Région académique
PAYS DE LA LOIRE

Des suites numériques - Première S :

Construction de suites numériques tendant vers π

Objectif : Il s'agit ici de proposer trois exemples de suites numériques dont la limite dépend de π. L'idée est à chaque fois de construire les premiers termes sur papier puis d'utiliser un logiciel de programmation pour allez plus loin

- En Première S, ces activités peuvent servir d'exemples de constructions récurrentes. On peut aussi donner ainsi une première idée de la notion de limite et de seuil.
- En Terminale S, le travail pourra être repris pour créer un programme de dépassement d'un seuil fixé de manière automatique.

Activité 1 : Construction d'une suite numérique
On définie la suite (I_n) par l'écriture suivante :

$$I_n = \underbrace{1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots}_{n \text{ termes}}$$

Les premiers termes de cette suite sont donc $I_1 = 1$ et $I_2 = \frac{2}{3}$.

1. Calculer les termes I_3, I_4 et I_5.
2. On définit la suite $u_n = 2n - 1$ pour n entier strictement positif. Décrire les nombres de cette suite d'une phrase la plus simple possible.
3. On peut alors remarquer que la suite (I_n) peut être définie plus facilement pour les deux conditions :

$$I_1 = 1 \text{ et } I_{n+1} = I_n + \frac{(-1)^{n+1}}{u_{n+1}}$$

   (a) Compléter l'algorithme suivant pour que la variable I soit égale à I_{10} à la fin de l'exécution.

```
n ← 10
I ← 1
Pour i allant de 2 à n
        I = I + (-1)^(n+1)/(.....)
```

   (b) Avec Python, programmer une fonction `I(n)` qui prend une valeur n en entrée et calcule ensuite I_n. On peut se rappeler que la puissance en Python s'écrit **.

4. Pour chaque valeur de I_n calculée, regarder la valeur de $4I_n$. Qu'observe-t-on lors que n est de plus en plus grand (10,100,... par exemple) ?

Exemples de programmes :

- La suite I_n :

```
def I(n): #Calcul de la somme alternée des inverses des nombres impairs
    I=1
    for i in range(2,n+1): #Calcul des termes de I_2 à I_n
        I=I+(-1)**(i+1)/(2*i-1) #rajout d'un terme
    return I
```

- Calcul des termes jusqu'à avoir une variation faible :

```
def variation(U,e): #Calculs des termes successifs
    a=U(1)
    b=U(2)
    i=2
    while abs(b-a)>e: #Condition d'arrêt: une variation inférieur à e
       a=b
       i=i+1
       b=U(i)
    return b
```

Activité 2 : Construction d'une suite numérique(2)
On définie la suite (C_n) par l'écriture suivante :

$$C_n = \underbrace{1 + \frac{1}{4} + \frac{1}{9} + \dots}_{n \text{ termes}}$$

Les premiers termes de cette suite sont donc $C_1 = 1$ et $C_2 = \frac{5}{4}$.

1. Calculer les termes C_3, C_4 et C_5.
2. On peut remarquer que la suite (C_n) peut être définie plus facilement pour les deux conditions :

$$C_1 = 1 \text{ et } C_{n+1} = C_n + \frac{1}{(n+1)^2}$$

   (a) Compléter l'algorithme suivant pour que la variable C soit égale à C_{10} à la fin de l'exécution.

```
n ← 10
C ← 1
Pour i allant de 2 à n
        C = C + 1/(.....)
```

   (b) Avec Python, programmer une fonction `C(n)` qui prend une valeur n en entrée et calcule ensuite C_n. On peut se rappeler que la puissance en Python s'écrit **.

3. Pour chaque valeur de C_n calculée, regarder la valeur de $\sqrt{6C_n}$ (il faudra importer la bibliothèque `math` pour le faire avec Python). Qu'observe-t-on lors que n est de plus en plus grand (10,100,... par exemple) ?
4. On peut définir la somme K_n des n premiers inverses des carrées des nombres impairs. Programmer alors la suite K_n puis calculer $\sqrt{8K_n}$

Exemples de programmes :

- La suite C_n :

```
def C(n): #Calcul de la somme des inverses des carrés
    C=1
    for i in range(2,n+1): #Calcul des termes de C_2 à C_n
        C=C+1/i**2 #rajout d'un terme
    return C
```

- La suite K_n :

```
def K(n): #Calcul de la somme des inverses des carrés
    K=1
    for i in range(2,n+1): #Calcul des termes de K_2 à K_n
        K=K+1/(2*i-1)**2 #rajout d'un terme
    return K
```

- Calcul des termes jusqu'à avoir une variation faible :

```
def variation(U,e): #Calculs des termes successifs
    a=U(1)
    b=U(2)
    i=2
    while abs(b-a)>e: #Condition d'arrêt: une variation inférieur à e
       a=b
       i=i+1
       b=U(i)
    return b
```

Activité 3 : Construction d'une suite numérique(3)
On définie la suite (R_n) par les conditions suivantes :

$$R_1 = \sqrt{2} \text{ et } R_{n+1} = \sqrt{2 + R_n}$$

1. Le terme $2^{n+1}\sqrt{2 - R_n}$ admet une limite lorsque n tend vers $+\infty$. Recherchez la valeur de celle-ci
   Indications : Vous pourrez créer un programme Python qui calculera une valeur approchée de R_n pour n choisi.
2. Pour des élèves de Terminale S, on peut montrer que la suite (R_n) est croissante et majorée par 2. La suite (R_n) converge donc bien.

La suite R_n :

```
from math import*

def R(n):
    R=sqrt(2)
    for i in range(2,n+1):
        R=sqrt(2+R)
    return R
```