

COURS

Chapitre 2

Suites numériques – Partie 1 : généralités

Version élève

Sommaire du chapitre

I	Des listes Python aux suites	2
II	Définitions et modes de génération	4
III	Représentation graphique d'une suite	7
IV	Sens de variation d'une suite	8
V	Calculatrice : générer et représenter une suite	11
VI	Python : calculer des termes et chercher un seuil	13
VII	Première intuition de la notion de limite	14
VIII	Bilan – Suites : généralités	15
IX	Annexe : Représentation en escalier d'une suite récurrente	17



Objectifs de la partie 1

- Passer d'une liste Python à la notation d'une suite et inversement.
- Calculer des termes d'une suite définie explicitement, par récurrence, par un algorithme ou par un motif.
- Représenter graphiquement une suite.
- Étudier le sens de variation d'une suite dans des cas simples.
- Reconnaître une suite majorée, minorée ou bornée.
- Utiliser Python pour générer des termes et rechercher un seuil.
- Conjecturer, sur des exemples simples, une limite éventuelle.

I Des listes Python aux suites

I.1 Une liste ordonnée de nombres



Des listes aux suites

Les suites numériques apparaissent depuis longtemps dans des problèmes de comptage, d'approximation ou d'évolution. Aujourd'hui, les listes en Python offrent une manière très concrète de manipuler une succession ordonnée de nombres avant de passer à la notation mathématique.



Une liste Python (list)

Une liste Python est une collection ordonnée d'objets. Les indices commencent à 0.

```
u = [3, 7, 11, 15, 19]
print(u[0])    # premier élément : 3
print(u[2])    # troisième élément : 11
```

$u = [3, 7, 11, 15, 19]$ $\xrightarrow{\text{indices}}$ $u_0 = 3, u_1 = 7, u_2 = 11, \dots$



Indice Python et indice mathématique

La correspondance est naturelle : l'élément d'indice n de la liste joue le rôle du terme u_n de la suite. Ainsi $u[3]$ correspond à u_3 .

I.2 Construire les premiers termes avec Python



Une première suite avec une liste en compréhension

On considère les nombres définis par $u(n) = 2n + 3$ pour $n \in \mathbb{N}$.

```
def u(n):
    return 2*n + 3

L = [u(n) for n in range(6)]
print(L)
```

.....

.....

.....

.....

.....



Construire une liste avec append

On considère la suite définie par

$$v_n = n^2 - 2n + 3.$$

La fonction Python `v` calcule un terme. La fonction `premiers_termes_v` construit ensuite la liste de v_0 à v_N par ajouts successifs.

```
def v(n):  
    return n**2 - 2*n + 3  
  
def premiers_termes_v(N):  
    L = []  
    for n in range(N + 1):  
        L.append(v(n))  
    return L  
  
print(premiers_termes_v(5))
```

Qu'affiche le programme ?

.....

.....

.....

.....

.....

.....



Trois manières de générer une liste

Le programme de Première demande de savoir produire des listes :

- **en extension** : `[3, 5, 7, 9]` ;
- **par ajouts successifs** avec `append` ;
- **en compréhension** : `[2*n+3 for n in range(6)]`.

II Définitions et modes de génération

II.1 Définition et notations

Définition 1 (Suite numérique)

Une **suite numérique** (*sequence*) est une fonction définie sur $\mathbb{N}$, ou sur une partie de $\mathbb{N}$, et à valeurs dans $\mathbb{R}$. On peut d'abord la noter sous forme fonctionnelle

$$u : n \mapsto u(n),$$

puis utiliser la notation indicielle

$$u_n = u(n).$$

La suite se note $(u_n)_{n \in \mathbb{N}}$, ou simplement (u_n) lorsqu'il n'y a pas d'ambiguïté.



Vocabulaire

- u_n est le **terme d'indice n** (*term of index n*);
- u_{n+1} est le **terme suivant** (*next term*) de u_n ;
- u_{n-1} est le **terme précédent** (*previous term*) de u_n si $n \geq 1$;
- une suite peut commencer à un rang n_0 différent de 0.



Deux suites données par leurs premiers termes

1. La suite (u_n) commence par

$$u_0 = 31, \quad u_1 = 20, \quad u_2 = 78, \quad u_3 = -15, \quad u_4 = 200, \quad u_5 = 2023.$$

2. La suite (v_n) est formée par les décimales successives d'un nombre. Par exemple, à partir de $\pi \approx 3,14159$, on peut écrire

$$v_1 = 1, \quad v_2 = 4, \quad v_3 = 1, \quad v_4 = 5, \quad v_5 = 9.$$



Les décimales de π

Le calcul des décimales de π constitue un exemple spectaculaire de suite finie produite par un algorithme. Les records modernes comptent des dizaines de milliers de milliards de décimales.

II.2 Suite définie explicitement

Définition 2 (Définition explicite)

Une suite est définie **explicitement** (*explicit formula*) lorsque son terme u_n est donné directement en fonction de n :

$$u_n = f(n).$$

Calcul de termes

Soit (u_n) définie pour $n \geqslant 2$ par

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

Une seconde suite explicite

Soit (v_n) définie pour $n \geqslant 1$ par

$$v_n = n^2.$$

Une suite qui alterne

Soit (w_n) définie pour tout $n \in \mathbb{N}$ par

$$w_n = (-1)^n.$$

II.3 Suite définie par récurrence

Définition 3 (Relation de récurrence)

Une suite est définie par une **relation de récurrence** (*recurrence relation*) lorsque chaque terme est obtenu à partir d'un ou de plusieurs termes précédents. Il faut alors préciser un terme initial.

Par exemple :

$$\begin{cases} u_0 = 2, \\ u_{n+1} = 3u_n - 1. \end{cases}$$

Calculer les premiers termes

Pour la suite précédente :

.....

.....



Même suite, langage naturel et Python

La relation $u_{n+1} = 3u_n - 1$ signifie : « multiplier le terme précédent par 3, puis soustraire 1 ».

```
def premiers_termes_u(N):
    u = 2
    L = [u]
    for n in range(N):
        u = 3*u - 1
        L.append(u)
    return L

print(premiers_termes_u(5))
```



Autre relation affine

On considère la suite définie par

$$u_0 = -5 \quad \text{et} \quad u_{n+1} = 2u_n + 3.$$

.....

.....

.....

.....

.....

.....

.....



Une relation non affine

On considère la suite définie par

$$v_1 = -2 \quad \text{et} \quad v_{n+1} = 2v_n^2 - 1.$$

.....

.....

.....

.....

.....

.....

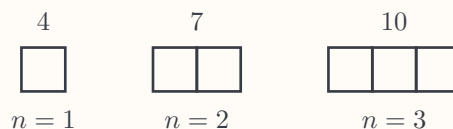
.....

II.4 Une suite donnée par un motif



Un motif de carrés

On aligne des carrés construits avec des allumettes. On note u_n le nombre d'allumettes nécessaires pour construire n carrés alignés.



On lit $u_1 = 4$, $u_2 = 7$, $u_3 = 10$. À chaque nouveau carré, on ajoute 3 allumettes.

.....

.....

.....

.....

.....

III Représentation graphique d'une suite

Définition 4 (Représentation graphique)

La représentation graphique d'une suite (u_n) est l'ensemble des points de coordonnées

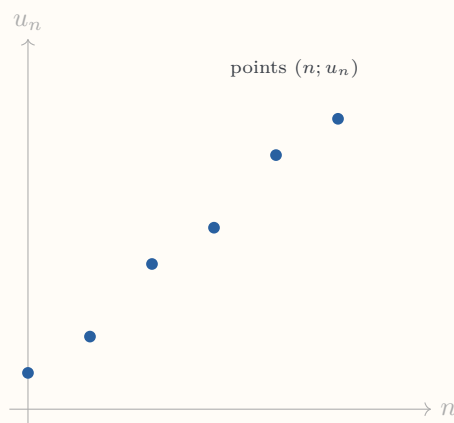
$$(n; u_n).$$

Contrairement à la courbe d'une fonction définie sur un intervalle réel, ces points sont isolés.



Lecture d'une représentation

La suite suivante est représentée par quelques-uns de ses termes.



On peut lire graphiquement, par exemple, $u_2 = 4$ et $u_5 = 8$.



Placer les termes sur une droite graduée

On considère la suite définie par $u_0 = 4$ et, pour tout $n \in \mathbb{N}$, par

$$u_{n+1} = 2u_n + 1.$$

On obtient $u_0 = 4$, $u_1 = 9$ et $u_2 = 19$. Ces trois nombres peuvent être placés directement sur une même droite graduée.



Suite récurrente et fonction associée

Lorsqu'une suite vérifie

$$u_{n+1} = f(u_n),$$

chaque nouveau terme est l'image du précédent par la fonction f . La courbe de f et la droite d'équation $y = x$ permettent de construire graphiquement les termes successifs sur l'axe des abscisses.

**Comprendre une construction en escalier**

Soit (u_n) définie par

$$u_0 = 15 \quad \text{et} \quad u_{n+1} = 1 + \frac{20}{u_n}.$$

On lui associe la fonction f définie par $f(x) = 1 + \frac{20}{x}$, de sorte que $u_{n+1} = f(u_n)$.

$$u_1 = \frac{7}{3} \approx 2,33, \quad u_2 = \frac{67}{7} \approx 9,57, \quad u_3 = \frac{207}{67} \approx 3,09, \quad u_4 = \frac{1547}{207} \approx 7,47.$$

Pour passer de u_n à u_{n+1} , on rejoint verticalement la courbe de f , puis horizontalement la droite $y = x$. On recommence ensuite le même procédé.

Voir l'annexe pour un exemple.

IV Sens de variation d'une suite**IV.1 Définitions****Définition 5** (Suite croissante ou décroissante)

Soit (u_n) une suite définie à partir d'un certain rang.

- (u_n) est **croissante** (*increasing*) si, pour tout n , $u_{n+1} \geq u_n$;
- (u_n) est **décroissante** (*decreasing*) si, pour tout n , $u_{n+1} \leq u_n$;
- elle est **constante** (*constant*) si, pour tout n , $u_{n+1} = u_n$.

IV.2 Suites majorées, minorées et bornées**Définition 6** (Majoration et minoration)

Soit (u_n) une suite définie à partir d'un certain rang.

- (u_n) est **majorée** s'il existe un réel M tel que, pour tout indice n , $u_n \leq M$;
- (u_n) est **minorée** s'il existe un réel m tel que, pour tout indice n , $u_n \geq m$;
- (u_n) est **bornée** si elle est à la fois majorée et minorée.

Les nombres M et m sont respectivement appelés un **majorant** et un **minorant** de la suite. Ils ne sont pas nécessairement des termes de la suite.

**Montrer qu'une suite est bornée**

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

$$u_n = \frac{n+1}{n+2}.$$

Comme $n+1 > 0$ et $n+2 > 0$, on a $u_n > 0$. De plus,

$$1 - u_n = 1 - \frac{n+1}{n+2} = \frac{1}{n+2} > 0.$$

Ainsi, pour tout $n \in \mathbb{N}$,

$$0 < u_n < 1.$$

La suite (u_n) est donc minorée par 0, majorée par 1, et par conséquent bornée.



- Les suites $(\sin n)$, $(\cos n)$ et $((-1)^n)$ sont minorées par -1 et majorées par 1 .
- La suite définie par $u_n = n$ est minorée par 0 , mais elle n'est pas majorée.
- Toute suite croissante est minorée par son premier terme ; toute suite décroissante est majorée par son premier terme.

Pour étudier le sens de variation d'une suite, on peut calculer

- si $u_{n+1} - u_n \geq 0$, la suite est croissante ;
- si $u_{n+1} - u_n \leq 0$, la suite est décroissante.

- Étape 1 : pour n entier naturel, on exprime $u_{n+1} - u_n$ en fonction de n en FACTORISANT le plus possible ;
- Étape 2 : on étudie le signe de chaque facteur, deux cas généralement :
 - Cas 1 : si la suite est définie explicitement, c'est souvent simple car n est un entier naturel ;
 - Cas 2 : si on a la suite définie par sa relation de récurrence, on a besoin d'une aide dans l'énoncé, soit un majorant, soit un minorant de u_n ;
- Étape 3 : on en déduit le signe de $u_{n+1} - u_n$ et donc la monotonie de la suite.



On considère la suite (u_n) définie par

On souhaite étudier le sens de variation de la suite (u_n) . On admet ici que la suite (u_n) est majorée par 2.

[illegible]

V Calculatrice : générer et représenter une suite



Ce que la calculatrice permet de faire

Pour une suite définie explicitement ou par récurrence, la calculatrice permet notamment :

- de calculer rapidement plusieurs termes ;
- d'afficher un tableau de valeurs ;
- de représenter les points $(n; u_n)$;
- de conjecturer un sens de variation ou un comportement à long terme.

Attention : un tableau ou un graphique permet de conjecturer une propriété ; il ne remplace pas une démonstration valable pour tout entier n .



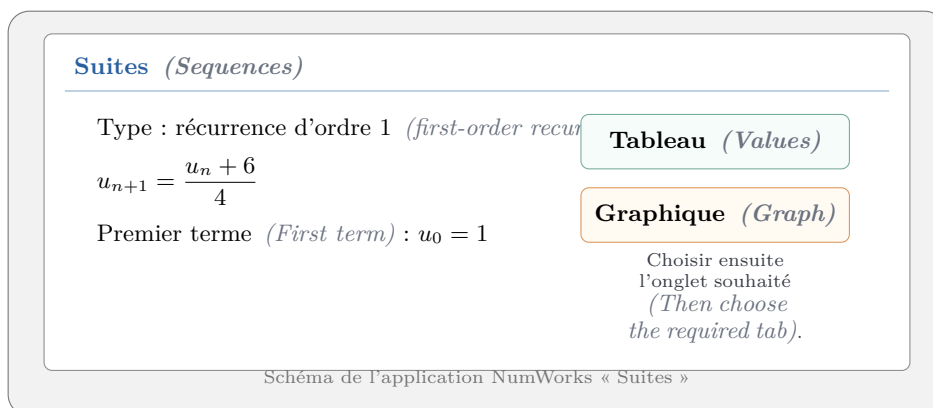
Exemple utilisé sur les deux calculatrices

On considère la suite (u_n) définie par

$$u_0 = 1 \quad \text{et} \quad u_{n+1} = \frac{u_n + 6}{4}.$$

On souhaite afficher ses premiers termes et sa représentation graphique.

V.1 Avec une NumWorks



1. Ouvrir l'application **Suites** (*Sequences*), puis choisir **Ajouter une suite** (*Add a sequence*).
2. Choisir **Récurrence d'ordre 1** (*First-order recurrence*).
3. Saisir la relation de récurrence, puis $u_0 = 1$ dans **Premier terme** (*First term*).
4. Ouvrir l'onglet **Tableau** (*Values*) pour lire les valeurs de u_n , ou l'onglet **Graphique** (*Graph*) pour représenter les points $(n; u_n)$.
5. Pour une suite de la forme $u_{n+1} = f(u_n)$, le menu de la suite permet aussi d'afficher un **graphique en escalier** (*cobweb plot*).

V.2 Avec une TI-Nspire CX CAS

Graphiques (*Graphs*)

Menu > Entrée/Modification graphique (*Graph Entry/Edit*)

> Suite (*Sequence*) > Suite (*Sequence*)

$$u1(n) = \frac{u1(n-1) + 6}{4}$$

Termes initiaux (*Initial terms*) : 1

$$1 \leq n \leq 99 \quad nstep = 1$$

Ctrl+T
affiche le tableau
(*displays the table*)

Schéma du mode « Suite » de la TI-Nspire CX CAS

Sur la TI-Nspire CX CAS, la relation est saisie avec le terme précédent : la formule $u_{n+1} = \frac{u_n + 6}{4}$ devient donc

$$u1(n) = \frac{u1(n-1) + 6}{4}.$$

1. Ouvrir une page **Graphiques** (*Graphs*).
2. Choisir successivement :

Menu > **Entrée/Modification graphique** (*Graph Entry/Edit*)
> **Suite** (*Sequence*) > **Suite** (*Sequence*).

3. Saisir la relation dans la ligne **u1(n)=**, puis saisir 1 dans **Termes initiaux** (*Initial terms*).
4. Valider avec **Entrée** (*Enter*) pour obtenir la représentation graphique.
5. Utiliser **Ctrl+T** pour afficher ou masquer le **tableau des valeurs** (*table of values*).



Indice du premier terme

Les écritures affichées par les calculatrices ne sont pas toujours exactement celles de l'énoncé. Il faut surtout vérifier avec soin **l'indice du premier terme** et la manière dont la machine note le **terme précédent**.

VI Python : calculer des termes et chercher un seuil

VI.1 Calculer les termes d'une suite récurrente



Boucle for (for loop)

Lorsque le nombre d'étapes est connu, une boucle **for** permet de calculer les termes successifs. Par exemple on considère la suite (u_n) définie par :

$$\begin{cases} u_0 = 5 \\ u_{n+1} = 1,2u_n + 3 \end{cases}$$

On cherche une fonction qui renvoie le terme d'indice N (donc le $(N + 1)$ ème terme de la suite puisqu'elle commence à u_0) :

```
def terme_u(N):
    u = 5
    for n in range(N):
        u = 1.2*u + 3
    return u

print(terme_u(8))
```

Après huit passages dans la boucle, la variable u contient le terme u_8 : le programme affiche donc u_8 .

VI.2 Rechercher un seuil

Définition 7 (Seuil)

Un **seuil** (*threshold*) est une valeur que l'on souhaite atteindre ou dépasser. Une boucle **while** est adaptée lorsque l'on ne connaît pas à l'avance le nombre d'itérations nécessaires.



Premier rang dépassant 1000

On considère $u_0 = 100$ et $u_{n+1} = 1,08u_n$. On cherche le premier rang n pour lequel $u_n > 1000$.

```
def seuil_u(S):
    u = 100
    n = 0
    while u <= S:
        u = 1.08*u
        n = n + 1
    return n

print(seuil_u(1000))
```

Le programme renvoie le premier rang n pour lequel $u_n > 1000$.



À savoir lire et écrire

Dans ce chapitre, on doit être capable de comprendre ou d'écrire des algorithmes simples permettant de calculer des termes d'une suite ou de rechercher un seuil.

VII

Première intuition de la notion de limite



Observer le comportement pour de grands indices

En Première, on ne donne pas de définition formelle de la limite d’une suite. On cherche seulement à **conjecturer** son comportement lorsque n devient très grand, à partir de calculs, de graphiques ou d’algorithmes.



Vers une limite finie

$$u_n = 2 + \frac{1}{n} \quad (n \geq 1).$$

.....

.....

.....

.....

.....



Vers $+\infty$

$$v_n = n^2.$$

.....

.....

.....

.....

.....



Pas de limite

$$w_n = (-1)^n.$$

.....

.....

.....

.....

.....



Le rôle de Python

Un programme de seuil permet d’explorer numériquement une conjecture de limite. Il ne constitue pas, à lui seul, une démonstration mathématique.

VIII Bilan – Suites : généralités

1. Une suite numérique

$$u : n \mapsto u(n), \quad u_n = u(n).$$

- u_n : terme d'indice n (*term of index n*);
- bien repérer le rang initial.

2. Générer une suite

- forme explicite : $u_n = f(n)$;
- récurrence : $u_{n+1} = f(u_n)$ avec un terme initial;
- algorithme / liste Python;
- motif géométrique ou combinatoire.

3. Représenter une suite

$$M_n(n; u_n)$$

La représentation est un ensemble de points isolés.

4. Étudier les variations Une méthode classique consiste à étudier le signe de

$$u_{n+1} - u_n.$$

- $u_{n+1} - u_n \geq 0$: suite croissante;
- $u_{n+1} - u_n \leq 0$: suite décroissante.

Une suite est **bornée** lorsqu'elle possède à la fois un mino- rant et un majorant.

5. Python

- liste : [...];
- **range**(N) produit les entiers de 0 à $N - 1$;
- **append** : ajouter un terme;
- **for** : nombre d'étapes connu;
- **while** : nombre d'étapes inconnu, notamment pour un seuil.

6. Limite : intuition seulement Pour de grands indices, on peut conjecturer :

- une limite finie;
- une croissance vers $+\infty$ ou $-\infty$;
- une absence de limite.

Les trois algorithmes-types à connaître

Dans les trois exemples suivants, la suite est définie par

$$u_0 = 5 \quad \text{et} \quad u_{n+1} = 1,2u_n + 3.$$

1. Calculer un terme u_N

```
def terme_u(N):
    u = 5
    for n in range(N):
        u = 1.2*u + 3
    return u
```

La boucle est répétée N fois : on passe de u_0 à u_N .

2. Obtenir les termes de u_0 à u_N

```
def termes_u(N):
    u = 5
    L = [u]
    for n in range(N):
        u = 1.2*u + 3
        L.append(u)
    return L
```

La liste renvoyée contient $N + 1$ valeurs : $u_0, u_1, \dots, u_N$.

3. Chercher un seuil

```
def seuil(S):
    u = 5
    n = 0
    while u < S:
        u = 1.2*u + 3
        n = n + 1
    return n
```

La fonction renvoie le premier rang n tel que $u_n \geq S$.

Bien comprendre range

$$\text{list}(\text{range}(5)) = [0, 1, 2, 3, 4].$$

- **range**(N) contient N entiers : $0, 1, \dots, N - 1$;
- **range**(1, N+1) contient les entiers $1, 2, \dots, N$;
- dans une boucle qui calcule une suite récurrente, ce qui compte est le nombre de passages dans la boucle : N passages permettent de passer de u_0 à u_N .

Réflexes

- repérer le rang de départ ;
- distinguer explicite et récurrence ;
- ne pas confondre u_n et u_{n+1} ;
- pour les variations, comparer deux termes consécutifs.

Lexique essentiel

- suite : (*sequence*) ; terme : (*term*) ;
- indice : (*index*) ;
- relation de récurrence : (*recurrence relation*) ;
- croissante : (*increasing*) ; décroissante : (*decreasing*) ;
- seuil : (*threshold*).

IX Annexe : Représentation en escalier d'une suite récurrente

Dans le cas d'une suite définie par une relation de récurrence de la forme

$$u_{n+1} = f(u_n),$$

où u_0 est donné, on peut représenter les premiers termes de la suite sur l'axe des abscisses en utilisant :

- la courbe représentative $\mathcal{C}_f$ de la fonction f ;
- la droite Δ d'équation $y = x$.

On considère la suite (u_n) définie par

$$u_0 = 15 \quad \text{et} \quad u_{n+1} = 1 + \frac{20}{u_n} \quad \text{pour tout } n \in \mathbb{N}.$$

On introduit la fonction f définie pour $x \neq 0$ par

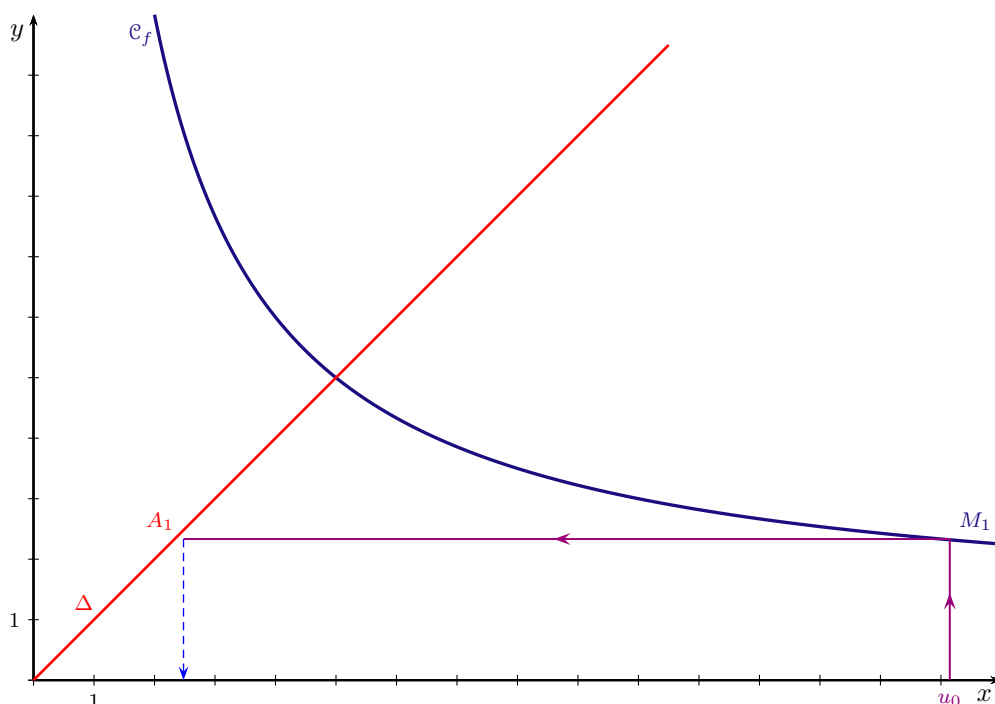
$$f(x) = 1 + \frac{20}{x}.$$

La relation de récurrence s'écrit alors

$$\boxed{u_{n+1} = f(u_n)}.$$

1. Calculer des valeurs approchées au dixième des termes u_1 , u_2 , u_3 et u_4 .

2. On place le terme initial u_0 sur l'axe des abscisses.
3. Comme $u_1 = f(u_0)$, le nombre u_1 est l'ordonnée du point M_1 de $\mathcal{C}_f$ d'abscisse u_0 .
On reporte ensuite la valeur de u_1 sur l'axe des abscisses à l'aide de la droite Δ d'équation $y = x$.
4. Le point A_1 de la droite Δ , de même ordonnée que M_1 , a pour coordonnées $A_1(u_1 ; u_1)$.
5. On recommence le même procédé à partir de u_1 pour obtenir u_2 , puis à partir de u_2 pour obtenir u_3 , et ainsi de suite.



Poursuivre sur le graphique la construction des termes u_2 , u_3 , u_4 , u_5 et u_6 , puis décrire le comportement que l'on peut conjecturer.

.....

.....

.....

.....

.....

↵ **Fin du cours** ↶