

**Enseignant : CDPetrescu****ICS - MAN****7 juillet 2022****Durée : 150 minutes**

Toto Tata

SCIPER : 999999

Attendez le début de l'épreuve avant de tourner la page. Ce document est imprimé recto-verso, il contient 16 pages. Ne pas dégrafer.

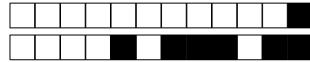
Indications générales

- 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.

Indications spécifiques

- 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
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> <input checked="" type="checkbox"/>
ce qu'il ne faut PAS faire what should NOT be done was man NICHT tun sollte		
<input checked="" type="checkbox"/>		



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 15 points.*

<input type="text"/>	,5								
<input type="text"/>	,5								
<input type="text"/>	,5								

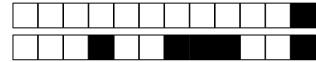
On donne ci-dessous le contenu d'une cellule Jupyter Notebook :

```
from copy import deepcopy
li1 = [10, [11, 12], 'hello']
li2 = li1
li3 = list(li1)
li4 = deepcopy(li1)
print('*****Première partie')
print(id(li1) == id(li2))
print(id(li1) == id(li3))
print(id(li1) == id(li4))
print('*****Deuxième partie')
li1[0] = 100
print(li2[0])
print(li3[0])
print(li4[0])
print('*****Troisième partie')
print(li1[1].append(13))
print(li2[1])
print(li3[1])
print(li4[1])
print('*****Quatrième partie')
print(li1[2].upper())
print(li2[2])
print(li3[2])
print(li4[2])
```

Précisez ci-dessous les messages affichés suite à l'exécution de la cellule indiquée plus haut.



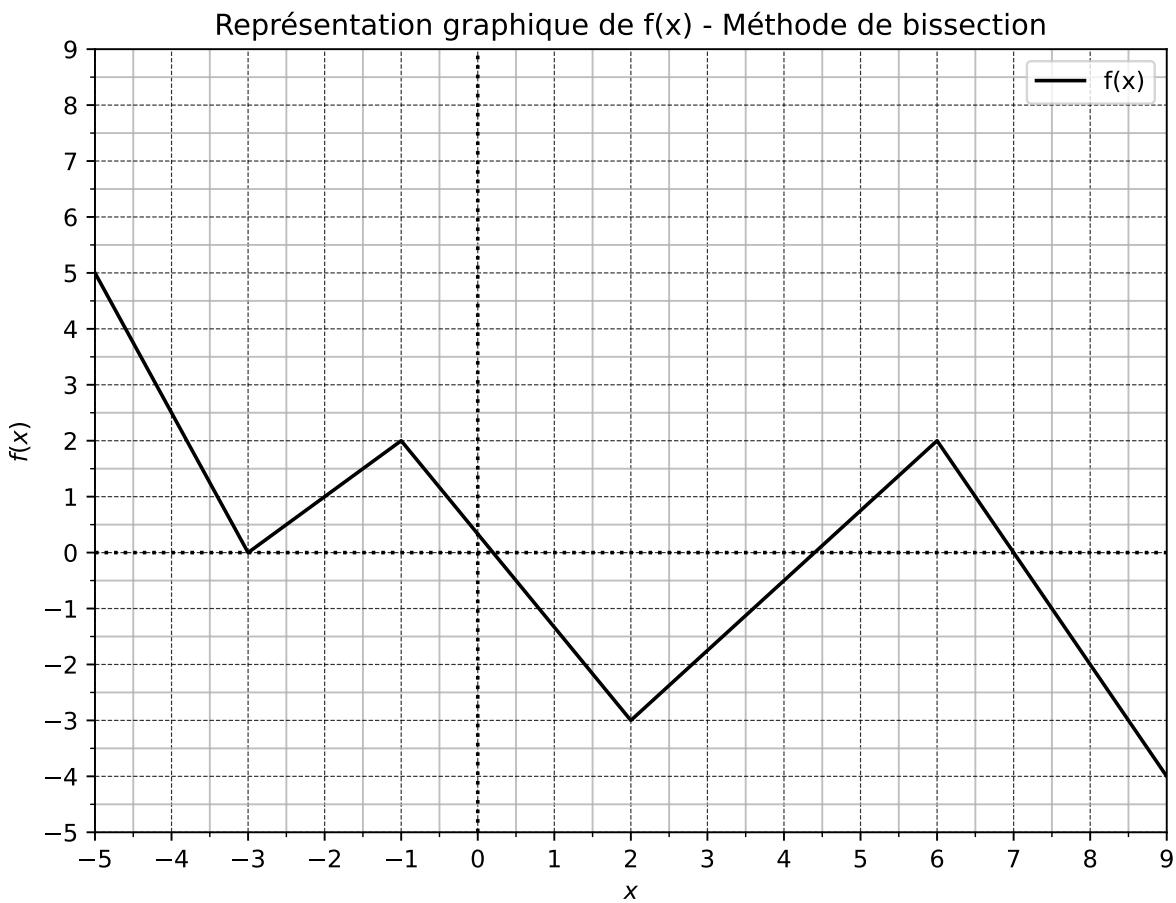
+1/3/58+



Question 2: Cette question est notée sur 14 points.

<input type="text"/> 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
.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5

Soit une fonction réelle d'une variable réelle $f : \mathbb{R} \rightarrow \mathbb{R}$ dont la représentation graphique sur l'intervalle $[-5, 9]$ 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 faire des calculs "à la main" et donc 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.



a) Afin de trouver numériquement **un** zéro de la fonction f dans l'intervalle $[-4, 8]$, on utilise la **méthode de bisection** et on considère l'intervalle de départ $[a_0, b_0] = [-4, 8]$ (et faites attention et prenez comme intervalle de départ l'intervalle $[a_0, b_0] = [-4, 8]$ et non pas l'intervalle $[-5, 9]$).

On considère les premières 4 itérations de la méthode de bisection (y compris la première itération pour l'intervalle de départ d'indice 0).

En utilisant le graphique donné, calculez et complétez ci-dessous les valeurs manquantes où, pour chaque indice $i \in \{0, 1, 2, 3\}$:

- a_i est la limite gauche de l'intervalle choisi à l'itération i ;
- b_i est la limite droite de l'intervalle choisi à l'itération i ;
- x_i est la valeur approchée du zéro calculé ou lu pour l'itération i .

$$a_0 = -4 \quad | \quad b_0 = 8 \quad | \quad x_0 = \dots$$

$$a_1 = \dots \quad | \quad b_1 = \dots \quad | \quad x_1 = \dots$$

$$a_2 = \dots \quad | \quad b_2 = \dots \quad | \quad x_2 = \dots$$

$$a_3 = \dots \quad | \quad b_3 = \dots \quad | \quad x_3 = \dots$$

b) Donnez une estimation (une majoration) convenable de l'erreur e_3 commise en utilisant la valeur approchée x_3 à la place de la valeur exacte (qui n'est pas connue) \bar{x} du zéro approximé par x_3 .

Complétez ci-dessous la valeur manquante.

$$e_3 = |\bar{x} - x_3| < \dots$$

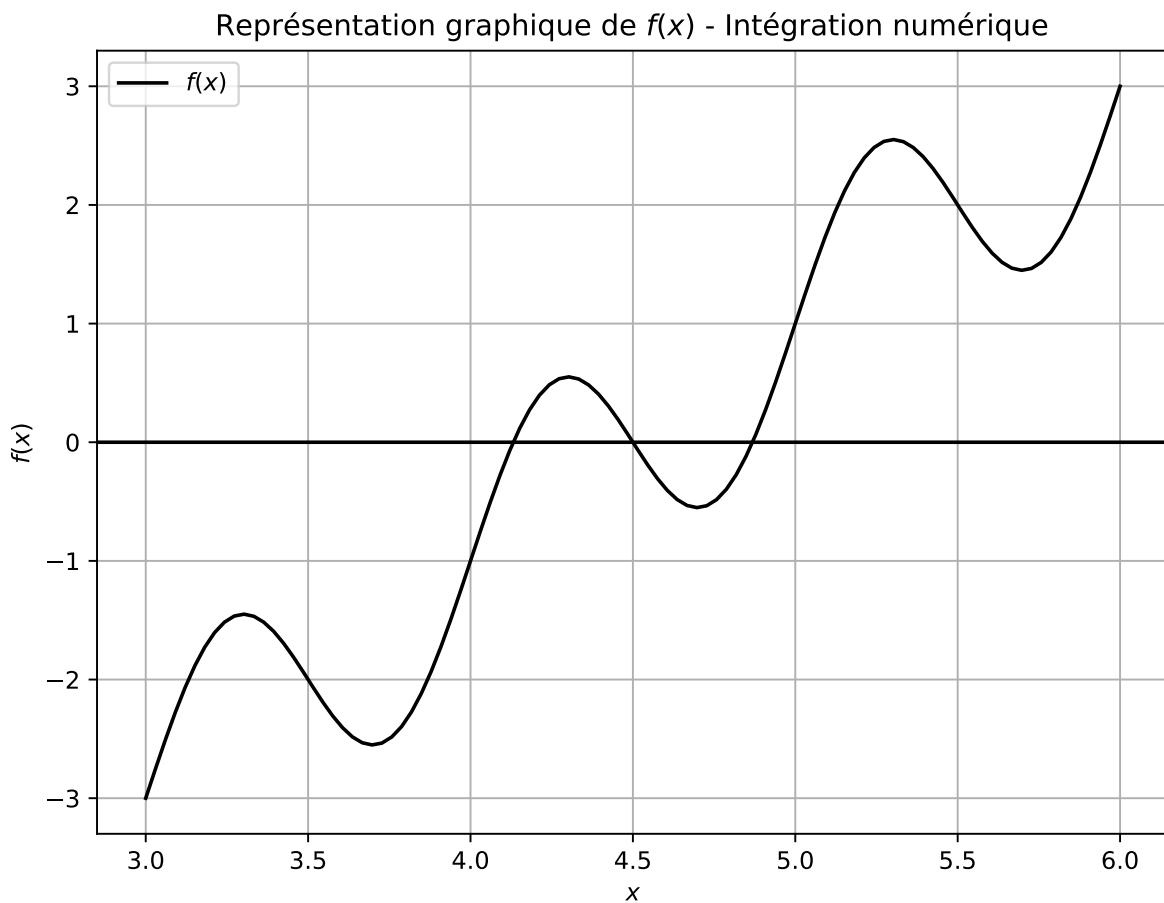
Question 3: Cette question est notée sur 14 points.

<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

On considère la fonction réelle d'une variable réelle $f : \mathbb{R} \rightarrow \mathbb{R}$ définie par :

$$f(x) = 2x - 9 + \sin(2\pi x)$$

et dont la représentation graphique sur l'intervalle $[3, 6]$ 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 faire des calculs "à la main" et donc 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.



On note I la valeur exacte de l'intégrale définie de la fonction f sur l'intervalle $[3, 6]$:

$$I = \int_3^6 f(x)dx$$

Vous devez calculer "à la main" (c'est-à-dire sans écrire un programme destiné à être exécuté par un ordinateur) les valeurs approchées de I obtenues en utilisant, respectivement, les formules de quadrature composites correspondant aux formules de Newton-Cotes non composites suivantes :

- la formule dite du **point de gauche** (cas où la valeur approchée de l'intégrale sera notée I^{PG}) ;
- la formule dite du **point de droite** (cas où la valeur approchée de l'intégrale sera notée I^{PD}) ;
- la formule du **point milieu** (cas où la valeur approchée de l'intégrale sera notée I^{PM}) ;
- la formule du **trapèze** (cas où la valeur approchée de l'intégrale sera notée I^{Tr}).

Rappel : On considère une partition σ d'un intervalle d'intégration réel $[a, b]$ en n sous-intervalles tel que $\sigma = \{x_0, x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_{n-1}, x_n\}$ où $x_0 = a$ et $x_n = b$.

Pour une fonction f à intégrer sur l'intervalle $[a, b]$:

- la formule de quadrature composite du **point de gauche** est donnée par :

$$I^{PG}(f) = \sum_{i=0}^{n-1} (x_{i+1} - x_i) f(x_i)$$

- la formule de quadrature composite du **point de droite** est donnée par :

$$I^{PD}(f) = \sum_{i=0}^{n-1} (x_{i+1} - x_i) f(x_{i+1})$$

Afin de calculer ces valeurs approchées de I , vous devez choisir chaque fois une partition (ou une subdivision) **régulière** de l'intervalle d'intégration **[3, 6]** en **trois sous-intervalles**.

Écrivez ci-dessous (aux endroits prévus à cet effet) les quatre valeurs approchées demandées.

$$I^{PG} = \dots \dots \dots$$

$$I^{PD} = \dots \dots \dots$$

$$I^{PM} = \dots \dots \dots$$

$$I^{Tr} = \dots \dots \dots$$



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 en supposant la première question résolue correctement.

Énoncé

Soit une fonction réelle d'une variable réelle $y(t) : \mathbb{R} \rightarrow \mathbb{R}$ qui est continûment différentiable sur \mathbb{R} (c'est-à-dire $y \in C^1(\mathbb{R})$) et qui satisfait l'équation différentielle et la condition initiale du problème de Cauchy suivant :

$$\begin{cases} y'(t) = y(t) + 2(t-1)e^t \\ y(0) = 1 \end{cases}$$

Question 4: *Cette question est notée sur 15 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"/> 13	<input type="text"/> 14	<input type="text"/> 15
<input type="text"/> .5															

Afin de résoudre numériquement le problème de Cauchy mentionné dans l'énoncé ci-dessus, on choisit le schéma de Runge-Kutta explicite à 3 étapes donné ci-dessous (avec les notations habituelles) :

$$\begin{cases} u_{n+1} = u_n + \frac{h}{6} (K_1 + 4K_2 + K_3) \\ u_0 = y_0 \end{cases}$$

où:

$$\begin{aligned} K_1 &= f(t_n, u_n) \\ K_2 &= f\left(t_n + \frac{h}{2}, u_n + \frac{h}{2}K_1\right) \\ K_3 &= f(t_{n+1}, u_n + h(2K_2 - K_1)) \end{aligned}$$

On vous demande d'implémenter ce schéma numérique en définissant une fonction Python nommée **rk3()** et qui doit :

* avoir 5 paramètres :

- f : la pente (la dérivée) de la solution recherchée ;
- t_0 : le moment initial de l'étude ;
- y_0 : la valeur initiale de la solution ;
- T : le moment final de l'étude ;
- N : le nombre de sous-intervalles ;

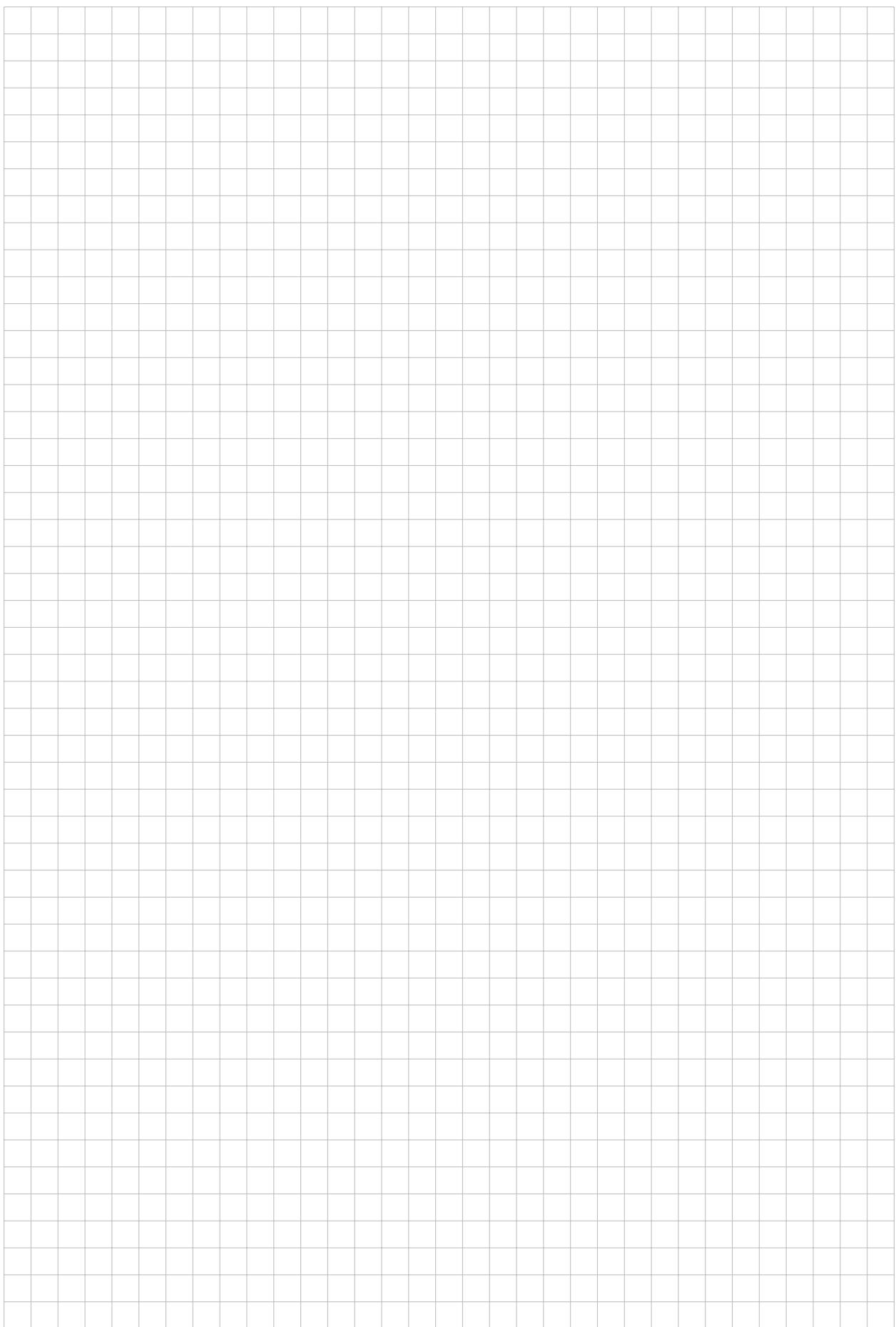
* retourner :

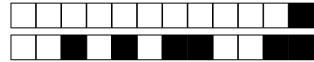
- t : le vecteur (1D array) avec les points (les instants) où la solution approchée a été calculée ;
- u : le vecteur (1D array) avec les valeurs calculées de la solution approchée.

Ecrivez à la page suivante la définition (l'en-tête et le corps) de la fonction Python **rk3()**.



+1/9/52+





Question 5: Cette question est notée sur 14 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"/> 13	<input type="text"/> 14
.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5	.5

On considère que la fonction Python `rk3()` (précisée à la question antérieure) a été définie correctement et qu'elle est disponible directement par son nom dans le code à écrire.

Vous devez résoudre le problème de Cauchy mentionné dans l'énoncé pour une **durée d'étude de 2.1** secondes et en respectant les consignes données sous forme de commentaires dans le canevas de code ci-dessous. Complétez le canevas de code ci-dessous comme indiqué.

```
# Importez la bibliothèque NumPy

# Définissez une fonction Python nommée f() avec le bon nombre d'arguments
# et qui correspond au membre de droite de l'équation différentielle du problème
# de Cauchy à résoudre, c'est-à-dire à la première dérivée de la solution recherchée

# Déclarez (créez) une variable locale nommée t_in et donnez-lui une valeur initiale qui
# correspond au moment initial de l'étude

# Déclarez (créez) une variable locale nommée y_in et donnez-lui une valeur initiale qui
# correspond à la valeur initiale de la solution recherchée

# Déclarez (créez) une variable locale nommée t_fin et donnez-lui une valeur initiale qui
# correspond au moment final de l'étude

# Déclarez (créez) une variable locale nommée nb qui correspond au nombre de sous-intervalles
# de la partition choisie et donnez-lui la valeur 32 comme valeur initiale

# Appelez la fonction rk3() avec les bons arguments pour résoudre le problème de Cauchy donné
# et stockez les résultats retournés par l'appel dans deux variables nommées temps et solutions

# Affichez le message suivant :
# Au moment final AAA la solution approchée vaut BBB !
# où AAA doit être la valeur du temps qui correspond à la fin de la durée d'étude
# et BBB doit être la valeur de la solution approchée à la fin de la durée d'étude

# Fin du code
```



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.

```
li_out = []
for m in range(1,4):
    li_in = []
    for n in range(1,4):
        if m % 2 == 1:
            li_in.append(m+n)
    li_out.append(li_in)
res = [print(el, end = '   ') for el in li_out]
print(res)
```

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

- [2, 3, 4] [] [4, 5, 6] [None, None, None]
- [2, 3, 4] [4, 5, 6] [None, None]
- [2, 3, 4, 5] [] [4, 5, 6, 7] [None, None, None, None]
- [2, 3, 4] [] [4, 5, 6] [None]

Question 7 (6 points)

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

```
nb = -12
li = [1, 2, 3]
dic = {100: 200, 200: 100}
nb = li
li = dic
dic = nb
print(F'{nb = }, {li = }, {dic = }')
```

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

- un message d'erreur
- nb = [1, 2, 3], li = {100: 200, 200: 100}, dic = [1, 2, 3]
- nb = [1, 2, 3], li = {100: 200, 200: 100}, dic = -12
- nb = {100: 200, 200: 100}, li = -12, dic = [1, 2, 3]

Question 8 (6 points)

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

- Le langage Python est un langage orienté objets.
- Le langage Python est un langage indépendant de plateforme.
- Le langage Python est un langage statiquement typé.
- L'implémentation de référence du langage Python est écrite en langage C.

**Question 9** (6 points)

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

```
x = 'je', 'tu', 'elle/il'  
y = ['suis', 'es', 'est']  
z = {el : y[-i] for el in x for i in range(1, len(x) + 1)}  
print(z)
```

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

- {'je': 'suis', 'tu': 'es', 'elle/il': 'est'}
- {'je': 'suis', 'tu': 'suis', 'elle/il': 'suis'}
- {'je': 'est', 'tu': 'es', 'elle/il': 'suis'}
- un message d'erreur

Question 10 (6 points)

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

```
import numpy as np  
tab = np.array([[1,2,3], [4,5,6], [7,8,9], [10,11,12]])  
print(tab.shape, end=' ')  
print(tab[:, 1], end=' ')  
print(tab[1, :], end=' ')  
print(tab[3:, :1])
```

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

- (3, 4) [1 4 7 10] [1 2 3] [[11]]
- (4, 3) [2 5 8 11] [4 5 6] [[11]]
- (3, 4) [1 4 7 10] [4 5 6] [[10]]
- (4, 3) [2 5 8 11] [4 5 6] [[10]]

Question 11 (6 points)

On calcule numériquement une intégrale définie (d'une fonction réelle d'une variable réelle qui est continue) à l'aide d'une formule de quadrature composite basée sur une formule non composite de degré d'exactitude 3 (comme la méthode de Simpson).

Si on passe d'une partition régulière avec n sous-intervalles à une partition régulière plus fine avec $2n$ sous-intervalles (c'est-à-dire si on divise le pas de la partition régulière par 2), alors l'erreur sera divisée par k (où k est une valeur que vous devez préciser).

Précisez la valeur de k (pour que l'affirmation précédente soit vraie) en cochant la case correspondante.

- $k = 2$
- $k = 8$
- $k = 16$
- $k = 4$


Question 12 (6 points)

Soient les points $P_0(t_0, p_0) = P_0(-2, 3)$, $P_1(t_1, p_1) = P_1(0, 5)$ et $P_2(t_2, p_2) = P_2(2, -1)$.

Cochez la case qui correspond aux polynômes de la base de Lagrange $\{\varphi_0(t), \varphi_1(t), \varphi_2(t)\}$ associée aux points mentionnés.

$\left\{ \varphi_0(t) = \frac{t^2}{8} + \frac{t}{4}, \varphi_1(t) = -\frac{t^2}{4} + 1, \varphi_2(t) = \frac{t^2}{8} - \frac{t}{4} \right\}$

$\left\{ \varphi_0(t) = 3 \left(\frac{t^2}{8} - \frac{t}{4} \right), \varphi_1(t) = 5 \left(-\frac{t^2}{4} + 1 \right), \varphi_2(t) = - \left(\frac{t^2}{8} + \frac{t}{4} \right) \right\}$

$\left\{ \varphi_0(t) = -\frac{t^2}{4} + 1, \varphi_1(t) = \frac{t^2}{8} - \frac{t}{4}, \varphi_2(t) = \frac{t^2}{8} + \frac{t}{4} \right\}$

$\left\{ \varphi_0(t) = \frac{t^2}{8} - \frac{t}{4}, \varphi_1(t) = -\frac{t^2}{4} + 1, \varphi_2(t) = \frac{t^2}{8} + \frac{t}{4} \right\}$

Question 13 (6 points)

Soit le schéma numérique de Runge-Kutta explicite à 3 étapes donné ci-dessous (avec les notations habituelles) :

$$\begin{cases} u_{n+1} = u_n + \frac{h}{6} (K_1 + 4K_2 + K_3) \\ u_0 = y_0 \end{cases}$$

où:

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

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

$$K_3 = f(t_{n+1}, u_n + h(2K_2 - K_1))$$

Parmi les tableaux de Butcher ci-dessous, choisissez celui qui correspond au schéma numérique mentionné et cochez la case respective.



0	0	0
1/2	1/2	0
1	-1	2
	1/6	4/6
		1/6



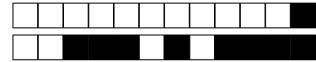
0	0	0
1/2	1/2	0
0	-1	2
	1/6	4/6
		1/6



1/6	0	0
4/6	1/2	0
1/6	-1	2
	0	1/2
		1



0	0	0
1/2	1/2	0
1	2	-1
	1/6	4/6
		1/6



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

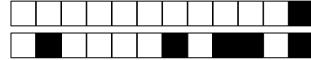
Point milieu : $m = 0$

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

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: $u_{n+1} = u_n + h f_n$
- Euler rétrograde: $u_{n+1} = u_n + h f_{n+1}$
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: $u_{n+1} = u_n + \frac{h}{2}[f_n + f(t_n + h, u_n + h f_n)]$

• Tableau de Butcher

c_1	$a_{1,1}$	\dots	$a_{1,s}$	
\vdots	\vdots	\ddots	\vdots	
c_s	$a_{s,1}$	\dots	$a_{s,s}$	
	b_1	\dots	b_s	

Le tableau suivant :

• Euler modifiée:

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

• Runge-Kutta classique:

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

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

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

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

$$\triangleright K_4 = f(t_{n+1}, u_n + h K_3)$$

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

$$u_{n+1} = u_n + h(b_1 K_1 + \dots + b_i K_i + \dots + b_s K_s)$$

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

Erreurs :

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

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

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

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

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

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

$$\text{Erreur de troncature \textbf{totale avec signe}} : d_n = y_n - u_n$$

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