TP 19 Python – Méthodes numériques

Ce TP vise à apprendre deux algorithmes importants du programme, qui servent à **approcher numériquement** des valeurs.

- La première permet d'obtenir la valeur approchée d'une **intégrale** sur un segment, à l'aide des sommes de Riemann. C'est la **méthode des rectangles**.
- La seconde permet d'obtenir la valeur approchée d'une **solution d'une équation** du type $f(x) = 0$, et est appelée la **méthode de Newton**.

1. Méthode des rectangles

Définition

La **méthode des rectangles** est simplement l'algorithme qui permet de calculer une valeur approchée d'une intégrale sur un segment en calculant la valeur de la n-ième **somme de Riemann** associée.

Exercice 1.

1. Soient $a, b \in \mathbb{R}$, $a < b$. Soit $f : [a, b] \to \mathbb{R}$ une fonction continue. Retrouver (sans le cours, à l'aide d'un dessin au brouillon par exemple) l'expression de la n-ième somme de Riemann pour f sur $[a, b]$, en utilisant les rectangles **à gauche**.
2. En déduire une fonction `rectangles_gauche(f,a,b,n)` qui prend en argument une fonction f définie sur un intervalle, deux réels a, b de cet intervalle vérifiant $a < b$, et un entier $n \in \mathbb{N}^*$, et qui renvoie une valeur approchée de $\displaystyle\int_a^b f(t)dt$.
3. Écrire de même une fonction `rectangles_droite(f,a,b,n)`.
4. Tester les deux fonctions pour donner une valeur approchée de $\displaystyle\int_0^{\frac{\pi}{2}} \cos(t)dt$, et vérifier les valeurs obtenues en calculant mathématiquement la valeur exacte de cette intégrale.

Fonction à retenir (à savoir retrouver) :

Exercice 2. Nous souhaitons reprogrammer la fonction logarithme népérien à partir des opérations arithmétiques de base. Pour cela, nous utilisons la relation :

$$\forall x > 0, \quad \ln(x) = \int_1^x \frac{1}{t} dt$$

1. Justifier rapidement la relation ci-dessus.
2. Écrire une fonction `ln` qui calcule le logarithme népérien par la méthode des rectangles à droite avec 1000 rectangles.
3. Tracer sur un même graphique les courbes représentatives de votre fonction `ln` et de la fonction `log` du module `numpy`, et comparer.

Exercice 3. On considère la fonction définie par $f(x) = \dfrac{\sin(x)}{x}$.

1. Justifier que cette fonction admet un prolongement par continuité en 0.
2. Définir une fonction Python `f(x)` qui correspond à ce prolongement.
3. Calculer une valeur approchée de $\displaystyle\int_0^{2\pi} \frac{\sin(x)}{x} dx$.

2. Méthode de Newton

Pour déterminer une valeur approchée d'une solution pour une équation de la forme $f(x) = 0$, on connaît déjà la méthode de .....
On apprend dans cette section une seconde méthode au programme pour la résolution d'équations $f(x) = 0$: la méthode de Newton.

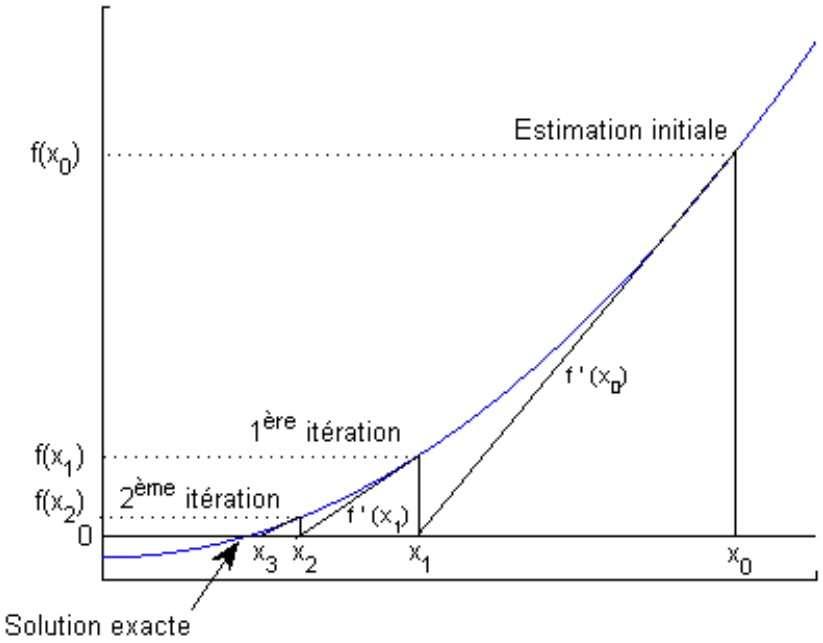
Théorème (*HP*)

Soit $f : I \to \mathbb{R}$ une fonction dérivable sur un intervalle I, et qui s'annule en un certain $a \in I$.

On définit une suite $(x_n)_{n\in\mathbb{N}}$ par $x_0 \in I$, et pour tout $n \in \mathbb{N}$: $\boxed{x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}}$.

Si f est de classe $\mathcal{C}^2$ et $f'(a) \neq 0$, alors pour tout choix de x_0 suffisamment proche de a, la suite (x_n) **converge vers** a.

À retenir : **Sous de bonnes hypothèses, la suite définie par** $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$ **converge vers une solution de** $f(x) = 0$.



Intuition de preuve : Pour y proche de x, on a $f(y) \approx f(x) + (y-x)f'(x)$ (équation de la tangente).
Ainsi, $y \approx \dfrac{f(y)-f(x)}{f'(x)} + x$.
C'est pourquoi, si on souhaite s'approcher d'un a qui vérifie $f(a) = 0$, il est naturel de poser :
$y = \dfrac{0-f(x)}{f'(x)} + x = x - \dfrac{f(x)}{f'(x)}$.

Exercice 4.
1. Écrire (mathématiquement) la suite donnée par la méthode de Newton pour la fonction $f = \sin$, en partant du point $x_0 = 2$.
2. Écrire une fonction Python qui calcule les termes successifs de la suite ci-dessus. Quelle est la limite de la suite ?
3. Après avoir regardé les 10 premiers termes de la suite : combien d'étapes faut-il pour approcher π à 10^{-15} près ? Comparer avec la méthode de dichotomie (aller voir le cours !).
4. Reprendre les questions 1 et 2 avec $x_0 = 1,5$, puis avec $x_0 = 1,57$. Que se passe-t-il ?
5. Mathématiquement, que se passe-t-il avec $x_0 = \dfrac{\pi}{2}$? Expliquer le phénomène de la question précédente.

Exercice 5.
1. Écrire une fonction Python `Newton(x0, f, fprime, n)` qui prend en argument un flottant `x0`, une fonction `f`, sa dérivée `fprime` et un entier `n`, et qui calcule le terme de rang n de la suite donnée par la méthode de Newton.
2. **Variante avec seuil :** Écrire une fonction Python `Newton_seuil(x0, f, fprime, eps)` qui fait la même chose, mais *en s'arrêtant au premier rang* n tel que $|x_n - x_{n-1}| < \varepsilon$.

Fonctions à savoir retrouver : (le principe de l'algorithme sera toujours rappelé dans un sujet)

Exercice 6. Déterminer des valeurs approchées de **toutes** les racines réelles de la fonction polynomiale $f(x) = x^3 - 2x^2 + 1$. On commencera par écrire des fonctions Python correspondant à f et à sa dérivée.
Que se passe-t-il si on choisit $x_0 = 0$?

Exercice 7. Avec la même méthode que précédemment, obtenir des valeurs approchées des nombres suivants, en utilisant seulement les opérations arithmétiques élémentaires.

a) $\sqrt{2}$, à 10^{-15} près

b) $\sqrt[3]{7}$, à 10^{-12} près

On commencera par écrire des équations vérifiées par ces nombres, de la forme $f(x) = 0$.

Exercice 8. On reprend la fonction $f(x) = x^3 - 2x^2 + 1$.
Pour x_0 qui varie entre -1 et 3 avec un pas de 0.01, afficher le couple formé par x_0 en abscisse, et la racine trouvée par la méthode de Newton en ordonnée. Commenter.

Exercice 9. Comparer les avantages et inconvénients de la méthode de Newton et de la dichotomie.
On pensera notamment aux hypothèses sur f, et à la *vitesse de convergence* de l'algorithme.