TP 17- Proposition de solutions

Solution 1 Méthode des rectangles

1. La fonction `rectangles` :

Méthode des rectangles

```
def rectangles(f,a,b,n):
    I=0
    h=(b-a)/n
    for i in range(n):
        I+=h*f(a+i*h)
    return(I)
```

2. On note que

$$\pi = 6\int_0^{\frac{1}{2}} \frac{1}{\sqrt{1-t^2}}\mathrm{d}t$$

```
n=int(input('Donner un entier naturel : n = '))
r=6*rectangles(lambda x:1/np.sqrt(1-x*x),0,1/2,n)
print('Valeur approchee : ',r)
print('Valeur machine : ',np.pi)
```

Solution 2 Méthode des trapèzes

- Calcul de la valeur moyenne :

Calcul de I_{moy}

```
def Imoy(L):
    n=len(L)
    x=[0.002*t for t in range(n)]
    s=0
    for k in range(n-1):
        s+=(x[k+1]-x[k])*(L[k]+L[k+1])/2
    return(s/0.002/(n-1))
```

Autre approche :

Calcul de I_{moy}

```
def Imoy2(L):
    I=(sum(L)-(L[0]+L[-1])/2)*0.002
    Tfinal=0.002*(len(L)-1)
    return(I/Tfinal)
```

- Calcul de l'écart type

Calcul de I_{ec}

```
import numpy as np

def Iec(L):
    Im=Imoy(L)
    y=[(e-Im)**2 for e in L]
    c=Imoy(y)
    return(np.sqrt(c))
```

Solution 3 Méthode des trapèzes

```
def trapezes(f,a,b,n):
    I=0
    h=(b-a)/n
    for i in range(n):
        I+=h*(f(a+i*h)+f(a+(i+1)*h))/2
    return(I)
```

Solution 4 Méthode de Simpson

L'approximation de l'intégrale est déjà donnée. Il est possible de retrouver le résultat :

- chercher l'expression du polynôme de degré 2 vérifiant : $P(c)=f(c)$, $P\left(\frac{c+d}{2}\right)=f\left(\frac{c+d}{2}\right)$ et $P(d)=f(d)$.
- calculer son intégrale sur $[c,d]$
- on trouve $\frac{d-c}{6}\left(f(c)+4f\left(\frac{c+d}{2}\right)+f(d)\right)$.

Méthode de Simpson

```
def simpson(f,a,b,n):
 I=0
 h=(b-a)/n
 for i in range(n):
  I+=h/6*(f(a+i*h)+4*f(a+(i+1/2)*h)+f(a+(i+1)*h))
 return(I)
```

Solution 5 Vitesse de convergence

1. Calcul du coefficient α : appliquer le logarithme donne une équation affine à deux inconnues ; ainsi, deux instances suffisent pour obtenir un système exploitable :

$$\begin{cases} \ln(|I_2-I|)\approx \ln(c)-\alpha\ln(2) & \textit{(pour } n=2\textit{)} \\ \ln(|I_{20}-I|)\approx \ln(c)-\alpha\ln(20) & \textit{(pour } n=20\textit{)} \end{cases}$$

Par opérations sur les lignes on trouve :

$$\alpha \approx \frac{\ln(|I_2-\pi|)-\ln(|I_{20}-\pi|)}{\ln(20)-\ln(2)}$$

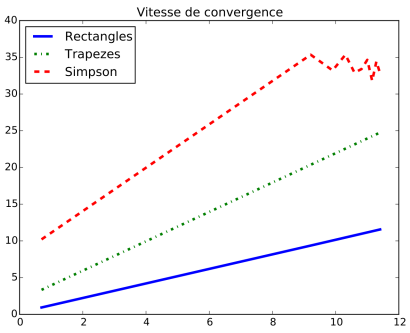
```
f=lambda x:1/np.sqrt(1-x*x)
I2=6*rectangles(f,0,1/2,2)
I20=6*rectangles(f,0,1/2,20)
print(round((np.log(abs(I2-np.pi))
  -np.log(abs(I20-np.pi)))/(np.log(20)-np.log(2)),2)
```

On trouve : $\alpha_{rect} = 1$
Puis : $\alpha_{trap} = 2$, et $\alpha_{Simp} = 4$
Remarque : Si l'on ne connaît pas la limite (et ne voulant pas la calculer) on peut considérer que I_{200} est une bonne approximation lorsque l'on travaille avec I_2 et I_{20}.
Ainsi, on peut remplacer la valeur théorique de la limite par un terme de rang important.
2. L'élément graphique le plus facilement identifiable est la droite. On trace pour chaque méthode les suites :

$$\Big(\ln(n)\ ,\ -\ln(|I_n - I|)\Big)$$

➢ La courbe est une droite dont le coefficient directeur est la vitesse de convergence.

```
plt.title("Vitesse de convergence")
N=range(2,100000,10000)
rect=[-np.log(abs(rectangles(np.exp,0,1,n)
        -np.exp(1)+1)) for n in N]
trap=[-np.log(abs(trapezes(np.exp,0,1,n)
        -np.exp(1)+1)) for n in N]
simp=[-np.log(abs(simpson(np.exp,0,1,n)
        -np.exp(1)+1)) for n in N]
plt.plot(np.log(N),rect,'-',label='Rectangles',lw=3)
plt.plot(np.log(N),trap,'-.',label='Trapezes',lw=3)
plt.plot(np.log(N),simp,'--',label='Simpson',lw=3)
plt.legend(loc='best')
plt.show()
```

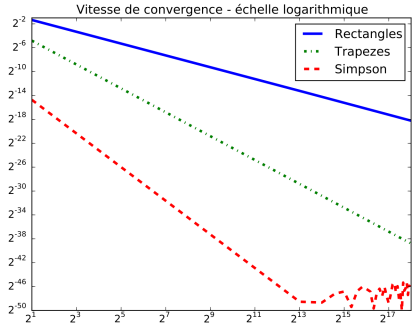


Remarque : Le décrochage de la courbe de la méthode de Simpson se produit pour :

$$e^{36} \approx 2^{52} \approx 10^{15}$$

On retrouve le problème de précision relatif à la représentation de nombre à virgule ne PYTHON.
On peut aussi représenter cette courbe directement dans un repère logarithmique comme suit :

```
plt.title("Vitesse de convergence - echelle log")
N=range(2,2**18,2**13)
rect=[abs(rectangles(np.exp,0,1,n)-np.exp(1)+1)
                          for n in N]
trap=...
simp=...
plt.loglog(N,rect,'-',basex=2,basey=2
                    ,label='Rectangles',lw=3)
plt.loglog(N,trap,'-.',basex=2,basey=2,...
plt.loglog(N,simp,'--',basex=2,basey=2,...
plt.legend(loc='best')
plt.show()
```



Solution 6 Intégration numérique
1. Choisissons la méthode des trapèzes : sur $[a,b]$ avec n subdivisions :

$$\frac{b-a}{n}\left(\frac{f(a)}{2}+\frac{f(b)}{2}+\sum_{k=1}^{n-1} f\left(a+k\frac{b-a}{n}\right)\right)$$

```
def argth(x,h):
    n=int(abs(x)/h+1)
    h=x/n
    f=lambda t:1/(1-t*t)
    y=f(0)/2+f(x)/2
    for i in range(1,n):
        y=y+f(i*h)
    return(y*h)
```

2. Les deux courbes sont symétriques par rapport à la première bissectrice, il suffit donc d'échanger les tableaux de valeurs abscisses-ordonnées :

```
h=1e-2
x=np.arange(-0.9,0.9,h)
y=[argth(e,h) for e in x]
plt.plot(x,y,'b-',lw=3,label='argth')
plt.plot(y,x,'r--',lw=3,label='th')
plt.plot([-1.5,1.5],[0,0],'k-',lw=1)
plt.plot([0,0],[-1.5,1.5],'k-',lw=1)
plt.plot([-1.5,1.5],[-1.5,1.5],'k-.',lw=1)
plt.legend(loc='best')
plt.grid()
plt.show()
```