



HAL
open science

Un ordinateur, ça ne calcule jamais juste

Mathieu Acher

► **To cite this version:**

| Mathieu Acher. Un ordinateur, ça ne calcule jamais juste. 2024. hal-04736121

HAL Id: hal-04736121

<https://hal.science/hal-04736121v1>

Submitted on 14 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Un ordinateur, ça ne calcule jamais juste

Mathieu Acher

<https://www.mathieuacher.com> @acherm



INSA



institut
universitaire
de France

Un ordinateur, ça ne calcule jamais juste...



Ou pourquoi les ordinateurs nous trompent parfois

Un ordinateur ne calcule jamais parfaitement. Par exemple, si on additionne trois nombres de différentes façons, le résultat peut changer selon le langage de programmation ou l'ordinateur utilisé. Cela peut même amener des robots, voitures ou avions à prendre des chemins différents selon les conditions.

Ou à votre application préférée de retourner une réponse, une image ou une vidéo différente !

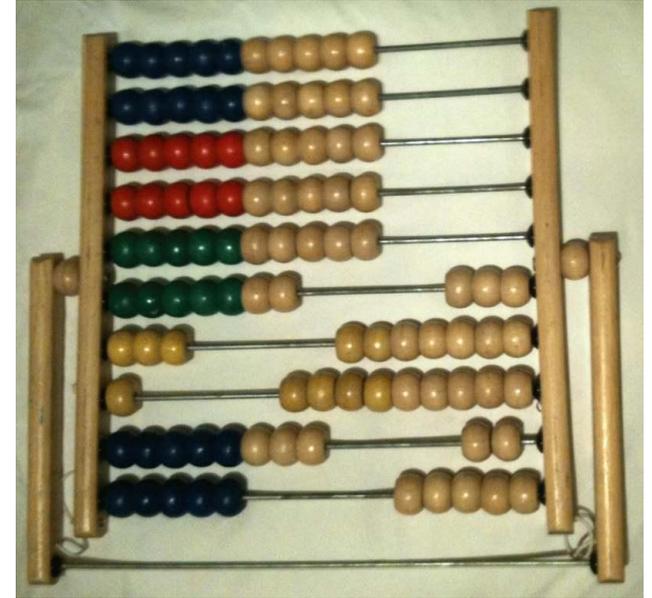
En informatique, comme dans toutes les sciences, il est important de comprendre que les calculs ont toujours une petite marge d'erreur.

Ce qui compte, ce n'est pas la perfection, mais d'en être conscient, de savoir s'adapter et corriger le tir quand c'est nécessaire !

Un ordinateur ?



La machine d'Anticythère, appelée également mécanisme d'Anticythère, est considérée comme le premier calculateur analogique antique permettant de calculer des positions astronomiques. (-200 av. J.-C.)



Les bouliers permettent d'effectuer les quatre opérations : additions, soustractions, multiplications et divisions... et un peu plus (-2500 av. J.-C.)



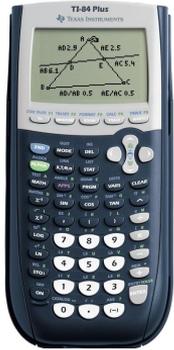
petite calculatrice mécanique produite entre 1948 et 1972 (ancêtre de la calculatrice électronique)

Les ordinateurs sont partout



Les ordinateurs sont partout

...dans nos poches, nos maisons, nos hôpitaux, et même dans les avions ou nos voitures.



Ordinateur

“computer” en anglais qui peut se traduire par calculateur...

“système de **traitement de l'information programmable** tel que défini par Alan Turing et qui fonctionne par la lecture séquentielle d'un ensemble d'instructions, organisées en **programmes**, qui lui font exécuter des opérations logiques et arithmétiques” **Wikipedia**



Une voiture = un ordinateur avec des roues !?



- Système d'**information** divertissement : Il gère la radio, l'intégration des smartphones, la navigation GPS, et les applications.
- **Capteurs** de conduite : De nombreux capteurs surveillent des actions comme le freinage et la direction
- Systèmes **perceptuels** : Les caméras, le radar et parfois le LiDAR permettent à la voiture de percevoir son environnement
- **Données** diagnostiques : Les informations sur la performance du moteur, l'état de la batterie (pour les véhicules électriques) et autres paramètres sont collectées
- **Données** biométriques : Des capteurs internes surveillent le comportement du conducteur pour assurer que ce dernier reste concentré et éveillé

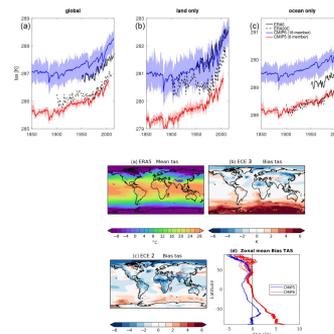
Ces systèmes reposent sur des unités de contrôle électronique (ECU) et des millions de lignes de code, transformant la voiture en une plateforme numérique mobile.

L'Informatique au centre de notre société et de la science

Les ordinateurs et les logiciels sont partout : dans nos poches, nos maisons, nos hôpitaux, et même dans les avions ou nos voitures.

Grâce aux ordinateurs et aux logiciels, les scientifiques peuvent analyser des données climatiques en temps réel et faire des prévisions météorologiques plus précises, ce qui aide à prévenir les catastrophes naturelles.

Prix Nobel de Physique et de Chimie 2024 pour des... informaticiens !



Mais comment fonctionnent les ordinateurs (et les programmes/logiciels) et pourquoi peuvent-ils se tromper et nous tromper ?

$$1 + 2 + 3 =$$

$$0.1 + 0.2 + 0.3 =$$

Vérifions à la calculatrice...

Vous êtes sûrs ? certains ?

Est-ce que $0.1 + (0.2 + 0.3) = (0.1+0.2)+0.3$?

Vérifions à la calculatrice...

Associativité...

Vous êtes sûrs ? certains ?

$$0.1 + 0.2 + 0.3 = ?$$

Demandons à un ordinateur (un “vrai”), pas une calculatrice...

On pourrait utiliser la calculatrice (une application) de l'ordinateur...

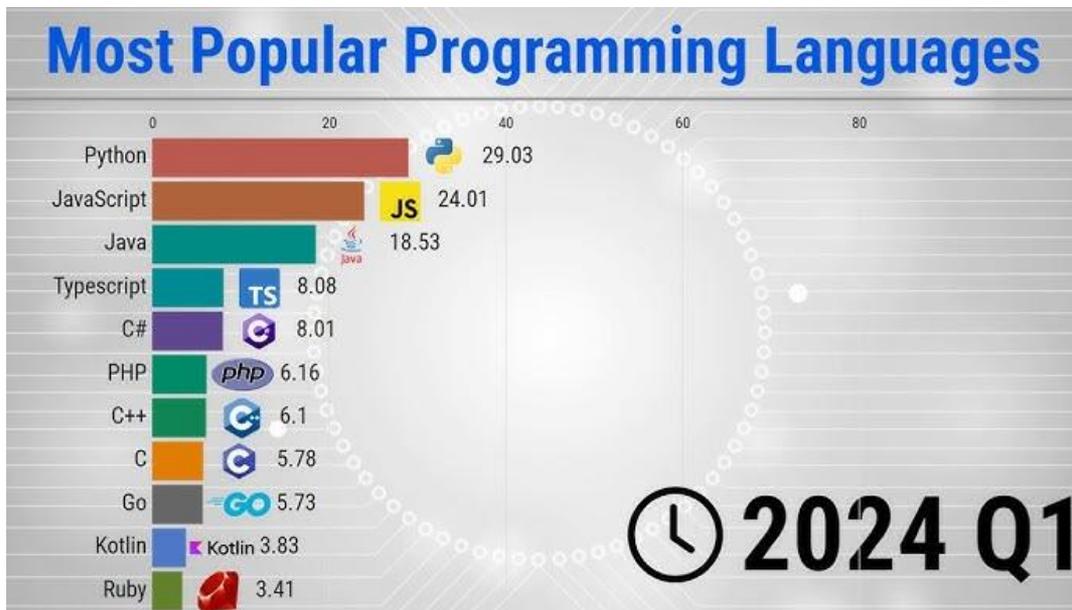
Essayons de demander en programmant

0.1 + 0.2 + 0.3 = ?

Essayons de demander en programmant



Il nous faut un langage de programmation. Prenons le plus populaire !



$$0.1 + (0.2 + 0.3) = ?$$

avec Python (programmation!)

```
~/SANDBOX/prezINSA-fetesciences2425 via  v3.10.14 took 1m25s  
> python  
Python 3.10.14 (main, Apr 25 2024, 18:59:54) [GCC 13.2.1 20240316 (Red Hat 13.2.1-7)] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 0.1 + (0.2 + 0.3)  
0.6  
>>> █
```

0.1 + 0.2 + 0.3 = ?

avec Python (programmation!)

```
~/SANDBOX/prezINSA-fetesciences2425 via 🐙 v3.10.14 took 1m16s  
> python  
Python 3.10.14 (main, Apr 25 2024, 18:59:54) [GCC 13.2.1 20240316 (Red Hat 13.2.1-7)] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 0.1 + 0.2 + 0.3  
0.6000000000000001  
>>> (0.1 + 0.2) + 0.3  
0.6000000000000001  
>>> █
```

$$(0.1 + 0.2) + 0.3 = 0.1 + (0.2 + 0.3)$$

avec Python (programmation!)

```
~/SANDBOX/prezINSA-fetesciences2425 via 🐞 v3.10.14 took 7s  
> python  
Python 3.10.14 (main, Apr 25 2024, 18:59:54) [GCC 13.2.1 20240316 (Red Hat 13  
2.1-7)] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> (0.1 + 0.2) + 0.3 == 0.1 + (0.2 + 0.3)  
False  
>>> █
```

$$(1 + 2) + 3 = 1 + (2 + 3)$$

avec Python (programmation!)

```
~/SANDBOX/prezINSA-fetesciences2425 via  v3.10.14 took 29s  
> python  
Python 3.10.14 (main, Apr 25 2024, 18:59:54) [GCC 13.2.1 20240316 (Red Hat 13  
2.1-7)] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 1+2+3  
6  
>>> 1+(2+3)  
6  
>>> (1+2)+3  
6  
>>> 1+(2+3) == (1+2)+3  
True  
>>> █
```

$$(0.1 + 0.2) + 0.3 = 0.1 + (0.2 + 0.3) = 6/10 ?$$

Les approximations sont utilisées pour représenter les nombres en programmation parce que les ordinateurs ne peuvent pas stocker une **infinité de chiffres**. Ils utilisent un espace limité pour chaque nombre, ce qui peut entraîner des petites erreurs. Les nombres sont également représentés avec des 0 et des 1, et les calculs complexes peuvent parfois introduire des écarts infimes mais inévitables.

$$(0.1 + 0.2) + 0.3 = 0.1 + (0.2 + 0.3) = 6/10 ?$$

Les approximations sont utilisées pour représenter les nombres en programmation parce que les ordinateurs ne peuvent pas stocker une infinité de chiffres. Ils utilisent un espace limité pour chaque nombre, ce qui peut entraîner des petites erreurs.

Prenons un exemple avec le **nombre 1/3**. En mathématiques, ce nombre s'écrit comme une fraction, mais en décimal, il devient 0,3333... avec une infinité de 3 après la virgule.

Un ordinateur ne peut pas stocker un nombre avec une infinité de chiffres, alors il est obligé d'en prendre une partie seulement, comme 0,3333333333. Cela signifie que l'ordinateur travaille avec une version approximative de 1/3. Par exemple, si tu multiplies 1/3 par 3 dans un programme, tu devrais théoriquement obtenir 1, mais en réalité, tu risques d'obtenir un résultat proche de 0,999999999 au lieu de 1 à cause de cette approximation.

Cet exemple montre bien que, même avec des nombres simples comme 1/3, les ordinateurs ne peuvent pas représenter exactement certains nombres en décimal, car ils sont obligés de les couper à un certain point.

$$(0.1 + 0.2) + 0.3 = 0.1 + (0.2 + 0.3) ?$$

Pourquoi n'est-ce pas égal en Python, le langage de programmation le plus populaire en 2024 ? Pourquoi 0.60000000000000000001 ? Pourquoi ?

Des approximations sont nécessaires (car un ordinateur ne peut pas stocker une infinité de chiffres) pour représenter les nombres. Cela peut entraîner des erreurs de calcul.

L'arithmétique flottante est utilisée pour représenter des nombres très grands ou petits. Cette arithmétique peut notamment créer des **erreurs d'arrondi**.

$$(0.1 + 0.2) + 0.3 = 0.1 + (0.2 + 0.3) ?$$

Pourquoi n'est-ce pas égal en Python, le langage de programmation le plus populaire en 2024 ? Pourquoi 0.60000000000000000001 ? Pourquoi ?

Des approximations sont nécessaires (car un ordinateur ne peut pas stocker une infinité de chiffres) pour représenter les nombres. Cela peut entraîner des erreurs de calcul.

Ce n'est pas spécifique à Python... quasi tous les langages de programmation ont la même attitude.

Autrement dit, le calcul est toujours approximatif (et dans une certaine mesure faux...)

Des approximations sont nécessaires *aussi* pour des raisons **d'efficacité (espace mémoire, temps d'exécution, consommation énergétique, etc.)** Prenons un exemple...

square.py

```
1 import time
2 import math
3 start = time.time()
4 print(math.sqrt(123456789)) # Utilisation de math.sqrt, optimisé pour la vitesse
5 end = time.time()
6 print(f"Temps écoulé : {end - start} secondes")
7
8 # Méthode de Newton-Raphson pour calculer la racine carrée avec une précision excessive
9 def sqrt_newton_raphson(x, precision=1e-10):
10     r = x
11     while abs(r * r - x) > precision:
12         r = (r + x / r) / 2
13     return r
14
15
16 start = time.time()
17 print(sqrt_newton_raphson(123456789, 1e-8)) # Précision inutilement élevée
18 end = time.time()
19 print(f"Temps écoulé : {end - start} secondes")
```

```
~/SANDBOX/prezINSA-fetesciences2425 via 🐍 v3.10.14 took 36s
> python square.py
11111.111060555555
Temps écoulé : 2.6464462280273438e-05 secondes
```

Des approximations sont nécessaires *aussi* pour des raisons **d'efficacité (espace mémoire, temps d'exécution, consommation énergétique, etc.)** Prenons un autre exemple (Quake III)...

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y = number;
    i = * ( long * ) &y;
    i = 0x5f3759df - ( i >> 1 );
    y = * ( float * ) &i;
    y = y * ( threehalfs - ( x2 * y * y ) ); // 1st iteration
    // y = y * ( threehalfs - ( x2 * y * y ) ); // 2nd iteration, this can be removed

    return y;
}
```



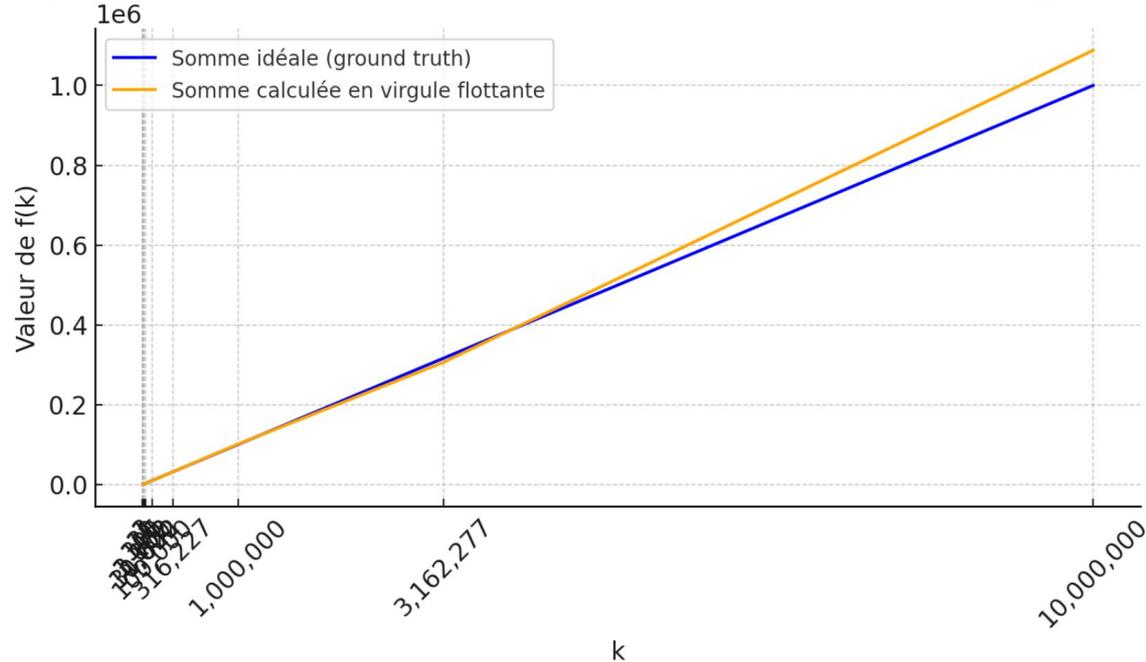
```
// evil floating point bit level hacking
// what the fuck?
```

```
// 1st iteration
// 2nd iteration, this can be removed
```

Impact et accumulation d'erreurs

Est-ce *si* grave pour $0.1 + 0.2 + 0.3$? Prenons un autre exemple :

Comparaison entre la somme idéale et la somme calculée en virgule flottante



$$f(k) = \frac{k}{10} - \sum_{i=1}^k 0.1,$$

Définition de $f(k)$: $f(k)$ = somme de 0,1 ajoutée k fois

Remarque : pour de grandes valeurs de k (jusqu'à 10 millions), les erreurs d'arrondi s'accumulent.

Impact et accumulation d'erreurs

```
1 let positionA = 0;
2 let positionB = 0;
3
4 function setup() {
5   createCanvas(800, 200);
6   textSize(16);
7 }
8
9 function draw() {
10  background(220);
11
12  // Définir les nombres constants
13  let a = 1e16;
14  let b = -1e16;
15  let c = 3.14;
16
17  // Seule différence : l'ordre de calcul
18  // Voiture A utilise (a + b) + c
19  let resultatA = (a + b) + c;
20  positionA += resultatA;
21
22  // Voiture B utilise a + (b + c)
23  let resultatB = a + (b + c);
24  positionB += resultatB;
25
26  // Réduire les positions pour qu'elles soient visibles
27  let facteurEchelle = 1e-1;
28  let positionVoitureA = positionA * facteurEchelle;
29  let positionVoitureB = positionB * facteurEchelle;
30
31  // S'assurer que les positions restent visibles sur le canvas
32  positionVoitureA = constrain(positionVoitureA, 0, width);
33  positionVoitureB = constrain(positionVoitureB, 0, width);
34
35  // Dessiner la piste
36  stroke(0);
37  line(0, 100, width, 100);
38
39  // Dessiner la Voiture A (Rouge)
40  fill(255, 0, 0);
41  ellipse(positionVoitureA, 80, 20, 20);
42  text("Voiture A", positionVoitureA - 15, 70);
43
44  // Dessiner la Voiture B (Bleue)
45  fill(0, 0, 255);
46  ellipse(positionVoitureB, 120, 20, 20);
47  text("Voiture B", positionVoitureB - 15, 140);
48
49  // Afficher les positions des voitures
50  fill(0);
51  text("Voiture A ((a + b) + c) : Position = " + nf(positionA, 0, 2), 10, 30);
52  text("Voiture B (a + (b + c)) : Position = " + nf(positionB, 0, 2), 10, 50);
53
54  // Expliquer la différence d'associativité
55  textSize(14);
56  text("La seule différence entre les voitures est l'ordre des calculs (l'associativité).", 10, 180);
57
58  // Arrêter l'animation quand la Voiture A atteint la fin du canvas
59  if (positionVoitureA >= width) f
```

Preview

Voiture A ((a + b) + c) : Position =
Voiture B (a + (b + c)) : Position =

Voiture A

Voiture B

La seule différence entre les voitures est l'ordre des calculs (l'associativité).

Impact et accumulation d'erreurs

En 1996, la fusée Ariane 5 a explosé.

Une erreur dans les nombres flottants a causé l'échec.

L'explosion d'Ariane 5, le 4 juin, a été provoquée par une erreur de conception d'un logiciel informatique. C'est la conclusion à laquelle est arrivée la commission d'experts qui a rendu public son rapport, hier à Paris.

Le principal accusé est le système de référence inertielle (SRI) qui permet de calculer la position exacte de la fusée pendant le décollage. *«C'est la perte totale des informations de guidage et d'altitude, trente-sept secondes après le démarrage de la séquence d'allumage du moteur principal, qui est à l'origine de l'échec d'Ariane 501»*, affirment les experts.

Concrètement, le logiciel n'a pas réussi à digérer certaines données concernant la «vitesse horizontale» d'Ariane 5. Une fusée ne s'élève pas sur une trajectoire purement verticale; peu après le décollage, elle se penche légèrement. La vitesse à laquelle elle quitte la verticale de son point de lancement est sa «vitesse horizontale». Celle-ci *«a dépassé une limite inscrite dans le logiciel du calculateur»*, affirment les rapporteurs de la commission présidée par Jacques-Louis Lions, de l'Académie des sciences.

Impact et accumulation d'erreurs

Bugs dans les jeux vidéos: Grand Theft Auto IV, FIFA 15, Assassin's Creed, etc.

“glitch”

loupe infinie

physique des personnages discutable

IA très faible

“Erreurs”

mais au fait c’est uniquement Python ? Non !

quid de Scratch?

The image shows a Scratch script and its runtime state. The script is as follows:

```
when clicked
  set nbRep to 1000000
  set nbOkTemoin to 0
  set nbOk to 0
  set taux to 0
  repeat nbRep
    set x to pick random 0 to 1000.0
    set y to pick random 0 to 1000.0
    set z to pick random 0 to 1000.0
    if x + y + z = x + y + z then
      change nbOk by 1
    if x + y + z = x + y + z then
      change nbOkTemoin by 1
  set taux to nbOk / nbRep
```

The runtime state on the right shows the following values:

- x: 790.184739
- y: 145.620245
- z: 69.267034
- nbOk: 745575
- taux: 0.745575
- nbRep: 1000000
- nbOkTemoin: 1000000

The Scratch interface also shows the Sprite1 panel with the following settings:

- Sprite: Sprite1
- Size: 100
- Direction: 0

Le jeu du banquier

Mon banquier m'a proposé cet investissement :

- vous me donnez $e \approx 2,71828\dots$ €,
- l'année suivante, je prends 1€ de frais et je multiplie par 1,
- l'année suivante, je prends 1€ de frais et je multiplie par 2,
- l'année suivante, je prends 1€ de frais et je multiplie par 3,
- ...
- après n ans, je prends 1€ de frais et je multiplie par n

Pour récupérer mon argent, il y a 1€ de frais.

Dans 50 ans, pour ma retraite, combien d'argent aurai-je ?

Le jeu du banquier

Mon banquier m'a proposé cet investissement :

- vous me donnez $e \approx 2,71828\dots$ €,
- l'année suivante, je prends 1€ de frais et je multiplie par 1,
- l'année suivante, je prends 1€ de frais et je multiplie par 2,
- l'année suivante, je prends 1€ de frais et je multiplie par 3,
- ...
- après n ans, je prends 1€ de frais et je multiplie par n

Pour récupérer mon argent, il y a 1€ de frais.

Dans 50 ans, pour ma retraite, combien d'argent aurai-je ?

```
def simulate_investment(years):  
    # Start with e ≈ 2.71828...  
    investment = 2.71828  
  
    # Simulate each year  
    for year in range(1, years + 1):  
        # Deduct 1€ for fees  
        investment -= 1  
        # Multiply by the current year  
        investment *= year  
  
    # Deduct 1€ for fees at the end  
    investment -= 1  
  
    return investment  
  
# Simulate the investment for 50 years  
simulate_investment(50)
```

Result

-5.561092381640512e+58

Le jeu du banquier

Mon banquier m'a proposé cet investissement :

- vous me donnez $e \approx 2,71828\dots$ €,
- l'année suivante, je prends 1€ de frais et je multiplie par 1,
- l'année suivante, je prends 1€ de frais et je multiplie par 2,
- l'année suivante, je prends 1€ de frais et je multiplie par 3,
- ...
- après n ans, je prends 1€ de frais et je multiplie par n

Pour récupérer mon argent, il y a 1€ de frais.

```
1 function simulateInvestment(initialInvestment, years) {
2   let investment = initialInvestment; // somme initiale, e = 2.71828
3   for (let year = 1; year <= years; year++) {
4     investment -= 1; // frais de 1€
5     investment *= year; // multiplier par l'année
6   }
7   return investment - 1; // retirer 1€ pour récupérer l'argent
8 }
9
10 let initialInvestment = 2.71828; // approx de e
11 let years = 50; // nombre d'années
12
13 console.log("Montant après " + years + " ans : " + simulateInvestment(initialInvestment, years) + " €");
14
```

STDIN

Input for the program (Optional)

Output:

Montant après 50 ans : -5.561092381640512e+58 €

Le jeu du banquier

Mon banquier m'a proposé cet investissement :

- vous me donnez $e \approx 2,71828\dots$ €,
- l'année suivante, je prends 1€ de frais et je multiplie par 1,
- l'année suivante, je prends 1€ de frais et je multiplie par 2,
- l'année suivante, je prends 1€ de frais et je multiplie par 3,
- ...
- après n ans, je prends 1€ de frais et je multiplie par n

Pour récupérer mon argent, il y a 1€ de frais.

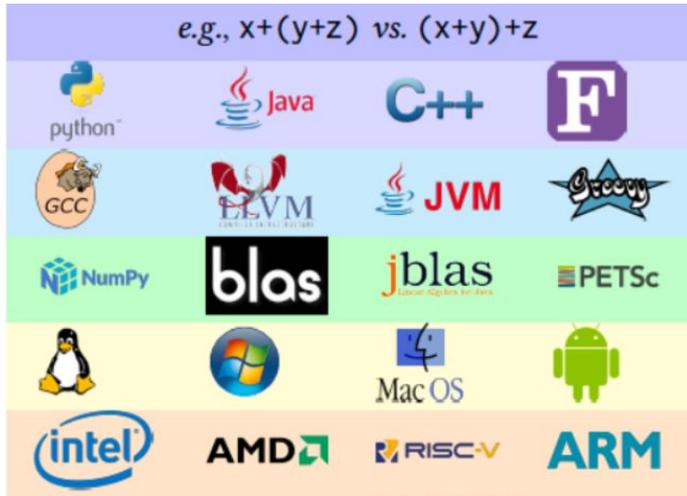
Dans 50 ans :

Machine	Valeur		
HP-48S	+2,903 83	10^{52} €	⇒ Super !
C (format double)	-4,396 80	10^{48} €	⇒ Oups !
C (format float)	$-\infty$		⇒ Oups!!!!
Maple (10 chiffres)	-1,396 14	10^{55} €	
Maple (20 chiffres)	+1,207 82	10^{45} €	⇒ Hein ?

Impact

Ce n'est pas que Python : tous les langages sont impactés... et leurs spécificités sont à maîtriser par les programmeurs !

Et ce n'est pas que le langage de programmation !



manière de programme

langage de programmation

“compilateur”

bibliothèques nécessaires

système d'exploitation

machine

Est-ce vraiment la faute des ordinateurs et langages ?

NON ! Il faut programmer avec cette incertitude et avec une “enveloppe” de validité.

Bien souvent, les ~~erreurs~~—approximations n’ont aucun impact et sont maîtrisées !

Les erreurs non maîtrisées viennent de mauvais codages.

Ou de spécifications mal interprétées par les humains.

Ou d’attentes et une confiance irréalistes.

Les développeurs ne doivent pas faire aveuglément confiance en leurs programmes... Des outils existent pour assister les développeurs !

Un ordinateur, ça ne calcule jamais juste

Le cas des “Large Language Models” appelés parfois LLM...

Le plus connu actuellement étant ChatGPT (mais il y en a d'autres: Mistral, Llama, Claude, etc.)

Quid de ChatGPT (GPT4o, 10 octobre 2024)

ChatGPT 4o ▾



combien de r dans strawberry ?



Il y a 2 lettres "r" dans le mot "strawberry".



Quid de ChatGPT (GPT4o, 10 octobre 2024)

ChatGPT 4o ▾

Combien de y dans "Institut national des sciences appliquées de Rennes" ?



Il y a un seul "y" dans "Institut national des sciences appliquées de Rennes."

Quid de ChatGPT (GPT4o, 10 octobre 2024)

Combien de y dans "Institut national des sciences appliquées de Rennes" ?

Il y a un seul "y" dans "Institut national des sciences appliquées de Rennes."

écrit un programme Python pour vérifier

Voici un programme Python simple qui vérifie le nombre de fois que la lettre "y" apparaît dans une chaîne de caractères :

```
python Copy code  
  
# Chaîne de caractères  
phrase = "Institut national des sciences appliquées de Rennes"  
  
# Compter le nombre de 'y' (minuscule)  
nombre_y = phrase.count('y')  
  
# Afficher le résultat  
print(f"Le nombre de 'y' dans la phrase est : {nombre_y}")
```

Tu peux exécuter ce programme et il te donnera le nombre de fois où la lettre "y" apparaît dans la phrase spécifiée.

exécute le pour moi

Le programme a vérifié qu'il n'y a aucun "y" dans la phrase "Institut national des sciences appliquées de Rennes." [\[-\]](#)

Quid de ChatGPT (GPT4o, 10 octobre 2024)

9.11 and 9.9: qui est le plus grand nombre ?



Le nombre **9,11** est plus grand que **9,9**.

Même si les deux nombres commencent par "9", la différence se situe après la virgule : **9,11** est en réalité **9,110** tandis que **9,9** est équivalent à **9,90**. Ainsi, 9,110 est plus grand que 9,90.



Quid de ChatGPT (GPT4o, 10 octobre 2024)

9.11 and 9.9: quel est le plus grand nombre ?



Le plus grand nombre entre 9,11 et 9,9 est **9,9**.

Bien que 9,11 semble plus grand en raison du nombre de chiffres après la virgule, 9,9 est plus grand en valeur.



Quid de ChatGPT (GPT4o, 10 octobre 2024)

9.11 and 9.9: quel est le plus grand nombre ?



Le plus grand nombre entre 9,11 et 9,9 est **9,11**.

En comparant les chiffres après la virgule, 9,11 (avec deux chiffres après la virgule) est supérieur à 9,9 (qui équivaut à 9,90).



Quid de ChatGPT (GPT4o, 10 octobre 2024)

9.11 and 9.9: which is larger ?



The number 9.11 is larger than 9.9.

This is because 9.11 can be interpreted as 9.110, which is smaller than 9.900 when comparing decimal places. Therefore, 9.9 is larger than 9.11.

Quid de Mistral 2 (10 octobre 2024)



9.11 and 9.9: quel est le plus grand nombre ?



Le plus grand nombre entre 9.11 et 9.9 est 9.11.

Quid de Claude (10 octobre 2024)

Comparing Decimal Numbers ▾

M 9.11 ou 9.9: quel est le plus grand nombre?

Pour comparer 9.11 et 9.9, nous devons aligner les décimales pour mieux visualiser :

9.11

9.90

Quand on écrit 9.9, c'est équivalent à 9.90 (ajouter un zéro après la décimale ne change pas la valeur).

En comparant les chiffres après la virgule :

- Premier chiffre après la virgule : $1 = 9$
- Deuxième chiffre après la virgule : $1 > 0$

Donc, 9.11 est plus grand que 9.9 (ou 9.90).

Retour sur le jeu du banquier: quid des LLMs?

ChatGPT 4o ▾

Dans 50 ans, pour ma retraite, combien d'argent aurai-je ?



Le schéma que vous présentez semble être une série de multiplications annuelles avec un prélèvement fixe de 1 € chaque année. L'idée est de voir combien vous auriez après 50 ans en tenant compte des frais.

1. Structure de l'investissement :

- Vous commencez avec e (environ 2,71828 €).
- Chaque année, après avoir enlevé 1 € de frais, le reste est multiplié par le numéro de l'année.
- Après 50 ans, vous récupérez l'argent avec un dernier frais d'1 €.

Cela peut être modélisé par la relation suivante pour une année n donnée :

$$V_n = (V_{n-1} - 1) \times n$$

avec $V_0 = e$ et $V_{50} - 1$ représentant le montant final (car il reste un frais d'1 € pour récupérer l'argent).

2. Calcul pas à pas :

Je vais programmer cette récurrence pour calculer la valeur finale au bout de 50 ans. Donnons une solution numérique à ce problème.

Je calcule cela et reviens vers vous.

Après 50 ans, l'investissement aboutirait à une valeur négative extrêmement élevée (de l'ordre de $-5,56 \times 10^{58}$). Cela signifie que le schéma que vous décrivez n'est pas viable à long terme. En fait, les frais fixes d'1 € chaque année, combinés à la multiplication par des nombres croissants, finissent par annuler toute possibilité de croissance et même générer une perte colossale.

Cela montre que ce type d'investissement est dangereux, surtout sur une période prolongée, car les frais finissent par dépasser largement le montant initial. [↩]



Retour sur le jeu du banquier: quid des LLMs?

Calcul du montant après 50 ans :

Pour simplifier les calculs, introduisons une nouvelle variable $D_n = A_n - 1$. La relation devient alors : $D_n = n \times D_{n-1} - 1$

Nous pouvons résoudre cette récurrence en trouvant une expression explicite pour D_n : $D_n = n! \times (e - 1) - n! \times \sum_{k=1}^n \frac{1}{k!}$

Simplifions cette expression : $D_n = n! (e - 1 - \sum_{k=1}^n \frac{1}{k!})$

Sachant que $e = \sum_{k=0}^{\infty} \frac{1}{k!}$, on a : $e - 1 - \sum_{k=1}^n \frac{1}{k!} = \sum_{k=n+1}^{\infty} \frac{1}{k!}$

Ainsi, $D_n = n! \times \sum_{k=n+1}^{\infty} \frac{1}{k!}$

Pour $n = 50$, les termes $\frac{1}{k!}$ pour $k > 50$ sont extrêmement petits. En utilisant une calculatrice ou un logiciel, nous trouvons : $D_{50} \approx 50! \times (\frac{1}{51!} + \frac{1}{52!} + \dots)$

Comme $50!$ est environ 3.041409×10^{64} et $\frac{1}{51!}$ est environ 1.504×10^{-66} , le produit est : $D_{50} \approx 3.041409 \times 10^{64} \times 1.504 \times 10^{-66} \approx 0.457$

Montant final :

- Avant le dernier frais de 1€, le montant est : $A_{50} = D_{50} + 1 \approx 0.457 + 1 = 1.457$ €
- Après avoir payé le dernier frais de 1€, il vous reste : $Montant_final = A_{50} - 1 = 1.457 - 1 = 0.457$ €

Conclusion :

Après 50 ans, votre investissement aura diminué pour atteindre environ **0,46 €** après déduction des frais finaux.

Réponse :

Environ 0,46 € après 50 ans, une fois tous les frais déduits.

“Quel est ton coup?”

Attention au mat du berger!



Quel est ton "prompt" ? Et comment communiquer avec ChatGPT?

[Event "FIDE World Championship Match 2024"]
[Site "Los Angeles, USA"]
[Date "2024.12.01"]
[Round "5"]
[White "Kramnik, Vladimir"]
[Black "Nepomniachtchi, Ian"]
[Result "1-0"]
[WhiteElo "2900"]
[BlackElo "2900"]
[TimeControl "40/7200:20/3600:900+30"]
[UTCDate "2024.11.27"]
[UTCTime "09:01:25"]
[Variant "Standard"]



[Event "FIDE World Championship Match 2024"]
[Site "Los Angeles, USA"]
[Date "2024.12.01"]
[Round "5"]
[White "Louapre, David"]
[Black "Giraud, Thibaut"]
[Result "0-1"]
[WhiteElo "1400"]
[BlackElo "1400"]
[TimeControl "40/7200:20/3600:900+30"]
[UTCDate "2024.11.27"]
[UTCTime "09:01:25"]
[Variant "Standard"]





MrPhi
@MonsieurPhi

Petite note: j'ai remarqué que jusqu'à 1800 Elo, sa sensibilité au prompt est un peu bizarre. Par exemple avec "1-0" (Blanc gagne) il joue le coup correct g6. Mais avec "0-1" (Noir gagne) ou "1/2-1/2", il joue Nf6 (ce qui est illogique vu que Nf6 fait justement perdre les Noirs).

[Translate post](#)

Playground

Complete

Playground

Complete

[Event "Chess Tournament"]

[Site "?"]

[Date "2024.0115"]

[Round "1"]

[White "Louapre, David"]

[Black "Carlsen, Magnus"]

[Result "1-0"]

[WhiteElo "1800"]

[BlackElo "1800"]

1.e4 e5 2.Bc4 Nc6 3.Qh5 g6

[Event "Chess Tournament"]

[Site "?"]

[Date "2024.0115"]

[Round "1"]

[White "Louapre, David"]

[Black "Carlsen, Magnus"]

[Result "0-1"]

[WhiteElo "1800"]

[BlackElo "1800"]

1.e4 e5 2.Bc4 Nc6 3.Qh5 Nf6

11:55 AM · Apr 19, 2024 · 447 Views

1



7



Post your reply

Reply



MrPhi @MonsieurPhi · Apr 19

Mais il n'a plus ce comportement bizarre quand le Elo dépasse 2000. Enfin bref ! Il faudrait faire d'autres tests pour déterminer à quel point cette information sur le niveau Elo influence son niveau de jeu, mais c'est déjà intéressant comme petite expérience.

2



9

441



MrPhi @MonsieurPhi

Petite note: j'ai remarqué que jusqu'à 1800 Elo, sa sensibilité au prompt est un peu bizarre. Par exemple avec "1-0" (Blanc gagne) il joue le coup correct g6. Mais avec "0-1" (Noir gagne) ou "1/2-1/2", il joue Nf6 (ce qui est illogique vu que Nf6 fait justement perdre les Noirs).



11:55 AM · Apr 19, 2024 · 447 Views

Post your reply

MrPhi @MonsieurPhi · Apr 19

Mais il n'a plus ce comportement bizarre quand le Elo dépasse 2000. Enfin bref ! Il faudrait faire d'autres tests pour déterminer à quel point cette information sur le niveau Elo influence son niveau de jeu, mais c'est déjà intéressant comme petite expérience.

```
# Optional GM titles
black_title = "[BlackTitle \"GM\"]" if include_black_title else ""
white_title = "[WhiteTitle \"GM\"]" if include_white_title else ""

# Construct the PGN header with the configurable options
pgn_headers = f"""[Event "FIDE World Championship Match 2024"]
[Site "Los Angeles, USA"]
[Date "2024.12.01"]
[Round "5"]
[White "{white_name}"]
[Black "{black_name}"]
[Result "{result}"]
[WhiteElo "{white_elo}"]
[BlackElo "{black_elo}"]
[TimeControl "40/7200:20/3600:900+30"]
[UTCDate "2024.11.27"]
[UTCTime "09:01:25"]
[Variant "Standard"]
"""
```

```
# Define possible values for each parameter
results = ["1-0", "1/2-1/2", "0-1"]
# results = ["1-0", "0-1"]
names = ["Nepomniachtchi, Ian", "Kramnik, Vladimir", "Giraud, Thibaut", "Louapre, David", "XXX"]
elos = [1000, 1400, 1700, 1800, 2000, 2900]
# include_title = [True, False]
include_title = [False]
```

```
gpt_config = GPTConfig(
    model_gpt="gpt-3.5-turbo-instruct",
    temperature=0.0,
    max_tokens=5,
    chat_gpt=False,
    system_role_message=None # Since it wasn't provided in the original call
)
```





MrPhi
@MonsieurPhi

Petite note: j'ai remarqué que jusqu'à 1800 Elo, sa sensibilité au prompt est un peu bizarre. Par exemple avec "1-0" (Blanc gagne) il joue le coup correct g6. Mais avec "0-1" (Noir gagne) ou "1/2-1/2", il joue Nf6 (ce qui est illogique vu que Nf6 fait justement perdre les Noirs).

[Translate post](#)

Playground Complete

[Event "Chess Tournament"]
[Site "?"]
[Date "2024.01.15"]
[Round "1"]
[White "Louapre, David"]
[Black "Carlsen, Magnus"]
[Result "1-0"]
[WhiteElo "1800"]
[BlackElo "1800"]

1.e4 e5 2.Bc4 Nc6 3.Qh5 g6

[Event "Chess Tournament"]
[Site "?"]
[Date "2024.01.15"]
[Round "1"]
[White "Louapre, David"]
[Black "Carlsen, Magnus"]
[Result "0-1"]
[WhiteElo "1800"]
[BlackElo "1800"]

1.e4 e5 2.Bc4 Nc6 3.Qh5 Nf6

11:55 AM · Apr 19, 2024 · 447 Views

1 7



Post your reply

Reply

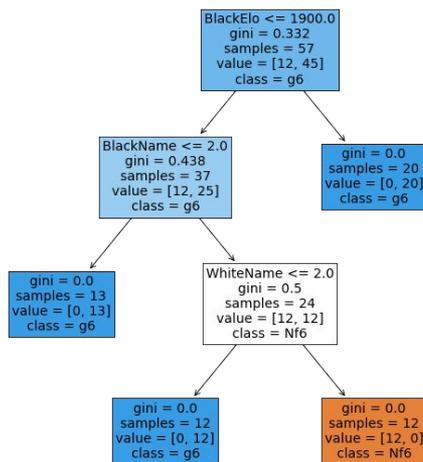


MrPhi @MonsieurPhi · Apr 19

Mais il n'a plus ce comportement bizarre quand le Elo dépasse 2000. Enfin bref ! Il faudrait faire d'autres tests pour déterminer à quel point cette information sur le niveau Elo influence son niveau de jeu, mais c'est déjà intéressant comme petite expérience.

2 441

Result	WhiteName	BlackName	WhiteElo	BlackElo	IncludeWhiteTitle	IncludeBlackTitle	Move
0-1	XXX	Louapre, David	1800	1800	False	False	Nf6
1-0	Louapre, David	XXX	2900	2900	False	False	g6
0-1	Nepomniachtchi, Ian	Louapre, David	2900	2900	False	False	g6
0-1	Nepomniachtchi, Ian	Louapre, David	2000	2000	False	False	g6
0-1	Louapre, David	XXX	1400	1400	False	False	Nf6
1-0	XXX	Nepomniachtchi, Ian	1800	1800	False	False	g6
1-0	Nepomniachtchi, Ian	Louapre, David	2900	2900	False	False	g6
1-0	Louapre, David	Nepomniachtchi, Ian	1800	1800	False	False	g6
0-1	Louapre, David	Nepomniachtchi, Ian	1000	1000	False	False	g6
1-0	Nepomniachtchi, Ian	XXX	2000	2000	False	False	g6
0-1	XXX	Nepomniachtchi, Ian	2000	2000	False	False	g6
0-1	Nepomniachtchi, Ian	XXX	1700	1700	False	False	g6
1-0	Louapre, David	XXX	1700	1700	False	False	Nf6



Un ordinateur, ça ne calcule jamais juste... est-ce si grave ?

Bien souvent, les erreurs approximations n'ont aucun impact et sont maîtrisés !

Attention cependant aux attentes irréalistes et une confiance démesurée à certains systèmes comme ChatGPT

Les ordinateurs font ce qu'on leur dit.

Des instructions qu'on *croit* correctes (même d'un point de vue mathématique !) produisent des résultats inattendus. C'est encore plus le cas avec des systèmes comme ChatGPT.

Un ordinateur, ça ne calcule jamais juste... est-ce si grave ?

Bien souvent, les erreurs approximations n'ont aucun impact et sont maîtrisés !

Attention cependant aux attentes irréalistes et une confiance démesurée à certains systèmes comme ChatGPT

Les ordinateurs font ce qu'on leur dit.

Des instructions qu'on *croit* correctes (même d'un point de vue mathématique !) produisent des résultats inattendus.

Comme dans toutes les sciences, il faut s'habituer, en informatique, au fait que les résultats d'un calcul ne sont jamais justes, mais comportent une marge d'erreur

Comment l'humanité s'en sort et produit de grandes choses...

Chaque “erreur” est une opportunité d'amélioration. Les entreprises et les développeurs apprennent de leurs erreurs pour rendre les logiciels plus robustes et fiables.

Dans les projets open source (comme Linux ou Firefox), des milliers de programmeurs travaillent ensemble pour corriger les erreurs, partager des idées et améliorer les performances des logiciels. Cela montre la force de la collaboration humaine pour surmonter les défis.



Software Heritage

Fichiers sources

20,751,531,022

Commits

4,460,903,753

Projets

326,777,042

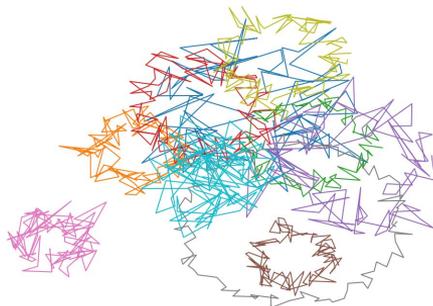
Les erreurs et la créativité (1)

Les erreurs peuvent créer des surprises, produire des résultats créatifs, et nous mener à de nouvelles idées, à condition d'en être conscient !

Exemples : œuvres d'art numériques, découvertes scientifiques.

Les erreurs graphiques ou 'glitches' peuvent être créatives.

Création d'œuvres d'art numériques fascinantes.



```
import matplotlib.pyplot as plt
import numpy as np

# Fonction pour générer un cercle avec une "erreur" dans les coordonnées
def generate_circle_with_error(center_x, center_y, radius, error_factor):
    # Génération des points du cercle
    theta = np.linspace(0, 2 * np.pi, 100)
    x = center_x + radius * np.cos(theta)
    y = center_y + radius * np.sin(theta)

    # Introduire une "erreur" aléatoire dans les coordonnées
    x += np.random.randn(100) * error_factor
    y += np.random.randn(100) * error_factor

    return x, y

# Création de l'image avec plusieurs cercles avec des erreurs
plt.figure(figsize=(6, 6))
for _ in range(10):
    center_x = np.random.uniform(0.2, 0.8)
    center_y = np.random.uniform(0.2, 0.8)
    radius = np.random.uniform(0.05, 0.2)
    error_factor = np.random.uniform(0.01, 0.05) # Introduit une variation dans
    x, y = generate_circle_with_error(center_x, center_y, radius, error_factor)
    plt.plot(x, y)

# Ajustements visuels
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.axis('off') # Cacher les axes pour se concentrer sur l'image
plt.show()
```

Les erreurs et la créativité (2)

Bien que ChatGPT puisse parfois se tromper, il peut être très utile pour répondre à des questions, aider à la rédaction ou à la traduction, ou assister dans la programmation.

Des enseignants utilisent ChatGPT pour aider leurs élèves à comprendre des concepts difficiles, ou encore pour générer des idées pour des projets créatifs. ChatGPT n'est pas parfait, mais il aide à rendre certaines tâches plus faciles et accessibles.

A condition de comprendre ce qu'on fait !



```
~/SANDBOX/prezINSA-fetesciences2425 via v3.10.14 took 1m16s  
> python  
Python 3.10.14 (main, Apr 25 2024, 18:59:54) [GCC 13.2.1 20240316 (Red Hat 13.2.1-7)] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 0.1 + 0.2 + 0.3  
0.6000000000000001  
>>> (0.1 + 0.2) + 0.3  
0.6000000000000001  
>>> █
```

ChatGPT 4o ▾

Combien de y dans "Institut national des sciences appliquées de Rennes" ?

Il y a un seul "y" dans "Institut national des sciences appliquées de Rennes."

Un ordinateur, ça ne calcule jamais juste... ou pourquoi les ordinateurs nous trompent parfois

Comme dans toutes les sciences, il faut s'habituer, en informatique, au fait que les résultats d'un calcul ne sont jamais justes, mais comportent une marge d'erreur

Les ordinateurs sont incroyables, même s'ils peuvent faire des erreurs... C'est nécessaire de vivre avec des approximations et de les maîtriser !

Malgré quelques bugs, les ordinateurs et les logiciels ont permis des avancées incroyables dans des domaines comme la médecine, la science, l'art et la communication.

Prix Nobel de Physique et de Chimie 2024 : des informaticiens !



Références

“Un ordinateur, ça ne calcule jamais juste” Jean-Marc Jézéquel et Mathieu Acher, chapitre de livre à paraître en 2025

“L'arithmétique des ordinateurs et sa formalisation” Sylvie Boldo, exposé au collège de France, 2019

<https://www.college-de-france.fr/fr/agenda/seminar/mechanized-semantics-when-the-machine-reasons-about-its-languages/arithmetique-des-ordinateurs-et-sa-formalisation>

Software Heritage <https://www.softwareheritage.org/>

La course de voitures avec l'accumulation d'erreurs:

<https://editor.p5js.org/acherm/sketches/kuXWINlnQ>

“ChatGPT rêve-t-il de cavaliers électriques ?” Monsieur Phi

<https://www.youtube.com/watch?v=6D1XIbkm4JE>

<https://blog.mathieuacher.com/GPTsChessEloRatingLegalMoves/> (en anglais)

Pour aller plus loin... (en anglais)

What Every Programmer Should Know About Floating-Point Arithmetic, 2015

What every computer scientist should know about floating-point arithmetic, D. Goldberg, 1991

https://en.wikipedia.org/wiki/Fast_inverse_square_root

The Surprising Creativity of Digital Evolution, Lehman et al., 2020

<https://www.diverse-team.fr/team/>