Ecole supérieure des Sciences et
de Technologie de Hammam Sousse **2022-2023**

Module : Simulation numérique **LM1**

TP 1

Exercice 1 : Les nombres complexes
On considère les nombres complexes a et b données par

$$a = 1 + i, \qquad b = 1 - 3i$$

Calculer les quantités suivantes

$$a + b \qquad ab \qquad \frac{a}{b} \qquad \overline{a} \qquad \Re e(a) \qquad \Im m(\overline{b}) \qquad |a| \qquad arg(b)\,(en\ degrés)$$

Exercice 2
Que donne le code suivant ?

```
import numpy as np
v = np.array([1, 2, 3, 4]) * np.pi / 4        # np.pi = π
w = np.sin(v)
print(w)
```

Exercice 3
Écrire un code Python qui :
- retourne un tableau de taille num ne contenant que des 0. ;
- retourne un tableau de taille num ne contenant que des 1. ;
- retourne un échantillonage uniforme de l'intervalle $[a, b]$ de taille n. Les éléments sont les $a + k\dfrac{b-a}{n-1}$ pour $k = 0, ..., n-1$;
- un vecteur de taille n dont les éléments sont tirés aléatoirement entre 0 et 1.

Matplotlib (A utiliser pour les exercices suivants)
Matplotlib est une bibliothèque Python qui permet de faire toutes sortes de tracés. Nous allons essentiellement l'utiliser pour tracer des fonctions. Les instructions de base pour se servir de Matplotlib sont illustrées ici :

```
import matplotlib.pyplot as plt # plt devient l'abrege de matplotlib.pyplot
x = [0, 1, 2]       # liste des abscisses
y = [1, -1, 0]      # liste des ordonnees
plt.plot(x, y)      # trace y en fonction de x
plt.show()      # affiche la fenetre du trace
```

Exercice 4 (Premiers tracés)
En utilisant les fonctions linspace et cos de NumPy, tracer la fonction $y = cos(x)$ sur $[0, 10\pi]$. Quelle est l'influence du nombre de points passé à la fonction linspace ? Grâce à un second appel à la fonction plt.plot avant l'appel à plt.show, superposer le graphe de la fonction $y = exp(-x/10)cos(x)$. Toujours avant l'appel à plt.show, ajouter un titre avec plt.title('Le titre de votre choix') et des noms aux axes avec plt.xlabel('x') et plt.ylabel ('$y = f(x)$').

Exercice 5 (Courbes paramétriques)
En modifiant le code ci-dessous :

```
import matplotlib.pyplot as plt
import numpy as np # np devient l'abrege de numpy
t = np.linspace(..., ..., ...)
x = ... # x(t)
y = ... # y(t)
plt.plot(x, y)
plt.show()
```

Tracer l'une des courbes suivantes :

— La Lemniscate de Bernoulli $\begin{cases} x(t) = \dfrac{sin(t)}{1+sin^2(t)} \\ y(t) = \dfrac{sin(t)cos(t)}{1+sin^2(t)} \end{cases}$ *sur* $[0,\ 2\pi]$.

— La spirale d'Archimède $\begin{cases} x(t) = tsin(t) \\ y(t) = tcos(t) \end{cases}$ *sur* $[0,\ 10\pi]$.

— La courbe du coeur $\begin{cases} x(t) = 16sin^3(t) \\ y(t) = 13cos(t) - 5cos(2t) - 2cos(3t) - cos(4t) \end{cases}$ *sur* $[0,\ 2\pi]$.

— Les cyclo-harmoniques $\begin{cases} x(t) = (1 + cos(\frac{p}{q}t))cos(t) \\ y(t) = (1 + cos(\frac{p}{q}t))sin(t) \end{cases}$ *pour p et q entiers sur* $[0,\ 2q\pi]$.

Exercice 6
Que fait le code suivant ? Expliquer.

```
def f(x) :
    return x ** 2
def g(x) :
    return 2 * x
def somme_ponderee(f1, f2, a, b, x) :
    return a * f1(x) + b * f2(x)
print(somme_ponderee(f, g, 2, 3, 10))
```

Exercice 7 (Méthode d'Euler)
On considère l'équation différentielle ordinaire (EDO) du premier ordre suivante

$$y'(x) = f(y(x), x), \qquad x \in [a,\ b]$$

On se propose d'étudier la méthode d'Euler qui consiste à approcher la solution en utilisant la formule de la tangente :

$$y(x+h) \simeq y(x) + hy'(x) \simeq y(x) + hf(y(x), x).$$

1. En utilisant cette dernière expression, donner l'équation de récurrence qui permet de calculer y_{n+1} en fonction de y_n, x_n, h et f.
2. Dans un fichier nommé edo.py, écrire une fonction euler(f, a, b, y0, N) qui retourne la liste des $y_n, n = 0, ..., N$ approchant la solution de notre EDO par la méthode d'Euler.
3. Quelle est la solution attendue pour $f(y, x) = \lambda y$ et $y(0) = 1$? Utiliser ce résultat pour tester l'implémentation de l'exercice précédent : tracer la solution exacte et la solution approchée avec Matplotlib. Quelle est l'influence du paramètre N ?

Exercice 8
Le module SciPy de Python est consacré au calcul scientifique avancé. En particulier, en utilisant la fonction scipy.integrate.odeint, il est possible d'approcher la solution d'une EDO grâce à des méthodes plus performantes (mais bien plus complexes aussi) que la méthode d'Euler. Il s'utilise comme ceci :

```
import scipy.integrate as scint
import numpy as np
import matplotlib.pyplot as plt
def f(y, x) :
        return np.cos(10 * y) * np.cos(x) # f(y(x), x) = cos(10y(x)) cos(x)
x = np.linspace(0, 10, 100)
y = scint.odeint(f, 0, x)
plt.plot(x, y)
plt.show()
```

Utiliser SciPy ainsi que la méthode d'Euler pour approcher la solution du problème de Cauchy associée à $f(y, x) = cos(10y)cos(x)$ et $y(0) = 0$. On prendra soin d'utiliser le même nombre de pas N dans les deux méthodes. Comparer visuellement les deux solutions à l'aide de Matplotlib.