

TRAVAUX PRATIQUES N° 2 : Algorithmes d'optimisation 1D

Préambule : Les questions notées Pour aller plus loin, peuvent être omise en première lecture.

1 Méthode de la dichotomie

Dans cette partie, nous allons chercher une racine d'une fonction f continue sur un intervalle $[a, b] \subset \mathbb{R}$ telle que $f(a)f(b) \leq 0$ en utilisant la méthode de la dichotomie¹ ( : *bisection method*).

Algorithme 1 : Dichotomie pour trouver un zéro d'une fonction (*i.e.*, x t.q. $f(x) \approx 0$)

```
input  :  $f, a, b$ 
init   :  $x^\downarrow = \min(a, b), x^\uparrow = \max(a, b)$ 
param  :  $\epsilon$ 
while  $x^\uparrow - x^\downarrow > \epsilon$  do
     $\bar{x} \leftarrow \frac{1}{2}(x^\uparrow + x^\downarrow)$                                 // Calcul du milieu du segment courant
    if  $f(x^\downarrow) \cdot f(\bar{x}) \leq 0$  then                          // Il y a une racine entre  $x^\downarrow$  et  $\bar{x}$ 
         $x^\uparrow \leftarrow \bar{x}$ 
    else                                                         // Il y a une racine entre  $\bar{x}$  et  $x^\uparrow$ 
         $x^\downarrow \leftarrow \bar{x}$ 
return  $\bar{x}$ 
```

QUESTION 1. (Arrêter un algorithme) Proposer un critère de convergence alternatif et implémenter l'algorithme avec ce critère. On modifiera aussi l'algorithme pour utiliser une boucle **for** et **break**² pour fixer un nombre d'itérations maximum et éviter ainsi une boucle infinie.

QUESTION 2. (Application de la dichotomie) Utiliser cet algorithme pour trouver le minimum global de la fonction $x \mapsto x^4 - (x - 1)^2 + 1$. Pour visualiser la convergence, on tracera la taille de l'intervalle courant en fonction de l'indice de l'itération (on pourra modifier l'algorithme pour sortir la liste des solutions courantes, en utilisant **append** pour ajouter un élément à une liste). Quelle est l'influence de la précision du critère de convergence sur le nombre d'itérations effectuées ? Comparer vos résultats avec l'implémentation de **scipy.optimize.bisect**.


Pour aller plus loin : Il est informatif de visualiser la convergence en fonction du temps de calcul et non de l'itération. Proposez une telle visualisation, par exemple avec³ :

-
1. Voir https://fr.wikipedia.org/wiki/M%C3%A9thode_de_dichotomie
 2. <https://docs.python.org/3/tutorial/controlflow.html#break-and-continue-statements-and-else-clauses-on-loops>
 3. <https://docs.python.org/3/library/time.html>

```
import time
t0 = time.perf_counter()
... # Instruction dont on veut mesurer le temps d'exécution
t1 = time.perf_counter()
print(t1-t0)
```

D'autres visualisations sont possibles, voir par exemple les graphiques générés par `Benchopt`⁴.

2 Recherche de minimum local par la méthode du nombre d'or

La méthode du nombre d'or ( : *golden-section search*) permet de trouver le minimum d'une fonction f continue et unimodale sur l'intervalle $[a, b] \subset \mathbb{R}$. On note par la suite $\varphi = \frac{1+\sqrt{5}}{2}$ le nombre d'or.

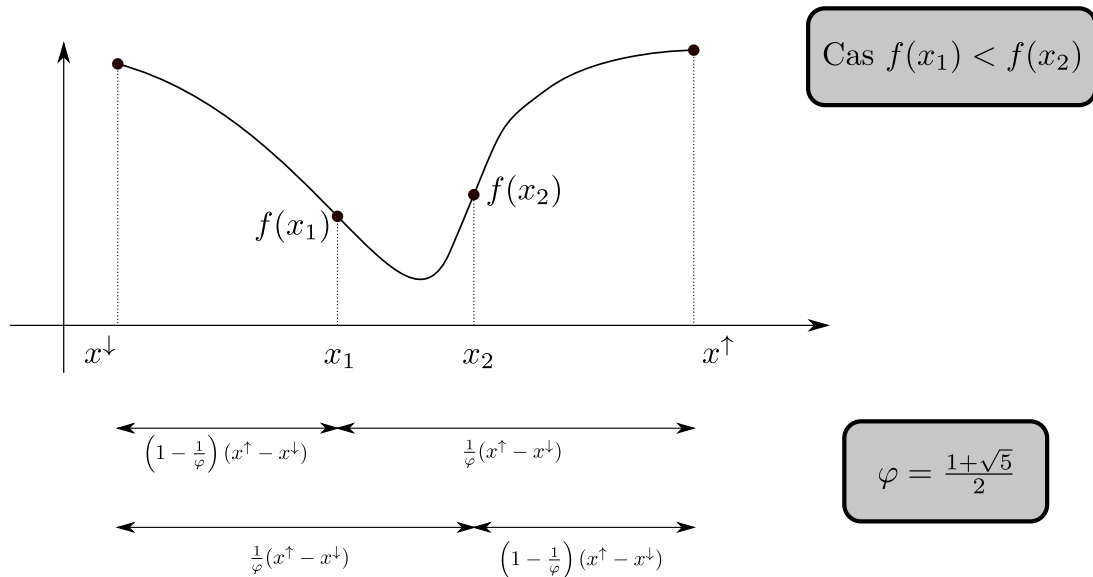


FIGURE 1 – Illustration des grandeurs utiles dans la méthode du nombre d'or

QUESTION 3. (Méthode du nombre d'or et extremum) Implémenter la méthode du nombre d'or pour une fonction f quelconque en entrée, en utilisant une boucle `for` et un `break`.

QUESTION 4. (Applications) Tester la méthode du nombre d'or sur différentes fonctions :

- $x \mapsto x^2 + 2$ sur $[-5, 5]$ puis sur $[-5, 20]$,
- $f : x \mapsto x^6 + 3 \exp(-x^2) + \frac{1}{2} \sin\left(\frac{5x}{2}\right)$ sur $[0.3, 1.5]$ (on fera une vérification visuelle ici des hypothèses).

Comparer avec l'implémentation de `scipy.optimize.golden`.

QUESTION 5. (Influence de l'initialisation) Reprendre la dernière fonction sur l'intervalle $[-\frac{3}{2}, \frac{3}{2}]$ en prenant différentes initialisations plusieurs fois. Que remarquez-vous ?

Chaque algorithme a des hypothèses différentes et tous ne vont pas forcément converger vers le même minimum suivant les initialisations. Pour vérifier que nos algorithmes convergent bien, on peut utiliser des méthodes déjà implémentées.

4. <https://benchopt.github.io/results/>

Algorithme 2 : La méthode du nombre d'or

```
input  :  $f, a, b$ 
init   :  $x^\downarrow = \min(a, b), x^\uparrow = \max(a, b)$ 
         $x_1 \leftarrow \frac{1}{\varphi}x^\downarrow + \left(1 - \frac{1}{\varphi}\right)x^\uparrow, f_1 \leftarrow f(x_1)$ 
         $x_2 \leftarrow \frac{1}{\varphi}x^\uparrow + \left(1 - \frac{1}{\varphi}\right)x^\downarrow, f_2 \leftarrow f(x_2)$ 
param :  $\epsilon$ 
while  $x^\uparrow - x^\downarrow > \epsilon$  do
    if  $f_1 < f_2$  then
         $x^\uparrow \leftarrow x_2$ 
         $x_2 \leftarrow x_1$ 
         $f_2 \leftarrow f_1$ 
         $x_1 \leftarrow \frac{1}{\varphi}x^\downarrow + \left(1 - \frac{1}{\varphi}\right)x^\uparrow$ 
         $f_1 \leftarrow f(x_1)$ 
    else
         $x^\downarrow \leftarrow x_1$ 
         $x_1 \leftarrow x_2$ 
         $f_1 \leftarrow f_2$ 
         $x_2 \leftarrow \frac{1}{\varphi}x^\uparrow + \left(1 - \frac{1}{\varphi}\right)x^\downarrow$ 
         $f_2 \leftarrow f(x_2)$ 
return  $\frac{x^\uparrow + x^\downarrow}{2}$ 
```

3 Optimisation avec scipy

Le package `scipy` a notamment une fonction `minimize` dans son module `optimize`⁵. Afficher la documentation de cette fonction dans le programme python à l'aide de la commande `help` (on pourra utiliser le raccourci ?).

QUESTION 6. (Avec `scipy`) Regarder attentivement :

- les méthodes disponibles,
- l'initialisation (en déduire un avantage de ces méthodes par rapport à la dichotomie),
- la tolérance,
- le rappel (🇬🇧 : *callback*).

Reprendre la troisième fonction de la question précédente et utiliser différents algorithmes disponibles, on comparera les résultats et les temps de calcul pour une tolérance de 10^{-1} , 10^{-5} puis 10^{-9} avec la même initialisation (les temps de calculs pourront être moyennés sur plusieurs répétitions pour plus de précision). Où se situe la dichotomie par rapport aux autres algorithmes ?

QUESTION 7. (Pour aller plus loin avec les *callback*) Si vous étudiez bien la documentation de cette fonction `minimize`, l'utilisateur n'a pas d'accès direct au numéro de l'itération (seulement aux itérés générés). Utilisez le `callback` (et `args` si vous le souhaitez) pour afficher le numéro de l'itération k , le point x_k considéré et la valeur de $f(x_k)$.

5. <https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html>

```
# Exemple d'utilisation de callback
iterations = [] # liste pour stocker les itérés

def callback(x): # callback ne prend que l'itéré courant
    iterations.append(x) # on ajoute l'itéré courant

scipy.optimize.minimize(ma_fonction, x0, callback=callback)
print(iterations) # liste des itérés
```

Stockez ces résultats et visualisez la trajectoire engendrée par $(f(x_k))_k$.

Dans la dernière fonction considérée, l'hypothèse d'unimodalité n'était pas vérifiée sur l'intervalle considéré. Les méthodes d'optimisation en général ne vont pas trouver le minimum global, mais un minimum local. Sortir des minima locaux est un problème qui requiert des techniques plus avancées⁶.

QUESTION 8. Utilisez la méthode par défaut de `scipy` avec une grille de points d'initialisation sur la fonction

$$g : x \mapsto \exp\left(\frac{1}{2}x^2\right) - x^3 + 10 \cos(x),$$

pour $x \in [-3, 3]$. Combien de bassin d'attractions observez-vous ? Associez leur une couleur et proposez une visualisation de la correspondance entre le point initial et la solution trouvée.

6. Par exemple : https://fr.wikipedia.org/wiki/Algorithme_du_gradient_stochastique