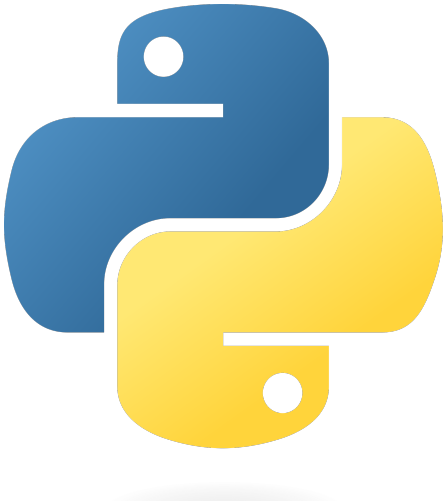
University of Liège
Faculty of Applied Sciences

Introduction to Python

Exercises



Academic year 2023-2024

Guidelines

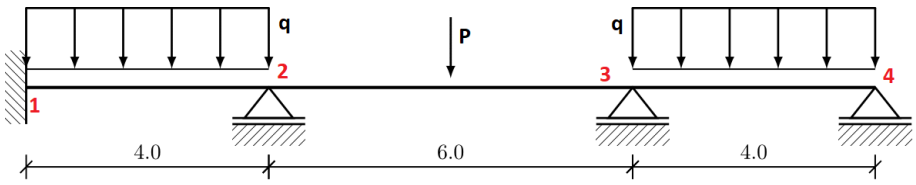
You are asked to solve as much of the following PYTHON problems. They are designed to be accessible for beginners, but show a sufficient level of difficulty so that they require a little bit of thinking before coding. The degree of difficulty of each exercise is given by the following icons: (∗) fairly straightforward, (∗∗) a little bit more difficult, (∗∗∗) longer and requires a larger amount of PYTHON concepts.
You are expected to solve all the eight exercises labelled by ♠ during your PYTHON training. The remaining exercises are supplementary exercises that you can realize by yourself to understand PYTHON even better.

Note that most of the exercises are put in an engineering context, so that you may be encounter similar problems during classical courses.

Contents

1. Bending moment diagram (∗) (♠)

Bending moment diagram is an analytical tool used in conjunction with structural analysis to help to perform structural design by determining the value of bending moment at a given point of a structural element such as a beam. Consider the following beam ($P = 400\,\text{kN}$, $q = 25\,\text{kN/m}$, and distances in m):



By using the rotation method, a method used in indeterminate structures analysis, we find the following bending moments at each point:

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{bmatrix} = \begin{bmatrix} -69.14 \\ -238.27 \\ -218.52 \\ 0 \end{bmatrix} \text{ kNm.}$$

Bending moments at each point between nodes can be evaluated by the following simplified formula:

$$M(x) = \mu(x) + M_{\text{left}} \cdot \frac{L_{\text{span}} - x}{L_{\text{span}}} + M_{\text{right}} \cdot \frac{x}{L_{\text{span}}} \tag{1}$$

with, for a simply supported beam,

$$\mu(x) = \begin{cases} \dfrac{P \cdot x}{4}, & \text{for a point loading } P \text{ at midspan and x smaller than L/2 ,} \\ \dfrac{P \cdot (L - x)}{4}, & \text{for a point loading } P \text{ at midspan and x bigger than L/2,} \\ \dfrac{q \cdot L \cdot x}{2} - \dfrac{q \cdot x^2}{2}, & \text{for an uniform load } q. \end{cases} \tag{2}$$

Questions:

(i) Write a script that calculates bending moment at each point (discretize each span as you want).

(ii) Plot the bending moment diagram. The figure should contain a title and axes should be labeled properly.

2. Eigenvalues and eigenvectors of a mechanical system (∗) (♠)

For an undamped mechanical system, the equations of motion take the following form:

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{f}, \tag{3}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{K}$ is the stiffness matrix, $\mathbf{f}$ is the vector of external loads, and $\mathbf{q}$ is the vector containing the degrees of freedom of the system.

The n eigenvalues of the system are then defined as the solutions ω_i of

$$\det(\mathbf{K} - \omega_i^2\mathbf{M}) = 0, \quad i = 1, ..., n \tag{4}$$

and the associated eigenvectors are given by the solutions $\mathbf{x}_i$ of

$$(\mathbf{K} - \omega_i^2\mathbf{M})\mathbf{x}_i = \mathbf{0}, \quad i = 1, ..., n. \tag{5}$$

You are given the features of the system portrayed in Fig.1:

$$\mathbf{q}(t) = \begin{Bmatrix} q_1(t) \\ q_2(t) \end{Bmatrix}, \quad \mathbf{M} = \begin{bmatrix} 3m & 0 \\ 0 & m \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 2k & -k \\ -k & k \end{bmatrix}, \quad \mathbf{f} = \begin{Bmatrix} 0 \\ F_0 \end{Bmatrix}, \tag{6}$$

where $m = 1\,\text{kg}$, $k = 5\,\text{N/m}$ and $F_0 = 10\,\text{N}$.

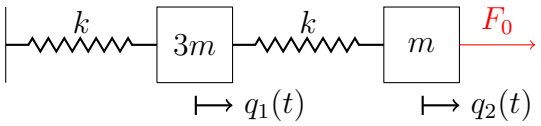


Figure 1: Schematic of the mechanical problem.

Questions:

(i) Determine, using PYTHON, the eigenvalues of the system.

(ii) Determine, using PYTHON, the eigenvectors of the system.

Hint: Use the `scipy.linalg` module

3. Dynamic balancing (∗)

Consider a rotating shaft (Fig. 2). If its center of mass does not go through the rotation axis, or if its inertia tensor is not diagonal in the reference frame associated to the rotation, there will be additional stress applied on the structure and its bearing, which is to be avoided.

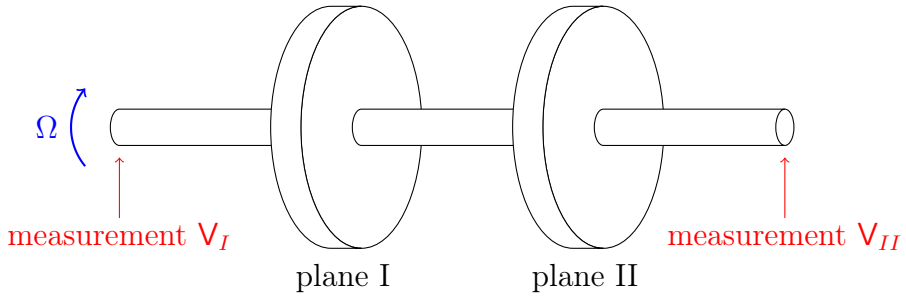


Figure 2: Schematic of the two-plane balancing method.

Therefore, there exist methods to detect where the unbalance comes from. One of them, called the *two-plane balancing method* consist in putting trial weights and measure their influence on the induced vibration, to come back to the initial unbalance.

We place ourselves in the rotating frame. The induced vibration is then represented as a vector, and the additional masses (initial unbalance and added weights) are also represented by vectors, their norm being the value of the mass, whereas their phase indicates their position. Instead of using vectors, we use the complex numbers formalism, so that the calculations become very straightforward.

Given the initial vibration measurement $\mathsf{V}_{0,I} = 1.7e^{i\frac{174}{180}\pi}$ mm/s, $\mathsf{V}_{0,II} = 0.47e^{i\frac{264}{180}\pi}$ mm/s, an added weight $\mathsf{M}_I = 3.2\,\mathrm{g}$ resulting in vibration measurement $\mathsf{V}_{1,I} = 1.41e^{i\frac{119}{180}\pi}$ mm/s, $\mathsf{V}_{1,II} = 0.32e^{i\frac{253}{180}\pi}$ mm/s and an added weight $\mathsf{M}_{II} = 3.2\,\mathrm{g}$ resulting in vibration measurement $\mathsf{V}_{2,I} = 1.62e^{i\frac{168}{180}\pi}$ mm/s, $\mathsf{V}_{2,II} = 0.94e^{i\frac{29}{180}\pi}$ mm/s, the initial unbalance in the planes I and II are given by

$$\begin{Bmatrix} \mathsf{B}_I \\ \mathsf{B}_{II} \end{Bmatrix} = \begin{bmatrix} \mathsf{a}_{1,I} & \mathsf{a}_{1,II} \\ \mathsf{a}_{2,I} & \mathsf{a}_{2,II} \end{bmatrix}^{-1} \begin{Bmatrix} \mathsf{V}_{0,I} \\ \mathsf{V}_{0,II} \end{Bmatrix}, \tag{7}$$

where

$$\mathsf{a}_{1,I} = \frac{\mathsf{V}_{1,I} - \mathsf{V}_{0,I}}{\mathsf{M}_I}, \quad \mathsf{a}_{1,II} = \frac{\mathsf{V}_{2,I} - \mathsf{V}_{0,I}}{\mathsf{M}_{II}}, \tag{8}$$

$$\mathsf{a}_{2,I} = \frac{\mathsf{V}_{1,II} - \mathsf{V}_{0,II}}{\mathsf{M}_I}, \quad \mathsf{a}_{2,II} = \frac{\mathsf{V}_{2,II} - \mathsf{V}_{0,II}}{\mathsf{M}_{II}}. \tag{9}$$

Questions:

(i) Compute the coefficients $\mathsf{a}_{i,J}$ for $i \in \{1, 2\}$ and $J \in \{I, II\}$.

(ii) Determine the initial unbalance in planes I and II, *i.e.* B_I and B_{II}.

4. Hermite polynomials (∗) (♠)

Hermite polynomials are defined as

$$H_n(x) = \sum_{k=0}^{\lfloor n/2 \rfloor} (-1)^k \frac{n!}{2^k k!(n-2k)!} x^{n-2k}, \quad n \in \mathbb{N}, \tag{10}$$

where $\lfloor x \rfloor$ is the floor function, *i.e.* the largest integer smaller or equal to x. It can be computed via the `floor` function in the NUMPY library. The factorial function is already implemented in the `math` module as `factorial`.

Questions:

(i) Write a function `hermite(n, x)` that computes the value of $H_n(x)$.

(ii) Plot the four first hermite polynomials, that is $H_n(x)$ with $n \in \{0, 1, 2, 3\}$, for $x \in [-2, 2]$. The axes should be labeled properly, and a legend should be displayed.

5. 2018 Evaluation test (∗)

Exercise 1: Factorial $N!$ of a number N

(i) Write scripts which enable to calculate $N!$ by using a `while` loop first and then with a `for` loop.

(ii) Thanks to the profiler, compare time taken by the two scripts and the `factorial` command from the `math` module.

Exercise 2: Fibonacci sequence

(i) Write a function that returns the n-th term of the Fibonacci sequence. An extract of this sequence is the following:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, ...$$

so that it can defined by $x_1 = 0$, $x_2 = 1$ and $x_n = x_{n-1} + x_{n-2}$, $n > 2$.

(ii) Write a second function that returns all of the n first terms of this sequence as a vector.

Exercise 3: Plots

Plot on a same figure the sine, cosine and tangent functions. Choose the intervals of axis as you want but axes should be labeled properly, and a legend should be displayed.

6. Bending resistance of a wood beam (∗)

Consider a simply supported beam made of wood. The bending resistance of the beam is given by the following equations

$$\begin{cases} k_m \dfrac{\sigma_y}{f_{m,d}} + \dfrac{\sigma_z}{f_{m,d}} & \leq 1, \\ \dfrac{\sigma_y}{f_{m,d}} + k_m \dfrac{\sigma_z}{f_{m,d}} & \leq 1, \end{cases} \tag{11}$$

where σ_y is the stress due to the bending moment along the y axis, σ_z is the stress due to the bending moment along the z axis and $k_m = 0.7$, $f_{m,d} = 0.9 \cdot 18/1.3\,\text{MPa}$ are coefficients defined by Eurocode 5.

The dimensions of the cross-section of the beam are $b = 80\,\text{mm}$ and $h = 80\,\text{mm}$. The length of the beam is $L = 2\,\text{m}$. It is subjected to a point load $P = 1000\,\text{N}$ along the y axis and a point load $Q = 750\,\text{N}$ along the z axis at the middle of the beam (Fig. 3).

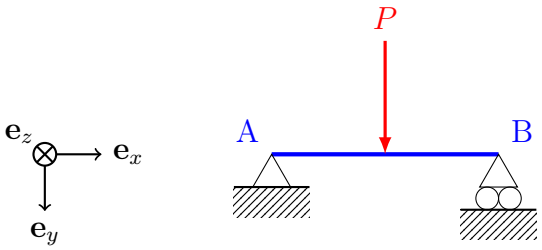


Figure 3: Simply supported beam subjected to a vertical point load at midspan.

Questions:

(i) Write a PYTHON function that takes as arguments the length of the beam L and the loads P and Q to check the resistance of the beam.

(ii) Adapt your script to display a message `"Resistance ok"` if the beam resists or `"Resistance not ok"` if it is not the case.

(iii) Now consider $L = 3\,\text{m}$, $P = 1250\,\text{N}$ and $Q = 500\,\text{N}$. What happens ?

7. Free response of a spring-mass-damper system (**) (♠)

Consider a spring-mass-damper system. If there is no other force acting on the system, Newton's law writes

$$m\ddot{x} + c\dot{x} + kx = 0 \tag{12}$$

where m, c and k are the equivalent mass, damping and spring of the system. Defining the resonant frequency $\omega_0 = \sqrt{k/m}$ and the damping ratio $\zeta = c/(2\sqrt{km})$, it reduces to the canonical form

$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2 x = 0 \tag{13}$$

For the initial conditions $(x(0), \dot{x}(0)) = (x_0, \dot{x}_0)$, the solution is then given by:

- if $0 < \zeta < 1$,

$$x(t) = e^{-\zeta\omega_0 t}\left[x_0\cos(\omega_0\sqrt{1-\zeta^2}\,t) + \frac{\dot{x}_0 + \zeta\omega_0 x_0}{\omega_0\sqrt{1-\zeta^2}}\sin(\omega_0\sqrt{1-\zeta^2}\,t)\right]. \tag{14}$$

- if $\zeta = 1$,

$$x(t) = e^{-\zeta\omega_0 t}\left[x_0 + (\dot{x}_0 + \zeta\omega_0 x_0)t\right]. \tag{15}$$

- if $\zeta > 1$,

$$\begin{aligned} x(t) = e^{-\zeta\omega_0 t}\Bigg[&\frac{\dot{x}_0 + (\zeta + \sqrt{\zeta^2-1})\omega_0 x_0}{2\omega_0\sqrt{\zeta^2-1}}e^{\omega_0\sqrt{\zeta^2-1}\,t} \\ &+ \frac{-\dot{x}_0 + (-\zeta + \sqrt{\zeta^2-1})\omega_0 x_0}{2\omega_0\sqrt{\zeta^2-1}}e^{-\omega_0\sqrt{\zeta^2-1}\,t}\Bigg]. \end{aligned} \tag{16}$$

Questions:

(i) Write a function that takes as arguments the parameters m, c and k and returns the resonant frequency ω_0 as well as the damping ratio ζ.

(ii) Write a function that takes as arguments the parameters m, c and k, as well as the initial conditions $(x_0, \dot{x}_0)$ and a time T, and plots $x(t)$ for $t \in [0, T]$.

(iii) Add an option to the previous function to specify the color of the plot.

(iv) Plot $x(t)$ if $m = 3\,\text{kg}$, $k = 12\,\text{N/m}$, $x_0 = 2\,\text{m}$, $\dot{x}_0 = 1\,\text{m/s}$, for $t \in [0, 10]\,\text{s}$, and for $c \in \{1, 12, 15\}\,\text{Ns/m}$. All the plots should be on the same figure. The axes should be labeled properly.

8. Linear regression (**)

The linear regression problem consists, giving a series of points $(x_1, y_1), ..., (x_n, y_n)$, in finding the appropriate coefficients (α, β) of the linear equation $y(x) = \alpha x + \beta$, so that the following relation holds:

$$y_i \simeq y(x_i) = \alpha x_i + \beta. \tag{17}$$

The *least square method* tries to find (α, β) by minimizing $\sum_{i=1}^{n} |y(x_i) - y_i|^2$ with respect to α and β; and one can show that by doing so, we would get the following equations:

$$\begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{bmatrix} \begin{Bmatrix} \alpha \\ \beta \end{Bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{bmatrix}. \tag{18}$$

Questions:

(i) Write a function that takes as arguments the two vectors of data $[\mathtt{x}_1, ..., \mathtt{x}_n]$, $[\mathtt{y}_1, ..., \mathtt{y}_n]$, and returns the coefficients α and β, using (18).

(ii) After doing an experiment, the following data is obtained:

$\varepsilon \times 10^4$ [-]	0.0535	0.1001	0.1462	0.1950	0.2400	0.2977	0.3408	0.3868
σ [MPa]	1.4701	1.8769	3.1496	3.9615	5.1777	5.8470	6.7195	7.7155

Write a script to plot these points. In the same figure, plot a linear regression, using the previously defined function. The axes should be labeled properly.

9. Numerical simulation of a spring-mass system (**) (♠)

One way to simulate the dynamics of a mechanical system is to discretize Newton's equation, according to the following scheme (here in 1D):

$$\begin{cases} \dot{v}(t) = \frac{1}{m}F(x(t), \dot{x}(t), t) \\ \dot{x}(t) = v(t) \end{cases} \quad \Rightarrow \quad \begin{cases} v(t+\Delta t) \leftarrow v(t) + \frac{\Delta t}{m}F(x(t), v(t), t) \\ x(t+\Delta t) \leftarrow x(t) + v(t+\Delta t)\Delta t \end{cases} \tag{19}$$

Questions:

(i) Implement this numerical scheme, for a spring force $F(x(t)) = -kx(t)$, as a function that computes $(x(t), v(t))$ for $t = 0, ..., (N-1)\Delta t$, given k, m, Δt, N and the initial conditions $(x(0), v(0))$.

(ii) Using the previously defined function, simulate the position for $t \in [0, 10]\,\mathrm{s}$, if $k = 5\,\mathrm{N/m}$, $m = 1\,\mathrm{kg}$ and $(x(0), v(0)) = (5\,\mathrm{m}, 0\,\mathrm{m/s})$, using a time step $\Delta t = 0.01\,\mathrm{s}$. The axes should be labeled properly.

(iii) Display the trajectory obtained in the phase space, *i.e.* a plot where the coordinates are $(x(t), v(t))$, $t \in [0, 10]\,\mathrm{s}$. The axes should be labeled properly.

10. Lagrange interpolation (**)

Lagrange interpolation is a method that allows to build a polynomial of degree $(n-1)$ that goes through a series of points $(x_1, y_1), ..., (x_n, y_n)$. The resulting polynomial is given by

$$\mathcal{L}(x) = \sum_{i=1}^{n} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}, \tag{20}$$

so that $\mathcal{L}(x_i) = y_i$, $i \in \{1, ..., n\}$. For instance, if $(x_1, y_1) = (1, 1)$, $(x_2, y_2) = (2, 3)$ and $(x_3, y_3) = (3, 1)$, then

$$\begin{aligned}
\mathcal{L}(x) &= y_1 \frac{x - x_2}{x_1 - x_2}\frac{x - x_3}{x_1 - x_3} + y_2 \frac{x - x_1}{x_2 - x_1}\frac{x - x_3}{x_2 - x_3} + y_3 \frac{x - x_1}{x_3 - x_1}\frac{x - x_2}{x_3 - x_2} && (21)\\
&= 1 \frac{x - 2}{1 - 2}\frac{x - 3}{1 - 3} + 3 \frac{x - 1}{2 - 1}\frac{x - 3}{2 - 3} + 1 \frac{x - 1}{3 - 1}\frac{x - 2}{3 - 2} && (22)\\
&= -2x^2 + 8x - 5. && (23)
\end{aligned}$$

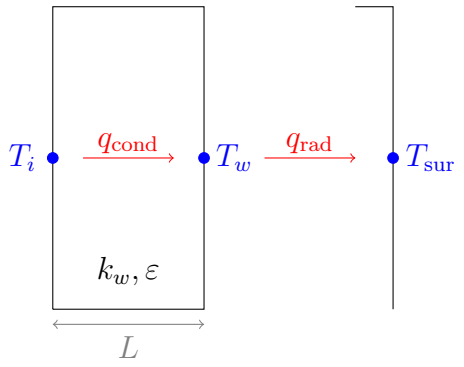
Questions:

(i) Write a function `lagrange(coord_x, coord_y, x)` that computes the value of $\mathcal{L}(\mathtt{x})$, given that `coord_x` $= [\mathtt{x}_1, ..., \mathtt{x}_n]$ and `coord_y` $= [\mathtt{y}_1, ..., \mathtt{y}_n]$.

(ii) Plot $\mathcal{L}(x)$ for the points $(x_1, y_1) = (-2, 1)$, $(x_2, y_2) = (-1, 4)$, $(x_3, y_3) = (1, 0)$, $(x_4, y_4) = (2, 2)$ for $x \in [-3, 3]$. The axes should be labeled properly.

11. Wall temperature (**) (♠)

We consider the simplified heat transfer problem portrayed in Fig.4: given a room temperature T_i, we seek to find the external room temperature T_w, if the external surroundings (mainly the sky) is at a given temperature T_{sur}.



$T_i \in \{280, 290, 300\}\,\text{K}$
$T_{\text{sur}} = 273.15\,\text{K}$
$k_w = 0.75\,\text{W/(m·K)}$
$\varepsilon = 0.92$
$L = 0.06\,\text{m}$

Figure 4: Schematic of the heat-transfer problem.

Under suitable assumptions[1], one gets the following non-linear equation:

$$\frac{k_w}{L}(T_i - T_w) = h_r(T_{\text{sur}}, T_w)(T_w - T_{\text{sur}}), \tag{24}$$

where $h_r(T_w, T_{\text{sur}}) = \sigma\varepsilon(T_w + T_{\text{sur}})(T_w^2 + T_{\text{sur}}^2)$ with Stefan-Boltzmann constant $\sigma = 5.67 \times 10^{-8}\,\text{W/(m}^2\text{·K}^4\text{)}$.

Because this equation is non-linear, we suggest the following numerical scheme, where $T_w^{(i)}$ is the estimation of T_w at the i-th step:

- guess a value $T_w^{(0)}$;
- **do**

  compute $h_r^{(i)} = \sigma\varepsilon(T_w^{(i)} + T_{\text{sur}})\left({T_w^{(i)}}^2 + T_{\text{sur}}^2\right)$;

  compute $T_w^{(i+1)} = \dfrac{h_r^{(i)} L T_{\text{sur}} + k_w T_i}{h_r^{(i)} L + k_w}$;

  while $|T_w^{(i+1)} - T_w^{(i)}| > \text{tol} = 10^{-4}$.

Questions:

(i) Implement the suggested numerical scheme as a function `T_w=Temp_Wall(T_w_guess, T_i)` that returns the iterative steps: `T_w=[T_w(1), T_w(2), ...]`, given a guess `T_w_guess` and `T_i`.

(ii) Plot the different values of $T_w^{(i)}$ if $T_i \in \{280, 290, 300\}\,\text{K}$, on a same graph. You can take 300 K as a first guess. The axes should be labeled properly.

[1]Here are the assumptions: steady-state, one dimensional conduction through the wall, constant thermal conductivity and emissivity, negligible convection, negligible solar radiation, the surroundings are large with respect to the wall, and the wall is a diffuse and grey surface.

12. Clamped bar subjected to point load (**) (♠)

Consider a vertical clamped bar which is subjected to a point load P at its free end (Fig.5):

(a) Single finite element model. (b) Two finite elements model.

Figure 5: Schematic of a clamped bar subjected to point load.

First, the bar is considered as single finite element. This element is a bar with a stiffness matrix

$$\mathbf{K}_{\text{el}} = \frac{EA}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = k\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{25}$$

To determine the support reaction, the following system of equations must be solved

$$\mathbf{K}_{\text{el}}\mathbf{q}_{\text{el}} = \mathbf{F}_{\text{el}} \quad \Leftrightarrow \quad k\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} R \\ P \end{bmatrix}, \tag{26}$$

where R is the support reaction. Given the support conditions, the nodal displacement u_1 is 0 and you can strike out the first line and the first column of the stiffness matrix. Then, you can solve the system to get the unknown displacement and finally retrieve the support reaction.

If you consider a bar divided in 2 finite elements, each element has 2 nodes and is characterized by the previously defined $\mathbf{K}_{\text{el}}$ matrix. However, since the structure has 3 nodes, the structural stiffness matrix $\mathbf{K}$ will be a 3×3 matrix. We need to add the stiffness matrices from each element which is called *assembly*. After the assembly procedure, the system to solve is

$$\mathbf{K}\mathbf{q} = \mathbf{F} \quad \Leftrightarrow \quad \begin{bmatrix} k_I & -k_I & 0 \\ -k_I & k_I + k_{II} & -k_{II} \\ 0 & -k_{II} & k_{II} \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} R \\ 0 \\ P \end{bmatrix} \tag{27}$$

and the principle to solve the system is the same as before.

Questions:

(i) Find the reaction R using the described procedure for a bar with only 1 element. Use $E = 210\,000\,\text{MPa}$, $A = 40 \times 40\,\text{mm}^2$, $L = 3\,\text{m}$ and $P = 10\,\text{kN}$. Do not take self weight into account.

(ii) Do the same for a bar divided in 2 finite elements.

(iii) Do the same for a bar divided in 100 finite elements.

13. Number of divisors (**)

We consider a natural number $n \in \mathbb{N}_0$. The main goal is to determine the number of divisors $d(n)$ of n.

Questions:

(i) Write a function that computes $d(n)$ naively, *i.e.* by going through all the numbers $\{1, ..., n\}$ and by checking whether or not they divide n.

(ii) The previous algorithm can be significantly improved by making the following remarks:

- 1 and n are always divisors of n;
- If p divides n, then n/p also divides n.

Improve your function given these considerations.

(iii) Compare the execution time of the two previous methods by testing them on $n = 10^8$, using the profiler or the `time` module.

(iv) Write a function that checks whether or not a number is prime, *i.e.* a natural number strictly greater than one that is only divisible by one and himself.

(v) Write a function that checks whether or not a number n is *perfect*, *i.e.* whether or not n can be written as the sum of its divisors (where n itself is not counted as a divisor). For instance, 28 is a perfect number since $28 = 1 + 2 + 4 + 7 + 14$.
Hint: slightly modify the function implemented in (ii).

14. Prime factorization (**)

Given a number $n \in \mathbb{N}\backslash\{0,1\}$, one may want to know its prime factorization, *i.e.* the set of prime numbers $\{p_1, p_2, ..., p_k\}$ such that n can be written as

$$n = \prod_{i=1}^{k} p_i^{\alpha_i}, \tag{28}$$

where α_i is the multiplicity degree of the prime number p_i in the decomposition of n. For instance, 150 can be written as $2 \times 3 \times 5^2$, and the multiplicity degree of 5 is 2.

Questions:

(i) Write a function `prime_factors(n)` that returns the prime factorization of `n` as a vector containing its prime factors. If the multiplicity degree of a prime is greater than one, then the prime number should appear as many times as the multiplicity degree. For instance, `prime_factors(150)` should return `[2, 3, 5, 5]`.

(ii) Write a function `decomposition(n)` that returns `[p, alpha]`, where `p` is the set of prime numbers present in the decomposition of `n`, but they should only be counted once. `alpha` is the vector containing the multiplicity degree associated to each prime in `p`. For instance, `decomposition(150)` should return `[p, alpha]` with `p=[2, 3, 5]` and `alpha=[1, 1, 2]`. You can use the `prime_factors` function.

15. European buckling curves (***)

Buckling instabilities are very important for the resistance validation of a metallic element in the construction field. They are introduced in the resistance modelling through a reduction factor χ that allows to take into account parameters, mainly the element elongation. Moreover, different buckling curves are defined for each section class.

The reduction factor will be calculated using the following formulas:

Euler's critical load: $N_{\text{cr}} = \dfrac{\pi^2 EI}{L_{\text{cr}}^2}$ Reduced slenderness: $\overline{\lambda} = \sqrt{\dfrac{A f_y}{N_{\text{cr}}}}$ Reduction factor: $\chi = \dfrac{1}{\phi + \sqrt{\phi^2 - \overline{\lambda}^2}} \leq 1$

where E is Young's modulus, I the inertia modulus, L_{cr} the critical length, A the cross-section area, f_y the elastic limit, and $\phi = 0.5\,[1 + \alpha(\overline{\lambda} - 0.2) + \overline{\lambda}^2]$, where α is the buckling imperfection factor defined by Eurocode 3.

Curve	a_0	a	b	c	d
α	0.13	0.21	0.34	0.49	0.76

Questions:

(i) Draw the curve a_0 first by using a loop and then without using any.

(ii) Create a loop that allows to draw the curves a, b, c and d on a same plot.

(iii) Complete the figure by adding a legend, axes, colors and increasing the curves width.

(iv) Plot each different curve in a separate figure, and include a title that changes in function of the portrayed curve.

(v) Include a request that asks to choose which curve we want to plot.

Consider a steel beam S235 ($f_y = 235\,\text{MPa}$, $E = 210\,000\,\text{MPa}$) of length $L = 8\,\text{m}$, simply supported (buckling coefficient $K = 1$) and with a rectangular cross section of area $A = 12000\,\text{mm}^2$.

(vi) Write a new function that takes as arguments L, E, I, K, A, α and f_y and returns the axial resistance of this beam.

16. Blancmange curve (***) (♠)

The Blancmange function, also called the Takagi fractal curve, is a pathological continuous function. The name blancmange comes from its resemblance to a pudding of the same name (Fig.6).

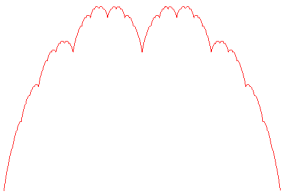


Figure 6: Blancmange curve.

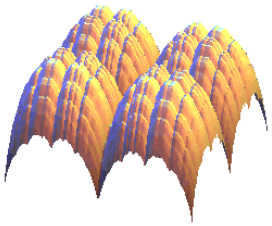


Figure 7: Mount Tagaki surface.

Cartesian equation of this curve is the following:

$$B(x) = \sum_{k=0}^{n} \frac{d(2^k \cdot x)}{2^k}, \tag{29}$$

where $d(x)$ is the distance from x to the nearest integer.

Questions:

(i) Write a function `blancmange(x,n)` that takes as arguments the vector `x` and the sum boundary limit `n` and returns a vector which contains each point of this particular curve.

(ii) Add a condition in this function to take into account the circular version of this curve. The equation of the circular version of this curve is the following:

$$B(x) = \sum_{k=0}^{n} \frac{f(2^k \cdot x)}{2^k} \text{ where } f(x) = 2 \cdot d(x)\dot{(}1 - d(x)). \tag{30}$$

Include a request that asks to user which shape he wants.

(iii) Write a script to plot four blancmange curves with `n=[1 2 3 5 10]` in a same figure.

(iv) Write a script to plot a 3D diagram. The translational surface generated by the translation of a blancmanger curve along another, following $z = B(x) + B(y)$, is an elegant mountain called "Mount Tagaki" (Fig.7).

17. Estimation of π using the `rand` function (***)

An original way to calculate π is to use probability theory. Indeed, consider a square of size 2×2, and randomly generate N points distributed uniformly in it. If we denote by N_c the number of points lying within the circle centered at the center of the square, and of unitary radius, we should get that N_c is close to $N\pi/4$; more formally, we have that[2]

$$\lim_{N \to +\infty} \frac{N_c}{N} = \frac{\pi}{4}. \tag{31}$$

Questions:

(i) Write a function that generates N points uniformly in $[-1, 1] \times [-1, 1]$ (*i.e.* there are two components for each point). Note that the `random.rand(N)` from numpy can generate a vector of `N` independent and uniformly distributed variables in $[0, 1]$.

(ii) Write a function that computes, given the points coordinates $[\mathtt{x}_1, ..., \mathtt{x}_n]$ and $[\mathtt{y}_1, ..., \mathtt{y}_n]$, the number of points lying within the circle centered at $(0, 0)$ with unitary radius. Try to find a solution without using a loop. We recall that our disk is characterized by $\mathcal{D} = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\}$.

(iii) Plot the realization of the experiment, that is the value of $4N_c/N$, as a function of N, where $N \in \{10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7\}$. The axes should be labeled properly.

(iv) Instead of increasing N, we can also simulate a large number of realizations, N being fixed, and calculate the mean of the results. Write a script that determine the mean and the variance of 1000 realizations of the previous experiment, for $N \in \{10, 10^2, 10^3, 10^4\}$.

[2] N.B.: In fact, this limit should be taken as an almost sure convergence.

18. Randomly moving balls (***)

Consider a situation where a large number of balls are coming from the center of a square. Those balls move in the (x, y) plane as time goes by, and their displacement is random.

When a ball reaches one of the square sides, it disappears (see Fig.8).

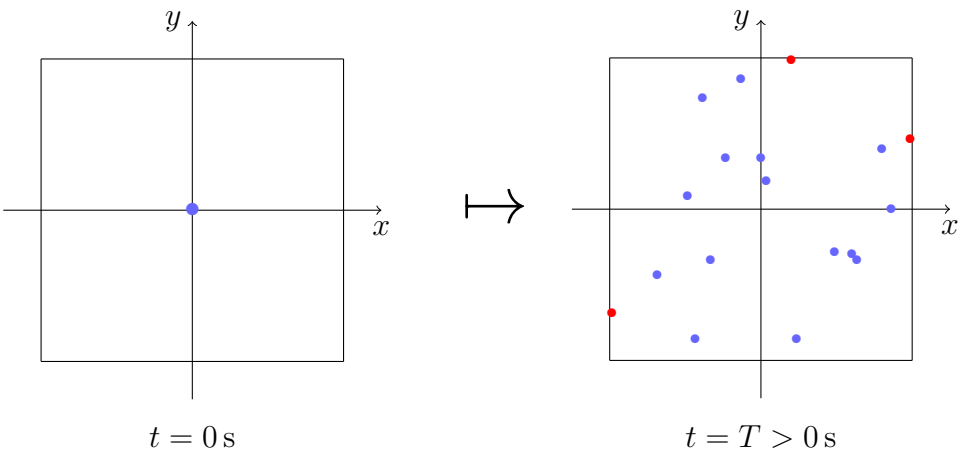


Figure 8: Illustration of the process.

Questions:

(i) Write a code that renders the previously described phenomenon.

(ii) Draw in a figure the square that displays at each iteration the balls that are moving.

(iii) Add a condition that allows to make the balls disappear whenever they reach the limits of the square.

(iv) Add a condition that make the balls change their color when they have reached half the square.

To achieve those you questions, you may need to use the following functions: `random.randn`, `clf` and `pause` from `numpy` and `matplotlib.pyplot`.