TP 15 informatique

Étude de suites réelles

BCPST 1 2019-2020

Points abordés

— suites de réels, tracé de graphes.

Exercice 1. Soit n un entier naturel non nul . On considère $f_n : \mathbb{R}+ \to \mathbb{R}, x \mapsto x^n + x^{n-1} + \ldots + x^2 + x - 1$.

1. Montrer que f_n s'annule une unique fois en un réel positif noté a_n.
2. En utilisant Python représenter les courbes de $f_1, \ldots, f_{20}$. Faire une conjecture sur la monotonie et la limite de la suite $(a_n)_{n\in\mathbb{N}^*}$.
3. (a) Donner le signe de $f_{n+1}(a_n)$.
   (b) En utilisant la monotonie de f_{n+1} comparer a_n et a_{n+1}.
   (c) Conclure quant à la monotonie de (a_n).
   (d) Prouver la convergence de (a_n) vers un réel notée l.
   (e) En utilisant pour $n > 1$ que $a_n \leq a_2 < 1$, déterminer $\lim\limits_{n\to+\infty} a_n^{n+1}$ et en déduire que $\dfrac{1}{1-l} - 2 = 0$.
   (f) Déduire la valeur de l.

Correction

Elément de correction. Soit n un entier naturel non nul . On considère $f_n : \mathbb{R}+ \to \mathbb{R}, x \mapsto x^n + x^{n-1} + \ldots + x^2 + x - 1$.

1. f_n est polynomiale donc dérivable sur $\mathbb{R}^+$ et sa dérivée est strictement positive. Donc f_n est strictement croissante sur $\mathbb{R}+$. De plus, $f_n(0) = -1, f_n(1) \geq 0$, f_n est continue sur $\mathbb{R}+$. En utilisant le T.V.I, on peut conclure qu'il existe un unique $a_n \in [0, 1]$ tel que $f_n(a_n) = 0$.
2. Voir Info. La suite $(a_n)_{n\in\mathbb{N}}$ semble être décroissante.
   (a) Par calcul, $f_{n+1}(a_n) = a_n^{n+1} \geq 0 = f_{n+1}(a_{n+1})$ car $a_n \geq 0$. Or f_{n+1} est strictement croissante sur $\mathbb{R}^+$. Donc $a_{n+1} \leq a_n$
   (b) La suite $(a_n)_{n\in\mathbb{N}}$ est donc décroissante.
   (c) La suite étant décroissante et minorée par 0, elle est donc convergente. Notons l sa limite.
   (d) Soit $n > 1$. On a $0 \leq a_n \leq a_2 < 1$. Donc $0 \leq a_n^{n+1} \leq a_2^{n+1}$. Comme $0 < a_2 < 1$, la suite géométrique de raison a_2 a pour limite 0. Par encadrement, on en déduit que $\lim_{n\to+\infty} a_n^{n+1} = 0$. Mais $\sum_{k=1}^{n}(a_n)^k - 1 = \sum_{k=0}^{n} a_n^k - 2 = \frac{1-(a_n)^{n+1}}{1-a_n} - 1 = 0$.
   En passant à la limite, on en déduit que $\frac{1}{1-l} - 2 = 0$.
   (e) Il en résulte que $l = \frac{1}{2}$.

Exercice 2. On définit la suite $(u_n)_{n\in\mathbb{N}}$ par : $u_0 = 2, \forall n \in \mathbb{N}, u_{n+1} = \frac{1}{2}(u_n + \frac{2}{u_n})$.

1. Montrer que pour tout $n \in \mathbb{N}, u_n > 1$.
2. Montrer que pour tout $n \in \mathbb{N}, 0 < u_{n+1} - \sqrt{2} < \frac{(u_n-\sqrt{2})^2}{2}$.
3. En déduire que pour tout $n \in \mathbb{N}, 0 < u_n - \sqrt{2} \leq \frac{(2-\sqrt{2})^{2^n}}{2^{2^n-1}}$
4. En déduire une fonction `approx(epsilon)` qui renvoie un couple (a, b) tel que $0 \leq b - a \leq \epsilon$ et $\sqrt{2} \in [a, b]$.

Correction

Elément de correction :

1. Soit $x > 0$. Résolvons l'inéquation
$$\frac{1}{2}(x + \frac{2}{x}) > 1.$$
Raisonnons par équivalence :
$$\begin{aligned} \tfrac{1}{2}(x + \tfrac{2}{x}) > 1 &\Leftrightarrow x + \tfrac{2}{x} - 2 > 0 \\ &\Leftrightarrow \tfrac{x^2 + -2x + 2}{x} > 0 \\ &\Leftrightarrow \tfrac{(x-1)^2+1}{x} > 0 \end{aligned}$$
x étant strictement positif, on en déduit que la dernière inégalité est vraie.
Pour tout $n \in \mathbb{N}$, on pose alors $P(n) : u_n > 1$. On raisonne par récurrence et pour l'hérédité, on utilise la résolution de l'inéquation précédente.
2. Soit $n \in \mathbb{N}$.
$$\begin{aligned} u_{n+1} - \sqrt{2} &= \tfrac{1}{2}(u_n + \tfrac{2}{u_n}) - \sqrt{2} \\ &= \tfrac{u_n^2 - 2\sqrt{2}u_n + 2}{2u_n} \\ &= \tfrac{(u_n - \sqrt{2})^2}{2u_n} \end{aligned}$$
Or $u_n > 1$. D'où $0 < u_{n+1} - \sqrt{2} < \frac{(u_n - \sqrt{2})^2}{2}$.
3. Pour tout $n \in \mathbb{N}$, on pose $P(n) : 0 < u_n - \sqrt{2} \leq \frac{(2-\sqrt{2})^{2^n}}{2^{2^n - 1}}$. Par récurrence sur n.
   — Initialisation : $u_0 - \sqrt{2} = \frac{(2-\sqrt{2})^{2^0}}{2^{2^0 - 1}}$. $P(0)$ est donc vraie.
   — Hérédité. Soit $n \in \mathbb{N}$. Supposons que $P(n)$ est vraie. Montrons que $P(n+1)$ est vraie.
   D'après la question 2 :
$$0 < u_{n+1} - \sqrt{2} < \frac{(u_n - \sqrt{2})^2}{2}.$$
   Or, d'après $P(n)$, $(u_n - \sqrt{2}) \leq \frac{\left(u_n \sqrt{2}\right)^{2^n}}{2^{2^n - 1}}$. D'où :
$$0 < (u_{n+1} - \sqrt{2}) < \frac{\left(\frac{\left(u_n - \sqrt{2}\right)^{2^n}}{2^{2^n - 1}}\right)^2}{2}.$$
   On obtient alors
$$0 < (u_{n+1} - \sqrt{2}) < \frac{\left(u_n - \sqrt{2}\right)^{2^{n+1}}}{2^{2^{n+1} - 1}}.$$
   $P(n+1)$ est donc vraie.
   — Conclusion : la propriété est donc vraie pour tout $n \in \mathbb{N}$.
   Comme $0 < 2 - \sqrt{2} < 1$, on en déduit que $\lim_{n \to +\infty} \frac{(2-\sqrt{2})^{2^n}}{2^{2^n - 1}} = 0$. D'après le théorème d'encadrement, on en déduit que $\lim_{n \to +\infty} u_n = \sqrt{2}$.
4. voir info.

Exercice 3. On fixe $x \in [0, 1]$ et n un entier naturel pair.

1. Montrer que pour tout $t \in [0, x]$, on a
$$\sum_{k=0}^{n+1} (-1)^k t^k \leq \frac{1}{1+t} \leq \sum_{k=0}^{n} (-1)^k t^k$$
2. En déduire que $\sum_{k=0}^{n+1} \frac{(-1)^k x^{k+1}}{k+1} \leq \ln(1+x) \leq \sum_{k=0}^{n} \frac{(-1)^{k+1}}{k+1}$.
3. Déduire un algorithme qui permet de calculer $\ln(1+x)$ pour $x \in [0, 1]$ à epsilon près et écrire une fonction `approx(x,ϵ)` qui renvoit une valeur approchée de $\ln(1+x)$ à ϵ près.
4. En déduire une approximation de $\ln(2)$ à l'aide de la fonction précédente.

5. De même, en adaptant le calcul précédent, on obtient de la même manière une approximation de $\ln(1-x)$ avec $x \in [0,1[$. Écrire ne fonction `approx_bis(x,`ϵ`)` qui renvoit une valeur approchée de $\ln(1-x)$ à ϵ près.
   Déterminer une approximation de $\ln(\frac{1}{2})$ comme précédemment, et en déduire une approximation de $\ln(2)$.
6. Comparer en les deux méthodes de calculs pour l'approximation de $\ln(2)$. On pourra par exemple compté le nombre d'additions nécessaires pour obtenir une valeur approchée à 10^{-10} près.

Correction

n fixe $x \in [0,1]$ et n un entier naturel pair.

1. Soit $t \in [0,x]$. On a, $-t \neq 1$. Donc d'après la somme des termes d'une suite géométrique :

$$\sum_{k=0}^{n+1}(-1)^k t^k = \frac{1-(-t)^{n+2}}{1-(-t)}, \sum_{k=0}^{n}(-1)^k t^k = \frac{1-(-t)^{n+1}}{1-(-t)}.$$

   D'où, car n est pair :

$$\sum_{k=0}^{n+1}(-1)^k t^k = \frac{1-t^{n+2}}{1+t}, \sum_{k=0}^{n}(-1)^k t^k = \frac{1+t^{n+1}}{1+t}.$$

   Or $-t^{n+2} \leq 0$ et $t^{n+1} \geq 0$. Donc :

$$\sum_{k=0}^{n+1}(-1)^k t^k \leq \frac{1}{1+t} \leq \sum_{k=0}^{n}(-1)^k t^k$$

2. L'inégalité étant vraie su $[0,x]$, en intégrant sur ce segment l'inégalité précédente :

$$\sum_{k=0}^{n+1}\frac{(-1)^k x^{k+1}}{k+1} \leq \ln(1+x) \leq \sum_{k=0}^{n}\frac{(-1)^{k+1}}{k+1}.$$

3. Déduire un algorithme qui permet de calculer $\ln(1+x)$ pour $x \in [0,1]$ à epsilon près et écrire une fonction `approx(x,`ϵ`)` qui renvoit une valeur approchée de $\ln(1+x)$ à ϵ près.

```
def approx(x,epsilon) :
    S0 = x
    S1 = x -x**2/2
    i=3
    while (S0-S1)>epsilon :
        S0 = S1 + x**i/i
        S1 = S0 -(x**(i+1))/(i+1)
        i = i+2
   return [S1,S0]
```

4. Il suffit de prendre le cas particulier $x = 1$.
5. De même, en adaptant le calcul précédent, on obtient de la même manière une approximation de $\ln(1-x)$ avec $x \in [0,1[$. On peut, remplacer x par $-x$ dans les calculs précédents. Du fait du changement de signe, les inégalités sont renversées et certains signes dans les sommes sont modifiés.

```
def approx\_bis(x,epsilon) :
    S0 = -x
    S1 = -x -x**2/2
    i=3
    while abs(S0-S1)>epsilon :
        S0 = S1 - x**i/i
        S1 = S0 -(x**(i+1))/(i+1)
        i = i+2
   return [S1,S0]
```

   Pour obtenir une approximation de $\ln(\frac{1}{2})$, il suffit de prendre $x = \frac{1}{2}$. Donc en multipliant par -1, on trouve une approximation de $\ln(2)$.
6. Considérons les deux codes suivants :

```
def approx_ln2(epsilon) :
    x = 1
    S0 = x
    S1 = x -x**2/2
    i=3
    compteur = 0
    while (S0-S1)>epsilon :
        S0 = S1 + x**i/i
        S1 = S0 -(x**(i+1))/(i+1)
        i = i+2
        compteur += 1
   return [S1,S0],compteur
```

et

```
def approx_bis(epsilon) :
    x = 1/2
    S0 = -x
    S1 = -x -x**2/2
    i=3
    compteur = 0
    while abs(S0-S1)>epsilon :
        S0 = S1 - x**i/i
        S1 = S0 -(x**(i+1))/(i+1)
        i = i+2
        compteur += 1
   return [S1,S0],compteur
```

Constatons que pour `approx_ln2(10**(-4))` il y a eu 4999 itérations dans la boucle, tandis que `approx_bis(10**(-4))` il n'y en n'a eu que 4. Ainsi, `approx_bis` est beaucoup plus efficace.