Compte rendu TP2
Intégration numérique

23 novembre 2017

1. (a) Dans un premier temps, on défini la fonction f(x) en Python à l'aide de la fonction sqrt, pour la racine carrée :

```
def f(x) :
        return np.sqrt(1 - x*x)
```

L'ordinateur renvoie une erreur car il ne reconnaît pas la fonction.
On essaie en ajoutant `"from math import sqrt"`, mais on obtient une autre erreur :
`"TypeError : only length-1 arrays can be converted to Python scalars"`
Après des recherches sur Internet, on résout les problèmes.
Notre code final est

```
def f(x) :
        import numpy as np
        return np.sqrt(1 - x*x)
```

(b) On trace ensuite sans difficulté le graphe à l'aide du code :

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(-0.5, 0.5, 0.1)
plt.plot(x, f(x),'r-')
plt.grid('on')
plt.axis('equal')
plt.show()
```

On obtient le graphe suivant (voir annexe 1)

(c) Pour integrer cette fonction, nous avons procédé par un changement de variable. nous avons posé x = sin(t). Nous trouvons donc une primitive qui nous permet de calculer notre integrale qui vaut approximativement 0.95959077626.

2. (a) Pour calculer l'integrale S4 avec la méthode du point du milieu, on a codé le programme suivant :

```
S=0
x1=-0.5
x2=-0.25

for i in range(4):

    ck=(x1+x2)/2
    s=0.25*f(ck)
    S = S + s
    x1=x1+0.25
    x2=x2+0.25

print "Integrale S:",S,"\n
```

(b) La méthode du point du milieu est une approximation de I. On cherche donc à calculer l'erreur entre S4 et I :
L'erreur commise vaut 0,43959077626

(c) Nous avons eu un problème pour insérer le compteur de temps d'exécution. L'ordinateur ne connaissait pas la fonction `time`.On donc ajouté `"import clock from time"` dans le code. La fonction `time.clock()` renvoie l'heure actuel. Pour calculer le temps d'exécution, on crée un`start = time.clock()` avant la boucle `for`.A la sortie de la boucle, nous ajoutons `elapsed`, qui est égale à la différence entre `start` et l'heure actuel.
Le code final est :

```
S=0
x1=-0.5
x2=-0.25
import time as time
start = time.clock()


for i in range(4):

    ck=(x1+x2)/2
    s=0.25*f(ck)
    S = S + s
    x1=x1+0.25
    x2=x2+0.25


elapsed = (time.clock() - start)
print "Integrale S:",S,"\n
Temps de calcul de l'integrale:",elapsed,"\n"
```

(d) On cherche à généraliser la question précédente afin d'utilisé la méthode du point du milieu pour une fonction `f` et un intervalle `[a,b]` :

```
def point_milieu(f,a,b,n):
        S = 0
        x1 = a
        w = (b-a)/n
x2 = a + w
import time as time
start = time.clock()

for i in range(n):
        c = (x1+x2)/2
        sk = w*f(c)
        S = S + sk
        x1 = x1 + w
        x2=x2 + w
elapsed = (time.clock() - start )
return S,elapsed
```

La fonction coupe l'intervalle [a ;b] en n parts à l'aide de w.On a en initialisation l'intervalle [x1 ;x2] telle que x1 = a et x2 = a + w.
La boucle calcule ensuite la surface du rectangle Sk,l'ajoute dans S et incrémente x1 et x2 avec w.

(e) On teste la methode du point du milieu avec $f(x) = \sqrt{1 - x * x}, a = -0.5, b = 0.5$ et
$n = 10^k$ k allant de 1 à 6
On reporte les resultats dans le tableau suivant :

n	erreur	temps (sec.)
10	2.498915e-03	7.100000e-05
100	2.974488e-03	1.470000e-04
1000	2.979251e-03	1.237000e-03
10000	2.979298e-03	1.200700e-02
100000	2.979299e-03	1.173060e-01
1000000	2.979299e-03	1.185334e+00

(f) Nous avons eu des difficultes pour ecrire la fonction qui prends en argument une méthode de calcul d'intégrale, une fonction, un intervalle et un entier n, et renvoie et enregistre un tableau. Avec l'aide d'un camarade, on a codé la fonction suivante

```
        def enregistrer (methode, func, string, a, b, n):
                with open ('tableau_{:s}.txt'.format(string),'w') as fichier :
                        print('{0:12s} | {1:9s} | {2:s}'.format('    erreur', ' temps (sec)', 'n'))
                        fichier.write('{0:15s}|{1:12s}|{2:s}\n'.format('
erreur', ' temps (sec)', 'n'))
                        for j in range(1, n+1):
                                integrale,temps = methode(func, a, b, 10**j)
                                erreur = abs(integrale - I)
                                print('{0:e} | {1:e} | {2:d}' .format(erreur, temps, 10**j))
                                fichier.write('{0:9e} | {1:e} | {2:d}\n'. format(erreur, temps, 10**j))
                print('')
```

Il y a eu un problème informatique général, on ne pouvait plus éditer les fichiers gedit car l'enregistrement était impossible. Le professeur nous a alors suggeré d'utiliser nano : Nano permet d'éditer les fichiers depuis le terminal en modifiant la sauvegarde. Pour utiliser nano, on tape sur le terminal la commande :

```
nano fichier.py
```

Nano se commande seulement avec les commandes du clavier, on utilise Ctrl + O pour enregistrer, Ctrl + X pour quitter.
Le professeur nous a aussi conseillé de séparer les programmes des TP en exercices, afin de faciliter le debugage et éviter la perte de données possible en utilisant nano.

3. Pour coder la méthode du trapèze, on reprend la méthode du point milieu en modifiant la boucle :

```
def trapeze(f,a,b,n):

        S = 0
        w =(b-a)/n
        x1 = a
        x2 = a + w
        import time as time
        start = time.clock()

        for i in range(n):
                s = ((f(x1)+f(x2))*w)/2
                S = S + s
                x1 = x2
                x2 = x2 + w
        elapsed = (time.clock() - start )
        return S, elapsed
```

Pour la méthode du trapèze, on calcule l'aire d'un trapèze contrairement à la méthode du point milieu où on calculait l'aire d'un rectangle. Pour calculer une intégrale avec la méthode du trapèze, on procède comme pour la méthode du point milieu, mis à part qu'au lieu de prendre la fonction appliquée au milieu de l'intervalle [a ;b] f((a+b)/2), on fait la moyenne de la fonction appliquée en a et en b de l'intervalle [a ;b] (f(a)+f(b))/2.

On reprend le travail précédent avec la méthode du trapèze

n	erreur	temps (sec.)
10	3.940701e-03	6.300000e-05
100	2.988921e-03	2.940000e-04
1000	2.979395e-03	2.373000e-03
10000	2.979300e-03	2.390900e-02
100000	2.979299e-03	2.321000e-01
1000000	2.979299e-03	2.303118e+00

4. Pour la fonction simpson, on reprend la même méthode du point milieu et du trapèze en modifiant la boucle :

```
def simpson(f,a,b,n):

        S=0
        w=(b-a)/n
        x1=a
        x2 = a+w
        import time as time
        start = time.clock()

        for i in range(n):

                s = w*(f(x1) + 4*f((x1+x2)/2) + f(x2))/6
                S = S+s
                x1=x2
                x2 = x2+w

        elapsed = (time.clock()-start)
        return S, elapsed
```

   Pour la méthode de simpson on procède de la même façon que pour la méthode du trapèze en prenant un polynome du second degré pour approximer la partie haute de chacque rectangle de la somme.

   Soit [a ;b] notre intervalle d'integration,on prend [c ;d], in intervalle de longueur (b-a)/n, alors,la methode de simpson consiste a faire la moyenne de f(c), f((c+d)/2) et f(d) avec f((c+d)/2) qui compte 4 fois plus que f(c) et f(d).

n	erreur	temps (sec.)
10	2.979510e-03	1.290000e-04
100	2.979299e-03	3.710000e-04
1000	2.979299e-03	3.310000e-03
10000	2.979299e-03	3.463100e-02
100000	2.979299e-03	3.259900e-01
1000000	2.979299e-03	3.249828e+00

5. En comparant les tableaux des trois méthodes, on remarque que les trois méthodes donnent la même valeur de l'intégrale de f. Cependant, la méthode du point milieu est plus rapide pour calculer l'intégrale. La méthode de simpson parait plus précise par rapport aux deux autrescar elle se rapproche mieux de la courbe, mais en faisant tendre n vers l'infini, les méthodes ont la même précison. On peut donc dire que les trois méthodes sont équivalentes pour un n assez grand.

6. (a) On a recherché sur Internet comment tracer un cercle, nous avons trouvé un programme traçant un cercle. On a ensuite testé chaque ligne de code afin de comprendre leur utilité et ce qu'elles apportent.

```
import numpy as np
import matplotlib.pyplot as plt
from pylab import *
import math

circle1 = plt.Circle((0, 0), 1, color='r')


fig, ax = plt.subplots()
ax.set_xlim((-5,5))
ax.set_ylim((-5,5))

ax.add_artist(circle1)
plt.grid('on')
plt.show()
```

La surface du disque unité estπ.

(b) Pour générer des valeurs aléatoires à l'aide de la fonction numpy.random.rand(), on a d'abord pensé à utiliser les fonctions cosinus et sinus, pour obtenir une valeur dans l'intervalle $[-1;1]$ On a ensuite appris que la fonction numpy.random.rand() donne une valeur aléatoire entre 0 et 1. Pour une valeur aléatoire entre 0 et 1, on code simplement :

```
for i in range(5):
        rand = np.random.rand()
        print rand,"\n"
```

Soit x $\in [0,1]$
$0 \leq x \leq 1$
$0 \leq 2x \leq 2$
$-1 \leq 2x - 1 \leq 1$
Pour obtenir une valeur aléatoire entre -1 et 1, on a donc :

```
for j in range(5):
        randn = 2*np.random.rand()-1
        if randn > 1:
                randn = cos(randn)
        print randn,"\n"
```

On ajoute la condition car nous avons eu des cas où le nombre était supérieur à 1.

(c) On doit placer 1000 points sur un graphe, en rouge les points intérieurs au disque unité, et en vert les points extérieurs au disque (voir annexe 2). On ajoute dans le code la condition :

```
if ( X*X + Y*Y )< 1:
    plt.scatter(X,Y,color='r', marker = '.')
else:
                plt.scatter(X,Y,color='green', marker = '.')
```

Nous avons eu un problème au niveau du graphe : à chaque fois qu'on executait le programme, on obtenait 1000 points de plus sur le graphe (voir annexe 3)
Le professeur nous a conseillé d'ajouter en fin de code une commande pour effacer le fichier avant d'enregistrer :

```
os.remove("cercle(annexe2).png")
plt.savefig('cercle(annexe2).png')
```

(d) Pour compter le nombre de points, on ajoute au programme de la question précédente des compteurs dans les conditions :

```
if ( X*X + Y*Y )< 1:
        plt.scatter(X,Y,color='r', marker = '.')
        I = I + 1
else:
        plt.scatter(X,Y,color='green', marker = '.')
        E = E + 1
```

Dans le programme suivant, on calcule l'erreur entre l'aire du cercle qu'on a calculé à la main et celle qu'on a calculé à l'aide des points aléatoires situé à l'intérieur du cercle. On a d'abord utilisé la formule dans l'énoncé avant qu'un collègue nous fasse remarquer qu'il y avait une erreur dans l'énoncé, il fallait ajouter un *4.

```
start = time.time()
Er = abs(4*I/N - math.pi )
r = time.time() - start
print "\nErreur :",Er
print " Temps :", r
```

(e) On généralise la question précédente en remplaçant 1000 par N

```
def Monte_Carlo(N):
        import math
        import numpy as np
        import time as time
        I = 0
        for i in range(N):
                X = 2*np.random.rand()-1
                Y = 2*np.random.rand()-1
                if X > 1:
                        X = cos(X)
                if Y > 1:
                        Y = cos(Y)

                if ( X*X + Y*Y )< 1:
                        I = I + 1
        start = time.time()
        Er = abs(4*I/N - math.pi )
        r = time.time() - start
        return Er,r
```

(f) On teste Monte Carlo pour $N = 10^k$, k allant de 1 à 6

n	erreur	temps (sec.)
10	1.415927e-01	9.536743e-07
100	1.141593e+00	9.536743e-07
1000	1.415927e-01	9.536743e-07
10000	1.415927e-01	1.907349e-06
100000	1.415927e-01	3.099442e-06
1000000	1.415927e-01	2.861023e-06

(g) On reprend la méthode de Monte-Carlo précédente en ajoutant une "e coordonné Z. et en modifiant le calcul de l'erreur, qui calcule la différence entre 2 volumes

```
def Monte_Carlo_boule(N):

        import math
        import numpy as np
        import time as time
        I = 0
        for i in range(N):
                X = 2*np.random.rand()-1
                Y = 2*np.random.rand()-1
                Z = 2*np.random.rand()-1
                if X > 1:
                        X = cos(X)
                if Y > 1:
                        Y = cos(Y)
                if Z > 1:
                        Z = cos(Z)

                if ( X*X + Y*Y + Z*Z )< 1:
                        I = I + 1
        start = time.time()
        Er = abs(8*I/N - math.pi*4/3 )
        r = time.time() - start
```

Dans ce TP, nous avons donc découvert et expérimenté différentes méthodes pour approcher le calcul d'intégral. Nous avons appris à construire des tableaux et tracer des cercles en langage python, ainsi que rédiger en utilisant Latex.