2MA100 - Devoir Maison 1

Les exercices de ce devoir maison noté sur 20 points doivent être rendus au choix au format **Notebook** (`.ipynb`) ou **Script** (`.py`) en un ou plusieurs fichiers au plus tard le **11 mars 2022** à 20h00 sur Moodle.

Attention Plagiat.

Les devoirs maisons doivent être codés et rendus de manière individuelle. A titre d'information, voici une définition du plagiat adaptée du Memento de l'Université de Genève :

Le plagiat consiste à insérer, dans un travail académique, des formulations, des phrases, des passages, des morceaux de code, des images, de même que des idées ou analyses repris de travaux d'autres auteurs, en les faisant passer pour siens.

En particulier, le copier-coller à partir de sources trouvées sur Internet ou sur des travaux d'autres étudiant·es sans citer les sources est considéré comme du plagiat et implique une note zéro. Le plagiat constitue également une tentative de tricherie sanctionnée par le règlement de l'université. La solution est d'indiquer dans vos devoirs tout de ce qui ne vient pas de vous en mentionnant les sources (page Internet, livres, autre étudiant·e,...). Tous les fichiers rendus seront analysés automatiquement avec un logiciel de détection des similarités (entre étudiant·es et depuis Internet).

Les modules Numpy et Matplotlib peuvent être utiles dans certains exercices :

```
import numpy as np
import matplotlib.pyplot as plt
```

Exercice 1 : Formule d'Euler

On rappelle la formule suivante due à Euler :

$$\frac{\pi^2}{8} = \sum_{k=0}^{\infty} \frac{1}{(2k+1)^2},$$

et le but est de l'utiliser pour obtenir une approximation de π. Pour un entier $n \geq 0$, on note

$$S_n = \sum_{k=0}^{n} \frac{1}{(2k+1)^2}.$$

a) Écrire une fonction qui, à n donné, calcule S_n

b) Écrire une fonction qui, à une précision ε donnée, renvoie le plus petit entier n tel que $|\pi^2 - 8S_n| < \varepsilon$

c) Tracer l'évolution de $|\pi^2 - 8S_n|$ en fonction de n pour $3 \leq n \leq 100$.

Exercice 2 : Approximation de points par un cercle

Le but de cet exercice est d'utiliser l'algèbre linéaire pour trouver le cercle qui approxime le mieux un ensemble de points donnés.

Un cercle du plan est l'ensemble des solutions dans $\mathbb{R}^2$ d'une équation de la forme :

$$(x - x_0)^2 + (x - y_0)^2 = r^2$$

(dite équation réduite) pour $(x_0, y_0) \in \mathbb{R}^2$ (le centre) et $0 \leq r \in \mathbb{R}$ (le rayon). Elle est équivalente à l'équation :

$$x^2 + y^2 + ax + by + c = 0$$

(dite équation développée) où $a = -2x_0$, $b = -2y_0$ et $c = x_0^2 + y_0^2 - r^2$.

On identifie l'espace de tous les cercles avec $\mathbb{R}^3$ en utilisant les paramètres a, b, c de l'équation développée. (On accepte ainsi aussi des cercles sans solution réelle quand $r^2 = \frac{a^2}{4} + \frac{b^2}{4} - c < 0$.)

a) Écrire une fonction `centre_rayon(a,b,c)` qui retourne le centre et le rayon du cercle d'équation $x^2 + y^2 + ax + by + c = 0$.

b) Écrire une fonction `plot_cercle(a,b,c)` qui dessine le cercle d'équation $x^2+y^2+ax+by+c = 0$. Pour tester la fonction, dessiner les cercles d'équation $x^2 + y^2 = 1$, $(x - 1)^2 + y^2 = 1$, et $x^2 + (y - 1)^2 = 1$.

Indication : Il est commode de travailler avec l'équation paramétrique du cercle $(x_0+r\cos(\theta), y_0+r\sin(\theta))$ avec θ allant de 0 à 2π. Pour forcer les axes à avoir la même échelle, il est possible d'utiliser la fonction `matplotlib.pyplot.axis('equal')`

c) Si le point (p, q) appartient au cercle $x^2 + y^2 + ax + bx + c = 0$, alors il est une solution de l'équation $p^2 + q^2 + ap + bq + c = 0$. La condition $ap + bq + c = -p^2 - q^2$ est une équation affine sur (a, b, c), et donc l'ensemble de cercles qui passent par (p, q) est un sous-espace affine de $\mathbb{R}^3$. Déterminer l'équation du cercle qui passe par trois points (distincts, non colinéaires) revient à résoudre un système de trois équations.

Faire une fonction `cercle_par_points(X,Y)` qui retourne le centre et le rayon du cercle qui passe par les points `(X[0],Y[0])`, `(X[1],Y[1])`, et `(X[2],Y[2])`. Tester pour cinq triplettes de points aléatoires dans $[0, 1]^2$, en dessinant les points choisis et le cercle.

Indication : Pour résoudre le système linéaire, on pourra utiliser la fonction `numpy.linalg.solve`.

d) Si le nombre de points est supérieur à trois, alors le système n'a pas forcement de solution. La méthode des moindres carrés appliquée à un système d'équations linéaires $A\boldsymbol{x} = \boldsymbol{b}$ retourne une solution $\boldsymbol{x}$ qui minimise la norme euclidienne de $A\boldsymbol{x} - \boldsymbol{b}$.

Faire une fonction `cercle_moindres_carres(X,Y)` qui donnée une collection de points (x_i, y_i) retourne le cercle qui les approxime avec la méthode des moindres carrés.

Tester pour cinq ensembles de 15 points aléatoires dans $[0, 1]^2$, en dessinant les points choisis et le cercle.

Indication : Pour résoudre un système linéaire par la méthode des moindres carrés, utiliser la fonction `numpy.linalg.lstsq`.

e) [bonus] Faire une fonction `cercle_perturbe(x0,y0,r,N,sigma)` qui génère `N` points de la forme $(p+s, q+t)$ où (p, q) est un point aléatoire du cercle de centre `(x0,y0)` et rayon `r`, et s et t sont tirés de façon aléatoire en suivant une loi normale de moyenne 0 et déviation typique `sigma`.

Répéter le test précédent avec 5 ensembles de 15 points obtenus avec cette procédure avec paramètres `x0=0`, `y0=0`, `r=1` et `sigma=0.1`.

Indication : Pour générer des points aléatoires selon une loi normale, voir la fonction `numpy.random.normal`.

Exercice 3 : Egalité de Parseval

De manière simplifiée, l'égalité de Parseval, affirme que la norme d'un vecteur ou l'intégrale d'une fonction ne dépend pas de la représentation. L'objectif de cet exercice est de vérifier cette égalité de manière numérique dans le cas de la transformée Fourier, à la fois dans le cas discret (vecteur) et dans le cas continu (fonction périodique).

Des fonctions pour les transformées de Fourier discrètes sont disponibles dans le sous-module `numpy.fft` de NumPy. **Il est interdit de les utiliser pour répondre aux questions**. Il est en revanche autorisé de les utiliser afin de vérifier vos implémentations dans les questions correspondantes. Dans ce cas-là, il faut utiliser `norm='ortho'` dans les arguments de la fonction pour obtenir une normalisation orthogonale.

a) Écrire une fonction `dft_complex(x)` qui prend en argument un vecteur NumPy complexe `x` et qui renvoie sa transformée de Fourier discrète, c'est-à-dire qui prend en entrée un vecteur $\boldsymbol{x} = (x_0, \dots, x_{N-1}) \in \mathbb{C}^N$ et qui renvoie le vecteur $\boldsymbol{s} = (s_0, \dots, s_{N-1}) \in \mathbb{C}^N$ définit par :

$$s_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n \exp\left(-2i\pi k \frac{n}{N}\right).$$

b) Écrire une fonction `idft_complex(s)` qui prend en argument un vecteur NumPy complexe `s` et qui renvoie la transformation de Fourier discrète inverse, c'est-à-dire qui prend en entrée un vecteur $\boldsymbol{s} = (s_0, \dots, s_{N-1}) \in \mathbb{C}^N$ et qui renvoie le vecteur $\boldsymbol{x} = (x_0, \dots, x_{N-1}) \in \mathbb{C}^N$ définit par :

$$x_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} s_k \exp\left(2i\pi n \frac{k}{N}\right).$$

c) Vérifier que les fonctions `dft_complex` et `idft_complex` sont bien l'inverse l'une de l'autre. Pour cela, prendre $N = 100$, générer aléatoirement un vecteur complexe `x` et vérifier que les distances euclidiennes deux-à-deux entre les vecteurs `x`, `idft_complex(dft_complex(x))` et `dft_complex(idft_complex(x))` sont petites.

d) L'égalité de Parseval indique que la norme euclidienne d'un vecteur $\boldsymbol{x}$ est égale à la norme euclidienne de sa transformée de Fourier discrète $\boldsymbol{s}$:

$$\|\boldsymbol{x}\|^2 = \sum_{n=0}^{N-1} |x_n|^2 = \sum_{k=0}^{N-1} |s_k|^2 = \|\boldsymbol{s}\|^2.$$

Vérifier cette égalité en comparant la différence (en valeur absolue) entre la norme euclidienne de `x`, et la norme euclidienne de `dft_complex(x)` pour `x` un vecteur complexe généré aléatoirement.

e) On se place maintenant dans le cas continu avec une fonction T-périodique $f : \mathbb{R} \to \mathbb{R}$, dont les coefficients de Fourier réels sont donnés par :

$$a_0 = \frac{1}{T} \int_{-T/2}^{T/2} f(t)\ \mathrm{d}t,$$

$$a_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos \frac{2\pi n t}{T}\ \mathrm{d}t,$$

$$b_n = \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin \frac{2\pi n t}{T}\ \mathrm{d}t,$$

pour $n \geq 1$.

Écrire une fonction `ft_real` qui prend en argument une fonction f, une période T et un entier N et qui renvoie le réel a_0 et les vecteurs $(a_1, \dots, a_N)$ et $(b_1, \dots, b_N)$.

Indication : Pour calculer chaque intégrale, on pourra utiliser la fonction `scipy.integrate.quad` avec l'argument `limit=500` pour améliorer la précision du calcul de l'intégrale. Par exemple pour calculer l'intégrale de la fonction $f : x \mapsto x^2 - kx$ entre $a = -8$ et $b = 12$ pour $k = 2$:

```
from scipy.integrate import quad

def integrand(x, k):
    return x ** 2 - k * x

quad(func=integrand, a=-8, b=12, args=2, limit=500)[0]
```

f) Soit f la fonction 2-périodique définie pour $x \in [-1, 1]$ par :

$$f(x) = \exp(x) + \exp(-x) + x^3 - x$$

et sinon par :

$$f(x) = f(x + 2k)$$

avec $k \in \mathbb{N}$ tel que $x + 2k \in [-1, 1]$. Écrire une fonction Python `f` qui prend en argument un vecteur NumPy et qui renvoie un vecteur NumPy de même taille avec la valeur $f(x)$ pour chaque x dans le tableau d'entrée. Afficher sur un graphique les valeurs renvoyées par votre fonction `f` sur l'intervalle $[-5, 5]$.

Indication : On pourra utiliser la fonction `numpy.remainder` pour obtenir le reste de la division euclidienne.

g) Vérifier l'égalité de Parseval :

$$\frac{1}{T}\int_{-T/2}^{T/2} |f(t)|^2 \, \mathrm{d}t = a_0^2 + \frac{1}{2}\sum_{n=1}^{\infty} \left(a_n^2 + b_n^2\right)$$

pour la fonction f définie à la question précédente. On se limitera aux $N = 100$ premières valeurs des suites $(a_n)_n$ et $(b_n)_n$.