TP2
Intégration numérique

23 novembre 2017

Réalisé par N'GOAN Auguste et LAHMADI Salaheddine

Encadré par Jean-Paul Cardinal

Le but de ce tp est de calculer approximativement l'intégrale d'une fonction sur un intervalle borné et fermé

1. Création d'une fonction simple et représentation graphique
   (a) Nous avons écris la fonction $f(x) = \sqrt{1-x^2}$ en python. Comme nous l'avions dit dans le TP précédent, ce genre de question ne nous pose pas de problèmes.

   Le code qu'on a utilisé est le suivant :

   ```
   def f(x) :
       return sqrt(1-x**2)
   ```

   (b) Nous avons représenté le graphique de la fonction f sur l'intervalle $[-0.5, 0.5]$ grâce au module matplotlib.plot et numpy que nous avons vu dans le précedent TP. Cette question ne nous a pas posé de problèmes.

   (c) Pour cette question, nous devons calculer, à la main, l'intégrale I de f sur l'intervalle donné. Nous l'avons fait à l'écrit puis nous vous l'avons rendu.
   Nous avons trouvé I = $\pi/6 + \sqrt{3}/4$ Soit environ 0.9566

2. Calcul approché de I au moyen de la méthode du point milieu
   (a)

   (b) On définit sur python une subdivision régulière de l'intervalle d'intégration [-0,5,0.5]. On a donc : x0 = -0.5 ; x1 = -0.25 ; x2 = 0 ; x3 = 0.25 ; x4 = 0.5

   Sur le graphique que nous vous avons remis à la question 2), nous avons dessiné les rectangles de bases $[X_{k-1}; X_k]$ et de hauteur f(c_k).

   Puis nous avons calculé leur surface à l'aide de python et nous avons obtenu :
   Sk = 0.95959
   La méthode du point milieu consiste à calculer la somme des surfaces pour approximer l'intégrale I.

   (c) On cherche à trouver l'erreur commise par la différence entre la somme des surfaces et l'intégrale de f pour évaluer la précision de la méthode du point milieu. À l'aide de python, on a l'erreur commise qui vaut environ 0.00298

   (d) Dans le but de mesurer le temps de calcul de l'intégrale nous avons eu recours à la fonction clock() du module time.Nous avons eu des difficultées à mesurer ce temps mais après l'aide de quelques camarades nous avons finalement réussi à mesurer ce temps.

(e) Nous avons commencé par écrire un programme qui semblait marcher mais il ne donnait pas un résultat correct à l'aide du professeur nous nous sommes rendu compte que nous avions oublié de mettre flot(), une erreur que nous ne commetrons plus à l'avenir.

(f) Nous avons testé la fonction point_milieu avec $f(x) = \sqrt{1-x^2}$, $a = -0.5, b = 0.5$ et $n = 10^k$ et k variant de 1 à 6.Nous avons obtenus

(g) On obtient le résultat de la question f :

n	erreur	temps(sec)
10	9,614021865e-04	1,212e-05
100	9,62241852e-06	5,782e-05
1000	9,622552e-08	5,619e-04
10000	9,72e-10	8,368e-03
100000	1,9e-11	5,852e-02
1000000	1e-11	4,523e-01

Nous avons rencontré plusieurs difficultés pour cette question et donc on a calculé l'erreur avec la calcultrice.

3. la méthode du trapèze

Cette méthode consiste à approximer f sur l'intervalle $[x_{k-1}, x_k]$ par la fonction affine qui prend les mêmes valeurs que f en x_{k-1} et x_k.

On a repris le travail précédent pour la méthode du trapèze et on obtient :

n	erreur	temps(sec)
10	9,61402196e-4	1,700e-05
100	9,622428e-4	6,600e-05
1000	9,6235e-8	5,900e-04
10000	9,6252e-10	5,600e-03
100000	9,52e-12	5,540e-02
1000000	5,2e-13	5,440e-01

De la même de manière on a calculé l'erreur commise avec la calculatrice.On pense qu'il y a une erreur dans le tableau car il nous parait bizarre de trouver la même erreur pour trois valeurs différentes de n et que la valeur de la dernière erreur soit superieur aux valeurs précédentes.

4. méthode de Simpson

La méthode de Simpson consiste à approximer f sur l'intervalle $[x_{k-1}, x_k]$ par le polynôme de degré 2 qui prend les mêmes valeurs que f en x_{k-1}, c_k et x_k.

On a repris le travail effectué à la question 2) pour la méthode de Simpson et on obtient :

n	erreur	temps(seconde)
10	2,11636e-7	1,9e-05
100	3,1e-11	1,700e-04
1000	9e-12	1,100e-03
10000	9e-12	1,800e-02
100000	9e-12	1,544e-01
1000000	1e-11	7,453e-01

5. Comparaison des méthodes

En faisant le test pour les trois méthodes , on en déduit que la méthode Simpson est la plus efficace suivie de la méthode des trapèzes et enfin de la méthode du point milieu.
En effet, le nombre de valeurs pris pour approximer la fonction diffère pour chacune de ces méthodes :
-Pour la méthode du Point Milieu, on approxime f par une fonction constante (donc de degré 0), on ne prend donc que 1 seul point $(f(c_k))$ pour approximer f.
-Pour la méthode des trapèzes, on approxime f par une fonction affine (donc de degré 1), on prend alors 2 points $(f(x_{k-1}))$ et $f(x_k))$ pour approximer f.
-Finalement, pour la méthode Simpson, on approxime f par une fonction du 2nd degré, on prend alors 3 points($f(x_{k-1})$, $f(x_k)$ et $f(c_k)$) pour approximer f. La précision est donc plus élevée pour cette méthode.

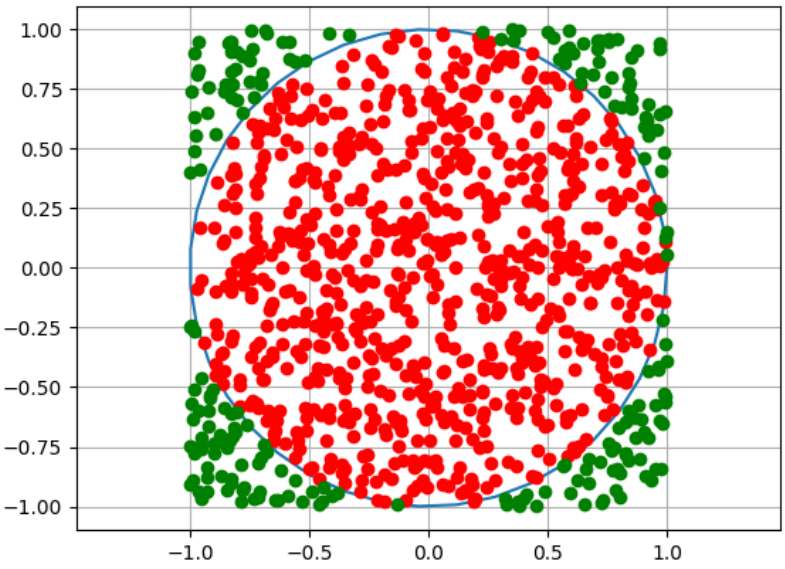
par conséquent, il est préferable d'utiliser la méthode Simpson

6. **méthode de Monte-Carlo**

(a) Nous devions dessiner le cercle unité à l'aide de matplotlib.pyplot. Cette question nous a posé quelques difficultés, nous nous sommes donc aidés d'un programme mais on arrivais pas à enlever la droite d'équation $f(x) = x$
La surface du disque unité est donnée par : S = π*(1**2) = π .

(b) On a regardé sur Internet comment utiliser la fonction **numpy.random.rand()** et ceci ne nous a pas posé de problèmes particuliers, on a généré alors 7 valeurs aléatoires suivant une distribution de loi uniforme sur [0,1] et [-1 ;1]. On a obtenu les valeurs suivantes :
-pour [0,1] : [0.94031137, 0.55742582, 0.57642561, 0.71867386, 0.89369628, 0.32443451, 0.6354503]
-pour [-1,1] : [0.5595875 , 0.73119842, -0.53028118, -0.469948009, 0.81628663, 0.07277804, 0.16590714]

(c) On a généré N = 1000 points suivant la distribution uniforme sur $[-1, 1]$. Parmi ces 1000 points, on a inséré les points intérieurs au disque unité en rouge et les points extérieurs en vert. Le graphique obtenu est le suivant :



(d) Il y a environ 731 points rouges et 269 points verts. Nous avons calculé l'approximation de la surface S du disque avec le rapport $4 * I/N$ avec I le nombre de points intérieur (rouge) ainsi nous avons calculé l'erreur commise $abs(4 * I/N - S)$. A l'aide de **time.clock()** nous avons mesuré le temps de calcul de cette approximation.
L'erreur commise vaut : 3.285e-02
Le temps associé au calcul vaut : 1.263e-03 secondes.

(e) La fonction Monte Carlo est intégré dans le module que nous avons créé, qui se nomme "methode.py". On a eu des difficultés pour écrire ce programme et après deux heures d'acharnement, nous l'avons finalement reussi.

(f) En testant la fonction Monte Carlo avec N=10**k pour k allant de 1 à 6, on obtient le résultat suivant :

n	erreur	temps(sec)
10	4,203e-01	5,320e-05
100	2,103e-01	2,329e-04
1000	4,656e-02	2,461e-03
10000	7,852e-03	2,134e-02
100000	2,954e-03	1,566e-01
1000000	6,255e-04	1,398

(g) En reprenant le travail précédent avec le calcul du volume de la boule unité par la méthode de Monte Carlo et nous obtenons :

n	erreur	temps(sec)
10	1,895	4,376e-05
100	8,325e-02	3,561e-04
1000	9,565e-02	3,218e-03
10000	5,852e-03	2,184e-02
100000	4,637e-03	2,192e-01
1000000	3,623e-03	2,221