TP2
Intégration numérique

3 octobre 2017

Dans ce TP, on présente quelques méthodes numériques pour calculer, approximativement, l'intégrale d'une fonction sur un intervalle compact (borné, fermé).

1. Création d'une fonction simple et représentation graphique [Python : module `math` ; Maths : calcul exact d'une intégrale].
   (a) Ecrire la fonction $f(x) = \sqrt{1-x^2}$ sous forme d'une fonction python (on utilisera la fonction `sqrt()` du module `math`).
   (b) Représentation graphique sur l'intervalle $[-0.5, 0.5]$.
   (c) Calculer, à la main, l'intégrale I de cette fonction sur l'intervalle donné.
2. Calcul approché de I au moyen de la **méthode du point milieu** [Python : module `time`, création et écriture d'un tableau de résultats dans un fichier texte au moyen des fonctions `open()` et `write()` ; Maths : approximation d'une fonction par une fonction en escalier, approximation numérique d'une intégrale].
   (a) Représenter graphiquement f sur papier, repère orthonormé, unité = 8 cm.
   (b) On définit une subdivision régulière $x_0, \cdots, x_4$ de l'intervalle d'intégration $[0, 1]$ en $n = 4$ parts égales (on a donc $x_0 = -0.5, x_1 = -0.25, \cdots, x_4 = 0.5$. On note $c_1, \cdots, c_4$ les milieux de ces sous-intervalles. Pour chaque $k = 1, \cdots, 4$, dessiner le **rectangle** de base $[x_{k-1}, x_k]$ et de hauteur $f(c_k)$, et calculer sa surface s_k. La méthode du point milieu consiste à prendre la somme $S_4 = \sum_{i=1}^{4} s_k$ comme approximation de I
   (c) Calculer l'erreur commise $|S_4 - I|$.
   (d) Utiliser la fonction `clock()` du module `time` pour mesurer le temps de calcul de votre intégrale.
   (e) Ecrire une fonction python `point_milieu` qui prend en arguments une foncion f, des bornes a, b et un entier n et qui renvoie l'intégrale approchée de f sur $[a, b]$ au moyen de la méthode du point milieu.
   (f) Tester `point_milieu` avec $f(x) = \sqrt{1-x^2}$, $a = -0.5, b = 0.5$ et $n = 10^k$, k variant de 1 à 6, puis remplir manuellement le tableau ci-dessous :

n	erreur	temps (sec.)
10		
100		
1000		
10000		
100000		
1000000		

   (g) En python, recréer automatiquement le tableau obtenu au moyen d'une boucle `for`. Utiliser les fonction `print` pour voir le tableau à l'écran et `write` pour écrire le tableau dans un fichier texte. Voir documentation à l'adresse `https://docs.python.org/3/tutorial/inputoutput.html`. Exemple de tableau produit automatiquement en python :

```
         n |   erreur    | temps (sec.)
---------------------------------------
        10 | 8.33333e-04 | 1.80000e-05
       100 | 8.33333e-06 | 4.10000e-05
      1000 | 8.33333e-08 | 3.98000e-04
     10000 | 8.33337e-10 | 3.95500e-03
    100000 | 8.33034e-12 | 4.02760e-02
   1000000 | 8.37108e-14 | 3.99448e-01
```

3. Dans la méthode du point milieu, on a approximé la fonction f sur l'intervalle $[x_{k-1}, x_k]$ par la fonction constante prenant la même valeur que f en c_k, le milieu de $[x_{k-1}, x_k]$; dans la **méthode du trapèze**, on approxime f sur cet intervalle par la fonction affine qui prend les mêmes valeurs que f en x_{k-1} et x_k. Reprendre le travail précédent, mais avec la méthode du trapèze.
4. Dans la méthode du point milieu, on a approximé f sur l'intervalle $[x_{k-1}, x_k]$ par une fonction constante - polynôme de degré 0 ; dans la méthode du trapèze, on a approximé f sur cet intervalle par une fonction affine - polynôme de degré 1 ; dans la **méthode de Simpson**, on approxime 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.
   Reprendre le travail précédent, mais avec la méthode de Simpson.
5. Comparer les trois méthodes précédentes.
6. La **méthode de Monte-Carlo** est intéressante pour calculer des surfaces, des volumes, etc., et on pourrait donc l'appliquer au problème ci-dessus. Présentons la méthode en modifiant un peu la situation précédente [Python : module `numpy.random` ; Maths : distribution de probabilité].
   (a) A l'aide de `matplotlib.pyplot`, dessiner le cercle unité (centré à l'origine et de rayon 1). Quelle est la surface du disque unité ?
   (b) En ligne de commande, à l'aide de la fonction `numpy.random.rand()`, générer quelques valeurs aléatoires suivant la distribution de loi uniforme sur $[0, 1]$. Même exercice avec une distribution de loi uniforme sur $[-1, 1]$.
   (c) Générer $N = 1000$ points suivant la distribution uniforme sur le carré $[-1, 1]^2$. Faire apparaitre ces N points sur le graphique précédent, en rouge les points intérieurs au disque unité, et en vert les points extérieurs au disque.
   (d) Compter le nombre I de points intérieurs et le nombre E de points extérieurs. La méthode de Monte-Carlo consiste à prendre le rapport $\frac{I}{N}$ comme approximation de la surface S du disque.
   (e) Calculer l'erreur commise $|\frac{I}{N} - S|$.
   (f) Utiliser la fonction `clock()` du module `time` pour mesurer le temps de calcul de votre intégrale.
   (g) Ecrire une fonction python `Monte_Carlo` qui prend en arguments un entier N et qui renvoie une approximation de la surface du disque unité au moyen de la méthode de Monte-Carlo.
   (h) Tester `Monte_Carlo` avec $N = 10^k$, k variant de 1 à 6, puis créer automatiquement, et l'enregistrer dans un fichier texte, un tableau de résutats du modèle ci-dessous :

N	erreur	temps (sec.)
10		
100		
1000		
10000		
100000		
1000000		