

Contexte

On étudie une suite numérique définie par récurrence, dont le comportement dépend d'un paramètre réel k . Cet exercice permet d'aborder plusieurs notions du programme : étude d'une fonction, raisonnement par récurrence, variations d'une suite, convergence, limite et un algorithme en Python.

Un exercice
complet pour
réviser les notions
essentielles !

Soit k un nombre réel.

On considère la suite (u_n) définie par son premier terme u_0 et, pour tout entier naturel n ,

$$u_{n+1} = ku_n(1 - u_n).$$

Les deux parties de cet exercice sont indépendantes.

On étudie deux cas de figure selon les valeurs de k .

Partie 1 Cas $k = 1,9$ et $u_0 = 0,1$

Dans cette partie, $k = 1,9$ et $u_0 = 0,1$,

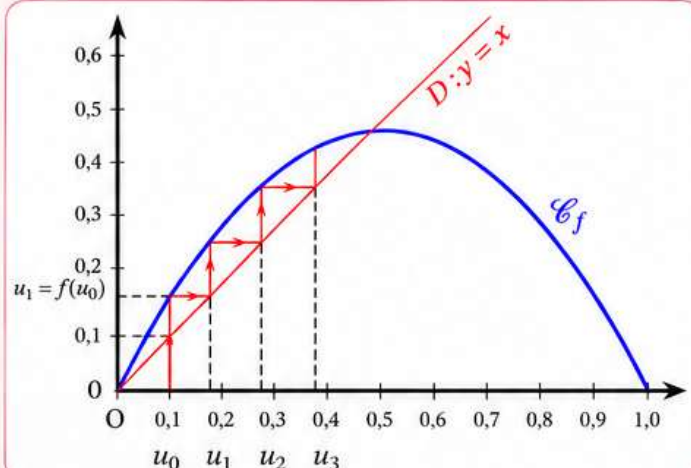
On a donc, pour tout entier naturel n , $u_{n+1} = 1,9u_n(1 - u_n)$.

1 On considère la fonction f définie sur $[0 ; 1]$ par

$$f(x) = 1,9x(1 - x).$$

- Étudier les variations de f sur l'intervalle $[0 ; 1]$.
- En déduire que si $x \in [0 ; 1]$ alors $f(x) \in [0 ; 1]$.

2 Ci-contre sont représentés les premiers termes de la suite (u_n) construits à partir de la courbe $\mathcal{C}_f$ de la fonction f et de la droite D d'équation $y = x$. Conjecturer le sens de variation de la suite (u_n) et sa limite éventuelle.



3 a. En utilisant les résultats de la question 1, démontrer par récurrence que pour tout entier naturel n :

$$0 \leq u_n \leq u_{n+1} \leq \frac{1}{2}.$$

- En déduire que la suite (u_n) converge.
- Déterminer sa limite.

Partie 2 Cas $k = \frac{1}{2}$ et $u_0 = \frac{1}{4}$

Dans cette partie, $k = \frac{1}{2}$ et $u_0 = \frac{1}{4}$.

On a donc, pour tout entier naturel n , $u_{n+1} = \frac{1}{2}u_n(1 - u_n)$ et $u_0 = \frac{1}{4}$.

On admet que pour tout entier naturel n : $0 \leq u_n \leq \left(\frac{1}{2}\right)^n$.

1 Démontrer que la suite (u_n) converge et déterminer sa limite.

2 On considère la fonction Python `algo(p)` où p désigne un entier naturel non nul :

Expliquer pourquoi, pour tout entier naturel non nul p , la boucle `while` ne tourne pas indéfiniment, ce qui permet à la commande `algo(p)` de renvoyer une valeur.

```
def algo(p) :
    u = 1/4
    n = 0
    while u > 10**(-p):
        u = 1/2*u*(1 - u)
        n = n + 1
    return(n)
```

