

Problème de Mathématiques

Référence pp1805 — Version du 14 octobre 2025

On sait que la série de Riemann $\sum 1/k^2$ est convergente, ce qui signifie que ses sommes partielles

$$U_n = \sum_{k=1}^n \frac{1}{k^2}$$

convergent vers un réel S (égal à $\pi^2/6$, mais ceci est une autre histoire). Nous allons voir comment calculer une valeur approchée de S en limitant au minimum les calculs.

Partie A. Estimations

- 1.a. Proposer une fonction $S(n)$ écrite en langage Python dont l'argument n est un entier naturel $n \geq 1$ et qui retourne la valeur de la somme partielle U_n .
- 1.b. Compter, en fonction de n , le nombre d'opérations effectuées lors de l'exécution de $S(n)$.
- 1.c. Quel doit être l'ordre de grandeur de l'entier n pour que U_n soit une valeur approchée de S à 10^{-3} près?

Partie B. Accélération de convergence

Pour tout entier $n \geq 1$, on pose

$$V_n = U_n + \frac{1}{n}.$$

2. Expliquer (sans le moindre calcul) pourquoi V_n est plus proche de S que U_n .
3. Expliciter la suite $(v_k)_{k \geq 1}$ telle que

$$\forall n \geq 1, \quad V_n = \sum_{k=1}^n v_k.$$

4. Calculer un équivalent de v_k lorsque k tend vers $+\infty$.
5. En déduire que

$$S - V_n \sim \frac{-1}{2n^2}$$

lorsque n tend vers $+\infty$.

6. On pose maintenant

$$\forall n \geq 1, \quad W_n = V_n - \frac{1}{2n^2}.$$

- 6.a. Expliciter la suite $(w_k)_{k \geq 1}$ telle que

$$\forall n \geq 1, \quad W_n = \sum_{k=1}^n w_k.$$

- 6.b. En déduire que

$$S - W_n \sim \frac{1}{6n^3}$$

lorsque n tend vers $+\infty$.

7. Expliquer sommairement comment définir une suite $(X_n)_{n \geq 1}$ telle que

$$S - X_n = \mathcal{O}\left(\frac{1}{n^4}\right)$$

lorsque n tend vers $+\infty$.

On peut poursuivre les calculs en suivant la même procédure et obtenir par exemple :

$$S - S_n = \frac{1}{n} - \frac{1}{2n^2} + \frac{1}{6n^3} - \frac{1}{30n^5} + \frac{1}{42n^7} + \mathcal{O}\left(\frac{1}{n^9}\right)$$

lorsque n tend vers $+\infty$.

Partie C. Exploitation des résultats

8. Proposer deux fonctions $V(n)$ et $W(n)$, écrites en langage Python, dont l'argument est un entier $n \in \mathbb{N}^*$ et qui retournent respectivement V_n et W_n .
9. On suppose connue une troisième fonction $X(n)$ qui retourne la valeur de X_n définie au [7.] et on donne la fonction suivante.

```
def erreur(v, ve):  
    return 100*(v-ve)/ve
```

- 9.a. Expliquer le sens de la valeur retournée par l'instruction `erreur(x, x0)`. On interprétera en particulier le signe de cette valeur.
- 9.b. On considère le code Python suivant.

```
from math import pi  
  
N = 10000  
x0 = pi**2/6  
L = [ S(N), V(N), W(N), X(N) ]  
for x in L:  
    print(x, erreur(x, x0))
```

L'exécution de ce code donne le tableau suivant.

Somme partielle	Erreur
1,644834071848065	-0,006078967064804067
1,644934071848065	3,0395373552964336e-07
1,644934066848065	-9,813550040324634e-12
1,6449340668482317	3,2396863957055187e-13

Commenter ces résultats.

Solution ✿ Télescopes et accélération de convergence

Partie A. Estimations

1. a. On calcule U_n de proche en proche, en partant de la valeur initiale $U_0 = 0$ (somme vide!) et en appliquant la relation de récurrence :

$$\forall 1 \leq k \leq n, \quad U_k = U_{k-1} + \frac{1}{k^2}.$$

La traduction en code Python est alors immédiate.

```
def S(n):
    s = 0.0                # U_0 = 0
    for k in range(1, n+1): # ∀ 1 ≤ k ≤ n
        s = s + 1/(k*k)     # U_k = U_{k-1} + 1/k^2
    return s              # U_n
```

1. b. À chaque itération, on effectue deux multiplications et une addition.

Comme on effectue n itérations, on effectue en tout $2n$ multiplications et n additions. La complexité en temps est donc linéaire (= le nombre d'opérations effectuées est en $\mathcal{O}(n)$).

La complexité en espace est constante (on n'utilise que deux variables locales).

1. c. Par définition, la différence entre la somme S et la somme partielle U_n est égale au reste de la série. On en déduit donc que

$$S - U_n = \sum_{k=n+1}^{+\infty} \frac{1}{k^2} \sim \frac{1}{n}$$

et qu'il faut donc prendre $n \approx 10^3$ pour que U_n soit une valeur approchée de S à 10^{-3} près.

REMARQUE.— Pour donner une valeur plus précise des entiers n convenables, il faudrait reprendre la comparaison de somme et d'intégrale pour trouver un *majorant explicite* de $|S - U_n|$. (On a perdu pas mal d'information en passant de l'encadrement à l'équivalent...)

Partie B. Accélération de convergence

2. D'après [1.c.],

$$S - U_n \sim \frac{1}{n} \quad \text{tandis que} \quad S - V_n = o\left(\frac{1}{n}\right)$$

donc l'erreur $(S - V_n)$ est infiniment plus petite que l'erreur $(S - U_n)$ lorsque n tend vers $+\infty$.

3. S'il existe une suite $(v_k)_{k \geq 1}$ telle que

$$\forall n \geq 1, \quad V_n = \sum_{k=1}^n v_k,$$

alors $v_1 = V_1$ et

$$\forall n \geq 2, \quad v_n = V_n - V_{n-1}$$

Réciproquement, si $v_1 = V_1$ et $v_n = V_n - V_{n-1}$ pour tout $n \geq 2$, alors on a bien

$$\forall n \geq 1, \quad V_n = \sum_{k=1}^n v_k.$$

Il existe donc bien une, et une seule, suite $(v_k)_{k \geq 1}$ telle que

$$\forall n \geq 1, \quad V_n = \sum_{k=1}^n v_k.$$

✿ D'après ce qui précède, $v_1 = U_1 + 1 = 2$ et, pour tout entier $n \geq 2$,

$$v_n = U_n - U_{n-1} + \frac{1}{n} - \frac{1}{n-1} = \frac{1}{n^2} - \frac{1}{n(n-1)} = \frac{-1}{n^2(n-1)}.$$

4. On en déduit que

$$v_k \underset{k \rightarrow +\infty}{\sim} -1/k^3.$$

5. La série de terme général positif $\sum 1/k^3$ est convergente. On déduit de l'équivalent précédent que le reste

$$S - V_n = \sum_{k=n+1}^{+\infty} v_k$$

est équivalent, lorsque n tend vers $+\infty$, au reste

$$- \sum_{k=n+1}^{+\infty} \frac{1}{k^3}.$$

Or, d'après la propriété rappelée par l'énoncé,

$$- \sum_{k=n+1}^{+\infty} \frac{1}{k^3} \sim \frac{-1}{2n^2}$$

donc

$$S - V_n \sim \frac{-1}{2n^2}$$

lorsque n tend vers $+\infty$.

6. a. Le même raisonnement qu'au [3.] s'applique. On en déduit que $w_1 = V_1 - 1/2 = 3/2$ et que, pour tout entier $k \geq 2$,

$$w_k = v_k - \frac{1}{2k^2} + \frac{1}{2(k-1)^2} = \frac{1}{2k^2(k-1)^2}.$$

6. b. On reprend le même raisonnement! Lorsque k tend vers $+\infty$,

$$w_k \sim \frac{1}{2k^4}.$$

Or la série de terme général positif $\sum 1/k^4$ est convergente, donc le reste

$$S - W_n = \sum_{k=n+1}^{+\infty} w_k$$

est équivalent, lorsque n tend vers $+\infty$, au reste

$$\sum_{k=n+1}^{+\infty} \frac{1}{2k^4}.$$

Or, d'après la propriété rappelée par l'énoncé,

$$\sum_{k=n+1}^{+\infty} \frac{1}{k^4} \sim \frac{1}{3n^3}$$

donc

$$S - W_n \sim \frac{1}{6n^3}$$

lorsque n tend vers $+\infty$.

7. On applique à nouveau la même méthode : on considère la suite de terme général

$$x_n = W_n + \frac{1}{6n^3}$$

et on explicite la suite $(x_k)_{k \geq 1}$ telle que

$$\forall n \geq 1, \quad x_n = \sum_{k=1}^n x_k.$$

On trouve, pour tout entier $k \geq 2$,

$$x_k = w_k + \frac{1}{6k^3} - \frac{1}{6(k-1)^3} = \frac{-1}{6k^3(k-1)^3}$$

donc

$$x_k \sim \frac{-1}{6k^6}$$

lorsque k tend vers $+\infty$. On en déduit cette fois que le reste $S - X_n$ est équivalent à

$$\frac{-1}{6} \sum_{k=n+1}^{+\infty} \frac{1}{k^6} \sim \frac{-1}{30n^5}$$

ce qui est encore plus précis que ce qu'on espérait...

Partie C. Exploitation des résultats

8. On utilise la fonction $S(n)$ définie plus haut.

```
def V(n):
    return S(n)+1/n

def W(n):
    return V(n)-1/(2*n*n)
```

Lorsque l'entier n devient grand, on peut considérer que l'exécution des fonctions $V(n)$ et $W(n)$ effectuée à très peu de chose près le même nombre de calculs que l'exécution de $S(n)$. On obtient ainsi un résultat sensiblement plus précis pour le même prix.

9. D'après le développement asymptotique du reste donné par l'énoncé :

```
def X(n):
    return W(n)+1/(6*n**3)
```

et l'erreur $S - X_n$ est équivalente à $-1/30n^5$.

9.a. On considère ici x comme une valeur approchée de la valeur exacte x_0 et l'exécution de `erreur(x, x0)` retourne l'erreur relative exprimée en pourcentage.

Si l'erreur relative est positive (resp. négative), alors la valeur de x est une valeur approchée par excès (resp. par défaut) de x_0 .

9.b. D'après [1.c.], pour $N = 10^4$, l'erreur absolue est de l'ordre de 10^{-4} et l'erreur relative est donc de l'ordre de $6 \cdot 10^{-4} / \pi^2$, soit $6 \cdot 10^{-5}$ (on rappelle que $\pi^2 \approx 10$). C'est conforme à ce qu'on lit sur la première ligne (erreur par défaut).

Sur la deuxième ligne, on constate que la quatrième décimale augmente d'une unité : on a ajouté $1/N = 10^{-4}$ à U_n . L'erreur absolue est maintenant de l'ordre de $5 \cdot 10^{-9}$ d'après [5.], donc l'erreur relative est de l'ordre de $3 \cdot 10^{-9}$ (puisque $5 \times 6 / \pi^2 \approx 3$). Là encore, les valeurs observées sont conformes à la théorie (y compris sur le fait qu'il s'agisse ici d'une erreur par excès).

Sur la troisième ligne, on a retranché $1/2N^2 = 5 \cdot 10^{-7}$ à V_n . On obtient une valeur approchée par défaut et l'ordre de grandeur de l'erreur commise calculé au [6.b.] est encore conforme aux valeurs observées.

Sur la dernière ligne, surprise ! On s'attend à une erreur absolue en $\mathcal{O}(1/N^5)$ (équivalente à $-1/30N^5$ d'après la formule donnée par l'énoncé à la fin de la partie B), ce qui correspondrait à une erreur relative d'environ

$$100 \cdot \frac{1}{3 \cdot 10^{5 \times 4 + 1}} \cdot \frac{6}{\pi^2} \approx 2 \cdot 10^{-20} \%$$

alors qu'on n'observe qu'une erreur relative de l'ordre de $3 \cdot 10^{-13} \%$! L'explication est toute simple et tient à la représentation des flottants sur huit octets (ou 64 bits) :

$$X_N = W_N + \frac{1}{6N^4} \approx W_n + 2 \cdot 10^{-17}$$

et la correction ajoutée est tellement petite par rapport à W_N qu'elle est tronquée par Python !

Cela nous indique comment exploiter utilement l'étude précédente : il est *vain* de chercher à obtenir une précision infinie lorsqu'on travaille avec des flottants codés sur huit octets ; il est *intéressant* de limiter au minimum les calculs effectués pour atteindre une précision donnée.

Ainsi, pour $N = 500$ seulement, l'erreur relative commise en approchant $\pi^2/6$ par X_N est de l'ordre de $5 \cdot 10^{-14} \%$. La précision du calcul est donc la même en calculant X_{500} ou X_{10^4} , alors que le calcul de X_{10^4} demande vingt fois plus d'opérations que celui de X_{500} ! Vingt fois plus cher pour la même qualité, je n'achète pas !