Chapitre 9

Recherche de racines

9.1 Recherche d'une racine de fonction

9.1.1 Recherche par dichotomie

Théorème 1 *Soit f une application réelle continue sur l'intervalle $[a,b]$, et telle que :*

$$f(a) \times f(b) \leqslant 0$$

Alors f admet une racine sur l'intervalle $[a,b]$ (c.à.d. $x \in [a,b]$, $f(x) = 0$).

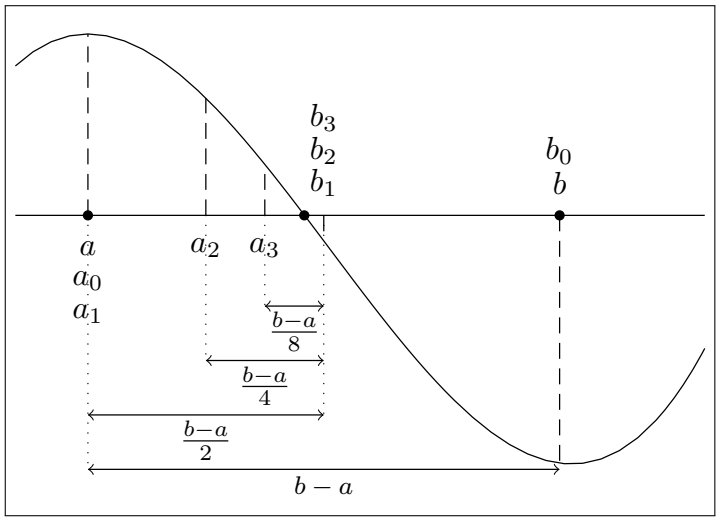
Preuve (Esquisse) : Considérer les deux suites (a_n) et (b_n) :

$$a_0 = a \qquad b_0 = b$$

Soit $m_n = \dfrac{a_n + b_n}{2}$ le milieu de $[a_n, b_n]$:

- Si $f(a_n) \times f(m_n) \leqslant 0$, alors :
  $a_{n+1} = a_n$ et $b_{n+1} = m_n$ (Dichotomie à gauche)
- Sinon :
  $a_{n+1} = m_n$ et $b_{n+1} = b_n$ (Dichotomie à droite)

Puis on montre que les deux suites (a_n) et (b_n) sont adjacentes et convergent vers une racine de f. □



9.1.2 Algorithme de recherche d'une racine par dichotomie

- Algorithme de recherche d'une racine par dichotomie (à 10^{-3} près) :

Des variables `a` et `b` contiennent les valeurs successives prises par les suites (a_n) et (b_n).

```
TANT QUE b-a > 10^-3:
    m = (a+b) / 2
    SI f(a) * f(m) <= 0 ALORS :
        a, b = a, m    # Dichotomie à gauche
    SINON :
        a, b = m, b    # Dichotomie à droite
FIN TANT QUE
RETOURNE (a+b)/2
```

La propriété `f(a)*f(b) <= 0` est un invariant de boucle. De plus $b_n - a_n = \dfrac{b_0 - a_0}{2^n}$. Donc la condition `b-a >` 10^{-3} finit par ne plus être vérifiée.

L'algorithme s'arrête en retournant :

`(a+b)/2` : valeur approchée à 10^{-3} près d'une racine, ou
`a` valeur approchée à 10^{-3} près par défaut d'une racine, ou
`b` valeur approchée à 10^{-3} près par excès d'une racine.

9.1.3 Code `python`

• Code `python` :

```
def dichotomie(f,a,b,e):
    assert f(a) * f(b) < 0
    while b-a > e:
        m = (a+b)/2
        if f(a)*f(m) <= 0:
            a, b = a, m    # Dichotomie à gauche
        else:
            a, b = m, b    # Dichotomie à droite
    return (a+b)/2
```

• Exemple :

```
>>> f = lambda x : x**2-2
>>> dichotomie(f,0,3,0.001)
1.4139404296875    # Valeur approchée à 0.001 près de √2
```

L'instruction `lambda` permet de définir facilement (en une ligne) une fonction mathématique. L'instruction `f = lambda x : x**2-2` est équivalente à :

```
def f(x):
    return x**2-2
```

9.1.4 Quand utiliser une dichotomie pour la recherche d'une racine ?

1. Sous une hypothèse de régularité faible de la fonction f : continuité,
2. Recherche dur l'intervalle $[a,b]$ lorsque $f(a)$ et $f(b)$ sont de signes opposés.

• Temps de calcul : du même ordre que le nombre de passage dans la boucle `while` : $\approx \log_2\left(\frac{b-a}{e}\right)$.

En effet après n passages dans la boucle : $b_n - a_n = \frac{b_0-a_0}{2^n}$. L'algorithme s'arrête quand :

$$b_n - a_n \leqslant e \iff \frac{b_0-a_0}{2^n} \leqslant e \iff \frac{b_0-a_0}{e} \leqslant 2^n$$

$$\iff \frac{b_0-a_0}{e} \leqslant \exp(n\ln(2)) \iff \ln\left(\frac{b_0-a_0}{e}\right) \leqslant n\ln(2)$$

$$\iff \ln\left(\frac{b_0-a_0}{e}\right)/\ln(2) \leqslant n \iff \log_2\left(\frac{b_0-a_0}{e}\right) \leqslant n$$

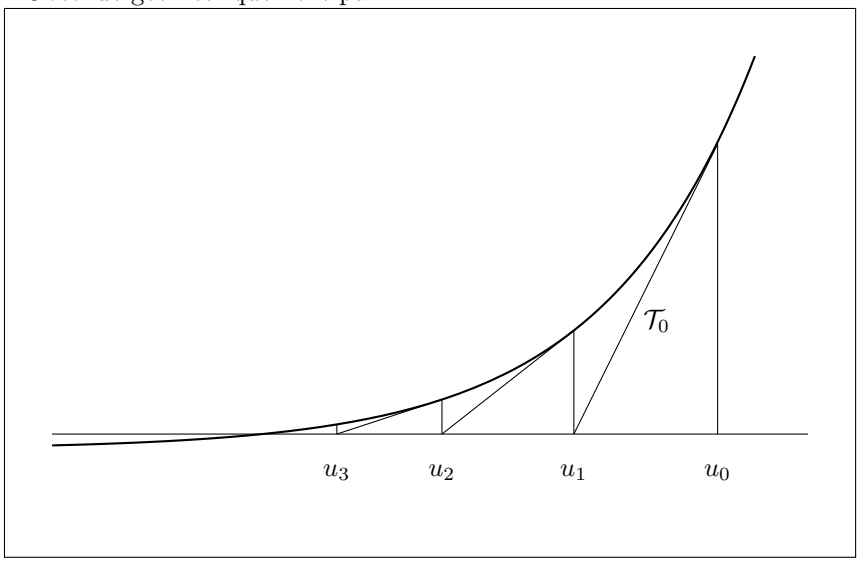
9.1.5 Méthode de Newton

• La méthode de Newton est une méthode célèbre pour la résolution approchée d'une équation $f(x) = 0$. Elle est efficace est très rapide pour peu que l'on soit assuré de la convergence.

• On considère la suite définie par u_0 et par la relation de récurrence :

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}$$

Obtenue géométriquement par :



La tangente à $y = f(x)$ en u_n a pour équation : $y = f'(u_n)(x - u_n) + f(u_n)$.
Si $f'(u_n) \neq 0$ elle intersecte (Ox) en $u_n - \frac{f(u_n)}{f'(u_n)} = u_{n+1}$.

Sous certaines hypothèses (u_n) converge vers une racine de la fonction f :

Théorème 2 (Méthode de Newton) *Soit $f : [a,b] \longrightarrow \mathbb{R}$ une application de classe C^1 et x_0 une solution de l'équation*

$$(E) : \ f(x) = 0$$

en laquelle $f'(x_0) \neq 0$.
• *Alors il existe $r > 0$ tel que dans l'intervalle $I =]x_0 - r; x_0 + r[$ l'équation (E) ait pour solution unique x_0, et de plus la suite $(u_n)_{n\in\mathbb{N}}$ définie par :*

$$u_0 \in I \quad \forall n \in \mathbb{N}, \quad u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}$$

est convergente et a pour limite cette racine x_0.

• *Si de plus f est de classe C^2, alors la convergence est quadratique, c.à.d., $\exists k > 0$, lorsque n est suffisamment grand, en notant $r_n = |x_0 - u_n|$ le reste au rang n :*

$$r_{n+1} \leqslant k \cdot r_n^2$$

9.1.6 Code `python` pour la méthode de Newton

On prend comme critère d'arrêt un nombre n d'itérations (calcul de u_n) :

```
def newton(f, g, u0, n):
    u = u0
    for k in range(n):
        u = u - f(u)/g(u)
    return u
```

ou bien $u_{n+1} - u_n < 10^{-N}$.

Il faut savoir calculer la fonction dérivée, et avoir démontré que pour une valeur u_0 la suite (u_n) est bien définie (pas de division par zéro) et converge.

```
def newton(f, g, u0, e):
    u = u0
    v = u0 - f(u0)/g(u0)
    while abs(v-u) > e:
        u = v
        v = v - f(v)/g(v)
    return v
```

Il est plus prudent de fixer un nombre d'itérations maximal, pour éviter une boucle infinie.

```
def newton(f, g, u0, e, iterMax=1000):
    u = u0
    v = u0 - f(u0)/g(u0)
    iter = 0
    while abs(v-u) > e and iter < iterMax:
        u = v
        v = v - f(v)/g(v)
        iter += 1
    return v
```

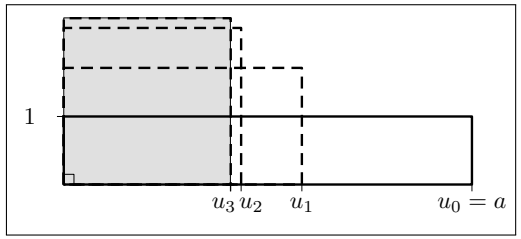
9.1.7 Exemple : extraction de racine par la méthode de Babylone

• Application (antérieure) célèbre : Méthode de Héron d'Alexandrie (ou encore Méthode Babylonienne) pour l'extraction d'une racine carrée.
Si $a > 0$:

$$f : x \mapsto x^2 - a$$

a deux racines, $\pm\sqrt{a}$.

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)} = u_n - \frac{u_n^2 - a}{2u_n} = \boxed{\frac{u_n}{2} + \frac{a}{2u_n} = u_{n+1}}$$



On prouve (facile, suite homographique) que la suite (u_n) :
- converge vers $\sqrt{a}$ lorsque $u_0 > 0$
- converge vers $-\sqrt{a}$ lorsque $u_0 < 0$

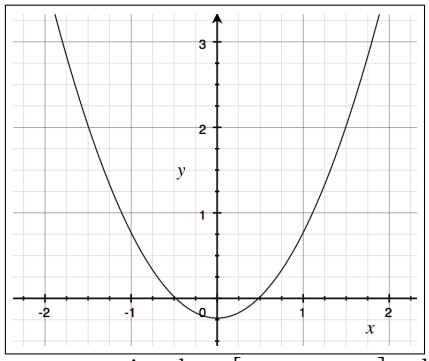
• Ici on prend pas de critère d'arrêt, on passe plutôt en paramètre le nombre d'itérations (juste pour réaliser la rapidité de convergence de l'algorithme) :

```
def babylone(a, n):    # n : nombre d'itérations
    u = a
    for i in range(n):
        u = u /2 + a/(2*u)
    return u
```

Exemple : Extraction de $\sqrt{2}$. Après 5 itérations toutes les 15 premières décimales sont correctes (avec $u_0 = 2$).

```
>>> babylone(2, 5)    # sqrt(2) ; 5 itérations
1.414213562373095
>>> 2**0.5    # Comparaison avec la fonction prédéfinie
1.4142135623730951
```

En tenant compte que $x \mapsto x^2 - a$ est croissante sur $\mathbb{R}^+$:



On s'arrête dès qu'il y a une racine dans $[u_n - e; u_n + e]$: alors u_n est à distance au plus e d'une racine :

Critère d'arrêt :

$$f(\max(0, u_n - e)) \times f(u_n + e) \leqslant 0$$

```
def babylone(a, e):    # e : erreur autorisée
    f = lambda x: x**2-a
    u = a
    while f(max(0,u-e)) * f(u+e) > 0 :
        u = u /2 + a/(2*u)
    return u
```

9.1.8 Recherche d'une racine de fonction réelle

On dispose de deux méthodes efficaces, à savoir programmer :

AVANTAGES (+) / INCONVENIENTS (–) :

1. **Par dichotomie**
   - \+ hypothèse de régularité de la fonction faible : continuité.
   - \+ on peut retourner une valeur par défaut, par excès, à une distance bornée de la racine.
   - \+ Robustesse : sous les hypothèses d'application, ça marche toujours.
   - – $f(a)$ et $f(b)$ doivent être de signes opposés.
   - – La racine recherchée ne doit pas être un extremum.
   - – Ne s'applique que pour des fonctions de $\mathbb{R}$ dans $\mathbb{R}$.
2. **Méthode de Newton**
   - \+ Lorsqu'elle converge, extraordinairement rapide si la fonction est de classe C^2.
   - \+ Se généralise très bien, sur $\mathbb{C}$ ou en dimension supérieure.
   - – hypothèse de régularité fortes C^1 ou C^2.
   - – Convergence non certaine : seulement lorsque le point initial est suffisamment proche de la racine. A choisir.
   - – La racine recherchée ne doit pas être un point stationnaire.