Thème 1 : Suites numériques

I. Séance 1 : définitions

Exercice 1
Soit $(u_n)_{n\in\mathbb{N}}$ une suite de réels et soit $\ell \in \mathbb{R}$.

1. Qu'est-ce qu'une suite monotone ?

Démonstration.
Une suite (u_n) est dite monotone si l'une des deux propriétés suivantes est vérifiée.

1) (u_n) est croissante *i.e.* : $\forall n \in \mathbb{N},\ u_{n+1} \geqslant u_n$.

2) (u_n) est décroissante *i.e.* : $\forall n \in \mathbb{N},\ u_{n+1} \leqslant u_n$.

Une suite (u_n) est monotone si ((u_n) est croissante) `OU` ((u_n) est décroissante).

□

2. Traduire en mathématiques (avec les quantificateurs) les propositions mathématiques suivantes.

a) La suite $(u_n)_{n\in\mathbb{N}}$ est décroissante.

Démonstration.

Une suite (u_n) est dite décroissante si : $\forall n \in \mathbb{N},\ u_{n+1} \leqslant u_n$.

□

b) La suite $(u_n)_{n\in\mathbb{N}}$ est majorée.

Démonstration.

Une suite (u_n) est dite majorée si : $\exists M \in \mathbb{R}, \forall n \in \mathbb{N},\ u_n \leqslant M$.

□

c) La suite $(u_n)_{n\in\mathbb{N}}$ converge vers $\ell \in \mathbb{R}$.

Démonstration.

Une suite (u_n) converge vers $\ell \in \mathbb{R}$ si : $\forall \varepsilon > 0, \exists n_0 \in \mathbb{N}, \forall n \geqslant n_0,\ \lvert u_n - \ell\rvert \leqslant \varepsilon$.

Commentaire

- Cette propriété signifie que quelle que soit la précision ε (> 0) choisie, on peut trouver un rang à partir duquel les éléments de la suite (u_n) ne s'écartent pas de ℓ de plus de ε.
- L'idée est que l'on a un contrôle aussi fin que souhaité sur les éléments de la suite (u_n) : à partir d'un certain rang, tous les élements de la suite sont dans l'intervalle $]\ell - \varepsilon, \ell + \varepsilon[$ où ε représente n'importe quel réel strictement positif choisi en amont.
- Pour plus de détails, on pourra se reporter au Chapitre « Convergences des suites réelles » de première année.

□

d) La suite $(u_n)_{n\in\mathbb{N}}$ diverge vers $+\infty$.

Une suite (u_n) diverge vers $+\infty$ si : $\forall A > 0, \exists n_0 \in \mathbb{N}, \forall n \geqslant n_0,\ u_n > A$.

Commentaire

- Cette propriété signifie que les termes de la suite (u_n) deviennent, à partir d'un certain rang, aussi grands que souhaités (*i.e.* plus grands que le réel A choisi en amont).
- Il est à noter qu'il n'est pas nécessaire, dans la définition de suite divergeant vers $+\infty$, de supposer $A > 0$. Plus précisément, on a :

$$u_n \underset{n\to+\infty}{\longrightarrow} +\infty \ \Leftrightarrow\ \forall A > 0, \exists n_0 \in \mathbb{N}, \forall n \geqslant n_0,\ u_n > A$$

  L'écriture (équivalente) donnée dans la démonstration présente l'avantage d'être un peu plus similaire à l'écriture de la propriété de convergence. On comprend aisément que ces deux écritures sont équivalentes. Comme on l'a vu, dire qu'une suite (u_n) divrege vers $+\infty$ signifie que les termes de la suite (u_n) deviennent à partir d'un certain rang aussi grands que souhaités. On peut décider, sans perte de généralité, que ce « grand » est strictement positif.
- On pourra une nouvelle fois se reporter au Chapitre « Converfences des suites réelles » pour plus de détails.

3. Reprendre les questions ***2.a)*** et ***2.b)*** dans le cas où les propriétés précédentes sont vérifiées seulement à partir d'un certain rang.

Démonstration.

- Dire qu'une propriété $\mathscr{P}(n)$ est vérifiée à partir d'un certain rang, cela signifie qu'**il existe** un entier n_0 tel que pour tout $n \geqslant n_0$, $\mathscr{P}(n)$ est vraie.
  Formellement, cela s'écrit :

$$\exists n_0 \in \mathbb{N},\ \forall n \in \mathbb{N},\ (n \geqslant n_0\ \Rightarrow\ \mathscr{P}(n))$$

  Par abus de notation, on privilégie souvent la forme suivante :

$$\exists n_0 \in \mathbb{N},\ \forall n \geqslant n_0,\ \mathscr{P}(n)$$

  en omettant ainsi de préciser que $n \in \mathbb{N}$ (sous-entendu).
- On voit apparaître cette construction pour les propriétés de convergence vers une réel ℓ et divergence vers l'infini (propriétés ***2.c)*** et ***2.d)***) : le contrôle des éléments de la suite a lieu à partir d'un certain rang n_0 (ce rang dépend des élements ε et A choisis).
- Pour écrire la notion de décroissance et le caractère majoré à partir d'un certain rang, il suffit de remplacer la construction « $\forall n \in \mathbb{N}$ » par « $\exists n_0 \in \mathbb{N}, \forall n \geqslant n_0$ ».

Une suite $(u_n)_{n\in\mathbb{N}}$ est décroissante à partir d'un certain rang si :
$\exists n_0 \in \mathbb{N},\ \forall n \geqslant n_0,\ u_{n+1} \leqslant u_n$.

Une suite $(u_n)_{n\in\mathbb{N}}$ est majorée à partir d'un certain rang si :
$\exists M \in \mathbb{R}, \exists n_0 \in \mathbb{N},\ \forall n \geqslant n_0,\ u_n \leqslant M$.

□

4. Écrire la négation des propositions de la question *2.*

Démonstration.
Il suffit d'appliquer les règles de négation des propriétés quantifiées.

Une suite (u_n) n'est pas décroissante si : $\exists n \in \mathbb{N},\ u_{n+1} > u_n$.

(il suffit de trouver un rang $n \in \mathbb{N}$ pour lequel $u_{n+1} > u_n$)

Une suite (u_n) n'est pas majorée si : $\forall M \in \mathbb{R}, \exists n \in \mathbb{N},\ u_n > M$.

(cela signifie qu'aucun réel $M \in \mathbb{R}$ n'est un majorant de la suite (u_n))

Une suite (u_n) ne converge par vers $\ell \in \mathbb{R}$ si : $\exists \varepsilon > 0, \forall n_0 \in \mathbb{N}, \exists n \geqslant n_0,\ \lvert u_n - \ell \rvert > \varepsilon$.

(pour un certain $\varepsilon > 0$, on ne peut trouver de rang n_0 à partir duquel $\lvert u_n - \ell \rvert \leqslant \varepsilon$ est vérifié)

Une suite (u_n) ne diverge par vers $+\infty$ si : $\exists A > 0, \forall n_0 \in \mathbb{N}, \exists n \geqslant n_0,\ u_n \leqslant A$.

(pour un certain $A > 0$, on ne peut trouver de rang n_0 à partir duquel $u_n > A$ est vérifié) □

Exercice 2. (**) *Vrai ou Faux ?*

1. Si la suite (u_n) diverge vers $+\infty$, alors elle n'est pas majorée.

Démonstration.
Vrai.
Procédons par l'absurde. On suppose que la suite (u_n) diverge vers $+\infty$ et qu'elle est majorée.

- La suite (u_n) étant majorée, il existe $M \in \mathbb{R}$ tel que : $\forall n \in \mathbb{N},\ u_n \leqslant M$.
- Notons $A = M + 1$.
  Comme la suite (u_n) diverge vers $+\infty$, il existe un rang n_0 tel que : $\forall n \geqslant n_0,\ u_n > M + 1$.

On en déduit que pour tout $n \geqslant n_0$:

$$M + 1 < u_n \leqslant M$$

Absurde !

Si la suite (u_n) diverge vers $+\infty$, alors elle n'est pas majorée.

Commentaire

La propriété à démontrer s'exprime sous la forme : `Si ...alors ....`
Il s'agit donc d'une implication. Plus précisément, on aurait pu l'écrire sous la forme :

$$u_n \underset{n\to+\infty}{\longrightarrow} +\infty \ \Rightarrow\ (u_n) \text{ n'est pas majorée}$$

Rappelons qu'une implication : $p \Rightarrow q$ a même valeur de vérité que : `NON`(p) `OU` q.
Procéder par l'absurde consiste à démontrer que la négation de la propriété à démontrer est fausse. Ainsi, si l'on souhaite démontrer par l'absurde $p \Rightarrow q$, on doit démontrer que `NON`$(p \Rightarrow q)$ est fausse. Or :

$$\texttt{NON}(p \Rightarrow q) \ \Leftrightarrow\ \texttt{NON}(\texttt{NON}(p)\ \texttt{OU}\ q) \ \Leftrightarrow\ p\ \texttt{ET}\ \texttt{NON}(q)$$

Ainsi, pour démontrer par l'absurde $p \Rightarrow q$, on suppose que les propriétés p et `NON`(q) sont vérifiées et on exhibe alors une propriété fausse. □

Commentaire

L'exercice est un « *Vrai ou Faux ?* ».
- Si l'énoncé est vrai, il faut en faire une démonstration.
- Si l'énoncé est faux, il suffit de fournir un contre-exemple.

2. Une suite (u_n) croissante à partir d'un certain rang est minorée.

Démonstration.
Vrai.
Supposons que la suite (u_n) est croissante à partir d'un certain rang. On en déduit qu'il existe un rang $n_0 \in \mathbb{N}$ tel que : $\forall n \geqslant n_0,\ u_{n+1} \geqslant u_n$. On en déduit notamment :

$$\forall n \geqslant n_0,\ u_n \geqslant u_{n_0}$$

Ainsi, la suite (u_n) est minorée par u_{n_0} à partir d'un certain rang (n_0 en l'occurrence).
Il reste à montrer qu'elle est minorée tout court.
Pour ce faire, considérons les éléments de la suite précédant le rang n_0 :

$$u_0,\ u_1,\ \ldots,\ u_{n_0-1}$$

Ces éléments sont en nombre fini et possèdent donc :
× un minimum : $a = \min\{u_n \mid n \in [\![0, n_0 - 1]\!]\}$,
× et un maximum $A = \max\{u_n \mid n \in [\![0, n_0 - 1]\!]\}$.
(on le mentionne mais cette propriété est inutile ici)

En résumé, on a démontré :

$$\forall n \in [\![0, n_0 - 1]\!],\ u_n \geqslant a \qquad \text{et} \qquad \forall n \geqslant n_0,\ u_n \geqslant U_{n_0}$$

En notant $m = \min(a, u_{n_0})$, on a alors :

$$\forall n \in \mathbb{N},\ u_n \geqslant m$$

Une suite (u_n) croissante à partir d'un certain rang est minorée.

Commentaire

- Dans cette question, on démontre notamment qu'une suite (u_n) minorée à partir d'un certain rang est minorée (tout court). Cette propriété est aussi vraie pour les suites majorées et pour les suites bornées. La démonstration est similaire à celle développée au-dessus.
- On peut démontrer, de manière similaire :

  (u_n) convergente $\Rightarrow$ (u_n) bornée

  Plus précisément, on démontre tout d'abord qu'une suite convergente est bornée à partir d'un certain rang (il suffit d'écrire la définition de suite convergente vers un réel ℓ). On conclut alors par la propriété du point précédent : la suite est bornée à partir d'un certain rang donc elle est bornée.

□

3. Si la suite $\big(|u_n|\big)$ converge alors la suite (u_n) converge.

Démonstration.
Faux.
On peut par exemple considérer la suite de terme général : $u_n = (-1)^n$.
On a alors : $|u_n| = \big|(-1)^n\big| = 1 \underset{n\to+\infty}{\longrightarrow} 1$.
La suite $\big(|u_n|\big)$ est bien convergente. Cependant, la suite (u_n) ne l'est pas. □

4. Si la suite $\big(|u_n|\big)$ tend vers 0 alors la suite (u_n) tend vers 0.

Démonstration.
Vrai.
C'est même une équivalence. Il suffit pour cela de revenir à la définition.

$$\begin{aligned}
&\text{La suite } \big(|u_n|\big) \text{ tend vers } 0\\
\Leftrightarrow\ &\forall \varepsilon > 0, \exists n_0 \in \mathbb{N}, \forall n \geqslant n_0,\ \big||u_n| - 0\big| \leqslant \varepsilon\\
\Leftrightarrow\ &\forall \varepsilon > 0, \exists n_0 \in \mathbb{N}, \forall n \geqslant n_0,\ |u_n| \leqslant \varepsilon\\
\Leftrightarrow\ &\forall \varepsilon > 0, \exists n_0 \in \mathbb{N}, \forall n \geqslant n_0,\ |u_n - 0| \leqslant \varepsilon\\
\Leftrightarrow\ &\text{La suite } (u_n) \text{ tend vers } 0
\end{aligned}$$

La suite $\big(|u_n|\big)$ tend vers 0 si et seulement si la suite (u_n) tend vers 0.

Commentaire

- Si une suite (u_n) converge vers un réel $\ell \in \mathbb{R}$ alors, pour toute fonction f continue en ℓ, la suite $\big(f(u_n)\big)$ est convergente, de limite $f(\ell)$.
- On en déduit en particulier que si (u_n) converge vers un réel $\ell \in \mathbb{R}$ alors $\big(|u_n|\big)$ est convergente, de limite $|\ell|$. Mais, comme on l'a vu dans la question ***3.***, la convergence de la suite $\big(|u_n|\big)$ ne permet pas de démontrer la convergence de la suite (u_n).

□

5. Une suite convergente est monotone à partir d'un certain rang.

Démonstration.
Faux.
La suite de terme général $u_n = \dfrac{(-1)^n}{n}$ est convergente, de limite 0. On peut le montrer à l'aide de la question précédente, en remarquant :

$$|u_n| = \left|\frac{(-1)^n}{n}\right| = \frac{|(-1)^n|}{|n|} = \frac{1}{n} \underset{n\to+\infty}{\longrightarrow} 0$$

Cette suite change de signe d'un rang sur l'autre et ainsi :

$$\forall n \in \mathbb{N},\ u_{2n+1} < u_{2n} \quad \texttt{ET} \quad u_{2n+2} > u_{2n+1}$$

On en déduit qu'elle n'est pas monotone à partir d'un certain rang. □

6. Une suite convergente et majorée est croissante.

Démonstration.
Faux.
La suite de terme général $u_n = \dfrac{1}{n} - 1$ est :
× convergente, de limite 1.
× majorée par 1 puisque pour tout $n \in \mathbb{N}$: $\dfrac{1}{n} \leqslant 1$ et donc $\dfrac{1}{n} - 1 \leqslant 0$.
Pour autant, cette suite (u_n) n'est pas croissante. Pour tout $n \in \mathbb{N}$:

$$u_{n+1} - u_n = \left(\frac{1}{n+1} - \cancel{1}\right) - \left(\frac{1}{n} - \cancel{1}\right) = \frac{1}{n+1} - \frac{1}{n} = \frac{\cancel{n} - (\cancel{n}+1)}{n(n+1)} = \frac{-1}{n(n+1)} < 0$$

(en l'occurrence, la suite (u_n) est strictement décroissante) □

7. Une suite divergeant vers $+\infty$ est croissante à partir d'un certain rang.

Démonstration.
Faux.
La suite de terme général $u_n = n - (-1)^n$ diverge vers $+\infty$ en tant que somme d'une suite qui diverge vers $+\infty$ ($n \underset{n\to+\infty}{\longrightarrow} +\infty$) et d'une suite bornée ($\forall n \in \mathbb{N}$, $|(-1)^n| \leqslant 1$).
Or pour tout $n \in \mathbb{N}$:

$$\begin{aligned}
u_{n+1} - u_n &= \left(\cancel{n} + 1 - (-1)^{n+1}\right) - \left(\cancel{n} - (-1)^n\right) \\
&= 1 - (-1)^{n+1} - (-1)^{n+1} && \textit{(car } -(-1)^n = (-1) \times (-1)^n = (-1)^{n+1}\textit{)} \\
&= 1 - 2\,(-1)^{n+1} \\
&= \begin{cases} 3 & \text{si } n \text{ est pair} \\ -1 & \text{si } n \text{ est impair} \end{cases}
\end{aligned}$$

Et ainsi la suite (u_n) n'est pas croissante à partir d'un certain rang. □

8. Une suite strictement croissante diverge vers $+\infty$.

Démonstration.
Faux.
La suite de terme général $u_n = 1 - \dfrac{1}{n}$ est :
× strictement croissante puisque pour tout $n \in \mathbb{N}$:

$$u_{n+1} - u_n = \left(\cancel{1} - \frac{1}{n+1}\right) - \left(\cancel{1} - \frac{1}{n}\right) = \frac{1}{n} - \frac{1}{n+1} = \frac{(\cancel{n}+1) - \cancel{n}}{n(n+1)} = \frac{1}{n(n+1)} > 0$$

× convergente de limite 1.
En vertu de l'unicité de la limite, si elle existe, d'une suite, la suite (u_n) ne tend pas vers $+\infty$.

Commentaire

On a vue en question ***1.*** qu'une suite qui diverge vers $+\infty$ n'est pas majorée. On aurait pu utiliser cet argument pour conclure puisque, pour tout $n \in \mathbb{N}$:

$$u_n = 1 - \frac{1}{n} \leqslant 1$$

La suite (u_n) étant majorée par 1, elle ne peut diverger vers $+\infty$.

□

9. Une suite strictement décroissante diverge vers $-\infty$.

Démonstration.
Faux.
La suite de terme général $u_n = \dfrac{1}{n} - 1$ est
× strictement décroissante (*cf* question ***6.***).
× convergente, de limite -1.
Ainsi, la suite (u_n) n'est pas divergente vers $-\infty$. □

10. Si (u_n) est croissante et, pour tout $n \in N$: $u_n \leqslant v_n$ alors (v_n) est croissante.

Démonstration.
Faux.
On considère les suites de terme général :

$$u_n = 1 - \frac{1}{n} \quad \text{et} \quad v_n = 3 + (-1)^n$$

Alors, pour tout $n \in \mathbb{N}$:

$$u_n = 1 - \frac{1}{n} \leqslant 1 \leqslant 2 \leqslant 3 + (-1)^n = v_n$$

Enfin, pour tout $n \in \mathbb{N}$:

$$\begin{aligned} v_{n+1} - v_n &= \left(3 + (-1)^{n+1}\right) - \left(3 + (-1)^n\right) \\ &= (-1)^{n+1} + (-1)^{n+1} \\ &= 2\,(-1)^{n+1} \\ &= \begin{cases} -2 & \text{si } n \text{ est pair} \\ 2 & \text{si } n \text{ est impair} \end{cases} \end{aligned}$$

Ainsi, la suite (v_n) n'est pas croissante. □

11. Si (u_n) tend vers 0 et (v_n) tend vers $+\infty$, alors on ne peut conclure sur la limite du quotient $\dfrac{u_n}{v_n}$.

Démonstration.
Faux.

$$\left.\begin{array}{l} u_n \underset{n\to+\infty}{\longrightarrow} 0 \\ v_n \underset{n\to+\infty}{\longrightarrow} +\infty \end{array}\right\} \Rightarrow \frac{u_n}{v_n} = u_n \times \frac{1}{v_n} \underset{n\to+\infty}{\longrightarrow} 0 \times 0 = 0$$

□

12. Si (u_n) est divergente, alors la suite définie par : $\forall n \in \mathbb{N},\ v_n = u_{n+1} - u_n$ est divergente.

Démonstration.
Faux.
La suite de terme général $u_n = n$ diverge (vers $+\infty$).
Or, pour tout $n \in \mathbb{N}$: $u_{n+1} - u_n = (n+1) - n = 1$.
Ainsi, la suite de terme général $v_n = u_{n+1} - u_n$ est convergente (car constante) de limite 1. □

13. Si (u_n) tend vers $\ell \neq 0$ alors : $\lim\limits_{n\to+\infty} u_{n+1} - u_n = 0$ et $\lim\limits_{n\to+\infty} \dfrac{u_{n+1}}{u_n} = 1$.

Démonstration.
Vrai.
Considérons une suite (u_n) convergente de limite $\ell \neq 0$. Alors :

$$u_{n+1} - u_n \underset{n\to+\infty}{\longrightarrow} \ell - \ell = 0 \quad \text{et} \quad \frac{u_{n+1}}{u_n} \underset{n\to+\infty}{\longrightarrow} \frac{\ell}{\ell} = 1$$

□

Commentaire

- Rappelons que si la suite (u_n) admet pour limite $\ell \in \overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$, alors, pour toute application strictement croissante $\varphi : \mathbb{N} \to \mathbb{N}$, la suite $(u_{\varphi(n)})$ admet pour limite ℓ. Les suites ainsi construites sont appelées sous-suites (ou suites extraites) de la suite (u_n). En particulier, si (u_n) admet la limite ℓ, il en est de même des suites (u_{n+1}), (u_{2n}) ou encore (u_{3n-1}) ...
- Notons qu'une suite peut admettre des sous-suites convergentes sans pour autant être elle-même convergente. On peut par exemple, considérer la suite de terme général $u_n = (-1)^n$. Cette suite n'admet pas de limite mais les suites (u_{2n}) et (u_{2n+1}) en ont bien une.
  En effet, pour tout $n \in \mathbb{N}$:
  × $u_{2n} = (-1)^{2n} = \left((-1)^2\right)^n = 1 \underset{n\to+\infty}{\longrightarrow} 1$,
  × $u_{2n+1} = (-1)^{2n+1} = \left((-1)^2\right)^n \times (-1) = -1 \underset{n\to+\infty}{\longrightarrow} -1$.
- La limite ℓ d'une sous-suite $(u_{\varphi(n)})$ d'une suite (u_n) est appelée **valeur d'adhérence** de la suite (u_n). Dans l'exemple précédent, la suite $\left((-1)^n\right)_{n\in\mathbb{N}}$ admet deux valeurs d'adhérence distinctes. Elle ne peut admettre de limite.
- Comme on l'a vu dans le point précédent, la connaissance des limites de certaines sous-suites peut permettre de réfuter l'existence d'une limite pour (u_n). On peut aussi démontrer qu'une suite admet une limite ℓ si on connaît la valeur de certaines de ses sous-suites. C'est ce que stipule le théorème de recouvrement :

$$\left.\begin{array}{r} u_{2n} \underset{n\to+\infty}{\longrightarrow} \ell \\ u_{2n+1} \underset{n\to+\infty}{\longrightarrow} \ell \end{array}\right\} \quad \Rightarrow \quad u_n \underset{n\to+\infty}{\longrightarrow} \ell$$

  L'idée de la démonstration est la suivante :
  × comme $u_{2n} \underset{n\to+\infty}{\longrightarrow} \ell$, à partir d'un certain rang, tous les termes d'indices pairs de la suite (u_n) sont « proches » de ℓ,
  × comme $u_{2n+1} \underset{n\to+\infty}{\longrightarrow} \ell$, à partir d'un certain rang, tous les termes d'indices impairs de la suite (u_n) sont eux aussi « proches » de ℓ.
  Ainsi, à partir d'un certain rang, tous les termes de la suite (u_n) sont « proches » de ℓ.
  Ceci démontre que la suite (u_n) admet pour limite ℓ.
  Il faut bien comprendre que chaque terme de la suite (u_n) est soit d'indice pair, soit d'indice impair. Ainsi, l'information donnée en hypothèse permet de recouvrir tous les cas possibles. On pourra d'ailleurs énoncer des théorèmes analogues en considérant des recouvrements différents :

$$\left.\begin{array}{r} u_{3n} \underset{n\to+\infty}{\longrightarrow} \ell \\ u_{3n+1} \underset{n\to+\infty}{\longrightarrow} \ell \\ u_{3n+2} \underset{n\to+\infty}{\longrightarrow} \ell \end{array}\right\} \quad \Rightarrow \quad u_n \underset{n\to+\infty}{\longrightarrow} \ell$$

Séance 2 : manipulations de base (et Python)

Exercice 3
On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N},\ u_{n+1} = \dfrac{u_n}{1+u_n^2} \end{cases}$$

1. a) Montrer : $\forall n \in \mathbb{N},\ u_n > 0$.

Démonstration.
Démontrons par récurrence : $\forall n \in \mathbb{N},\ \mathscr{P}(n)$ où $\mathscr{P}(n) : u_n > 0$.

- **Initialisation** :
  Par définition : $u_0 = 1 > 0$.
  D'où $\mathscr{P}(0)$.
- **Hérédité** : soit $n \in \mathbb{N}$.
  Supposons $\mathscr{P}(n)$ et démontrons $\mathscr{P}(n+1)$ (*i.e.* : $u_{n+1} > 0$).
  Par hypothèse de récurrence : $u_n > 0$.
  De plus : $1 + {u_n}^2 > 0$. On en déduit :

$$u_{n+1} = \frac{u_n}{1+u_n^2} > 0$$

  D'où $\mathscr{P}(n+1)$.

Par principe de récurrence : $\forall n \in \mathbb{N},\ u_n > 0$.

□

b) Montrer que la suite $(u_n)_{n\in\mathbb{N}}$ est monotone.

Démonstration.
Soit $n \in \mathbb{N}$.
D'après la question précédente, $u_n \neq 0$. En particulier : $u_n \neq 0$.
On peut donc considérer :

$$\frac{u_{n+1}}{u_n} = \frac{\frac{u_n}{1+u_n^2}}{u_n} = \frac{\cancel{u_n}}{1+u_n^2}\,\frac{1}{\cancel{u_n}} = \frac{1}{1+u_n^2}$$

Comme $u_n^2 > 0$ alors $1 + u_n^2 > 1$.
Par stricte décroissance de l'application inverse sur $]0, +\infty[$: $\dfrac{1}{1+u_n^2} < 1$. On en conclut :

$$\frac{1}{1+u_n^2} = \frac{u_{n+1}}{u_n} < 1$$

Et en multipliant de part et d'autre par $u_n > 0$:

$$u_{n+1} < u_n$$

La suite $(u_n)_{n\in\mathbb{N}}$ est donc strictement décroissante.

Commentaire

- Pour déterminer la monotonie d'une suite, on peut toujours étudier le signe de la quantité $u_{n+1} - u_n$. Si on reprend la question précédente :

$$u_{n+1} - u_n = \frac{u_n}{1+u_n^2} - u_n = \frac{u_n - u_n\,(1+u_n^2)}{1+u_n^2} = \frac{-u_n^3}{1+u_n^2} < 0$$

  En effet, $1 + u_n^2 > 0$ et comme $u_n > 0$ alors : $u_n^3 > 0$.
- Dans la démonstration précédente, u_{n+1} apparaît naturellement sous forme d'un produit (ou plutôt d'un quotient). La démonstration a alors consisté à former le quotient $\dfrac{u_{n+1}}{u_n}$ et à le comparer à 1. Et comme $u_n > 0$:

$$\frac{u_{n+1}}{u_n} < 1 \;\Rightarrow\; u_{n+1} < u_n$$

  On notera au passage que l'hypothèse $u_n > 0$ est fondamentale et ce pour deux raisons :
  - × comme $u_n \neq 0$, on peut former le quotient $\frac{u_{n+1}}{u_n}$,
  - × si $u_n < 0$ alors : $\dfrac{u_{n+1}}{u_n} < 1 \;\Rightarrow\; u_{n+1} > u_n$.
- Comme on le voit, il faut toujours conclure en revenant à la comparaison entre u_n et u_{n+1}. On peut donc privilégier la méthode consistant à étudier le signe de $u_{n+1} - u_n$ quelle que soit la forme du terme général u_n.

□

c) Étudier la convergence de la suite $(u_n)_{n\in\mathbb{N}}$.

Démonstration.

- La suite $(u_n)_{n\in\mathbb{N}}$ est :
  - × décroissante,
  - × minorée par 0 (car $u_n > 0$ pour tout $n \in \mathbb{N}$).

  On en déduit que la suite $(u_n)_{n\in\mathbb{N}}$ converge vers un réel $\ell \geqslant 0$.
- Déterminons alors la valeur de ℓ.
  Par passage à la limite dans la relation de définition de u_{n+1} :

$$\begin{array}{ccc} u_{n+1} & = & \dfrac{u_n}{1+u_n^2} \\ \downarrow{\scriptstyle n\to+\infty} & & \downarrow{\scriptstyle n\to+\infty} \\ \ell & = & \dfrac{\ell}{1+\ell^2} \end{array}$$

  Ainsi, ℓ est solution de l'équation : $\ell(1+\ell^2) = \ell$. Or :

$$\ell(1+\ell^2) = \ell \;\Leftrightarrow\; \cancel{\ell} + \ell^3 = \cancel{\ell} \;\Leftrightarrow\; \ell^3 = 0 \;\Leftrightarrow\; \ell = 0$$

  La suite $(u_n)_{n\in\mathbb{N}}$ converge vers 0.

□

2. a) Écrire en **Python** la fonction `calculSuite` qui prend en paramètre un entier `n` et renvoie le $\texttt{n}^{\text{ème}}$ terme de la suite $(u_n)_{n\in\mathbb{N}}$.

Démonstration.
On propose la fonction suivante :

```
1  def calculSuite(n) :
2      u = 1
3      for i in range(n) :
4          u = u / (1 + u**2)
5      return u
```

Détaillons les éléments de ce script.

- **Début de la fonction**
  L'énoncé commence par préciser la structure de la fonction :
  - × cette fonction se nomme `calculSuite`,
  - × elle prend en paramètre d'entrée l'entier `n`,
  - × elle renvoie la valeur stockée dans la variable `u`.

```
1  def calculSuite(n) :
```

```
5      return u
```

  La variable `u`, qui contiendra les valeurs successives de la suite (u_n) est initialisée à 1 : la valeur de u_0.

```
2      u = 1
```

- **Structure itérative**
  Les lignes 3 à 4 consistent à calculer les valeurs successives de la suite (u_n).
  Pour cela, on utilise une structure itérative (boucle for) :

```
3      for i in range(n) :
4          u = u / (1 + u**2)
```

  On tire ici partie de la définition récursive (d'ordre 1) de cette suite. La nouvelle valeur de la suite, que l'on stockera dans la variable `u`, est obtenue à l'aide de la valeur précédente qui est celle alors stockée dans `u`.
- **Fin de la fonction**
  À l'issue de cette boucle, la variable `u` contient la quantité $u_{\texttt{n}}$ où `n` est le paramètre entré lors de l'appel de la fonction.

Commentaire

- Afin de permettre une bonne compréhension des mécanismes en jeu, on a détaillé la réponse à cette question. Cependant, proposer un programme **Python** correct démontre la bonne compréhension de ces mécanismes et permet certainement d'obtenir la majorité des points alloués à cette question.
- Dans l'énoncé, on demande de calculer le $\mathtt{n}^{\text{ème}}$ terme de la suite. Le premier terme étant u_0, le $\mathtt{n}^{\text{ème}}$ est $u_{\mathtt{n}-1}$ et pas $u_{\mathtt{n}}$. Généralement, on demande dans les énoncés de renvoyer $u_{\mathtt{n}}$ ce qui lève toute ambiguïté.
- Dans tous les cas, il faut vérifier si la suite commence par le terme d'indice 0 ou celui d'indice 1. Plus précisément :
  - × si la suite commence au terme d'indice 0, il faut $\mathtt{n}$ itérations pour obtenir $u_{\mathtt{n}}$.
  - × si la suite commence au terme d'indice 1, il faut $\mathtt{n}-1$ itérations pour obtenir $u_{\mathtt{n}}$.

Commentaire

- Le programme **Python** consiste à mettre à jour successivement la variable `u` jusqu'à obtention de la valeur que l'on souhaite calculer. L'idée est la suivante.
  Si avant le $i^{\text{ème}}$ tour de boucle (avec `i` $\in [\![1, \texttt{n}-1]\!]$) :

  $$\text{la variable } \texttt{u} \text{ contient la valeur } u_{i-1}$$

  alors, à l'issue de ce tour de boucle :

  $$\text{la variable } \texttt{u} \text{ contient la valeur } u_i$$

  Cette propriété est ce qu'on appelle un **invariant de boucle**. Elle permet d'assurer la correction de la fonction implémentée et notamment le fait qu'à l'issue du dernier tour de boucle la variable `u` contient $u_{\texttt{n}-1}$.

□

b) Quel appel permet de calculer u_8 ?

Démonstration.
Le terme u_8 est le $9^{\text{ème}}$ terme de la suite.

Pour calculer u_8 à l'aide de la fonction précédente, il faut réaliser l'appel : `calculSuite(9)`.

□

3. a) Écrire en **Python** la fonction `calculPremiersTermes` qui prend en paramètre un entier `n` et renvoie la liste `L` contenant les `n` premiers termes de la suite $(u_n)_{n\in\mathbb{N}}$.

Démonstration.
On propose la fonction suivante :

```
1  def calculPremiersTermes(n) :
2      L = [1]
3      for i in range(n-1) :
4          L.append( L[i] / (1 + L[i]**2) )
5      return L
```

- **Début de la fonction**
  L'énoncé commence par préciser la structure de la fonction :
  - × cette fonction se nomme `calculPremiersTermes`,
  - × elle prend en paramètre d'entrée l'entier `n`,
  - × elle renvoie la liste stockée dans la variable `L`.

```
1  def calculPremiersTermes(n) :
```

```
5      return L
```

La variable `L`, liste qui contiendra à terme les `n` premières valeurs de (u_n), est initialisée à une liste à un élément contenant la valeur de u_0 : 1.

```
2      L = [1]
```

- **Structure itérative**
  Les lignes 3 à 4 consistent à calculer les valeurs successives de la suite (u_n) et à les concaténer à la liste `L`. Pour cela, on utilise une structure itérative (boucle for) :

```
3       for i in range(n-1) :
4           L.append( L[i] / (1 + L[i]**2) )
```

  On tire ici partie de la définition récursive (d'ordre 1) de cette suite. Plus précisément, la nouvelle valeur de la suite, qui sera concaténée à la liste `L`, est obtenue à l'aide de la valeur précédente qui est celle stockée dans le $\texttt{i}^{\text{ème}}$ élément de la liste `L`.
- **Fin de la fonction**
  À l'issue de cette boucle, la variable `L` contient les `n` premiers éléments de la suite (u_n).

Commentaire

- Le programme **Python** consiste à mettre à jour successivement la liste `L` jusqu'à stockage des `n` premières valeurs de la suite (u_n). L'idée est la suivante.
  Si avant le $i^{\text{ème}}$ tour de boucle (avec $\texttt{i} \in [\![1, \texttt{n}-1]\!]$) :

  la variable `L` est la liste des `i` premières valeurs de la suite (u_n)

  alors, à l'issue de ce tour de boucle :

  la variable `L` est la liste des `i` + 1 premières valeurs de la suite (u_n)

  Cette propriété est ce qu'on appelle un **invariant de boucle**. Elle permet d'assurer la correction de la fonction implémentée et notamment le fait qu'à l'issue du dernier tour de boucle la variable `L` contient les `n` premières valeurs de la suite (u_n).
- On peut présenter la boucle légérement différemment :

```
3       for i in range(n-1) :
4           L = L + [ L[i] / (1 + L[i]**2) ]
```

  Cette écriture met en avant l'opération « + » de concaténation de listes.
  Ces deux choix sont pertinents et seront donc évidemment tous les deux acceptés aux concours.

□

b) On considère alors le programme **Python** suivant :

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  n = 100
5  absc = np.linspace(0, n-1, n)
6  t = calculPremiersTermes(n)
7  plt.plot(absc, t)
8  plt.show()
```

Que réalise ce programme ?

Démonstration.

- La variable `absc` contient le tableau (`np.array`) $[0, 1, 2, \ldots, \texttt{n}-1]$.
- La variable `t` contient la liste $[u_0, u_1, u_2, \ldots, u_{\texttt{n}-1}]$ obtenue grâce à un appel à la fonction précédente.

- Enfin, la fonction `plot` permet la représentation graphique des points de coordonnées (i, u_i) pour i variant de 0 à `n` -1. Plus précisément, la fonction `plot` prend en paramètre deux listes ou tableaux de réels :
  - × `absc` contient les abscisses successives des points à représenter (les éléments i).
  - × `t` contient les ordonnées successives des points à représenter (les éléments u_i).

Ce programme permet de représenter graphiquement les 100 premiers éléments de la suite (u_n). □

4. a) Justifier qu'il existe un rang $n_0 \in \mathbb{N}$ tel que : $\forall n \geqslant n_0,\ |u_n| \leqslant 10^{-4}$.

Démonstration.
En question ***1.c)***, on a démontré : $u_n \underset{n\to+\infty}{\longrightarrow} 0$.
Ainsi, pour tout $\varepsilon > 0$, il existe un rang n_0 tel que : $\forall n \geqslant n_0,\ |u_n - 0| \leqslant \varepsilon$.

En prenant $\varepsilon = 10^{-4}$, on a bien démontré qu'il existe un rang $n_0 \in \mathbb{N}$ tel que : $\forall n \geqslant n_0,\ |u_n| \leqslant 10^{-4}$. □

b) Écrire en **Python** un programme qui permet de déterminer le premier entier n tel que $|u_n| \leqslant 10^{-4}$.

Démonstration.
On propose le programme suivant :

```
1  n = 0
2  u = 1
3  while abs(u) > 10**(-4) :
4      n = n + 1
5      u = u / (1 + u**2)
6  print(n)
```

ou

```
1  def premierIndice() :
2      n = 0
3      u = 1
4      while abs(u) > 10**(-4) :
5          n = n + 1
6          u = u / (1 + u**2)
7      return n
```

Détaillons les éléments de ce script.

- **Début du programme**
  Commençons par préciser la structure de la fonction :
  - × cette fonction se nomme `premierIndice`,
  - × elle ne prend aucun paramètre en entrée,
  - × elle renvoie la valeur stockée dans la variable `n`.

```
1  def premierIndice() :
```

```
7      return n
```

On initialise ensuite la variable `n` à 0.

```
2  n = 0
```

La variable `u`, qui contiendra les valeurs successives de la suite (u_n) est initialisée à 1 : la valeur de u_0.

```
3      u = 1
```

- **Structure itérative**
  Les lignes 4 à 6 consistent à :
  1) déterminer un entier n tel que : $|u_n| \leqslant 10^{-4}$,
  2) calculer les valeurs succesives de u_n.
  On doit donc :
  1) incrémenter la variable `n` de 1 jusqu'à ce que : $|u_n| \leqslant 10^{-4}$. Autrement dit, on doit incrémenter la variable `n` de 1 tant que : $|u_n| > 10^{-4}$.
  Pour cela on met en place une boucle while :

```
4  while abs(u) > 10**(-4) :
```

  Puis on met à jour la variable `n`.

```
5          n = n + 1
```

  2) calculer les valeurs successives de la suite (u_n), à l'aide de la définition récursive d'ordre 1 de cette suite :

```
6          u = u / (1 + u^2)
```

- **Fin du programme**
  À l'issue de cette boucle, la variable `n` contient le premier entier n tel que : $|u_n| \leqslant 10^{-4}$.

Commentaire

- Remarquons tout d'abord que la terminaison de la boucle `while` est assurée par la propriété décrite en question ***4.a)*** : il existe forcément un rang n_0 pour lequel $u_{n_0} \leqslant 10^{-4}$.
- En sortie de boucle, on est assuré que la variable `u` contient un élément de la suite (u_n) plus petit en valeur absolue que 10^{-4}. L'indice de cet élément est repéré à l'aide d'un compteur `n`, initialement affecté à 0 et incrémenté de 1 à chaque tour de boucle.
- Si l'énoncé demande explicitement d'écrire une fonction, il faut obligatoirement se servir de la construction à l'aide de `function`. Comme on demande d'écrire un programme, on peut :
  × soit écrire un programme qui permet d'**afficher** la valeur de `n` (à l'aide de l'instruction `print`). C'est la solution à gauche ci-dessus.
  × soit écrire une fonction qui permet de **calculer** et renvoyer la valeur de `n`. L'intérêt d'une fonction est que son résultat peut être utilisé par un simple appel.
  C'est la solution à droite ci-dessus. Ce choix semble ici un peu artificiel puisque cette fonction ne requiert aucun argument.

□

Séance 3 : suites récurrentes linéaires usuelles

Exercice 4
On considère une suite $(u_n)_{n\in\mathbb{N}}$ telle que $u_0 > 0$, $u_1 > 0$, et vérifiant la relation de récurrence :

$$\forall n \in \mathbb{N},\ u_{n+2} = u_n^3 \times u_{n+1}^2$$

1. Montrer par une récurrence double que $u_n > 0$ pour tout entier naturel n.

Démonstration.
Démontrons par récurrence double : $\forall n \in \mathbb{N},\ \mathscr{P}(n)$.
où $\mathscr{P}(n)$: $u_n > 0$.

- **Initialisation**

  Par définition, $u_0 > 0$ et $u_1 > 0$.

  D'où $\mathscr{P}(0)$ et $\mathscr{P}(1)$.

- **Hérédité** : soit $n \in \mathbb{N}$.
  Supposons $\mathscr{P}(n)$ et $\mathscr{P}(n+1)$ et démontrons $\mathscr{P}(n+2)$ (*i.e.* $u_{n+2} > 0$).

  Par définition, on a : $u_{n+2} = u_n^3 \times u_{n+1}^2$.

  Or, par hypothèses de récurrence ($\mathscr{P}(n)$ `ET` $\mathscr{P}(n+1)$), on sait : $u_n > 0$ et $u_{n+1} > 0$.

  D'où : $u_n^3 > 0$ et $u_{n+1}^2 > 0$. On a donc : $u_{n+2} > 0$.

  D'où $\mathscr{P}(n+2)$.

Ainsi, par principe de récurrence, on a : $\forall n \in \mathbb{N},\ \mathscr{P}(n)$. □

2. On note $t_n = \ln(u_n)$.

a) Montrer que la suite (t_n) vérifie : $\forall n \in \mathbb{N},\ t_{n+2} = 3t_n + 2t_{n+1}$.

Démonstration.
Soit $n \in \mathbb{N}$.

- Notons tout d'abord que $\ln(u_n)$ est bien définie puisque : $u_n > 0$ d'après la question précédente.
- On a :

$$\begin{aligned} t_{n+2} &= \ln(u_{n+2}) = \ln(u_n^3 \times u_{n+1}^2) \\ &= \ln(u_n^3) + \ln(u_{n+1}^2) \\ &= 3\ln(u_n) + 2\ln(u_{n+1}) \\ &= 3t_n + 2t_{n+1} \end{aligned}$$

$\forall n \in \mathbb{N},\ t_{n+2} = 3t_n + 2t_{n+1}$ □

b) De quel type de suite s'agit-il ?

Démonstration.

D'après la question précédente, la suite (t_n) est une suite récurrente linéaire d'ordre 2. □

3. Déterminer, en fonction de u_0 et u_1, le terme général de la suite (t_n).

Démonstration.

- L'équation caractéristique associée à la suite (t_n) est $x^2 = 3 + 2x$.
  Notons P le polynôme : $P(X) = X^2 - 2X - 3$.
  - Ce polynôme admet $r_1 = -1$ comme racine évidente.
  - Sa seconde racine est donnée par le calcul : $r_1 \times r_2 = \dfrac{c}{a} = \dfrac{-3}{1} = -3$.
    Comme $r_1 = -1$, on en déduit $r_2 = -(-3) = 3$.
- On en déduit la formule explicite de (t_n) :

$$\forall n \in \mathbb{N},\ t_n = \lambda r_1^n + \beta r_2^n = \lambda(-1)^n + \beta 3^n$$

où les valeurs λ et β sont données par le système : $(S) \left\{ \begin{array}{rcrcl} t_0 & = & \lambda & + & \beta \\ t_1 & = & \lambda r_1 & + & \beta r_2 \end{array} \right.$. Résolvons-le.

$$(S) \Leftrightarrow \left\{ \begin{array}{rcrcll} t_0 & = & \lambda & + & \beta & (L_1) \\ t_1 & = & -\lambda & + & 3\beta & (L_2) \end{array} \right. \Leftrightarrow \left\{ \begin{array}{rcll} t_0 + t_1 & = & 4\beta & (L_1)+(L_2) \\ 3t_0 - t_1 & = & 4\lambda & 3(L_1)-(L_2) \end{array} \right.$$

On en déduit : $\lambda = \dfrac{1}{4}(3t_0 - t_1)$ et $\beta = \dfrac{1}{4}(t_0 + t_1)$ avec $t_0 = \ln(u_0)$ et $t_1 = \ln(u_1)$.

Enfin : $3t_0 - t_1 = 3\ln(u_0) - \ln(u_1) = \ln(u_0^3) - \ln(u_1) = \ln\left(\dfrac{u_0^3}{u_1}\right)$.
Et : $t_0 + t_1 = \ln(u_0) + \ln(u_1) = \ln(u_0 u_1)$.

$$\boxed{\forall n \in \mathbb{N},\ u_n = \frac{(-1)^n}{4} \ln\left(\frac{u_0^3}{u_1}\right) + \frac{3^n}{4} \ln(u_0 u_1)}$$

□

4. En déduire :

$$\forall n \in \mathbb{N},\ u_n = \exp\left(\frac{3^n}{4} \ln(u_0 u_1) + \frac{(-1)^n}{4} \ln\left(\frac{u_0^3}{u_1}\right)\right)$$

Démonstration.
Soit $n \in \mathbb{N}$. Comme : $t_n = \ln(u_n)$, on a : $u_n = \exp(t_n)$.

$$\boxed{\forall n \in \mathbb{N},\ t_n = \exp\left(\frac{(-1)^n}{4} \ln\left(\frac{u_0^3}{u_1}\right) + \frac{3^n}{4} \ln(u_0 u_1)\right)}$$

□

Exercice 5

On considère la suite (u_n) définie dans l'Exercice 4.

1. Écrire en **Python** la fonction `calculPremiersTermes` qui prend en paramètre un entier `n`, des valeurs `u0`, `u1` et renvoie la liste `L` contenant les `n` premiers termes de la suite $(u_n)_{n\in\mathbb{N}}$.

Démonstration.
On propose la fonction suivante :

```
1  def calculPremiersTermes(n, u0, u1) :
2      L = [u0, u1]
3      for k in range(n-2) :
4          L.append( (L[k]**3)*(L[k+1]**2) )
5      return L
```

- **Début de la fonction**
  L'énoncé commence par préciser la structure de la fonction :
  - × cette fonction se nomme `calculPremiersTermes`,
  - × elle prend en paramètre d'entrée l'entier `n`, le réel `u0` et le réel `u1`,
  - × elle renvoie la liste stockée dans la variable `L`.

```
1  def calculPremiersTermes(n, u0, u1) :
```

```
5      return L
```

  La variable `L`, liste qui contiendra à terme les `n` premières valeurs de (u_n), est initialisée à une liste à deux éléments : le premier contient la valeur de u_0, et le second contient la valeur de u_1.

```
2      L = [u0, u1]
```

- **Structure itérative**
  Les lignes 3 à 4 consistent à calculer les valeurs successives de la suite (u_n) et à les concaténer à la liste `L`. Pour cela, on utilise une structure itérative (boucle for) :

```
3      for k in range(n-2) :
4          L.append( (L[k]**3)*(L[k+1]**2) )
```

  On tire ici partie de la définition récursive (d'ordre 2) de cette suite. Plus précisément, la nouvelle valeur de la suite, que l'on concatènera à la liste `L`, est obtenue à l'aide des deux valeurs précédentes, respectivement stockées dans les éléments `k` et `k + 1` de `L`.
- **Fin de la fonction**
  À l'issue de cette boucle, la variable `L` contient les `n` premiers éléments de la suite (u_n).

Commentaire

- Le programme **Python** consiste à mettre à jour successivement la liste `L` jusqu'à stockage des `n` premières valeurs de la suite (u_n). L'idée est la suivante.
  Si avant le $k^{\text{ème}}$ tour de boucle (avec $\mathtt{k} \in [\![1, \mathtt{n}-2]\!]$) :

  la variable `L` contient les $\mathtt{k}+1$ premières valeurs de la suite (u_n)

  alors, à l'issue de ce tour de boucle :

  la variable `L` contient les $\mathtt{k}+2$ premières valeurs de la suite (u_n)

  Cette propriété est ce qu'on appelle un **invariant de boucle**. Elle permet d'assurer la correction de la fonction implémentée et notamment le fait qu'à l'issue du dernier tour de boucle (le $(\mathtt{n}-2)^{\text{ème}}$) la variable `L` contient les `n` premières valeurs de la suite (u_n).
- On peut présenter la boucle légérement différemment :

```
3       for k in range(n-2) :
4           L = L + [ L[k]**3)*(L[k+1]**2) ]
```

  Cette écriture met en avant l'opération « + » de concaténation de listes.
  Ces deux choix sont pertinents et seront donc évidemment tous les deux acceptés aux concours.

□

2. Écrire un programme permettant de tracer les 100 premiers termes de la suite (u_n) pour les valeurs $u_0 = 0,7$ et $u_1 = 1,2$.

Démonstration.

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  n = 100
5  u0 = 0.7
6  u1 = 1.2
7  U = calculPremiersTermes(n, u0, u1) # calcul des 100 premiers termes de (u_n)
8  absc = np.linspace(0, n-1, n)
9  plt.plot(absc, U) # tracé des 100 premiers termes
10 plt.show
```

Commentaire

- Les explications liées à ce programme sont développées dans la remarque de la question ***3.b)*** de l'Exercice 3.
- Pour effectuer le tracé d'un nuage de points plutôt qu'une courbe lisse, on pourra remplacer la ligne 9 par :

```
9  plt.scatter(absc, U)
```

□

Exercice 6

On appelle $(u_n)_{n\geqslant 0}$ la suite définie par : $\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N},\ u_{n+1} = \sqrt{2}\, u_n + 1 \end{cases}$

1. Démontrer par récurrence : $\forall n \in \mathbb{N},\ u_n \geqslant n$.

Démonstration.
Démontrons par récurrence : $\forall n \in \mathbb{N},\ \mathscr{P}(n)$
où $\mathscr{P}(n)$: $u_n \geqslant n$.

- **Initialisation**
  $u_0 = 1 \geqslant 0$

  D'où $\mathscr{P}(0)$.
- **Hérédité :** soit $n \in \mathbb{N}$.
  Supposons $\mathscr{P}(n)$ et démontrons $\mathscr{P}(n+1)$ (*i.e.* $u_{n+1} \geqslant n+1$).

$$\begin{array}{rcll} u_{n+1} & = & \sqrt{2}\, u_n + 1 & \\ & \geqslant & \sqrt{2}\, n + 1 & \textit{(car par hypothèse de récurrence : } u_n \geqslant n\textit{)} \\ & \geqslant & n+1 & \textit{(car } \sqrt{2} \geqslant 1\textit{)} \end{array}$$

D'où $\mathscr{P}(n+1)$.

Ainsi, par principe de récurrence : $\forall n \in \mathbb{N},\ u_n \geqslant n$. □

2. En déduire la limite de (u_n) quand n tend vers $+\infty$.

Démonstration.
Soit $n \in \mathbb{N}$. D'après la question précédente : $u_n \geqslant n$.
Or : $n \underset{n\to+\infty}{\longrightarrow} +\infty$.

On en déduit : $u_n \underset{n\to+\infty}{\longrightarrow} +\infty$. □

3. Déterminer la formule explicite de u_n.

Démonstration.
La suite (u_n) est arithmético-géométrique. On lui applique la méthode d'étude associée.

- On commence par résoudre l'équation de point fixe associée :

$$x = \sqrt{2}\,x + 1 \ \Leftrightarrow\ x - \sqrt{2}\,x = 1 \ \Leftrightarrow\ x(1-\sqrt{2}) = 1 \ \Leftrightarrow\ x = \frac{1}{1-\sqrt{2}}$$

Ainsi, $\lambda = \dfrac{1}{1-\sqrt{2}} = \dfrac{1+\sqrt{2}}{1-2} = -(1+\sqrt{2})$ est l'unique solution de l'équation.

- Soit $n \in \mathbb{N}$.

On écrit : $\left|\begin{array}{rcrcl} u_{n+1} & = & \sqrt{2} & u_n & + \ 1 \\ \lambda & = & \sqrt{2} & \lambda & + \ 1 \end{array}\right.$

Par soustraction de ces deux inégalités, on obtient : $u_{n+1} - \lambda = \sqrt{2}(u_n - \lambda)$.

- On considère alors la suite (v_n), de terme général : $s_n = u_n - \lambda$.
  D'après l'égalité précédente, cette suite est géométrique, de raison $\sqrt{2}$.
  En effet, $s_{n+1} = u_{n+1} - \lambda = \sqrt{2}(u_n - \lambda) = \sqrt{2}\, s_n$. On en déduit :

$$\forall n \in \mathbb{N},\ s_n = s_0 \times (\sqrt{2})^n$$

- On détermine alors la valeur de s_0 :

$$\begin{aligned} s_0 &= u_0 - \lambda \\ &= 1 - (-(1+\sqrt{2})) \\ &= 2 + \sqrt{2} \end{aligned}$$

- Enfin, on a : $u_n = s_n + \lambda = s_n - (1+\sqrt{2}) = (2+\sqrt{2})(\sqrt{2})^n - (1+\sqrt{2})$.

$$\boxed{\forall n \in \mathbb{N},\ u_n = (2+\sqrt{2})(\sqrt{2})^n - (1+\sqrt{2})}$$

Commentaire

Comme $2+\sqrt{2} > 1$, on a : $(\sqrt{2})^n \underset{n\to+\infty}{\longrightarrow} +\infty$ et on retrouve le résultat de la question ***2.***

□

Séance 4 : suites récurrentes et inégalités des accroissements finis

Exercice 7

Soit f la fonction définie sur $\mathbb{R}$ par $f(x) = \dfrac{\mathrm{e}^x}{\mathrm{e}^{2x}+1}$.

1. a) Démontrer que f est paire sur $\mathbb{R}$.

Démonstration.
Soit $x \in \mathbb{R}$.

$$f(-x) = \frac{\mathrm{e}^{-x}}{\mathrm{e}^{-2x}+1} = \frac{\mathrm{e}^{2x}}{\mathrm{e}^{2x}}\frac{\mathrm{e}^{-x}}{\mathrm{e}^{-2x}+1} = \frac{\mathrm{e}^{x}}{1+\mathrm{e}^{2x}} = f(x)$$

Ainsi, la fonction est paire sur $\mathbb{R}$. □

b) Justifier que f est de classe $\mathcal{C}^1$ sur $\mathbb{R}$ et étudier ses variations.

Démonstration.

- La fonction f est de classe $\mathcal{C}^1$ sur $\mathbb{R}$ car est le quotient de :
  - × la fonction $x \mapsto \mathrm{e}^x$, de classe $\mathcal{C}^1$ sur $\mathbb{R}$,
  - × la fonction $x \mapsto \mathrm{e}^{2x}+1$, de classe $\mathcal{C}^1$ sur $\mathbb{R}$ et **qui ne s'annule pas sur** $\mathbb{R}$.

  (en fait, on peut démontrer par une argumentation similaire que f est de classe $\mathcal{C}^\infty$ sur $\mathbb{R}$)
- Soit $x \in \mathbb{R}$.

$$f'(x) = \frac{\mathrm{e}^x\,(\mathrm{e}^{2x}+1) - \mathrm{e}^x\,(2\,\mathrm{e}^{2x})}{(\mathrm{e}^{2x}+1)^2} = \frac{\mathrm{e}^{3x}+\mathrm{e}^x-2\,\mathrm{e}^{3x}}{(\mathrm{e}^{2x}+1)^2} = \frac{\mathrm{e}^x-\mathrm{e}^{3x}}{(\mathrm{e}^{2x}+1)^2}$$

  Comme $(\mathrm{e}^{2x}+1)^2 > 0$, $f'(x)$ est du signe de $\mathrm{e}^x - \mathrm{e}^{3x}$.
- Si $x > 0$, $3x > x$ et donc $\mathrm{e}^{3x} > \mathrm{e}^x$ par stricte croissance de la fonction exponentielle.
  Dans ce cas, $f'(x) < 0$.
  L'autre cas se déduit par parité de la fonction f.
- On en déduit le tableau de variations de f.

x	$-\infty$		0		$+\infty$
Signe de $f'(x)$		$+$	0	$-$	
Variations de f	0	↗	$\frac{1}{2}$	↘	0

Détaillons les différents éléments de ce tableau :

× $f(0) = \dfrac{\mathrm{e}^0}{\mathrm{e}^0+1} = \dfrac{1}{1+1} = \dfrac{1}{2}$,

× $\dfrac{\mathrm{e}^x}{\mathrm{e}^{2x}+1} = \dfrac{\mathrm{e}^x}{\mathrm{e}^{2x}}\dfrac{1}{1+\frac{1}{\mathrm{e}^{2x}}} = \dfrac{1}{\mathrm{e}^x}\dfrac{1}{1+\frac{1}{\mathrm{e}^{2x}}} \underset{x\to+\infty}{\longrightarrow} 0.$

□

c) Montrer que l'équation $f(x) = x$ admet une unique solution $\ell \in \mathbb{R}^+$.

Démonstration.
Notons $g : x \mapsto f(x) - x$. La fonction g est de classe $\mathcal{C}^1$ sur $\mathbb{R}$ par somme de fonctions $\mathcal{C}^1$ sur $\mathbb{R}$.

Soit $x \in \mathbb{R}$. On procède par disjonction de cas :

- Si $x \leqslant 0$: comme $f(x) > 0$, l'équation $f(x) = x$ n'admet pas de solution.
- Si $x \geqslant 0$: tout d'abord, $g'(x) = f'(x) - 1$.
  Or $f'(x) \leqslant 0$ donc $g'(x) = f'(x) - 1 < 0$ et la fonction g est strictement décroissante.
  La fonction g est :
  × continue sur $[0, +\infty[$,
  × strictement décroissante sur $[0, +\infty[$.
  Elle réalise donc une bijection de $[0, +\infty[$ sur $g([0, +\infty[)$. Or :

$$g([0, +\infty[) =]\lim_{x\to+\infty} g(x),\ g(0)] =]-\infty,\ \frac{1}{2}]$$

En effet :
× $g(0) = f(0) - 0 = \dfrac{1}{2}$.
× comme $f(x) \underset{x\to+\infty}{\longrightarrow} 0$, on obtient : $g(x) = f(x) - x \underset{x\to+\infty}{\longrightarrow} -\infty$.

Enfin, comme $0 \in]-\infty, \frac{1}{2}]$, l'équation $g(x) = 0$ admet une unique solution $\ell \in \mathbb{R}^+$.

On en déduit que l'équation $f(x) = x$ admet une unique solution $\ell \in [0, +\infty[$.

□

d) Justifier : $0 \leqslant \ell \leqslant \dfrac{1}{2}$.
Données numériques : $\mathrm{e}^{1/2} \simeq 1,65$ et $\mathrm{e} \simeq 2,72$ au centième près.

Démonstration.
Tout d'abord, remarquons :
× $g(0) = f(0) - 0 = \dfrac{1}{2} \geqslant 0$,
× $g(\ell) = f(\ell) - \ell = 0$,
× $g\left(\dfrac{1}{2}\right) = f\left(\dfrac{1}{2}\right) - \dfrac{1}{2} = \dfrac{\mathrm{e}^{\frac{1}{2}}}{\mathrm{e}^1 + 1} - \dfrac{1}{2} = \dfrac{2\,\mathrm{e}^{\frac{1}{2}} - \mathrm{e} - 1}{2\,(\mathrm{e}+1)} \leqslant 0.$

En effet :
$2\,\mathrm{e}^{\frac{1}{2}} - \mathrm{e} - 1 \simeq 2 \times 1,65 - 2,72 - 1 = 3,3 - 3,72 = -0,42 \leqslant 0$
(les valeurs étant données au centième près, l'approximation obtenue est exacte au moins au dixième près)

On obtient donc : $g(0) \geqslant g(\ell) \geqslant g\left(\dfrac{1}{2}\right)$.

Ces trois éléments sont dans l'ensemble $]-\infty, \frac{1}{2}]$.

En appliquant $g^{-1} :]-\infty, \frac{1}{2}] \to \mathbb{R}^+$ de part et d'autre de l'inégalité, on obtient : $0 \leqslant \ell \leqslant \dfrac{1}{2}$.

Commentaire

On pouvait aussi tout simplement remarquer que, d'après la question précédente :
× $\ell \in \mathbb{R}^+$
× $\ell = f(\ell)$
Ainsi : $\ell = f(\ell) \in f(\mathbb{R}^+)$. Or $f(\mathbb{R}^+) =]0, \frac{1}{2}]$. Ainsi $\ell \in]0, \frac{1}{2}]$. □

e) Montrer : $\forall x \geqslant 0,\ |f'(x)| \leqslant f(x)$.
En déduire : $\forall x \geqslant 0,\ |f'(x)| \leqslant \dfrac{1}{2}$.

Démonstration.
Soit $x \geqslant 0$.

- D'après la question ***1.b)*** :

$$f'(x) = \frac{\mathrm{e}^x - \mathrm{e}^{3x}}{1+\mathrm{e}^{2x}} \leqslant 0$$

Donc : $|f'(x)| = -f'(x) = \dfrac{\mathrm{e}^{3x} - \mathrm{e}^x}{1+\mathrm{e}^{2x}}$. Et ainsi :

$$|f'(x)| - f(x) = \frac{\mathrm{e}^{3x} - \mathrm{e}^x}{(\mathrm{e}^{2x}+1)^2} - \frac{\mathrm{e}^x}{\mathrm{e}^{2x}+1} = \frac{\mathrm{e}^{3x} - \mathrm{e}^x - \mathrm{e}^x \times (\mathrm{e}^{2x}+1)}{(\mathrm{e}^{2x}+1)^2}$$
$$= \frac{\cancel{\mathrm{e}^{3x}} - \mathrm{e}^x - \cancel{\mathrm{e}^{3x}} - \mathrm{e}^x}{(\mathrm{e}^{2x}+1)^2} = \frac{-2\ \mathrm{e}^x}{(\mathrm{e}^{2x}+1)^2} \leqslant 0$$

Ainsi : $\forall x \geqslant 0,\ |f'(x)| \leqslant f(x)$.

- Or, l'étude de la fonction f (question ***1.b)***) démontre qu'elle atteint son maximum en 0.

On en déduit : $\forall x \geqslant 0,\ |f'(x)| \leqslant f(x) \leqslant f(0) = \dfrac{1}{2}$.

□

f) Vérifier que $f(\left[0, \frac{1}{2}\right]) \subset \left[0, \frac{1}{2}\right]$.

Démonstration.
La fonction f est continue et décroissante sur $\left[0, \frac{1}{2}\right]$. On en déduit :

$$f\left(\left[0, \frac{1}{2}\right]\right) = \left[f\left(\frac{1}{2}\right),\ f(0)\right] = \left[\frac{\mathrm{e}^{\frac{1}{2}}}{\mathrm{e}+1},\ \frac{1}{2}\right] \subset \left[0, \frac{1}{2}\right]$$

L'intervalle $\left[0, \frac{1}{2}\right]$ est stable par f.

□

2. On définit la suite $(u_n)_{n\in\mathbb{N}}$ par :

$$u_0 = 0 \qquad \text{et} \qquad \forall n \in \mathbb{N},\quad u_{n+1} = f(u_n)$$

a) Montrer que, pour tout $n \in \mathbb{N}$, $\quad u_n \in \left[0, \frac{1}{2}\right]$.

Démonstration.
Démontrons par récurrence : $\forall n \in \mathbb{N},\ \mathscr{P}(n)$
où $\mathscr{P}(n) : u_n \in \left[0, \frac{1}{2}\right]$.

▸ **Initialisation** :
$u_0 = 0 \in \left[0, \frac{1}{2}\right]$.
D'où $\mathscr{P}(0)$.

▸ **Hérédité** :
Soit $n \in \mathbb{N}$. Supposons $\mathscr{P}(n)$ et démontrons $\mathscr{P}(n+1)$
(*i.e.* $u_{n+1} \in \left[0, \frac{1}{2}\right]$).
Par définition, $u_{n+1} = f(u_n)$. Or :
× par hypothèse de récurrence, on sait : $u_n \in \left[0, \frac{1}{2}\right]$.
× l'intervalle $\left[0, \frac{1}{2}\right]$ est stable par f.
On obtient donc : $u_{n+1} = f(u_n) \in \left[0, \frac{1}{2}\right]$.
D'où $\mathscr{P}(n+1)$.

Par principe de récurrence : $\forall n \in \mathbb{N},\ u_n \in \left[0, \frac{1}{2}\right]$. □

b) Montrer que, pour tout $n \in \mathbb{N}$:

$$|u_{n+1} - \ell| \leqslant \frac{1}{2} |u_n - \ell| \qquad \text{puis que} \qquad |u_n - \ell| \leqslant \frac{1}{2^{n+1}}$$

Démonstration.

- D'après les questions précédentes :
  × f est dérivable sur $\left[0, \frac{1}{2}\right]$,
  × $\forall x \in \left[0, \frac{1}{2}\right],\ |f'(x)| \leqslant \frac{1}{2}$.
  On en déduit, par l'inégalité des accroissements finis :

$$\forall (x, y) \in \left[0, \frac{1}{2}\right]^2,\ |f(y) - f(x)| \leqslant \frac{1}{2} |y - x|$$

  Soit $n \in \mathbb{N}$. En appliquant cette inégalité à $y = u_n \in \left[0, \frac{1}{2}\right]$ et $x = \ell \in \left[0, \frac{1}{2}\right]$, on obtient :

$$|f(u_n) - f(\ell)| \leqslant \frac{1}{2} |u_n - \ell|$$

  Et ainsi : $|u_{n+1} - \ell| \leqslant \frac{1}{2} |u_n - \ell|$.

- Démontrons par récurrence : $\forall n \in \mathbb{N},\ \mathscr{P}(n)$
  où $\mathscr{P}(n) : |u_n - \ell| \leqslant \frac{1}{2^{n+1}}$.
  ▸ **Initialisation** :
  $|u_0 - \ell| \leqslant \frac{1}{2}$ car u_0 et ℓ sont des éléments de $\left[0, \frac{1}{2}\right]$.
  D'où $\mathscr{P}(0)$.
  ▸ **Hérédité** :
  Soit $n \in \mathbb{N}$. Supposons $\mathscr{P}(n)$ et démontrons $\mathscr{P}(n+1)$
  (*i.e.* $|u_{n+1} - \ell| \leqslant \frac{1}{2^{n+2}}$).
  D'après le résultat précédent : $|u_{n+1} - \ell| \leqslant \frac{1}{2} |u_n - \ell|$.
  Or, par hypothèse de récurrence : $|u_n - \ell| \leqslant \frac{1}{2^{n+1}}$.
  En combinant ces deux résultats, on obtient :

$$|u_{n+1} - \ell| \leqslant \frac{1}{2} |u_n - \ell| \leqslant \frac{1}{2} \frac{1}{2^{n+1}} = \frac{1}{2^{n+2}}$$

  D'où $\mathscr{P}(n+1)$.

Par principe de récurrence : $\forall n \in \mathbb{N},\ |u_n - \ell| \leqslant \frac{1}{2^{n+1}}$. □

c) En déduire que la suite (u_n) converge vers ℓ.

Démonstration.
Soit $n \in \mathbb{N}$.

- D'après la question précédente : $|u_n - \ell| \leqslant \dfrac{1}{2^{n+1}}$. Autrement dit :

$$-\frac{1}{2^{n+1}} \leqslant u_n - \ell \leqslant \frac{1}{2^{n+1}}$$

- Or :
  - $\dfrac{1}{2^{n+1}} \underset{n\to+\infty}{\longrightarrow} 0$ car $2^{n+1} \underset{n\to+\infty}{\longrightarrow} +\infty$,
  - $-\dfrac{1}{2^{n+1}} \underset{n\to+\infty}{\longrightarrow} 0$.

Donc, d'après le théorème d'encadrement, la suite $(u_n - \ell)$ est convergente de limite 0.

Ainsi, la suite (u_n) est convergente et de limite ℓ. □

3. Informatique

a) Écrire une fonction **Python** `f` qui prend en entrée un réel `x` et qui calcule $f(\texttt{x})$.

Démonstration.

```
1  import numpy as np
2
3  def f(x) :
4      y = np.exp(x) / (np.exp(2*x) + 1)
5      return y
```

□

b) En utilisant la fonction `f` précédente, écrire une fonction `SuiteU` qui prend en entrée un entier positif n et qui calcule u_n.

Démonstration.
On propose la fonction suivante :

```
1  def SuiteU(n) :
2      u = 0
3      for i in range(n) :
4          u = f(u)
5      return u
```

Détaillons les éléments de cette fonction.

- **Début de la fonction**
  L'énoncé commence par préciser la structure de la fonction :
  - cette fonction se nomme `SuiteU`,
  - elle prend en paramètre d'entrée l'entier `n`,
  - elle renvoie la valeur stockée dans la variable `u`.

```
1  def SuiteU(n) :
```

```
5      return u
```

La variable `u`, qui contiendra les valeurs successives de la suite (u_n) est initialisée à 0 : la valeur de u_0.

```
2        u = 0
```

- **Structure itérative**
  Les lignes 3 à 4 consistent à calculer les valeurs successives de la suite (u_n).
  Pour cela, on utilise une structure itérative (boucle for) :

```
3        for i in range(n) :
4            u = f(u)
```

  On tire ici partie de la définition récursive (d'ordre 1) de cette suite. La nouvelle valeur de la suite, que l'on stockera dans la variable `u`, est obtenue à l'aide de la valeur précédente qui est celle alors stockée dans `u`.
- **Fin de la fonction**
  À l'issue de cette boucle, la variable `u` contient la quantité $u_{\mathtt{n}}$ où `n` est le paramètre entré lors de l'appel de la fonction.

Commentaire

- Afin de permettre une bonne compréhension des mécanismes en jeu, on a détaillé la réponse à cette question. Cependant, proposer un programme **Python** correct démontre la bonne compréhension de ces mécanismes et permet certainement d'obtenir la majorité des points alloués à cette question.
- Le programme **Python** consiste à mettre à jour successivement la variable `u` jusqu'à obtention de la valeur que l'on souhaite calculer. L'idée est la suivante.
  Si avant le $i^{\text{ème}}$ tour de boucle (avec $\mathtt{i} \in [\![1, \mathtt{n}]\!]$) :

  la variable `u` contient la valeur u_{i-1}

  alors, à l'issue de ce tour de boucle :

  la variable `u` contient la valeur u_i

  Cette propriété est ce qu'on appelle un **invariant de boucle**. Elle permet d'assurer la correction de la fonction implémentée et notamment le fait qu'à l'issue du dernier tour de boucle la variable `u` contient $u_{\mathtt{n}}$.
- Dans ce programme, on réalise un appel à la fonction `f`. On obtient ainsi le programme générique permettant de calculer $u_{\mathtt{n}}$ pour toute suite (u_n) récurrente d'ordre 1 (c'est à dire qui vérifie : $\forall n \in \mathbb{N}$, $u_{n+1} = f(u_n)$). Il est vivement conseillé d'apprendre ce programme.
- Aux concours, il est toujours possible de coder la fonction `f` à part. Cependant, les questions **Python** sont généralement plus dirigées (on demande généralement de remplir des programmes à trous) ce qui laisse au candidat moins de possibilité de prendre des initiatives. Il faut aussi savoir coder la fonction `SuiteU` lorsque l'on ne demande pas au préalable de coder la fonction `f`. Il suffit alors de remplacer `f(u)` par son expression.

```
1  def SuiteU(n) :
2      u = 0
3      for i in range(n) :
4          u = np.exp(u) / (np.exp(2*u) + 1)
5      return u
```

□

c) En utilisant la fonction `SuiteU` précédente, comment peut-on obtenir à l'aide de **Python** une valeur approchée de ℓ à 10^{-4} près ?

Démonstration.
D'après la question ***2.b.***, pour tout $n \in \mathbb{N}$ on a : $|u_n - \ell| \leqslant \dfrac{1}{2^{n+1}}$.
S'il existe $n \in \mathbb{N}$ tel que : $\dfrac{1}{2^{n+1}} \leqslant 10^{-6}$, on obtiendra par transitivité : $|u_n - \ell| \leqslant 10^{-6}$. Or :

$$\begin{aligned}
\frac{1}{2^{n+1}} \leqslant 10^{-6} &\Leftrightarrow 2^{n+1} \geqslant 10^6 && \textit{(par décroissance de la fonction inverse sur }]0,+\infty[\textit{)}\\
&\Leftrightarrow \ln\left(2^{n+1}\right) \geqslant \ln\left(10^6\right) && \textit{(par croissance de la fonction } \ln \textit{ sur }]0,+\infty[\textit{)}\\
&\Leftrightarrow (n+1)\ln(2) \geqslant 6\ln(10)\\
&\Leftrightarrow n+1 \geqslant \frac{6\ln(10)}{\ln(2)} && \textit{(car } \ln(2) > 0\textit{)}\\
&\Leftrightarrow n \geqslant \frac{6\ln(10)}{\ln(2)} - 1
\end{aligned}$$

Pour $N = \left\lceil \dfrac{6\ln(10)}{\ln(2)} - 1 \right\rceil$ (et les entiers plus grands) on est donc assuré que :

$$|u_N - \ell| \leqslant 10^{-6}$$

ce qui signifie que u_N est une approximation de ℓ à 10^{-6} près.
Il suffit alors d'appeler la fonction `SuiteU` avec pour paramètre `N`.

```
1 N = int(np.ceil(6 * np.log(10) / np.log(2) - 1))
2 u = SuiteU(N)
```

Notons l'utilisation de la fonction `int` qui permet de considérer la variable `N` non pas comme un réel (type `float`) mais comme un entier (type `int`). Sans cela, on ne peut pas appliquer la fonction `SuiteU` à cette variable `N`. □

Exercice 8
Dans tout cet exercice, f désigne la fonction définie sur $]0,+\infty[$ par :

$$\forall x \in]0,+\infty[,\ f(x) = x - \ln(x)$$

Partie I : Étude de la fonction f

1. Dresser le tableau de variations de f en précisant ses limites en 0 et en $+\infty$.

Démonstration.

- La fonction f est dérivable sur $]0,+\infty[$ en tant que somme de fonctions dérivables sur $]0,+\infty[$.
- Soit $x \in]0,+\infty[$.

$$f'(x) \ = \ 1 - \frac{1}{x} \ = \ \frac{x-1}{x}$$

Alors, comme $x > 0$:

$$f'(x) \geqslant 0 \ \Leftrightarrow \ x - 1 \geqslant 0 \ \Leftrightarrow \ x \geqslant 1$$

On obtient le tableau de variations suivant :

x	0		1		$+\infty$
Signe de $f'(x)$	‖	$-$	0	$+$	
Variations de f	$+\infty$ ‖	↘	1	↗	$+\infty$

- Détaillons les éléments de ce tableau.
  - Tout d'abord : $f(1) = 1 - \ln(1) = 1$.
  - Ensuite : $\lim\limits_{x\to 0} \ln(x) = -\infty$.

$$\boxed{\text{Donc : } \lim_{x\to 0} f(x) = +\infty.}$$

  - Enfin, soit $x \in]0,+\infty[$:

$$f(x) \ = \ x - \ln(x) \ = \ x\Big(1 - \frac{\ln(x)}{x}\Big)$$

De plus, par croissances comparées : $\lim\limits_{x\to+\infty} \dfrac{\ln(x)}{x} = 0$.

$$\boxed{\text{On en déduit : } \lim_{x\to+\infty} f(x) = +\infty.}$$

□

2. Montrer que l'équation $f(x) = 2$, d'inconnue $x \in]0,+\infty[$, admet exactement deux solutions, que l'on note a et b, telles que $0 < a < 1 < b$.

Démonstration.

- La fonction f est :
  - × continue sur $]0,1[$ (car dérivable sur $]0,1[$),
  - × strictement décroissante sur $]0,1[$.

  Ainsi f réalise une bijection de $]0,1[$ dans $f(]0,1[)$.

$$f(]0,1[) \ = \Big]\lim_{x\to 1^-} f(x),\ \lim_{x\to 0} f(x)\Big[=]1,+\infty[$$

Or $2 \in]1,+\infty[$.
Donc l'équation $f(x) = 2$ admet une unique solution sur $]0,1[$, notée a.

- La fonction f est :
  - × continue sur $]1,+\infty[$ (car dérivable sur $]1,+\infty[$),
  - × strictement croissante sur $]1,+\infty[$.

  Ainsi f réalise une bijection de $]1,+\infty[$ dans $f(]1,+\infty[)$.

$$f(]1,+\infty[) = \left]\lim_{x\to 1^-} f(x), \lim_{x\to+\infty} f(x)\right[=]1,+\infty[$$

  Or $2 \in]1,+\infty[$.
  Donc l'équation $f(x) = 2$ admet une unique solution sur $]1,+\infty[$, notée b.

Finalement, l'équation $f(x) = 2$ admet exactement 2 solutions sur $]0,+\infty[$ notées a et b telles que $0 < a < 1 < b$.

Commentaire

- Il est important dans cette question d'avoir parfaitement en tête toutes les hypothèses du théorème de la bijection. En particulier, la fonction f doit être **strictement monotone** sur l'intervalle considéré.
- On ne pouvait donc pas appliquer le théorème de la bijection directement sur l'intervalle $]0,+\infty[$, mais il fallait découper cet intervalle en plusieurs sous-intervalles sur lesquels f est strictement monotone (ici $]0,1[$ et $]1,+\infty[$).

□

3. Montrer : $b \in [2,4]$. On donne : $\ln(2) \simeq 0,7$.

Démonstration.

- Remarquons tout d'abord :
  - × $f(2) = 2 - \ln(2) \leqslant 2$,
  - × $f(4) = 4 - \ln(4) = 4 - \ln(2^2) = 4 - 2\ln(2) = 2(2 - \ln(2))$.
    De plus, $\ln(2) \simeq 0,7$, donc : $2 - \ln(2) \simeq 1,3$ et ainsi : $f(4) = 2(2 - \ln(2)) \simeq 2,6 \geqslant 2$.
  - × $f(b) = 2$.

  On a donc : $f(2) \leqslant f(b) \leqslant f(4)$.
- Notons g la réciproque de f sur $]1,+\infty[$. D'après le théorème de la bijection, $g :]1,+\infty[\to]1,+\infty[$ est strictement croissante sur $]1,+\infty[$. En appliquant g de part et d'autre de l'inégalité précédente :

$$\begin{array}{ccccc} g\big(f(2)\big) & \leqslant & g\big(f(b)\big) & \leqslant & g\big(f(4)\big) \\ \| & & \| & & \| \\ 2 & \leqslant & b & \leqslant & 4 \end{array}$$

On a bien démontré : $b \in [2,4]$.

Commentaire

L'indication de l'énoncé $\ln(2) \simeq 0,7$ ne permet pas de savoir s'il s'agit d'une sur ou d'une sous-approximation. Un encadrement, tel que $0,6 \leqslant \ln(2) \leqslant 0,8$, permettrait de résoudre ce problème.

□

Partie II : Étude d'une suite

On pose : $u_0 = 4$ et $\forall n \in \mathbb{N},\ u_{n+1} = \ln(u_n) + 2$.

4. Montrer que la suite $(u_n)_{n\in\mathbb{N}}$ est bien définie et que l'on a : $\forall n \in \mathbb{N},\ u_n \in [b, +\infty[$.

Démonstration.

Démontrons par récurrence : $\forall n \in \mathbb{N},\ \mathscr{P}(n)$ où $\mathscr{P}(n) : \begin{cases} u_n \text{ est bien défini} \\ u_n \in [b, +\infty[\end{cases}$

- **Initialisation** :
  $u_0 = 4$. Or, d'après la question ***3.***, $b \leqslant 4$. Donc : $u_0 \in [b, +\infty[$.
  D'où $\mathscr{P}(0)$.
- **Hérédité** : Soit $n \in \mathbb{N}$.
  Supposons $\mathscr{P}(n)$ et démontrons $\mathscr{P}(n+1)$ (*i.e.* $\begin{cases} u_{n+1} \text{ est bien défini} \\ u_{n+1} \in [b, +\infty[\end{cases}$)

  Par hypothèse de récurrence, u_n est bien défini et $u_n \in [b, +\infty[$.

  - Comme $u_n \geqslant b \geqslant 2$, on a en particulier $u_n > 0$.
    Donc $\ln(u_n)$ est bien définie. D'où u_{n+1} est bien défini.
  - Comme $u_n \geqslant b$

    alors $\ln(u_n) \geqslant \ln(b)$ *(par croissance de la fonction* ln *sur* $]0, +\infty[$*)*

    et $\ln(u_n) + 2 \geqslant \ln(b) + 2$

    $\ln(u_n) + 2 = u_{n+1}$

    Enfin, par définition de b : $f(b) = 2$, c'est-à-dire $b - \ln(b) = 2$. Ainsi : $\ln(b) = b - 2$.
    On obtient alors :

    $$u_{n+1} \geqslant \ln(b) + 2 = (b - 2) + 2$$

  D'où $\mathscr{P}(n+1)$.

Par principe de récurrence, on obtient que (u_n) est bien définie et : $\forall n \in \mathbb{N},\ u_n \in [b, +\infty[$.

Commentaire

- Cette question est un classique des suites récurrentes.
  Elle se traite généralement par récurrence.
- Il faut ici faire attention à bien énoncer l'hypothèse de récurrence.
  Pour montrer que « **la suite** (u_n) est bien définie », on démontre en réalité :

  $$\forall n \in \mathbb{N},\ \mathscr{P}(n) \quad \text{où} \quad \mathscr{P}(n) : \textbf{le réel } u_n \text{ est bien défini}$$

□

5. Déterminer la monotonie de la suite $(u_n)_{n\in\mathbb{N}}$. En déduire qu'elle converge et préciser sa limite.

Démonstration.

- Soit $n \in \mathbb{N}$.

$$u_{n+1} - u_n = \ln(u_n) + 2 - u_n = 2 - (u_n - \ln(u_n)) = f(b) - f(u_n)$$

Or, d'après la question précédente : $u_n \geqslant b$.
De plus, par croissance de la fonction f sur $[b, +\infty[$: $f(u_n) \geqslant f(b)$.
D'où : $u_{n+1} - u_n = f(b) - f(u_n) \leqslant 0$.

Ainsi, la suite (u_n) est décroissante.

Commentaire

On pouvait aussi démontrer la décroissance de la suite (u_n) par récurrence.
Démontrons par récurrence : $\forall n \in \mathbb{N}$, $\mathscr{P}(n)$ où $\mathscr{P}(n) : u_{n+1} \leqslant u_n$.

- **Initialisation** :
  $u_1 = \ln(u_0) + 2 = \ln(4) + 2 = 2\ln(2) + 2 \simeq 2 \times 0,7 + 2 \simeq 3,4$.
  Donc $u_1 \leqslant u_0$.
  D'où $\mathscr{P}(0)$.
- **Hérédité** : Soit $n \in \mathbb{N}$.
  Supposons $\mathscr{P}(n)$ et démontrons $\mathscr{P}(n+1)$ (*i.e.* $u_{n+2} \leqslant u_{n+1}$).

Tout d'abord	u_{n+1}	$\leqslant$	u_n	*(par hypothèse de récurrence)*
donc	$\ln(u_{n+1})$	$\leqslant$	$\ln(u_n)$	*(par croissance de la fonction* ln *sur* $]0,+\infty[$*)*
et	$\ln(u_{n+1}) + 2$	$\leqslant$	$\ln(u_n) + 2$	
	$\shortparallel$		$\shortparallel$	
	u_{n+2}		u_{n+1}	

  D'où $\mathscr{P}(n+1)$.

Par principe de récurrence, on en déduit : $\forall n \in \mathbb{N}$, $u_{n+1} \leqslant u_n$.

- La suite (u_n) est donc :
  × décroissante,
  × minorée par b (car : $\forall n \in \mathbb{N}$, $u_n \in [b,+\infty[$).

  On en déduit que la suite (u_n) converge. On note ℓ sa limite.

- - Tout d'abord : $\forall n \in \mathbb{N}$, $u_n \geqslant b$.
    Par passage à limite, on en déduit : $\ell \geqslant b$.
  - Ensuite : $\forall n \in \mathbb{N}$, $u_{n+1} = \ln(u_n) + 2$.
    Donc, par continuité de ln sur $]0,+\infty[$: $\ell = \ln(\ell) + 2$. Or :

$$\ell = \ln(\ell) + 2 \;\Leftrightarrow\; \ell - \ln(\ell) = 2 \;\Leftrightarrow\; f(\ell) = 2$$

  Or, d'après la question ***2.***, b est l'unique solution de l'équation $f(x) = 2$ sur $]1,+\infty[$.

  Donc $\ell = b$.

□

6. a) Montrer : $\forall n \in \mathbb{N}$, $u_{n+1} - b \leqslant \dfrac{1}{2}(u_n - b)$.

Démonstration.
On note h la fonction définie par $h : x \mapsto \ln(x) + 2$.

- La fonction h est dérivable sur $[b,+\infty[$ en tant que somme de fonctions dérivables sur $[b,+\infty[$.
  Soit $x \in [b,+\infty[$. Alors $h'(x) = \dfrac{1}{x} \geqslant 0$. Ainsi : $\forall x \in [b,+\infty[,\ |h'(x)| = h(x) = \dfrac{1}{x}$.
  Or, d'après la question ***3.***, $b \geqslant 2$. Donc, pour tout $x \in [b,+\infty[$: $x \geqslant b \geqslant 2$.
  Par décroissance de la fonction inverse sur $]0,+\infty[$, on en déduit : $\dfrac{1}{x} \leqslant \dfrac{1}{2}$.
  Ainsi :

$$\forall x \in [b,+\infty[,\ h'(x) \leqslant \frac{1}{2}$$

- On sait alors :
  × h est dérivable sur $[b, +\infty[$,
  × $\forall x \in [b, +\infty[,\ |h'(x)| = h'(x) \leqslant \dfrac{1}{2}$.
  On en déduit, par l'inégalité des accroissements finis :

$$\forall (x, y) \in [b, +\infty[^2,\ |\ h(y) - h(x)\ | \leqslant \frac{1}{2}\ |y - x|$$

  Soit $n \in \mathbb{N}$. En appliquant cette inégalité à $y = u_n \in [b, +\infty[$ et $x = b \in [b, +\infty[$, on obtient :

$$h(u_n) - h(b) \ = \ |\ h(u_n) - h(b)\ | \leqslant \frac{1}{2}\ |u_n - b| \ = \ \frac{1}{2}\ (u_n - b)$$

  Or :
  × $h(u_n) = \ln(u_n) + 2 = u_{n+1}$
  × $h(b) = \ln(b) + 2 = (b - 2) + 2 = b$, car b est solution de l'équation $f(x) = 2$.

  On en déduit : $\forall n \in \mathbb{N},\ u_{n+1} - b \leqslant \dfrac{1}{2}(u_n - b)$.

□

b) En déduire : $\forall n \in \mathbb{N},\ 0 \leqslant u_n - b \leqslant \dfrac{1}{2^{n-1}}$.

Démonstration.

- Soit $n \in \mathbb{N}$. D'après la question ***4.*** : $u_n \geqslant b$.

  Donc : $u_n - b \geqslant 0$.

- Démontrons par récurrence : $\forall n \in \mathbb{N},\ \mathscr{P}(n)$ où $\mathscr{P}(n) : u_n - b \leqslant \dfrac{1}{2^{n-1}}$.
  ▸ **Initialisation** :
  D'une part : $u_0 - b = 4 - b$.
  D'autre part : $\dfrac{1}{2^{0-1}} = \dfrac{1}{2^{-1}} = 2$.
  Ainsi : $u_0 - b \ = \ 4 - b$
  $\leqslant \ 4 - 2$ *(car $b \geqslant 2$ d'après la question **3**)*
  $= \ 2 \ = \ \dfrac{1}{2^{0-1}}$
  D'où $\mathscr{P}(0)$.
  ▸ **Hérédité** : Soit $n \in \mathbb{N}$.
  Supposons $\mathscr{P}(n)$ et démontrons $\mathscr{P}(n+1)$ (*i.e.* $u_{n+1} - b \leqslant \dfrac{1}{2^n}$).
  D'après la question précédente : $u_{n+1} - b \leqslant \dfrac{1}{2}(u_n - b)$.
  Or, par hypothèse de récurrence : $u_n - b \leqslant \dfrac{1}{2^{n-1}}$.
  En combinant ces deux résultats, on obtient :

$$u_{n+1} - b \ \leqslant \ \frac{1}{2}(u_n - b) \ \leqslant \ \frac{1}{2}\,\frac{1}{2^{n-1}} = \frac{1}{2^n}$$

  D'où $\mathscr{P}(n+1)$.

  Par principe de récurrence : $\forall n \in \mathbb{N},\ u_n - b \leqslant \dfrac{1}{2^{n-1}}$.

□

7. a) Écrire une fonction **Python** d'en-tête `def suite(n) :` qui, prenant en argument un entier n de $\mathbb{N}$, renvoie la valeur de u_n. On admettra que la bibliothèque `numpy` est déjà chargée.

Démonstration.
On propose la fonction suivante :

```
1  def suite(n) :
2      u = 4
3      for i in range(n) :
4          u = np.log(u) + 2
5      return u
```

Détaillons les éléments de cette fonction.

- **Début de la fonction**
  L'énoncé commence par préciser la structure de la fonction :
  × cette fonction se nomme `suite`,
  × elle prend en paramètre d'entrée l'entier `n`,
  × elle renvoie la valeur stockée dans la variable `u`.

```
1  def suite(n) :
```

```
5      return u
```

  La variable `u`, qui contiendra les valeurs successives de la suite (u_n) est initialisée à 4 : la valeur de u_0.

```
2      u = 4
```

- **Structure itérative**
  Les lignes 3 à 4 consistent à calculer les valeurs successives de la suite (u_n).
  Pour cela, on utilise une structure itérative (boucle for) :

```
3      for i in range(n) :
4          u = np.log(u) + 2
```

  On tire ici partie de la définition récursive (d'ordre 1) de cette suite. La nouvelle valeur de la suite, que l'on stockera dans la variable `u`, est obtenue à l'aide de la valeur précédente qui est celle alors stockée dans `u`.
- **Fin de la fonction**
  À l'issue de cette boucle, la variable `u` contient la quantité $u_{\texttt{n}}$ où `n` est le paramètre entré lors de l'appel de la fonction.

Commentaire

- Le programme **Python** consiste à mettre à jour successivement la variable `u` jusqu'à obtention de la valeur que l'on souhaite calculer. L'idée est la suivante.
  Si avant le $i^{\text{ème}}$ tour de boucle (avec $\texttt{i} \in [\![1, \texttt{n}]\!]$) :

  la variable `u` contient la valeur u_{i-1}

  alors, à l'issue de ce tour de boucle :

  la variable `u` contient la valeur u_i

  Cette propriété est ce qu'on appelle un **invariant de boucle**. Elle permet d'assurer la correction de la fonction implémentée et notamment le fait qu'à l'issue du dernier tour de boucle la variable `u` contient $u_{\texttt{n}}$.

b) Recopier et compléter la ligne 3 de la fonction **Python** suivante afin que, prenant en argument un réel `epsilon` strictement positif, elle renvoie une valeur approchée de b à `epsilon` près.

```
1  def valeur_approchee(epsilon) :
2      n = 0
3      while ..... :
4          n = n + 1
5      return suite(n)
```

Démonstration.

- D'après la question ***6.b)*** :

$$\forall n \in \mathbb{N},\ 0 \leqslant u_n - b \leqslant \frac{1}{2^{n-1}}$$

S'il existe $N \in \mathbb{N}$ tel que $\dfrac{1}{2^{n-1}} \leqslant \varepsilon$, on obtiendra par transitivité :

$$0 \leqslant u_N - b \leqslant \varepsilon$$

Donc u_N est une valeur approchée de b à ε près.

- On complète alors le programme **Python** de la façon suivante :

```
3      while 1 / 2**(n-1) > epsilon
```

On propose le programme suivant :

```
1  def valeur_approchee(epsilon) :
2      n = 0
3      while 1 / 2**(n-1) > epsilon :
4          n = n + 1
5      return suite(n)
```

Détaillons les éléments de ce script.

- **Début du programme**

Commençons par préciser la structure de la fonction :

× cette fonction se nomme `valeur_approchee`,

× elle prend comme paramètre d'entrée le réel `epsilon`,

× elle renvoie le résultat de la commande `suite(n)`.

```
1  def valeur_approchee(epsilon)
```

```
5      return suite(n)
```

On initialise ensuite la variable `n` à 0.

```
2  n = 0
```

- **Structure itérative**

Les lignes 3 à 4 consistent à déterminer un entier n tel que : $|u_n - b| \leqslant \varepsilon$. Pour ce faire, on se sert de la condition suffisante exposée ci-dessus à savoir : $\frac{1}{2^{n-1}} \leqslant \varepsilon$ (si cette condition est vérifiée alors il en est de même de la précédente).

Le programme consiste donc à incrémenter la variable `n` de 1 jusqu'à ce que : $\frac{1}{2^{n-1}} \leqslant \varepsilon$. Autrement dit, on doit incrémenter la variable `n` de 1 tant que : $\frac{1}{2^{n-1}} > \varepsilon$.

Pour cela on met en place une boucle while :

```
3        while 1 / 2**(n-1) > epsilon :
```

Puis on met à jour la variable `n`.

```
4            n = n + 1
```

- **Fin du programme**
  À l'issue de cette boucle, la variable `n` contient un entier n tel que : $\frac{1}{2^{n-1}} \leqslant \varepsilon$, ce qui assure : $|u_n - b| \leqslant \varepsilon$. Il n'y a plus qu'à calculer $u_{\texttt{n}}$ à l'aide de la fonction `suite` définie précédemment.

```
6        return suite(n)
```

□

Séance 5 : suites implicites

Exercice 9

Pour tout entier n positif, on définit sur $[0,+\infty[$ la fonction f_n par :

$$\forall x \in [0,+\infty[,\ f_n(x) = \mathrm{e}^x + nx^2 - 3$$

1. a) Montrer que f_n est continue et dérivable sur son ensemble de définition.
Dresser son tableau de variations.

Démonstration.
La fonction f_n est dérivable sur $\mathbb{R}$ par somme de fonctions dérivables sur $\mathbb{R}$.

$$\forall x \geqslant 0,\ f_n'(x) = \mathrm{e}^x + 2nx > 0$$

x	0 $\qquad$ $+\infty$
Signe de $f_n'(x)$	$+$
Variations de f_n	-2 ↗ $+\infty$

□

b) Donner l'équation de la tangente de f_n en 1.

Démonstration.
La tangente de f_n en 1 a pour équation $y = f_n(1) + f_n'(1)\,(x-1)$.
Or : $f_n(1) = \mathrm{e}^1 + n - 3$ et $f_n'(1) = \mathrm{e}^1 + 2n$. Ainsi :

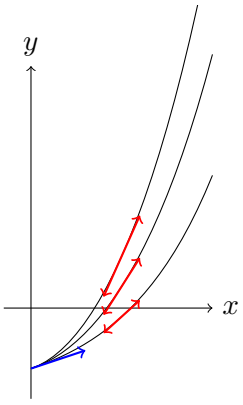
$$f_n(1) + f_n'(1)\,(x-1) = \mathrm{e}^1 + n - 3 + (\mathrm{e}^1 + 2n)\,(x-1) = -n - 3 + (\mathrm{e}^1 + 2n)\,x$$

La tangente de f_n en 1 a pour équation $y = -(n+3) + (\mathrm{e}^1 + 2n)\,x$.

□

c) Tracer dans un même repère les courbes de f_0, f_1 et f_2.

Démonstration.
Il était important ici de tracer les tangentes en 1 *de* f_0, f_1, f_2.
Afin d'avoir une idée plus précise du tracé, déterminons l'équation de la tangente de f_n en 0.
On a : $f_n(0) = -2$ et $f_n'(0) = 1$.
Ainsi, l'équation de la tangente de f_n en 0 est $y = -2 + x$.



□

d) Montrer que l'équation $f_n(x) = 0$ a exactement une solution positive, notée u_n.

Démonstration.
La fonction f_n est :
× continue sur $[0, +\infty[$,
× strictement croissante sur $[0, +\infty[$.
Elle réalise donc une bijection de $[0, +\infty[$ sur $f_n([0, +\infty[) = [-2, +\infty[$.

Ainsi, $0 \in [-2, +\infty[$ admet un unique antécédent $u_n \in [0, +\infty[$ par la fonction f_n. □

e) Préciser la valeur de u_0. Dans la suite on supposera que $n \geqslant 1$.

Démonstration.

La fonction $f_0 : x \mapsto \mathrm{e}^x - 3$ s'annule en $u_0 = \ln(3)$. □

f) Vérifier : $\forall n \in \mathbb{N}^*$, $u_n \in]0, 1[$.

Démonstration.
Soit $n \in \mathbb{N}^*$.
- On a :
  × $f_n(0) = -2 < 0$,
  × $f_n(u_n) = 0$ par définition,
  × $f_n(1) = \mathrm{e}^1 + n - 3 > 0$ car $n \geqslant 1$ et $\mathrm{e}^1 > 2,71$.

$$f_n(0) < f_n(u_n) < f_n(1)$$

- D'après la question ***1.d)***, la fonction f_n réalise une bijection de $[0, +\infty[$ sur $[-2, +\infty[$. D'après le théorème de la bijection, $f_n^{-1} : [-2, +\infty[\to [0, +\infty[$ est strictement croissante sur $[-2, +\infty[$. En appliquant f_n^{-1}, on obtient alors :

$$\begin{array}{ccccc} f_n^{-1}\big(f_n(0)\big) & < & f_n^{-1}\big(f_n(u_n)\big) & < & f_n^{-1}\big(f_n(1)\big) \\ \| & & \| & & \| \\ 0 & & u_n & & 1 \end{array}$$

On en déduit : $0 < u_n < 1$. □

2. Écrire une fonction **Scilab** qui prend un entier n et qui calcule une valeur approchée de u_n à $0,001$ près par la méthode de dichotomie.

Démonstration.
Commençons par rappeler le cadre de la recherche par dichotomie.

Calcul approché d'un zéro d'une fonction par dichotomie
Données :
× une fonction $f : \mathbb{R} \to \mathbb{R}$,
× un intervalle de recherche $[a, b]$,
× une précision de recherche ε.

Résultat : une valeur approchée à ε près d'un zéro (sur l'intervalle $[a, b]$) de la fonction f.
Autrement dit, une valeur approchée (à ε près) d'un réel $x \in [a, b]$ tel que : $f(x) = 0$.

- La dichotomie est une méthode itérative dont le principe, comme son nom l'indique, est de découper à chaque itération l'intervalle de recherche en deux nouveaux intervalles. L'intervalle de recherche est découpé en son milieu. On obtient deux nouveaux intervalles :
  - × un intervalle dans lequel on sait que l'on va trouver un zéro de f.
    Cet intervalle est conservé pour l'itération suivante.
  - × un intervalle dans lequel ne se trouve pas forcément un zéro de f.
    Cet intervalle n'est pas conservé dans la suite de l'algorithme.

  La largeur de l'intervalle de recherche est ainsi divisée par 2 à chaque itération.
  On itère jusqu'à obtenir un intervalle I contenant un zéro de f et de largeur plus faible que ε.
  Les points de cet intervalle I sont tous de bonnes approximations du zéro contenu dans I.
- C'est le **théorème des valeurs intermédiaires** qui permet de choisir l'intervalle qu'il faut garder à chaque étape. Rappelons son énoncé et précisons maintenant l'algorithme :

Théorème des Valeurs Intermédiaires

Soit $f : [a, b] \to \mathbb{R}$ continue sur l'**intervalle** $[a, b]$.

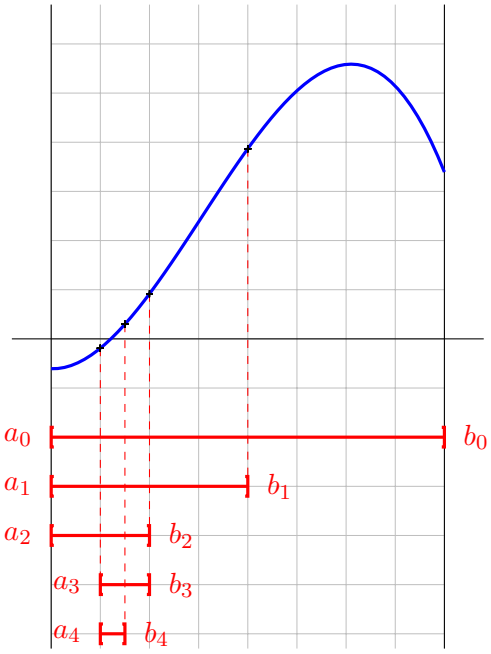
Supposons : $f(a)\ f(b) \leqslant 0$.

Alors il existe $c \in [a, b]$ tel que $f(c) = 0$.

Calcul des suites (a_m), (b_m), (c_m)

Cas $f(a) \leqslant 0$ et $f(b) \geqslant 0$

- Initialement, $a_0 = a$, $b_0 = b$
- À chaque tour de boucle (tant que $b_m - a_m > \varepsilon$) :
  - × $c_m = \dfrac{a_m + b_m}{2}$ (point milieu de $[a_m, b_m]$)
    - × si $f(c_m) < 0$ alors :
      - $a_{m+1} = c_m$
      - $b_{m+1} = b_m$
    - × si $f(c_m) \geqslant 0$ alors :
      - $a_{m+1} = a_m$
      - $b_{m+1} = c_m$



- On construit ainsi une suite $\left([a_m, b_m]\right)_{m\in\mathbb{N}}$ de segments emboîtés :
  - × contenant tous un zéro de f,
  - × dont la largeur est divisée par deux d'un rang au suivant.
- Il reste enfin à adapter cet algorithme à l'énoncé.
  Soit $n \in \mathbb{N}^*$. On cherche une valeur de x telle que : $f_n(x) = 0$.
  On se fixe initialement l'intervalle de recherche $[0, 1]$ de sorte que l'équation $f_n(x) = 0$ ne possède qu'une solution, à savoir la valeur u_n qu'on cherche à approcher (d'après la question ***1.d)*** et ***1.f)***). D'un point de vue informatique, on crée des variables `g` (comme « gauche ») et `d` (comme « droite ») destinées à contenir les valeurs succesives de a_m et b_m. Ces variables sont initialisées respectivement à 0 et 1.

```
2     g = 0
```

  On procède alors de manière itérative, tant que l'intervalle de recherche n'est pas de largeur plus faible que la précision 0.001 escomptée.

```
4     while d - g > 0.001
```

  On commence par définir le point milieu du segment de recherche.

```
5        m = (g+d) / 2
```

Puis on teste si $f_n(\texttt{m}) < 0$.
Si c'est le cas, la recherche s'effectue dans le demi-segment de droite.

```
6        if fn(n, m) < 0 :
7          g = m
```

Sinon, elle s'effectue dans le demi-segment de gauche.

```
8        else :
9          d = m
```

En sortie de boucle, on est assuré que le segment de recherche, mis à jour au fur et à mesure de l'algorithme, est de largeur plus faible que 0.001 et contient un zéro de f_n. Tout point de cet intervalle est donc une valeur approchée à 0.001 près de ce zéro.
On peut alors choisir de renvoyer le point le plus à gauche du segment.

```
11     return g
```

On peut tout aussi bien choisir le point le plus à droite :

```
11     return d
```

Ou encore le point milieu :

```
11     return (g + d) / 2
```

Ce dernier choix présente un avantage : tout point (dont le zéro recherché) du dernier intervalle de recherche se situe à une distance d'au plus $\dfrac{0.001}{2}$ de ce point milieu.
On obtient ainsi une valeur approchée à $\dfrac{0.001}{2}$ du zéro recherché.

Commentaire

- On peut se demander combien de tours de boucle sont nécessaires pour obtenir le résultat. Pour le déterminer, il suffit d'avoir en tête les éléments suivants :
  - × l'intervalle de recherche initial $[0, 1]$ est de largeur 1.
  - × la largeur de l'intervalle de recherche est divisée par 2 à chaque tour de boucle.
    À la fin du $m^{\text{ème}}$ tour de boucle, l'intervalle de recherche est donc de largeur $\dfrac{1}{2^m}$.
  - × l'algorithme s'arrête lorsque l'intervalle devient de largeur plus faible que $0.001 = 10^{-2}$.

  On obtient le nombre d'itérations nécessaires en procédant par équivalence :

$$\begin{aligned} \frac{1}{2^m} \leqslant 10^{-2} &\iff 2^m \geqslant 10^2 && \textit{(par stricte décroissance de la fonction inverse sur } \mathbb{R}_+^*\textit{)} \\ &\iff m\ \ln(2) \geqslant 2\ \ln(10) && \textit{(par stricte croissance de la fonction } \ln \textit{ sur } \mathbb{R}_+^*\textit{)} \end{aligned}$$

  Ainsi : $\left\lceil 2 \times \dfrac{\ln(10)}{\ln(2)} \right\rceil$ tours de boucle suffisent.

  On retiendra que si l'on souhaite obtenir une précision de 2 chiffres après la virgule, il suffit d'effectuer de l'ordre de 2 tours de boucle. Cette algorithme est donc extrêmement rapide.

Commentaire

- Afin de permettre une bonne compréhension des mécanismes en jeu, on a détaillé avec beaucoup de précision la réponse à cette question. Cependant, compléter correctement le programme **Python** (on place ci-dessous le programme obtenu) démontre la bonne compréhension de l'algorithme demandé et permet d'obtenir tous les points alloués à cette question.

- On obtient le programme complet suivant.
(même si ce n'était pas explicitement demandé par l'énoncé, on avait tout intérêt à commencer par coder les fonctions f_n)

```
def fn(n, x) :
    y = np.exp(x) + n*x**2 - 3
    return y
```

```
def dichoto(n) :
    g = 0
    d = 1
    m = (g + d)/2
    while d - g > 0.001 :
        if fn(n,m) < 0 :
            g = m
        else :
            d = m
    m = (g + d)/2
    return m
```

□

3. a) Montrer : $\forall x \in]0,1[,\ f_{n+1}(x) > f_n(x)$.

Démonstration.
Soient $n \in \mathbb{N}^*$ et $x \in]0,1[$.

$$f_{n+1}(x) - f_n(x) \ = \ (\cancel{e^x} + (n+1)\,x^2 - \cancel{3}) - (\cancel{e^x} + n\,x^2 - \cancel{3}) \ = \ x^2 \ > \ 0$$

Ainsi, pour tout $x \in]0,1[$, on a : $f_{n+1}(x) > f_n(x)$.

□

b) En déduire le signe de $f_n(u_{n+1})$, puis le sens de variation de la suite (u_n).

Démonstration.

- Comme $u_{n+1} \in]0,1[$, on peut appliquer le résultat de la question précédente.

On obtient : $f_n(u_{n+1}) \ < \ f_{n+1}(u_{n+1})$.

En particulier, comme $f_{n+1}(u_{n+1}) = 0$, alors : $f_n(u_{n+1}) \ < \ 0$.

- D'après la question ***1.d)***, la fonction f_n réalise une bijection de $[0,+\infty[$ sur $[-2,+\infty[$.
D'après le théorème de la bijection, $f_n^{-1} : [-2,+\infty[\to [0,+\infty[$ est strictement croissante sur $[-2,+\infty[$. En appliquant f_n^{-1}, on obtient alors :

$$\begin{array}{ccc} f_n^{-1}\big(f_n(u_{n+1})\big) & < & f_n^{-1}\big(f_n(u_n)\big) \\ \| & & \| \\ u_{n+1} & & u_n \end{array}$$

La suite (u_n) est strictement décroissante.

Commentaire

- Cette question ***3.b)*** consiste en l'étude de la suite (u_n). On parle ici de « suite implicite » car on n'a pas accès à la définition explicite de la suite (u_n) mais simplement à la propriété qui permet de définir chacun de ses termes, à savoir :

  Pour tout $n \in \mathbb{N}$, u_n est l'unique solution dans $[0, +\infty[$ de l'équation $f_n(x) = 0$

  On comprend alors que l'étude de (u_n) va passer par l'étude des propriétés de la fonction P_n.

- De cette définition, on tire la propriété : $\boxed{\forall m \in \mathbb{N},\ f_m(u_m) = 0}$.
  Cette propriété est au cœur de l'étude de la suite implicite (u_n).
  C'est de cette propriété dont on se sert ici (en $m = n$) pour démontrer la monotonie de la suite (u_n). Comme la suite (u_n) est définie de manière implicite, cette étude ne se réalise pas directement en étudiant la différence $u_{n+1} - u_n$. Il est par contre très classique de passer par l'inégalité :
  $$f_n(u_{n+1}) \leqslant f_n(u_n)$$
  et de conclure : $u_n \leqslant u_{n+1}$ à l'aide d'une propriété de f_n.

□

c) Montrer que (u_n) est convergente. On note ℓ sa limite.

Démonstration.

La suite (u_n) est décroissante et minorée par 1. Elle converge donc vers une limite ℓ qui vérifie $0 \leqslant \ell \leqslant 1$.

□

d) On suppose dans cette question que $\ell > 0$.
Calculer la limite de $\mathrm{e}^{u_n} + n\,u_n^2 - 3$ et en déduire une contradiction.

Démonstration.
Supposons : $\ell > 0$. Alors :
× $\mathrm{e}^{u_n} \underset{n\to+\infty}{\longrightarrow} \mathrm{e}^{\ell}$,
× $n\,{u_n}^2 \underset{n\to+\infty}{\sim} n\,\ell^2 \underset{n\to+\infty}{\longrightarrow} +\infty$.
On en déduit que $f_n(u_n) = \mathrm{e}^{u_n} + n\,u_n^2 - 3 \underset{n\to+\infty}{\longrightarrow} +\infty$.

Or, par définition de u_n, on a : $f_n(u_n) = 0$, ce qui contredit le calcul précédent.

□

e) Donner enfin la valeur de ℓ.

Démonstration.

D'après la question précédente, $\ell \leqslant 0$. Comme de plus $0 \leqslant \ell \leqslant 1$, on a : $\ell = 0$

□

f) Montrer que $\sqrt{\frac{n}{2}}\,u_n$ tend vers 1 quand n tend vers $+\infty$.

Démonstration.

- On a : $f_n(u_n) = 0 = \mathrm{e}^{u_n} + n\,u_n^2 - 3$.

  Ainsi : $n\,u_n^2 = 3 - \mathrm{e}^{u_n}$.

- On en déduit :

$$\frac{n}{2}\,u_n^2 = \frac{3-\mathrm{e}^{u_n}}{2} \qquad \text{et} \qquad \sqrt{\frac{n}{2}}\,\sqrt{u_n^2} = \sqrt{\frac{3-\mathrm{e}^{u_n}}{2}}$$

Enfin, comme $u_n \underset{n\to+\infty}{\longrightarrow} 0$, on en déduit : $\sqrt{\frac{n}{2}}\,u_n \underset{n\to+\infty}{\longrightarrow} \sqrt{\frac{3-\mathrm{e}^0}{2}} = \sqrt{\frac{2}{2}} = 1$.

Commentaire

On a démontré :

$$\sqrt{\frac{n}{2}}\,u_n \;=\; \frac{u_n}{\sqrt{\frac{2}{n}}} \;\underset{n\to+\infty}{\longrightarrow}\; 1$$

Ceci signifie que les suites (u_n) et $\left(\sqrt{\frac{2}{n}}\right)$ ont même comportement asymptotique.

Autrement dit : $u_n \underset{n\to+\infty}{\sim} \sqrt{\frac{2}{n}}$.

□

Exercice 10
Pour tout entier n non nul, on note h_n la fonction définie sur $\mathbb{R}_+^*$ par :

$$\forall x > 0, \quad h_n(x) \;=\; f(x^n, 1) \;=\; x^n + 1 + \frac{1}{x^n}$$

1. Démontrer que pour tout entier naturel n non nul, la fonction h_n est strictement décroissante sur $]0,1[$ et strictement croissante sur $[1,+\infty[$.

Démonstration.
Soit $n \in \mathbb{N}^*$.

- La fonction h_n est dérivable sur $\mathbb{R}_+^*$ car elle est la somme des fonctions :
  - × $x \mapsto x^n + 1$ dérivable sur $\mathbb{R}_+^*$ car polynomiale.
  - × $x \mapsto \dfrac{1}{x^n}$ dérivable sur $\mathbb{R}_+^*$ en tant qu'inverse de la fonction $x \mapsto x^n$:
    - dérivable sur $\mathbb{R}_+^*$ car polynomiale,
    - et qui ne s'annule pas sur $\mathbb{R}_+^*$.
- Soit $x \in \mathbb{R}_+^*$.

$$h_n'(x) \;=\; n\,x^{n-1} - \frac{n}{x^{n+1}} \;=\; n\,\frac{x^{2n}-1}{x^{n+1}}$$

Comme $n > 0$ et $x^{n+1} > 0$, la quantité $h_n'(x)$ est du signe de $x^{2n} - 1$. On en déduit :

$$h_n'(x) > 0 \;\Leftrightarrow\; x^{2n} - 1 > 0 \;\Leftrightarrow\; x^{2n} > 1 \;\Leftrightarrow\; x > 1$$

La dernière équivalence est obtenue par stricte croissance de la fonction $x \mapsto x^{2n}$ sur $\mathbb{R}_+^*$.

- On en déduit le tableau de variations suivant.

x	0		1		$+\infty$
Signe de $h_n'(x)$	‖	$-$	0	$+$	
Variations de h_n	‖ $+\infty$	↘	3	↗	$+\infty$

Commentaire

Afin de déterminer le signe de la quantité $x^{2n} - 1$, on pouvait rédiger autrement.

- Une première méthode consiste à étudier la fonction $g_n : x \mapsto x^{2n} - 1$.
  Cette fonction étant polynomiale, elle est dérivable sur $\mathbb{R}$ et pour tout $x \in \mathbb{R}_+^*$:

$$g_n'(x) \;=\; 2n\,x^{2n-1} > 0$$

  La fonction g_n est donc strictement croissante sur $]0,+\infty[$. Enfin, comme $g_n(1) = 0$:

$$\forall x \in]0,1[,\; x^{2n} - 1 < 0 \qquad \text{et} \qquad \forall x \in]1,+\infty[,\; x^{2n} - 1 > 0$$

- Une deuxième méthode consiste à factoriser $x^{2n} - 1$:

$$x^{2n} - 1 \;=\; \left(x^n\right)^2 - 1^2 \;=\; (x^n - 1)\,(x^n + 1)$$

  Comme $x > 0$, alors $x^n > 0$ et $x^n + 1 > 0$. Le signe de $x^{2n} - 1$ est donc celui de $x^n - 1$. Or :

$$x^n - 1 \;=\; (x-1)(1 + x + \ldots + x^{n-1})$$

  Comme $1 + x + \ldots + x^{n-1} > 0$, le signe de $x^n - 1$ (et donc de $x^{2n} - 1$) est celui de $x - 1$. □

2. En déduire que pour tout entier n non nul, l'équation $h_n(x) = 4$ admet exactement deux solutions, notées u_n et v_n et vérifiant : $0 < u_n < 1 < v_n$.

Démonstration.
Soit $n \in \mathbb{N}^*$.

- La fonction h_n est :
  - × continue sur $]0, 1[$,
  - × strictement décroissante sur $]0, 1[$.

  Elle réalise donc une bijection de $]0, 1[$ sur $h_n\big(]0, 1[\big)$. Or :

$$h_n\big(]0, 1[\big) =]h_n(1), \lim_{x \to 0} h_n(x)[=]3, +\infty[$$

  Comme $4 \in]3, +\infty[$, l'équation $h_n(x) = 4$ admet une unique solution sur $]0, 1[$.
  On note u_n cette solution.
- De même, la fonction h_n est :
  - × continue sur $]1, +\infty[$,
  - × strictement croissante sur $]1, +\infty[$.

  Elle réalise donc une bijection de $]1, +\infty[$ sur $h_n\big(]1, +\infty[\big)$. Or :

$$h_n\big(]1, +\infty[\big) =]h_n(1), \lim_{x \to +\infty} h_n(x)[=]3, +\infty[$$

  Comme $4 \in]3, +\infty[$, l'équation $h_n(x) = 4$ admet une unique solution sur $]1, +\infty[$.
  On note v_n cette solution.
- On remarque enfin : $h_n(1) = 3 \neq 4$. Ainsi, $x = 1$ n'est pas solution de l'équation $h_n(x) = 4$.

Pour tout entier n non nul, l'équation $h_n(x) = 4$ admet exactement deux solutions, notées u_n et v_n et vérifiant : $0 < u_n < 1 < v_n$. □

3. a) Démontrer :

$$\forall x > 0,\ \forall n \in \mathbb{N}^*,\ h_{n+1}(x) - h_n(x) = \frac{(x-1)(x^{2n+1}-1)}{x^{n+1}}$$

Démonstration.
Soit $x > 0$ et soit $n \in \mathbb{N}^*$.

$$\begin{aligned} h_{n+1}(x) - h_n(x) &= \left(x^{n+1} + \cancel{1} + \frac{1}{x^{n+1}}\right) - \left(x^n + \cancel{1} + \frac{1}{x^n}\right) \\ &= \frac{x^{2n+2}+1}{x^{n+1}} - \frac{x^{2n+1}+x}{x^{n+1}} \\ &= \frac{x^{2n+2} - x^{2n+1} - (x-1)}{x^{n+1}} \\ &= \frac{x^{2n+1}(x-1) - (x-1)}{x^{n+1}} = \frac{(x-1)(x^{2n+1}-1)}{x^{n+1}} \end{aligned}$$

$$\boxed{\forall x > 0,\ \forall n \in \mathbb{N}^*,\ h_{n+1}(x) - h_n(x) = \frac{(x-1)(x^{2n+1}-1)}{x^{n+1}}}$$

□

b) En déduire : $\forall n \in \mathbb{N}^*,\ h_{n+1}(v_n) \geqslant 4$.

Démonstration.
Soit $n \in \mathbb{N}^*$.

- D'après la question précédente, pour tout $x > 0$:

$$h_{n+1}(x) - h_n(x) = \frac{(x-1)(x^{2n+1}-1)}{x^{n+1}}$$

Comme $x > 0$ alors $x^{n+1} > 0$. Ainsi, la quantité $h_{n+1}(x) - h_n(x)$ est du signe du produit $(x-1)(x^{2n+1}-1)$. Notons alors que pour tout $x > 1$, $x^{2n+1} - 1 > 0$ (vu précédemment).

$$\boxed{\text{Ainsi : } \forall n \in \mathbb{N}^*,\ \forall x > 1,\ h_{n+1}(x) - h_n(x) > 0.}$$

- On applique l'inégalité précédente à $v_n > 1$ (d'après la question ***4.***). On obtient :

$$h_{n+1}(v_n) - h_n(v_n) > 0 \quad \text{ou encore} \quad h_{n+1}(v_n) > h_n(v_n) = 4$$

$$\boxed{\text{On a bien : } \forall n \in \mathbb{N}^*,\ h_{n+1}(v_n) \geqslant 4.}$$

□

c) Montrer alors que la suite (v_n) est décroissante.

Démonstration.
Soit $n \in \mathbb{N}^*$.

- Par définition de v_{n+1}, on a : $h_{n+1}(v_{n+1}) = 4$. Ainsi, d'après la question précédente :

$$h_{n+1}(v_n) \geqslant h_{n+1}(v_{n+1})$$

- De plus, on sait d'après la question ***4.*** :
  - × $v_n > 1$ et $v_{n+1} > 1$,
  - × la fonction h_{n+1} réalise une bijection de $]1, +\infty[$ sur $]3, +\infty[$.

La réciproque de cette bijection, définie de $]3,+\infty[$ sur $]1,+\infty[$ est strictement croissante car de même monotonie que h_{n+1} sur $]1,+\infty[$. En l'appliquant de part et d'autre de l'inégalité :

$$v_n \geqslant v_{n+1}$$

On en conclut : $\forall n \in \mathbb{N}^*$, $v_n \geqslant v_{n+1}$. La suite (v_n) est donc décroissante.

Commentaire

- La **Partie B** consiste en l'étude de la suite (v_n). On parle ici de « suite implicite » car on n'a pas accès à la définition explicite de la suite (v_n) mais simplement à la propriété qui permet de définir chacun de ses termes, à savoir :

  Pour tout $n \in \mathbb{N}^*$, v_n est l'unique solution de l'équation $h_n(x) = 4$ sur $]1,+\infty[$

  On comprend alors que l'étude de (v_n) va passer par l'étude des propriétés de la fonction h_n.

- De cette définition, on tire la propriété : $\forall m \in \mathbb{N}^*,\ h_m(v_m) = 4$.
  Cette propriété est au cœur de l'étude de la suite implicite (v_n).
  On l'utilise en ***5.b)*** pour $m = n$ et en ***5.c)*** pour $m = n+1$.

- Comme la suite (v_n) est définie de manière implicite, on n'étudie pas la monotonie de (v_n) à l'aide de la différence $v_{n+1} - v_n$. Il est par contre très classique de passer par l'inégalité :

  $$h_{n+1}(v_n) \geqslant h_{n+1}(v_{n+1})$$

  et de conclure : $v_n \geqslant v_{n+1}$ à l'aide d'une propriété de h_{n+1}. □

4. a) Démontrer que la suite (v_n) converge vers un réel ℓ et montrer : $\ell \geqslant 1$.

Démonstration.
La suite (v_n) est :
× décroissante d'après la question ***5.c)***,
× minorée par 1 ($\forall n \in \mathbb{N}^*$, $v_n > 1$) d'après la question ***4***.

On en conclut que la suite (v_n) converge vers un réel $\ell \geqslant 1$. □

b) En supposant que $\ell > 1$, démontrer : $\lim\limits_{n\to+\infty} v_n^n = +\infty$.
En déduire une contradiction.

Démonstration.
Dans cette question, on suppose $\ell > 1$.

- Comme $\ell \neq 0$, on a : $v_n \underset{n\to+\infty}{\sim} \ell$. Ainsi :

$$(v_n)^n \underset{n\to+\infty}{\sim} \ell^n \underset{n\to+\infty}{\longrightarrow} +\infty \quad (\textit{car } \ell > 1)$$

On a bien : $\lim\limits_{n\to+\infty} (v_n)^n = +\infty$.

Commentaire

Comme la suite (v_n) est à termes strictement positifs, on pouvait aussi écrire :

$$(v_n)^n \;=\; \exp\big(n\ln(v_n)\big)$$

Puis : $\ln(v_n) \underset{n\to+\infty}{\sim} \ln(\ell)$ puisque $\ell \neq 1$.
Ainsi : $n\ln(v_n) \underset{n\to+\infty}{\sim} n\,\ell \to +\infty$ car $\ell > 0$. Et enfin, par composition de limites :

$$\lim_{n\to+\infty} \exp\big(n\ln(v_n)\big) \;=\; +\infty$$

- Remarquons alors :

$$h_n(v_n) \;=\; (v_n)^n + 1 + \frac{1}{(v_n)^n}$$

Comme $\lim\limits_{n\to+\infty} (v_n)^n = +\infty$, on en conclut :

$$\lim_{n\to+\infty} h_n(v_n) = +\infty$$

Or, par définition de la suite (v_n), pour tout $n \in \mathbb{N}^*$, $h_n(v_n) = 4$ et ainsi :

$$\lim_{n\to+\infty} h_n(v_n) = 4$$

Contradiciton !

Commentaire

C'est encore une fois la propriété de définition des termes de la suite (v_n) qui est utilisée ici ($\forall n \in \mathbb{N}^*$, $h_n(v_n) = 4$). On insiste sur le fait que cette propriété est fondamentale pour l'étude de la suite implicite (v_n). □

c) Déterminer la limite de (v_n).

Démonstration.

- En question ***6.a)***, on a démontré que la suite (v_n) converge vers un réel $\ell \geqslant 1$.
- En question ***6.b)***, on a démontré que l'hypothèse $\ell > 1$ menait à une contradiction.
  On en conclut que sa négation est vérifiée, à savoir : $\ell \leqslant 1$.

Ainsi, la suite (v_n) converge ver $\ell = 1$.

Commentaire

L'énoncé donne les réponses aux questions ***6.a)*** et ***6.b)***. Il est donc possible de traiter la question ***6.c)***, question bilan, sans avoir réussi à traiter les questions précédentes. □

5. a) Montrer : $\forall n \geqslant 1,\ v_n \leqslant 3$.

Démonstration.
Soit $n \in \mathbb{N}^*$.

- Rappelons tout d'abord : $v_n > 1$ (question ***4.***). D'autre part :

$$\begin{aligned} v_n \leqslant 3 \quad &\Leftrightarrow \quad h_n(v_n) \leqslant h_n(3) && \textit{(car } h_n \textit{ est strictement croissante sur }]1,+\infty[\textit{)} \\ &\Leftrightarrow \quad 4 \leqslant h_n(3) && \textit{(par définition de } v_n\textit{)} \end{aligned}$$

- Or, comme $3^n \geqslant 3$ (puisque $n > 0$) :

$$h_n(3) \;=\; 3^n + 1 + \frac{1}{3^n} \;\geqslant\; 3 + 1 + \frac{1}{3^n} > 4$$

Pour tout $n \in \mathbb{N}^*$, on a bien : $h_n(3) \geqslant 4$ et donc $v_n \geqslant 3$.

Commentaire

- Encore une fois, c'est la propriété de définition des termes de la suite (v_n) qui est utilisée ici. C'est logique puisqu'on ne connaît pas d'expression explicite des termes de (v_n).
- La présence de la quantification « $\forall n \in \mathbb{N}^*$ » peut faire penser à utiliser une récurrence. Ce type de raisonnement nécessite l'existence d'une propriété liant les termes de rangs successifs afin pouvoir mettre en œuvre l'étape d'hérédité. C'est pourquoi la récurrence est l'outil de base de démonstration des propriétés des suites récurrentes d'ordre 1 (le terme au rang $n+1$ s'exprime directement en fonction du terme au rang n). L'utilisation est plus rare dans le cas des suites implicites mais était possible dans cette question. En effet, comme la suite (v_n) est décroissante, on sait que pour tout $n \in \mathbb{N}^*$:

$$v_{n+1} \leqslant v_n$$

L'étape d'hérédité ne pose pas de problème.
En effet, si l'on sait $v_n \leqslant 3$, on obtient alors, par transitivité : $v_{n+1} \leqslant v_n \leqslant 3$.
Il reste alors à démontrer la propriété d'initialisation, à savoir $v_1 \leqslant 3$.
Encore une fois, il faut revenir à la propriété de définition des termes de la suite (v_n) :

$$h_1(v_1) \;=\; v_1 + 1 + \frac{1}{v_1} \;=\; 4$$

Ainsi : $v_1 \;=\; 3 - \frac{1}{v_1} \;\leqslant\; 3$ car $v_1 > 0$ (question *4.)*).

□

b) Écrire une fonction **Python** d'en-tête `def h(n,x)` qui renvoie la valeur de $h_n(x)$ lorsqu'on lui fournit un entier naturel n non nul et un réel $x \in \mathbb{R}_+^*$ en entrée.

Démonstration.

```
1  def h(n, x) :
2      y = x**n + 1 + 1/x**n
3      return y
```

Commentaire

Il n'y a aucune difficulté à coder en **Python** une fonction dont l'expression est donnée dans l'énoncé. Il est donc impensable de ne pas traiter cette question.

□

c) Compléter la fonction suivante pour qu'elle renvoie une valeur approchée à 10^{-5} près de v_n par la méthode de dichotomie lorsqu'on lui fournit un entier $n \geqslant 1$ en entrée :

```
def v(n) :
    a = 1
    b = 3
    while (b-a) > 10**(-5) :
        c = (a + b)/2
        if h(n,c) < 4 :
            ..........
        else :
            ..........
    return ..........
```

Démonstration.
Commençons par rappeler le cadre de la recherche par dichotomie.

> *Calcul approché d'un zéro d'une fonction par dichotomie*
> **Données** :
> × une fonction $f : \mathbb{R} \to \mathbb{R}$,
> × un intervalle de recherche $[a, b]$,
> × une précision de recherche ε.
>
> **Résultat** : une valeur approchée à ε près d'un zéro (sur l'intervalle $[a, b]$) de la fonction f.
> Autrement dit, une valeur approchée (à ε près) d'un réel $x \in [a, b]$ tel que : $f(x) = 0$.

- La dichotomie est une méthode itérative dont le principe, comme son nom l'indique, est de découper à chaque itération l'intervalle de recherche en deux nouveaux intervalles. L'intervalle de recherche est découpé en son milieu. On obtient deux nouveaux intervalles :
  × un intervalle dans lequel on sait que l'on va trouver un zéro de f.
  Cet intervalle est conservé pour l'itération suivante.
  × un intervalle dans lequel ne se trouve pas forcément un zéro de f.
  Cet intervalle n'est pas conservé dans la suite de l'algorithme.

  La largeur de l'intervalle de recherche est ainsi divisée par 2 à chaque itération.
  On itère jusqu'à obtenir un intervalle I contenant un zéro de f et de largeur plus faible que ε.
  Les points de cet intervalle I sont tous de bonnes approximations du zéro contenu dans I.

- C'est le **théorème des valeurs intermédiaires** qui permet de choisir l'intervalle qu'il faut garder à chaque étape. Rappelons son énoncé et précisons maintenant l'algorithme :

Théorème des Valeurs Intermédiaires

Soit $f : [a, b] \to \mathbb{R}$ continue sur l'**intervalle** $[a, b]$.

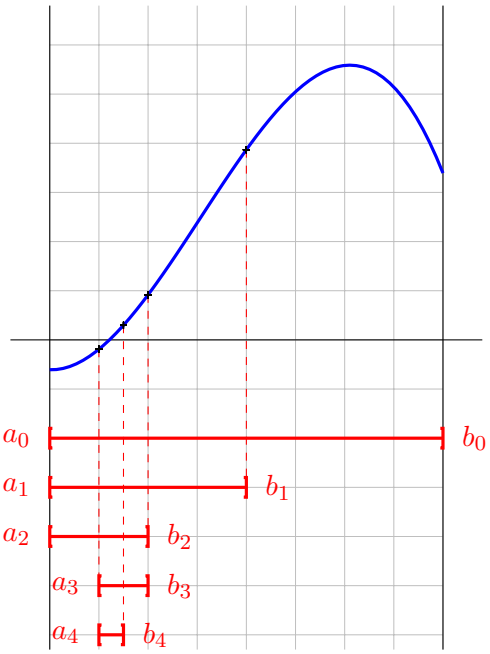
Supposons : $f(a)\ f(b) \leqslant 0$.

Alors il existe $c \in [a, b]$ tel que $f(c) = 0$.

Calcul des suites (a_m), (b_m), (c_m)

Cas $f(a) \leqslant 0$ et $f(b) \geqslant 0$

- Initialement, $a_0 = a$, $b_0 = b$
- À chaque tour de boucle (tant que $b_m - a_m > \varepsilon$) :
  - $c_m = \dfrac{a_m + b_m}{2}$ (point milieu de $[a_m, b_m]$)
  - si $f(c_m) < 0$ alors : $a_{m+1} = c_m$, $b_{m+1} = b_m$
  - si $f(c_m) \geqslant 0$ alors : $a_{m+1} = a_m$, $b_{m+1} = c_m$



- On construit ainsi une suite $\big([a_m, b_m]\big)_{m\in\mathbb{N}}$ de segments emboîtés :
  - contenant tous un zéro de f,
  - dont la largeur est divisée par deux d'un rang au suivant.
- Il reste enfin à adapter cet algorithme à l'énoncé.
  Soit $n \in \mathbb{N}^*$. On cherche une valeur de x telle que : $h_n(x) = 4$ ce qui s'écrit :

$$h_n(x) - 4 = 0 \quad \text{ou encore} \quad f_n(x) = 0 \quad \text{où} \quad f_n : x \mapsto h_n(x) - 4$$

On se fixe initialement l'intervalle de recherche $[1, 3]$ de sorte que l'équation $f_n(x) = 0$ ne possède qu'une solution, à savoir la valeur v_n qu'on cherche à approcher. D'un point de vue informatique, on crée des variables `a` et `b` destinées à contenir les valeurs succesives de a_m et b_m. Ces variables sont initialisées respectivement à 1 et 3.

```
2        a = 1
3        b = 3
```

On procède alors de manière itérative, tant que l'intervalle de recherche n'est pas de largeur plus faible que la précision 10^{-5} escomptée.

```
4    while (b-a) > 10**(-5)
```

On commence par définir le point milieu du segment de recherche.

```
5            c = (a+b) / 2
```

Puis on teste si $f_n(\texttt{c}) < 0$, c'est-à-dire si $h_n(\texttt{c}) < 4$.
Si c'est le cas, la recherche s'effectue dans le demi-segment de droite.

```
6            if h(n,c) < 4 :
7                a = c
```

Sinon, elle s'effectue dans le demi-segment de gauche.

```
8            else :
9                b = c
```

En sortie de boucle, on est assuré que le segment de recherche, mis à jour au fur et à mesure de l'algorithme, est de largeur plus faible que 10^{-5} et contient un zéro de f_n. Tout point de cet intervalle est donc une valeur approchée à 10^{-5} près de ce zéro.
On peut alors choisir de renvoyer le point le plus à gauche du segment.

```
11       return a
```

On peut tout aussi bien choisir le point le plus à droite :

```
11       return b
```

Ou encore le point milieu :

```
11       return (a + b) / 2
```

Ce dernier choix présente un avantage : tout point (dont le zéro recherché) du dernier intervalle de recherche se situe à une distance d'au plus $\dfrac{10^{-5}}{2}$ de ce point milieu.
On obtient ainsi une valeur approchée à $\dfrac{10^{-5}}{2}$ du zéro recherché.

Commentaire

- On peut se demander combien de tours de boucle sont nécessaires pour obtenir le résultat. Pour le déterminer, il suffit d'avoir en tête les éléments suivants :
  - × l'intervalle de recherche initial $[1, 3]$ est de largeur 2.
  - × la largeur de l'intervalle de recherche est divisée par 2 à chaque tour de boucle. À la fin du $m^{\text{ème}}$ tour de boucle, l'intervalle de recherche est donc de largeur $\dfrac{2}{2^m}$.
  - × l'algorithme s'arrête lorsque l'intervalle devient de largeur plus faible que 10^{-5}.

  On obtient le nombre d'itérations nécessaires en procédant par équivalence :

$$\begin{aligned}
\frac{2}{2^m} \leqslant 10^{-5} &\Leftrightarrow \frac{2^m}{2} \geqslant 10^5 && \textit{(par stricte décroissance de la fonction inverse sur } \mathbb{R}_+^*\textit{)}\\
&\Leftrightarrow 2^m \geqslant 2 \times 10^5 && \textit{(car } 4 > 0\textit{)}\\
&\Leftrightarrow m\ \ln(2) \geqslant \ln(2) + 5\ \ln(10) && \textit{(par stricte croissance de la fonction } \ln \textit{ sur } \mathbb{R}_+^*\textit{)}
\end{aligned}$$

  Ainsi : $\left\lceil 5\,\dfrac{\ln(10)}{\ln(2)} + 1 \right\rceil$ tours de boucle suffisent.

  On retiendra que si l'on souhaite obtenir une précision de 5 chiffres après la virgule, il suffit d'effectuer de l'ordre de 5 tours de boucle. Cette algorithme est donc extrêmement rapide.
- Afin de permettre une bonne compréhension des mécanismes en jeu, on a détaillé avec beaucoup de précision la réponse à cette question. Cependant, compléter correctement le programme **Python** (on place ci-dessous le programme obtenu) démontre la bonne compréhension de l'algorithme demandé et permet d'obtenir tous les points alloués à cette question.

- On obtient le programme complet suivant.

```
def v(n) :
    a = 1
    b = 3
    while (b-a) > 10**(-5) :
        c = (a + b)/2
        if h(n,c) < 4 :
            a = c
        else :
            b = c
    return (a + b)/2
```

□