CALCULS FAUX

FICHE ENSEIGNANT

Niveau concerné :

1ère S, Terminale S

Durée :

Séance de 1/2H

Type de travail :

Programmer en Python

Thèmes du programme :

Utiliser l'outil informatique pour calculer

Scénario :

En exercice à la maison ou en classe, l'objectif est de faire prendre conscience que le calcul numérique et la représentation des nombres en binaire peuvent occasionner des situations inattendues. On amène donc l'élève à réfléchir à l'utilisation de l'outil numérique dans une démonstration.

FICHE ELEVE

1. Prendre conscience du problème.

On considère les suites suivantes :

$$\begin{cases} u_0 = 0{,}125 \\ u_{n+1} = 100u_n - 12{,}375 \end{cases} \qquad \begin{cases} v_0 = 0{,}1 \\ v_{n+1} = 100v_n - 9{,}9 \end{cases}$$

a. A l'aide du langage Python, calculer les 10 premiers termes de chacune des deux suites. Quelles semblent être les variations de ces deux suites ?

b. Calculer les trois premiers termes de chaque suite à la main. Qu'y a-t-il de surprenant ?

c. Pour les élèves de terminale S, démontrer par récurrence que la suite (v_n) est stationnaire. On admettra dans la suite du DM que ces deux suites sont stationnaires.

2. Utiliser une bibliothèque

L'écriture des nombres en binaire ne permet pas de représenter certains nombres décimaux. C'est le cas du nombre 0,1.

Au même titre que la valeur intrinsèque du nombre $\frac{1}{3}$ dans notre base décimale n'admet pas une écriture décimale finie, $\frac{1}{3} \approx 0{,}3333\ldots$, la valeur intrinsèque du nombre $\frac{1}{10}$ n'admet pas d'écriture finie en binaire.

Utiliser une variable de type *float* pour stocker 0,1 revient à utiliser une valeur approchée de 0,1. C'est la raison pour laquelle nous avons pu observer que malheureusement il arrive que nos ordinateurs calculent faux.

Pour pallier à ce problème, il existe une bibliothèque de Python qui permet de manipuler les fractions.

Il faut commencer par importer la bibliothèque *fractions*. Il suffit de saisir la ligne de commande ci-dessous où * signifie que l'on souhaite charger toutes les méthodes contenues dans la bibliothèque *fractions* :

```
>>> from fractions import *
```

Pour écrire la fraction $\frac{125}{1000}$, il suffit de saisir d'utiliser la commande *Fraction* qui fait partie de la bibliothèque *fractions* :

```
>>> Fraction(125,1000) #On peut également saisir fractions.Fraction(125,1000)
```

Pour additionner deux fractions, on peut saisir

```
>>> Fraction(1,2)+Fraction(1,4)
```

Réécrire le programme de la question 1.a. à l'aide des fractions et vérifier que les suites sont bien stationnaires.

3. Démontrer avec Python

On considère la suite définie pour tout entier *n* par : $u_n = -\frac{1}{240}n^4 + \frac{1}{40}n^3 - \frac{11}{240}n^2 + \frac{1}{40}n + \frac{1}{10}$.

Démontrer en s'aidant du langage Python que cette suite n'est pas stationnaire.

Remarque : pour calculer n^4 on peut utiliser l'opération n*n*n*n

FICHE PROF

1. Prendre conscience du problème.

On considère les suites suivantes :

$$\begin{cases} u_0 = 0{,}125 \\ u_{n+1} = 100u_n - 12{,}375 \end{cases} \qquad \begin{cases} v_0 = 0{,}1 \\ v_{n+1} = 100v_n - 9{,}9 \end{cases}$$

a. A l'aide du langage Python, calculer les 10 premiers termes de chacune des deux suites.
Quelles semblent être les variations de ces deux suites ?

```
u=0.125
v=0.1
for i in range(10):
    print("u%s=%s et v%s=%s" %(i,u,i,v))
    #les pourcentages dans la fonction print désignent les variables
    #à afficher, l'option de formatage %s affiche la variable comme un texte
    u=100*u-12.375
    v=100*v-9.9
```

Affichage

```
u0=0.125 et v0=0.1
u1=0.125 et v1=0.09999999999999964
u2=0.125 et v2=0.09999999999996412
u3=0.125 et v3=0.0999999999964114
u4=0.125 et v4=0.09999999964114004
u5=0.125 et v5=0.09999996411400325
u6=0.125 et v6=0.09999641140032445
u7=0.125 et v7=0.0996411400324444
u8=0.125 et v8=0.06411400324443939
u9=0.125 et v9=-3.4885996755560615
```

b. Calculer les trois premiers termes de chaque suite à la main. Qu'y a-t-il de surprenant ?
On obtient $u_0 = u_1 = u_2 = 0{,}125$ et $v_0 = v_1 = v_2 = 0{,}1$… Ces suites semblent stationnaires.

c. Pour les élèves de terminale S, démontrer par récurrence que la suite (v_n) est stationnaire. On admettra dans la suite du DM que ces deux suites sont stationnaires.
Montrons que la suite (v_n) est stationnaire et donc que pour tout entier *n*, $v_n = 0{,}1$.
Initialisation : $v_0 = 0{,}1$ donc la propriété est initialisée au rang 0.
Hérédité : On suppose la propriété vraie à un rang *n*, c'est-à-dire $v_n = 0{,}1$ et on souhaite montrer qu'elle est vraie au rang *n+1* soit $v_{n+1} = 0{,}1$.
Or, $v_{n+1} = 100 \times v_n - 9{,}9 = 100 \times 0{,}1 - 9{,}9 = 0{,}1$.
La propriété est donc vraie au rang *n+1*.
On en déduit que pour tout entier *n* la suite est stationnaire.

4. Utiliser une bibliothèque

L'écriture des nombres en binaire ne permet pas de représenter certains nombres décimaux. C'est le cas du nombre 0,1.

Au même titre que la valeur intrinsèque du nombre $\frac{1}{3}$ dans notre base décimale n'admet pas une écriture décimale finie, $\frac{1}{3} \approx 0{,}3333$ …, la valeur intrinsèque du nombre $\frac{1}{10}$ n'admet pas d'écriture finie en binaire.

Utiliser une variable de type *float* pour stocker 0,1 revient à utiliser une valeur approchée de 0,1. C'est la raison pour laquelle nous avons pu observer que malheureusement il arrive que nos ordinateurs calculent faux.

Pour pallier à ce problème, il existe une bibliothèque de Python qui permet de manipuler les fractions.

Il faut commencer par importer la bibliothèque *fractions*. Il suffit de saisir la ligne de commande ci-dessous où * signifie que l'on souhaite charger toutes les méthodes contenues dans la bibliothèque *fractions* :

```
>>> from fractions import *
```

Pour écrire la fraction $\frac{125}{1000}$, il suffit de saisir d'utiliser la commande *Fraction* qui fait partie de la bibliothèque *fractions* :

```
>>> Fraction(125,1000) #On peut également saisir fractions.Fraction(125,1000)
```

Pour additionner deux fractions, on peut saisir

```
>>> Fraction(1,2)+Fraction(1,4)
```

Réécrire le programme de la question 1.a. à l'aide des fractions et vérifier que les suites sont bien stationnaires.

```
from fractions import *
u=Fraction(125,1000)
v=Fraction(1,10)

for i in range(10):
    print("u%s=%s et v%s=%s" %(i,u,i,v))
    u=100*u-Fraction(12375,1000)
    v=100*v-Fraction(99,10)
```

Affichage

```
u0=1/8 et v0=1/10
u1=1/8 et v1=1/10
u2=1/8 et v2=1/10
u3=1/8 et v3=1/10
u4=1/8 et v4=1/10
u5=1/8 et v5=1/10
u6=1/8 et v6=1/10
u7=1/8 et v7=1/10
u8=1/8 et v8=1/10
u9=1/8 et v9=1/10
```

2. Démontrer avec Python

On considère la suite définie pour tout entier *n* par : $u_n = -\frac{1}{240}n^4 + \frac{1}{40}n^3 - \frac{11}{240}n^2 + \frac{1}{40}n + \frac{1}{10}$.

Démontrer en utilisant le langage Python que cette suite n'est pas stationnaire.

Remarque : pour calculer n^4 on peut utiliser l'opération n*n*n*n

```
from fractions import Fraction

#Fonction qui renvoie le terme d'indice n de la suite (version décimale)
def f(n):
    return (-1/240)*n*n*n*n+(1/40)**n*n*n-(11/240)*n*n+(1/40)*n+1/10
#Fonction qui renvoie le terme d'indice n de la suite (version fraction)
def f_fract(n):
    return Fraction(-1,240)*n*n*n*n+Fraction(1,40)*n*n*n-Fraction(11,240)*n*n+Fraction(1,40)*n+Fraction(1,10)

for i in range(10):
    print("u%s=%s et u%s=%s" %(i,f(i),i,f_fract(i)))
```

Affichage

```
u0=0.1 et u0=1/10
u1=0.1 et u1=1/10
u2=-0.0975 et u2=1/10
u3=-0.5748593750000001 et u3=1/10
u4=-1.5999937499999999 et u4=0
u5=-3.524999755859374 et u5=-2/5
u6=-6.799999991210937 et u6=-7/5
u7=-11.974999999700927 et u7=-17/5
u8=-19.699999999990233 et u8=-69/10
u9=-30.724999999999685 et u9=-25/2
```

On remarque que l'on peut accorder une confiance toute relative aux calculs avec les nombres en écriture flottante, par contre en utilisant la classe *fractions* on s'aperçoit qu'il suffit de calculer u_4 pour montrer que la suite n'est pas stationnaire.