Programmation sous `python` – calcul scientifique

2025-2026

Une solution pour l'exercice n° 1 de la feuille n° 4

On définit la suite (u_n) par : $u_n = \text{pgcd}\left(n^2+4n+3, n^3+3n-5\right)$ pour tout $n \in \mathbb{N}$.

1) Définir une fonction `u` prenant en paramètre un nombre entier n et renvoyant le terme u_n. On pourra utiliser la fonction `gcd` du module `math`, ou voir le point

2) Afficher les 30 premiers termes de la suite, comme dans la figure précédente. Cette suite est-elle périodique ? Si oui, déterminer sa période.

3) Afficher quelques termes suivants et conclure.

4) Pour faire une liaison avec le cours d'ariméthique, écrire une fonction `euclide_bezout` qui prend en arguments deux entiers a et b et qui renvoie les entiers d, u et v, où $d = \text{pgcd}(a,b)$ et $ua+vb=d$.

Solution.

1)

```python
def pgcd(a: int, b: int):
    """
    a : int
    b : int

    a = q * b + r  (la division euclidienne de a par b)
    a = b, b = r
    et on continue jusqu'a r = 0

    Renvoie le pgcd de a et de b, le dernier reste different de 0
    """
    test = isinstance(a, int) and isinstance(a, int)
    if not test:
        return -0.5
    while b != 0:
        r = a % b
        a = b
        b = r
    return abs(a)
```

2)

```python
def suite_u(n):
    """
    Renvoie le pgcd des deux nombres construits a partir de l'entier
    n >= 1.
    """
    a = n**2 + 4 * n + 3
    b = n**3 + 3 * n - 5
    return pgcd(a, b)

N = 100
X = [k for k in range(1, N)]
Y = [suite_u(k) for k in X]

fig = plt.figure(figsize=(10, 9))
```

```
axe = fig.add_subplot(111)
axe.plot(X, Y, marker='o', ls='--', lw=.5, ms=3, c='b')
axe.set_xlabel("n l'indice")  # gca() get current axes
axe.set_ylabel("u la suite")
axe.set_title("etude d'une suite non periodique")
axe.grid()
```

3) Concernant la périodicité, les premiers tests montrent que de nouvelles valeurs apparaissent après la borne 30. On écrit une fonction pour étudier les images de u et déterminer leurs premières apparitions.

```
def premieres_apparitions(N):
    """
    N : entier, la borne jusqu'a laquelle on cherche des
    elements nouveaux dans l'image de u

    Renvoie deux listes :
    -- la deuxieme est composee des premieres apparitions
    des elements de l'image de u
    --  la preumiewre de listes [indice, u] correspondants a
    ces premieres apparitions.
    """
    u = suite_u(1)
    les_u = [u]
    res = [[1, u]]
    for k in range(2, N + 1):
        u = suite_u(k)
        if u not in les_u:
            res.append([k, u])
            les_u.append(u)
    return res, les_u
```

Pour $N = 1000$ ou $N = 10000$, la première liste renvoyée par la fonction est

```
[[1, 1], [2, 3], [8, 9], [38, 123], [79, 41], [161, 369]]
```

En dessinant ses 1000 premières valeurs, la suite $(u_n)_{n\geq 1}$ semble être périodique. On cherche un argument mathématique pour justifier cette hypothèse.

Argument théorique. On remarque que

$$n^2 + 3n + 4 = (n + 1)(n + 3)$$

et qu'on a les divisions euclidiennes de la deuxième composante de u_n :

$$n^3 + 3n - 5 = (n^2 - n + 4)(n + 1) - 9$$
$$n^3 + 3n - 5 = (n^2 - 3n + 12)(n + 3) - 41.$$

Trois conséquences s'ensuivent :

- $\text{pgcd}(n + 1, n^3 + 3n - 5)$ peut être 1, 3 ou 9
- $\text{pgcd}(n + 3, n^3 + 3n - 5)$ peut être 1 et 41
- l'identité

$$\text{pgcd}(n^2 + 3n + 4, n^3 + 3n - 5) = \text{pgcd}(n + 1, n^3 + 3n - 5)\,\text{pgcd}(n + 3, n^3 + 3n - 5)$$

car 9 **et** 41 **sont premiers entre eux**.

On en déduit qu'il y a six possibilités pour le pgcd recherché :

$$1. \quad 3, \quad 9, \quad 41, \quad 3 \cdot 41 = 123 \quad \text{et} \quad 9 \cdot 41 = 369.$$

Pour finir, il nous reste à déterminer les n pour lesquels le pgcd prend chacune de ces six valeurs et déduire que $(u_n)_n$ est périodique.

PGCD $= 369$. En utlisant les trois conséquences ci-dessus, on est dans cette situation si et seulement si

$$n + 1 = 0 \mod 9 \qquad \text{et} \qquad n + 3 = 0 \mod 41$$

c'est-à-dire

$$n = -1 \mod 9 \qquad \text{et} \qquad n = -3 \mod 41. \tag{♯}$$

Mais 9 et 41 vérifient (la relation de Bezout)

$$(-9) \cdot 9 + 2 \cdot 41 = 1.$$

Par conséquent, d'après le théorème des restes chinois, les n qui vérifient (♯) sont les entiers définis par

$$n = (-3) \cdot (-9) \cdot 9 + (-1) \cdot 2 \cdot 41 = 3 \cdot 81 - 82 = 161 \mod 369.$$

PGCD $= 123$. On utilise le même argument : on est dans cette situation si et seulement si

$$n = -1 \mod 3 \qquad \text{et} \qquad n = -3 \mod 41. \tag{♭}$$

Mais

$$14 \cdot 3 + (-1) \cdot 41 = 1,$$

donc, les n qui vérifient (♭) sont les entiers définis par

$$n = (-3) \cdot 14 \cdot 3 + (-1) \cdot (-1) \cdot 41 = (-3) \cdot 42 + 41 = -85 = 38 \mod 123.$$

Par conséquent, on a

$$n = 38 \mod 123 \qquad \text{et} \qquad n \neq 161 \mod 369.$$

PGCD $= 41$. En développant les mêmes calculs, on obtient

$$n = 38 \mod 41 \qquad \text{et} \qquad n \neq 38 \mod 123.$$

PGCD $= 9$. On arrive à

$$n = 8 \mod 9 \qquad \text{et} \qquad n \neq 161 \mod 369.$$

PGCD $= 3$. On arrive à

$$n = 2 \mod 3 \qquad \text{et} \qquad n \neq 38 \mod 41.$$

PGCD $= 1$. On arrive à

$$n \neq 2 \mod 3 \qquad \text{et} \qquad n \neq 38 \mod 41.$$

En conclusion, la suite u_n est périodique de période principale $T = 369$. De plus, pour tout $n \in \mathbb{N}^*$,

$$u_n \in \{1, 3, 9, 41, 123, 369\}.$$

Les indices des premières apparitions de ces valeurs sont 1, 2, 8, 79, 38 et, respectivement, 161 ce qui confirme nos remarques numériques de départ.

4) On a besoin de se rappeller le théorème de Bezout pour le pgcd et de son preuve (de la première partie).

Théorème (Bezout)**.** *Soient $a, b \in \mathbb{Z}$ deux entiers non nuls.*

(i) *Si $d = \text{pgcd}(a, b)$, alors il existe $u, v \in \mathbb{Z}$ uniques tels que*

$$\begin{gathered} d = ua + vb \\ -|b| + 1 \leq u \leq |b| - 1 \qquad -|a| + 1 \leq v \leq |a| - 1. \end{gathered} \tag{$\star$}$$

(ii) $\text{pgcd}(a, b) = 1$ *si et seulement s'il existe $u, v \in \mathbb{Z}$ uniques tels que* ($\star$).

Démonstration. En posant $r_0 = a$ et $r_1 = b$, par l'algorithme d'Euclide on a

$$\begin{aligned} r_0 &= q_2\, r_1 + r_2 && \text{où} \quad 0 < r_2 < r_1 \\ r_1 &= q_3\, r_2 + r_3 && \text{où} \quad 0 < r_3 < r_2 \\ r_2 &= q_4\, r_3 + r_4 && \text{où} \quad 0 < r_4 < r_3 \\ &\vdots \\ r_{n-3} &= q_{n-1}\, r_{n-2} + r_{n-1} && \text{où} \quad 0 < r_{n-1} < r_{n-2} \\ r_{n-2} &= q_n\, r_{n-1} + r_n && \text{où} \quad 0 < r_n < r_{n-1} \\ r_{n-1} &= q_{n+1}\, r_n \end{aligned}$$

et $d = r_n$. Pour trouver les u et v il faut remonter l'algorithme. Explicitement, on a

$$\begin{aligned} d &= 0\, r_{n-1} + 1\, r_n \\ &= u_0\, r_{n-1} + v_0\, r_n = u_0\, r_{n-1} + v_0\, (r_{n-2} - q_n\, r_{n-1}) \\ &= v_0\, r_{n-2} + (u_0 - v_0\, q_n)\, r_{n-1} \\ &= u_1\, r_{n-2} + v_1\, r_{n-1} = u_1\, r_{n-2} + v_1\, (r_{n-3} - q_{n-1}\, r_{n-2}) \\ &= v_1\, r_{n-3} + (u_1 - v_1\, q_{n-1})\, r_{n-2} \\ &= u_2\, r_{n-3} + v_2\, r_{n-2} \\ &\vdots \\ &= u_{n-1}\, r_0 + v_{n-1}\, r_1 \end{aligned}$$

où $u_0 = 0$, $v_0 = 1$ et pour tout k, $0 \leq k \leq n - 1$,

$$\begin{cases} u_{k+1} = v_k \\ v_{k+1} = u_k - v_k\, q_{n-k}. \end{cases}$$

En posant $u = u_{n-1}$ et $v = v_{n-1}$, il s'ensuit que $d = u\, a + v\, b$. □

Un algorithme est obtenu en modifiant légérement la fonction pour le calcul du pgcd (l'algorithme d'Euclide) et en suivant la preuve du théorème de Bezout.

```python
def bezout(a: int, b: int):
    """
    a : int
    b : int

    Renvoie u et v tels que pgcd(a, b) = u*a + v*b
    """
    test = isinstance(a, int) and isinstance(a, int)
    if not test:
        return 0, 0
    R = [a, b]
    Q = []
    while R[-1] != 0:
        Q.append(R[-2] // R[-1])
        R.append(R[-2] % R[-1])
    Q = Q[: -1]  # on n'a pas besoin du dernier quotient
    U = [0]
    V = [1]
    for q in Q[:: -1]:  # on parcourt la liste Q dans le sens descendant
        u = V[-1]
        v = U[-1] - q * V[-1]
        U.append(u)
        V.append(v)
    # print("test ", U[-1] * a + V[-1] * b)
    return U[-1], V[-1]
```

Un autre algorithme serais basé sur l'unicité des deux nombres u et v dans l'énoncé du théorème. Il est **beaucoup plus simple**!

```python
def bezout_2(a, b):
    """
    a : int
    b : int

    Renvoie u et v tels que pgcd(a, b) = u*a + v*b.
    L'algorithme est base sur l'unicite de u et de v dans les intervalles
    correspondants (fournis par le theoreme de Bezout).
    """
    d = pgcd(a, b)
    for u in range(-abs(b) + 1, abs(b)):
        for v in range(-abs(a) + 1, abs(a)):
            test = u * a + v * b
            if test == d:
                return u, v
    return 0, 0
```

QUESTION. Trouver une suite $(u_n)_n$ construite comme celle de l'exercice (c'est-à-dire comme pgcd) et qui ne soit pas périodique.