3 T.P.3 : Dérivation et intégration

I. Dérivation

Soit $a \in I$ où I est un intervalle de $\mathbb{R}$ et $f : I \to \mathbb{R}$ supposée de classe $\mathcal{C}^1$ sur I.

Pour calculer une valeur approchée de $f'(a)$, on retiendra trois méthodes :

$$f'(a) \approx \frac{f(a+h)-f(a)}{h} \quad \| \quad f'(a) \approx \frac{f(a)-f(a-h)}{h} \quad \| \quad f'(a) \approx \frac{f(a+h)-f(a-h)}{2h}$$

Une méthode « naïve » pour évaluer la dérivée est donc de tabuler les valeurs de l'une de ces trois formules pour différentes valeurs de h. On s'attend à ce que, plus la valeur de h est proche de 0, plus l'approximation de la dérivée soit précise.

Exercice I.1 :

① Écrire trois fonctions `derivee1`, `derivee2` et `derivee3` dont les variables d'entrée sont `f`, `a` et `h` et qui retournent chacune une valeur approchée de la dérivée de f en a selon les trois formules proposées ci-dessus.

② Écrire une fonction `graphe1(f,fpr,x1,x2,h)` qui, dans deux fenêtres superposées, trace sur l'intervalle $[x1, x2]$ respectivement la courbe de f et, dans la seconde fenêtre, les courbes de f' (notée `fpr`) ainsi que celles obtenues grâce à `derivee1` et `derivee3`, avec un pas de h.
Commenter à l'appui de fonctions f de votre choix.

Exercice I.2 :

Soit f la fonction définie sur $\mathbb{R}$ par : $f(x) = x^4$. Calculons sa dérivée numériquement en $a = 1$ dont nous savons qu'elle vaut 4.

① Écrire une fonction Python qui retourne, pour `derivee1` et `derivee3`, l'erreur commise $|E(h)| = |4 - \texttt{derivee(f,1,h)}|$ pour $h = 10^{-p}$ où $p \in [\![1, 16]\!]$. Qu'observez-vous ?

② Utiliser les logarithmes décimaux pour retourner une représentation graphique de $|E(h)|$ en fonction de h. Pour quelle valeur de h l'approximation est-elle la meilleure ?

Application : Recherche de zéros par méthode de Newton.

Cette méthode consiste à approximer le graphe de la fonction au voisinage du point x_0 par la tangente en ce point. Soit $f \in \mathcal{C}^2([a, b])$ avec $x_0 \in [a, b]$ et $f'(x)$ strictement positif ou négatif sur cet intervalle.
On rappelle qu'un un point x_0 la tangente a pour équation :

$$y = f(x_0) + f'(x_0) \cdot (x - x_0)$$

Si $f'(x_0) \neq 0$ le point d'intersection de la tangente avec l'axe des x a pour abscisse :

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Soit la suite $(x_n)_{n\geq 0}$ définie par :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

On admettra que si f est strictement monotone, convexe ou concave sur $[a, b]$ et s'annulant sur l'intervalle $[a, b]$, alors la suite (x_n) converge très rapidement vers une solution de $f(x) = 0$.

Exercice I.3 :

① Écrire une fonction `newton` dont les variables d'entrée sont la fonction f, une valeur de départ $x_0 \in [a, b]$ et le nombre d'itérations souhaitées n et qui retourne grâce à la méthode de Newton une valeur approchée du zéro de la fonction f sur l'intervalle $[a, b]$.

② On cherche à résoudre l'équation $x - 2 + \frac{1}{2}\ln(x) = 0$ dont on sait qu'elle admet une unique solution α sur $[1, 2]$.

- Utiliser l'algorithme de dichotomie pour retourner une valeur approchée à 10^{-10} près de α. Combien d'itérations ont-elles été nécessaire ?
- Appeler `newton(f,1,n)` pour différentes valeurs de n, en commençant à $n = 2$. A partir de quelles valeurs de n obtient-on une valeur approchée à $1e - 10$ près ?

II. Intégration

Soit f une fonction à valeurs réelles, continue ou continue par morceaux sur un intervalle $[a, b]$. On appellera **subdivision régulière** de l'intervalle $[a, b]$ la suite $\sigma = (c_k)_{k\geq 0}$ définie par

$$c_k = a + k\frac{b-a}{n} = a + kh$$

On dira que $h = \dfrac{b-a}{n}$ est son **pas**.

Pour calculer une valeur approchée de l'intégrale $I = \displaystyle\int_a^b f(t)dt$, on retiendra trois méthodes :

Méth. des rectangles à gauche	Méth. des rectangles à droite	Méth. des trapèzes
$I \approx \displaystyle\sum_{k=0}^{n-1} h \cdot f(c_k) = h\sum_{k=0}^{n-1} f(c_k)$	$I \approx h\displaystyle\sum_{k=1}^{n} f(c_k)$	$I \approx \displaystyle\sum_{k=0}^{n-1} h\frac{f(c_k)+f(c_{k+1})}{2}$

Exercice II.1 :

① Écrire une fonction `IntMethRectangleD(f,a,b,n)` qui retourne une valeur approchée de I par la méthode des rectangles à droite.

② Écrire une fonction `IntMethRectangleG(f,a,b,n)` qui retourne une valeur approchée de I par la méthode des rectangles à gauche.

③ Écrire une fonction `IntMethTrapezes(f,a,b,n)` qui retourne une valeur approchée de I par la méthode des trapèzes.

✎*Remarque* : Python dispose d'une bibliothèque dont les méthodes permettent d'obtenir directement une approximation d'une intégrale donnée. Il s'agit du module `integrate` de la bibliothèque `scipy`.Pour pouvoir l'importer, on exécutera :

```
from scipy import integrate
```

Alors `integrate.quad(f,a,b)` retourne un couple dont la première valeur est la valeur approchée de I et la seconde est l'erreur commise.

Exercice II.2 :

Vérifier la validité des approximations obtenues en confrontant sur des exemples de votre choix les méthodes programmées dans l'exercice 1 et les valeurs retournées par la méthode `integrate` de la bibliothèque `scipy` (on pourra s'inspirer des fonctions proposées dans le notebook « td0BCPST2d-revisionsIntegration.ipynb » de l'été).

La méthode de Monte Carlo :

Nous supposons f continue et positive sur $[a, b]$ et nous posons $[c, d] = f([a, b])$. La courbe s'inscrit dans un rectangle dont la surface est connue et vaut $Sr = (b - a) * d$.
L'idée de cette méthode repose une succession de m tirages aléatoires indépendants de points pris au hasard dans ce rectangle à l'aide d'une loi uniforme (modélisée par la fonction `random.uniform()`). On dénombre les tirages qui retournent des points situés sous la courbe et on considère que le rapport entre ce nombre et le nombre de tirages tend vers $\frac{I}{S_r}$ quand m tend vers l'infini.
On retiendra que cette méthode est facile à programmer mais peu efficace.

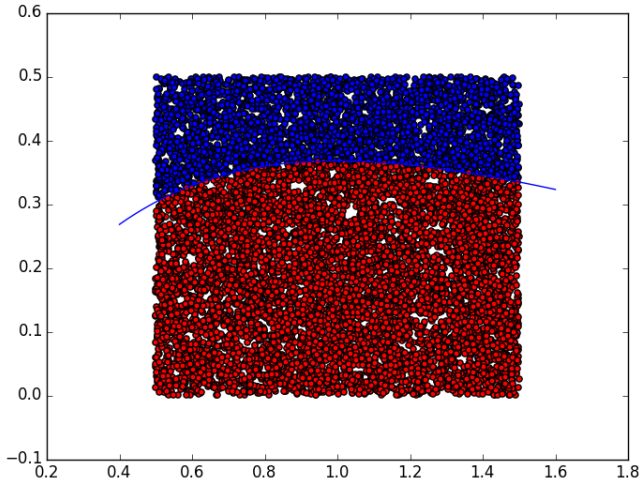

FIGURE 1 – Méthode de Monté Carlo