
Outils numériques
TP4 : Premiers algorithmes

Les exercices de ce TP, dont le but est de vous familiariser avec l'implémentation d'algorithmes, sont liés : il est recommandé de travailler dans un unique fichier Python.

Soit $f : \mathbb{R} \rightarrow \mathbb{R}$ une fonction de classe $\mathcal{C}^1$ sur un intervalle $[a, b]$. On suppose que f admet un zéro $x_\star \in [a, b]$ que l'on va chercher à approcher. On va pour cela implémenter différentes méthodes numériques. Dans les algorithmes décrits ci-dessous, $\varepsilon > 0$ définit la précision souhaitée. On note également $N_{\max}$ le nombre maximal d'itérations que l'on autorise.

Toutes les fonctions Python que vous allez implémenter proposeront les mêmes variables de sortie :

- **sol** l'approximation trouvée de $x_\star$;
- **itersol** une liste comportant les *itérées* x_n pour n compris entre 0 et **iter** ;
- **iteres** une liste comportant les *résidus* $|f(x_n)|$ pour n compris entre 0 et **iter** (on parle de *résidu* car la solution vérifie $|f(x_\star)| = 0$: on souhaite donc que les résidus soient de plus en plus petits) ;
- **iter** l'itération à laquelle l'approximation a été trouvée.

Exercice 1 : Méthode de la dichotomie. On suppose que $f(a)f(b) \leq 0$. L'algorithme de dichotomie construit deux suites (a_n) et (b_n) , dont on peut montrer qu'elles sont adjacentes, et une suite (x_n) de la façon suivante :

Algorithme 1 : Méthode de la dichotomie

```
Input :  $a, b, f, \varepsilon, N_{\max}$   
 $a_0 \leftarrow a, b_0 \leftarrow b, x_0 \leftarrow (a_0 + b_0)/2, n \leftarrow 0$ ;  
while  $|f(x_n)| \geq \varepsilon$  et  $n \leq N_{\max}$  do  
    if  $f(a_n)f(x_n) \leq 0$  then  
         $b_{n+1} \leftarrow x_n$  et  $a_{n+1} \leftarrow a_n$ ;  
    else  
         $a_{n+1} \leftarrow x_n$  et  $b_{n+1} \leftarrow b_n$ ;  
     $x_{n+1} \leftarrow (a_{n+1} + b_{n+1})/2$ ;  
     $n \leftarrow n + 1$ ;
```

1. Programmer en Python une fonction `dichotomie()` implémentant l'algorithme de dichotomie. La sortie devra correspondre à ce qui est demandé dans le préambule et elle prendra comme arguments :
 - **a, b** les bornes de l'intervalle ;
 - **f** la fonction dont on recherche le zéro ;
 - **eps** la précision souhaitée ;
 - **itermax** le nombre maximal d'itérations.
2. Trouver une solution approchée à 10^{-5} près de la solution positive de l'équation $x^2 = 3$.

Exercice 2 : Méthode de Newton. La méthode de Newton est décrite par l'algorithme suivant :

Algorithme 2 : Méthode de la dichotomie

Input : $x_0, f, f', \varepsilon, N_{\max}$
 $n \leftarrow 0$;
while $|f(x_n)| \geq \varepsilon$ et $n \leq N_{\max}$ **do**
 $x_{n+1} \leftarrow x_n - f(x_n)/f'(x_n)$;
 $n \leftarrow n + 1$;

1. Programmer en Python une fonction `newton()` implémentant l'algorithme de dichotomie. La sortie devra correspondre à ce qui est demandé dans le préambule et elle prendra comme arguments :
 - `x0` le point de départ ;
 - `f` la fonction dont on recherche le zéro ;
 - `fprime` la dérivée de `f` ;
 - `eps` la précision souhaitée ;
 - `itermax` le nombre maximal d'itérations.
2. Trouver une solution approchée à 10^{-5} près de la solution positive de l'équation $x^2 = 3$.

Exercice 3 : Méthode de la sécante. La méthode de Newton dans l'exercice précédent nécessite de calculer la dérivée de f à chaque itération, ce qui peut être cher si la fonction f est elle-même issue d'un autre algorithme. Une variante pour éviter ce calcul est la méthode de la sécante, que l'on présente dans l'algorithme suivant.

Algorithme 3 : Méthode de la dichotomie

Input : $x_0, x_1, f, \varepsilon, N_{\max}$
 $n \leftarrow 0$;
while $|f(x_n)| \geq \varepsilon$ et $n \leq N_{\max}$ **do**
 $x_{n+1} \leftarrow x_n - f(x_n) \times \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$;
 $n \leftarrow n + 1$;

1. Programmer en Python une fonction `secante()` implémentant l'algorithme de dichotomie. La sortie devra correspondre à ce qui est demandé dans le préambule. La fonction initialisera x_0 à a et x_1 à b et prendra comme arguments :
 - `a, b` les bornes de l'intervalle ;
 - `f` la fonction dont on recherche le zéro ;
 - `eps` la précision souhaitée ;
 - `itermax` le nombre maximal d'itérations.

Attention, ici la méthode a besoin de deux initialisations x_0 et x_1 , que l'on initialise ici respectivement à a et b , mais d'autres choix sont possibles.

2. Trouver une solution approchée à 10^{-5} près de la solution positive de l'équation $x^2 = 3$.

Exercice 4 : Vitesse de convergence.

1. Afficher sur un même graphe, en fonction de l'itération n , les *résidus* successifs $|f(x_n)|$ obtenus avec les trois méthodes implémentées. On utilisera une échelle linéaire : cela vous semble-t-il convaincant ? Si non, proposer une autre méthode de visualisation.
2. Comparer les vitesses de convergence des trois algorithmes, jouer avec la précision demandée et commenter.
3. Reprendre les questions avec les *erreurs* $|x_n - x_\star|$ où $x_\star$ est la solution exacte de $x^2 = 3$ dans $[1, 2]$. Commenter.