

# BROUILLON - SOMMER LES CARRÉS DES CHIFFRES D'UN NATUREL

CHRISTOPHE BAL

*Document, avec son source L<sup>A</sup>T<sub>E</sub>X, disponible sur la page  
<https://github.com/bc-writing/drafts>.*

---

## Mentions « légales »

Ce document est mis à disposition selon les termes de la licence Creative Commons “Attribution - Pas d'utilisation commerciale - Partage dans les mêmes conditions 4.0 International”.



---

## TABLE DES MATIÈRES

1. Faire une tête au carré à tous les entiers naturels	1
2. Une preuve	2
3. Coder - Étudier la « période » d'un naturel	3
4. Peut-on généraliser à un exposant $p \geq 3$ ?	6

### 1. FAIRE UNE TÊTE AU CARRÉ À TOUS LES ENTIERS NATURELS

Voici un procédé facile à faire à l'aide d'une calculatrice. Considérons un entier naturel  $n$ , puis calculons la somme de ses chiffres élevés au carré. Ceci nous donne un nouveau naturel auquel on peut appliquer le même procédé. Voici deux exemples.

**Exemple 1.1.** *Pour  $n = 19$ , nous obtenons :*

- $1^2 + 9^2 = 82$
- $8^2 + 2^2 = 68$
- $6^2 + 8^2 = 100$
- $1^2 + 0^2 + 0^2 = 1 \rightarrow$  Rien de nouveau à attendre.

**Exemple 1.2.** *Pour  $n = 1\,234\,567\,890$ , après  $1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 + 7^2 + 8^2 + 9^2 + 0^2 = 285$  nous obtenons :*

- |                          |  |
|--------------------------|--|
| • $2^2 + 8^2 + 5^2 = 93$ | • $5^2 + 8^2 = 89$                             |
| • $9^2 + 3^2 = 90$       | • $8^2 + 9^2 = 145$                            |
| • $9^2 + 0^2 = 81$       | • $1^2 + 4^2 + 5^2 = 42$                       |
| • $8^2 + 1^2 = 65$       | • $4^2 + 2^2 = 20$                             |
| • $6^2 + 5^2 = 61$       | • $2^2 + 0^2 = 4$                              |
| • $6^2 + 1^2 = 37$       | • $4^2 = 16$                                   |
| • $3^2 + 7^2 = 58$       | • $1^2 + 6^2 = 37 \rightarrow$ Déjà rencontré. |

Dans le 1<sup>er</sup> cas, au bout d'un moment le procédé ne produit que des 1. Ce sera le cas dès que l'on commence avec une puissance de 10. Le 2<sup>e</sup> exemple montre que le mieux que l'on puisse espérer c'est que le procédé devienne périodique à partir d'un moment (*on parle de phénomène ultimement périodique*).

On peut explorer le comportement de ce procédé sur plusieurs valeurs grâce à un programme. Voici un code possible non optimisé<sup>1</sup> écrit en Python 3.7 qui prend un peu de temps pour vérifier que pour tous les naturels  $n \in \llbracket 1; 10^6 \rrbracket$ , le procédé devient ultimement périodique.

---

```

NMAX      = 10**6
MAXLOOP   = 10**20

for n in range(1, NMAX + 1):
    nbloops = 0
    results = []

    while nbloops < MAXLOOP and n not in results:
        nbloops += 1
        results.append(n)
        n = sum(int(d)**2 for d in str(n))

    if n not in results:
        print(f"Test raté pour n = {n}.")

print("Tests finis.")

```

---

Une fois lancé, le code précédent affiche juste `Tests finis`. Il reste à voir ce qu'il se passe dans le cas général. La section qui suit démontre que pour tout naturel  $n$ , le procédé sera toujours ultimement périodique.

## 2. UNE PREUVE

On introduit les notations suivantes.

- Pour un naturel  $n$ ,  $n = [c_{d-1}c_{d-2} \cdots c_1c_0]_{10} \stackrel{\text{def}}{=} \sum_{k=0}^{d-1} c_k 10^k$ , avec  $c_{d-1} \neq 0$ , désigne l'écriture décimale propre de  $n$ .
- On pose ensuite  $sq(n) = \sum_{k=0}^{d-1} (c_k)^2$  et  $\text{taille}(n) = d$  sera appelé « *taille de  $n$*  ».
- Pour  $(n; k) \in \mathbb{N}^2$ , on définit  $\boxed{n}_0 = n$  et  $\boxed{n}_k = sq^k(n) \stackrel{\text{def}}{=} sq \circ sq \circ \cdots \circ sq(n)$  avec  $(k-1)$  compositions si  $k > 0$ .  
Autrement dit, nous avons  $\boxed{n}_0 = n$  et  $\boxed{n}_{k+1} = sq(\boxed{n}_k)$ .
- Enfin on note  $V_n = \{\boxed{n}_k \mid k \in \mathbb{N}\}$  l'ensemble des valeurs prises par la suite  $(\boxed{n}_k)_k$ .

**Fait 2.1.**  $\forall n \in \mathbb{N}$ ,  $sq(n) \leq 81d$  où  $d = \text{taille}(n)$ .

1. Voir la section proposant des graphiques pour découvrir un code qui explore un peu plus efficacement les périodes.

**Preuve.** Si  $n = [c_{d-1}c_{d-2} \cdots c_1c_0]_{10}$  alors  $sq(n) = \sum_{k=0}^{d-1} (c_k)^2 \leq \sum_{k=0}^{d-1} 9^2 = 81d$ .

**Fait 2.2.**  $\forall n \in \mathbb{N}$ , notant  $d = \text{taille}(n)$ , nous avons les résultats suivants :

- (1) Si  $d \geq 4$  alors  $sq(n) < n$ .
- (2) Si  $d \leq 3$  alors  $sq(n) < 10^3$ .

**Preuve.** Comme  $n \geq 10^{d-1}$  et compte tenu du fait précédent, nous cherchons à comparer  $10^{d-1}$  et  $81d$ . Pour cela, regardons ce qu'il se passe pour les premières valeurs de  $d$ .

$d$	1	2	3	4	5
$10^{d-1}$	1	10	100	1000	10 000
$81d$	81	162	243	324	405

Or lorsque  $d \geq 2$  augmente de 1, alors  $81d$  augmente de 81 tandis que  $10^{d-1}$  augmente de  $9 \times 10^{d-1}$  soit d'au moins 90. En effet,  $10^d = 10 \times 10^{d-1} = 10^{d-1} + 9 \times 10^{d-1}$ .

Donc  $n \geq 10^{d-1} > 81d \geq sq(n)$ , d'où  $n > sq(n)$ , dès que  $d \geq 4$ . Ceci prouve le 1<sup>er</sup> point. Pour les fans de Nicolas B.<sup>2</sup>, voir la preuve page 6 du fait 4.2 traitant les puissances quelconques.

Le 2nd point pour  $d \leq 3$  découle directement de  $sq(999) = 243$ .

**Fait 2.3.**  $\forall n \in \mathbb{N}$ , l'ensemble  $V_n$  est fini et donc la suite  $([n]_k)_{k \in \mathbb{N}}$  est ultimement périodique, i.e. périodique à partir d'un certain rang.

**Preuve.** Le 2nd point dépend directement du 1er point via le principe des tiroirs et la définition récursive de la suite  $([n]_k)_k$ .

Pour le 1er point, pour  $n \leq 999$ , on a directement  $V_n \subset [0; 999]$ , sinon il suffit de montrer que  $V_n \subset [0; 10^{\text{taille}(n)}]$  pour  $n \geq 10^4$  via une petite récurrence descendante finie.

### 3. CODER - ÉTUDIER LA « PÉRIODE » D'UN NATUREL

Quand il ne se fige pas, le code suivant donne la « période » d'un naturel auquel on applique le procédé.

---

```

n      = 20181209
nmemo = n

results = []

while n not in results:
    results.append(n)
    n = sum(int(d)**2 for d in str(n))

print(f"{nmemo} a la période suivante :")
print(results[results.index(n):])

print()
```

---

2. Alias Nicolas BOURBAKI.

```

before = results[:results.index(n)]

if before:
    print("Avant la 1ère période nous avons :")
    print(before)
else:
    print("On commence directement par la période.")

```

---

Le code précédent, où  $n = 20181209$ , nous affiche :

---

20181209 a la période suivante :  
 [16, 37, 58, 89, 145, 42, 20, 4]

Avant la 1ère période nous avons :  
 [20181209, 155, 51, 26, 40]

---

Amusons-nous maintenant à représenter un histogramme des tailles des « périodes ». À l'adresse <https://github.com/bc-writing/drafts>, dans le dossier `squares-int`, vous trouverez le fichier `squareint-sizeplots.py` qui été utilisé pour obtenir le graphique<sup>3</sup>. Le traitement des

---

3. À la même adresse dans le dossier `squares-int` se trouve l'image `before.png` qui est un histogramme des nombres de termes calculés avant l'apparition de la 1<sup>re</sup> « période ».

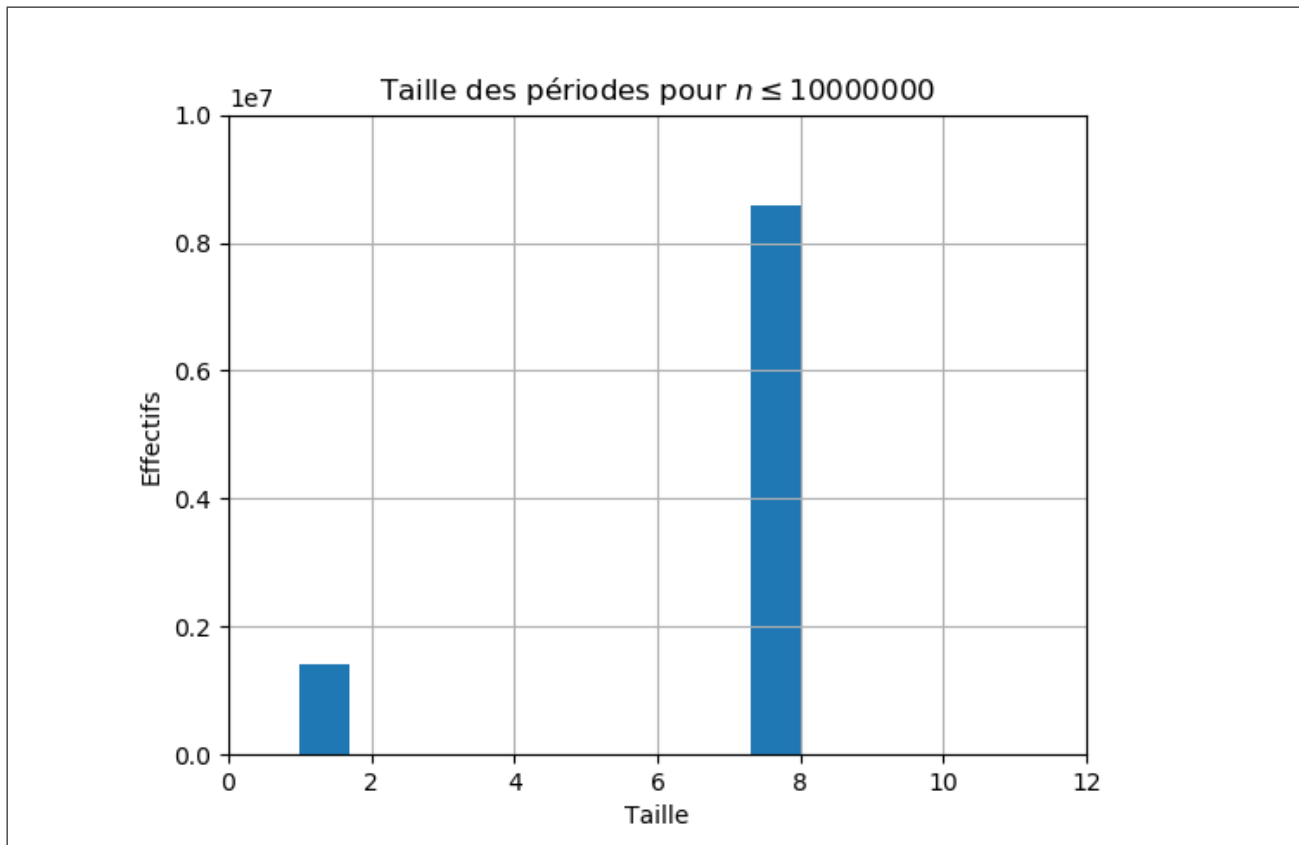


FIGURE 1: Histogramme des tailles des périodes

données a été amélioré pour éviter de refaire des calculs déjà rencontrés (*pour plus de précisions, se reporter aux commentaires du code*). Le résultat est donné dans la figure 1 page 4.

Voilà quelque chose de frappant ! Il semblerait que l'on ait soit des périodes de taille 1, penser à 0 et 1, soit des périodes de taille 8 comme pour  $37 - 58 - 89 - 145 - 42 - 20 - 4 - 16 \dots$ . Magie ou coïncidence ? Les résultats de la section 2 vont nous permettre de le savoir. Dans la suite, nous reprenons les notations de la dite section.

Tout d'abord, comme  $\text{taille}(sq(n)) < \text{taille}(n)$  dès que  $\text{taille}(n) \geq 4$  d'après le fait 4.2, la périodicité n'arrivera que lorsque  $\text{taille}(\lfloor n \rfloor_k) \leq 3$ .

De plus, nous savons aussi que  $\text{taille}(sq(n)) \leq 3$  dès que  $\text{taille}(n) \leq 3$ .

Tout ceci nous permet d'analyser brutalement via un programme ce qu'il se passe pour les périodes des naturels appartenant à  $\llbracket 0; 999 \rrbracket$ . Nous pouvons pour cela utiliser le code suivant, qui n'est absolument pas optimisé mais fait le travail immédiatement.

---

```
nmax = 999

periodsfound = []

for n in range(nmax + 1):
    results = []

    while n not in results:
        results.append(n)
        n = sum(int(d)**2 for d in str(n))

    period = results[results.index(n):]

    if period not in periodsfound:
        periodsfound.append(period)

for oneperiod in periodsfound:
    print(oneperiod)
```

---

Le code précédent nous fournit toutes les périodes possibles.

---

```
[0]
[1]
[4, 16, 37, 58, 89, 145, 42, 20]
[37, 58, 89, 145, 42, 20, 4, 16]
[89, 145, 42, 20, 4, 16, 37, 58]
[16, 37, 58, 89, 145, 42, 20, 4]
[20, 4, 16, 37, 58, 89, 145, 42]
[58, 89, 145, 42, 20, 4, 16, 37]
[42, 20, 4, 16, 37, 58, 89, 145]
[145, 42, 20, 4, 16, 37, 58, 89]
```

---

Et là cela devient joli car nous notons au passage que trois types de périodes :  $[0]$ ,  $[1]$  et  $[4, 16, 37, 58, 89, 145, 42, 20]$  avec toutes ses « *permutées circulaires* ».

#### 4. PEUT-ON GÉNÉRALISER À UN EXPOSANT $p \geq 3$ ?

Pour finir, nous allons analyser ce qu'il se passe si l'on somme à la puissance  $p \geq 3$  au lieu d'élever au carré. Nous reprenons des notations similaires à celles de la section 2.

- Pour un naturel  $n = [c_{d-1}c_{d-2} \cdots c_1c_0]_{10}$  avec  $c_{d-1} \neq 0$ , on pose  $pw(n) = \sum_{k=0}^{d-1} (c_k)^p$  et  $\text{taille}(n) = d$ .
- Pour  $(n; k) \in \mathbb{N}^2$ , on définit  $[n]_0 = n$  et  $[n]_{k+1} = pw([n]_k)$ .

**Fait 4.1.**  $\forall n \in \mathbb{N}$ ,  $pw(n) \leq 9^p d$  où  $d = \text{taille}(n)$ .

**Preuve.** Si  $n = [c_{d-1}c_{d-2} \cdots c_1c_0]_{10}$  alors  $pw(n) = \sum_{k=0}^{d-1} (c_k)^p \leq \sum_{k=0}^{d-1} 9^p = 9^p d$ .

**Fait 4.2.** Il existe  $d_0 \in \mathbb{N}$  tel que  $\forall n \in \mathbb{N}$ ,  $[\text{taille}(n) \geq d_0 \Rightarrow pw(n) < n]$ .

**Preuve.** Notons  $d = \text{taille}(n)$  de sorte que  $n \geq 10^{d-1}$ . Compte tenu du fait précédent, nous cherchons à comparer  $10^{d-1}$  et  $9^p d$ .

Nous allons procéder de façon analogue au cas  $p = 2$  démontré dans la section 2 mais en étant ici plus rigoureux dans la rédaction.

$\exists d_0 \in \mathbb{N}^*$  tel que  $10^{d_0-1} > 9^p d_0 > 9^{p-1}$ . Ceci s'obtient en utilisant la croissance comparée des fonctions  $f(x) = 10^{x-1}$  et  $g(x) = 9^p x$ .

Montrons par récurrence sur  $d \geq d_0$  que  $10^{d-1} > 9^p d$ . Ceci donnera  $n \geq 10^{d-1} > 9^p d \geq pw(n)$  d'où  $n > pw(n)$  dès que  $d \geq d_0$  comme souhaité.

- Initialisation. Par choix de  $d_0$ , nous avons  $10^{d-1} > 9^p d$  si  $d = d_0$ .
- Hérité. Faisons l'hypothèse que  $10^{d-1} > 9^p d$  est vérifiée pour un naturel  $d \geq d_0$  « fixé quelconque ».

Nous avons :  $10^{(d+1)-1} = 10 \times 10^{d-1} = 10^{d-1} + 9 \times 10^{d-1} > 10^{d-1} + 9^p$  car  $10^{d-1} > 9^{p-1}$  puisque  $d \geq d_0$ .

Comme  $10^{d-1} > 9^p d$ , nous avons ensuite :  $10^{(d+1)-1} > 9^p d + 9^p = 9^p(d+1)$ . L'inégalité est donc vérifiée au rang suivant  $(d+1)$ .

- Conclusion. Par récurrence sur  $d \geq d_0$ , nous avons  $10^{d-1} > 9^p d$  pour tout naturel  $d$  tel que  $d \geq d_0$ .

**Remarque 4.1.** Informatiquement une valeur de  $d_0$  peut s'obtenir en testant  $10^{d-1} > 9^p d$  successivement pour les naturels non nuls  $d$ .

Pour gagner du temps, on peut tester les valeurs successives de  $2^k d'$  pour  $k = 0, 1, 2, \dots$  pour obtenir  $D$  tel que  $10^{D-1} > 9^p D$ . Si la valeur de  $D$  est trop grande pour faire des tests brutaux, on peut chercher la valeur minimale de  $d$  tel que  $10^{d-1} > 9^p d$  en utilisant une recherche de type dichotomique.

**Fait 4.3.**  $\forall n \in \mathbb{N}$ , la suite  $(\boxed{n}_k)_{k \in \mathbb{N}}$  est ultimement périodique.

**Preuve.** Tout est en fait contenu dans le fait 4.2, dont on reprend la signification de  $d_0$ . Expliquons pourquoi.

- Le fait 4.2 donne l'existence d'un indice  $k_0 \in \mathbb{N}$  tel que  $\text{taille}(\boxed{n}_{k_0}) < d_0$  (dans le cas contraire, on pourrait construire une suite strictement décroissante de naturels).
- Si pour tout naturel  $k \in \llbracket k_0; +\infty \llbracket$ ,  $\text{taille}(\boxed{n}_k) < d_0$ , nous avons l'ultime périodicité via le principe des tiroirs (si besoin revoir la fin de la section 2).
- Sinon il existe  $k'_0 \in \llbracket k_0; +\infty \llbracket$  tel que  $\text{taille}(\boxed{n}_{k'_0}) \geq d_0$ . Comme dans le premier point, nous pouvons alors trouver  $k_1 \in \llbracket k'_0; +\infty \llbracket$  tel que  $\text{taille}(\boxed{n}_{k_1}) < d_0$ .
- En répétant notre raisonnement, on peut aboutir à une situation similaire au 2<sup>e</sup> point, et c'est gagné.

Sinon on arrive à construire une suite strictement croissante  $(k_i)_i$  d'indices tels que  $\forall i \in \mathbb{N}$ ,  $\text{taille}(\boxed{n}_{k_i}) < d_0$ . Le principe des tiroirs s'applique ici aussi !

**Remarque 4.2.** La preuve précédente montre que pour rechercher toutes les périodes il « suffit » d'étudier les naturels appartenant à  $\llbracket 0; 10^{d_0} \llbracket$ .