Chapitre 8

Les suites numériques réelles

Que l'on se place d'un point de vue pratique ou d'un point de vue théorique, la **notion de suite est fondamentale**. Toutes les autres notions d'analyse réelle peuvent d'ailleurs s'en déduire.
Dans ce chapitre, vous trouverez beaucoup d'exercices résolus qui seront un guide pour le travail personnel.

8.1 Introduction

8.1.1 Définition

Soit E un ensemble quelconque.
On appelle suite d'éléments de E, une application f de $I \subset \mathbb{N}$ dans E

$$\begin{cases} f : I & \xrightarrow{f} & E \\ n & \xmapsto{f} & f(n) \end{cases}$$

Remarque 1 :

1. I peut être un ensemble fini ou infini.
   — Si I est un ensemble fini, **la suite est dite finie**.
   — Si I est un ensemble infini, la suite est dite infinie
2. On préfère utiliser la notation indicielle dans le cas des suites.
   On pose donc $f(n) = x_n$ où x_n est un élément tel que $x_n \in E$; $(x_n)_{n\in\mathbb{N}}$ représente l'application *(ou la suite)* f
3. On peut dire *(et on le dit ! !)* que x_n est le terme d'indice n de la suite $(x_n)_{n\in\mathbb{N}}$
4. Si $E = \mathbb{R}$, on parle alors de **suites numériques réelles**, et c'est ce type de suite que nous étudions dans ce chapitre
5. On peut aussi remarquer que si la suite est infinie, l'ensemble des valeurs prises par la suite de terme général $u_n = (-1)^n$, $w_n = \cos\left(\frac{\pi}{6} + n\frac{\pi}{2}\right)$ sont des exemples de suites numériques infinies qui ne prennent qu'un nombre fini de valeurs
6. $\left(v_n = \dfrac{n-1}{n^2+1}\right)$ pour $1 \leqslant n \leqslant 6$ est l'exemple *(trop)* simple d'une suite finie

8.1.2 Comment définit-on une suite ?

Il y a plusieurs façons de définir une suite :

1. Par une formule explicite, en fonction de l'entier n ;

   Exemples :

— $u_n = \sin\dfrac{1}{n}$, suite qui n'est définie que pour $n \in \mathbb{N}^*$; nous avons, en particulier, $u_1 = \sin 1$, $u_2 = \sin\dfrac{1}{2}$

— $v_n = \pi 2^{-n}$; nous avons $v_0 = \pi$, $v_1 = \dfrac{\pi}{2}$, $v_2 = \dfrac{\pi}{4}$

2. Par une formule itérative *(Formule de récurrence)*
Par exemple, la **suite de FIBONACCI**.

$$\begin{cases} F_{n+2} = F_{n+1} + F_n \\ F_0 = F_1 = 1 \end{cases}$$

Le tableau suivant donne les premières valeurs de cette suite :

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
1	1	2	3	5	8	13	21	34	55

TABLE 8.1 – Le calcul des premiers termes de la suite de Fibonacci $F_{n+2} = F_{n+1} + F_n$ avec $F_0 = F_1 = 1$

On peut utiliser un algorithme simple pour le calcul des valeurs de la suite de Fibonacci :
Algorithme donnant les termes successifs de la suite de Fibonnacci
(a) En `TestAlgo`

```
algo Fibonnacci
#Cet algorithme donne les N premiers termes d'une suite de Fibonnacci
#en langage de description TestAlgo
    principal
        var reel a, b,c;
            entier N, i;
    debut
         saisir("Donner le premier terme de la suite", @a);
         saisir("Donner le second  terme de la suite", @b);
         saisir("jusqu'à quel ordre souhaitez vous afficher les termes de la suite?",@N);
         i:=2;
          tantque (i<=N)
        c:=a+b;
        afficher("Le terme d'ordre "+i+" de la suite de Fibonnacci est " +c);
        alaligne();
        a:=b;
        b:=c;
        i:=i+1;
         fintantque
     fin
```

(b) Codage d'une suite de Fibonnacci en langage Python :

```
#On définit d'abord le calcul du terme d'ordre n
def fibonacci(n):
    a=b=1  #affectation typique du langage Python
        for i in range(n):
                a, b = a+b, b #Autre affectation typique du langage Python
        return b
#On imprime les termes de la suites de 1 à 120
for n in range(120):
        print (fibonacci(n))
```

3. On peut aussi définir une suite par $u_0 = a$ et $u_{n+1} = f(u_n)$ où f est une fonction numérique d'une variable réelle
Exemple : On considère la suite numérique définie par :

$$\begin{cases} u_0 = 5 \\ u_{n+1} = \dfrac{1}{2}\left(u_n + \dfrac{3}{u_n}\right) \end{cases}$$

qui est une suite numérique qui converge $\sqrt{3}$; cette suite est connue comme **l'algorithme des babyloniens**, qui permet d'approximer [1] la racine carrée du nombre $\sqrt{3}$. Elle est définie à partir

1. Ou approcher

de la fonction : $f(x) = \frac{1}{2}\left(x + \frac{3}{x}\right)$ *(cf figure 8.1)*[2] C'est une suite qui converge très rapidement.

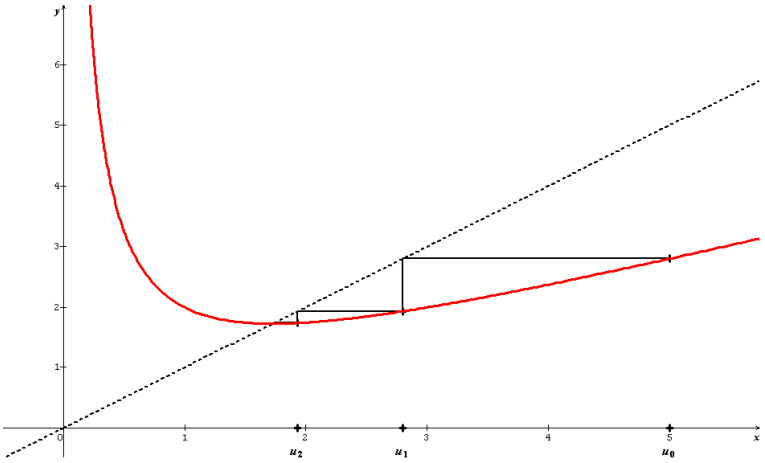

FIGURE 8.1 – La visualisation de la suite $u_{n+1} = \frac{1}{2}\left(u_n + \frac{3}{u_n}\right)$

Le tableau suivant donne les premières valeurs de la suite :

u_0	u_1	u_2	u_3	u_4	u_5	u_6
5	2,8	1,97571	1,74276	1,73208	1,73205	1,73205

TABLE 8.2 – Le calcul des premiers termes de la suite des babyloniens $u_{n+1} = \frac{1}{2}\left(u_n + \frac{3}{u_n}\right)$ avec $u_0 = 5$

Algorithmes donnant les termes successifs de la suite des babyloniens

(a) Dans le langage `TestAlgo`

```
algo Babylonien
#Cet algorithme recherche la racine carrée de 3 par l'algorithme des Babyloniens

principal
var reel a, b,  epsilon;
debut

saisir("Donner le premier terme de la suite", @a);
saisir("Donner l'erreur admissible",@epsilon);
b:=effe(a);
 tantque  ( abs(b-a)>epsilon)
 faire
a:=b;
b:=effe(a);
 fintantque
afficher("La racine de 3  à "+epsilon+" près est" +b);
fin

#La fonction valeur absolue n'existe pas dans test algo; elle est donc créée ici
   fonction abs(reel x): reel var
   reel c;
   debut
    si (x>=0) alors c:=x; sinon c:=-x; finsi
    retourne c;
   fin

#On crée ici, la fonction qui définit l'algorithme
```

2. En fait, on peut trouver une approximation de $\sqrt{A}$ avec $A > 0$ en utilisant la suite $U_{n+1} = \frac{1}{2}\left(U_n + \frac{A}{U_n}\right)$

```
fonction effe(reel x): reel var
      reel c;
      debut
      c:=1/2*(x+(3/x));
      retourne c;
fin
```

(b) Dans le langage Python

```
# Ce script définit une fonction babylon dans laquelle on insère 2 variables:
# Une variable A dont on veut déterminer la racine carrée
# Une variable n qui donnera lerme de rang n de la suite des babyloniens
def babylon(A,n):
    if A <= 0:
        A=-A
    A = float(A) #On force A à avoir le type flottant
    n = int(n) #On force n à avoir le type entier
    B = A+2 #C'est notre premier terme, totalement arbitraire plus grand que A
    for i in range(n):
        B = 0.5*(B+(A/B))
    return B
```

Nous obtenir comme valeurs : $babylon\,(144,5) = 12.11935\ldots$ et $babylon\,(144,30) = 12.0$

4. Il est aussi possible de définir les termes successifs d'une suite **par un algorithme** ; c'est le cas de la célèbre **conjecture de Syracuse**

   La conjecture de Syracuse est une conjecture mathématique qui reste improuvée à ce jour et qui est définie de manère suivante :

   Soit $n \in \mathbb{N}$

   Si n est pair, alors le terme suivant est obtenu en divisant n par 2

   Si n est impair, alors le terme suivant est obtenu en multipliant n par 3 et on lui ajoute 1

   En répétant cette procédure, la suite nombre atteint la valeur 1, puis se prolonge indéfiniment par une suite de 3 valeurs triviales, appelée « cycle trivial »

   Jusqu'à présent, la conjecture de Syracuse selon laquelle, depuis n'importe quel entier positif de départ, la suite de Syracuse atteint 1, n'a pas été mise en défaut.

   Programme Python, calculant les n premiers termes d'une suite de Syracuse commençant par un premier terme noté p

```
def syracuse(p,n):
    for i in range(n):
        if p%2 == 0: #On teste si p est pair
            p =p/2
        else:
            p = 3*p+1
    print("Le terme de la suite d'ordre {} est {}").format(i+1,p)
```

   Calculez $syracuse\,(1,12)$ ou $syracuse\,(12,20)$

Exercice 1 :

Que pouvez vous dire de la suite définie pour tout $n \in \mathbb{N}$ par : $u_n = \dfrac{n}{2n+1}\left(1+(-1)^n\right)$

Résolution

Pour le moment, il est difficile d'en dire beaucoup sur cette suite. Dans un premier temps, ce qu'il nous est possible de dire, c'est ceci :

— Si n est impair, alors, $u_n = 0$

— Si n est pair, alors $u_n = \dfrac{2n}{2n+1}$

Il nous est possible de faire un tableau des premières valeurs : On voit donc, facilement, un comportement différent suivant que l'indice des termes est pair ou impair. On peut donc créer ainsi 2 « **sous-suites** » ou « **suites extraites** » :

— La sous-suite des termes de rang pair dont les termes se rapprochent de 1 ; elle admet +1 comme limite.

— La sous-suite des termes de rang impair, toujours nulle

— En fait, cette suite n'admet aucune limite.

$n=$	0	1	2	3	4	5	6	7	8	9	10
$u_n=$	0	0	$\frac{4}{5}$	0	$\frac{8}{9}$	0	$\frac{12}{13}$	0	$\frac{16}{17}$	0	$\frac{20}{21}$

TABLE 8.3 – Calcul des premières valeurs de la suite $u_n = \dfrac{n}{2n+1}\left(1+(-1)^n\right)$

8.1.3 Définition

1. **Une suite est dite croissante si,** $\forall n \in \mathbb{N}\ u_{n+1} \geqslant u_n$
2. **Une suite est dite décroissante si,** $\forall n \in \mathbb{N}\ u_{n+1} \leqslant u_n$
3. **Une suite est dite monotone, si elle est toujours croissante ou toujours décroissante**

Exercice 2 :

1. Que dire de la suite de terme général $u_n = \dfrac{10^n}{n!}$?

   (a) Pour étudier la croissance ou la décroissance de cette suite, il est possible de faire le rapport $\frac{u_{n+1}}{u_n}$, rapport que l'on compare à 1 ; ici,

   $$\frac{u_{n+1}}{u_n} = \frac{10^{n+1}}{(n+1)!} \times \frac{n!}{10^n} = \frac{10}{n+1}$$

   Ainsi :

   $$\begin{cases} \dfrac{10}{n+1} < 1 & \text{si } n > 9 \\ \dfrac{10}{n+1} > 1 & \text{si } n < 9 \\ \dfrac{10}{n+1} = 1 & \text{si } n = 9 \end{cases}$$

   Ainsi, la suite $(u_n)_{n\in\mathbb{N}}$ est croissante jusque $n=9$, puis, $u_9 = u_{10}$, et la suite $(u_n)_{n\in\mathbb{N}}$ est décroissante à partir de $n=10$

   (b) Bien sur qu'il est aussi possible de faire la différence entre 2 termes consécutifs : $u_{n+1} - u_n$

   $$\begin{cases} u_{n+1} - u_n = & \dfrac{10^{n+1}}{(n+1)!} - \dfrac{10^n}{n!} \\ = & \dfrac{10^n}{n!}\left(\dfrac{10}{n+1} - 1\right) \\ = & \dfrac{10^n}{n!}\left(\dfrac{10-(n+1)}{n+1}\right) \\ = & \dfrac{10^n}{n!}\left(\dfrac{9-n}{n+1}\right) \end{cases}$$

   On retrouve donc $u_{n+1} - u_n > 0 \iff n < 9$ et $u_{n+1} - u_n < 0 \iff n > 9$ et $u_9 = u_{10}$

0	1	2	3	4	5	6	7	8	9	10	11	12
1	10	50	166,67	416,67	833,34	1388,88	1984,13	2480,16	2755,32	2755,32	2505,21	2087.67

TABLE 8.4 – Le calcul des douze premiers termes de la suite $\dfrac{10^n}{n!}$

2. Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_n = \dfrac{n^n}{n!}$; étudier la croissance ou la décroissance de cette suite.

   Correction de l'exercice

   Commençons par faire le rapport $\frac{u_{n+1}}{u_n}$. Nous avons donc :

   $$\frac{u_{n+1}}{u_n} = \frac{(n+1)^{n+1}}{(n+1)!} \times \frac{n!}{n^n}$$

   Après simplification :

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

   Donc, $u_{n+1} > u_n$, et la suite $(u_n)_{n\in\mathbb{N}}$ est donc strictement croissante

3. Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_{n+1} = -u_n^2 + u_n$; étudier la croissance ou la décroissance de cette suite.

   C'est très simple : pour tout $n \in \mathbb{N}$, nous avons : $u_{n+1} - u_n = -u_n^2 \leqslant 0$. Donc, $u_{n+1} \leqslant u_n$ et la suite est bien décroissante