovet(4:6)=A(b-2:b,i);
            nosnovos(a/3,1)=nos_elementos(a/3,1)+Autovet(1);
            nosnovos(a/3,2)=nos_elementos(a/3,2)+Autovet(2);
            nosnovos(b/3,1)=nos_elementos(b/3,1)+Autovet(4);
            nosnovos(b/3,2)=nos_elementos(b/3,2)+Autovet(5);
            cosseno=(elementos(j,3)-elementos(j,1))/Lele(j);
            seno=(elementos(j,4)-elementos(j,2))/Lele(j);
            for ip=1:ipx+1
                x=Lele(j)*(ip-1)/ipx;
                %pega o ponto x do elemento discretizado pra interpolar
                L=Lele(j);
                %Lagrange
                N(1)=(L-x)/L;
                N(2)=x/L;
                %Hermite
                N(3)=1-3*x^2/L^2+2*x^3/L^3;
                N(4)=x-2*x^2/L+x^3/L^2;
                N(5)=3*x^2/L^2-2*x^3/L^3;
                N(6)=-x^2/L+x^3/L^2;
                Df(ip)=N(3)*Autovet(2)+N(4)*Autovet(3)+...
```

```
                    N(5)*Autovet(5)+N(6)*Autovet(6);
                xf(ip)=x+N(1)*Autovet(1)+N(2)*Autovet(4);
            end
            if i==1
            pos=find(nos_elementos(:,1)==elementos(j,1) &...
                nos_elementos(:,2)==elementos(j,2));
            xvet(j,:)=nosnovos(pos,1)+xf*cosseno-Df*seno;
            yvet(j,:)=nosnovos(pos,2)+Df*cosseno+xf*seno;
            else
            xvet(j,:)=elementos(j,1)+xf*cosseno-Df*seno;
            yvet(j,:)=elementos(j,2)+Df*cosseno+xf*seno;
            end
            if elementos(j,6)==d2e
                plot(xvet(j,:),yvet(j,:),'r','LineWidth',3)
                hold on
            else
                plot(xvet(j,:),yvet(j,:),'r','LineWidth',1)
                hold on
            end
        end
        hold on
        if i==1
            title("Tower's 1st Vibration Mode")
        elseif i==2
            title("Tower's 2nd Vibration Mode")
        elseif i==3
            title("Tower's 3rd Vibration Mode")
        elseif i>3
            titulo=strcat("Tower's ",num2str(i),"th Vibration Mode");
            title(titulo)
        end
        xlabel('X axis');
        ylabel('Y axis');
        axis image
    end

end
```

B.4 Transient Analysis Script

```
function [MCC,KCC,nos_F1,nos_F2] = transiente(nos_elementos,CC,Mglobs,Kglobs)

%Função que realiza a análise transiente
global Elementos alfa beta gama beta2 L1 L2 L3 E

nos_F1=find(nos_elementos(:,2)==14);
nos_F1=nos_F1(1:Elementos:end);
nos_F2=find(nos_elementos(:,1)==1);
nos_F2=nos_F2(1:Elementos:end);

ti=0;
dt=0.02;
tf=10;

MCC=Mglobs;
KCC=Kglobs;


KCC(CC(2)-1,:)=0;
KCC(:,CC(2)-1)=0;
KCC(CC(2)-1,CC(2)-1)=1;

```

```
KCC(CC(1)-2:CC(1)-1,:)=0;
KCC(:,CC(1)-2:CC(1)-1)=0;
KCC(CC(1)-1,CC(1)-1)=1;
KCC(CC(1)-2,CC(1)-2)=1;

MCC(CC(2)-1,:)=0;
MCC(:,CC(2)-1)=0;
MCC(CC(2)-1,CC(2)-1)=1;

MCC(CC(1)-2:CC(1)-1,:)=0;
MCC(:,CC(1)-2:CC(1)-1)=0;
MCC(CC(1)-1,CC(1)-1)=1;
MCC(CC(1)-2,CC(1)-2)=1;

Cglobs=alfa*MCC+beta*KCC;

Forcas=zeros(length(Mglobs),length(ti:dt:tf));
desloc=zeros(length(Mglobs),length(ti:dt:tf));
vel=zeros(length(Mglobs),length(ti:dt:tf));
acel=zeros(length(Mglobs),length(ti:dt:tf));

modo=1; %transiente
Forcas(:,1)=geraF(nos_F1,nos_F2,0,length(Mglobs),modo);

acel(:,1)=inv(MCC)*(Forcas(:,1)-Cglobs*vel(:,1)-KCC*desloc(:,1));
Mbarra=MCC+dt*gama*Cglobs+dt^2*beta2*KCC;

for t=(ti+dt):dt:tf %A cada iteração calcula a{n+1=t} e F{n+1=t}
    Forcas(:,round(t/dt)+1)=geraF(nos_F1,nos_F2,t,length(MCC),modo);
    Mbarra=MCC+dt*gama*Cglobs+dt^2*beta2*KCC;

    Fbarra=Forcas(:,round(t/dt)+1)-Cglobs*(vel(:,round(t/dt))+...
    dt*(1-gama)*acel(:,round(t/dt)))-KCC*(desloc(:,round(t/dt))+...
    dt*vel(:,round(t/dt))+dt^2/2*(1-2*beta2)*acel(:,round(t/dt)));

    acel(:,round(t/dt)+1)=Mbarra\Fbarra;
    vel(:,round(t/dt)+1)=vel(:,round(t/dt))+...
    dt*((1-gama)*acel(:,round(t/dt))+gama*acel(:,round(t/dt)+1));
    desloc(:,round(t/dt)+1)=desloc(:,round(t/dt))+dt*vel(:,round(t/dt))...
    +dt^2/2*((1-2*beta2)*acel(:,round(t/dt))+2*beta2*acel(:,round(t/dt)+1));

    desloc(CC(2)-1,round(t/dt)+1)=0;
    desloc(CC(1)-2:CC(1)-1,round(t/dt)+1)=0;
    vel(CC(2)-1,round(t/dt)+1)=0;
    vel(CC(1)-2:CC(1)-1,round(t/dt)+1)=0;
    acel(CC(2)-1,round(t/dt)+1)=0;
    acel(CC(1)-2:CC(1)-1,round(t/dt)+1)=0;
end

noA=find(nos_elementos(:,1)==1 & nos_elementos(:,2)==14);
noB=find(nos_elementos(:,1)==1.75 & nos_elementos(:,2)==4.5);
noD=find(nos_elementos(:,1)==0.25 & nos_elementos(:,2)==1.5);
noE=find(nos_elementos(:,1)==3 & nos_elementos(:,2)==7);

%Plota os deslocamentos
figure()
plot(ti:dt:tf,desloc(3*noD-2,:)) %deslocamento em x
hold on
plot(ti:dt:tf,desloc(3*noE-2,:)) %deslocamento em x
legend('Node D','Node E')
xlabel('Time (s)')
ylabel('Displacement (m)')
title('Nodes displacement in the X axis')
saveas(gcf,'transemx.jpg')

```

```
figure()
plot(ti:dt:tf,desloc(3*noD-1,:)) %deslocamento em y
hold on
plot(ti:dt:tf,desloc(3*noE-1,:)) %deslocamento em y
legend('Node D','Node E')
xlabel('Time (s)')
ylabel('Displacement (m)')
title('Nodes displacement in the Y axis')
saveas(gcf,'transemy.jpg')

figure()
plot(ti:dt:tf,sqrt(desloc(3*noD-1,:).^2+desloc(3*noD-2,:).^2))
hold on
plot(ti:dt:tf,sqrt(desloc(3*noE-1,:).^2+desloc(3*noE-2,:).^2))
legend('Node D','Node E')
xlabel('Time (s)')
ylabel('Absolute Displacement (m)')
title('Nodes absolute displacement value')
saveas(gcf,'transemabs.jpg')

%Plota as tensões
%Tensão versão 1
dxA=E*(desloc(3*noA-2,:)-desloc(3*find(nos_elementos(:,1)==1+L1/...
    Elementos & nos_elementos(:,2)==14)-2,:))/(L1/Elementos);
tensaoA=abs(dxA);

dxB=E*(desloc(3*noB-2,:)-desloc(3*find(nos_elementos(:,1)==1.75+2.5/...
    Elementos & ...
        nos_elementos(:,2)==4.5+L2/Elementos)-2,:))/(sqrt(L2^2+(L3-1.5)^2)/Elementos);
dyB=E*(desloc(3*noB-1,:)-desloc(3*find(nos_elementos(:,1)==1.75-2.5/Elementos &...
    nos_elementos(:,2)==4.5-L2/Elementos)-1,:))/(sqrt(L2^2+(L3-1.5)^2)/Elementos);

tensaoB=sqrt(dxB.^2+dyB.^2);

figure()
plot(ti:dt:tf,tensaoA)
hold on
plot(ti:dt:tf,tensaoB)
xlabel('Time (s)')
ylabel('Stress (Pa)')
title('Stress Curve along the time')
legend('Node A','Node B')

end
```

```
function Forca= geraF(nos_F1,nos_F2,t,tamanho,modo)
    %Função para gerar o vetor das forças pra análise transiente e
    %harm nica
    global t1 t2 F D

    Forca=zeros(tamanho,1);
    if modo==1
        Forca(3*nos_F1-1)=2*F*sin(2*pi*t);
        if t>=t1 && t<=t2
            Forca(3*nos_F2-2)=5*D;
        end
    else
        Forca(3*nos_F1-1)=2*F;
    end
end
```

B.5 Harmonic Analysis Script

```
function harmonica(nos_elementos,MCC,KCC,nos_F1,nos_F2)

%Função que realiza a análise harm nica

modo=2;
Forcas=geraF(nos_F1,nos_F2,0,length(MCC),modo);

w=1:0.1:35;

noB=find(nos_elementos(:,1)==1.75 & nos_elementos(:,2)==4.5);
noC=find(nos_elementos(:,1)==1.75 & nos_elementos(:,2)==1.5);
vetB=zeros(length(w),2);
vetC=zeros(length(w),2);

for i=1:length(w)
    freq=(2*pi*w(i))^2;
    Mbarra=KCC-freq*MCC;
    solucao=Mbarra\Forcas;
    vetB(i,:)=[solucao(3*noB-2) solucao(3*noB-1)];
    vetC(i,:)=[solucao(3*noC-2) solucao(3*noC-1)];
end

modB=sqrt(vetB(:,1).^2+vetB(:,2).^2);
modC=sqrt(vetC(:,1).^2+vetC(:,2).^2);

figure
plot(w,modB)
hold on
plot(w,modC)
legend('Node B','Node C')
xlabel('Frequency (Hz)')
ylabel('Absolute Displacement (m)')
title('Frequency influence in Nodal Absolute Displacement')
saveas(gcf,'harmonicod.jpg')

end
```