



Enseignant : CD Petrescu

ICS - MAN

6 juillet 2023

Durée : 150 minutes

# Adam Barthélémy Steven

SCIPER: 355754

Attendez le début de l'épreuve avant de tourner la page. Ce document est imprimé recto-verso, il contient 20 pages (les dernières pouvant être vides) et 13 questions. Ne pas dégrafer.

### Indications

- Posez votre carte d'étudiant sur la table.
  - L'utilisation d'une **calculatrice** et de tout outil électronique est interdite pendant l'épreuve.
  - Pour les questions à **choix multiple**, on comptera les points indiqués si la réponse est correcte ou 0 point autrement (par exemple s'il n'y a aucune ou plus d'une réponse inscrite ou si la réponse est incorrecte).
  - Utilisez un **stylo** à encre **noire ou bleu foncé** et effacez proprement avec du **correcteur blanc** si nécessaire. Toute réponse doit être rédigée en utilisant la place réservée à cet effet à la suite de la question. N'écrivez **pas dans les marges** !
  - Si une question est erronée, l'enseignant se réserve le droit de l'annuler.
  - Les brouillons sont à rendre mais ne seront pas corrigés.
  - A part le polycopié "Informatique et calcul scientifique (ICS - MAN) Première partie Programmation Python" imprimé au Centre d'impression EPFL, **aucun** autre document n'est autorisé et ne doit donc être consulté durant l'examen.
  - Le polycopié mentionné peut être annoté mais les notes ajoutées doivent correspondre seulement à la première partie "Programmation Python" du cours.
  - Le polycopié mentionné ne doit contenir aucune feuille supplémentaire ajoutée d'une façon quelconque au document imprimé au centre d'impression EPFL.

Respectez les consignes suivantes   Observe this guidelines   Beachten Sie bitte die unten stehenden Richtlinien		
choisir une réponse   select an answer Antwort auswählen	ne PAS choisir une réponse   NOT select an answer NICHT Antwort auswählen	Corriger une réponse   Correct an answer Antwort korrigieren
  		 
<hr/>		
ce qu'il ne faut <b>PAS</b> faire   what should <b>NOT</b> be done   was man <b>NICHT</b> tun sollte		
     		



## Première partie : trois questions de type ouvert indépendantes

Répondre dans l'espace dédié.

Laisser libres les cases à cocher : elles sont réservées au correcteur.

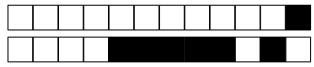
**Question 1:** *Cette question est notée sur 13 points.*

<input type="checkbox"/>	.5	<input type="checkbox"/>	.5	<input type="checkbox"/>	.5																					
<input type="checkbox"/> 0		<input type="checkbox"/> 1		<input type="checkbox"/> 2		<input type="checkbox"/> 3		<input type="checkbox"/> 4		<input type="checkbox"/> 5		<input type="checkbox"/> 6		<input type="checkbox"/> 7		<input type="checkbox"/> 8		<input type="checkbox"/> 9		<input type="checkbox"/> 10		<input type="checkbox"/> 11		<input type="checkbox"/> 12		<input type="checkbox"/> 13

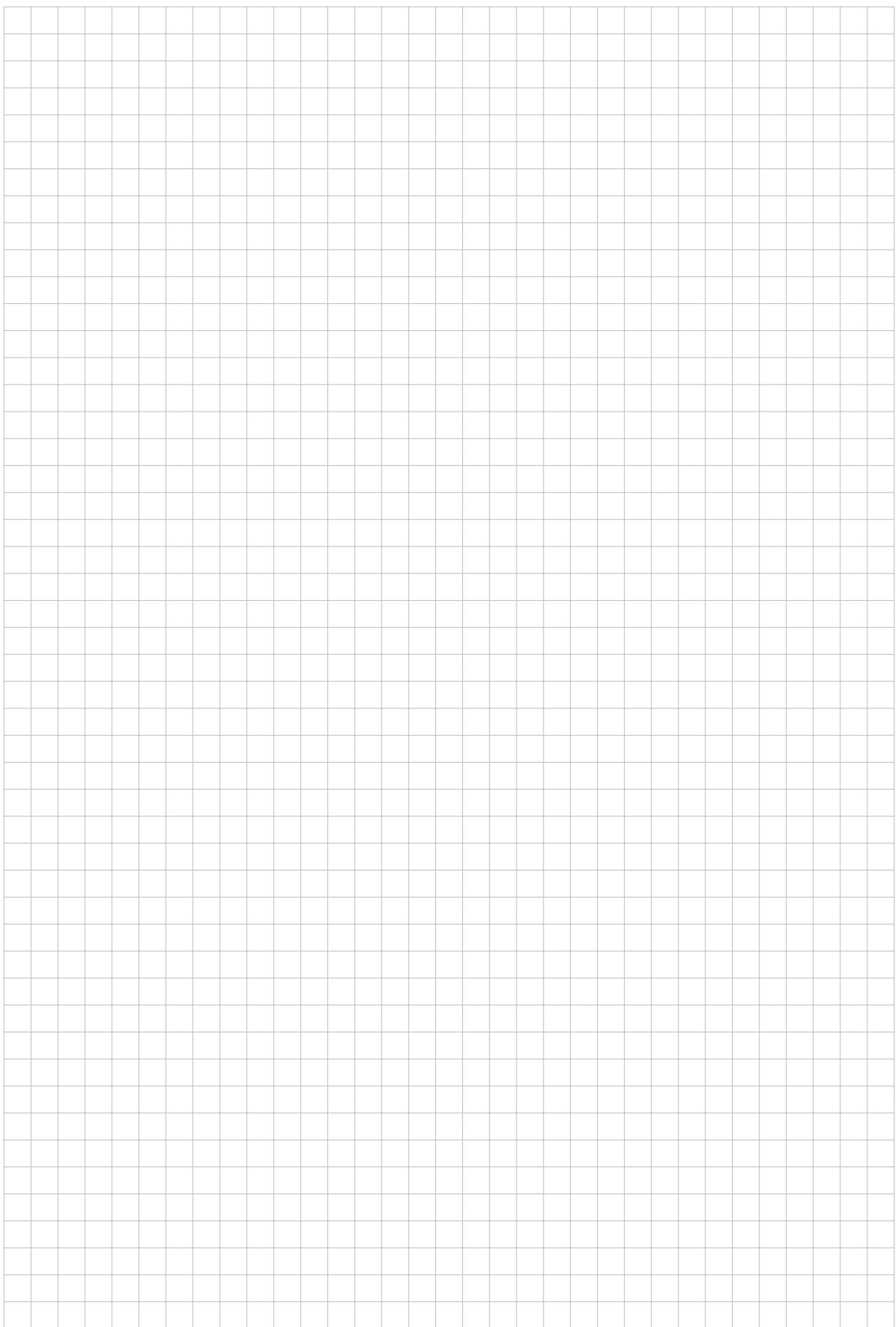
On donne ci-dessous le contenu d'une cellule Jupyter Notebook.

```
import numpy as np
x = np.array( [ 1, 3, 6, 10, 15 ] )
print( x + 1 )
print( x[ 2: ] - x[ :-2 ] )
print( x < 6 )
print( x[ x < 6 ] - x[ x > 6 ] )
print('*****')
tab = np.array( [ [ 1, 2, 3, 4 ], [ 12, 13, 14, 15 ], [ 5, 6, 7, 8 ] ] )
print( tab.shape )
print( tab[ 2 ] )
print( tab[ 1, 2 ] )
print( tab[ tab.shape[ 0 ] - 1, tab.shape[ 1 ] - 2 ] )
print('*****')
print( tab[ 0 ]**2 )
print( tab[ 1 ] - tab[ 2 ] )
print( tab[ :, 3 ] )
print( tab[ 1:2, 3:0:-2 ] )
```

Précisez **à la page suivante** les messages affichés suite à l'exécution de la cellule indiquée plus haut.



+1/3/58+

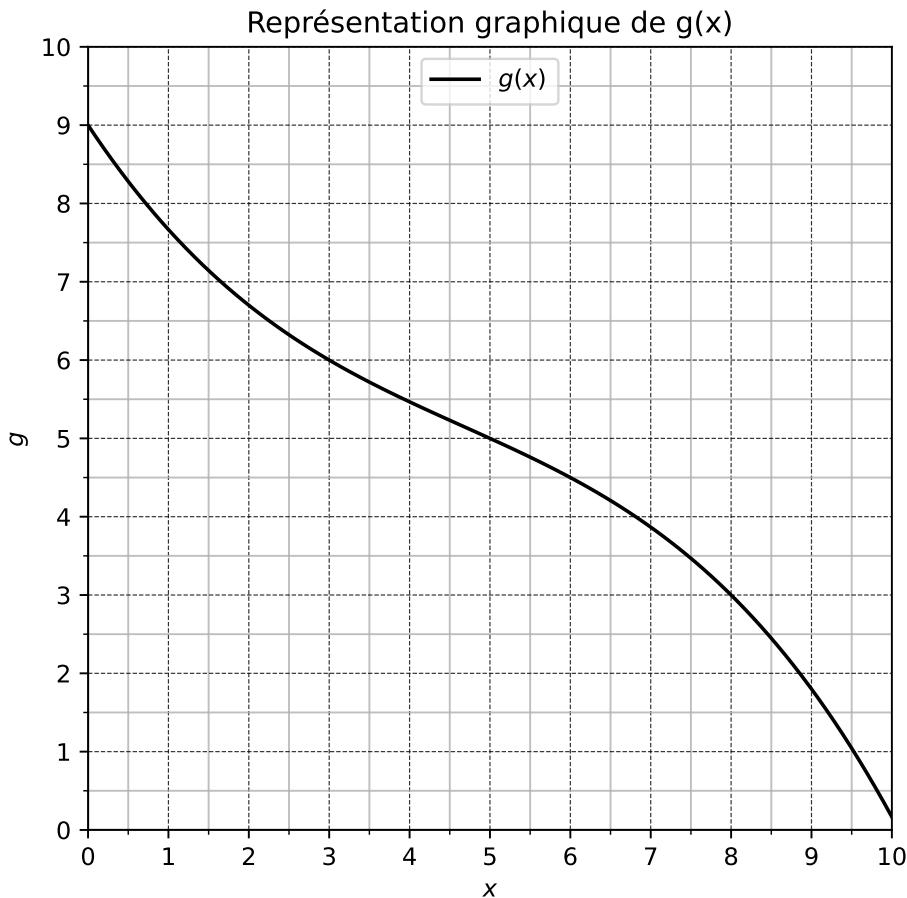




**Question 2:** Cette question est notée sur 14 points.

<input type="text"/> .5														
<input type="text"/> 0	<input type="text"/> 1	<input type="text"/> 2	<input type="text"/> 3	<input type="text"/> 4	<input type="text"/> 5	<input type="text"/> 6	<input type="text"/> 7	<input type="text"/> 8	<input type="text"/> 9	<input type="text"/> 10	<input type="text"/> 11	<input type="text"/> 12	<input type="text"/> 13	<input type="text"/> 14

Soit une fonction (d'itération polynomiale) réelle d'une variable réelle  $g : \mathbb{R} \rightarrow \mathbb{R}$  dont la représentation graphique sur l'intervalle  $[0, 10]$  est donnée dans la figure ci-dessous.



Vous pouvez dessiner et/ou écrire sur cette figure mais les notes que vous y ajouterez ne seront pas corrigées.

Par la suite, afin de répondre aux questions posées, vous devez :

- utiliser certaines valeurs numériques lues directement sur le graphique mentionné ;
- n'écrire aucune ligne de code (c'est-à-dire n'écrire aucun programme destiné à être exécuté par un ordinateur) ;
- écrire vos réponses aux endroits prévus à cet effet (sans justifications et sans détails de calcul).

Passez à la page suivante, s'il vous plaît.



En étudiant cette figure, on peut constater que la fonction  $g(x)$  a un (seul) point fixe (noté par la suite  $\bar{x}$ ) dans l'intervalle mentionné  $[0, 10]$ .

a) Afin de trouver des valeurs approchées de ce point fixe, vous devez utiliser la **méthode** itérative de point fixe de **Picard** en choisissant la valeur de départ  $x_0 = 8$ .

On considère les premières 5 itérations de la méthode de Picard (y compris la première itération d'indice 0).

En utilisant la figure donnée, déterminez et précisez ci-dessous (aux endroits prévus à cet effet) les valeurs approchées successives  $x_i$ , où  $i \in \{1, 2, 3, 4\}$ , du point fixe  $\bar{x}$  obtenues avec la méthode de Picard.

$$x_0 = 8$$

$$x_1 = \dots$$

$$x_2 = \dots$$

$$x_3 = \dots$$

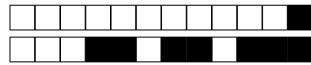
$$x_4 = \dots$$

b) Pour le cas de l'approche itérative entamée au point a), précisez la nature du point fixe (en cochant la case correspondante).

- Le point fixe est attracteur.
- Le point fixe est répulsif.

c) Indiquez ci-dessous la valeur (exacte) du point fixe  $\bar{x}$ .

$$\bar{x} = \dots$$

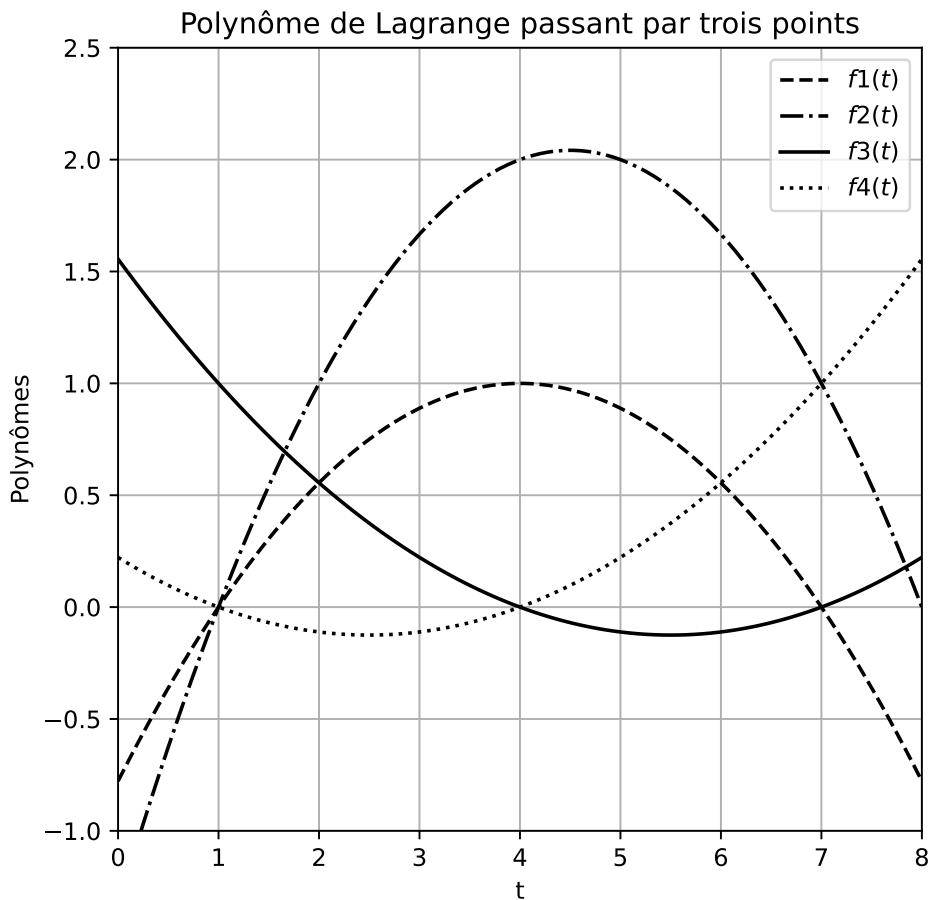


**Question 3:** Cette question est notée sur 14 points.

<input type="checkbox"/>														
.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Soient trois points  $P_0(t_0, p_0)$ ,  $P_1(t_1, p_1)$  et  $P_2(t_2, p_2)$  où  $0 < t_0 < t_1 < t_2 < 8$ . On calcule les polynômes de la base de Lagrange associée à ces trois points, à savoir  $\varphi_0(t)$ ,  $\varphi_1(t)$  et  $\varphi_2(t)$ , ainsi que le polynôme d'interpolation de Lagrange  $p(t)$  qui passe par ces trois points.

Les graphiques de ces quatre polynômes ont été représentés dans la figure ci-dessous et les courbes correspondantes ont été notées (dans un ordre que vous devez préciser au point c)) par  $f_1$ ,  $f_2$ ,  $f_3$  et  $f_4$ .



Vous pouvez dessiner et/ou écrire sur cette figure mais les notes que vous y ajouterez ne seront pas corrigées.

a) Précisez ci-dessous (aux endroits prévus à cet effet) les valeurs des abscisses des points  $P_0$ ,  $P_1$  et  $P_2$ .

$$t_0 = \dots$$

$$t_1 = \dots$$

$$t_2 = \dots$$



b) Précisez ci-dessous (aux endroits prévus à cet effet) les valeurs des ordonnées des points  $P_0$ ,  $P_1$  et  $P_2$ .

$$p_0 = \dots$$

$$p_1 = \dots$$

$$p_2 = \dots$$

c) Indiquez ci-dessous (aux endroits prévus à cet effet) la correspondance entre les quatre polynômes  $\varphi_0(t)$ ,  $\varphi_1(t)$ ,  $\varphi_2(t)$ ,  $p(t)$  et les quatre courbes  $f_1$ ,  $f_2$ ,  $f_3$ ,  $f_4$  représentées dans la figure ci-dessus.

$\varphi_0(t)$  correspond à la courbe .....

$\varphi_1(t)$  correspond à la courbe .....

$\varphi_2(t)$  correspond à la courbe .....

$p(t)$  correspond à la courbe .....



## Deuxième partie - deux questions de type ouvert se rapportant au même énoncé

Répondre dans l'espace dédié. Laisser libres les cases à cocher : elles sont réservées au correcteur.

Les deux questions suivantes se rapportent au même énoncé. Vous pouvez répondre à la deuxième question avant de répondre à la première question.

### Énoncé

Soit une fonction réelle d'une variable réelle  $f(x) : \mathbb{R} \rightarrow \mathbb{R}$  qui est intégrable.

Afin de calculer numériquement l'intégrale définie de  $f$  sur l'intervalle  $[a, b]$ , à savoir  $\int_a^b f(x)dx$ , on utilise l'approche de Newton-Cotes et on choisit  $m + 1 = 4$  points de quadrature équidistants. On obtient ainsi la formule de **quadrature non composite** suivante, appelée formule des  $\frac{3}{8}$  de Newton :

$$\begin{aligned} J_i(f) &= (x_{i+1} - x_i) \left( \frac{1}{8}f(x_i) + \frac{3}{8}f\left(x_i + \frac{1}{3}(x_{i+1} - x_i)\right) \right. \\ &\quad \left. + \frac{3}{8}f\left(x_i + \frac{2}{3}(x_{i+1} - x_i)\right) + \frac{1}{8}f(x_{i+1}) \right) \end{aligned}$$

où  $[x_i, x_{i+1}]$  est un sous-intervalle quelconque de  $[a, b]$  et  $J_i(f)$  est la valeur approchée de l'intégrale  $\int_{x_i}^{x_{i+1}} f(x)dx$ .

On note par la suite  $I_{reg}(f)$  la valeur approchée de l'intégrale  $\int_a^b f(x)dx$  calculée numériquement à l'aide d'une formule de **quadrature composite** obtenue à partir de la formule de quadrature non composite ci-dessus et pour une **division régulière** de l'intervalle d'intégration  $[a, b]$ .

**Question 4:** Cette question est notée sur 12 points.

<input type="text"/> 0	<input type="text"/> 1	<input type="text"/> 2	<input type="text"/> 3	<input type="text"/> 4	<input type="text"/> 5	<input type="text"/> 6	<input type="text"/> 7	<input type="text"/> 8	<input type="text"/> 9	<input type="text"/> 10	<input type="text"/> 11	<input type="text"/> 12
<input type="text/&gt;.5"/>												

- a) Soit  $r$  le degré d'exactitude de la formule de quadrature non composite des  $\frac{3}{8}$  de Newton mentionnée dans l'énoncé (c'est-à-dire le degré le plus élevé des polynômes intégrables exactement par cette formule). Précisez ci-dessous la valeur de  $r$ .

$r = \dots$

- b) On considère le calcul numérique d'une intégrale définie à l'aide de la formule de quadrature composite basée sur la formule de quadrature non composite mentionnée dans l'énoncé.

L'erreur absolue est alors :  $e_{abs} = \left| \int_a^b f(x)dx - I_{reg}(f) \right|$ .

Si on passe d'une partition régulière avec  $n$  sous-intervalles à une partition régulière plus fine avec  $4n$  sous-intervalles (c'est-à-dire si on divise le pas de la partition régulière par 4), alors l'erreur absolue sera divisée par  $k$ .

Précisez ci-dessous la valeur de  $k$ .

$k = \dots$



**Question 5:** Cette question est notée sur 19 points.

<input type="text"/> .5																
<input type="text"/> 0	<input type="text"/> 1	<input type="text"/> 2	<input type="text"/> 3	<input type="text"/> 4	<input type="text"/> 5	<input type="text"/> 6	<input type="text"/> 7	<input type="text"/> 8	<input type="text"/> 9	<input type="text"/> 10	<input type="text"/> 11	<input type="text"/> 12	<input type="text"/> 13	<input type="text"/> 14	<input type="text"/> 15	

On vous demande d'écrire une fonction Python nommée ***quadrature()*** et qui calcule numériquement la valeur approchée d'une intégrale définie  $\int_a^b f(x)dx$  à l'aide de la formule de **quadrature composite** obtenue à partir de la formule de quadrature non composite des  $\frac{3}{8}$  de Newton mentionnée dans l'énoncé.

La valeur approchée doit être calculée pour une **division régulière** de l'intervalle d'intégration  $[a, b]$  en un nombre convenable de sous-intervalles  $n$  afin de respecter une tolérance relative imposée.

Plus précisément, la fonction Python ***quadrature()*** doit :

\* avoir 4 paramètres :

- ***f*** : la fonction à intégrer ;
- ***a*** : la limite gauche de l'intervalle d'intégration ;
- ***b*** : la limite droite de l'intervalle d'intégration ;
- ***EPS\_REL*** : la tolérance relative imposée ;

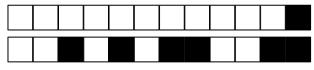
\* retourner :

- ***I\_new*** : la valeur approchée de l'intégrale définie ;
- ***n*** : le nombre de sous-intervalles utilisés afin d'assurer la tolérance relative imposée.

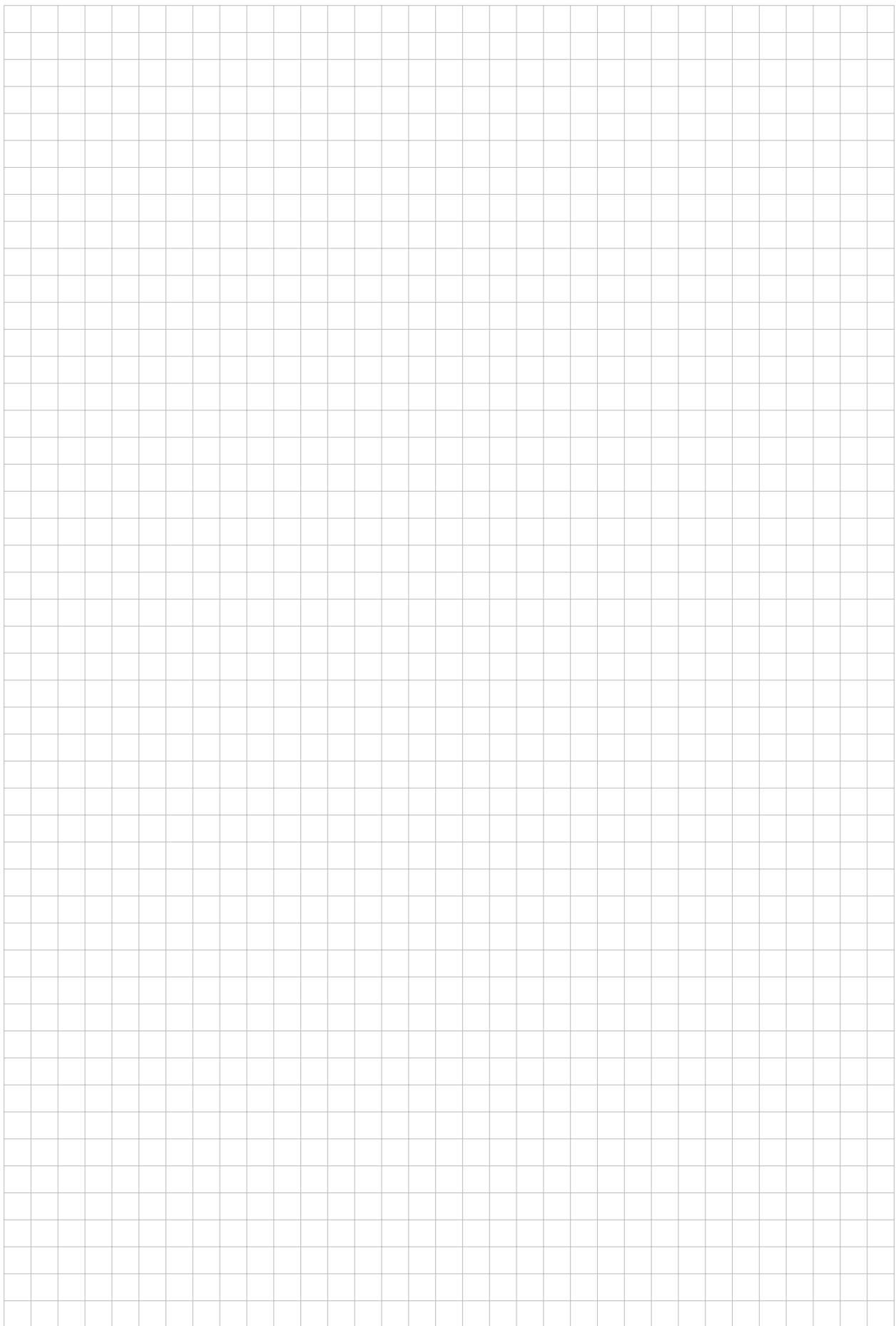
**Indication** : afin d'écrire la fonction Python demandée, on peut prévoir (comme nous l'avons fait aux TP pour la formule de Simpson) un calcul itératif (englobant) de la manière suivante :

- pour commencer, on utilise la formule de quadrature non composite pour l'intervalle d'intégration global (c'est-à-dire, au début, il n'y a qu'un seul "sous-intervalle") ;
- à chaque nouvelle itération, on double le nombre de sous-intervalles de la partition régulière antérieure et on applique la formule de quadrature composite ;
- finalement, on arrête le calcul itératif (englobant) quand une certaine condition d'arrêt (qui fait intervenir la tolérance relative) est remplie.

Écrivez à la page suivante la définition (l'en-tête et le corps) de la fonction Python ***quadrature()***.

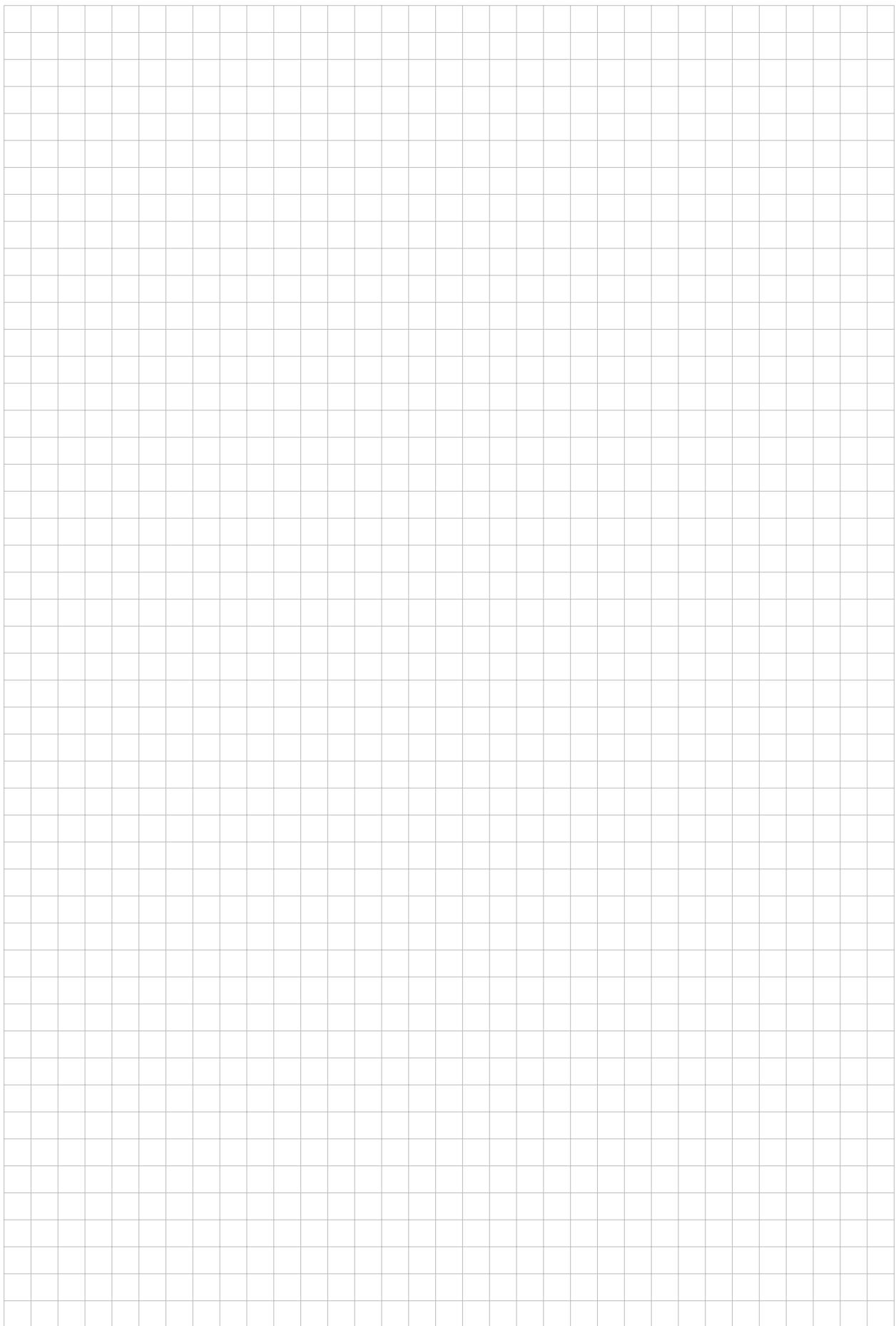


+1/10/51+





+1/11/50+





### Troisième partie, questions à choix unique

Pour chaque question, marquer la case correspondante à la réponse correcte sans faire de ratures.  
Il n'y a qu'**une seule réponse correcte** par question.

#### Question 6 (6 points)

On donne ci-dessous le contenu d'une cellule Jupyter Notebook.

```
i = 1

while i < 99:
    i *= 3
    if i % 9:
        print('Suite', end = ' ')
        continue
    if not (i % 27):
        print('Arrêt', end = ' ')
        break
    if not (i % 999):
        print('Passe', end = ' ')
        pass
    print('Fin', end = ' ')
else:
    print('Autrement', end = ' ')
    print(i, end = ' ')
```

Cochez la case qui correspond à l'affichage obtenu suite à l'exécution de la cellule ci-dessus.

- Suite Fin Arrêt Autrement 27
- Suite Fin Arrêt
- Suite Arrêt Autrement 9
- Suite Arrêt Passe Fin

#### Question 7 (6 points)

Cochez la case qui correspond à l'instruction qui n'affiche pas de message d'erreur à l'exécution.

- `d = {{1}, {2}, {3}}`
- `d = {[1]:{1}, [2]:{2}, [3]:{3}}`
- `d = {{1}:[1], {2}:[2], {3}:[3]}`
- `d = {1:1, 2:2, 3:3}`

**Question 8** (6 points)

On donne ci-dessous le contenu d'une cellule Jupyter Notebook.

```
x = (-1, 1, 3, 5)  
y = [ 2, 4, 6, 8 ]  
print( [ [ elx, ely ] for elx in x for ely in y if elx + ely > 7 ] )
```

Cochez la case qui correspond à l'affichage obtenu suite à l'exécution de la cellule ci-dessus.

- [ [ 3, 6], [5, 8] ]
- [ [1, 8], [3, 8], [5, 8] ]
- [ [1, 8], [3, 6], [3, 8], [5, 4], [5, 6], [5, 8] ]
- [ [-1, 8], [1, 8], [3, 8], [5, 8] ]

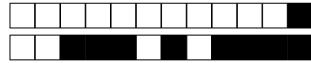
**Question 9** (6 points)

On donne ci-dessous le contenu d'une cellule Jupyter Notebook.

```
from copy import deepcopy  
  
def combiner( li1, li2 ):  
  
    lis = deepcopy( li1 )  
  
    for i in range( len( li1 ) ):  
        li1[ i ].append( li2[ -i-1 ] )  
        lis.append( li1[ i ] )  
  
print( li1, end = ' *** ' )  
print( lis )  
  
combiner( [ [ 1 ], [ 2 ] ], [ 'a', 'b' ] )
```

Cochez la case qui correspond à l'affichage obtenu suite à l'exécution de la cellule ci-dessus.

- [[1], [2]] \*\*\* [[1, 'b'], [2, 'a'], [1, 'b'], [2, 'a']]
- [[1, 'b'], [2, 'a']] \*\*\* [[1, 'b'], [2, 'a'], [1, 'b'], [2, 'a']]
- [[1, 'b'], [2, 'a']] \*\*\* [[1], [2], [1, 'b'], [2, 'a']]
- [[1], [2]] \*\*\* [[1], [2], [1, 'b'], [2, 'a']]

**Question 10** (6 points)

Durant l'étude expérimentale d'un objet en mouvement rectiligne, on a mesuré la position de ce mobile à différents moments. Ces résultats ont été enregistrés sous la forme de deux colonnes (sans titre) dans un fichier texte nommé **mesures.txt** et qui ne contient aucune autre information. La première colonne contient les valeurs du temps et la deuxième colonne contient les positions correspondantes du mobile.

Vous avez ci-dessous les contenus de quatre cellules Jupyter Notebook qui s'exécutent sans erreur et qui affichent des graphiques obtenus à partir du fichier texte mentionné. Cochez la case qui correspond à la cellule qui **ne représente pas correctement** la position du mobile en fonction du temps.

```
import numpy as np; import matplotlib.pyplot as plt  
res = np.loadtxt( 'mesures.txt', usecols = ( 0, 1 ), unpack = False )  
plt.plot( res[ 0 ], res[ 1 ], 'bo ', label = "Données expérimentales" )  
plt.show()
```

```
import numpy as np; import matplotlib.pyplot as plt  
res0, res1 = np.loadtxt( 'mesures.txt', usecols = ( 0, 1 ), unpack = True)  
plt.plot( res0, res1, 'bo ', label = "Données expérimentales" )  
plt.show()
```

```
import numpy as np; import matplotlib.pyplot as plt  
res = np.loadtxt( 'mesures.txt', usecols = ( 0, 1 ), unpack = False )  
plt.plot( res[ :, 0 ], res[ :, 1 ], 'bo ', label = "Données expérimentales" )  
plt.show()
```

```
import numpy as np; import matplotlib.pyplot as plt  
res = np.loadtxt( 'mesures.txt', usecols = ( 0, 1 ), unpack = True )  
plt.plot( res[ 0 ], res[ 1 ], 'bo ', label = "Données expérimentales" )  
plt.show()
```

**Question 11** (6 points)

Soit la fonction réelle d'une variable réelle  $f : \mathbb{R} \rightarrow \mathbb{R}$  définie par  $f(x) = x^2 - 1$ . Afin d'utiliser une méthode de point fixe pour trouver les zéros de la fonction  $f$ , on transforme l'équation  $f(x) = 0$  en une **équation équivalente** de la forme  $g(x) = x$  de sorte que l'ensemble des points fixes de  $g$  est le même que l'ensemble des zéros de  $f$ . La fonction  $g$  est appelée couramment **fonction d'itération**.

Cochez la case qui correspond à la seule fonction d'itération  $g$  pour laquelle la recherche des points fixes n'est **pas équivalente** à la recherche des zéros de la fonction  $f$  donnée.

$g(x) = x^4 + x - 1$

$g(x) = -x^2 + x + 1$

$g(x) = x^4 - x - 1$

$g(x) = x^2 + x - 1$

**Question 12** (6 points)

Soit le tableau de Butcher ci-dessous.

0	0	0	0	0
1/3	1/3	0	0	0
2/3	-1/3	1	0	0
1	1	-1	1	0
	1/8	3/8	3/8	1/8

Parmi les schémas numériques de Runge-Kutta donnés ci-dessous (avec les notations habituelles), choisissez celui qui correspond au tableau de Butcher ci-dessus et cochez la case respective.

$$u_{n+1} = u_n + h \left( \frac{K_2}{3} + \frac{2K_3}{3} + K_4 \right)$$

où :

$$K_1 = f \left( t_n + \frac{h}{8}, u_n \right)$$

$$K_2 = f \left( t_n + \frac{3h}{8}, u_n + h \frac{K_1}{3} \right)$$

$$K_3 = f \left( t_n + \frac{3h}{8}, u_n + h \left( -\frac{K_1}{3} + K_2 \right) \right)$$

$$K_4 = f \left( t_n + \frac{h}{8}, u_n + h(K_1 - K_2 + K_3) \right)$$

$$u_{n+1} = u_n + h \left( \frac{K_2}{3} + \frac{2K_3}{3} + K_4 \right)$$

où :

$$K_1 = f \left( t_n + \frac{h}{8}, u_n \right)$$

$$K_2 = f \left( t_n + \frac{3h}{8}, u_n + h \frac{K_1}{3} \right)$$

$$K_3 = f \left( t_n + \frac{3h}{8}, u_n + h \left( -\frac{K_1}{3} + K_2 \right) \right)$$

$$K_4 = f \left( t_n + \frac{h}{8}, u_n + h(K_2 - K_3 + K_4) \right)$$

$$u_{n+1} = u_n + \frac{h}{8} (K_1 + 3K_2 + 3K_3 + K_4)$$

où :

$$K_1 = f(t_n, u_n)$$

$$K_2 = f \left( t_n + \frac{h}{3}, u_n + h \frac{K_1}{3} \right)$$

$$K_3 = f \left( t_n + \frac{2h}{3}, u_n + h \left( -\frac{K_1}{3} + K_2 \right) \right)$$

$$K_4 = f(t_n + h, u_n + h(K_1 - K_2 + K_3))$$

$$u_{n+1} = u_n + \frac{h}{8} (K_1 + 3K_2 + 3K_3 + K_4)$$

où :

$$K_1 = f(t_n, u_n)$$

$$K_2 = f \left( t_n + \frac{h}{3}, u_n + h \frac{K_1}{3} \right)$$

$$K_3 = f \left( t_n + \frac{2h}{3}, u_n + h \left( -\frac{K_1}{3} + K_2 \right) \right)$$

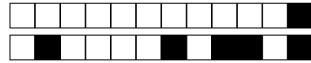
$$K_4 = f(t_n + h, u_n + h(K_2 - K_3 + K_4))$$

**Question 13** (6 points)

Les affirmations suivantes concernent la résolution numérique du problème de Cauchy (pour des équations différentielles ordinaires du premier ordre).

Parmi les affirmations ci-dessous, précisez l'**affirmation** qui est **fausse** en cochant la case correspondante.

- La consistance (d'un schéma numérique) est une condition nécessaire pour la convergence (de ce schéma numérique).
- La méthode d'Euler progressive est convergente d'ordre 1 en  $h$ , où  $h$  est le pas de la discréttisation.
- Les erreurs dites de troncature sont liées à la façon de stocker l'information, en général, et les valeurs numériques, en particulier, dans la mémoire de l'ordinateur.
- La méthode d'Euler rétrograde est inconditionnellement stable.



## Formulaire

### Bibliothèque Python NumPy

```
import numpy as np

np.array(object, dtype=None)
np.linspace(start, stop, num=50, endpoint=True, retstep=False)
np.logspace(start, stop, num=50, endpoint=True)
np.arange(start, stop, step)
np.zeros(shape, dtype=float)
np.ones(shape, dtype=None)
np.empty(shape, dtype=float)
np.zeros_like(a, dtype=None)
np.ones_like(a, dtype=None)
np.empty_like(a, dtype=None)
ndarray.ndim
ndarray.shape
np.shape(a)
ndarray.dtype
ndarray.size
np.eye(N)
np.reshape(a, newshape)
np.dot(a, b)
ndarray.T
ndarray.transpose()
np.transpose(a)
np.linalg.det(a)
np.linalg.inv(a)
np.linalg.eig(a)
np.random.rand()
np.random.rand(N)
np.random.uniform(low=0.0, high=1.0)
np.copy(a)
np.loadtxt(fname, comments='#', skiprows=0, usecols=None, unpack=False)
np.savetxt(fname, X, fmt='%.18e', delimiter=' ', newline='\n', header=' ', footer=' ', comments='#')
```

### Bibliothèque Python Matplotlib

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

plt.figure(num=None, figsize=None)
plt.plot(x, y, arguments_optionnels)
plt.axhline(y=0, xmin=0, xmax=1, arguments_optionnels)
plt.axvline(x=0, ymin=0, ymax=1, arguments_optionnels)
plt.errorbar(x, y, yerr=None, xerr=None, fmt=' ', ecolor=None, arguments_optionnels)
plt.bar(x, height, width=0.8, bottom=None, arguments_optionnels)
plt.fill(x, y, arguments_optionnels)
plt.axis('equal')
plt.axis('scaled')
plt.grid(visible=None)
plt.xlabel(xlabel, loc=None)
plt.ylabel(ylabel, loc=None)
plt.xlim(left, right)
```



```

plt.ylim(bottom, top)
plt.legend(loc='best')
plt.title(label)
plt.show()
plt.savefig(fname, format=None)
mpimg.imread(fname, format=None)
plt.imshow(X)
mpimg.imsave(fname, X)

```

## Méthodes numériques (préparées par les étudiant(e)s)

### Equations non linéaires

#### 1 - Méthode de bisection

On considère  $[a_0, b_0]$  et  $f(a_0) * f(b_0) < 0$  (condition de Bolzano)

**Si**  $f(a_n) * f(x_n) < 0$ ,  $a_{n+1} = a_n$  et  $b_{n+1} = x_n$

**Si**  $f(b_n) * f(x_n) < 0$ ,  $a_{n+1} = x_n$  et  $b_{n+1} = b_n$

Avantages : aucune hypothèse hormis la continuité et converge du moment que  $f(a_0) * f(b_0) < 0$

Mais : méthode lente qui ne peut pas être généralisée à  $\mathbb{R}^n$

#### 2 - Méthode de Picard

On considère une approximation de départ  $x_0$  et  $x_{n+1} = g(x_n)$ .

Si  $g \in C^0$  et  $x_n \rightarrow l$ , alors méthode Picard convergente et  $l$  point fixe de  $g$ .

$$|x_m - x_n| \leq \frac{K^n}{1-K} |g(x_0) - x_0|$$

Condition d'arrêt :  $\bar{x} - x_n = \frac{1}{1-g'(\xi_n)}(x_{n+1} - x_n)$  et  $|x_{n+1} - x_n| < \epsilon$  (convergence pas forcément garantie)

#### 3 - Méthode de Newton

Théorème Soit  $f \in C^2$ . Si  $f$  admet un 0 simple, alors :

$\exists \epsilon > 0$  t.q. si on choisit  $x_0 \in [\bar{x} - \epsilon, \bar{x} + \epsilon]$ , alors  $(x_n)$  converge vers  $\bar{x}$  et convergence quadratique.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

et condition d'arrêt :  $|x_n - x_{n-1}| < \epsilon$  ou  $|r_n| = |f(x_n)| < \epsilon$

Avantages : vitesse de convergence élevée, à condition d'avoir un bon  $x_0$ , peut-être généralisée à des systèmes d'équations linéaires, mais à chaque étape il faut calculer la dérivée, dont l'expression explicite n'est pas forcément connue,  $f \in C^2$ .

### Calcul intégral

#### 1 - Interpolation de Lagrange

$$\varphi_k(t) = \prod_{j=0, j \neq k}^m \frac{t - t_j}{t_k - t_j} \text{ polynôme de degré } m$$

$$\varphi_k(t_k) = 1, \varphi_k(t_j) = 0 \text{ pour } j \neq k$$

$$\text{Polynôme de Lagrange : } p(t) = \sum_{j=0}^m p_j \varphi_j(t)$$

#### 2 - Formules de quadrature non composites

Soit  $[x_i, x_{i+1}] \subset [a, b]$ , on le remplace par  $[-1, 1]$  en posant

$$x = \frac{x_i + x_{i+1}}{2} + \frac{x_{i+1} - x_i}{2}t \quad (1) \text{ et on note } f((1)) = g(t) \text{ et } \int_{x_i}^{x_{i+1}} f(x) dx = \frac{x_{i+1} - x_i}{2} \int_{-1}^1 g(t) dt$$

$$\mu_j = \int_{-1}^1 \phi_j(t) dt \text{ et } J(g) = \sum_{j=0}^m \mu_j g(t_j)$$

#### 3 - Formules de Newton-Cotes

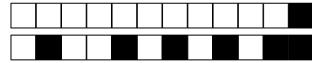
Point milieu :  $m = 0$

$$\varphi_0(t) = 1 ; \mu_0 = 2 ; J_i^{PM}(f) = (x_{i+1} - x_i) f\left(\frac{x_i + x_{i+1}}{2}\right)$$

Trapèze :  $m = 1$

$$t_0 = -1, t_1 = 1, \varphi_0(t) = \frac{t-1}{-2}, \varphi_1(t) = \frac{t+1}{2}, \mu_0 = \mu_1 = 1,$$

$$J_i^{Tr}(f) = (x_{i+1} - x_i) \frac{f(x_i) + f(x_{i+1})}{2}$$



Simpson :  $m = 2$

$$t_0 = -1, t_1 = 0, t_2 = 1, \varphi_0(t) = \frac{1}{2}(t^2 - t), \varphi_1(t) = -t^2 + 1, \varphi_2(t) = \frac{1}{2}(t^2 + t), \mu_0 = \frac{1}{3}, \mu_1 = \frac{4}{3}, \mu_2 = \frac{1}{3}$$

$$J_i^S(f) = (x_{i+1} - x_i) \left( \frac{1}{6}f(x_i) + \frac{4}{6}f\left(\frac{x_{i+1} + x_i}{2}\right) + \frac{1}{6}f(x_{i+1}) \right)$$

Newton :  $m = 3$

$$t_0 = -1, t_1 = -\frac{1}{3}, t_2 = \frac{1}{3}, t_3 = 1, \varphi_0(t) = -\frac{1}{16}(9t^3 - 9t^2 + t + 1)$$

$$\varphi_1(t) = \frac{9}{16}(3t^3 - t^2 - 3t + 1)$$

$$\varphi_2(t) = -\frac{9}{16}(3t^3 + t^2 - 3t - 1)$$

$$\varphi_3(t) = \frac{1}{16}(9t^3 + 9t^2 - t - 1)$$

$$\mu_0 = \frac{1}{4}, \mu_1 = \frac{3}{4} = \mu_2, \mu_3 = \frac{1}{4}$$

$$J_i^N(f) = (x_{i+1} - x_i) \left( \frac{1}{8}f(x_i) + \frac{3}{8}f\left(x_i + \frac{1}{3}(x_{i+1} - x_i)\right) + \frac{3}{8}f\left(x_i + \frac{2}{3}(x_{i+1} - x_i)\right) + \frac{1}{8}f(x_{i+1}) \right)$$

## EDO de premier ordre - Problème de Cauchy

$$h = \frac{T - t_0}{N} \text{ si partition régulière}$$

On pose  $f_n = f(t_n, u_n)$ ,  $h = t_{n+1} - t_n$  et  $f_{n+1} = f(t_{n+1}, u_{n+1})$  :

- **Schéma général**

$$\begin{cases} u_{n+1} = u_n + h \cdot (\text{méthode}) \\ u_0 = y_0 \end{cases}$$

- Euler progressif:  $u_{n+1} = u_n + hf_n$
- Euler rétrograde:  $u_{n+1} = u_n + hf_{n+1}$   
avec  $f_{n+1}$  qui dépend de  $u_{n+1}$
- Crank-Nicolson:  $u_{n+1} = u_n + \frac{h}{2}(f_n + f_{n+1})$
- Heun:  $u_{n+1} = u_n + \frac{h}{2}[f_n + f(t_n + h, u_n + hf_n)]$

- **Tableau de Butcher**

$c_1$	$a_{1,1}$	...	$a_{1,s}$	Le tableau suivant :
:	:	..	:	
$c_s$	$a_{s,1}$	...	$a_{s,s}$	
	$b_1$	...	$b_s$	

- Euler modifiée:

$$u_{n+1} = u_n + hf\left(t_n + \frac{1}{2}h, u_n + \frac{h}{2}f_n\right)$$

- Runge-Kutta classique:

$$u_{n+1} = u_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

$$\triangleright K_1 = f(t_n, u_n)$$

$$\triangleright K_2 = f\left(t_n + \frac{h}{2}, u_n + \frac{h}{2}K_1\right)$$

$$\triangleright K_3 = f\left(t_n + \frac{h}{2}, u_n + \frac{h}{2}K_2\right)$$

$$\triangleright K_4 = f(t_{n+1}, u_n + hK_3)$$

se décode dans la forme du *schéma général d'une méthode de résolution d'EDO1* de la manière suivante:

$$u_{n+1} = u_n + h(b_1K_1 + \dots + b_iK_i + \dots + b_sK_s)$$

$$\text{avec } K_i = f\left(t_n + c_ih, u_n + h \sum_{j=1}^s a_{i,j}K_j\right)$$

Erreurs :

$\tilde{u}_n = y_{n-1} + hf(t_{n-1}, y_{n-1})$  (pour Euler progressif)

Erreur de troncature **totale** :  $e_n = |y_n - u_n| = |y_n - \tilde{u}_n + \tilde{u}_n - u_n|$

Erreur de troncature **locale** :  $|y_n - \tilde{u}_n|$

Erreur de troncature **locale unitaire** :  $\tau_n(h) = \frac{|y_n - \tilde{u}_n|}{h}$

Erreur de troncature **locale unitaire maximale** :  $\tau(h) = \max_{n=1,\dots,N} \tau_n(h)$

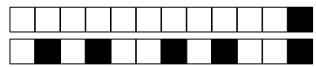
Erreur de troncature **transportée** :  $|\tilde{u}_n - u_n|$

Erreur de troncature **totale avec signe** :  $d_n = y_n - u_n$

Pour la **stabilité** :  $h \leq \frac{2}{\max_{t \in [t_0, T]} \left| \frac{\partial f}{\partial y}(t, y(t)) \right|}$



+1/19/42+



+1/20/41+