

RÉSOLUTION NUMÉRIQUE D'ÉQUATIONS

Résoudre une équation à une inconnue x revient à résoudre une équation du type $f(x) = 0$, où f est une fonction plus ou moins compliquée.

On supposera que f est continue sur un segment $[a, b]$ avec $f(a)f(b) < 0$.

Le théorème des valeurs intermédiaires nous assure alors que l'équation $f(x) = 0$ admet une solution dans l'intervalle $[a, b]$.

Par ailleurs, si f est de plus strictement monotone, cette solution est unique.

Dans la suite, on supposera que f est strictement monotone, bien que cela ne soit pas nécessaire à l'exécution des deux méthodes que nous allons voir, qui permettent d'obtenir une valeur numérique approchée de la solution.

I. Méthode par dichotomie

1) Principe

L'idée est la suivante : on calcule le milieu m de l'intervalle $[a, b]$.

On a alors l'une des deux possibilités suivantes :

- Si $f(a)f(m) < 0$, alors f change de signe entre a et m , donc admet une racine sur cet intervalle. On pose $b = m$ et on recommence à partir de ce nouvel intervalle $[a, b]$.
- Sinon, nécessairement on a $f(b)f(m) \leq 0$, (car $f(a)$ et $f(b)$ sont de signes contraires). Donc f s'annule sur $[m, b]$, on pose alors $a = m$ et on recommence à partir du nouvel intervalle $[a, b]$ ainsi obtenu.

On s'arrête lorsque l'intervalle $[a, b]$ obtenu est suffisamment petit, par exemple lorsque sa longueur est inférieure à un ε passé en paramètre.

2) Implémentation

```
def dichotomie(f, a, b, eps):
    while b-a > eps :
        m = (a+b)/2
        if f(m)*f(a) <= 0 : #f change de signe sur [a,m], donc
            b = m           #la racine cherchée est dans [a,m]
        else:               #sinon, elle est dans [m,b]
            a = m
    return (a+b)/2          #on a fini la boucle, on renvoie
                           #le centre du dernier intervalle [a,b]
```

Dans l'algorithme précédent, on calcule plusieurs fois les mêmes images par f . Pour éviter cela, on peut garder en mémoire les valeurs aux bornes de l'intervalle.

```
def dichotomie(f, a, b, eps):
    fa, fb = f(a), f(b)
    while b-a > eps :
        m = (a+b)/2
        fm = f(m)
        if fm*fa <= 0 :    #f change de signe sur [a,m], donc
            b, fb = m, fm #la racine cherchée est dans [a,m]
        else:              #sinon, elle est dans [m,b]
            a, fa = m, fm
    return (a+b)/2          #on a fini la boucle, on renvoie
                           #le centre du dernier intervalle [a,b]
```

3) Terminaison et complexité

On note a_i et b_i le contenu des variables a et b à la fin de la i -ème itération, en convenant que a_0 et b_0 sont les valeurs initiales de a et b . Montrons que $b_i - a_i = \frac{b_0 - a_0}{2^i}$.

- Avant d'entrer dans la boucle : on a bien $b_0 - a_0 = \frac{b_0 - a_0}{2^0}$.

- Supposons qu'après i itérations, $b_i - a_i = \frac{b_0 - a_0}{2^i}$. Si on effectue une itération supplémentaire :
On affecte $m = (a_i + b_i)/2$. Ensuite,
 - soit $b_{i+1} = m$ et $a_{i+1} = a_i$ et dans ce cas :

$$b_{i+1} - a_{i+1} = \frac{a_i + b_i}{2} - a_i = \frac{b_i - a_i}{2} = \frac{b_0 - a_0}{2^i} \times \frac{1}{2} = \frac{b_0 - a_0}{2^{i+1}}$$

- soit $a_{i+1} = m$ et $b_{i+1} = b_i$ et dans ce cas :

$$b_{i+1} - a_{i+1} = b_i - \frac{a_i + b_i}{2} = \frac{b_i - a_i}{2} = \frac{b_0 - a_0}{2^i} \times \frac{1}{2} = \frac{b_0 - a_0}{2^{i+1}}$$

- Or on a :

$$\begin{aligned} b_i - a_i \leq \varepsilon &\Leftrightarrow \frac{b_0 - a_0}{2^i} \leq \varepsilon \\ &\Leftrightarrow 2^i \geq \frac{b_0 - a_0}{\varepsilon} \\ &\Leftrightarrow i \geq \log_2 \left(\frac{b_0 - a_0}{\varepsilon} \right) \end{aligned}$$

Finalement, la boucle **while** se termine, et plus précisément, elle s'exécute $\left\lceil \log_2 \left(\frac{b_0 - a_0}{\varepsilon} \right) \right\rceil$ fois (si ce nombre est positif ; sinon, elle s'exécute 0 fois).

La complexité est donc un $O \left(\ln \left(\frac{b_0 - a_0}{\varepsilon} \right) \right)$.

4) Correction

On va montrer que la valeur renvoyée est bien un nombre dont la distance à la solution r cherchée est inférieure à ε .

On utilise l'invariant de boucle suivant : après chaque itération de la boucle **while**, $a_i \leq r \leq b_i$.

- Avant d'entrer dans la boucle : d'après les hypothèses sur f , $a_0 \leq r \leq b_0$.
- On suppose la propriété vraie au bout de i itérations. Si on effectue une itération supplémentaire, on a $m = (a_i + b_i)/2$.
 - Si $f(m)f(a_i) < 0$: f change de signe sur $[a_i, m]$, donc $r \in [a_i, m]$, or $a_{i+1} = a_i$ et $b_{i+1} = m$, donc $r \in [a_{i+1}, b_{i+1}]$.
 - Sinon, $f(m)f(a_i) \geq 0$, donc $f(m)f(b_i) \leq 0$, donc $r \in [m, b_i]$, or $a_{i+1} = m$ et $b_{i+1} = b_i$, donc $r \in [a_{i+1}, b_{i+1}]$.

Il s'agit donc bien d'un invariant de boucle.

En particulier, à la sortie de la boucle, on a $r \in [a_i, b_i]$, et $b_i - a_i \leq \varepsilon$. Posons $m = (a_i + b_i)/2$ la valeur renvoyée par la fonction, alors

$$\begin{aligned} a_i - m &\leq r - m \leq b_i - m \\ -\frac{b_i - a_i}{2} &\leq r - m \leq \frac{b_i - a_i}{2} \end{aligned}$$

Donc $|r - m| \leq \frac{\varepsilon}{2} < \varepsilon$.

La valeur renvoyée remplit donc bien la condition demandée.

5) Exemple d'exécution

On applique l'algorithme à la fonction est $f : x \mapsto x^2 - 2$ sur l'intervalle $[1; 2]$, avec $\varepsilon = 10^{-6}$.

Les valeurs de a et b données dans le tableau suivant sont prises à la fin de la i -ème itération de la boucle, (ou juste avant l'exécution de la boucle lorsque $i = 0$).

i	a	b	(a+b)/2
0	1.00000000000000	2.00000000000000	1.50000000000000
1	1.00000000000000	1.50000000000000	1.25000000000000
2	1.25000000000000	1.50000000000000	1.37500000000000
3	1.37500000000000	1.50000000000000	1.43750000000000

4		1.375000000000		1.437500000000		1.406250000000	
5		1.406250000000		1.437500000000		1.421875000000	
6		1.406250000000		1.421875000000		1.414062500000	
7		1.414062500000		1.421875000000		1.417968750000	
8		1.414062500000		1.417968750000		1.416015625000	
9		1.414062500000		1.416015625000		1.415039062500	
10		1.414062500000		1.415039062500		1.414550781250	
11		1.414062500000		1.414550781250		1.414306640625	
12		1.414062500000		1.414306640625		1.4141845703125	
13		1.4141845703125		1.414306640625		1.4142456054688	
14		1.4141845703125		1.4142456054688		1.4142150878906	
15		1.4141845703125		1.4142150878906		1.4141998291016	
16		1.4141998291016		1.4142150878906		1.4142074584961	
17		1.4142074584961		1.4142150878906		1.4142112731934	
18		1.4142112731934		1.4142150878906		1.4142131805420	
19		1.4142131805420		1.4142150878906		1.4142141342163	
20		1.4142131805420		1.4142141342163		1.4142136573792	

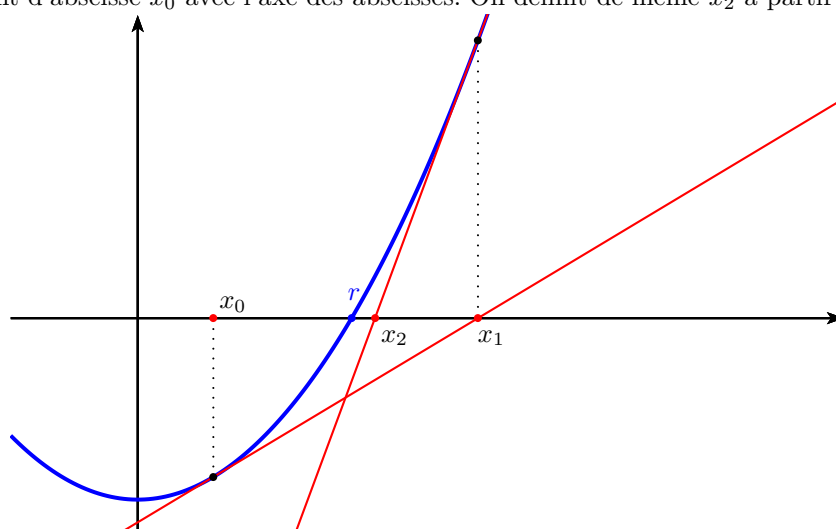
6) Avantages et inconvénients de la méthode

- *Avantages* : Méthode très robuste, qui donne un résultat garanti en un temps garanti.
- *Inconvénients* : Méthode lente.

II. Méthode de Newton

1) Principe

On part d'un réel x_0 assez proche de la racine r cherchée. Le réel x_1 est alors l'abscisse de l'intersection de la tangente à $\mathcal{C}_f$ au point d'abscisse x_0 avec l'axe des abscisses. On définit de même x_2 à partir de x_1 , etc.



On montre aisément que si $f'(x_n) \neq 0$, alors

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

Plusieurs critères sont possibles pour déterminer l'arrêt : on peut envisager de s'arrêter au bout d'un nombre donné d'étapes, ou lorsque le nombre x_n obtenu satisfait $|f(x_n)| < \varepsilon$, etc.

Dans ce qui suit, on choisit d'arrêter le calcul lorsque $|x_{n+1} - x_n|$ est suffisamment petit, en espérant que dans ce cas, la suite $(x_n)_{n \in \mathbb{N}}$ converge et que sa limite ne soit pas trop éloignée de x_{n+1} .

2) Implémentation

```
def newton(f, fp, x0, eps):
    x = x0                #Valeur courante de x_n
    y = x - f(x)/fp(x)    #Valeur courante de x_{n+1}
    while abs(y-x) > eps:
        x, y = y, y - f(y)/fp(y)
    return y
```

3) Exemple d'exécution

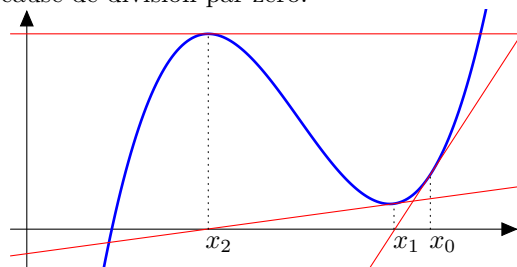
On considère de nouveau la fonction $f : x \mapsto x^2 - 2$, et on part de $x_0 = 1$. On note i le nombre d'itérations de la boucle :

i		x_i	
0		1.0000000000000000	
1		1.5000000000000000	
2		1.4166666666666667	
3		1.4142156862745	
4		1.4142135623747	
5		1.4142135623731	
6		1.4142135623731	

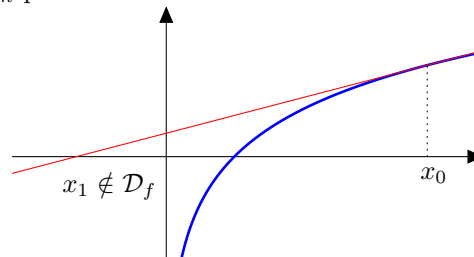
4) Quelques cas de non convergence

En utilisant cette méthode, on peut rencontrer une erreur malgré l'existence d'une solution :

- $f'(x_n)$ peut s'annuler après un certain nombre d'itérations; le programme sort alors en erreur pour cause de division par zéro.

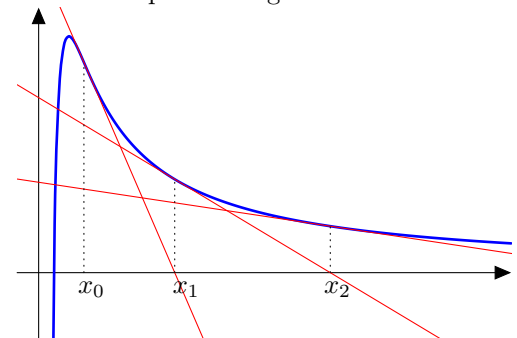


- x_n peut sortir de l'ensemble de définition de f

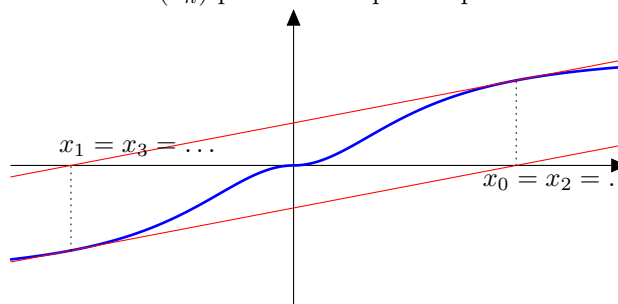


Par ailleurs, la terminaison n'est pas assurée :

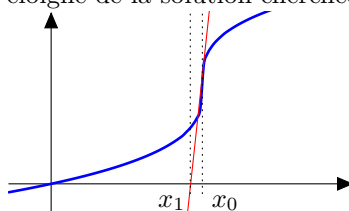
- La méthode peut diverger :



- La suite (x_n) peut devenir périodique :



Enfin, le résultat obtenu peut être très éloigné de la solution cherchée :



5) Avantages et inconvénients

- *Avantages* : Méthode très rapide dans de nombreux cas. Pour $x \mapsto x^2 - 2$, on obtient une meilleure approximation de $\sqrt{2}$ en 5 itérations qu'en une cinquantaine d'itérations de la méthode par dichotomie.

On peut montrer qu'en général, le nombre de décimales correctes double à chaque itération.

- *Inconvénients* :

- Cette méthode utilise la dérivée de f : il faut donc que f' existe et soit facile à calculer.
- On peut rencontrer une erreur malgré l'existence d'une solution.
- La terminaison n'est pas assurée.
- Même si le programme termine, le résultat renvoyé peut être éloigné de la solution.
- Il se peut que la suite converge lentement. Par exemple, pour $f : x \mapsto x^8$ avec $x_0 = 0, 1$:

i		x_i	
-		0.10000000000000	
1		0.08750000000000	
2		0.07656250000000	
3		0.06699218750000	
4		0.0586181640625	
5		0.0512908935547	
6		0.0448795318604	
7		0.0392695903778	
...		...	
193		0.00000000000006	
194		0.00000000000006	
195		0.00000000000005	
196		0.00000000000004	
197		0.00000000000004	
198		0.00000000000003	
199		0.00000000000003	