TP Python – 1

Suites récurrentes (5/2)

1 Algorithme de Babylone

Soit $A > 0$. On considère la suite définie par :

$$x_0 \in \mathbb{R}^* \quad \text{et} \quad \forall n \in \mathbb{N},\ x_{n+1} = \frac{1}{2}\left(x_n + \frac{A}{x_n}\right).$$

1. (a)

```
1 def Babylone(A,x0=1,n=10):
2     L = [x0]
3     for k in range(1,n+1):
4         L.append(1/2*(L[-1]+A/L[-1]))
5     return L
```

   (b) On conjecture que, pour $x_0 = 1$, la suite converge vers $\sqrt{A}$. La convergence semble très rapide.

2. (a) Soit $f : x \mapsto \frac{1}{2}\left(x + \frac{A}{x}\right)$ définie et dérivable sur $\mathbb{R}^*$. Pour tout $x \in \mathbb{R}^*$ on a :

$$f'(x) = \frac{1}{2}\left(1 - \frac{A}{x^2}\right) = \frac{x^2 - A}{2x^2}.$$

On en déduit :

x	$-\infty$		$-\sqrt{A}$		0		$\sqrt{A}$		$+\infty$
$f'(x)$		$+$	0	$-$	‖	$-$	0	$+$	
f	$-\infty$	↗	$-\sqrt{A}$	↘	‖	$+\infty$	↘	↗	$+\infty$
					‖		$\sqrt{A}$		

   (b) On suppose que $x_0 \geq \sqrt{A}$. Montrons par récurrence que : $\forall n \in \mathbb{N}, \quad x_n \geq \sqrt{A}$

   — **Initialisation :** immédiat.

   — **Hérédité :** soit $n \in \mathbb{N}$ et supposons $x_n \geq \sqrt{A}$. D'après le tableau de variation on a :

$$x_{n+1} = f(x_n) \geq f(\sqrt{A}) = \sqrt{A}.$$

   L'hérédité est démontrée.

   — **Conclusion :** ...

Pour tout $n \in \mathbb{N}$:

$$x_{n+1} - x_n = \frac{1}{2}\left(x_n + \frac{A}{x_n}\right) - x_n = \frac{A - x_n^2}{2x_n} \leq 0.$$

Ainsi (x_n) est décroissante et minorée donc elle converge.

(c) On suppose $x_0 \in]0, \sqrt{A}[$. Alors, d'après l'étude de f, $x_1 = f(x_0) \geq \sqrt{A}$ donc la question précédente montre que (x_n) est décroissante et minorée à partir du rang 1 donc converge.

(d) Soit ℓ la limite de la suite. D'après ce qui précède et par passage à la limite dans la relation de récurrence :

$$\ell = \frac{1}{2}\left(\ell + \frac{A}{\ell}\right) \quad \text{i.e.} \quad \ell^2 = A.$$

Ainsi $\ell = \pm\sqrt{A}$ et comme la suite est positive, alors $\ell = \sqrt{A}$.

(e) En raisonnant de manière similaire, (x_n) est croissante, majorée (à partir du rang 1) et converge vers $-\sqrt{A}$.

3. (a)

```
def dicho(A,a,b,n):
    def f(x):
        return x**2-A
    for k in range(n):
        c = (a+b)/2
        if f(a)*f(c)<0:
            b = c
        elif f(a)*f(c)>0:
            a =c
        else:
            return c
    return c
```

(b) À chaque étape, on localise $\sqrt{A}$ dans un intervalle de longueur deux fois plus petite que le précédent : $|\texttt{dicho}(A,a,b,n) - \sqrt{A}| \leq \dfrac{|\texttt{dicho}(A,a,b,n-1) - \sqrt{A}}{2}$. Une récurrence immédiate permet de conclure.

4. Dans cette question, on prend $A = 2$, $x_0 = 1$ et $a = 1$ et $b = 2$.

(a)

```
def erreur_dicho(n,A=2,a=1,b=2):
    L=[]
    sol = np.sqrt(A)
    for k in range(1,n+1):
        L.append(np.abs(dicho(A,a,b,k)-sol))
    return L

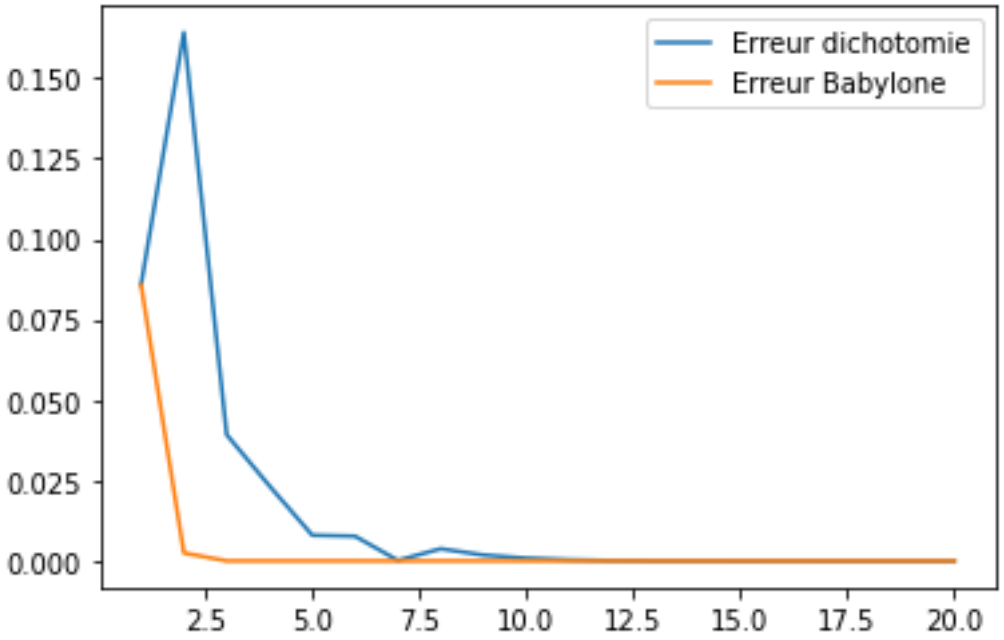

def erreur_babylone(n,A=2,x0=1):
    L=[]
```

```
    sol = np.sqrt(A)
    B = Babylone(A,x0,n)
    for k in range(1,n+1):
        L.append(np.abs(B[k]-sol))
    return L
```

(b)

```
plt.plot(range(1,21),erreur_dicho(20))
plt.plot(range(1,21),erreur_babylone(20))
```

On obtient le graphique suivant :



La méthode de Babylone converge beaucoup plus vite que la méthode de dichotomie.

5. Pour tout $n \in \mathbb{N}^*$ on pose : $w_n = \dfrac{x_n - \sqrt{A}}{x_n + \sqrt{A}}$.

   (a) Soit $n \geq 1$.

$$w_{n+1} = \frac{x_{n+1} - \sqrt{A}}{x_{n+1} + \sqrt{A}} = \frac{\frac{1}{2}\left(x_n + \frac{A}{x_n}\right) - \sqrt{A}}{\frac{1}{2}\left(x_n + \frac{A}{x_n}\right) + \sqrt{A}} = \frac{x_n^2 + A - 2\sqrt{A}x_n}{x_n^2 + A + 2\sqrt{A}x_n}$$
$$= \frac{(x_n - \sqrt{A})^2}{(x_n + \sqrt{A})^2}$$
$$= w_n^2$$

Par une récurrence immédiate, on en déduit :

$$\forall n \in \mathbb{N}, \quad w_n = \left(\frac{x_1 - \sqrt{A}}{x_1 + \sqrt{A}}\right)^{2^{n-1}}.$$

(b) Comme $x_0 > 0$, on a déjà vu que $x_1 - \sqrt{A} \geq 0$ et comme $0 \geq x_1 - \sqrt{A} < x_1 + \sqrt{A}$ on a :

$$0 \leq \frac{u_1 - \sqrt{A}}{u_1 + \sqrt{A}} < 1.$$

On a vu que $(x_n)_{n\geq 1}$ est décroissante donc :

$$\forall n \geq 1, \quad 0 \leq x_n - \sqrt{A} = w_n(x_n + \sqrt{A}) \leq (x_1 + \sqrt{A}) \left(\frac{x_1 - \sqrt{A}}{x_1 + \sqrt{A}}\right)^{2^{n-1}}.$$

Comme $\left(\frac{x_1 - \sqrt{A}}{x_1 + \sqrt{A}}\right)^{2^{n-1}}$ est négligeable devant $\frac{|b-a|}{2^n}$ cela montre bien que la méthode de Babylone converge plus vite que la méthode de dichotomie.

Remarque : si on note $C = x_1 + \sqrt{A}$, on a alors :

$$0 \leq x_{n+1} - \sqrt{A} \leq \frac{x_1 + \sqrt{A}}{4A}(x_n - \sqrt{A})^2$$

ce qui donne pour $A = 2$ et $x_1 = 1$:

$$0 \leq x_{n+1} - \sqrt{A} \leq (x_n - \sqrt{A})^2.$$

Si à l'étape n, on a $0 \leq x_{n+1} - \sqrt{A} \leq 10^{-k}$ (*i.e.* k décimales exactes) alors à l'étape suivante on a

$$0 \leq x_{n+1} - \sqrt{A} \leq 10^{-2k}$$

soit $2k$ décimales exactes : on double le nombre de décimales exactes à chaque tour !

2 Méthode de Newton

La méthode de Newton est une méthode de résolution approchée de solution d'une d'équation du type $f(x) = 0$. Le principe est le suivant :

- on part d'une valeur arbitraire x_0 (pas trop loin de la solution recherchée) ;
- comme $f(x) \simeq f(x_0) + f'(x_0)(x - x_0)$ on approxime la solution de $f(x) = 0$ par la solution de $f(x_0) + f'(x_0)(x - x_0) = 0$;
- on itère le procédé.

On supposera donc qu'il existe une solution de cette équation, notée x_∞, que f est de classe C^1 sur un voisinage de x_∞ et que f' ne s'annule pas sur un voisinage de x_∞.

1. À l'étape $n+1$, on cherche la valeur d'annulation de $x \mapsto f(x_n) + f'(x_n)(x - x_n)$. En résolvant $f(x_n) + f'(x_n)(x - x_n)$ et compte tenu que $f'(x_n) \neq 0$ (hypothèse) alors

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

2. Pour $f : x \mapsto x^2 - A$ où $A > 0$ on obtient

$$\forall n \geq 1, \quad x_{n+1} = x_n - \frac{x_n^2 - A}{2x_n} = \frac{x_n^2 + A}{2x_n} = \frac{1}{2}\left(x_n + \frac{A}{x_n}\right).$$

On retrouve la relation de récurrence de l'algorithme de Babylone.

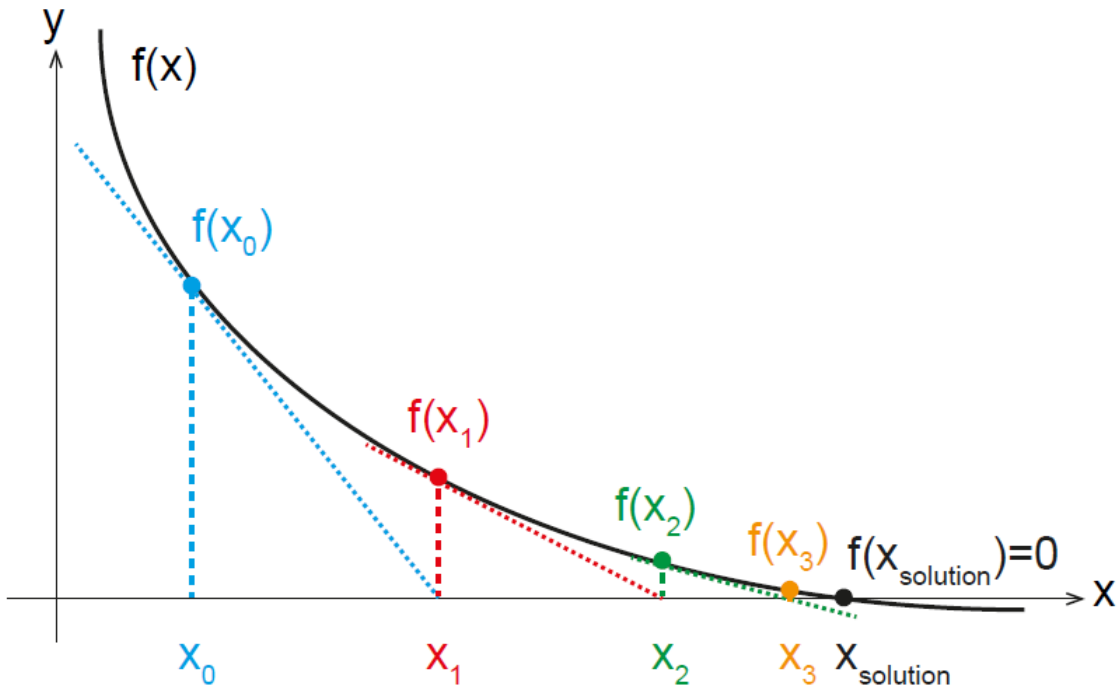


FIGURE 1 – Illustration de la méthode de Newton

3.

```
def Newton(f,fprime, x0,n):
    L= [x0]
    for k in range(1,n+1):
        L.append(L[-1]-f(L[-1])/fprime(L[-1]))
    return L
```

4. (**D'après oral d'Agro**) Soit $f : x \longmapsto \dfrac{1}{1+x} + \dfrac{2}{2+x} + \dfrac{3}{3+x}$.

   (a) La fonction f est définie et dérivable sur $\mathbb{R}\backslash\{-1,-2,\ldots,-3\}$ et

   $$\forall x \in \mathbb{R}\backslash\{-1,\ldots,-3\}, \quad f'(x) = -\sum_{k=1}^{3} \frac{k}{(x+k)^2} < 0.$$

   Ainsi f est strictement décroissante sur $]-\infty,-3[$, $]-3,-2[$, $]-2,-1[$ et $]-1,+\infty[$.
   En considérant les limites aux bords de chacun des intervalles ci-dessus et en appliquant le TVI, on obtient une unique solution dans chacun des intervalles $]-1,+\infty[$ et $]-3,-2[$, $]-2,-1[$ et pas ne solution sur $]-\infty,-3[$.

   (b) On commence par définir f et f' :

```
def f(x):
    return 1/(1+x)+2/(2+x)+3/(3+x)-1

def fprime(x):
    return -(1/(1+x)**2+2/(2+x)**2+3/(3+x)**2)
```

On applique ensuite notre fonction Newton et on trouve $\lambda \simeq 3.766\ldots$

(c) Soit $x \in \mathbb{R}$.

$$\operatorname{rg}\begin{pmatrix} -x & 2 & 3 \\ 1 & -x & 3 \\ 1 & 2 & -x \end{pmatrix} = \operatorname{rg}\begin{pmatrix} 1 & -x & 3 \\ -x & 2 & 3 \\ 1 & 2 & -x \end{pmatrix} = \operatorname{rg}\begin{pmatrix} 1 & -x & 3 \\ 0 & 2-x^2 & 3+3x \\ 0 & 2+x & -x-3 \end{pmatrix}$$
$$= \operatorname{rg}\begin{pmatrix} 2-x^2 & 3+3x \\ 2+x & -x-3 \end{pmatrix}$$

Donc x est une valeur propre de A si et seulement si

$$\det\begin{pmatrix} 2-x^2 & 3+3x \\ 2+x & -x-3 \end{pmatrix} = 0 \iff (2-x^2)(-x-3) - (3+3x)(2+x) = 0.$$

On vérifie que -1, -2 et -3 ne sont pas solution et en divisant par $(x+1)(x+2)(x+3)$ on obtient :

$$\begin{aligned} x \in \operatorname{Sp}(A) &\iff \frac{(x+3)(2-x^2)+3(1+x)(2+x)}{(x+1)(x+2)(x+3)} = 0 \\ &\iff \frac{2-x^2}{(x+1)(x+2)} + \frac{3}{3+x} = 0 \\ &\iff \frac{x+2-x-2+2-x^2}{(x+1)(x+2)} + \frac{3}{3+x} = 0 \\ &\iff \frac{1}{(x+1)} + \frac{-x-x^2}{(x+1)(x+2)} + \frac{3}{3+x} = 0 \\ &\iff f(x) - \frac{2}{2+x} + \frac{-x-x^2}{(x+1)(x+2)} = 0 \\ &\iff f(x) - \frac{x^2+3x+2}{(x+1)(x+2)} = 0 \\ &\iff f(x) - 1 = 0. \end{aligned}$$

(d)

```
1 print(max(la.eigvals(np.array([[0,2,3],[1,0,3],[1,2,0]]))))
2 print(Newton(f,fprime,0,5)[-1])
```

On remarque encore une convergence très rapide de la méthode de Newton puisqu'avec seulement $n = 5$, on obtient déjà les 8 premières décimales données par `eigvals`.

5. (**Niveau ENS**) On admettra le résultat suivant : soit f de classe C^2 sur un intervalle $[a, b]$; il existe $c \in]a, b[$ tel que :

$$f(b) = f(a) + f'(a)(b-a) + \frac{f''(c)(b-a)^2}{2}.$$

On considère f de classe C^2 sur $[a, b]$ telle que $f(c) < 0 < f(d)$ et $f' > 0$ sur $[a, b]$. On note alors $F : x \mapsto x - \dfrac{f(x)}{f'(x)}$.

(a) Les hypothèses impliquent que f est strictement croissante et $f(a) < 0 < f(b)$: par le corollaire du TVI $f(x) = 0$ admet une unique solution sur $[a, b]$ noté λ.

(b) Soit $x \in [a, b]$.

$$F(x) - \lambda = x - \frac{f(x)}{f'(x)} - \lambda = -\frac{(\lambda - x)f'(x) + f(x)}{f'(x)}.$$

En appliquant le résultat admis avec $a = x$ et $b = \lambda$, il existe c_x compris entre λ et x tel que :

$$0 = f(\lambda) = f(x) + f'(x)(\lambda - x) + \frac{f''(c_x)(\lambda - x)^2}{2}$$

i.e. $\dfrac{f''(c_x)(\lambda - x)^2}{2} = -f(x) - f'(x)(\lambda - x)$.
En substituant on obtient alors

$$F(x) - \lambda = \frac{f''(c_x)(x - \lambda)^2}{2f'(x)}.$$

(c) Comme f est de classe C^2, f'' et f' sont continues sur $[a, b]$ donc bornées et atteignent leurs bornes.

Notons $m = \min_{[a,b]} f' > 0$ et $M = \max_{[a,b]} |f''|$ et posons $C = \dfrac{M}{m}$. D'après la question précédente, pour tout $x \in [a, b]$:

$$|F(x) - \lambda| \leq C|x - \lambda|^2.$$

(d) Soit $\varepsilon > 0$. Si $x \in [\lambda - \varepsilon, \lambda + \varepsilon]$, alors on a :

$$|F(x) - \lambda| \leq C|x - \lambda|^2 \leq C\varepsilon^2.$$

Pour que $F(x)$ soit dans $[\lambda - \varepsilon, \lambda + \varepsilon]$, c'est-à-dire pour que $|F(x) - \lambda| \leq \varepsilon$, il suffit donc que :

$$C\varepsilon^2 \leq \varepsilon \quad i.e. \quad C\varepsilon \leq 1.$$

N'importe quel ε vérifiant cette condition convient.
Pour un tel ε, un récurrence immédiate montre que si $x_0 \in [\lambda - \varepsilon, \lambda + \varepsilon]$ alors pour tout $n \in \mathbb{N}$, $x_n \in [\lambda - \varepsilon, \lambda + \varepsilon]$.
On peut alors appliquer l'inégalité précédente en x_n :

$$\forall n \in \mathbb{N}, \quad C|x_{n+1} - \lambda| = C|F(x_n) - \lambda| \leq C^2|x_n - \lambda|^2.$$

Un récurrence montre alors :

$$\forall n \in \mathbb{N}, \quad C|x_n - \lambda| \leq (C\varepsilon)^{2^n}.$$

(e) En prenant $C\varepsilon < 1$, comme $(C\varepsilon)^{2^n}$ est négligeable devant $\dfrac{1}{2^n}$, la convergence de la méthode de Newton est plus rapide que celle de la méthode de dichotomie.