1. ATS 2016

51) En résumé, la méthode des rectangles consiste à faire l'approximation :

$$\int f(x)\,dx \approx \sum_i f(x_i)\,\Delta x$$

D'après la procédure indiquée, l'intégrale S se calcule en ajoutant des termes de la forme $f\left(\frac{x_1+x_2}{2}\right)\times(x_2-x_1)$; $f(x)$ est donc bien la fonction à intégrer,

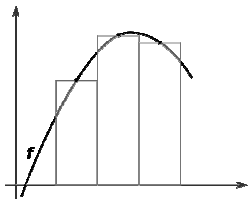


illustration succincte

l'intervalle $[x_1,x_2]$ variant entre $\left[a, a+\frac{b-a}{n}\right]$, pour $i=1$, et $\left[a+(n-1)\frac{(b-a)}{n}, b\right]$, pour $i=n-1$.

Sachant que $\Delta V = U_{AT} = \int_A^T \vec{E}.d\vec{r} = \int_{R+z_0}^{R} -\frac{Q}{4\pi\varepsilon_0 r^2}.dr = \int_{r=R}^{r=R+z_0} \frac{Q}{4\pi\varepsilon_0 r^2}.dr$

on obtient par identification $\Delta V = \int_{x=a}^{x=b} f(x)\,dx$ avec $\boxed{a=R\ ;\ b=R+z_0\ ;\ f(x)=\frac{Q}{4\pi\varepsilon_0 x^2}}$

question complémentaire

```
# application
Q=24e3
e0=9e-12
def f(x) :
    return Q/(4*np.pi*e0*x**2)

a=6000e3    # R
b=6060e3    # R+z0
n=2000
x=np.linspace(a,b,n)
plt.plot(x,f(x))
plt.show()
I=rectangles(a,b,n,f)
print('on obtient deltaV =',round(I),'V')
```

2. ATS 2017 *(remarque : la méthode d'Euler n'est plus au programme)*

8. La dérivée partielle $\frac{\partial V(y,z)}{\partial z}$ est associée à la quantité $\frac{V_{i,j+1}-V_{i,j}}{\Delta z}$: $\boxed{\frac{\partial V(y,z)}{\partial z} \approx \frac{V_{i,j+1}-V_{i,j}}{\Delta z}}$

Remarque : $\Delta y = \Delta z = 1\,\text{mm}$ représente le pas de discrétisation de la méthode d'Euler, c'est ici un pas spatial, dans les deux directions y et z.

9. La dérivée seconde correspondant à la variation de la dérivée première $\frac{\partial^2 V(y,z)}{\partial z^2} \approx \frac{\frac{V_{i,j+1}-V_{i,j}}{\Delta z} - \frac{V_{i,j}-V_{i,j-1}}{\Delta z}}{\Delta z}$

$\Rightarrow$ $\frac{\partial^2 V(y,z)}{\partial z^2} \approx \frac{V_{i,j+1}-V_{i,j}-V_{i,j}+V_{i,j-1}}{\Delta z^2}$, soit $\boxed{\frac{\partial^2 V(y,z)}{\partial z^2} \approx \frac{V_{i,j+1}-2V_{i,j}+V_{i,j-1}}{\Delta z^2}}$

10. $V_{i,j}$ représente $V(y,z)$, qui satisfait l'équation de Laplace $\frac{\partial^2 V(y,z)}{\partial y^2} + \frac{\partial^2 V(y,z)}{\partial z^2} = 0$

Par analogie avec le résultat précédent, on peut écrire que $\frac{\partial^2 V(y,z)}{\partial y^2} \approx \frac{V_{i+1,j}-2V_{i,j}+V_{i-1,j}}{\Delta y^2}$

L'équation de Laplace correspond donc à $\frac{V_{i+1,j}-2V_{i,j}+V_{i-1,j}}{\Delta y^2} + \frac{V_{i,j+1}-2V_{i,j}+V_{i,j-1}}{\Delta z^2} = 0$, avec, rappelons-le, $\Delta y = \Delta z$

On a donc $V_{i+1,j}-2V_{i,j}+V_{i-1,j}+V_{i,j+1}-2V_{i,j}+V_{i,j-1}=0$ d'où $\boxed{V_{i,j} = \frac{V_{i+1,j}+V_{i-1,j}+V_{i,j+1}+V_{i,j-1}}{4}}$

3. ATS 2018

48) Il s'agit de transposer sans trop se poser de questions le script fourni.

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

def f(Z):
    return Z/(1+Z**2)**4

Z=np.linspace(0,2,100)
plt.plot(Z,f(Z))
plt.xlabel('Z')
plt.ylabel('f(Z)')
plt.show()
```

4. ATS 2019

14) Pour obtenir le tracé de $U_p(t)$, il suffit de modifier une ligne dans la boucle définissant la deuxième fonction :

```
L.append(i(t))
```

devient

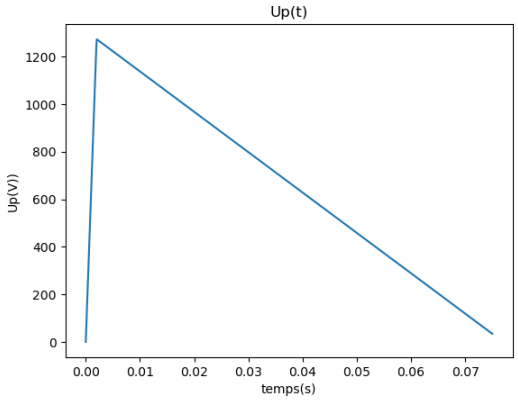
```
L.append(R*i(t))
```

Pour modifier le nom du graphique ainsi que l'étiquette des ordonnées, on remplace, dans la section "tracé" :

```
plt.ylabel('i(A)')
plt.title('i(t)')
```

par

```
plt.ylabel('Up(V))')
plt.title('Up(t)')
```



graphe obtenu (pour information)
(même forme que i(t), car U_p=Ri)

Remarque : on constate qu'il n'est pas nécessaire de comprendre le détail de ce programme pour répondre à la question.

5. ATS 2020

21)

```
def Temp(t,k) :
    return  30*(1-np.exp(-k*t))
```

22)

```
def erreur(k) :
    return sum((tab_T - Temp(tab_temps,k))**2)
```

23)

```
k = np.linspace(0.0001,0.01,100)
tab_e = []
for i in range(len(k)):
    tab_e.append(erreur(k[i]))
```

En complément, programme complet avec mesures supposées et utilisation possible :

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

#mesures supposées
tab_temps=np.linspace(0,3600,60)     # toutes les minutes pendant 1 heure
tab_T=30*(1-np.exp(-0.001*tab_temps))+.01

# question 21
def Temp(t,k) :
    return  30*(1-np.exp(-k*t))

# utilisation possible
print('(q.21) le tableau tab_Ttheo est :')
k=0.001
tab_Ttheo=Temp(tab_temps,k) # renommage suivant l'énoncé
```

```
print(tab_Ttheo)
print()
plt.plot (tab_temps,tab_Ttheo)
plt.xlabel('temps(s)')
plt.ylabel('Ttheo(°C)')
plt.title('évolution théorique de T (q.21)')
plt.show()

# question 22
def erreur(k) :
    return sum((tab_T - Temp(tab_temps,k))**2)

# utilisation possible
print('(q.22) Pour k =',k,', l'erreur est :')
print(erreur(k))
print()

# question 23
k = np.linspace(0.0001,0.01,100)
tab_e = []
for i in range(len(k)):
    tab_e.append(erreur(k[i]))

# utilisation possible
print('(q.23) le tableau tab_e est :')
print(tab_e)
```

6. ATS 2021

19) Voici un exemple de code Python

```
import matplotlib.pyplot as plt
import numpy as np
# M = np.loadtxt('mesures.csv')
# simulation (complément) : la ligne précédente est remplacée ici par
M = np.array([[0, 5],
              [3, 9.1],
              [6, 11.8],
              [9, 13.7],
              [12, 15.4],
              [15, 16.7],
              [18, 17.4],
              [21, 18.2],
              [24, 18.7],
              [27, 19.1],
              [30, 19.3],
              [33, 19.5],
              [36, 19.5],
              [39, 19.6],
              [42, 19.7],
              [45, 19.8],
              [48, 19.9],
              [51, 19.9],
              [54, 19.9]])
print('lecture du tableau "mesures.csv" :')
print('M =',M)

# construction des tableaux demandés
t = M[:,0]
T = M[:,1]

# affichage des tableaux demandés (complément)
print('tableau des temps t d''acquisition',t)
print('tableau des températures T',T)

# tracé
plt.plot(t,T,'o',label='points expérimentaux')
plt.legend()
plt.grid()
plt.title('T(t)')
plt.xlabel('t(s)')
plt.ylabel('T(°C)')
plt.show()
```

31)

```
import numpy as np
# données
delta=8600;
PmesH=999.8;     # P(H) mesurée
Pmes0=1000;      # P(0) mesurée
u_P=0.02;
N=10000;

tab_H=[]     # tableau vide
for i in range(N):
    # valeur au hasard de P(H) (simulée)
    PsimH=np.random.normal(loc=PmesH,scale=u_P)
    # valeur au hasard de P(0)
    Psim0=np.random.normal(loc=Pmes0,scale=u_P)
    # calcul de H correspondante
    H=delta*(1-PsimH/Psim0)
    # entrée dans le tableau
    tab_H.append(H)

# complément demandé : incertitude-type u(H)
u_H=np.std(tab_H)

# calcul de la moyenne et affichages (non demandé)
moy=np.mean(tab_H)
print('la hauteur moyenne vaut :',moy,' m')
print('incertitude-type :',u_H,' m')
```

Remarque : le programme conduit à une hauteur moyenne de 1,72 m *avec une incertitude-type de 24* cm.

7. ATS 2022 *(remarque : la méthode d'Euler n'est plus au programme)*

33. On a $v = \dfrac{dz}{dt} \Rightarrow dz = v\,dt$. Donc $z(t+dt) = z(t) + dz = z(t) + v\,dt$.

C'est le principe de la méthode d'Euler, avec un pas noté `dt`, pour la fonction `z(t)` liée à sa dérivée `v`.

Transposé en langage informatique dans une boucle, on obtient :

```
z(i+1) = z(i) + v(i)*dt .
```

34. D'une manière analogue, on va lier la fonction $v(t)$ à sa dérivée $\dot{v}$: $v(t+dt) = v(t) + \dot{v}\,dt$, avec $\dot{v} = g - \dfrac{\alpha}{m}v^2$

On obtient donc :

```
v(i+1) = v(i) + (g-alpha*v(i)**2/m)*dt .
```

8. ATS 2023

```
# données approximatives relevées sur les courbes (12 points)
# suivies de 138 zéros pour faire 150 termes, afin de tester le programme
p3= [0, 50,   100,  150,  200,  225,  250,  300,   350,   400,   450,  500]  +[0]*138
eta=[0, 0.36, 0.39, 0.40, 0.41, 0.415, 0.41, 0.40, 0.385, 0.375, 0.37, 0.365]+[0]*138
x4 =[1, 0.81, 0.76, 0.72, 0.68, 0.67, 0.65, 0.605, 0.55,  0.52,  0.50, 0.49] +[0]*138

maxi = 0
imax = 0

for i in range(1,150) :
    if eta[i] >= maxi :
        # question 42
        maxi = eta[i] # permet de conserver la valeur maximale du rendement
        imax = i      # permet de conserver l'indice correspondant au rendement maximum

print('mesure numéro =',imax)     # affichage amélioré dans la console
print('rendement maximum =',maxi) # affichage amélioré dans la console
# question 43
print('pour une pression p3 =',p3[imax],'bar') # p3 correspondant au rendement maximum
print('et un titre massique x4 =',x4[imax])    # x4 correspondant au rendement maximum
```

9. ATS 2024

16 - Note : l'énoncé indique d'utiliser la méthode des rectangles mais ne précise pas s'il s'agit des rectangles à gauche, à droite ou moyen. Les trois sont donc proposées ici. On rappelle toutefois que la méthode des rectangles moyens est la plus précise des trois.

exemple de lignes à ajouter en début de script pour faire un test en Python :

```
import numpy as np
t = np.linspace(600,1200,1000)  # 1000 valeurs entre 600 et 1200 min → assez bon résultat
y = np.linspace(0,600,1000)     # rayonnement croissant linéairement de 0 à 600 W/m²
Q = 3.24E+09   # valeur exacte de l'intégrale correspondante (aire d'un triangle rectangle)
print('Q attendue =',Q,'J')
```

réponse :

Dans la méthode "habituelle" on a une fonction `f(x)` associée à une variable `x`. Ici nous avons une fonction `y` associée à une variable `t`. Il faudra calculer à chaque fois l'aire du rectangle, c'est-à-dire multiplier la puissance surfacique `y` par l'intervalle de temps discrétisé, converti en secondes (l'intervalle `t` étant mesuré en minutes, il faudra le multiplier par *60*.

```
n = len(t)
somme = 0
for i in range(n-1):

   # Méthode des rectangles à gauche :
   somme = somme + y[i]*(t[i+1]-t[i])*60
   # Méthode des rectangles à droite :
   somme = somme + y[i+1]*(t[i+1]-t[i])*60
   # Méthode des rectangles moyens :
   somme = somme + (y[i]+y[i+1])/2*(t[i+1]-t[i])*60

print(somme*300)    # affiche l'énergie reçue, en joules
```

variante : on peut remplacer "`somme = somme + …`" par "`somme += …`"

33 -

exemple de lignes à ajouter en début de script pour faire un test en Python :

```
import numpy as np
# exemple de tableau :
h=np.linspace(9,11,100)  #liste de 100 valeurs entre 9 et 11 : moyenne ≈ 10
```

réponse :

```
n = len(h)
somme = 0
for i in range(n):
   somme += h[i]              # instruction 1
moyenne = somme / n           # instruction 2
print('moyenne =',moyenne)    # affiche la moyenne
```

34 - réponse :

```
somme = 0
for i in range(n):
   somme += (h[i]-moyenne)**2   # instruction 1
V = somme / n                   # instruction 2
print('variance =',V)           # affiche la variance
```