Devoir Maison de mahtématiques N°1

Oscar Plaisant

15 septembre 2020

Exercice 1 :

1. **a)** Déterminer u_0, u_1, u_2 et u_3.

$u_0 = 200$	$u_1 = \frac{u_0}{2} + 180 = 280$
$u_2 = \frac{u_1}{2} + 180 = 320$	$u_3 = \frac{u_2}{2} + 180 = 340$

b) Peut-on confirmer les espérences de Marc ?

Il semble que oui.

c) Donner la relation de récurrence liant u_{n+1} à u_n.

$$u_n : \begin{cases} u_0 = 200 \\ u_{n+1} = \frac{u_n}{2} + 180 \end{cases}$$

2. On note (v_n) la suite définie sur $\mathbb{N}$ par $v_n = u_n - 360$

a) Montrer que $\forall n \in \mathbb{N}, v_{n+1} = \frac{1}{2} v_n$

$$\begin{aligned} v_{n+1} = \frac{1}{2} v_n \quad & \iff v_{n+1} = u_{n-1} - 360 \\ & \iff v_{n+1} = \frac{u_n}{2} + 180 - 360 \\ & \iff v_{n+1} = \frac{u_n}{2} - 180 \\ & \iff v_{n+1} = \frac{u_n - 360}{2} \\ & \iff v_{n+1} = \frac{v_n}{2} \end{aligned}$$

$$v_{n+1} \quad v_{n+1} = \iff \frac{1}{2} v_n$$

b) En déduire une expression du terme général de (v_n)

On sait que :

$$v_{n+1} = \frac{1}{2} v_n$$

Et que :

$$\begin{aligned} v_0 & = u_0 - 360 \\ & = 200 - 360 \\ & = -160 \end{aligned}$$

On peut donc poser :

$$v : \begin{cases} v_0 = -160 \\ v_{n+1} = \dfrac{1}{2} v_n \end{cases}$$

Et on à donc :

$$v_n = -160 \left(\frac{1}{2}\right)^n \iff v_{n+1} = \left(\frac{1}{2}\right)(-160)\left(\frac{1}{2}\right)^n \iff v_{n+1} = \frac{1}{2} v_n$$

On peut donc dire que :

$$v_n = -160 \left(\frac{1}{2}\right)^n$$

c) En déduire une expression générale de (u_n)
On sait que $u_n = v_n + 360$. On en déduit que :

$$u_n = -160 \left(\frac{1}{2}\right)^n + 360$$

d) Préciser ce que peut espérer Marc en étudiant les variations et la limite de (u_n).
On sait que $u_n = -160 \left(\frac{1}{2}\right)^n + 360$
On peut dire que :

$$-160 \left(\frac{1}{2}\right)^n = -160 \left(\frac{1}{2^n}\right)$$

Or, la suite $n \mapsto 2^n$ est strictement croissante et est toujours positive. Donc, la suite $n \mapsto \frac{1}{2^n}$ est strictement décroissante (puisque la fonction inverse est strictement décroissante sur $\mathbb{R}^{+*}$). On peut aussi dire que $\lim\limits_{n\to+\infty} \left(\dfrac{1}{2^n}\right) = 0$ car $\lim\limits_{n\to+\infty} (2^n) = +\infty$ et que $\lim\limits_{n\to+\infty} \left(\dfrac{1}{n}\right) = 0$. On peut donc dire que $\lim\limits_{n\to+\infty} \left(-160 \dfrac{1}{2^n}\right) = 0$, et finalement que :

$$\lim_{n\to+\infty} \left(-160 \left(\frac{1}{2}\right)^n + 360\right) = 360$$

La suite (u_n) tends donc vers 360 en $-\infty$

3. a) Plus généralement, en notant C le montant initial des économies de Marc, exprimer u_n en fonction de n et C

$$u_n = (C - 360) \left(\frac{1}{2}\right)^n + 360$$

b) Le montant initial de ses économies influe-t-il sur la limite de (u_n) ? sur ses variations ?

En reprenant le raisonnement dévoloppé à la question 2.d), on peut montrer que $\lim\limits_{n\to+\infty} \left(\dfrac{1}{2^n}\right) = 0$

Or, on sait que quelque soit ψ appartenant à $\mathbb{R}$:

$$lim_{n\to+\infty} \left(\psi \frac{1}{2^n}\right) = 0$$

On en déduit donc que le montant initial des économies de Marc n'influe pas sur la limite de (u_n).
Le montant initial des économies de Marc n'influe pas sur les variations de (u_n) si il est inférieur à 360. Si le montant initial C dépasse 360, le sens de variation de (u_n) est inversé (elle devient décroissante) mais sans changer de limite en $+\infty$. Si $C = 360$, la suite devient constante (puisque tous les termes sont annulés sauf le $+360$, ce qui fait que $C = 360 \implies \forall n \in \mathbb{N} u_n = 360$)

Exercice 2 :

1. Déterminer les valeurs de u_3 et u_4.
Le nombre de lapins au troisième mois est 2 car le couple initial à engendré un nouveau couple.

$$u_3 = 2$$

Le nombre de lapins au quatrième mois est 3 car le premier couple à engendré un nouveau couple, tandis que le second couple n'est pas encore pret à engendrer un nouveau couple.

$$u_4 = 3$$

2. Expliquer pourquoi les mois suivants, la suite vérifie la relation de récurrence $u_{n+2} = u_{n+1} + u_n$
Le nombre de lapins au mois n est égal à la somme du nombre de lapins au mois $n-1$ et du nombre de lapins matures (c'est à dire aptes à faire des enfants) au mois n. Le nombre de lapins matures au mois n est égal au nombre de lapins nés au mois $n-2$ ou avant, soit u_{n-2}.
On peut donc dire que :

$$\begin{aligned} (\text{lapins au mois } n) &= (\text{lapins au mois } n-1) + (\text{lapins matures au mois } n-1) \\ &= (\text{lapins au mois } n-1) + (\text{lapins nés avant le mois } n-1) \\ &= (\text{lapins au mois } n-1) + (\text{lapins au mois } n-2) \end{aligned}$$

En traduisant cela en langage mahtématique, on obtient :

$$u_n = u_{n-1} + u_{n-2} \iff u_{n+2} = u_{n+1} + u_n$$

3. Le(s)quel(s) des algorithmes ci-dessous a (ont) permis d'obtenir les premier termes suivants de la suite, et pour quelle valeur de n ?
Le programme ci-dessous permet de déterminer quel algorithme donne le bon résultat. Pour trouver la valeur de n correspondante, il suffit de tester quelques valeur jusqu'à ce que au moins l'un des algorithmes renvoient la bonne valeur

```python
def Algo1(n: int) -> iter:
    """Algo 1 traduit en fonction

    Args:
                    n (int): The number of numbers to compute.

    Yields:
                    iter: An iterable containing the fibonacci numbers.
    """
    u = 1
    v = 1

    yield u
    for j in range(2, n + 1):
        v = u + v
        u = v - u
        yield u


def Algo2(n: int) -> iter:
    """Algo 2 traduit en fonction

    Args:
                    n (int): The number of numbers to compute.

    Yields:
                    iter: An iterable containing the fibonacci numbers.
    """
    u = 1
```

```python
    v = 1
    yield v
    for j in range(1, n + 1):
        w = u + v
        u = v
        v = w
        yield v


def Algo3(n: int) -> iter:
    """Algo 3 traduit en fonction

    Args:
        n (int): The number of numbers to compute.

    Yields:
        iter: An iterable containing the powers of 2.
        """
    u = 1
    v = 1
    yield v
    for j in range(2, n + 1):
        v = u + v
        u = v
        yield v

if __name__ == "__main__":
    # on trouve la valeur de n par tatonnements, en testant
    # les valeurs possibles.
    n = 7

    print("")
    # print(list(Algo1(n)))
    try:
        assert list(Algo1(n)) == [1, 1, 2, 3, 5, 8, 13]
        print("Algo 1 fonctionne")
    except AssertionError:
        print("Algo 1 ne fonctionne pas")

    print("")
    # print(list(Algo2(n)))
    try:
        assert list(Algo2(n)) == [1, 1, 2, 3, 5, 8, 13]
        print("Algo 2 fonctionne")
    except AssertionError:
        print("Algo 2 ne fonctionne pas")

    print("")
    # print(list(Algo3(n)))
    try:
        assert list(Algo3(n)) == [1, 1, 2, 3, 5, 8, 13]
        print("Algo 3 fonctionne")
    except AssertionError:
        print("Algo 3 ne fonctionne pas")
```