IN2-Cours

Recherche des zéros d'une fonction

On apprend dans ce cours à **résoudre numériquement une équation algébrique**, du type

$$f(x) = 0$$

avec f une fonction de $\mathbb{R}$ dans $\mathbb{R}$. On dit qu'on cherche les **zéros** de f, ou encore ses **racines**.

Table des matières

1 Exemples et intérêt

Mentionnons quelques situations dans lesquelles on est amené à chercher les zéros d'une fonction. Notamment, on peut penser à la détermination des décimales d'un nombre irrationnel, ou encore à des exemples en physique. En particulier,

- on peut chercher à déterminer les premières décimales de π sachant que π est une racine de la fonction sinus

$$\sin x = 0 \qquad \text{pour} \qquad x = \pi$$

- on peut chercher à déterminer les premières décimales de e sachant que e est une racine de la fonction $\ln x - 1$

$$\ln x - 1 = 0 \qquad \text{pour} \qquad x = e$$

- en physique, on montre qu'un corps noir vérifie la loi de Wien $h\,\nu_{\max} \,/\, (k_B\,T) \approx x$ où $\nu_{\max}$ est la fréquence à laquelle le corps noir rayonne le plus d'énergie, T est la température du corps noir, et h et k_B des constantes fondamentales de la physique ; et où x est la racine de

$$e^{-x} + \frac{1}{5}\,x - 1 = 0 \qquad \text{soit} \qquad x \approx 4{,}96...$$

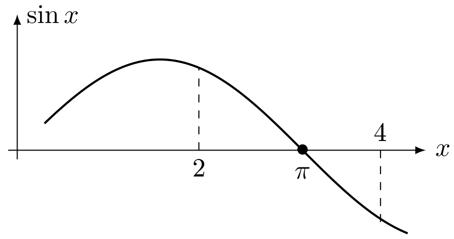
Ces trois équations peuvent être résolues numériquement. Nous discutons dans un premier temps la méthode de **résolution par dichotomie**.

2 Recherche des zéros d'une fonction par dichotomie

2.1 Principe de la résolution par dichotomie

Le but est de diminuer progressivement l'amplitude de l'intervalle dans lequel on sait que se trouve la racine de la fonction. Expliquons pas à pas la méthode sur l'exemple de la recherche de $\pi = 3{,}14159...$ à partir de la résolution de $\sin x = 0$.

On ne cherche pas la solution sur tout l'espace des réels : il nous faut un intervalle de départ. Nous savons que π est entre 2 et 4 par exemple.

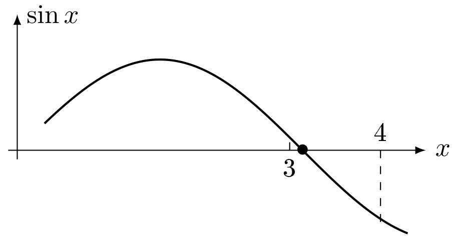


L'intérêt du choix de cet intervalle est d'avoir

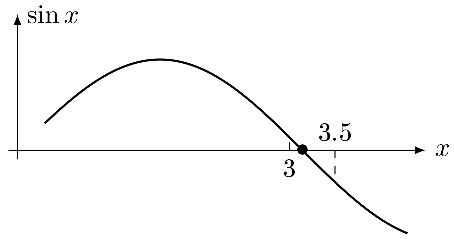
$$\sin 2 > 0 \qquad \text{et} \qquad \sin 4 < 0$$

Puisque la fonction sin est continue, si elle est positive en 2 et négative en 4, c'est qu'elle est passée par 0 entre ces deux valeurs ! (justement en π, la valeur que l'on cherche !).

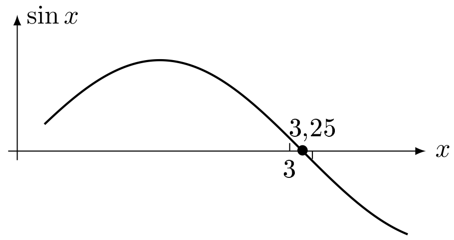
Comme son nom l'indique, la recherche du zéro par dichotomie consiste à découper l'intervalle en deux. Pour cela, on prend le milieu de l'intervalle $[2\,;4]$: c'est 3. Et on se demande si $(\sin 3)$ est positif ou négatif. On a en fait $\sin 3 \approx 0{,}14 > 0$. Donc le sinus est positif en 3 et négatif en 4 : il est passé par zéro entre ces deux valeurs par continuité ! Inutile de cherche la solution entre 2 et 3 donc, on sait qu'elle est entre 3 et 4 ! Il ne reste qu'à refaire ce raisonnement dans l'intervalle $[3\,;4]$: et on comprend que lors d'une étape, on a diminué la taille de l'intervalle de recherche par 2 !



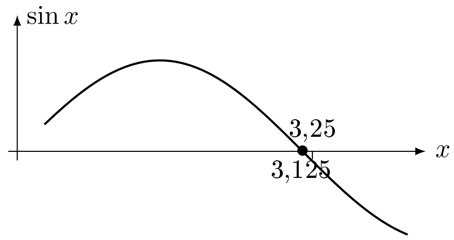
On regarde le milieu du nouvel intervalle : c'est 3,5. On a $\sin(3{,}5) \approx -0{,}35 < 0$. Donc par continuité la racine de $\sin x$ est entre 3 et 3,5.



On regarde le milieu du nouvel intervalle : c'est 3,25. On a $\sin(3{,}25) \approx -0{,}11 < 0$. Donc par continuité la racine de $\sin x$ est entre 3 et 3,25.



On regarde le milieu du nouvel intervalle : c'est 3,125. On a $\sin(3{,}125) \approx 0{,}017 > 0$. Donc par continuité la racine de $\sin x$ est entre 3,125 et 3,25. Vous avez compris la suite...



Quand arrêter l'algorithme ? Par exemple en fixant la précision souhaitée par la taille de l'intervalle à partir duquel on décide de s'arrêter (ou par le nombre d'itérations maximales qu'on souhaite autoriser).

2.2 Proposition d'implémentation

On propose d'implémenter l'algorithme de recherche de zéros par dichotomie comme suit : `f` est la fonction dont on cherche à déterminer la racine, `a` la valeur basse de l'intervalle de départ, `b` l'intervalle de la valeur haute et enfin `precision` la taille de l'intervalle à partir de laquelle on souhaite s'arrêter.

```
def zeroDicho(f, a, b, precision) :

    while b - a > precision :                # tant qu'on a pas une
                                               precision suffisante, on poursuit
        if f( (a + b) / 2 ) * f(b) > 0 :     # si f((a+b)/2) est du meme signe que f(b),
            b = (a + b) / 2                    on garde la premiere moitie de
                                               l'intervalle
        else :                               # sinon on garde la deuxieme moitie
            a = (a + b) / 2

    return (a + b) / 2                       # on renvoie le milieu du dernier intervalle
```

Et on appelle par exemple cet algorithme comme suit (en chargeant la bibliothèque `numpy` pour avoir accès à la fonction sinus...)

```
>>> import numpy as np
>>> zeroDicho(np.sin, 2, 4, 1e-6)
3.141592502593994
>>> np.pi
3.141592653589793
```

Si on veut une meilleure précision, il suffit de diminuer `precision`, par exemple de le passer à `1e-9`. Dans ce cas, on obtient

```
>>> import numpy as np
>>> zeroDicho(np.sin, 2, 4, 1e-9)
3.1415926539339125
>>> np.pi
3.141592653589793
```

2.3 Retour sur les exemples de début de chapitre

Cherchons une valeur approchée de e en définissant la fonction $\ln(x) - 1$.

```
>>> import numpy as np
>>> f = lambda x : np.log(x) - 1
>>> zeroDicho(f, 2, 4, 1e-6)
2.7182822227478027
>>> np.e
2.718281828459045
```

Puis cherchons la valeur de la constante dans la loi de Wien.

```
>>> import numpy as np
>>> f = lambda x : np.exp(-x) + x / 5 - 1
>>> zeroDicho(f, 3, 7, 1e-6)
4.965114116668701
```

La méthode fonctionne effectivement.

2.4 Complexité de l'algorithme de recherche de zéros par dichotomie

Comme nous l'avons décrit dans la présentation de l'algorithme, une étape consiste à couper l'intervalle en deux. le nombre n d'étapes pour arriver à la précision ε en fonction de la taille de départ $b-a$ vérifie donc

$$\frac{b-a}{2^n} < \varepsilon \qquad \text{soit} \qquad n > \ln_2\left(\frac{b-a}{\varepsilon}\right) \propto \ln \varepsilon$$

Complexité de l'algorithme par dichotomie. On dit alors que l'algorithme est **logarithmique**, car le nombre d'étapes $\#n$ nécessaires est proportionnel au logarithme de la précision $\ln \varepsilon$. On note

$$\#n = O\left(\ln \varepsilon\right)$$

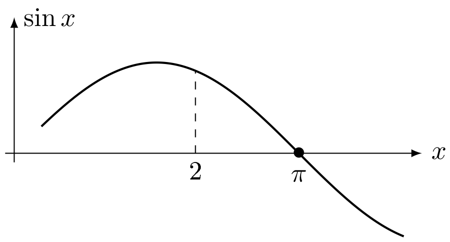
Peut-on faire mieux, c'est-à-dire plus rapide, c'est-à-dire obtenir la même précision mais en moins d'étapes ? Oui ! Avec la méthode de Newton.

3 Recherche des zéros d'une fonction par méthode de Newton

3.1 Principe de la résolution par la méthode de Newton

Le but est cette fois d'approcher la solution en suivant les tangentes à la courbe. Expliquons pas à pas la méthode sur l'exemple de la recherche de $\pi = 3{,}14159...$ à partir de la résolution de $\sin x = 0$.

Partons d'un point quelconque : par exemple 2. Notons le r_0 (pour exprimer que c'est notre première estimation de la racine du sinus).



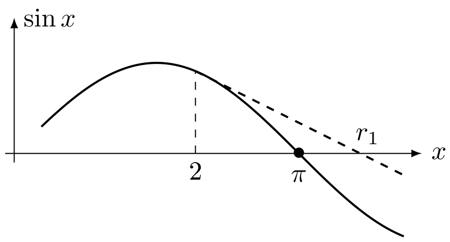
Et traçons la tangente du sinus en $r_0 = 2$.

Rappel. Expression de la tangente en a d'une fonction f. Elle s'écrit

$$y = f(a) + (x - a)\, f'(a)$$

L'équation de la tangente du sinus en $r_0 = 2$ est donc

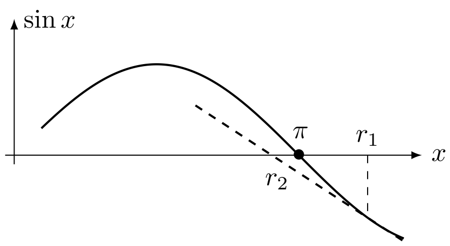
$$y = \sin r_0 + (x - r_0)\, \cos r_0$$



Cette tangente coupe l'axe des abscisses lorsque $y = 0$ donc pour

$$x = r_0 + \frac{\sin r_0}{\cos r_0} = r_1$$

Et l'idée est de recommencer cette procédure à partir de r_1 : on détermine la tangente du sinus en r_1, puis l'abscisse où celle-ci coupe l'axe des abscisses : cela définit r_2, puis on recommence de nouveau pour obtenir r_3, puis r_4,...



Sans chercher à la démontrer, on observe que la suite des racines r_0, r_1, r_2, ... **converge effectivement vers le zéro** de la fonction sinus. *(En fait, il y plusieurs conditions pour assurer la convergence : l'algorithme ne converge pas forcément... et notamment cela dépend du choix du point de départ r_0).*

3.2 Méthode de Newton

Méthode de Newton. Si l'on cherche la racine d'une fonction f à partir d'une première estimation r_0, alors la suite définie par récurrence

$$\boxed{r_{i+1} = r_i - \frac{f(r_i)}{f'(r_i)}}$$

qui correspond aux abscisses des passages par zéros des tangentes successives

$$y = f(r_i) + (x - r_i)\, f'(r_i)$$

converge sous certaines conditions vers cette racine.

3.3 Implémentation en python de la méthode de Newton

Proposition d'implémentation. La difficulté est de savoir quand arrêter l'algorithme... Ici on peut simplement proposer un nombre de récurrences fixé N. Et on peut chercher à évaluer la dérivée numériquement ! (Voir le chapitre IN1). On propose

```
def newton(f, r0, N) :
    res = r0
    h = 1e-6
    for i in range(N) :
        der_f = ( f(res + h) - f(res) ) / h
        res -= f(res) / der_f
    return res
```

On teste cet algorithme avec les commandes suivantes

```
>>> import numpy as np
>>> newton(np.sin, 2, 100)
3.14159265359
```

Toutes les décimales annoncées sont bonnes !

3.4 Comparaison de la méthode de Newton et de celle par dichotomie

De manière générale, la méthode de Newton converge plus rapidement vers la racine que la méthode par dichotomie.

Δ Δ Δ Fin du cours.

Exercice 1. On considère l'équation $f(x) = x^3 - 2\,x^2 + 1 = 0$. Implémenter un programme basé sur la méthode de Newton qui renvoie une racine de f. On imposera un critère d'arrêt du type $|f(x)| < \varepsilon$ pour une petite valeur de ε. Commentez les valeurs obtenues.

Correction exercice 1. Il y a une solution évidente $x = 1$. On démarre la recherche de solution à partir de $x = 10$. Le programme ci-dessous renvoie une autre solution $x \approx 1{,}618034$.

```
f = lambda x : x**3 - 2. * x**2 + 1.
derf = lambda x : 3. * x**2 - 4. * x

eps = 1e-6
x = 10

while abs(f(x)) > eps :
    x = x - f(x) / derf(x)

print(x)
```