Chapitre 2

Suites

I Exercices

2.1 Exemples de suites

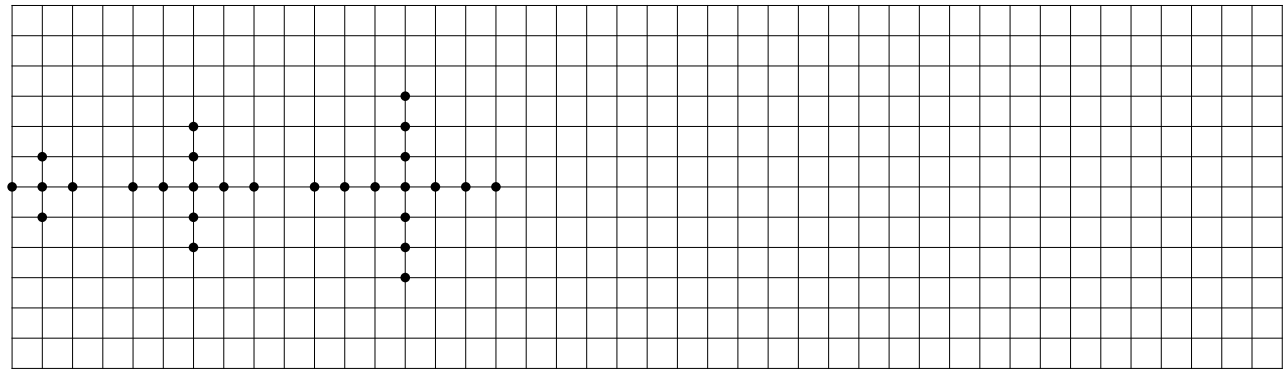
Exercice 2.1

On voit ci-dessous une suite de trois dessins.

1. Tracer les deux dessins suivants.
2. On appelle u_0 le nombre de points du premier dessin, u_1 le nombre de points du deuxième dessin, et ainsi de suite ... ainsi $u_0 = 5$ et $u_1 = 9$.

   Déterminer en fonction de n le nombre de points du dessin numéro n.

$n = 0$ $n = 1$ $n = 2$



Exercice 2.2 (Utilisation de la calculatrice pour une suite explicite)

Voici la procédure à la calculatrice Numworks pour avoir la liste des valeurs successives de la suite de l'exercice précédent.

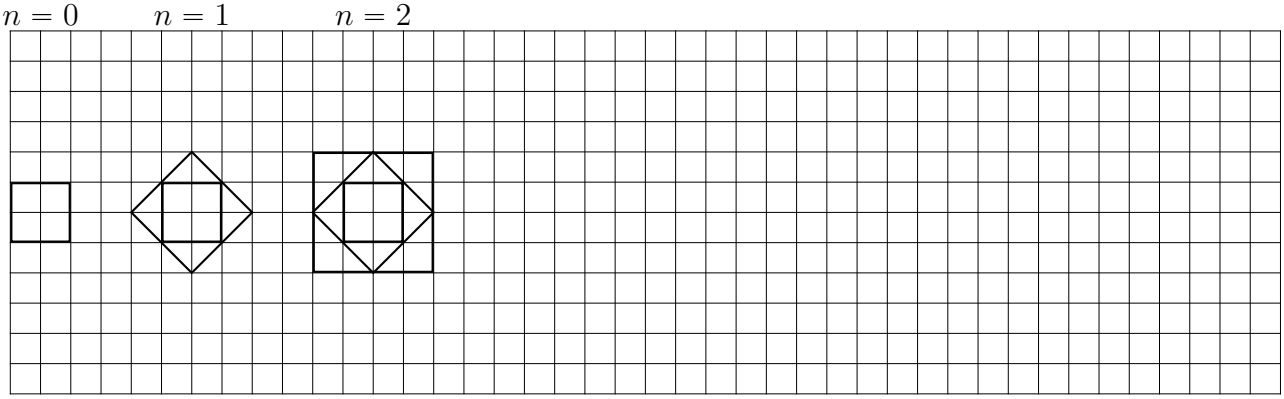
- Aller dans le module Suites
- Aller sur *Ajouter une suite* et valider
- Choisir u_n *Explicite* et valider
- Saisir la formule en fonction de n, valider
- Aller sur l'onglet *Tableau* et valider

Exercice 2.3

On voit ci-dessous une suite de trois dessins.

1. Tracer les deux dessins suivants.
2. On appelle a_0 l'aire du carré du premier dessin en carreaux : $a_0 = 4$, et pour les suivants on appelle a_1, a_2 ...l'aire totale de chaque dessin.
   a) Déterminer les aires a_1, a_2, a_3, a_4
   b) Déterminer a_n en fonction de n.

$n = 0$ $n = 1$ $n = 2$



Exercice 2.4 (Nombres de matchs d'un championnat)

On appelle n le nombre d'équipes participant à un championnat, déterminer la formule du nombre de matchs en fonction de n. On précise que chaque équipe doit jouer contre toutes les autres et qu'il y a un match aller et un match retour.

Exercice 2.5 (Nombres de mots de n lettres)

On utilise n lettres différentes déterminer en fonction de n le nombre de « mots » de n lettres que l'on peut écrire, en utilisant chaque lettre une seule fois dans le « mot ».

Par exemple pour $n = 2$, on peut écrire `AB` ou `BA`, mais pas `AA`.

Exercice 2.6 (Effectifs annuels d'un club (1))

Un club sportif a 80 adhérents. Chaque année, ce club perd un quart de son effectif et gagne 50 nouveaux adhérents. On appelle u_0 l'effectif de départ : $u_0 = 80$, et u_1, u_2 etc. les effectifs des années suivantes.

1. Calculer u_1 et u_2.
2. On appelle u_n l'effectif de l'année n, et u_{n+1} l'effectif de l'année suivante $n + 1$.
   Écrire u_{n+1} en fonction de u_n.

Exercice 2.7 (Utilisation de la calculatrice pour une suite définie par récurrence)

Voici la procédure à la calculatrice Numworks pour avoir la liste des valeurs successives de la suite de l'exercice 2.6.

- Aller dans le module Suites
- Aller sur *Ajouter une suite* et valider
- Choisir u_{n+1} *Récurrente d'ordre 1* et valider
- Saisir la formule en fonction de u_n, valider
- Saisir la valeur de u_0
- Aller sur l'onglet *Tableau* et valider

Exercice 2.8 (Effectifs annuels d'un club (2))

On continue avec la suite de l'exercice 2.6.

1. Utiliser la calculatrice pour répondre à ces deux questions.
   a) Quel semble être le sens de variations de la suite (u_n) ?
   b) Lorsque n devient grand, les termes successifs semblent-ils s'approcher d'une limite ? Si oui, combien ?
2. Interpréter concrètement les réponses aux deux questions précédentes.

Exercice 2.9 (La suite de Héron)

Héron d'Alexandrie est un ingénieur, mécanicien, mathématicien grec du premier siècle après Jésus-Christ.

Il cherche une valeur approchée de $\sqrt{2}$ sous forme de fraction. Il traduit cela sous la forme géométrique : il cherche à tracer un carré de côté c dont l'aire est égale à 2, ainsi $c^2 = 2$ et $c = \sqrt{2}$.

Il commence par tracer un rectangle R_0 de longueur 2 et de largeur 1, ainsi l'aire du rectangle est bien égale à $2 : 2 \times 1 = 2$. Mais ce n'est pas un carré. On appelle u_0 cette première longueur. Voir la figure plus bas.

Pour avoir un rectangle d'une forme plus proche d'un carré, il prend comme nouvelle longueur la moyenne de 2 et 1 : $\dfrac{2+1}{2} = \dfrac{3}{2}$, ainsi : $u_1 = \dfrac{3}{2}$.

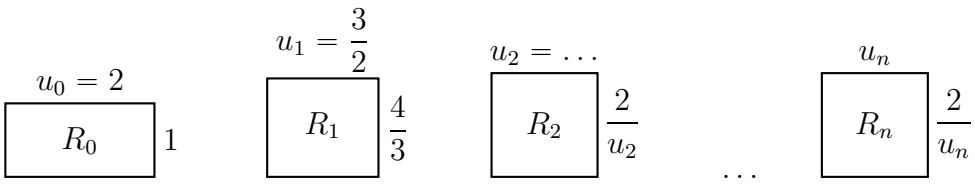
Il calcule alors la largeur pour que l'aire soit égale à 2 : largeur $= \dfrac{2}{\frac{3}{2}} = 2 \times \dfrac{2}{3} = \dfrac{4}{3}$

Et on a bien : $\dfrac{3}{2} \times \dfrac{4}{3} = \dfrac{12}{6} = 2$

1. Calculer u_2 la moyenne de $\dfrac{3}{2}$ et $\dfrac{4}{3}$, sous forme de fraction.
2. On continue ainsi et à l'étape n, on obtient un rectangle R_n de longueur u_n et de largeur $\dfrac{2}{u_n}$.
   Justifier que u_{n+1}, qui est la moyenne de u_n et de $\dfrac{2}{u_n}$ s'écrit : $u_{n+1} = \dfrac{u_n}{2} + \dfrac{1}{u_n}$
3. Saisir alors cette suite dans le module Suite de la calculatrice, et compléter ci-dessous à 10^{-6} près.

   $u_0 =$ ..... $u_1 =$ ..... $u_2 \approx$ .....

   $u_3 \approx$ ..... $u_4 \approx$ ..... $u_5 \approx$ .....
4. Comparons avec la valeur de $\sqrt{2}$ donnée par la calculatrice.
   a) Compléter à 10^{-6} près : $\sqrt{2} \approx$ .....
   b) Après combien d'étapes, la suite de Héron arrive-t-elle au même résultat ?



2.2 Suites définies explicitement ou par récurrence

Exercice 2.10

1. Une suite est définie par $u_n = n^2 + 5n$. Calculer u_0, u_1, u_2.
2. Une suite est définie par $u(n) = \dfrac{1}{2n+1}$. Calculer $u(0)$, $u(1)$, $u(2)$.

Exercice 2.11

1. Définir par une formule en fonction de n
   a) la suite des nombres impairs ;
   b) la suite des multiples de 5 ;
   c) la suite : 1 0,1 0,01 0,001 0,000 1 etc.
   d) la suite : 1 $\dfrac{1}{4}$ $\dfrac{1}{9}$ $\dfrac{1}{16}$ etc.
2. Définir par une formule **de récurrence**
   a) la suite des nombres impairs ;
   b) la suite des multiples de 5 ;
   c) la suite : 1 0,1 0,01 0,001 0,000 1 etc.

Exercice 2.12

1. Une suite est définie par $u_{n+1} = 2u_n + 1$ et $u_0 = 3$.
   Calculer u_1, u_2, u_3.
2. Une suite est définie par $u(n+1) = 2 + \dfrac{1}{u(n)}$ et $u(0) = 1$.
   Calculer $u(1)$, $u(2)$, $u(3)$ sous forme de fractions.

Exercice 2.13

On appelle $u(n)$ le nombre de segments qui relient n points du plan, pour $n \geqslant 2$

1. Déterminer $u(2)$, $u(3)$, $u(4)$.
2. Déterminer la relation de récurrence c'est à dire la relation entre $u(n)$ et $u(n+1)$.
   Indication : S'il y a n points, le nombre de segments est $u(n)$. On ajoute alors un point, il y a donc $n+1$ points. En rajoutant ce point, combien de segments doit on rajouter ?
3. Utiliser la calculatrice pour déterminer le nombre de segments pour 50 points.

Exercice 2.14

Modéliser par une suite les situations suivantes.

1. On part de 7, et on ajoute 10 à chaque étape.
2. On part de 1 et on double à chaque étape.
3. La suite $(u(n))$ est définie par $u(0) = 5$ et ensuite on calcule chaque terme suivant en multipliant le terme précédent par 4 et en soustrayant 3.
4. Un site Internet a 500 abonnés, et chaque mois le nombre d'abonnés est multiplié par 0,8 et 110 nouveaux abonnés arrivent.

Exercice 2.15

Dans l'algorithme ci-dessous, les variables k et n sont des entiers naturels et u est un réel.

```
u ← 6
Pour k = 1 jusqu'à k = n
    u ← 3u − 4
Fin du Pour
```

1. Exécuter cet algorithme pour $n = 3$ en détaillant tous les calculs.
2. Les valeurs successives de la variable u sont les termes d'une suite (u_n).
   Donner la définition de cette suite.
3. Pour un entier naturel n, et pour cette suite (u_n), que représente exactement la valeur finale de la variable u ?

Exercice 2.16

Le script Python ci-dessous correspond à l'algorithme de l'exercice 2.15 et il retourne la valeur finale de la variable u.

```
def sr(n):
  u=6
  for k in range (1,n+1):
    u=3*u-4
  return u
```

1. Saisir ce script et le nommer par exemple `suitrec`.
2. Exécuter ce script pour $n = 3$, et vérifier avec les résultats de l'exercice 2.15.
3. Exécuter ce script pour $n = 55$. Il n'est pas demandé de recopier le résultat !

2.3 Sens de variation

Avant de traiter les exercices qui suivent, étudier, à partir de la page 42, la définition 2.3, les propriétés 2.1 et 2.2, ainsi que les exemples 2.3 et 2.4.

Exercice 2.17

La suite $(u(n))$ est définie par $u(n) = 5n + 2$.

1. Conjecturer le sens de variation de cette suite, en calculant les premiers termes, ou en utilisant la calculatrice (graphique ou tableau ou les deux).
2. Démontrer sa réponse à la question précédente.

Exercice 2.18

Même exercice que l'exercice 2.17, avec la suite $(u(n))$ définie, pour tout entier naturel non nul, par $u(n) = n^2 + 3n$.

Exercice 2.19

Même exercice que l'exercice 2.17, avec la suite $(u(n))$ définie, par $u(n) = \dfrac{4}{n+1}$.

Exercice 2.20

Même exercice que l'exercice 2.17, avec la suite $(u(n))$ définie par $u(n) = n - n^2$.

Exercice 2.21

La suite $(u(n))$ est définie par $u(1) = 1$ et pour tout entier naturel n, $u(n+1) = u(n) + \dfrac{1}{n+1}$.

1. Calculer $u(2)$, $u(3)$, $u(4)$ en arrondissant à 10^{-3} près.
2. Conjecturer le sens de variation de cette suite.
3. Démontrer le sens de variation de cette suite.

2.4 Limites

Avant de traiter les exercices suivants, lire dans le cours le paragraphe 2.3 page 42.

Exercice 2.22

Pour chacune des suites définies ci-dessous, conjecturer si
- cette suite a une limite finie, dans ce cas, quelle limite ?
- cette suite a une limite égale à $+\infty$ ou à $-\infty$;
- cette suite n'a pas de limite.

1. $u_n = \dfrac{4n-1}{n^2}$ **2.** $u_n = \dfrac{n^3}{1-n}$ pour $n \geqslant 2$ **3.** $u_n = (-2)^n$ **4.** $u_n = \dfrac{6n+1}{n}$ **5.** $u_n = 1,2^n$

Exercice 2.23

La suite (u_n) est définie par $u_n = n^3 + n$. On admet que cette suite est croissante et tend vers $+\infty$.

1. Exécuter le script Python ci-dessous pour $A = 100$ en détaillant les calculs.
2. Quelle est la valeur de n retournée en fin d'exécution ?
3. Que signifie cette valeur de n ?
4. Saisir ce script Python, qu'on pourra nommer `suiteseuil`.
5. L'exécuter pour $A = 100$ et vérifier la réponse de la question **2.**
6. Exécuter ce script pour $A = 10^8$.
7. Écrire l'algorithme, en français, dans la partie droite du tableau.

Fonction Python	Algorithme
`def seuil(A):`	
`  n=0`	.....
`  u=0`	.....
`  while(u<A):`	.....
`    n=n+1`	.....
`    u=n**3+n`	.....
	.....
`  return n`	

2.5 Liste des termes d'une suite en Python

Exercice 2.24

La suite (u_n) est définie par $u_n = n^2$, et on veut la liste des termes u_1, u_2, u_3, u_4.

Aller dans le module Python, aller sur *Console d'exécution* et valider.

Saisir ensuite ceci : `[i**2 for i in range(1,5)]`

1. La suite (u_n) est définie par $u_n = 2^n$. Que faut-il saisir dans la console pour avoir la liste des termes de u_0 à u_7 ?
2. La suite (u_n) est définie par $u_n = 10n + 4$. Que faut-il saisir dans la console pour avoir la liste des termes de u_2 à u_9 ?

Exercice 2.25

La suite (u_n) est définie par $u_0 = 10$ et $u_{n+1} = 2u_n + 5$, et on veut la liste des termes u_0, u_1, u_2, u_3, u_4.

La méthode de l'exercice 2.24 n'est pas applicable parce qu'on a ici une suite définie par récurrence.

Le script à saisir est le suivant :

```
def lt(n):
  u=10
  liste=[10]                la liste ne contient que u0
  for k in range(1,n+1):    Pour k = 1 jusqu'à k = n
    u=2*u+5                   calcul du terme suivant de la suite
    liste.append(u)           on ajoute ce terme à la liste
  return liste              on renvoie la liste
```

Pour obtenir la liste voulue on saisit à la console d'exécution : `lt(4)`
et on obtient normalement : `[10, 25, 55, 115, 235]`

Exercice 2.26

La suite (u_n) est définie par $u_0 = 4$ et $u_{n+1} = 3u_n - 10$.

Modifier le script de l'exercice 2.25 pour obtenir la liste des termes u_0, u_1, u_2, u_3, u_4, u_5.

2.6 Somme ou produit de termes consécutifs

Exercice 2.27

La suite (u_n) est définie pour tout entier naturel n par $u_n = n^2$.

Dans l'algorithme ci-dessous, les variables sont s, i, n : s est un réel, i et n sont des entiers naturels.

1. Détailler l'exécution de l'algorithme ci-dessous pour $n = 4$.
2. Quelle est la valeur finale de la variable s pour $n = 4$?
3. Pour un entier naturel donné n, que représente la valeur finale de la variable s ?
4. Saisir le script Python correspondant.
   On pourra nommer ce script `suitsom`, et dans ce script, définir une fonction `som`.
5. Vérifier que la fonction `som` donne le bon résultat pour $n = 4$, en saisissant `som(4)` à la console.
6. Recopier cette fonction `som` dans le tableau ci-dessous.
7. Exécuter ce script pour $n = 1\,000$.

Algorithme	**Fonction Python**
	`def som( .....)`
$s \leftarrow 0$	.....
Pour $i = 1$ jusqu'à $i = n$	.....
$\quad s \leftarrow s + i^2$	.....
Fin du Pour	
	`return( .....)`

Exercice 2.28

La suite (u_n) est définie pour $n \geqslant 1$ par $u_n = \dfrac{1}{n}$.

1. Calculer $u_1 + u_2 + u_3$. Donner la valeur exacte sous forme de fraction, et l'arrondi au millième près.
2. Modifier le script de l'exercice 2.27 pour qu'il calcule $u_1 + u_2 + \cdots + u_n$.
3. Exécuter ce script pour $n = 3$ et vérifier avec les résultats du **1.**
4. Exécuter ce script pour qu'il calcule $u_1 + u_2 + \cdots + u_{100}$.

Exercice 2.29 (Factorielle)

Pour un entier naturel n, la factorielle de n est $1 \times 2 \times \cdots \times n$ et on l'écrit $n!$.

Ainsi, on a : $2! = 1 \times 2 = 2$ $\quad 3! = 1 \times 2 \times 3 = 6$ etc.

Modifier l'algorithme et le script de l'exercice 2.27 pour qu'il calcule la factorielle de n.

2.7 Exercices divers

Exercice 2.30 (Suite de Fibonacci)

La suite de Fibonacci propose un modèle de l'évolution de l'effectif d'un élevage de lapins de mois en mois.

L'évolution de l'effectif obéit aux règles ci-dessous.

- Au départ il y a un couple de lapins.
- Un couple de lapins nés un mois donné ne procrée que 2 mois après et donne naissance à un couple de lapereaux.

On appelle u_n le nombre de couples de lapins le mois n.

Ainsi, au départ : $u_0 = 1$.

Après 1 mois, il n'y a pas de naissance : $u_1 = 1$.

Après 2 mois, ce couple procrée, donc : $u_2 = 2$

Après 3 mois, il y a 2 couples, dont un seul procrée, donc : $u_3 = 3$.

1. Calculer u_4, u_5, u_6, u_7.

2. On peut définir cette suite par récurrence, en donnant $u_0 = 1$, $u_1 = 1$ et en donnant une égalité entre u_n, u_{n+1}, et u_{n+2}. Déterminer cette égalité.

3. Saisir un script en Python, nommé `fibonacci.py` qui contienne une fonction `fib` telle que `fib(n)` retourne le terme u_n de la suite de Fibonacci.

4. Utiliser ce script pour calculer u_{60}.

Exercice 2.31 (Suite de Syracuse)

Pour un entier naturel, on définit ainsi sa suite de Syracuse.

Si cet entier naturel est pair, on le divise par 2, s'il est impair, on calcule 3 fois ce nombre plus 1, et on recommence.

1. Partons par exemple du nombre 3.
- 3 est impair, $3 \times 3 + 1 = 10$
- 10 est pair, $\dfrac{10}{2} = 5$
- 5 est impair, ...
- ...

a) Continuer la suite de Syracuse du nombre 3, jusqu'à obtenir 1.

b) Quand on obtient 1, que se passe-t-il ensuite ?

2. Calculer les suites de Syracuse des nombres 5, 6, 7 jusqu'à obtenir 1.

3. Créer un script Python, nommé par exemple `syracuse.py`, qui contienne une fonction sr telle que `sr(n)` retourne la liste de Syracuse de n jusqu'à 1.

Ce problème a été inventé en 1928 par Lothar Collatz, un mathématicien allemand. Un professeur de l'université de Syracuse aux États-Unis l'a aussi fait connaître, ce qui a donné le nom de *suite de Syracuse*.

La conjecture de Syracuse est que toute suite de Syracuse d'un entier aboutit à 1. Cette conjecture n'a pas encore été démontrée.

Exercice 2.32 (Approximation de π)

On considère une suite (P_n) de polygones réguliers inscrits dans un cercle de rayon 1.

P_2 est le carré de la figure 2.1.

P_3 est l'octogone régulier (8 côtés) de la figure 2.2.

P_4 est le polygone régulier à 16 côtés de la figure 2.3.

et ainsi de suite P_5 a 32 côtés, P_6 a 64 côtés, c'est à dire qu'on double le nombre de côtés à chaque étape.

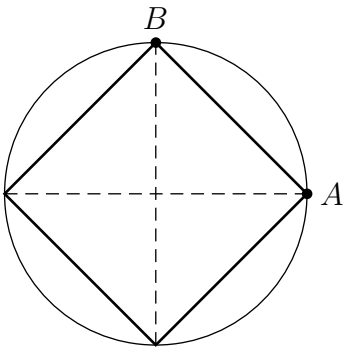


Fig. 2.1

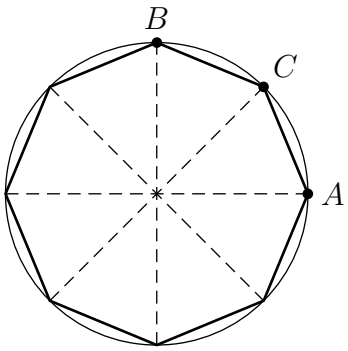


Fig. 2.2

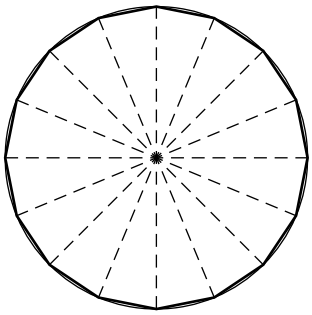


Fig. 2.3

Pour tout entier naturel $n \geqslant 2$,

- on appelle p_n le périmètre du polygone P_n ;
- on appelle c_n la longueur d'un côté du polygone P_n.

1. Expliquer pourquoi la suite (p_n) des périmètres permet d'obtenir des approximations de π.

2. Calculer la valeur exacte de c_2, c'est à dire AB dans la figure 2.1.

3. Calculer la valeur exacte de c_3, c'est à dire AC ou BC dans la figure 2.2.

4. En utilisant la propriété de Pythagore on démontre que pour tout entier naturel $n \geqslant 2$, $c_{n+1} = \sqrt{2 - \sqrt{4 - c_n^2}}$. On admettra cette égalité.

a) Vérifier qu'on retrouve bien la valeur de c_3 avec cette égalité.

b) Écrire l'expression de p_n en fonction de n et de c_n.

c) Créer un programme qui calcule $\dfrac{p_n}{2}$ quand on entre la valeur de n, qu'on pourra appeler `APPROXPI`.

d) La calculatrice affiche une valeur approchée du nombre π avec 9 décimales.
À partir de quel rang obtient-on 4 décimales exactes avec le programme précédent ?

Exercice 2.33 (Approximation de π (2))

La suite (S_n) est définie par $S_0 = \sqrt{12}$ et pour tout entier naturel n, $S_{n+1} = S_n + \sqrt{12}\,\dfrac{(-1)^{n+1}}{(2n+3)3^{n+1}}$.

On admet que la suite (S_n) converge vers π.

Pour répondre aux deux questions ci-dessous, on pourra utiliser le module suite de la calculatrice ou un programme.

À partir de quel rang la suite (S_n) permet-elle d'obtenir

1. 4 décimales exactes ? **2.** 8 décimales exactes ?

II Cours

2.0 Programme

Contenus

- Exemples de modes de génération d'une suite : explicite $u_n = f(n)$, par une relation de récurrence $u_{n+1} = f(u_n)$, par un algorithme, par des motifs géométriques.
  Notations : $u(n)$, u_n, $(u(n))$, (un).
- Sens de variation d'une suite.
- Sur des exemples, introduction intuitive de la notion de limite, finie ou infinie, d'une suite.

Capacités attendues

- Dans le cadre de l'étude d'une suite, utiliser le registre de la langue naturelle, le registre algébrique, le registre graphique, et passer de l'un à l'autre.
- Proposer, modéliser une situation permettant de générer une suite de nombres. Déterminer une relation explicite ou une relation de récurrence pour une suite définie par un motif géométrique, par une question de dénombrement.
- Calculer des termes d'une suite définie explicitement, par récurrence ou par un algorithme.
- Conjecturer, dans des cas simples, la limite éventuelle d'une suite.

Exemples d'algorithme

- Calcul de termes d'une suite, de sommes de termes, de seuil.
- Calcul de factorielle.
- Liste des premiers termes d'une suite : suites de Syracuse, suite de Fibonacci.

Approfondissements possibles

Somme des n premiers carrés, des n premiers cubes.

2.1 Définition et modes de génération d'une suite numérique.

2.1.a Définition et vocabulaire

Rappel : l'ensemble $\mathbb{N}$ est l'ensemble des nombres entiers positifs ou nul, on dit aussi « ensemble des nombres entiers naturels ».

Définition 2.1

Une suite numérique u est une fonction définie sur $\mathbb{N}$. L'image d'un entier naturel n par u est notée $u(n)$ ou u_n.

Vocabulaire

Un nombre n est appelé le *rang* et u_n est appelé *terme* de rang n.

2.1.b Modes de générations d'une suite

Il y deux façons de définir une suite :

- par une relation $u_n = f(n)$, ou relation explicite ;
- par une relation $u_{n+1} = f(u_n)$, ou relation de récurrence.

Exemple 1 (relation $u_n = f(n)$ ou relation explicite)

$u(n) = 50 + 1,5n$ ou $u_n = 50 + 1,5n$

$u_0 = 50 + 1,5 \times 0 = 50$ $u_1 = 50 + 1,5 \times 1 = 51,5$ $u_2 = 50 + 1,5 \times 2 = 53$

Exemple 2 (relation $u_{n+1} = f(u_n)$ ou relation de récurrence) $u_0 = 1$ et $u_{n+1} = 2u_n + 1$

$u_0 = 1$ $u_1 = 2 \times u_0 + 1 = 2 \times 1 + 1 = 3$ $u_2 = 2 \times u_1 + 1 = 2 \times 3 + 1 = 7$

$u_3 = 2 \times u_2 + 1 = 2 \times 7 + 1 = 15$

Définition 2.2

- Une suite où, pour tout entier naturel n, chaque terme u_n est écrit en fonction de n, c'est à dire $u_n = f(n)$, est une suite définie par une **relation explicite**.
- Une suite où chaque terme est défini en fonction du précédent, c'est à dire $u_{n+1} = f(u_n)$ est une suite définie **par une relation de récurrence**.

2.1.c Utilisation du tableur pour afficher les termes d'une suite

Relation explicite $u_n = f(n)$

Exemple 1 : $u_n = 50 + 1,5n$

	A	B
1	n	Un
2	0	50
3	1	51,5
4	2	53

Formule dans la cellule B2 : =50+1,5*A2

Relation par récurrence $u_{n+1} = f(u_n)$

Exemple 2 : $u_0 = 1$ et $u_{n+1} = 2u_n + 1$

	A	B
1	n	Un
2	0	1
3	1	3
4	2	7

Formule dans la cellule B2 : =2*B2+1

2.1.d Exemples de programmes et d'algorithmes permettant de calculer un terme de rang donné.

Exemple 2.1 (Relation explicite $u_n = f(n)$)

Pour la suite définie par : $u_n = 50 + 1,5n$, on a par exemple $u_8 = 50 + 1,5 \times 8 = 62$.

On peut définir la fonction Python nommée `stexp` ci-contre.
Pour calculer u_8, on saisit `stexp(8)` à la console.

```
def stexp(n):
  return(50+1.5*n)
```

On obtient alors l'affichage ci-dessous :

```
>>> stexp(8)
62
```

Exemple 2.2 (Relation par récurrence $u_{n+1} = f(u_n)$)

Pour la suite définie par :
$u_0 = 1$ et $u_{n+1} = 1 + 2u_n$, si on veut par exemple calculer u_4, il faut d'abord calculer u_1, u_2, u_3.
On utilise alors une boucle Pour comme cela est indiqué ci-contre.

Fonction Python	Algorithme

`def strec(n):`
`  u=1`
`  for i in range(1,n+1):`
`  u=2*u+1`
 	
`  return(u)`
	$u \leftarrow 1$
	Pour $k = 1$ jusqu'à $k = n$
	$u \leftarrow 2u + 1$
	Fin du Pour

Pour calculer u_4, on saisit `strec(4)` à la console. On obtient alors l'affichage ci-dessous :

```
>>> strec(4)
31
```

2.2 Sens de variation d'une suite numérique.

2.2.a Définitions et propriétés

Définition 2.3

- Dire qu'une suite u est **croissante** signifie que pour tout entier naturel n, $u_{n+1} \geqslant u_n$;
- Dire qu'une suite u est **décroissante** signifie que pour tout entier naturel n, $u_{n+1} \leqslant u_n$;
- Dire qu'une suite u est **constante** signifie que pour tout entier naturel n, $u_{n+1} = u_n$;

Propriété 2.1

- Si pour tout entier naturel n, $u_{n+1} - u_n \geqslant 0$, alors la suite u est **croissante**.
- Si pour tout entier naturel n, $u_{n+1} - u_n \leqslant 0$, alors la suite u est **décroissante**.

Propriété 2.2

Une fonction f est définie sur $[0\ ;\ +\infty[$ et pour pour tout entier naturel $u_n = f(n)$.

- Si la fonction f est **croissante** sur $[0\ ;\ +\infty[$, alors la suite u est **croissante**.
- Si la fonction f est **décroissante** sur $[0\ ;\ +\infty[$, alors la suite u est **décroissante**.

2.2.b Exemples

Exemple 2.3

Sens de variation de la suite définie pour tout entier naturel n par $u_n = n^2 + n$.

Étudions le signe de $u_{n+1} - u_n$, ainsi on pourra appliquer la propriété 2.1.

$u_{n+1} - u_n = (n+1)^2 + (n+1) - (n^2 + n) = n^2 + 2n + 1 + n + 1 - n^2 - n = 2n + 2$

Pour étudier le signe de $2n + 2$, on peut chercher à résoudre l'inéquation $2n + 2 \geqslant 0$, mais il est beaucoup plus simple de raisonner ainsi :

puisque n est entier naturel donc $n \geqslant 0$, donc $2n \geqslant 0$ donc $2n + 2 \geqslant 2 > 0$ donc $u_{n+1} - u_n > 0$,

donc la suite (u_n) est croissante.

Exemple 2.4

Sens de variation de la suite définie par $u_n = n^2$.

Nous allons appliquer la propriété 2.2.

On sait que la fonction $f : x \longmapsto x^2$ est croissante sur $[0\ ;\ +\infty[$, donc la suite définie par $u_n = n^2$ est croissante.

2.3 Limites

2.3.a Exemples de limites

Exemple 2.5

La suite (u_n) est définie pour tout entier naturel $n \geqslant 1$ par $u_n = \dfrac{3n+1}{n}$.

En observant les valeurs successives à la calculatrice, il semble que lorsque n devient grand, u_n semble s'approcher de 3.

En effet :

$u_1 = 4$ $\quad u_5 = 3,2$ $\quad u_{10} = 3,1$ $\quad u_{100} = 3,01$ $\quad u_{500} = 3,02$ $\quad u_{1000} = 3,001$ $\quad$ etc.

Conjecture : la limite de $\dfrac{3n+1}{n}$ lorsque n tend vers $+\infty$ est égale à 3.

Exemple 2.6

La suite (u_n) est définie pour tout entier naturel n par $u_n = n^3$.

En observant les valeurs successives à la calculatrice, il semble que lorsque n devient grand, u_n semble augmenter indéfiniment.

En effet :

$u_1 = 1$ $\quad u_5 = 625$ $\quad u_{10} = 1\,000$ $\quad u_{100} = 1\,000\,000$ $\quad u_{1000} = 1\,000\,000\,000$ $\quad$ etc.

Conjecture : la limite de n^3 lorsque n tend vers $+\infty$ est égale à $+\infty$.

Exemple 2.7

La suite (u_n) est définie pour tout entier naturel n par $u_n = (-1)^n$.

Les valeurs successives sont :

$u_0 = (-1)^0 = 1$ $\quad u_1 = (-1)^1 = -1$ $\quad u_2 = (-1)^2 = 1$ $\quad u_3 = (-1)^3 = -1$ $\quad$ etc.

La suite $((-1)^n)$ n'a pas de limite.

Conclusion sur les suites et les limites

Les différentes situations possibles pour les suites et les limites sont énumérées ci-dessous.

- Une suite a une limite égale à un nombre réel, c'est à dire une limite finie.
- Une suite a une limite égale à $+\infty$.
- Une suite a une limite égale à $-\infty$.
- Une suite n'a pas de limite.

2.3.b Programme et algorithme de seuil

Exemple 2.8

On admet que la suite (u_n) définie par $u_n = 3^n$ est croissante et que lorsque n tend vers $+\infty$, sa limite est $+\infty$. On veut savoir à partir de quel rang n on a : $u_n \geqslant 500\,000$. Le nombre 500 000 est appelé *un seuil*.

Dans le tableau ci-contre se trouvent une fonction Python nommée `seuil` qui permet de savoir à partir de quel rang n on a $u_n \geqslant s$, et à droite l'algorithme correspondant.

	Fonction Python	**Algorithme**
1	`def seuil(s):`	
2	`  n=1`	$n \leftarrow 1$
3	`  u=3`	$u \leftarrow 3$
4	`  while(u<s):`	Tant que $u < s$
5	`  n=n+1`	$\quad n \leftarrow n+1$
6	`  u=3**n`	$\quad u \leftarrow 3^n$
7		Fin du Tant que
8	`  return(u)`	

Explications

- Ligne 1 : définition de la fonction `seuil`
- Ligne 2 : on commence à $n = 1$
- Ligne 3 : on calcule d'abord $u_1 = 3^1 = 3$
- Lignes 4, 5, 6 : on exécute les lignes 5 et 6 tant que $u < s$, ce qui veut dire qu'on arrête dès que $u \geqslant s$
  - Ligne 5 : n augmente de 1
  - Ligne 6 : u prend la valeur $u_n = 3^n$
- Ligne 7 : fin de la boucle *Tant que*, que l'on n'indique pas en Python.
- Ligne 8 : la boucle Tant que est terminée, donc $u \geqslant s$, et on retourne la valeur de n qui est le rang n à partir duquel $u_n \geqslant s$.

Exécution

À la console on saisit `seuil(500000)`, et la réponse est 12.

Conclusion : le rang n à partir duquel $u_n \geqslant 500\,000$ est 12.

Exemple 2.9 (Suite définie par récurrence)

On admet que la suite (u_n) définie par $u_0 = 10$ et $u_{n+1} = 0,5u_n$ est décroissante et que lorsque n tend vers $+\infty$, sa limite est 0. On veut savoir à partir de quel rang n on a : $u_n < 10^{-6}$. Le seuil est donc 10^{-6}.
Dans le tableau ci-contre se trouvent une fonction Python nommée `seuil` qui permet de savoir à partir de quel rang n on a $u_n < s$, et à droite l'algorithme correspondant.

	Fonction Python	Algorithme
1	`def seuil(s):`	
2	`  n=0`	$n \leftarrow 0$
3	`  u=10`	$u \leftarrow 10$
4	`  while(u>=s):`	Tant que $u \geqslant s$
5	`  n=n+1`	$n \leftarrow n+1$
6	`  u=0.5*u`	$u \leftarrow 0,5u$
7		Fin du Tant que
8	`  return(n)`	

Exécution

À la console on saisit `seuil(1e-6)`, et la réponse est 24.
Conclusion : le rang n à partir duquel $u_n < 10^{-6}$ est 24.

2.4 Liste des termes d'une suite en Python

La façon de créer une liste de termes d'une suite en Python est différente selon que cette suite est définie explicitement (en fonction de n) ou que cette suite est définie par récurrence, mais avant de créer une fonction Python qui retourne une liste de termes d'une suite, il faut donner quelques explications sur le fonctionnement d'une liste en Python.

2.4.a Les listes en Python

Dans une console Python, créons par exemple la liste `[1,2,3]` en la nommant lst

```
>>> lst=[1,2,3]
```

Vérifions le contenu de `lst`

```
>>> lst
[1, 2, 3]
```

Pour les listes définies par récurrence, nous aurons besoin de créer une liste ne contenant au départ que le premier terme, puis il faudra ensuite ajouter le deuxième terme à cette liste, puis ajouter le troisième terme et ainsi de suite.

La commande pour ajouter un terme à une liste est *NomDeLaListe*`.append`

Voici un exemple : créons ci-dessous la liste `[10]` ne contenant que le nombre 10, nommée `lst2`, puis vérifions le contenu.

```
>>> lst2=[10]
>>> lst2
[10]
```

Ajoutons le nombre 50 et vérifions.

```
>>> lst2.append(50)
>>> lst2
[10, 50]
```

2.4.b Liste des termes d'une suite définie explicitement

Définition 2.4

Pour une suite (u_n) définie par $u_n = f(n)$, la liste des termes consécutifs de u_n à u_p est obtenue par la commande : `[f(i) for i in range (n,p+1)]`

Exemple 2.10

La suite (u_n) est définie par $u_n = n^3 + 6n - 7$, et on veut obtenir la liste $[u_5, u_6, u_7, u_8, u_9]$.

```
>>> [i**3-6*i-7 for i in range (5,10)]
[148, 245, 378, 553, 776]
```

2.4.c Liste des premiers termes d'une suite définie par récurrence

Exemple 2.11

La suite (u_n) est définie par $u_0 = 1$ et, pour tout entier naturel n, $u_{n+1} = 3u_n + 4$.

On veut obtenir la liste $[u_0, u_1, u_2]$.

Comme cela a été indiqué plus haut, on crée d'abord la liste ne contenant que u_0, puis on ajoute u_1 à la liste, puis on ajoute u_2.

Pour ajouter un terme à une liste, on utilise la commande *NomDeLaListe*`.append`

On ouvre une console Python et on peut procéder comme cela est indiqué ci-dessous.

```
>>> lst=[1]
>>> lst
[1]
>>> u=1
1
>>> u=3*u+4
>>> u
7
>>> lst.append(u)
>>> lst
[1,7]
>>> u=3*u+4
>>> u
25
>>> lst.append(u)
>>> lst
[1,7,25]
>>> u=3*u+4
>>> u
79
>>> lst.append(u)
>>> lst
[1,7,25,79]
```

Remarque 2.1

Il est bien évident que la méthode de l'exemple 2.11 est beaucoup trop longue à exécuter.

C'est pourquoi l'exemple suivant va donner un meilleur procédé.

Exemple 2.12

Reprenons l'exemple de la suite (u_n) définie par $u_0 = 1$ et, $u_{n+1} = 3u_n + 4$, et de la liste $[u_0, u_1, u_2]$.

Dans l'exemple 2.11 on commence par les commandes `lst=[1]` et `u=1` puis on répète trois fois les commandes `u=3*u+4` et `lst.append(u)`.

Nous allons donc placer ces deux dernières commandes dans une boucle Pour `for i in range(1,4)`.

Nous obtenons ainsi la procédure Python et l'algorithme ci-dessous.

Les variables sont

- `listetermes` qui est une liste
- u qui est un réel
- i qui est un entier naturel

Fonction Python	**Algorithme**
`def lt():`	
`  listetermes=[1]`	`listetermes` ← [1]
`  u=1`	$u \leftarrow 1$
`  for i in range(1,4):`	Pour $k = 1$ jusqu'à $k = 3$
`    u=3*u+4`	$u \leftarrow 3u + 4$
`    listetermes.append(u)`	ajouter u à la liste `listetermes`
`  return listetermes`	

On exécute à la console cette procédure en saisissant simplement `lt()`

Propriété 2.3

Considérons une suite (u_n) définie par récurrence par son premier terme et par $u_{n+1} = f(u_n)$.
Alors pour obtenir la liste des premiers termes de cette suite jusqu'au terme u_n, on peut utiliser la fonction Python et l'algorithme plus bas.
Les variables sont précisées ci-dessous.

- `premterme` est un réel
- `rangdt` est un entier naturel
- `listetermes` est une liste
- u est un réel
- i est un entier naturel

Fonction Python	**Algorithme**
`def lt(premterme,rangdt):`	
`  listetermes=[premterme]`	`listetermes` ← [premterme]
`  u=premterme`	$u \leftarrow$ premterme
`  for i in range(1,rangdt+1):`	Pour $k = 1$ jusqu'à $k =$ `rangdt`
`    u=f(u)`	$u \leftarrow f(u)$
`    listetermes.append(u)`	ajouter u à la liste `listetermes`
`  return listetermes`	

2.5 Somme de termes consécutifs d'une suite

2.5.a Somme de termes avec la calculatrice Numworks

Exemple 2.13

La suite (u_n) est définie par $u_n = n^2 + 5n$. À l'aide de la calculatrice Numworks, calculons la somme $S = u_0 + u_1 + u_2 + u_3 + u_4 + u_5 + u_6 + u_7 + u_8$.

- Aller dans le module Calculs.
- Appuyer sur la touche *Boîte à outils* (paste).
- Descendre sur Calcul ▷.
- Appuyer sur la touche ▷.
- Descendre sur sum(f(n),n,nmin,nmax).
- Appuyer sur la touche EXE.
- Compléter ainsi : $\sum_{0}^{8} n^2 + 5n$ (pour le n, utiliser la touche x,n,t).
- Appuyer sur la touche EXE.

Résultat : $S = 384$

2.5.b Somme de termes avec Python ou un algorithme

Pour une suite (u_n), on veut écrire une fonction Python ou un algorithme qui permette de calculer la somme : $S = u_0 + u_1 + u_2 + \cdots + u_n$.

Le principe est le suivant : on commence par affecter 0 à une variable `somme`, puis on utilise une boucle Pour de $i = 0$ jusqu'à $i = n$, et, à chaque étape de cette boucle Pour :

- on calcule u_i pour chaque valeur de i ;
- on ajoute à chaque fois la valeur de u_i dans la variable `somme`.

Il reste à préciser une fonction Python ou un algorithme selon que la suite concernée soit définie explicitement ou soit définie par récurrence. C'est ce qui est précisé dans les deux propriétés ci-dessous.

Propriété 2.4 (Somme de termes pour une suite définie explicitement)

Pour une suite (u_n) définie par $u_n = f(n)$, la fonction Python et l'algorithme ci-dessous permettent de calculer $S = u_0 + u_1 + u_2 + \cdots + u_n$, en précisant que :

- n est nommé `rangfin` ;
- la somme S est la valeur finale de la variable `somme` ;
- les variables `rangfin` et i sont des entiers naturels, et la variable `somme` est un réel.

Fonction Python	**Algorithme**

`def somexp(rangfin):`
`  somme=0`
`  for i in range(0,rangfin+1)`
`  somme = somme + f(i)`
 	
`  return(somme)`
	somme ← 0
	Pour $i = 0$ jusqu'à i =rangfin
	somme ← somme + $f(i)$
	Fin du Pour

Propriété 2.5 (Somme de termes pour une suite définie par récurrence)

Pour une suite (u_n) définie par récurrence par son premier terme u_0 et par $u_{n+1} = f(u_n)$, la fonction Python et l'algorithme ci-dessous permettent de calculer $S = u_0 + u_1 + u_2 + \cdots + u_n$, en précisant que :

- n est nommé `rangfin` ;
- le premier terme est nommé `premterme` ;
- la variable `somme` n'est pas initialisée à 0 mais à la valeur du premier terme ;
- la somme S est la valeur finale de la variable `somme` ;
- les variables `rangfin` et i sont des entiers naturels, et la variable `somme` est un réel.

Fonction Python	**Algorithme**

`def somrec(premterme,rangfin):`
`  u=premterme`
`  somme=premterme`
`  for i in range(1,rangfin+1)`
`  u=f(u)`
`  somme=somme+u`
 	
`  return(somme)`
	$u \leftarrow$ premterme
	somme $\leftarrow$ premterme
	Pour $i = 1$ jusqu'à $i =$rangfin
	$u \leftarrow f(u)$
	somme $\leftarrow$ somme $+u$
	Fin du Pour

Exemple 2.14

La suite (u_n) est définie par $u_n = \dfrac{3n}{n+1}$. Calculons la somme $S = u_0 + u_1 + u_2 + u_3 + u_4 + u_5$.

On utilise la fonction Python nommée `somexp` de la propriété 2.4 en remplaçant `f(i)` par `(3*i)/(i+1)`, puis on saisit à la console `somexp(5)` :

```
>>> somexp(5)
10.65
```

Donc : $S = \boxed{10,65}$

Exemple 2.15

La suite (u_n) est définie par $u_0 = 1$, et, pour tout entier naturel n, $u_n = 1,5u_n + 2$.
Calculons la somme $S = u_0 + u_1 + u_2 + u_3 + u_4 + u_5 + u_6 + u_7$.

On utilise la fonction Python nommée `somrec` de la propriété 2.5 en remplaçant `f(u)` par `1.5*u+2`, puis on saisit à la console `somrec(1,7)` :

```
>>> somrec(1,7)
214.2890625
```

Donc : $S = \boxed{214{,}289\,062\,5}$

III Approfondissement

Exercice 2.34 (Somme des n premiers carrés, des n premiers cubes)

1. On veut calculer la somme $S_n = 1^2 + 2^2 + 3^2 + \cdots + (n-1)^2 + n^2$.
On définit un polynôme du 3e degré $P(x) = ax^3 + bx^2 + cx + d$ tel que $P(x+1) - P(x) = x^2$

a) Développer et réduire l'expression $P(x+1) - P(x)$.

b) En identifiant $P(x+1) - P(x)$ avec x^2, écrire un système d'équations d'inconnues a, b, c, d, et le résoudre.
On obtient ainsi $P(x)$.

c) On écrit alors ceci :

$$\begin{array}{ccccc} P(n+1) & - & P(n) & = & n^2 \\ P(n) & - & P(n-1) & = & (n-1)^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ P(4) & - & P(3) & = & 3^2 \\ P(3) & - & P(2) & = & 2^2 \\ P(2) & - & P(1) & = & 1^2 \end{array}$$

On ajoute ensuite membre à membre toutes ces égalités, de nombreux termes s'éliminent et on obtient un égalité qui permet de calculer la somme S_n en fonction de n.

2. Pour calculer la somme $T_n = 1^3 + 2^3 + 3^3 + \cdots + (n-1)^3 + n^3$, on définit un polynôme du 4e degré $Q(x) = ax^4 + bx^3 + cx^2 + dx + e$ tel que $Q(x+1) - Q(x) = x^3$.
On procède ensuite de la même manière que pour la somme S_n.