555F

AVIS IMPORTANT AUX ETUDIANTS

1. Chacune des feuilles de votre copie doit comporter une étiquette code à barres placée à l'endroit indiqué «coller ici votre code à barres».
2. Une copie d'examen comporte une seule «feuille principale» et des «feuilles suites». Sur chacune de vos feuilles, le code à barres est obligatoire.
3. Cette feuille d'examen est strictement personnelle. Elle ne doit comporter aucun signe distinctif. Elle doit être écrite en noir et/ou bleu.
4. Le non respect de l'une de ces recommandations peut faire attribuer la note ZERO à l'épreuve.

esprit Se former autrement
HONORIS UNITED UNIVERSITIES

NOTE

Coller ici votre code à barre

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20

00	25	50	75

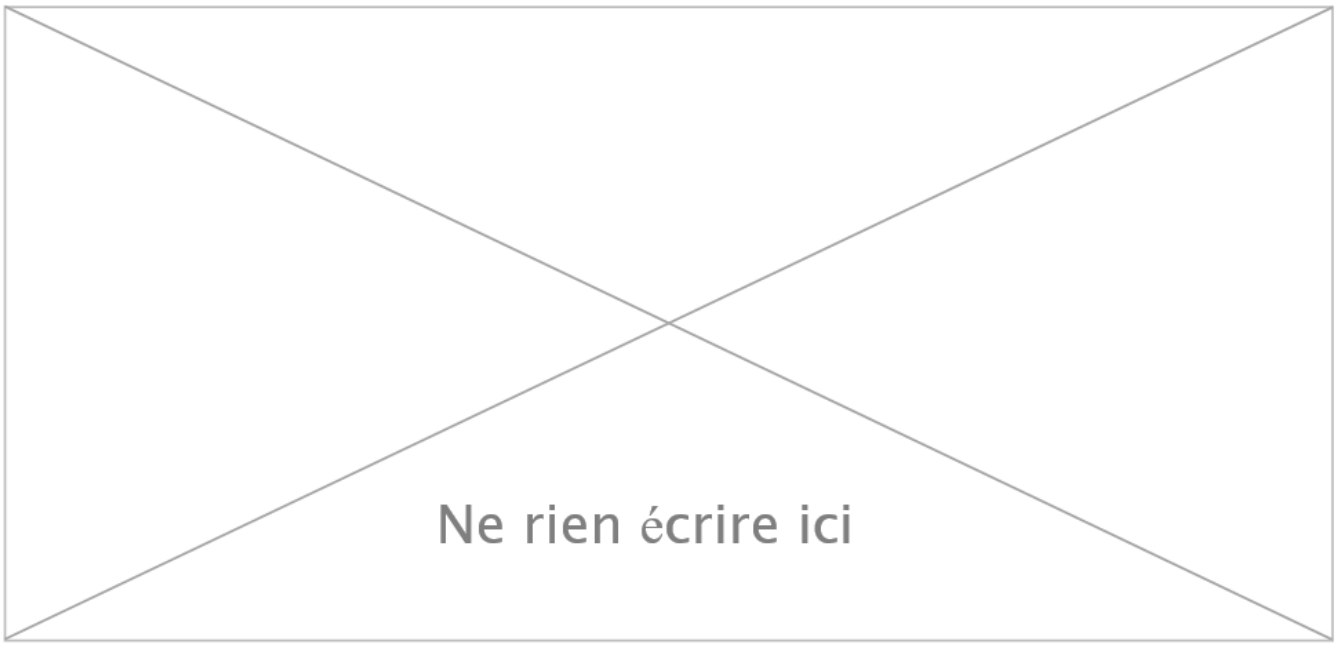
Module : Calcul Scientifique	Documents autorisés : OUI ☐ NON ■
Enseignant(s): Equipe CS – UP Math	Calculatrice autorisée : OUI ☐ NON ■
Classe(s) : 3A – 3B	Internet autorisée : OUI ☐ NON ■
Session : Principale Semestre : 1	Nombre de pages : 8
Date : 18/01/2024 Heure : 13h00	Durée : 1h30

Exercice 1 : QCU (3 points)

1. (1 point) Cocher la bonne instruction pour que la fonction `W_Newton(X)` ci-dessous renvoie correctement la famille du polynôme de Newton $W = [w0, w1, w2]$ en prenant en entrée une liste des abscisses $X = [x_0, x_1, x_2]$.

[ ]:
```
from numpy.polynomial import Polynomial
def W_Newton(X):
    w0=Polynomial([1])                      #w0(x)= 1
    w1=Polynomial([-X[0],1])                #w1(x)= x-x0
    w2=Polynomial(...................)      #w2(x)=(x-x0)(x-x1)
    w=[w0,w1,w2]
    return w
```

A `[X[0]*X[1] , -(X[0]+X[1]) , 1]` B `[1 , X[0]*X[1] , -(X[0]+X[1])]`
C `[X[0]*X[1] , -(X[0]-X[1]) , 1]` D `[1 , -(X[0]+X[1]) , X[0]*X[1]]`



2. La fonction `trapeze_composite(f,a,b,n,I)` ci-dessous permet de renvoyer la valeur de l'erreur d'intégration notée `E` de l'intégrale $I = \int_a^b f(x)dx$, par la formule composite des trapèzes suivante:

$$I_T(f) = \frac{h}{2}\Big(f(a) + f(b)\Big) + h\sum_{k=1}^{n-1} f(a+kh).$$

Elle prend en entrée la fonction f, les bornes de $[a,b]$, le nombre de sous-intervalles de subdivision n, et la valeur exacte de l'intégrale I. Compléter les pointillés en cochant les bonnes instructions pour que la fonction soit correctement implémentée.

[ ]:
```
def trapeze_composite(f,a,b,n,I):
    h = (b-a)/n
    s = ......................    #a)
    for k in np.arange(1,n):
        s += h*f(a+k*h)
    ......................       #b)
    return E
```

a) (1 point)
- A : 0
- B : $(h/2) * (f(a) + f(b))$
- C : $0.5 * (f(a) + f(b))$
- D : $0.5 * (-f(a) - f(b))$

b) (1 point)
- A : E=0
- B : E= abs$(s/2 - I)$
- C : E= abs$(s - I)$
- D : E= abs$(h * s - I)$

Exercice 2 : (17 points)

Pendant une séance de travaux pratiques (TP) d'une durée de 3 heures, un étudiant a observé que le ventilateur de son ordinateur se mettait en marche même avant le début de l'exécution des codes.

Afin de vérifier le bon fonctionnement du ventilateur, des mesures de températures instantanées, en degré Celsius (°c), ont été prises au cours de la séance de TP. Les résultats expérimentaux ont été consignés dans le tableau suivant :

t (heure)	0	0.5	1	1.5	2	2.5	3
x (°c)	0	60.57	66.71	63.63	57.22	49.83	42.51

Pour commencer, on importe la bibliothèque **NumPy** qui sera utilisée dans la suite, en utilisant un alias `np`.

[ ]:
```
import numpy as np
```

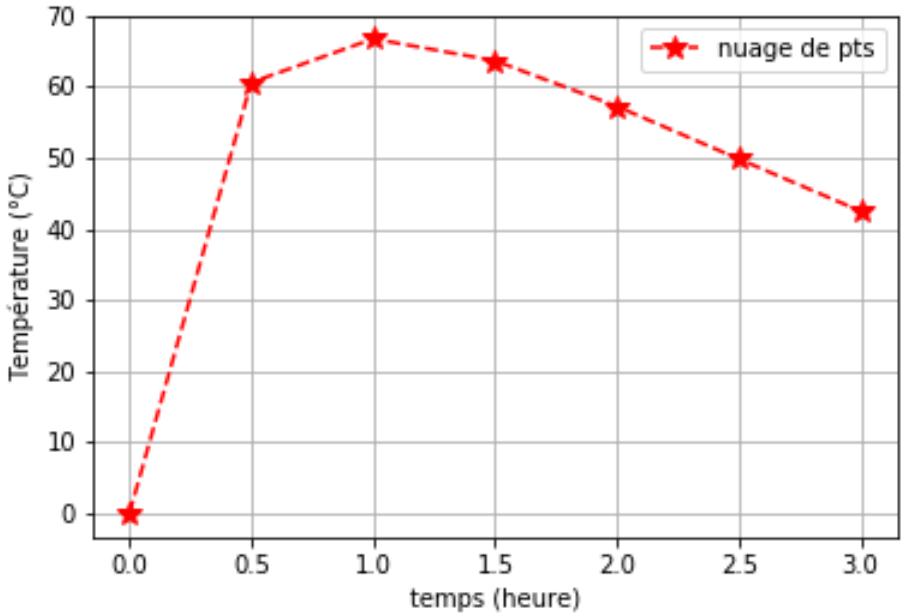
1. (0.75 point) Déclarer le vecteur temps **t** représentant les instants (en heure) auxquels les mesures ont été prises en utilisant la bibliothèque **NumPy**.

[ ]:
```
t= np.linspace(0,3,7)
```

2. (0.5 point) Déclarer manuellement le vecteur **x** représentant les températures mesurées (en °c), de manière à ce que **t** et **x** soient de même type.

[ ]:
```
x= np.array([0, 60.57, 66.71, 63.63, 57.22 ,49.83, 42.51])
```

3. (1.5 points) Écrire les commandes Python nécessaires pour tracer la courbe (ci-dessous) des mesures de température en fonction du temps en utilisant la bibliothèque appropriée.



[ ]:
```
import matplotlib.pyplot as plt
plt.plot(t,x,'--*')
plt.xlabel("temps (heure)")
plt.ylabel("Température (°C)")
plt.legend(('nuage de pts',))
plt.grid(True)
```

4. Entre les instants $1.5h$ et $3h$, on constate que les températures semblent s'ajuster linéairement.

   On souhaite déterminer la droite qui ajuste au mieux les observations pendant cette séance. Cette droite prend la forme suivante :

$$P(t) = a + bt, \quad t \in [1.5\ ,\ 3].$$

   (a) (1 point) Écrire une fonction nommée **moyenne**(X) qui prend en entrée un vecteur $X = (x_1, x_2, \ldots, x_n)$, et renvoie la valeur moyenne de X définie comme suit:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

[ ]:
```
def moyenne(X):
    return 1/len(X) * np.sum(X)
```

   (b) (2 points) Compléter la fonction nommée **covariance**(X,Y) prenant en entrée deux vecteurs $X = (x_1, x_2, \ldots, x_n)$ et $Y = (y_1, y_2, \ldots, y_n)$ de même dimension n, et renvoie la covariance entre X et Y, définie comme suit:

$$cov(X, Y) = \frac{1}{n}\sum_{i=1}^{n} (x_i - \overline{X})\ (y_i - \overline{Y})$$

[ ]:
```
def covariance(X,Y):
    moy_X= moyenne(X)        # moyenne de X

    moy_Y= moyenne(Y)          # moyenne de Y

    S= 0       # initialisation de la somme

    n= len(X)        # longueur de X

    for i in range(n)  :

        S+=(X[i]-moy_X)*(Y[i]-moy_Y)



    return S/n     (0.25 point)
```

   (c) (1.5 points) Définir une fonction nommée **coefficient**(X,Y) prenant en entrée deux vecteurs X et Y de même dimension et renvoie les coefficients a et b du polynôme P, avec

$$b = \frac{cov(X, Y)}{cov(X, X)} \quad \text{et} \quad a = \overline{Y} - b\overline{X}.$$

[ ]:
```
def coefficient(X,Y):
    b=covariance(X,Y)/covariance(X,X)
    a=moyenne(Y)-b*moyenne(X)
    return a,b
```

Coller ici votre
code à barre

5. On suppose que la température instantanée peut être décrite par la fonction :

$$f(t) = \sqrt{t}\, e^{-\theta_2 t + \theta_1} \tag{1}$$

avec θ_1 et θ_2 sont deux grandeurs physiques qui dépendent du choix de la gamme d'ordinateur et ses caractéristiques. Pour déterminer θ_1 et θ_2, on aboutit à résoudre le système (S) suivant :

$$(S) \begin{cases} \theta_1 - \theta_2 & = ln(66.71) \\ \theta_1 - 2\theta_2 & = ln\,(57.22) - ln\left(\sqrt{2}\right) \end{cases}$$

(a) (1.5 points) Compléter la fonction nommée **Resolution**(A,b) qui prend en entrée une matrice A quelconque et un vecteur colonne b, et qui retourne la solution exacte si le système (S) est de **Cramer**, et la booléenne *False* sinon.

[ ]:
```
def Resolution(A,b):

    if A.shape[0]==A.shape[1]  and np.linalg.det(A)!=0 :

        return np.linalg.solve(A,b)
    else:
        return False
```

(b) (1.5 points) Déclarer la matrice A et le second membre b associés au système (S) : $A\theta = b$, avec $\theta = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$, puis donner l'instruction qui permet de résoudre (S).

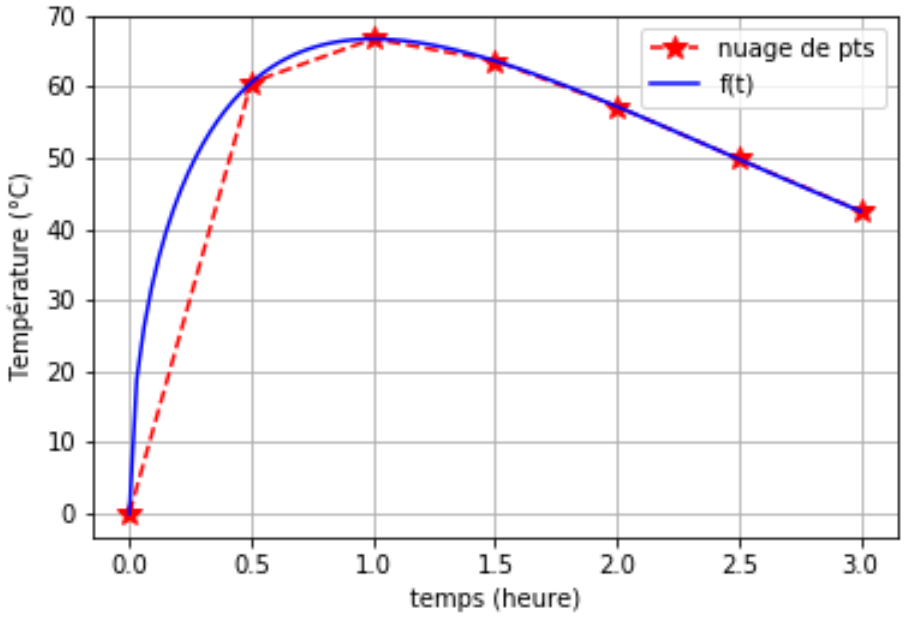
[ ]:
```
A= np.array([[1,-1],[1,-2]])

b= np.array([[np.log(66.71)],[np.log(57.22)-np.log(np.sqrt(2))]])

Solution= Resolution(A,b)
```

Dans la suite, on considère $\theta_1 \simeq 4.7$ et $\theta_2 \simeq 0.5$.

6. Afin de déterminer l'instant auquel le ventilateur se déclenche automatiquement, on s'intéresse à résoudre numériquement l'équation $(E) : f'(t) = 0$ avec $t \in]0, 3[$.



D'après la représentation graphique de f ci-dessous, l'équation (E) possède une unique solution $t^* \in]0, 3[$ tel que $f'(t^*) = 0$.

Afin d'obtenir une approximation de t^*, nous utilisons la méthode de Newton pour résoudre l'équation $f'(t) = 0$. Le schéma itératif correspondant est donné par :

$$\begin{cases} t_0 \in]0, 3] \\ t_{n+1} = t_n - \dfrac{f'(t_n)}{f''(t_n)} \quad \forall n \geq 0. \end{cases}$$

où $\lim\limits_{n \to +\infty} t_n = t^*$.

(a) (2 points) Écrire une fonction nommée Newton$(t0, \text{df}, \text{ddf}, \text{epsilon}, \text{Nmax})$ qui renvoie une liste contenant les itérés $[t_0, t_1, t_2, \ldots]$. Les paramètres de la fonction sont les suivants :

- $t0$: Valeur initiale dans l'intervalle $]0, 3]$.
- df : Fonction représentant f' la dérivée analytique de f.
- ddf : Fonction représentant f'' la dérivée seconde analytique de f.
- epsilon : Tolérance vérifiant le critère d'arrêt $|f'(t_n)| \leq$ epsilon.
- Nmax : Nombre maximal d'itérations.

[ ]:
```
def Newton(t0,df,ddf,epsilon, Nmax):
    t=[t0]
    k=0


    while  (np.abs(df(t[-1]))>epsilon) and  (k<Nmax) :

        k=k+1
        t.append(t[-1]-( df(t[-1]).evalf() / ddf(t[-1]).evalf() ) )









    return t
```

(b) Écrire les instructions Python qui permet de:

i) (1.25 points) Calculer symboliquement $f'(t)$ et $f''(t)$, avec $f(t) = \sqrt{t}\ e^{-0.5t+4.7}$.

[ ]:
```
import sympy as sp

t= sp.symbols('t')


f= lambda t: sp.sqrt(t)*sp.exp(-0.5*t+4.7)


df= sp.Lambda(t,sp.diff(f(t),t))


ddf= sp.Lambda(t,sp.diff(df(t),t))
```

ii) (1 point) Donner le(s) instruction(s) qui permet(tent) d'afficher l'instant à partir de quel le ventilateur déclenche automatiquement en considérant $t_0 = 0.01$, $epsilon = 10^{-3}$ et $Nmax = 100$.

[ ]:
```
Newton(0.01,df,ddf,10**-3, 100)[-1]
```

7. Pour trouver la température maximale à partir de quel le ventilateur se déclenche automatiquement, on se propose de résoudre numériquement le problème de Cauchy (PC) suivant:

$$(PC)\begin{cases} X'(t) = \dfrac{1-t}{2t}X(t), \quad t \in [t_0, t_0+T],\ t_0 > 0, \\ X(t_0) = f(t_0), \end{cases}$$

en considérant le schéma d'Euler itératif (Sc) suivant :

$$(Sc)\begin{cases} \text{En partant de :} \\ X_0 = X(t_0) = f(t_0), \\ X_1 = X(t_1) = f(t_1), \\ X_{n+1} = X_{n-1} + h\dfrac{1-t_n}{t_n}X_n \quad \forall n \geq 1, \end{cases}$$

avec h le pas de la subdivision de l'intervalle $[t_0, t_0+T]$, et $t_n = t_0 + nh$.

a) (1.5 points) Compléter la fonction nommée $schema(t0, X0, X1, T, h)$ qui prend en entrée $t0$ l'instant initiale, $X0$, $X1$ les températures initiales aux instants t_0 et t_1 respectivement, T la longueur de l'intervalle $[t0, t0+T]$ et h le pas de subdivision uniforme. Cette fonction doit retourner une liste des valeurs approchées $X = [X_0, X_1, X_2, \ldots]$ en utilisant le schéma (Sc).

[ ]:
```
def schema(t0,X0,X1,T,h):

    X= [X0,X1]              #initialisation de la liste X

    t= t0+h
    while t < t0+T:
        X.append(X[-2]+ h*X[-1]*((1-t)/t ))
        t+=h

    return X
```

b) (1 point) Donner les instructions nécessaires permettant de trouver une valeur approchée de la température maximale en considérant l'instant initial $t_0 = 0.1$, l'instant final $t_0 + T = 3$ et le pas $h = 0.1$.

[ ]:
```
t0= 0.1
h= 0.1
T= 3-t0

X0= f(t0)

X1= f(t0+h)

np.max(schema(t0,X0,X1,T,h))
```