Les barèmes sont indicatifs et pourront être reconsidérés.

Méthode de Simpson (26 points)

Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction continue. La méthode de Simpson est une méthode pour numériquement calculer $I_{a,b}^{(\text{Simpson})}(f)$ une approximation de $\int_a^b f(t)dt$, avec la formule

$$I_{a,b}^{(\text{Simpson})}(f) = \frac{b-a}{6}(f(a) + f(b) + 4f(\frac{a+b}{2}))$$

On rappelle qu'une méthode d'intégration numérique est d'ordre k si elle fournit un résultat exact lorsque f est un polynôme de degré inférieur ou égal à k.

1. (3 points) Montrer que la méthode de Simpson n'est pas d'ordre 4.

2. (5 points) Montrer que la méthode de Simpson est d'ordre au moins 2.

3. (2 points) Montrer que la méthode de Simpson donne un résultat exact pour la fonction $P_{a,b} : x \mapsto (x - \frac{a+b}{2})^3$.

On admet le résultat suivant : pour tout $P \in \mathbb{R}_3[X]$ polynôme de degré au plus 3, il existe $\alpha \in \mathbb{R}, Q \in \mathbb{R}_2[X]$ tel que

$$P = \alpha P_{a,b} + Q$$

4. (5 points) En utilisant la linéarité de l'intégrale, déduire des questions précédentes que la méthode de Simpson est d'ordre 3.

5. a) (3 points) Implémenter une fonction `simpson(f, a, b)` qui calcule l'approximation de $\int_a^b f$ donnée par la méthode de Simpson.

b) (5 points) En déduire une fonction `simpson_composite(f, a, b, n)` qui calcule la généralisation composite de la méthode de Simpson, c'est-à-dire qui calcule

$$I_{a,b,n}^{(\text{Simpson})}(f) = \sum_{k=0}^{n-1} I_{x_k,x_{k+1}}^{(Simpson)}(f)$$

où $(x_0, \ldots, x_n)$ forme une subdivision régulièrement espacée de l'intervalle $[a, b]$.

On rappelle que la fonction `np.linspace(a, b, n)` fournit une telle subdivision.

6. (3 points) Rappeler l'ordre de la méthode des rectangles. En général, faut-il préférer la méthode des rectangles ou la méthode de Simpson ? Expliquer la différence entre les deux méthodes que l'on observe expérimentalement qui justifie cette préférence.

Gymnastique Python (19 points)

Les exercices de cette section sont indépendants, et testent votre maîtrise générale de Python.

7. a) (3 points) Écrire une fonction `trouve_minimum(L)` qui prend en argument une liste non vide d'entiers `L` et renvoie le plus petit élément de `L`. On se refusera d'utiliser la fonction Python `min`.

b) (3 points) Modifier la fonction pour obtenir `trouve_indice_du_minimum(L)` que renvoie *l'indice* du plus petit élément de `L`. S'il y a plusieurs indices où `L` atteint son minimum, vous êtes libres de choisir quel indice est renvoyé.

8. (3 points) Écrire une fonction `sous_liste_pair(L)` qui prend en argument une liste d'entiers `L` et renvoie la liste composée des entiers pairs de `L`. Par exemple, `sous_liste_pair([1, 3, 2, 8, 5, 5, 6])` devra renvoyer `[2, 8, 6]`.

9. (3 points) Écrire une fonction `applati(L)` qui prend en argument une liste de listes `L` et renvoie la liste qui met bout à bout les sous-listes de `L`. Par exemple, `applati([[1, 2, 3], [2, 3, 4], [3, 4, 5]])` devra renvoyer `[1, 2, 3, 2, 3, 4, 3, 4, 5]`.

10. (1 point) À partir d'une liste `L` et d'un élément `a`, comment obtenir la liste où l'on a ajouté `a` à la fin de `L`.

11. a) (1 point) Que vaut `True or False` ?

b) (1 point) Que vaut `True or True and False` ?

c) (1 point) Que vaut `(True or True) and False` ?

12. a) (1 point) Qu'affiche le programme suivant ?

```
L = [1, 3, 5, 7]
print(L[1:2])
```

b) (1 point) Qu'affiche le programme suivant ?

```
s = "bcpst"
print(s[1:])
```

c) (1 point) Qu'affiche le programme suivant ?

```
for i in range(2, 6, 2):
    print(i)
```