

## TD2 – Planche 4

### Corrigé

#### Rappel : algorithme de dichotomie

On considère une fonction  $g$  continue sur un segment  $[a, b]$ . On suppose que  $g$  s'annule exactement une fois sur  $[a, b]$ , en un point  $\alpha$ . On construit deux suites  $(a_k)$  et  $(b_k)$  en posant  $a_0 = a$ ,  $b_0 = b$  et

$$c_k = \frac{a_k + b_k}{2},$$

puis

$$\begin{cases} a_{k+1} = a_k, & b_{k+1} = c_k, & \text{si } g(a_k)g(c_k) \leq 0, \\ a_{k+1} = c_k, & b_{k+1} = b_k, & \text{sinon.} \end{cases}$$

On sait alors que  $(a_k)$  et  $(b_k)$  convergent toutes deux vers  $\alpha$ .

Dans tout l'exercice, pour  $n \geq 2$ , on pose

$$f_n(x) = x^n + x - n.$$

#### 1. Existence et unicité de $\varepsilon_n$

Pour tout  $n \geq 2$ , on a

$$2^n \geq n.$$

Par ailleurs,

$$f_n(1) = 2 - n \leq 0 \quad \text{et} \quad f_n(2) = 2^n + 2 - n \geq 2 > 0.$$

La fonction  $f_n$  est continue sur  $[1, 2]$ . Le théorème des valeurs intermédiaires donne donc l'existence d'un réel

$$\varepsilon_n \in [1, 2]$$

tel que

$$f_n(\varepsilon_n) = 0.$$

De plus,

$$f'_n(x) = nx^{n-1} + 1 > 0 \quad (x > 0).$$

Ainsi  $f_n$  est strictement croissante sur  $\mathbb{R}_+^*$ , donc l'équation  $f_n(x) = 0$  possède au plus une solution positive.

Il existe donc un unique  $\varepsilon_n \in \mathbb{R}_+^*$  tel que

$$\boxed{f_n(\varepsilon_n) = 0, \quad 1 \leq \varepsilon_n \leq 2.}$$

## 2. Étude de la suite $(\varepsilon_n)$

### a) Encadrement

Comme  $f_n(\varepsilon_n) = 0$ ,

$$\varepsilon_n^n + \varepsilon_n - n = 0,$$

d'où

$$\varepsilon_n^n = n - \varepsilon_n \leq n.$$

Or  $\varepsilon_n \geq 1$ , donc

$$1 \leq \varepsilon_n \leq n^{1/n}.$$

Ainsi

$1 \leq \varepsilon_n \leq n^{1/n}.$

### b) Convergence

On sait que

$$n^{1/n} = \exp\left(\frac{\ln n}{n}\right) \rightarrow 1,$$

car  $\frac{\ln n}{n} \rightarrow 0$ .

L'encadrement précédent donne alors, par le théorème des gendarmes,

$\varepsilon_n \rightarrow 1.$

La limite  $\ell$  demandée est donc

$\ell = 1.$

## 3. Calcul d'une valeur approchée par dichotomie

On sait que  $\varepsilon_n \in [1, 2]$  et que  $f_n$  est strictement croissante. Il suffit donc d'appliquer la dichotomie sur  $[1, 2]$ .

On arrête l'algorithme lorsque la longueur de l'intervalle est inférieure ou égale à  $2^{-n}$ . En renvoyant l'extrémité droite  $b$ , on obtient alors

$$0 \leq b - \varepsilon_n \leq b - a \leq \frac{1}{2^n}.$$

Une implémentation possible est :

```
def f(n, x):
    return x**n + x - n

def dichotomie(n):
    a, b = 1, 2
    while b - a > 1/(2**n):
        c = (a + b)/2
        if f(n, a) * f(n, c) <= 0:
            b = c
        else:
            a = c
    return b
```

Par exemple, pour  $n = 50$ , cette méthode fournit une approximation de  $\varepsilon_{50}$  à une erreur inférieure ou égale à  $2^{-50}$ .

## 4. Méthode de Newton

On pose

$$g_n(x) = x - \frac{f_n(x)}{f'_n(x)} = x - \frac{x^n + x - n}{nx^{n-1} + 1}.$$

On considère une suite  $(x_k)$  définie par

$$0 \leq x_0 - \varepsilon_n \leq \frac{1}{2^n}, \quad x_{k+1} = g_n(x_k).$$

### a) Étude de $g_n$

Pour  $x > 0$ ,

$$f'_n(x) = nx^{n-1} + 1 > 0,$$

donc  $g_n$  est bien définie et dérivable sur  $\mathbb{R}_+^*$ .

En dérivant,

$$g'_n(x) = 1 - \frac{(f'_n(x))^2 - f_n(x)f''_n(x)}{(f'_n(x))^2} = \frac{f_n(x)f''_n(x)}{(f'_n(x))^2}.$$

Or

$$f''_n(x) = n(n-1)x^{n-2}.$$

Ainsi

$$g'_n(x) = \frac{n(n-1)x^{n-2}f_n(x)}{(nx^{n-1} + 1)^2}.$$

Comme  $f_n$  est strictement croissante et  $f_n(\varepsilon_n) = 0$ ,

$$f_n(x) < 0 \quad \text{sur } [1, \varepsilon_n[, \quad f_n(x) > 0 \quad \text{sur } ]\varepsilon_n, +\infty[.$$

Il en résulte que

$$g'_n(x) < 0 \quad \text{sur } [1, \varepsilon_n[, \quad g'_n(x) > 0 \quad \text{sur } ]\varepsilon_n, +\infty[.$$

Ainsi  $g_n$  est décroissante sur  $[1, \varepsilon_n]$  et croissante sur  $[\varepsilon_n, +\infty[$ .

### b) Position des termes de la suite

On a

$$g_n(\varepsilon_n) = \varepsilon_n$$

puisque  $f_n(\varepsilon_n) = 0$ .

De plus, pour  $x \geq \varepsilon_n$ , on a  $f_n(x) \geq 0$  et  $f'_n(x) > 0$ , donc

$$g_n(x) = x - \frac{f_n(x)}{f'_n(x)} \leq x.$$

Comme  $x_0 \geq \varepsilon_n$ , supposons  $x_k \geq \varepsilon_n$ . Alors, puisque  $g_n$  est croissante sur  $[\varepsilon_n, +\infty[$ ,

$$x_{k+1} = g_n(x_k) \geq g_n(\varepsilon_n) = \varepsilon_n.$$

Ainsi, par récurrence,

$$\forall k \in \mathbb{N}, \quad x_k \geq \varepsilon_n.$$

**c) Convergence de  $(x_k)$**

Pour tout  $k$ ,

$$x_{k+1} = x_k - \frac{f_n(x_k)}{f'_n(x_k)} \leq x_k,$$

car  $x_k \geq \varepsilon_n$  implique  $f_n(x_k) \geq 0$ .

La suite  $(x_k)$  est donc décroissante et minorée par  $\varepsilon_n$ . Elle converge vers une limite  $\ell \geq \varepsilon_n$ .

Par continuité de  $g_n$ ,

$$\ell = \lim_{k \rightarrow \infty} x_{k+1} = \lim_{k \rightarrow \infty} g_n(x_k) = g_n(\ell).$$

Ainsi

$$\ell - \frac{f_n(\ell)}{f'_n(\ell)} = \ell,$$

donc  $f_n(\ell) = 0$ . Par unicité de la solution positive,

$$\boxed{\ell = \varepsilon_n.}$$

Ainsi

$$\boxed{x_k \longrightarrow \varepsilon_n.}$$

**d) Majoration de l'erreur**

On a

$$x_{k+1} - \varepsilon_n = g_n(x_k) - g_n(\varepsilon_n).$$

Comme  $g_n$  est continue et dérivable sur  $[\varepsilon_n, x_k]$ , le théorème des accroissements finis donne l'existence de  $c \in [\varepsilon_n, x_k]$  tel que

$$x_{k+1} - \varepsilon_n = g'_n(c)(x_k - \varepsilon_n).$$

Or

$$g'_n(c) = \frac{n(n-1)c^{n-2}f_n(c)}{(nc^{n-1} + 1)^2}.$$

On peut écrire

$$g'_n(c) = (n-1) \frac{f_n(c)}{nc^{n-1} + 1} \frac{nc^{n-2}}{nc^{n-1} + 1}.$$

Comme  $c \geq \varepsilon_n \geq 1$ ,

$$\frac{nc^{n-2}}{nc^{n-1} + 1} < 1.$$

De plus,

$$\frac{f_n(c)}{nc^{n-1} + 1} = c - g_n(c).$$

Puisque  $g_n$  est croissante sur  $[\varepsilon_n, +\infty[$ ,

$$\varepsilon_n = g_n(\varepsilon_n) \leq g_n(c) \leq g_n(x_k) = x_{k+1},$$

et comme  $c \leq x_k$ ,

$$0 \leq c - g_n(c) \leq x_k - \varepsilon_n.$$

On obtient donc

$$0 \leq g'_n(c) \leq (n-1)(x_k - \varepsilon_n).$$

Finalement,

$$\boxed{0 \leq x_{k+1} - \varepsilon_n \leq (n-1)(x_k - \varepsilon_n)^2.}$$

### e) Estimation quadratique

La question précédente donne directement

$$0 \leq x_{k+1} - \varepsilon_n \leq (n-1)(x_k - \varepsilon_n)^2.$$

Cette inégalité traduit la convergence quadratique de la méthode de Newton dans ce cadre.

### f) Estimation explicite de l'erreur

Montrons par récurrence que

$$0 \leq x_k - \varepsilon_n \leq \frac{1}{n 2^{2^k}} \quad (k \in \mathbb{N}).$$

Pour  $k = 0$ , on sait que

$$0 \leq x_0 - \varepsilon_n \leq \frac{1}{2^n}.$$

Or, pour  $n \geq 2$ ,  $2^n \geq 2n$ , donc

$$\frac{1}{2^n} \leq \frac{1}{2n}.$$

L'inégalité est donc vraie au rang 0.

Supposons-la vraie à un rang  $k$ . D'après la question précédente,

$$0 \leq x_{k+1} - \varepsilon_n \leq (n-1)(x_k - \varepsilon_n)^2.$$

Par hypothèse de récurrence,

$$x_{k+1} - \varepsilon_n \leq (n-1) \frac{1}{n^2 2^{2^{k+1}}} \leq \frac{1}{n 2^{2^{k+1}}},$$

puisque  $n-1 \leq n$ .

La propriété est donc vraie pour tout  $k \in \mathbb{N}$ .

## 5. Approximation à une précision donnée

On cherche un entier  $k$  tel que

$$0 \leq x_k - \varepsilon_n \leq e.$$

D'après la question précédente, il suffit d'avoir

$$\frac{1}{n 2^{2^k}} \leq e.$$

En supposant  $ne < 1$ , cette condition équivaut à

$$2^k \geq \frac{-\ln(ne)}{\ln 2},$$

puis à

$$k \geq \frac{\ln\left(\frac{-\ln(ne)}{\ln 2}\right)}{\ln 2}.$$

On peut donc choisir

$$N = \left\lceil \frac{\ln\left(\frac{-\ln(ne)}{\ln 2}\right)}{\ln 2} \right\rceil.$$

Pour obtenir une approximation de  $\varepsilon_n$  à  $e$  près, on effectue d'abord une dichotomie pour construire  $x_0$ , puis  $N$  étapes de Newton.

Une implémentation possible est :

```

def f(n, x):
    return x**n + x - n

def fp(n, x):
    return n*x**(n-1) + 1

def dichotomie(n):
    a, b = 1, 2
    while b - a > 1/(2**n):
        c = (a + b)/2
        if f(n, a) * f(n, c) <= 0:
            b = c
        else:
            a = c
    return b

def Approx(n, e):
    x = dichotomie(n)

    # Nombre iterations suffisantes pour avoir erreur <= e.
    if n*e >= 1:
        N = 0
    else:
        N = int(
            __import__("math").ceil(
                __import__("math").log(
                    -__import__("math").log(n*e) /
                    __import__("math").log(2)
                ) / __import__("math").log(2)
        )

    for _ in range(N):
        x = x - f(n, x)/fp(n, x)

    return x

```

Pour une écriture plus lisible, on peut naturellement importer les fonctions logarithme et plafond au début du programme :

```

from math import log, ceil

def Approx(n, e):
    x = dichotomie(n)

    if n*e < 1:
        N = ceil(log(-log(n*e)/log(2))/log(2))
    else:
        N = 0

    for _ in range(N):
        x = x - f(n, x)/fp(n, x)

    return x

```