POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

FEM ASSIGNMENT

In-plane Finite Element Model of a bike frame

DMS - Mechanical Engineering, Politecnico di Milano

Authors:

Paolo Trenta, Student ID: 10718668

Matteo Trotta, Student ID: 10337755

Lorenzo Vignoli, Student ID: 10712405

Professor: Alberto Zasso

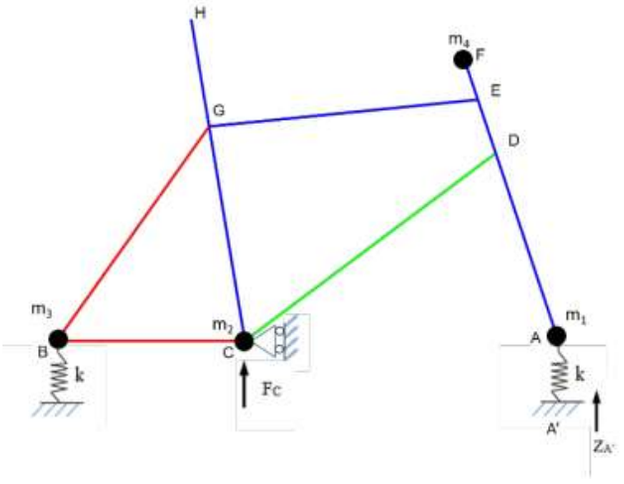
Co-professor: Giulia Pomaranzi

Academic Year: 2023-24

Contents

1. Introduction

1.1. Bike schematization and mechanical properties



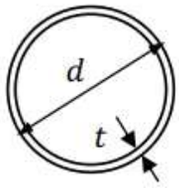


Figure 1: bike schematization

This assignment analyses a road bike frame through the Finite Element Method, providing the system's response to different types of excitations.

Lumped masses	
m_1	1 kg
m_2	2 kg
m_3	2 kg
m_4	1.5 kg
Tyre stiffness (for 1 tyre)	
k	$4 \cdot 10^5$ N/m

beam section	*d* [*mm*]	*t* [*mm*]
Red	20	1
Blue	32	1.5
Green	50	2.5

Point	x [*m*]	y [*m*]		Point	x [*m*]	y [*m*]
A	0.70	0		E	0.52	0.54
B	-0.42	0		F	0.49	0.63
C	0	0		G	-0.08	0.48
D	0.56	0.42		H	-0.12	0.72

The frame's material is aluminum ($E = 7 \cdot 10^{10}$ N/m^2, $\varrho = 2700$ kg/m^3) and damping is defined according to the "proportional damping" assumption: $[C] = \alpha \cdot [M] + \beta \cdot [K]$, with $\alpha = 6$ s^{-1} and $\beta = 1 \cdot 10^{-5}$ s.

1.2. Finite Element Model definition

After having defined the structure and its properties, the bike frame is divided into finite elements by using as nodes the points from A to H. However, each element must work in the quasi-static region: the length of any beam FE shall not exceed a certain limit value, which depends on the maximum Ω_{max} of the analysis.

This means that, for each k FE:

$$\omega_k^{(1)} > \eta\Omega_{max}$$

η is a safety coefficient (in our case equal to 2). The element first natural pulsation is:

$$\omega_k^{(1)} = (\frac{\pi}{L_k})^2 \cdot \sqrt{\frac{EJ_k}{m_k}}$$

Ω_{max} is given, being $\Omega_{max} = 2\pi \cdot f_{max}$, with $f_{max} = 200\ Hz$.

From the computation of each element's first natural pulsation, it's deduced that the condition is not satisfied for elements 2-3 (BC), 2-7 (BG), 3-7 (CG), 3-4 (CD) and 5-7 (GE). The simplest solution is to put another node in the middle of these: this way the condition is always verified, although for the element 1-4 (AD) it's borderline.
The obtained solution will be:

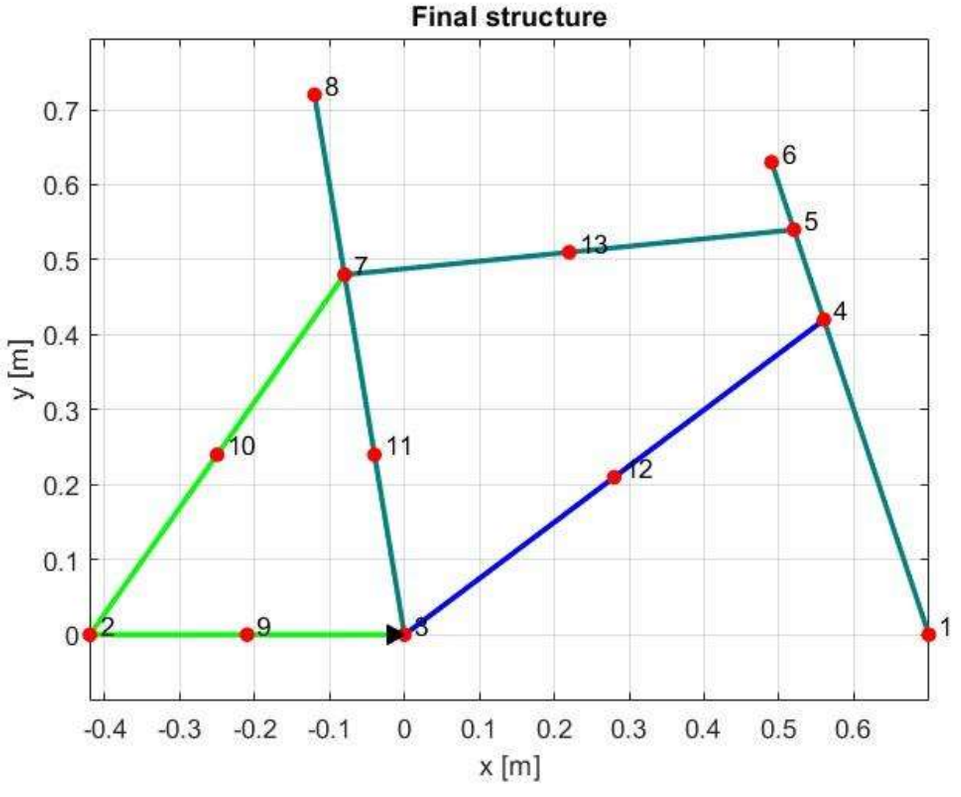


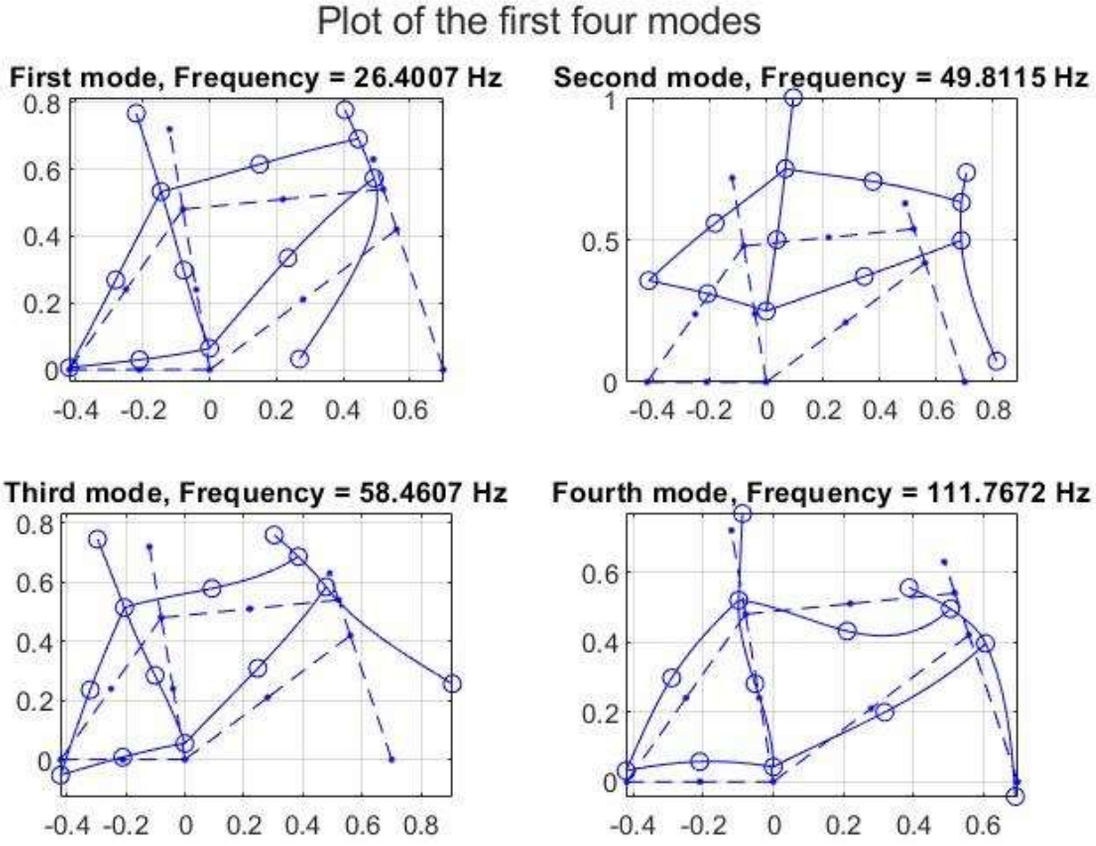
Figure 2: nodes' numeration

2. Natural frequencies and vibration modes

To compute the structure's natural frequencies and vibration modes (up to the 4th one) the mass matrix [M] and the stiffness one [K] of the system are required.
Through the MatLab function *assem* they are computed only referring to its distributed properties: next passage is to include the lumped masses and stiffnesses too, which are then added one by one to the bike frame. It's important to underline that the concentrated masses have no moment of inertia, resulting in no effect on the rotational degree of freedom, while springs are only effective in the vertical direction.

Node C is constrained by a cart which prevents horizontal displacement, as assessed by the input file, so this nodal coordinate will be null and so are its derivatives (constrained coordinate).
The mass matrix can be thus partitioned into [M_{FF}], [M_{FC}], [M_{CF}] and [M_{CC}] (and the same for [K]), which are the submatrices referred to the free and constrained dynamics of the system. The free motion equation will be:

$$\begin{bmatrix}[M_{FF}] & [M_{FC}]\\ [M_{CF}] & [M_{CC}]\end{bmatrix}\begin{bmatrix}\ddot{\underline{x}}_F\\ \ddot{x}_C\end{bmatrix}+\begin{bmatrix}[K_{FF}] & [K_{FC}]\\ [K_{CF}] & [K_{CC}]\end{bmatrix}\begin{bmatrix}\underline{x}_F\\ x_C\end{bmatrix}=\underline{0}$$



Only free coordinates must be considered for the modal computation, obtaining:

$$[M_{FF}]\ddot{\underline{x}}_F + [K_{FF}]\underline{x}_F = \underline{0}$$

Being $\underline{x}_F = \underline{x}_{0F} \cdot e^{\pm i\omega_0 t}$, immediately follows:

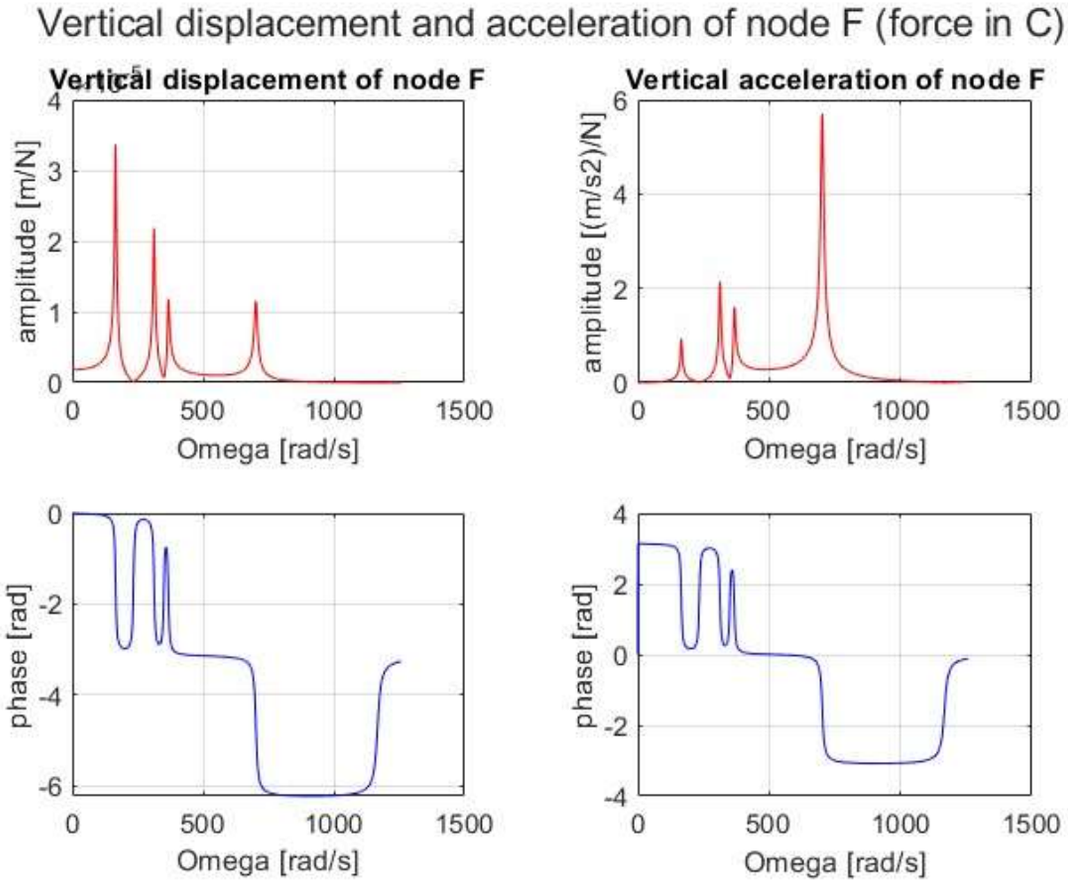
$$(-\omega_0^2[M_{FF}] + [K_{FF}]) \cdot \underline{x}_{0F} = \underline{0}$$

Natural frequencies are then computed as square roots of the eigenvalues of the matrix $[M_{FF}]^{-1}[K_{FF}]$, while vibration modes are the corresponding eigenvectors.

Note that the eigenvectors are defined up to a constant value, so the actual amplitude of the modes is determined only through a "scale factor" (equal to 1.5 in our case). The *diseg2* function can be used to plot the first four vibration modes.

3. FRF due to an external force in C

3.1. Displacement and acceleration of node F and H



The Rayleigh hypothesis can be applied here to determine the distributed damping matrix [C]:

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

The same partition which was applied for [M] and [K] can be used here too, obtaining a forced (and damped) motion equation:

$$\begin{bmatrix}[M_{FF}] & [M_{FC}]\\ [M_{CF}] & [M_{CC}]\end{bmatrix}\begin{bmatrix}\underline{\ddot{x}_F}\\ \ddot{x}_C\end{bmatrix}+\begin{bmatrix}[C_{FF}] & [C_{FC}]\\ [C_{CF}] & [C_{CC}]\end{bmatrix}\begin{bmatrix}\underline{\dot{x}_F}\\ \dot{x}_C\end{bmatrix}+\begin{bmatrix}[K_{FF}] & [K_{FC}]\\ [K_{CF}] & [K_{CC}]\end{bmatrix}\begin{bmatrix}\underline{x_F}\\ x_C\end{bmatrix}=\begin{bmatrix}\underline{F_0}\\ R_{C0}\end{bmatrix}\cdot e^{i\Omega t}$$

Only a vertical external force in C is considered, so all the elements of $\underline{F_0}$ will be null except the one corresponding to the vertical degree of freedom of C. Being:

$$\underline{x_F}=\underline{x_0}e^{i\Omega t}$$

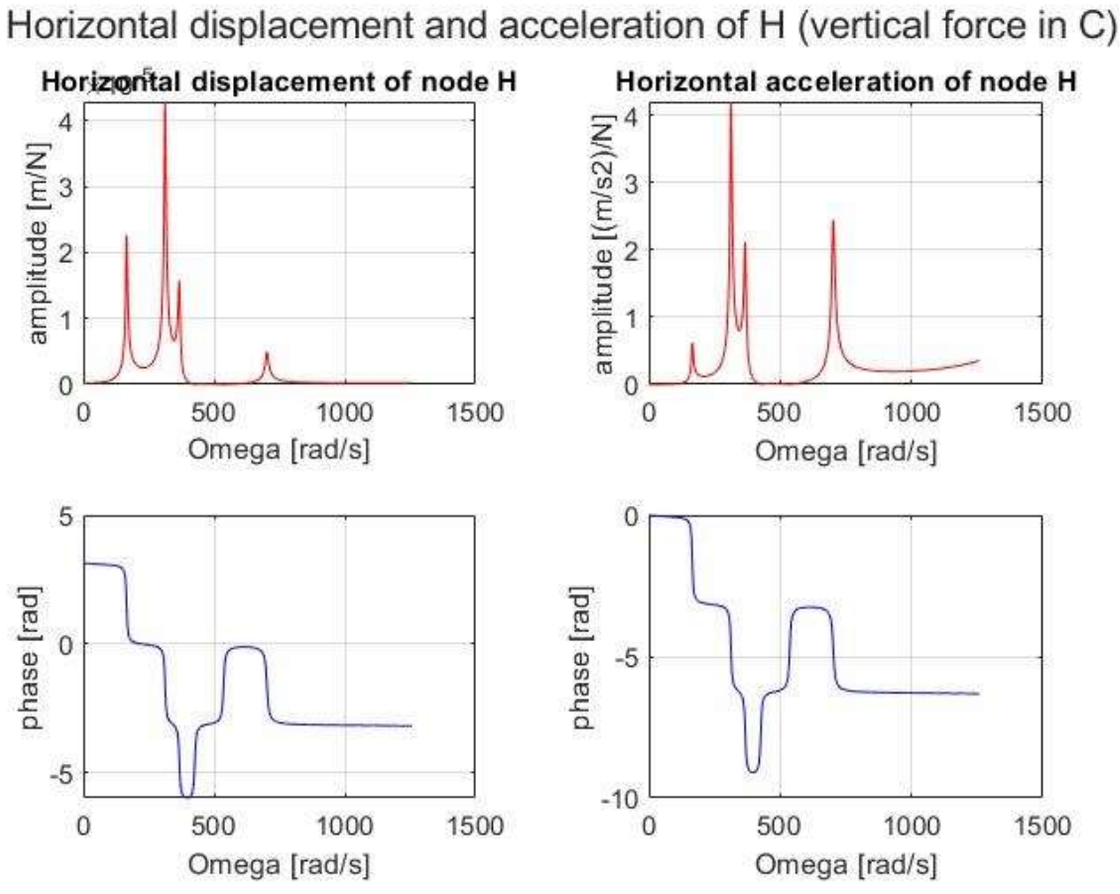
And:

$$\underline{\ddot{x}_F}=-\Omega^2\underline{x_0}e^{i\Omega t}$$

The (free coordinates) equation will be:

$$(-\Omega^2[M_{FF}]+i\Omega[C_{FF}]+[K_{FF}])\cdot\underline{x_0}=\underline{F_0}$$

The vector $\underline{x_0}$ is then given by the solution to that system, and the acceleration can be easily computed multiplying by $-\Omega^2$.



3.2. Internal actions in the midpoint of the GE tube

Using the FE formulation, the axial and transversal displacements can be written as:

$$u(\xi,t)=a_0+b_0\xi$$

$$\omega(\xi,t)=a+b\xi+c\xi^2+d\xi^3$$

In these equations, ξ is the internal free running coordinate, ω and u respectively the transversal and axial displacement.
According to the Euler-Bernoulli theory, the internal actions can be written as:

$$N = EA\frac{\partial u}{\partial \xi} = EAb_0$$

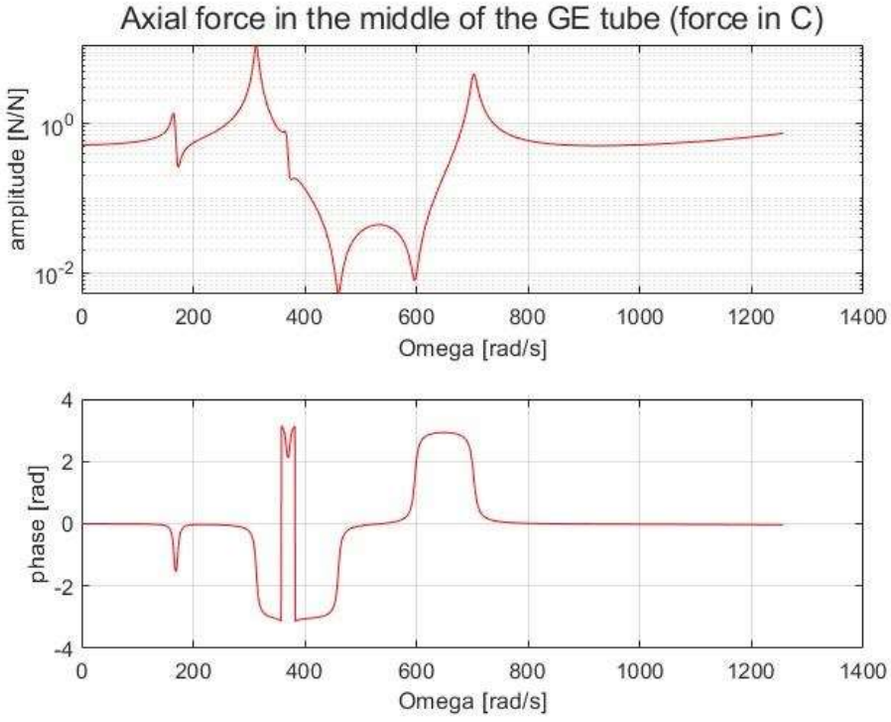
$$M = EJ\frac{\partial^2 \omega}{\partial \xi^2} = EJ(2c + 6d\xi)$$

$$\mathrm{T} = EJ\frac{\partial^3 \omega}{\partial \xi^3} = EJ6d$$

The coefficients of the shape functions can be computed knowing the displacement of the right "R", and left "L" nodes of the beam, through the application of the boundary conditions:

$$u(\xi, t) = a_0 + b_0\xi \implies \begin{cases} a_0 = x_L \\ b_0 = \dfrac{x_R - x_L}{L_k} \end{cases}$$

$$\omega(\xi, t) = a + b\xi + c\xi^2 + d\xi^3 \implies \begin{cases} a = y_L \\ b = \vartheta_L \\ c = -\dfrac{3}{{L_k}^2}y_L + \dfrac{3}{{L_k}^2}y_R - \dfrac{2}{L_k}\vartheta_L - \dfrac{1}{L_k}\vartheta_R \\ d = \dfrac{2}{{L_k}^3}y_L - \dfrac{2}{{L_k}^3}y_R + \dfrac{1}{{L_k}^2}\vartheta_L + \dfrac{1}{{L_k}^2}\vartheta_R \end{cases}$$

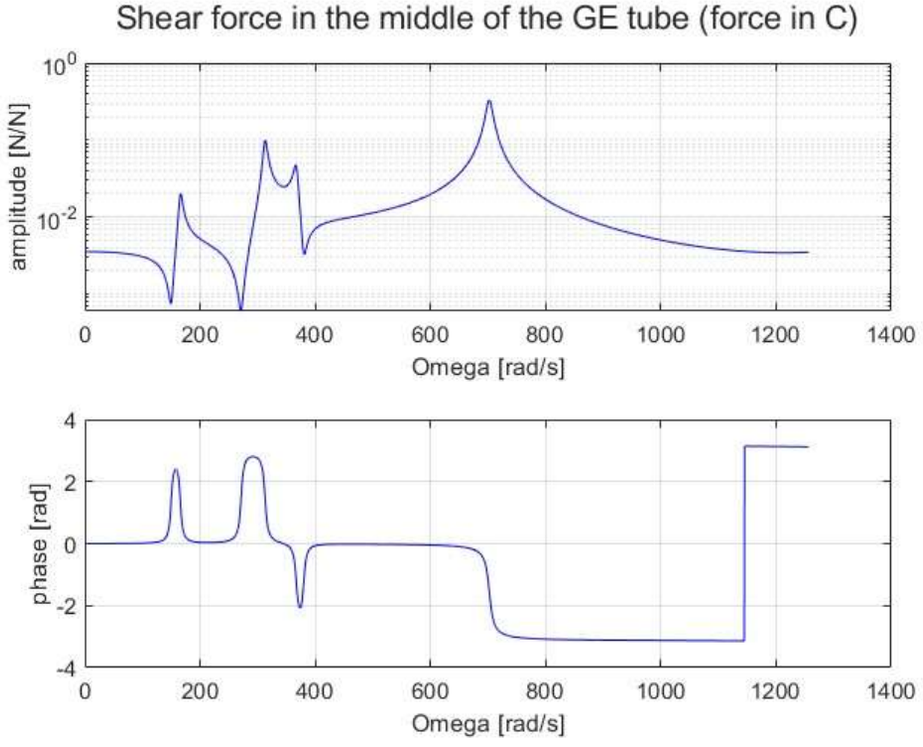


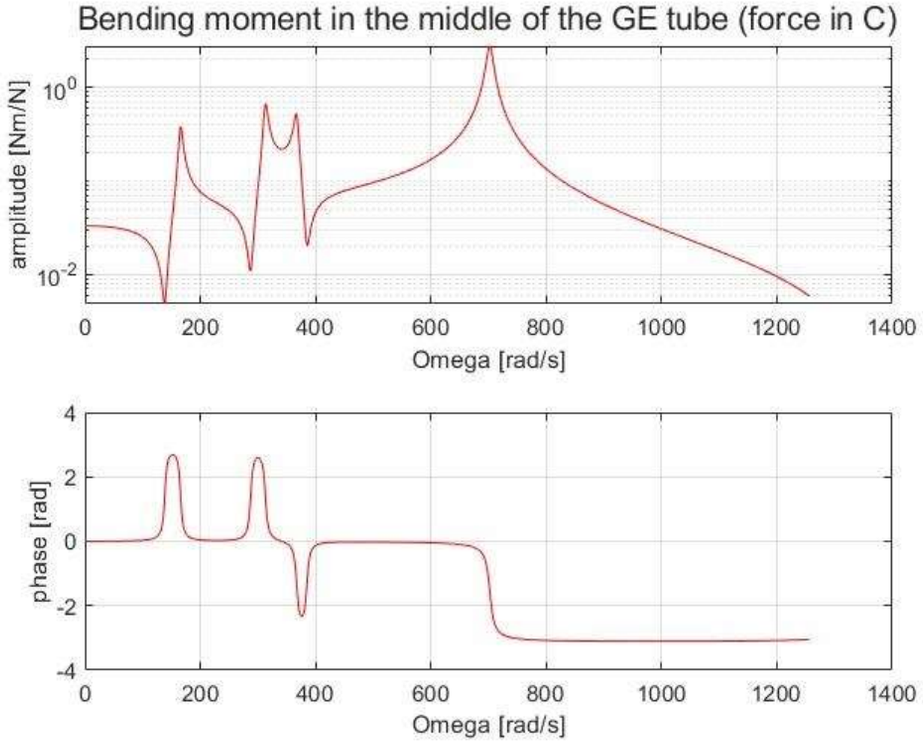
In this specific case the midpoint of the tube GE happens to be where the 13th node is placed. For the computations, the 7th and the 13th nodes are respectively the left and right ones. To correctly calculate the internal action, the displacement of the two nodes,

found in the previous step in global coordinates must be converted in local coordinates with the following rotation matrix:

$$\lambda = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

With the angle γ between the FE beam itself and the global reference system.



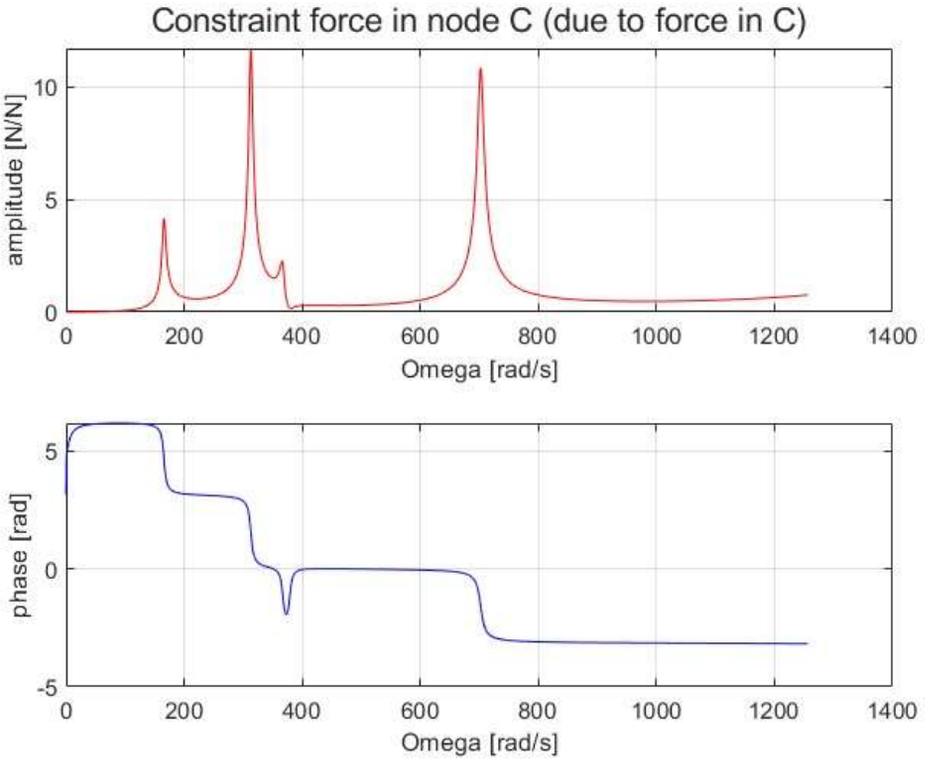


3.3. Constraint force in C

Knowing the global total displacements of the system the (only) constraint force can be written as:

$$R_{C0} = (-\Omega^2[M_{CF}] + i\Omega\,[C_{CF}] + [K_{CF}]) \cdot \underline{x}_0$$

Please note that in this specific case $[M_{CF}]$, $[C_{CF}]$ and $[K_{CF}]$ are row vectors, resulting in R_{CO} being a scalar.



4. Modal superposition approach

The modal approach is used to evaluate the FRFs of displacement and acceleration of handlebar (node F) and saddle (node H) by considering the first two modes only.

The modal matrix [Φ] is obtained by juxtaposing the first two eigenvectors (so [Φ] will be a 38x2, being 13 the total number of nodes and one degree of freedom constrained).

From the global total free coordinates' equation of motion:

$$[\mathrm{M_{FF}}]\ddot{\underline{x}}_F + [\mathrm{C_{FF}}]\dot{\underline{x}}_F + [\mathrm{K_{FF}}]\underline{x}_F = \underline{F}_0 e^{i\Omega t}$$

Applying the modal approach:

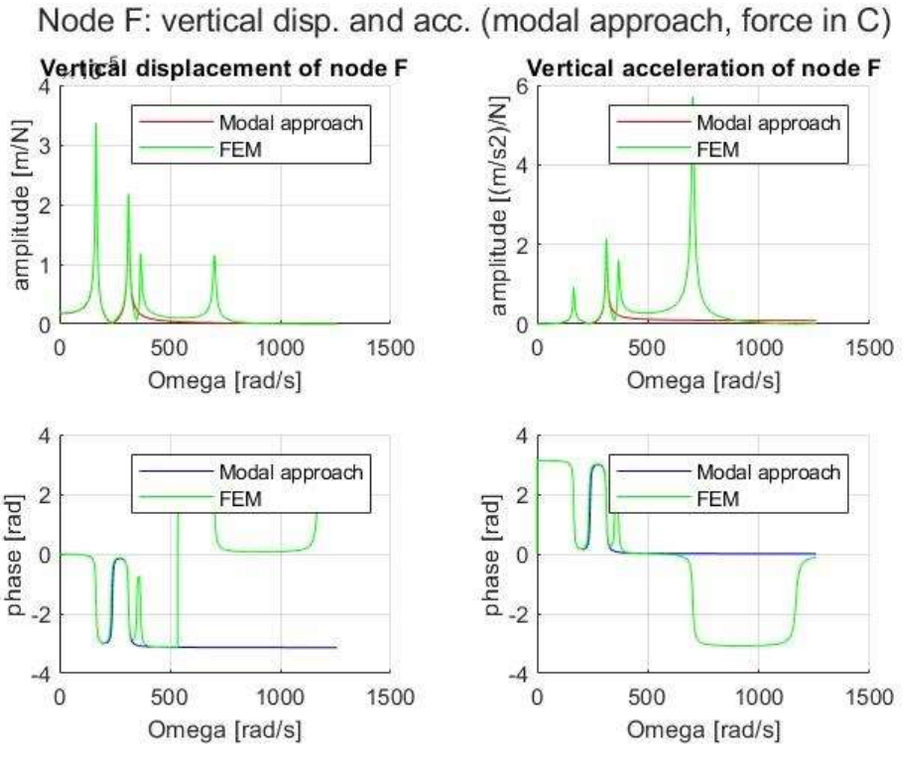
$$\underline{x}_{FF} = [\Phi]\underline{q}(t)$$

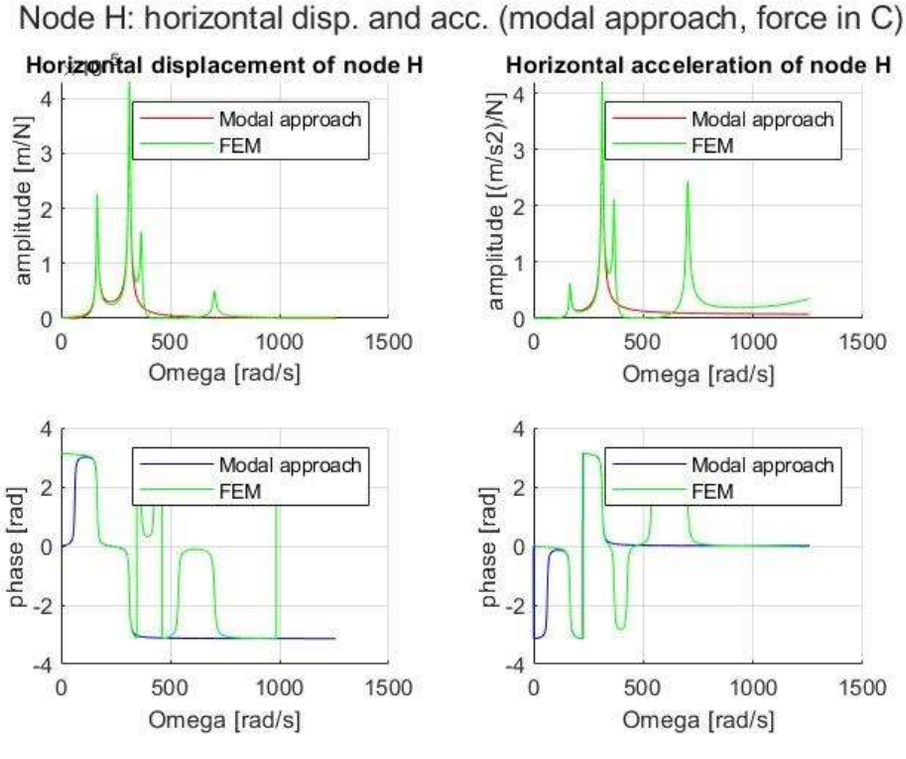
$$[m^*] = [\Phi]^T[M][\Phi]\,, \quad [c^*] = [\Phi]^T[C][\Phi]\,, \quad [k^*] = [\Phi]^T[K][\Phi]$$

$$\underline{Q}_0 = [\Phi]^T\underline{F}_0$$

Thus:

$$[m^*]\ddot{\underline{q}} + [c^*]\dot{\underline{q}} + [k^*]\underline{q} = \underline{Q}_0 e^{i\Omega t}, \text{ with } \underline{q} = \underline{q}_0 e^{i\Omega t}$$





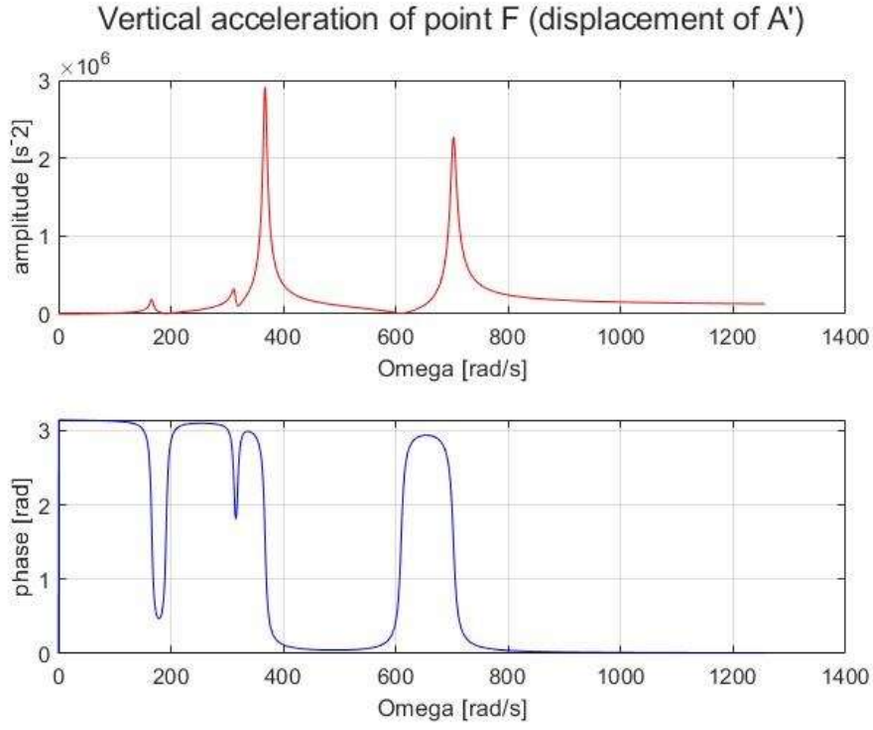
Concerning the outputs related both to point H and F, the approximation works well up to the second resonance peak, since these are the vibration modes we are working with. The quasistatic region is well represented too.

5. Acceleration of F due to the displacement of A’

If an input vertical displacement at point A’ is applied (likely due to the road interacting with the front wheel), computing the FRF is equivalent to evaluating the effects of a unitary compression on the vertical spring in A.
The spring will produce a reaction force directed outwards to respond to the state of compression; being the displacement equal to one, this force will be:

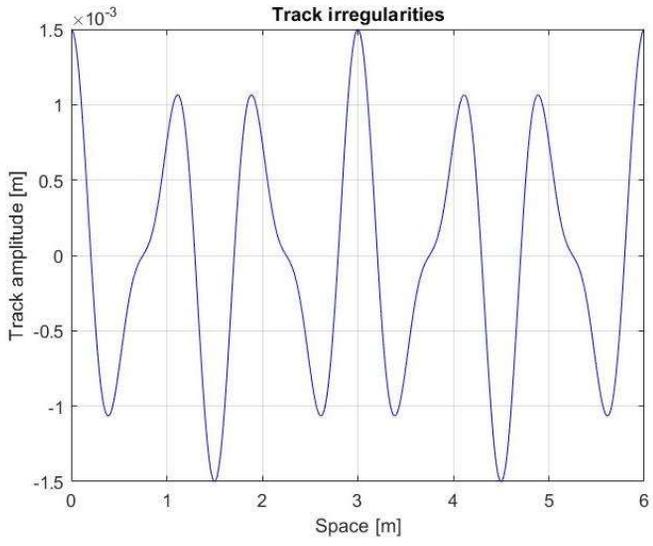
$$F_{k,vertical} = k \cdot 1$$

Following the same steps of paragraph 3.1, the output vertical acceleration at point F is then computed.



6. Time history of the steady-state vertical acceleration of point H

The saddle (point H) vertical acceleration can be computed applying the same reasoning shown in chapter 5. The bike is considered to travel at a speed of 12 m/s on a sinusoidal road irregularity described mathematically by the superposition of two harmonic functions; at time t = 0 the center of the front wheel (point A) is located above a positive maximum of the irregularity profile.



	1ST HARMONIC	2ND HARMONIC
$\boldsymbol{\lambda}$	1 m	0.6 m
$\boldsymbol{A}$	1 m	0.5 m
$\boldsymbol{\varphi}$	0 rad	0 rad

Figure 3: road profile in A'

It must be pointed out that the bike is not considered to be moving on the road, but the road itself is moving under the bike (being a relative motion, it's the same).

Evaluating the two harmonics, each frequency can be computed as (i = 1, 2):

$$f_i = \frac{v}{\lambda_i}$$

By hypothesis, the front wheel has a null phase for both harmonics ($\varphi_{1,A'} = \varphi_{2,A'} = 0$), while for the rear wheel (introduced thanks to the point B') it must be known that the two wheels' centers outdistance:

$$d_{AB} = x_A - x_B = 1.12\ m$$

Then, phase delays will be (i = 1, 2):

$$\varphi_{i,B'} = -\frac{2\pi}{\lambda_i} \cdot d_{AB}$$

At point A', then, the track profile will be:

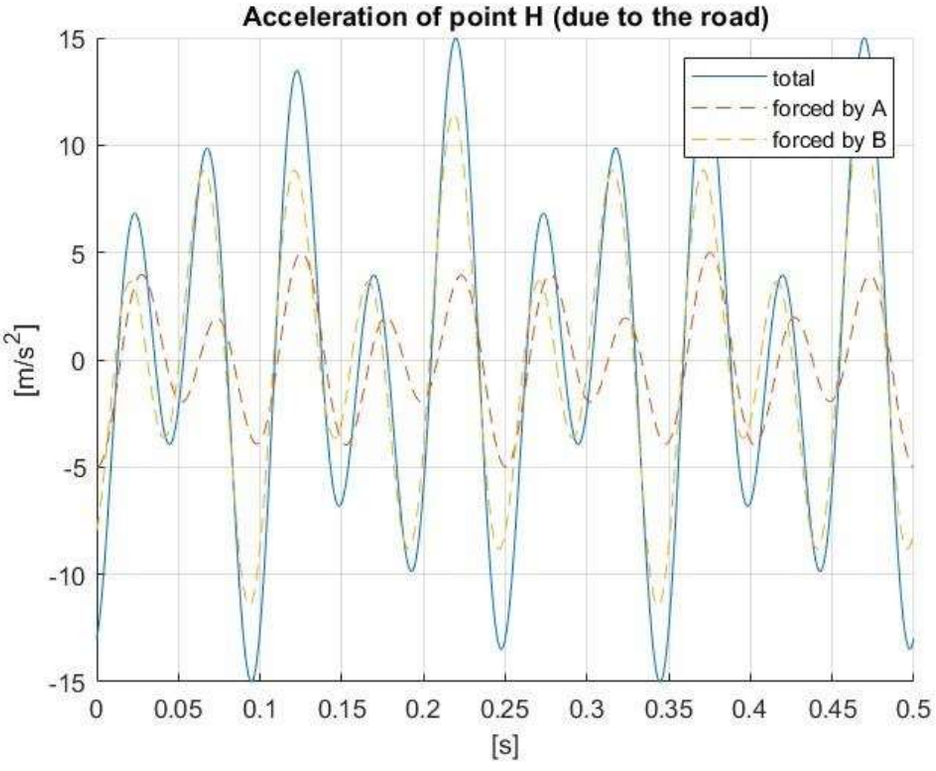
$$z_{A'} = \sum_{i=1,2} A_i \cdot \cos\left(2\pi f_i \cdot t\right)$$

At point B' instead:

$$z_{B'} = \sum_{i=1,2} A_i \cdot \cos\left(2\pi f_i \cdot t - \varphi_{i,B'}\right)$$

Two equivalent elastic forces are obtained in A and B, leading to two different effects that are then superimposed. The calculations are performed computing directly the accelerations (as in chapter 5), remembering to add to the phase of the FRF the phase of the forcing in B'.

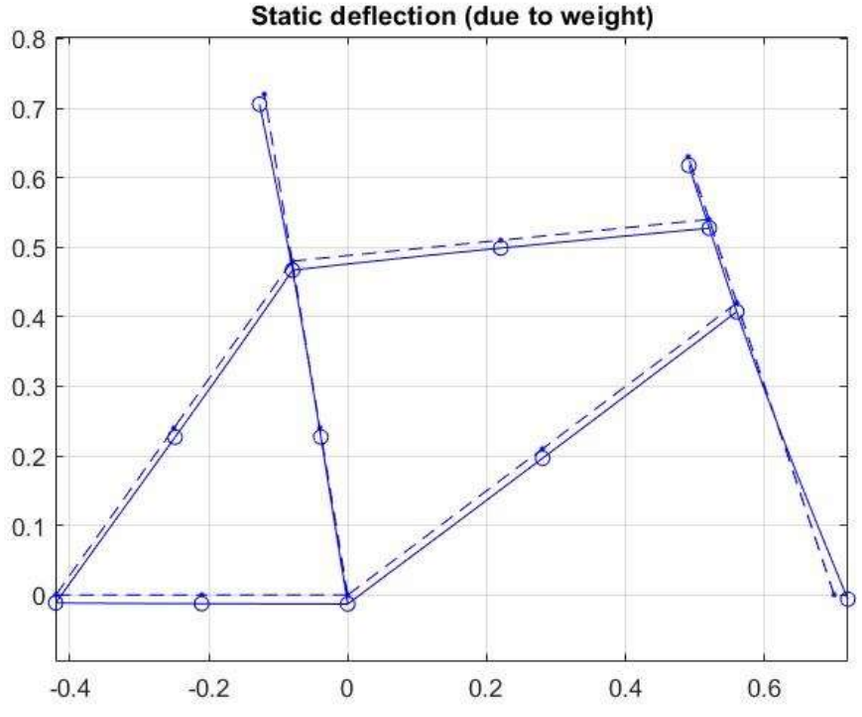
As a check, a similar procedure has also been applied but starting from displacements instead (with identical results).



7. Static deflection

The weight of the cyclist is represented by a 600 N and 100 N vertical forces applied respectively in the saddle (H) and in the handlebar (F), obviously directed downwards. Since all the derivatives (velocity and acceleration) are null:

$$[K_{FF}]\underline{x}_F = \underline{F}_g$$



The vertical displacement of the midpoint of the GE tube (node 13) is directed downwards and equal to (-) 1.1213 mm.

8. MATLAB script

```
clear all;
close all;
clc;
%% 1. Define a FEM model of the structure between 0-200 HZ...
fmax = 200;
omega_max = fmax*2*pi;
eta = 2;
El = 7e10;
rho = 2700;
alpha = 6;
beta = 1e-5;
% we define now all points as vectors, according to the input code, and we
% arrange them in the same matrix
A = [0.70; 0];
B = [-0.42; 0];
C = [0; 0];
D = [0.56; 0.42];
E = [0.52; 0.54];
F = [0.49; 0.63];
G = [-0.08; 0.48];
H = [-0.12; 0.72];
nodes = [A, B, C, D, E, F, G, H];
% Definition of the lengths matrix
L(1) = sqrt((nodes(1,2)-nodes(1,3))^2+(nodes(2,2)-nodes(2,3))^2);
L(2) = sqrt((nodes(1,2)-nodes(1,7))^2+(nodes(2,2)-nodes(2,7))^2);
L(3) = sqrt((nodes(1,3)-nodes(1,7))^2+(nodes(2,3)-nodes(2,7))^2);
L(4) = sqrt((nodes(1,3)-nodes(1,4))^2+(nodes(2,3)-nodes(2,4))^2);
L(5) = sqrt((nodes(1,7)-nodes(1,8))^2+(nodes(2,7)-nodes(2,8))^2);
L(6) = sqrt((nodes(1,5)-nodes(1,7))^2+(nodes(2,5)-nodes(2,7))^2);
L(7) = sqrt((nodes(1,5)-nodes(1,6))^2+(nodes(2,5)-nodes(2,6))^2);
L(8) = sqrt((nodes(1,4)-nodes(1,5))^2+(nodes(2,4)-nodes(2,5))^2);
```

```
L(9) = sqrt((nodes(1,1)-nodes(1,4))^2+(nodes(2,1)-nodes(2,4))^2);
t_red = 1e-3;
t_blue = 1.5e-3;
t_green = 2.5e-3;
d_red = 20e-3;
d_blue = 32e-3;
d_green = 50e-3;
t(1)=t_red;
t(2)=t_red;
t(3)=t_blue;
t(4)=t_green;
t(5)=t_blue;
t(6)=t_blue;
t(7)=t_blue;
t(8)=t_blue;
t(9)=t_blue;
d(1)=d_red;
d(2)=d_red;
d(3)=d_blue;
d(4)=d_green;
d(5)=d_blue;
d(6)=d_blue;
d(7)=d_blue;
d(8)=d_blue;
d(9)=d_blue;
for ii = 1:9
   d_int = d(ii) - 2*t(ii);
   EJ(ii) = El*pi/64*(d(ii)^4-d_int^4);
   section(ii) = pi/4*(d(ii)^2-d_int^2);
   m(ii) = rho*section(ii);
end
% We have computed here EJ, section and distributed mass for each beam. The
% system is now provided also with diameter, length and thickness of each
% beam (and the coordinates too)
for ii = 1:9
    omega_1(ii) = (pi/L(ii))^2*sqrt(EJ(ii)/m(ii)); % first natural pulsation
end
if ~ all(omega_1 > 2*omega_max)
    fprintf ("The length of some FE exceeds the limit value\n");
```

```
end
to_change = omega_1 < 2*omega_max;
% Thanks to that we know that the elements 1, 2, 3, 4, 6 are to change. We
% could put another node in the middle points of each one, keeping the same
% names to the previous nodes and to the unchanged elements
node_9 = [(nodes(1,2)+nodes(1,3))/2; (nodes(2,2)+nodes(2,3))/2]; % n9, e(1), 10
nodes = [nodes, node_9];
L(1) = sqrt((nodes(1,2)-nodes(1,9))^2+(nodes(2,2)-nodes(2,9))^2);
L(10) = sqrt((nodes(1,9)-nodes(1,3))^2+(nodes(2,9)-nodes(2,3))^2);
EJ(10) = EJ(1);
section(10) = section(1);
m(10) = m(1);
node_10 = [(nodes(1,2)+nodes(1,7))/2; (nodes(2,2)+nodes(2,7))/2]; %n10, e(2), 11
nodes = [nodes, node_10];
L(2) = sqrt((nodes(1,2)-nodes(1,10))^2+(nodes(2,2)-nodes(2,10))^2);
L(11) = sqrt((nodes(1,7)-nodes(1,10))^2+(nodes(2,7)-nodes(2,10))^2);
EJ(11) = EJ(2);
section(11) = section(2);
m(11) = m(2);
node_11 = [(nodes(1,3)+nodes(1,7))/2; (nodes(2,3)+nodes(2,7))/2]; %n11, e(3), 12
nodes = [nodes, node_11];
L(3) = sqrt((nodes(1,3)-nodes(1,11))^2+(nodes(2,3)-nodes(2,11))^2);
L(12) = sqrt((nodes(1,7)-nodes(1,11))^2+(nodes(2,7)-nodes(2,11))^2);
EJ(12) = EJ(3);
section(12) = section(3);
m(12) = m(3);
node_12 = [(nodes(1,3)+nodes(1,4))/2; (nodes(2,3)+nodes(2,4))/2]; %n12, e(4), 13
nodes = [nodes, node_12];
L(4) = sqrt((nodes(1,3)-nodes(1,12))^2+(nodes(2,3)-nodes(2,12))^2);
L(13) = sqrt((nodes(1,4)-nodes(1,12))^2+(nodes(2,4)-nodes(2,12))^2);
EJ(13) = EJ(4);
section(13) = section(4);
m(13) = m(4);
node_13 = [(nodes(1,5)+nodes(1,7))/2; (nodes(2,5)+nodes(2,7))/2]; %n13, e(6), 14
nodes = [nodes, node_13];
L(6) = sqrt((nodes(1,7)-nodes(1,13))^2+(nodes(2,7)-nodes(2,13))^2);
L(14) = sqrt((nodes(1,5)-nodes(1,13))^2+(nodes(2,5)-nodes(2,13))^2);
EJ(14) = EJ(6);
section(14) = section(6);
```

```
m(14) = m(6);
% We can now perform the same check on the 14 nodes we got
for ii = 1:9
    omega_1(ii) = (pi/L(ii))^2*sqrt(EJ(ii)/m(ii)); % first natural pulsation
end
if all(omega_1 > 2*omega_max)
    fprintf ("The length of each FE does not exceeds the limit value NOW\n");
end
% Draw structure (right one, with the good nodes)
[file_i,xy,nnod,sizee,idb,ndof,incid,l,gamma,m,EA,EJ,T,posit,nbeam,pr]=loadstructu
re;
dis_stru(posit, l, gamma, xy, pr, idb, ndof);
hold on;
xlabel('x [m]'); ylabel('y [m]');
title('Final structure');
hold off;

%% 2. Compute the natural frequencies and vibration modes (up to 4)...
% Assemble mass and stiffness matrices
[M, K] = assem(incid, l, m, EA, EJ, T, gamma, idb);
% We need to assemble now the "new" mass and stiffness matrices
i_dof1 = idb(1,:);
i_dof2 = idb(2,:);
i_dof3 = idb(3,:);
i_dof6 = idb(6,:);
% The masses have no moment of inertia (point-shaped)
Mc1 = [1 0 0; 0 1 0; 0 0 0];
Mc2 = [2 0 0; 0 2 0; 0 0 0];
Mc3 = [2 0 0; 0 2 0; 0 0 0];
Mc6 = [1.5 0 0; 0 1.5 0; 0 0 0];
M(i_dof1, i_dof1) = M(i_dof1, i_dof1) + Mc1;
M(i_dof2, i_dof2) = M(i_dof2, i_dof2) + Mc2;
M(i_dof3, i_dof3) = M(i_dof3, i_dof3) + Mc3;
M(i_dof6, i_dof6) = M(i_dof6, i_dof6) + Mc6;
ky = 4e05;
K(i_dof1(2), i_dof1(2)) = K(i_dof1(2), i_dof1(2)) + ky;
K(i_dof2(2), i_dof2(2)) = K(i_dof2(2), i_dof2(2)) + ky;
% Partition of the matrices to get MFF and KFF
MFF = M(1:ndof, 1:ndof);
```

```
MCF = M(ndof+1:end, 1:ndof);
MFC = M(1:ndof, ndof+1:end);
MCC = M(ndof+1:end, ndof+1:end);
KFF = K(1:ndof, 1:ndof);
KCF = K(ndof+1:end, 1:ndof);
KFC = K(1:ndof, ndof+1:end);
KCC = K(ndof+1:end, ndof+1:end);
% computation of natural frequencies and mode shapes
[modes, omega2] = eig(MFF\KFF);
omega = sqrt(diag(omega2));
% sort frequencies in ascending order
[omega, i_omega] = sort(omega);
freq0 = omega/(2*pi);
%sort modes in ascending order (thanks to the previous function)
modes = modes(:, i_omega);
% first 4 frequencies
freq_until_4 = freq0(1:4);
scale_factor = 1.5;
mode1 = modes(:, 1);
mode2 = modes(:, 2);
mode3 = modes(:, 3);
mode4 = modes(:, 4);

% First four modes plot
figure(2);
t = tiledlayout(2,2);
title(t, 'Plot of the first four modes');
nexttile(1);
diseg2(mode1,scale_factor,incid,l,gamma,posit,idb,xy);
str = "First mode, Frequency = " + freq0(1) + " Hz";
title(str);
nexttile(2);
diseg2(mode2,scale_factor,incid,l,gamma,posit,idb,xy);
str = "Second mode, Frequency = " + freq0(2) + " Hz";
title(str);
nexttile(3);
diseg2(mode3,scale_factor,incid,l,gamma,posit,idb,xy);
str = "Third mode, Frequency = " + freq0(3) + " Hz";
title(str);
```

```
nexttile(4);
diseg2(mode4,scale_factor,incid,l,gamma,posit,idb,xy);
str = "Fourth mode, Frequency = " + freq0(4) + " Hz";
title(str);

%% 3. Frequency response functions...
% Applying the Rayleigh hypotesis, we build the damping matrix:
C = alpha*M + beta*K;
CFF = C(1:ndof,1:ndof);
CCF = C(ndof+1:end,1:ndof);
CFC = C(1:ndof,ndof+1:end);
CCC = C(ndof+1:end,ndof+1:end);

F0 = zeros(ndof, 1);
F0(idb(3,2)) = 1; % unitary force for the FRF
resolution_Hz = 0.1;
freq_max = 200;
om = (0:resolution_Hz:freq_max)*2*pi;
for ii = 1:length(om)
    A = (-om(ii)^2*MFF + 1i*om(ii)*CFF + KFF);
    X(:, ii) = A\F0;
    Xpp(:, ii) = -om(ii)^2*X(:,ii);
    rr(:, ii) = (-om(ii)^2*MCF+sqrt(-1)*om(ii)*CCF+KCF)*X(:,ii);
end

%% (3A) Vertical displacement and vertical acceleration of node F, horizontal
% displacement and horizontal acceleration of node H. Plot the
% corresponding magnitude and phase diagrams
figure(3);
t = tiledlayout(2,2);
title(t, 'Vertical displacement and acceleration of node F (force in C)');
nexttile(1);
plot(om, abs(X(idb(6,2), :)), 'r');
title('Vertical displacement of node F');
ylabel('amplitude [m/N]');
xlabel('Omega [rad/s]');
grid on;
nexttile(3);
plot(om, unwrap(angle(X(idb(6,2), :))), 'b');
```

```
ylabel('phase [rad]');
xlabel('Omega [rad/s]');
grid on;
nexttile(2);
plot(om, abs(Xpp(idb(6,2), :)), 'r');
title('Vertical acceleration of node F');
ylabel('amplitude [(m/s2)/N]');
xlabel('Omega [rad/s]');
grid on;
nexttile(4);
plot(om, unwrap(angle(Xpp(idb(6,2), :))), 'b');
ylabel('phase [rad]');
xlabel('Omega [rad/s]');
grid on;

figure(4);
t = tiledlayout(2,2);
title(t, 'Horizontal displacement and acceleration of H (vertical force in C)');
nexttile(1);
plot(om, abs(X(idb(8,1), :)), 'r');
title('Horizontal displacement of node H');
ylabel('amplitude [m/N]');
xlabel('Omega [rad/s]');
grid on;
nexttile(3);
plot(om, unwrap(angle(X(idb(8,1), :))), 'b');
ylabel('phase [rad]');
xlabel('Omega [rad/s]');
grid on;
nexttile(2);
plot(om, abs(Xpp(idb(8,1), :)), 'r');
title('Horizontal acceleration of node H');
ylabel('amplitude [(m/s2)/N]');
xlabel('Omega [rad/s]');
grid on;
nexttile(4);
plot(om, unwrap(angle(Xpp(idb(8,1), :))), 'b');
ylabel('phase [rad]');
xlabel('Omega [rad/s]');
```

```
grid on;

%% (3B) Shear force, bending moment and axial force evaluated in the midpoint
% of the GE tube. Plot the corresponding magnitude and phase diagrams.
n_el = 6; % n° of element
L_el = L(n_el);
csi = L_el;

% We are interested in the displacement in local coordinates; for this
% reason, we use the "lambda2 rotation matrix relative to the nel-element
lambda = [cos(gamma(n_el)) sin(gamma(n_el)) 0;
    -sin(gamma(n_el)) cos(gamma(n_el)) 0;
    0 0 1]; % "lambda" is an output of "loadstructure"
% FRF of nodal displacements:
Xi = lambda*X(idb(7,:),:);
Xj = lambda*X(idb(13,:),:);
% Computation of the polynomial-beam-coefficients:
b = (Xj(1,:) - Xi(1,:))/L_el;
c = -3/L_el^2*Xi(2,:)+3/L_el^2*Xj(2,:)-2/L_el*Xi(3,:)-1/L_el*Xj(3,:);
d = 2/L_el^3*Xi(2,:)-2/L_el^3*Xj(2,:)+1/L_el^2*Xi(3,:)+1/L_el^2*Xj(3,:);
% Computation of the internal actions (using the output of "loadstructure":
N = EA(n_el)*b;
T = EJ(n_el)*(2*c + 6*d*csi);
M = EJ(n_el)*6*d;
% Plotting of the magnitude and phase diagrams of internal actions
figure(5);
t = tiledlayout(2, 1);
title(t, 'Axial force in the middle of the GE tube (force in C)');
nexttile(1);
semilogy(om, abs(N), 'r');
grid on;
xlabel('Omega [rad/s]');
ylabel('amplitude [N/N]');
nexttile(2);
plot(om, angle(N), 'r');
grid on;
xlabel('Omega [rad/s]');
ylabel('phase [rad]');
```

```
figure(6);
t = tiledlayout(2, 1);
title(t, 'Shear force in the middle of the GE tube (force in C)');
nexttile(1);
semilogy(om, abs(T), 'b');
grid on;
xlabel('Omega [rad/s]');
ylabel('amplitude [N/N]');
nexttile(2);
plot(om, angle(T), 'b');
grid on;
xlabel('Omega [rad/s]');
ylabel('phase [rad]');

figure(7);
t = tiledlayout(2, 1);
title(t, 'Bending moment in the middle of the GE tube (force in C)');
nexttile(1);
semilogy(om, abs(M), 'r');
grid on;
xlabel('Omega [rad/s]');
ylabel('amplitude [Nm/N]');
nexttile(2);
plot(om, angle(M), 'r');
grid on;
xlabel('Omega [rad/s]');
ylabel('phase [rad]');

%% (3C) Constraint force in C. Plot the corresponding magnitude and phase.
% The horizontal reaction force of the "C" support will be:
ndof_RO3 = idb(3,1) - ndof;
FRF_RO3 = rr(ndof_RO3, :); % computing the FRF
figure(8);
t = tiledlayout(2,1);
title(t, 'Constraint force in node C (due to force in C)');
nexttile(1);
plot(om, abs(FRF_RO3), 'r');
ylabel('amplitude [N/N]');
xlabel('Omega [rad/s]');
```

```
grid on;
nexttile(2);
plot(om, unwrap(angle(FRF_RO3)), 'b');
ylabel('phase [rad]');
xlabel('Omega [rad/s]');
grid on;

%% 4. Using the modal superposition approach, calculate the frequency...
% Modal matrices (from the "loadstructure" function):
NMODES = 2;
ii = 1:NMODES;
Phi = modes(:, ii);
Mmod = Phi'*MFF*Phi;
Kmod = Phi'*KFF*Phi;
Cmod = Phi'*CFF*Phi;
Fmod = Phi'*F0;
% For the first two mode shapes, FRF is computed:
for ii = 1:length(om)
   % To obtain the vector in principal coordinates:
   xx_mod(:, ii) = (-om(ii)^2*Mmod+1i*om(ii)*Cmod+Kmod)\Fmod;
   xx_acc_mod(:, ii) = -om(ii)^2*xx_mod(:, ii);
end
xx_m = Phi*xx_mod; % vector of free coordinates
xx_acc_m = Phi*xx_acc_mod; % computing the acceleration in the same way

% As before, being F0 unitary, xx_m are already the FRFs we wanted
figure(9); % plotting point F
t = tiledlayout(2,2);
title(t, 'Node F: vertical disp. and acc. (modal approach, force in C)');
nexttile(1);
hold on;
plot(om, abs(xx_m(idb(6,2), :)), 'r');
plot(om, abs(X(idb(6,2), :)), 'g');
legend('Modal approach', 'FEM'); % first comparison
hold off;
title('Vertical displacement of node F');
ylabel('amplitude [m/N]');
xlabel('Omega [rad/s]');
grid on;
```

```
nexttile(3);
hold on;
plot(om, angle(xx_m(idb(6,2), :)), 'b');
plot(om, angle(X(idb(6,2), :)), 'g');
legend('Modal approach', 'FEM'); % second comparison
hold off;
ylabel('phase [rad]');
xlabel('Omega [rad/s]');
grid on;
nexttile(2);
hold on;
plot(om, abs(xx_acc_m(idb(6,2), :)), 'r');
plot(om, abs(Xpp(idb(6,2), :)), 'g');
legend('Modal approach', 'FEM');
hold off;
title('Vertical acceleration of node F');
ylabel('amplitude [(m/s2)/N]');
xlabel('Omega [rad/s]');
grid on;
nexttile(4);
hold on;
plot(om, angle(xx_acc_m(idb(6,2), :)), 'b');
plot(om, angle(Xpp(idb(6,2), :)), 'g');
legend('Modal approach', 'FEM');
hold off;
ylabel('phase [rad]');
xlabel('Omega [rad/s]');
grid on;

figure(10); % plotting point H
t = tiledlayout(2,2);
title(t, 'Node H: horizontal disp. and acc. (modal approach, force in C)');
nexttile(1);
hold on;
plot(om, abs(xx_m(idb(8,1), :)), 'r');
plot(om, abs(X(idb(8,1), :)), 'g');
legend('Modal approach', 'FEM');
hold off;
title('Horizontal displacement of node H');
```

```
ylabel('amplitude [m/N]');
xlabel('Omega [rad/s]');
grid on;
nexttile(3);
hold on;
plot(om, angle(xx_m(idb(8,1), :)), 'b');
plot(om, angle(X(idb(8,1), :)), 'g');
legend('Modal approach', 'FEM');
hold off;
ylabel('phase [rad]');
xlabel('Omega [rad/s]');
grid on;
nexttile(2);
hold on;
plot(om, abs(xx_acc_m(idb(8,1), :)), 'r');
plot(om, abs(Xpp(idb(8,1), :)), 'g');
legend('Modal approach', 'FEM');
hold off;
title('Horizontal acceleration of node H');
ylabel('amplitude [(m/s2)/N]');
xlabel('Omega [rad/s]');
grid on;
nexttile(4);
hold on;
plot(om, angle(xx_acc_m(idb(8,1), :)), 'b');
plot(om, angle(Xpp(idb(8,1), :)), 'g');
legend('Modal approach', 'FEM');
hold off;
ylabel('phase [rad]');
xlabel('Omega [rad/s]');
grid on;
% Having used the modal superposition approach, the FRF gives accurate
% results in the frequency range of the first 2 modes (until f2 = 330 Hz).
% The quasi-static region is well represented considering
% the first 2 modes with the modal FRF.

%% 5. Plot the... relating the input vertical displacement at point A'...
% To obtain the FRF, the displacement must be imposed equal to 1. We use
% the superposition approach: it's equivalent (being a linear system) to
```

```
% use a force, pointing upwards (since zA' is pointing upward,
% from the forces' diagram), that is equal to ky (being zA' unitary)
Fk = zeros(ndof, 1);
Fk(idb(1,2)) = ky;
for ii = 1:length(om)
    A = (-om(ii)^2*MFF + 1i*om(ii)*CFF + KFF);
    X_k(:, ii) = A\Fk;
    Xpp_k(:, ii) = -om(ii)^2*X_k(:,ii);
end
FRF_accF_A = Xpp_k(idb(6,2),:); % transfer function (acc)
figure(11);
t = tiledlayout(2,1);
title(t, 'Vertical acceleration of point F (displacement of A'')');
nexttile(1);
plot(om, abs(FRF_accF_A), 'r');
ylabel('amplitude [s^-2]');
xlabel('Omega [rad/s]');
grid on;
nexttile(2);
plot(om, unwrap(angle(FRF_accF_A)), 'b');
ylabel('phase [rad]');
xlabel('Omega [rad/s]');
grid on;

%% 6. When the bike travels at a speed of 12 m/s on a sinuisodal road...
% We must convert the data into the cosinuisodal functions with
% calculations. Writing down the data:
lambda1 = 1; % [m]
amp1 = 1e-3; % [m]
lambda2 = 0.6; % [m]
amp2 = 0.5e-3; % [m]
v = 12; % [m/s]
% z(i)_A = Ai*cos(2pi/lambdai*v*time);
% z(i)_B = Ai*cos(2pi/lambdai*v*time - 2pi/lambdai*1.12);
distanceAB = 1.12; % [m]
phi1_A = 0;
phi2_A = 0;
phi1_B = -2*pi/lambda1*distanceAB;
phi2_B = -2*pi/lambda2*distanceAB;
```

```
OMEGA1 = 2*pi/lambda1*v;
OMEGA2 = 2*pi/lambda2*v;

% Let's draw the track irregularities (amplitude):
dt = 0.0001;
time = 0:dt:0.5;
z_A = @(t) amp1*cos(OMEGA1*t)+amp2*cos(OMEGA2*t);
irregularities = z_A(time);
figure(12);
plot((v*time), irregularities, 'b');
title('Track irregularities');
xlabel('Space [m]');
ylabel('Track amplitude [m]');
grid on;

% We can find now the output from A:
FRF_accH_A = Xpp_k(idb(8,2),:);
% Doing the same with displacements:
FRF_z_H_A = X_k(idb(8,2),:);
% Now we compute the output from B:
Fk = zeros(ndof, 1);
Fk(idb(2,2)) = ky; % the same, different node
for ii = 1:length(om)
    A = (-om(ii)^2*MFF + 1i*om(ii)*CFF + KFF);
    X_k(:, ii) = A\Fk;
    Xpp_k(:, ii) = -om(ii)^2*X_k(:,ii);
end
FRF_accH_B = Xpp_k(idb(8,2),:); % transfer function (acc)
% Doing the same with displacements:
FRF_z_H_B = X_k(idb(8,2),:);

% We use the the superposition principle for both now
% Finding where the most "suitable" omega is (minimum difference, precautionary):
diff_om1 = abs(om - OMEGA1);
diff_om2 = abs(om - OMEGA2);
find1 = diff_om1==min(diff_om1);
find2 = diff_om2==min(diff_om2);
aH_A1 = FRF_accH_A(find1);
aH_A2 = FRF_accH_A(find2);
```

```
aH_B1 = FRF_accH_B(find1);
aH_B2 = FRF_accH_B(find2);
% Doing the same for displacements:
zH_A1 = FRF_z_H_A(find1);
zH_A2 = FRF_z_H_A(find2);
zH_B1 = FRF_z_H_B(find1);
zH_B2 = FRF_z_H_B(find2);
% We have selected the accelerations in the FRFs that match the desired omegas
acc_t_A = @(t) abs(aH_A1)*amp1*cos(OMEGA1*t+angle(aH_A1)) + ...
    abs(aH_A2)*amp2*cos(OMEGA2*t+angle(aH_A2));
acc_t_B = @(t) abs(aH_B1)*amp1*cos(OMEGA1*t+angle(aH_B1)+phi1_B) + ...
    abs(aH_B2)*amp2*cos(OMEGA2*t+angle(aH_B2)+phi2_B);
% Let's compute each acceleration:
acc_H_A = acc_t_A(time);
acc_H_B = acc_t_B(time);
acc_H = acc_H_A + acc_H_B; % overlapping the two accelerations
% We now plot the total acceleration (and its components):
figure(13);
title('Acceleration of point H (due to the road)');
hold on;
plot(time, acc_H);
plot(time, acc_H_A, '--');
plot(time, acc_H_B, '--');
legend('total', 'forced by A', 'forced by B');
xlabel('[s]');
ylabel('[m/s^2]');
grid on;
hold off;
% Doing the same for the one obtained with displacements:
acc_tfromz_A = @(t) - OMEGA1^2*abs(zH_A1)*amp1*cos(OMEGA1*t+angle(zH_A1)) + ...
    - OMEGA2^2*abs(zH_A2)*amp2*cos(OMEGA2*t+angle(zH_A2));
acc_tfromz_B = @(t) - OMEGA1^2*abs(zH_B1)*amp1*cos(OMEGA1*t+angle(zH_B1)+phi1_B) +
...
    - OMEGA2^2*abs(zH_B2)*amp2*cos(OMEGA2*t+angle(zH_B2)+phi2_B);
acc_H_Afromz = acc_tfromz_A(time);
acc_H_Bfromz = acc_tfromz_B(time);
acc_H_z = acc_H_Afromz+acc_H_Bfromz;
figure(14);
title('Acceleration of point H, computed with displacements');
```

```
hold on;
plot(time, acc_H_z);
plot(time, acc_H_Afromz, '--');
plot(time, acc_H_Bfromz, '--');
legend('total', 'forced by A', 'forced by B');
xlabel('[s]');
ylabel('[m/s^2]');
grid on;
hold off;

%% 7. Compute the static response of the structure due to the weight...
FS = zeros(ndof, 1);
FS(idb(8,2)) = - 600;
FS(idb(6,2)) = - 100;
xF = KFF\FS;
% We now rely on the "diseg2" function to draw the static deflection
scale_factor2 = 10;
figure(15);
diseg2(xF, scale_factor2, incid, l, gamma, posit, idb, xy);
title('Static deflection (due to weight)');
% Vector xF contains all the ACTUAL nodal displacement due to the force F
% in the global reference frame
vertical_displacement_midpoint_GE = xF(idb(13,2))*1e3 % Having it in mm
% The vertical displacement will be (-0.0011 m), which is equal to -1.1213 mm.
```