Chapitre 1 : Suites et raisonnement par récurrence

1. Rappels

Suites arithmétiques	Suites géométriques
Définition. • (u_n) est une suite arithmétique si, et seulement si, il existe un réel r tel que, pour tout entier naturel n, • (u_n) est une suite arithmétique si, et seulement si, la suite	**Définition.** • (u_n) est une suite géométrique si, et seulement si, il existe un réel q tel que, pour tout entier naturel n, • Si la suite (u_n) ne s'annule pas, (u_n) est une suite géométrique si, et seulement si, la suite
Expression de (u_n) en fonction de n. • Si la suite (u_n) est arithmétique de premier terme u_0 et de raison r, pour tout entier naturel n, • Pour tous entiers naturels n et p,	**Expression de (u_n) en fonction de n.** • Si la suite (u_n) est géométrique de premier terme u_0 et de raison q, pour tout entier naturel n, • Pour tous entiers naturels n et p,
Somme de termes consécutifs. • Pour tout entier naturel n, • Pour tous entiers naturels n et p tels que $p \leqslant n$,	**Somme de termes consécutifs.** • Pour tout entier naturel n et tout nombre réel q, • Pour tous entiers naturels n et p tels que $p \leqslant n$,

2. Raisonnement par récurrence

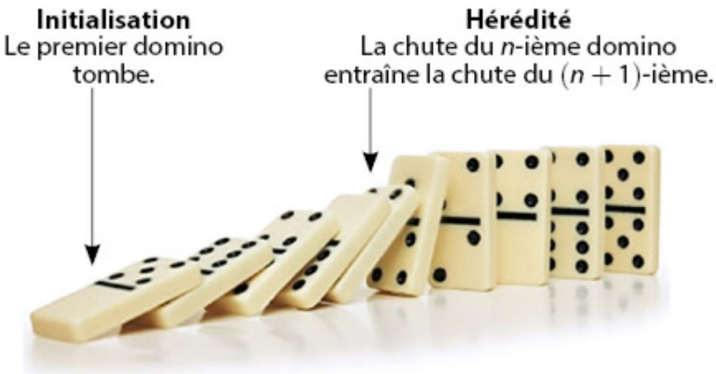
Théorème 1. **Principe du raisonnement par récurrence**

Soit $n_0 \in \mathbb{N}$. On considère une proposition $P(n)$ définie pour tout entier $n \geqslant n_0$.
Si la proposition $P(n)$ vérifie les deux conditions suivantes :

— **Initialisation :** $P(n_0)$ est vraie

— **Hérédité :** Pour tout entier naturel $k \geqslant n_0$, « $P(k)$ est vraie » implique « $P(k+1)$ est vraie »

Alors on peut conclure que, **pour tout entier naturel** $n \geqslant n_0$, la proposition $P(n)$ est vraie.



Exemple 1. On considère la suite (u_n) définie par $u_0 = 1$ et pour tout entier naturel n, $u_{n+1} = 2u_n + 1$.
Démontrer par récurrence que pour tout entier naturel n, $u_n = 2^{n+1} - 1$.

Solution

Pour tout $n \in \mathbb{N}$, on note :

$$\mathcal{P}(n) : u_n = 2^{n+1} - 1$$

Pour $n = 0$, on a d'une part $u_0 = 1$.
D'autre part, $2^{0+1} - 1 = 2 - 1 = 1$.
On a bien égalité donc on a démontré que $\mathcal{P}(0)$ est vraie.

Soit $k \in \mathbb{N}$ tel que $\mathcal{P}(k)$ est vraie.
Autrement dit, on suppose que $u_k = 2^{k+1} - 1$.
On veut alors démontrer que $\mathcal{P}(k+1)$ est vraie, c'est-à-dire que $u_{k+1} = 2^{(k+1)+1} - 1$.

$$\begin{aligned} u_{k+1} &= 2u_k + 1 \\ &= 2 \times (2^{k+1} - 1) + 1 \\ &\quad \text{(par hypothèse de récurrence)} \\ &= 2^{(k+1)+1} - 2 + 1 \\ &= 2^{(k+1)+1} - 1 \end{aligned}$$

Par conséquent, $P(k+1)$ est vraie.

Par le principe de récurrence, on en déduit que
$\boxed{\text{pour tout entier naturel } n,\ u_n = 2^{n+1} - 1}$

Description des étapes de la solution

Étape 1. On écrit explicitement la propriété à démontrer sans oublier de préciser explicitement les valeurs de n pour lesquelles la propriété va être démontrée.

Étape 2 (initialisation).
On vérifie que la propriété à démontrer est vraie pour la première valeur de n envisagée, c'est-à-dire ici $n = 0$.

Étape 3 (hérédité).
On se donne un entier k fixé mais quelconque, supérieur ou égal à la valeur initiale, c'est-à-dire ici 0, grâce à la phrase « Soit $k \in \mathbb{N}$ ».
On décrit explicitement le travail à effectuer : on suppose la propriété vraie au rang k et sous cette hypothèse, on la démontre au rang suivant $k+1$.
Puis on effectue ce travail.

Étape 4 (conclusion). On énonce de nouveau le résultat qu'on a maintenant démontré et on encadre ce résultat.

Exemple 2. Démontrer par récurrence que, pour tout $n \in \mathbb{N}$, $1 + 2 + 3 + \ldots + n = \dfrac{n(n+1)}{2}$.

Remarque 1. L'initialisation est indispensable sinon on peut démontrer des propriétés fausses !

Démontrons par exemple que la propriété « 2^n est divisible par 3 » est héréditaire sans vérifier l'initialisation.
Supposons qu'il existe un entier k tel que 2^k est divisible par 3.
$2^{k+1} = 2^k \times 2 = 3p \times 2$, où p est un entier (d'après l'hypothèse de récurrence)
$\phantom{2^{k+1}} = 6p$
Par conséquent, 2^{k+1} est divisible par 3.
L'hérédité est vérifiée et pourtant la propriété n'est jamais vraie.

2. 1. Inégalité de Bernoulli

Propriété 2.

Pour tout réel a strictement positif et pour tout entier naturel n, on a $(1+a)^n \geqslant 1+na$

Démonstration 1.

3. Comportement global d'une suite

3. 1. Suites monotones

Définition 1.

- Une suite (u_n) est **croissante** si pour tout entier naturel n, $u_{n+1} \geqslant u_n$.
- Une suite (u_n) est **décroissante** si pour tout entier naturel n, $u_{n+1} \leqslant u_n$.
- Une suite (u_n) est **constante** si pour tout entier naturel n, $u_{n+1} = u_n$.
- Une suite croissante ou décroissante est dite **monotone**.

Propriété 3.

On a trois méthodes pour étudier le sens de variation d'une suite (u_n) :

• **Méthode algébrique :** on compare directement, **pour tout entier** n, les termes consécutifs u_{n+1} et u_n :

— soit étudier le signe de la différence $u_{n+1} - u_n$

— soit étudier si $\dfrac{u_{n+1}}{u_n} \geqslant 1$ ou $\dfrac{u_{n+1}}{u_n} \leqslant 1$ si pour tout entier n, $u_n > 0$.

• **Méthode fonctionnelle :** Si (u_n) est une suite définie **explicitement** par $u_n = f(n)$, où f est une fonction définie sur $\mathbb{R}_+$, alors (u_n) et f ont le même sens de variation :

— si f est croissante, alors la suite (u_n) est croissante.

— si f est décroissante, alors la suite (u_n) est décroissante.

• **Méthode de raisonnement par récurrence :** elle s'applique si (u_n) est définie par une relation de récurrence du type $u_{n+1} = f(u_n)$, et consiste à démontrer que l'une des propriétés $P(n)$: $u_{n+1} \leqslant u_n$ ou $P(n)$: $u_{n+1} \geqslant u_n$ est vraie pour tout entier naturel n.

Exemple 3. En adoptant la bonne stratégie, étudier le sens de variation des suites définies par les expressions :

a. $u_n = n^2 - n + 2$ **b.** $u_n = \dfrac{2^n}{3^n}$ **c.** $u_n = \dfrac{3n-2}{n+1}$ **d.** $u_n = -\dfrac{1}{3}n + 3$

e. (u_n) définie par $u_0 = 1$ et pour $n \geqslant 0$, $u_{n+1} = 2u_n + 3$ **f.** $u_n = (n-5)^2$

g. (u_n) définie par $u_0 = 80$ et pour $n \geqslant 0$, $u_{n+1} = 0,5u_n + 30$ **h.** $u_n = -\left(\dfrac{1}{2}\right)^n$

3. 2. Suites majorées, minorées et bornées

Définition 2. On dit qu'une suite (u_n) définie sur $\mathbb{N}$ est :

- **majorée** s'il existe un réel M tel que, pour tout entier naturel n, $u_n \leqslant M$.
  M est appelé un **majorant** de (u_n).
- **minorée** s'il existe un réel m tel que, pour tout entier naturel n, $u_n \geqslant M$.
  m est appelé un **minorant** de (u_n).
- **bornée** si elle est à la fois majorée et minorée.

Remarque 2. La plupart du temps, pour montrer qu'une suite est majorée, minorée ou bornée, on utilise le raisonnement par récurrence, mais il arrive que certaines inégalités à déterminer soit triviales, comme le montre l'exemple suivant.

Exemple 4.

3. 3. Un exemple type BAC

Soit (a_n) la suite définie pour $n \geqslant 1$ par :

$$\left\{\begin{array}{l} a_1 = 0,5 \\ a_{n+1} = 0,5a_n + 0,3 \end{array}\right.$$

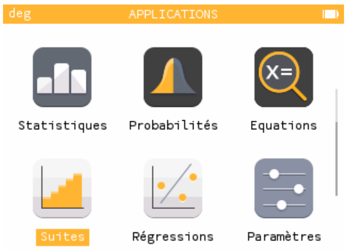
Démontrer par récurrence que, pour tout entier naturel $n \geqslant 1$, on a $0 \leqslant a_n \leqslant 0,6$.

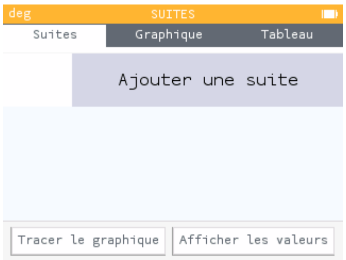
4. Programmer des suites sur calculatrice et en Python

Supposons qu'on ait une suite définie par récurrence sous la forme $u_{n+1} = f(u_n)$ et qu'on veuille calculer le terme de rang n. Si l'on cherche par exemple à calculer u_{500}, il faudra avoir les 499 termes précédents, ce qui peut s'avérer long, très très long... Heureusement, les outils numériques nous sont d'une grande aide !

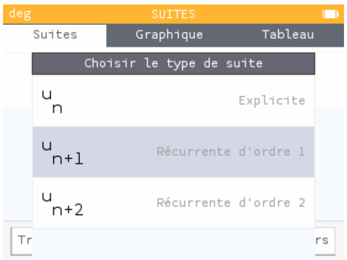
Exemple 5. Calculer et représenter des termes d'une suite à la calculatrice

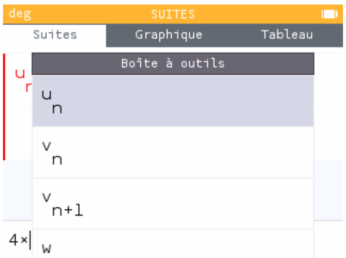
On considère la suite définie par $u_0 = 0,1$ et $u_{n+1} = 4u_n(1 - u_n)$ pour tout $n \in \mathbb{N}$.
On sélectionne la fonctionnalité *Suites* en utilisant les flèches directionnelles.



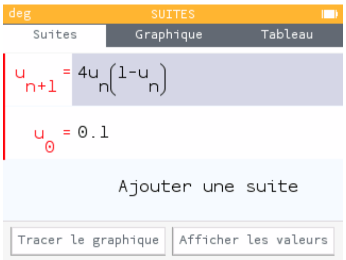


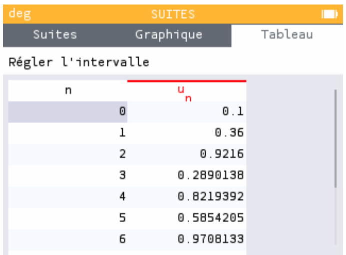
Ensuite, on sélectionne u_{n+1} et on complète la suite voulue.
Pour obtenir u_n, on utilise la touche *boîte à outils*.

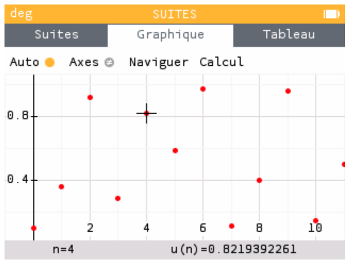




En utilisant les touches directionnelles, on peut maintenant sélectionner les onglets *Tableau* et *Graphique*.







Exemple 6. Afficher les termes d'une suite et déterminer un seuil avec Python

On considère la suite définie par $u_0 = 50$ et $u_{n+1} = 0,95u_n + 3$ pour tout $n \in \mathbb{N}$.

① Écrire un programme Python permettant d'afficher les 20 premiers termes de la suite (u_n).

② Écrire un programme Python permettant de déterminer le rang du premier terme strictement supérieur à 55.