Nombres flottants

INFORMATIQUE COMMUNE - TP n° 2.8 - Olivier Reynet

À la fin de ce chapitre, je sais :

☞ sortir de la salle dans laquelle on m'a pris au piège

A Comment sortir de la salle ?

Ça y est. Vous y êtes. Dernier TP d'informatique de l'année. En fermant la porte derrière vous, un bruit sinistre, un enclenchement de verrou rouillé résonne dans la salle. Vous êtes coincé par un beau soleil de juin, à deux pas de la plage alors que vous avez presque fini les cours !

C'est à ce moment que vous le voyez [1], sur la porte, un petit message de votre professeur [2] : «Les portes peuvent être déverrouillées en suivant ce lien. Le code pour sortir se calcule de la manière suivante :

- faire la somme des résultats des questions B3 et B4,
- multiplier par la partie fractionnaire du résultat de la dernière question de la section C,
- multiplier le tout par l'indice n_0 de la limite trouvée à la section D,
- le code est le reste de la division euclidienne de ce nombre par 21062023.

Bonne chance !»

B Du rififi dans la sommation

Un élève tente de calculer la somme suivante :

$$\sum_{k=1}^{n} \frac{1}{n} \tag{1}$$

Il lui semble que le résultat vaut un, mais, pas très sûr de ses méthodes de calcul analytique, il décide de vérifier avec Python si son résultat est correct.

B1. Écrire une fonction de signature `somme64bits(n:int)-> int` qui renvoie cette somme.

B2. Écrire une fonction de signature `somme32bits(n:numpy.float32)-> numpy.float32` qui renvoie cette somme calculée avec des flottants codés sur 32 bits grâce à Numpy.

Le dernier cours d'informatique sur les bonnes pratiques lui donne l'idée suivante : il va tester les deux fonctions précédentes pour toutes les valeurs de n de un à 2^{14} inclus.

1. trop tard…
2. qui n'est plus dans la salle d'ailleurs !

B3. À la suite de ce test, combien de valeurs de n vérifie $\sum_{k=1}^{n} \frac{1}{n} = 1$ lorsqu'on utilise la fonction `somme64bits(n:int)-> int`?

B4. Même question lorsqu'on utilise la fonction `somme32bits(n)`?

B5. Interpréter ces résultats.

C Du bruit dans l'intégration

On souhaite connaître l'évolution d'un système dynamique après une seconde de fonctionnement. Ce système obéit à l'équation d'évolution suivante :

$$\frac{\mathrm{d}x}{\mathrm{d}t} = 1 \tag{2}$$

Par ailleurs, la valeur initiale de x est connue : $x(0) = 0$.

Grâce à ses cours de mathématiques et sa science du calcul, un élève se doute que x vaut 1 à la fin du temps écoulé. Cependant, il souhaite vérifier en codant avec Python la résolution numérique. Grâce à ses cours de physique, il parvient à écrire le code suivant :

```
def f(x):
    return 1

def one_step(f, x, dt):
    return x + dt * f(x)

def euler(f, x0, dt, T=1):
    t = 0
    x = x0
    n = int(T / dt)
    for _ in range(n):
        x = one_step(f, x, dt)
    return x
```

C1. À l'aide de la méthode d'Euler, simuler ce système pendant une seconde par pas de 0,1 seconde.

C2. En divisant à chaque fois par 10, faire varier le pas d'intégration de 0,1 à 10^{-6} et mémoriser les résultats obtenus.

C3. Sachant que x devrait valoir 1 au bout d'une seconde, calculer l'erreur absolue commise pour chaque pas d'intégration de 0,1 à 10^{-6} .

C4. Expliquer le phénomène observé sur l'erreur absolue.

C5. Proposer un pas d'intégration compris entre 0,1 et 0,2 qui annule l'erreur absolue dans ce cas.

D Un scandale en série

La série harmonique s'écrit :

$$H_n = \sum_{k \geqslant 1} \frac{1}{k} \tag{3}$$

D1. La série harmonique est-elle convergente?

D2. Écrire une fonction de signature `f32_harm(n:int)-> numpy.float32` qui calcule la somme des n premiers termes de la série harmonique dans l'ordre croissant des k en utilisant des flottants codés sur 32 bits.

D3. Écrire une fonction de signature `rf32_harm(n:int)-> numpy.float32` qui calcule la somme des n premiers termes de la série harmonique dans l'ordre décroissants des k en utilisant des flottants codés sur 32 bits.

D4. Y-a-t-il un intérêt à procéder dans le sens croissant ou décroissant. Pourquoi?

D5. Montrer que, si on s'y prend mal, la série harmonique calculée sur des flottants 32 bits converge et atteint sa limite. Estimer l'indice n_0 à partir duquel la limite est atteinte.

D6. Trouver expérimentalement cette limite n_0.