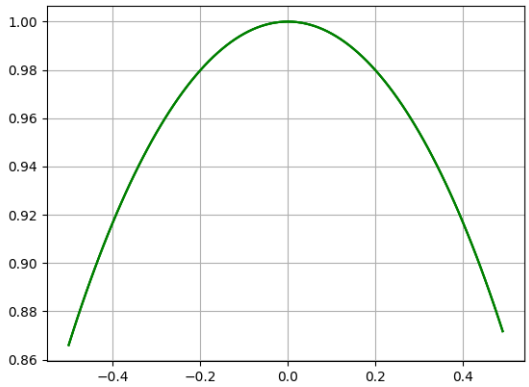
TP2
Intégration numérique

BELOUCIF Malik LAICHE Issam

13 décembre 2017

Dans ce TP élaboré par Mr Cardinal,nous illustrons plusieurs méthodes pour calculer, approximativement, l'intégrale d'une fonction sur un intervalle borné et fermé.

1. **Définition de la fonction f**



Nous avons commencé par définir la fonction $f(x)$ à l'aide de la fonction `sqrt()` du module `math`. Après avoir représenté graphiquement la fonction sur l'intervalle $[-0.5, 0.5]$ nous avons calculé l'integrale suivante à la main :

$$I = \int_{-0.5}^{0.5} \sqrt{1 - x^2}\, \mathrm{d}x$$

2. **Méthode du point milieu**

Nous avons calculé l'intégrale I à l'aide de la **méthode du point milieu**. Ensuite, on a défini une subdivision régulière de l'intervalle d'intégration $[0, 1]$ en n= 4 parts égales.

Puis on a défini les milieux de ces sous-intervalles. Avec ces données on calcule l'air sk du rectanggle de base $[xk-1, xk]$ et de hauteur $f(ck)$. L'air d'un rectangle est égale à sa longueur multiplié par sa largueur, donc on a $S = b * h$ avec b la base et h la hauteur du rectangle.

En effet,nous avons dû calculer l'erreur entre l'intégrale et l'approximation qu'on a faite. Alors nous avons calculé la valeur absolue de $S4$ puis nous l'avons soutrait à l'intégrale I.

Nous avons utilisé la fonction `clock()` du module `time` pour mesurer le temps de calcul de notre intégrale.

Enfin, nous avons défini la fonction permettant d'utiliser la *méthode du point milieu*.Les valeur obtenu sont rangé dans le tableau suivant :

n	erreur	temps (sec.)
10	0.000480384	3.6e-05
100	4.811178e-06	0.000184
1000	4.81125e-08	0.001663
10000	4.8113e-10	0.01584
100000	4.81315e-12	0.161942
1000000	5.06262e-14	1.57448

3. Méthode du trapèze

Pour la *méthode du trapèze*,nous avons trouvé des approximation `f` sur l'intervalle $[xk-1, xk]$ par la fonction qui prend les mêmes valeurs que `f` en x_{k-1} et x_k.

Nous avons essayé de calculer l'intégrale à l'aide de la *méthode du trapèze*. Pour cela, nous avons défini la fonction *trapeze* qui prend en arguments une fonction `f`, des bornes `a` et `b` et un entier `n` et qui renvoie l'intégrale approchée de `f` dans $[a, b]$.

Nous avons pu rempli le tableau ci-dessous à l'aide des resultats obtenu :

n	erreur	temps (sec.)
10	0.000961402	2.1e-05
100	9062242e-06	7.8e-05
1000	9.6225e-08	0.00077
10000	9.6225e-10	0.007611
100000	9.63307e-12	0.076049
1000000	1.11022e-13	0.744644

4. Méthode de Simpson

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

Ces données sont ordonnées dans le tableau ci-dessous :

n	erreur	temps (sec.)
10	2.11626e-07	2.8e-05
100	2.13807e-11	0.000185
1000	1.33227e-15	0.002373
10000	6.76126e-14	0.02393
100000	6.27609e-13	0.239471
1000000	1.12086e-11	2.40692

5. Comparaison

Les trois méthodes **point milieu, trapèze** et **simpson** nous permettent de calculer une approximation d'intégrale d'une fonction sur un intervalle.

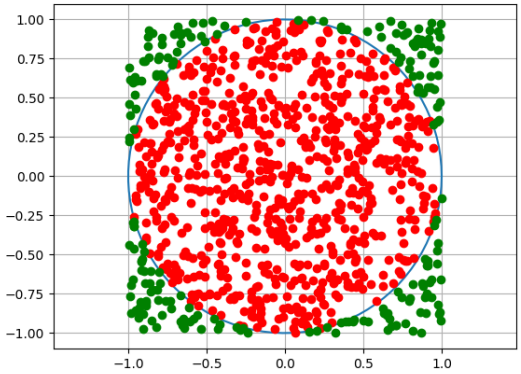
Il est vrai qu'il y a certaines différences entre chacune de ces méthodes. Dans la *méthode du point milieu* on approxime `f` par la fonction constante , dans la *méthode du Trapèze* on approxime par la fonction affine et dans la *méthode de Simpson* on approxime par la fonction du polynôme de degré 2.

On peut constater également que l'erreur est différente pour chaque méthode. On a pu en déduire que la *méthode de Simpson* est plus efficace que les deux autres méthodes.

6. Méthode de Monte-Carlo

A l'aide de la fonction *numpy.random.rand()*, nous avons pu acquérir plusieurs valeurs aléatoirement représentées en `bleu` pour celles inférieures au disque unité et en `jaune` pour celles qui lui sont supérieures.

Ensuite, nous avons généré `1000` points qui suivent la distribution uniforme sur le carré $[-1, 1]^2$ et nous avons fait apparaître les points inférieurs au disque unité en rouge et les points extérieurs au disque en vert :



Nous avons compté le nombre `I` de points intérieurs et le nombre `E` de points extérieurs.

Effectivement, la *méthode de Monte Carlo* consiste à prendre le rapport $\frac{I}{N}$ comme approximation de la surface `S` du disque. Ainsi nous avons calculé la valeur absolue du rapport $\frac{I}{N}$ soustrait à S pour savoir l'erreur commise par la fonction *Monte Carlo*. On a utilisé `clock()` pour mesurer le temps de calcul.

On a testé la fonction *Monte Carlo* avec $N = 10^k$, k variant de 1 à 6.

Le tableau suivant changera à chaque fois que le programme sera exécuté.

n	erreur	temps (sec.)
10	0.0975927	7.8e-05
100	0.0975927	0.000256
1000	0.0975927	0.002503
10000	0.0975927	0.02528
100000	0.0975927	0.246219
1000000	0.0975927	2.47361

En somme, nous avons défini une fonction *monte_carlo_2* pour le calcul du volume de la boule unité.