Partie 3

Ingénierie Numérique & Simulation

Chapitre 2– Problèmes stationnaires – Résolution numérique de l'équation $f(x)=0$

Informatique

Fiche PT

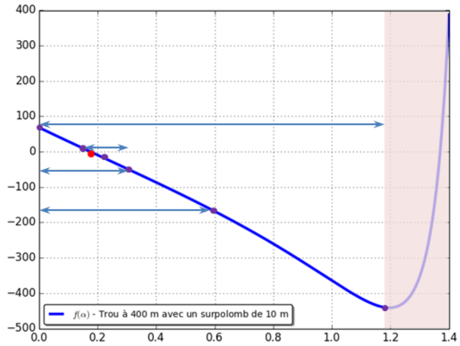
Principe de la méthode de dichotomie

Théorème **Théorème des valeurs intermédiaires**

Soit f une fonction définie et continue sur l'intervalle $[a,b]$ à valeur dans $\mathbb{R}$. Pour tout $u \in [f(a), f(b)]$, il existe au moins un réel $c \in [a,b]$ tel que $f(c)=u$.

En particulier (Théorème de Bolzano), si $f(a)$ et $f(b)$ sont de signes différents, il existe au moins un réel c tel que $f(c)=0$.

Ainsi, pour une fonction donnée définie sur un intervalle donné, le but de l'algorithme de dichotomie va être de découper en 2 l'intervalle [a,b] en deux, afin d'y trouver la solution. Par divisions successives de l'intervalle, on convergera vers la solution.



R **Tester le signe de $f(a)$ et $f(b)$.**

Il existe plusieurs méthodes pour tester si $f(a)$ et $f(b)$ sont de signes différents. Si on ne se préoccupe pas de savoir la relation d'ordre entre $f(a)$ et $f(b)$, un test efficace consiste en un test du signe de $f(a)\cdot f(b)$.

Principe de la méthode de Newton

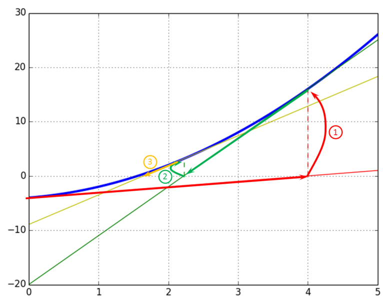
Théorème **Développement de Taylor à l'ordre 1**

Soit f une fonction C^1 sur un intervalle I et $a \in I$. Le développement de Taylor à l'ordre 1 de f est donné par

$$f(x)=f(a)+f'(a)\cdot(x-a)+o(x-a)$$

Géométriquement, lorsqu'on néglige le reste, le développement de Taylor donne l'équation de la tangente en a. Notons $\Delta(x)$ cette équation. L'abscisse c de l'intersection de la tangente avec l'axe des abscisses est donnée par la résolution de

$$\Delta(c)=0 \Longleftrightarrow f(a)+f'(a)\cdot(c-a)=0 \Longleftrightarrow c=a-\frac{f(a)}{f'(a)}$$



Évaluation de la dérivée numérique

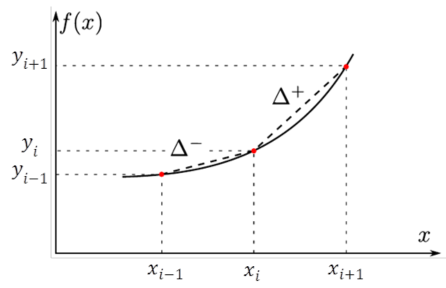
Résultat En première approximation, il est possible d'approximer la dérivée en approximant la tangente à la courbe par une droite passant par deux points successifs. Dans ces conditions, pour une valeur de h suffisamment faible, on a :

$$f'(x_0)\simeq\frac{f(x_0+h)-f(x_0)}{h}.$$

Méthodes à un pas

Résultat Différence avant – Schéma d'Euler explicite
Dans ce cas, l'estimation de la dérivée au point P_i s'appuie sur le point P_{i+1} :

$$f'(x_i) \simeq \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$



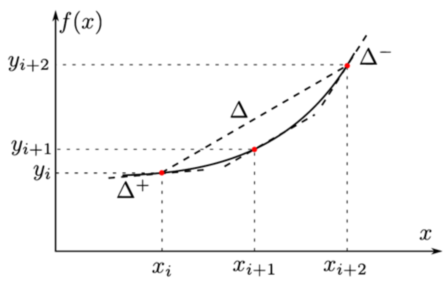
Résultat Différence arrière – Schéma d'Euler implicite
Dans ce cas, l'estimation de la dérivée au point P_i s'appuie sur le point P_{i-1} :

$$f'(x_i) \simeq \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

Méthode à deux pas

Résultat On peut aussi utiliser les points P_{i-1} et P_{i+1} pour estimer la dérivée en P_i :

$$f'(x_i) \simeq \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}}$$



R

- Lorsqu'il s'agit de dériver une fonction temporelle « en temps réel », le point suivant n'est pas encore connu donc seule la différence arrière peut être calculée.
- Le calcul de la dérivée conduit à un tableau de valeurs de dimension $n-1$.

Bibliothèque Python

Il est possible de résoudre l'équation $f(x) = 0$ en utilisant les modules de la bibliothèque `scipy` :

■ **Python**

Résolution de $\sin(x) = 0$ avec 0,5 comme valeur d'initialisation.

```
def f(x):
    return sin(x)

sol = newton(f,0.5)
print(sol)
print(f(sol))
```

Résolution du système :

$$\begin{cases} x + 10y - 3z - 5 = 0 \\ 2x - y + 2z - 2 = 0 \\ -x + y + z + 3 = 0 \end{cases}$$

```
from scipy.optimize import fsolve
# définition du système
def syst(var):
    # définition des variables
    x, y, z = var[0], var[1], var[2]
    eq1 = x +10*y-3*z-5
    eq2 = 2*x-y+2*z-2
    eq3 = -x+y+z+3
    res = [eq1, eq2, eq3]
    return res
    # Initialisation de la recherche
    # des solutions numériques
x0, y0, z0 = 0, 0, 0
sol_ini = [x0, y0, z0]
sol = fsolve(syst, sol_ini)
sol = newton(f,0.5)
print(sol)
```