Codage de l'information

Laurent Noé-Léopold Weinberg
paternité : Francesco De Comité

Licence 1 Mathématiques Informatique Semestre 1
Université de Lille
Faculté des Sciences et Technologies

10 septembre 2025

Représentation des réels

Entiers, réels : des utilisations différentes

Entiers

On utilise les entiers pour :

- Identifier des objets.
- Dénombrer des objets discrets.

Les réels

Les réels apparaissent :

- Comme résultats de mesures.
- Calcul de valeurs continues : vitesses, masses, ...
- Résultats de division.
- Valeurs approchées de quantités très grandes ou très petites (quand on n'a pas besoin de connaître tous les chiffres)

Entiers, réels : des représentations différentes

Informations à préserver

Pour les entiers

- Tous les chiffres du nombre.
- Ce qui limite la plage des nombre représentables.

Pour les réels

- L'ordre de grandeur (magnitude du nombre).
- Un nombre de chiffres suffisant pour limiter les erreurs d'arrondi.

- On accepte de perdre des informations (des chiffres)
- Ce n'est pas particulier à l'informatique : combien de chiffres de π utilise-t-on dans des calculs ?

La contrainte de l'espace limité

- Comme pour les entiers, on cherche à représenter les nombres réels dans un espace fini : la mémoire de l'ordinateur.
- On doit définir un format commun à toutes les plate-formes et tous les programmes qui auront à manipuler ces réels.
- Les informaticiens ont défini une norme de représentation des réels : la norme IEEE 754.
- Avant de décrire cette norme, nous allons étudier l'écriture des nombres réels en binaire.

Nombres réels en binaire

Réel en base 10

Un nombre décimal en base 10 est une suite de chiffres, avec éventuellement une virgule :

$$a_n a_{n-1} \cdots a_2 a_1 a_0, a_{-1} a_{-2} \cdots a_{-m}$$

Sa valeur est :

$$\sum_{i=0}^{n} a_i \times 10^i + \sum_{i=1}^{m} a_i \times 10^{-i}$$

Exemple :

$$27,235 = 2 \times 10^1 + 7 \times 10^0 + 2 \times 10^{-1} + 3 \times 10^{-2} + 5 \times 10^{-3}$$

Nombres réels en binaire

Traduction décimal binaire

- La partie fractionnaire d'un nombre est inférieure à 1.
- Sa partie entière est un entier.
- On peut utiliser les techniques vues en cours pour traduire la partie entière en binaire.
- Sur l'exemple précédent $(27{,}235)$, la partie entière vaut 27, qui s'écrit en binaire $\overline{11011}_2$
- Il reste à traduire 0,235 en binaire et à la concaténer à $\overline{11011}_2$

Traduire une partie fractionnaire en binaire

Une idée de méthode

- le nombre fractionnaire $N = \overline{0, \mathtt{a}_{-1}\mathtt{a}_{-2}\cdots\mathtt{a}_{-\mathtt{m}}}_{\ 2}$ aura un '1' immédiatement à droite de la virgule s'il est supérieur à $\frac{1}{2}$...
- ...donc si $2 \times N > 1$
- D'où le début de méthode :
  - Multiplier n par 2.
  - Si le résultat est supérieur à 1, le premier chiffre à droite de la virgule en base 2 est un '1'.
  - Sinon, le chiffre est un zéro.
  - Ne garder que la partie fractionnaire, et recommencer pour le chiffre a_{-2}

Traduire une partie fractionnaire en binaire : exemple

Soit à traduire 0,235 en binaire.

$0,235 \times 2 = 0,47 < 1$ le chiffre en position -1 est 0

$0,47 \times 2 = 0,94 < 1$ le chiffre en position -2 est 0

$0,94 \times 2 = 1,88 > 1$ le chiffre en position -3 est 1

On soustrait 1 pour ne garder que la partie fractionnaire

$0,88 \times 2 = 1,76 > 1$ le chiffre en position -4 est 1

On soustrait 1 pour ne garder que la partie fractionnaire

$0,76 \times 2 = 1,52 > 1$ le chiffre en position -5 est 1

On soustrait 1 pour ne garder que la partie fractionnaire

$0,52 \times 2 = 1,04 > 1$ le chiffre en position -6 est 1

On soustrait 1 pour ne garder que la partie fractionnaire

$0,04 \times 2 = 0,08 > 1$ le chiffre en position -7 est 0

$0,08 \times 2 = 0,16 > 1$ le chiffre en position -8 est 0

Traduire une partie fractionnaire en binaire : exemple

Conclusion, vérification

- Est-ce que $\overline{0,235_{10}} \approx \overline{0,00111100_2}$?
- On doit retrouver la valeur représentée par $\overline{0,00111100_2}$
- $\overline{0,00111100_2} = 1 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5} + 1 \times 2^{-6}$
- Ça promet d'être laborieux ...
- ...mais on peut se simplifier la vie :
- $\overline{0,00111100_2} \times 2^6 = \overline{1111_2} = \overline{15_{10}}$
- et donc : $\overline{0,00111100_2} = 15/2^6 = 15/64 = 0.234375$

Traduire une partie fractionnaire en binaire : exemple

0	0.235		13	0.12	1
1	0.47	0	14	0.24	0
2	0.94	0	15	0.48	0
3	1.88	1	16	0.96	0
4	1.76	1	17	1.92	1
5	1.52	1	18	1.84	1
6	1.04	1	19	1.68	1
7	0.08	0	20	1.36	1
8	0.16	0	21	0.72	0
9	0.32	0	22	1.44	1
10	0.64	0	23	0.88	0
11	1.28	1	24	1.76	1
12	0.56	0	25	1.52	1

$$\overline{0.235}_{10} \approx \overline{0.001111000010100011110101 1\cdots}_2$$

Traduire une partie fractionnaire en binaire : exemple

Période $\longrightarrow$

0	0.235		13	0.12	1
1	0.47	0	14	0.24	0
2	0.94	0	15	0.48	0
3	1.88	1	16	0.96	0
4	1.76	1	17	1.92	1
5	1.52	1	18	1.84	1
6	1.04	1	19	1.68	1
7	0.08	0	20	1.36	1
8	0.16	0	21	0.72	0
9	0.32	0	22	1.44	1
10	0.64	0	23	0.88	0
11	1.28	1	24	1.76	1
12	0.56	0	25	1.52	1

$\longleftarrow$Période

$$\overline{0.235_{10}} \approx \overline{0.001111000010100011110101 1 \cdots_2}$$

Traduire une partie fractionnaire en binaire : exemple

Remarques

- Si on traduit un nombre fractionnaire ayant un nombre donné de chiffres, son développement en base 2 sera soit fini, soit périodique :
  - Par exemple, si on démarre avec 2 chiffres après la virgule, les termes successifs auront au plus deux chiffres après la virgule.
  - Le nombre de chiffres ne diminue que si le nombre initial se termine par 5.
  - Il y a donc au plus 100 termes différents.
  - Au plus tard au bout de 100 étapes, on retombera sur un terme déjà rencontré, et la suite se répétera.
- Les seuls nombres ayant un développement fini sont les sommes exactes de puissances de 2.

propriété

définition

On dit qu'une suite $(u_n)_{n\in\mathbb{N}}$ est ultimement périodique s'il existe un entier N et un entier p tel que pour tout entier n plus grand que N, on a $u_{n+p} = u_n$

théorème

si X est un ensemble fini, si f est une fonction de X dans$X$, alors toute suite récurente $(x_n)_{n\in\mathbb{N}}$ définie par $x_0 \in X$ et $x_{n+1} = f(x_n)$ est ultimement périodique.

remarque

- ce théorème est assez utilisé en info.
- il existe des algorithmes de detection de cycles.

Traduire une partie fractionnaire en binaire : le code

```
def fracversbinaire(n,p=10):
    """
     Traduit n, nombre fractionnaire inférieur à 1
     En calculant au plus p décimales
    """
    resu="0."
    i=0
    while (n!=0) and (i<p) :
        i=i+1
        n=2*n
        if n>=1:
            resu=resu+'1'
            n=n-1
        else:
            resu=resu+'0'
        print(i,n,resu)
    return resu
```

```
>>> fracversbinaire(0.235)
1 0.47 0.0
2 0.94 0.00
3 0.8799999999999999 0.001
4 0.7599999999999998 0.0011
5 0.5199999999999996 0.00111
6 0.03999999999999915 0.001111
7 0.07999999999999983 0.0011110
8 0.1599999999999966 0.00111100
9 0.3199999999999932 0.001111000
10 0.6399999999999864 0.0011110000
'0.0011110000'
```

```
>>> fracversbinaire(0.235,53)
1 0.47 0.0
2 0.94 0.00
3 0.8799999999999999 0.001
4 0.7599999999999998 0.0011
5 0.5199999999999996 0.00111
6 0.03999999999999915 0.001111
7 0.0799999999999983 0.0011110
8 0.1599999999999966 0.00111100
9 0.3199999999999932 0.001111000
10 0.6399999999999864 0.0011110000
11 0.2799999999999727 0.00111100001
12 0.5599999999999454 0.001111000010
13 0.11999999999989086 0.0011110000101
14 0.23999999999978172 0.00111100001010
15 0.47999999999956344 0.001111000010100
16 0.9599999999991269 0.0011110000101000
17 0.9199999999982538 0.00111100001010001
18 0.8399999999965075 0.001111000010100011
19 0.6799999999930151 0.0011110000101000111
20 0.35999999998603016 0.00111100001010001111
21 0.7199999999720603 0.001111000010100011110
.....
.....
.....
46 0.0390625 0.00111100001010001111010111000010100011111010111
47 0.078125 0.001111000010100011111010111000010100011111010110
48 0.15625 0.00111100001010001111010111000010100011111010111100
49 0.3125 0.00111100001010001111010111000010100011111010111000
50 0.625 0.001111000010100011111010111000010100011111010110000
51 0.25 0.001111000010100011111010111000010100011111010111000001
52 0.5 0.001111000010100011111010111000010100011111010111000010
53 0.0 0.0011110000101000111101011100001010001111010111000010 1
'0.0011110000101000111101011100001010001111010111000010 1'
```

Traduire une partie fractionnaire en binaire : le code

Remarques sur le code

- Le programme ci-dessus ne fonctionne pas très bien (testez-le !) :
- On a tout de suite des erreurs d'arrondi (mais elles n'influent pas sur le résultat).
- Ces erreurs d'arrondi empêchent la détection de la périodicité.
- Plus grave : Après avoir calculé 53 chiffres, le programme s'arrête, quel que soit le nombre de départ.
- Vous devriez pouvoir expliquer ce comportement *miraculeux* à la fin du cours...

Quand y a du flou, y a un biais

Le programme utilise la représentation interne des flottants pour construire cette même représentation interne : on boucle !

Un deuxième exemple

Traduire $\frac{1}{3}$ en binaire.

$$\frac{1}{3} \times 2 = \frac{2}{3} < 1 \text{ le chiffre en position -1 est } 0$$

$$\frac{2}{3} \times 2 = \frac{4}{3} = 1 + \frac{1}{3} \text{ le chiffre en position -2 est } 1$$

$$\frac{1}{3} \times 2 = \frac{2}{3} < 1 \text{ le chiffre en position -3 est } 0$$

On est revenu au départ :

le cycle est plus court que dans l'exemple précédent

Donc

$$\frac{1}{3} = \overline{0.01010101\cdots}_2 = \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \cdots$$

Un deuxième exemple : vérification

Donc

$$\frac{1}{3} = \overline{0.01010101\cdots}_2 = \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \cdots$$

$$\frac{1}{3} = \overline{0.01010101\cdots}_2 = \frac{1}{4} + \frac{1}{4^2} + \frac{1}{4^3} + \cdots + \frac{1}{4^i}\cdots$$

$$\frac{1}{3} = \frac{1}{4} \times (1 + \frac{1}{4} + \frac{1}{4^2} + \cdots + \frac{1}{4^i}\cdots)$$

$$\frac{1}{3} = \frac{1}{4} \times (1 + \frac{1}{3}) = \frac{1}{4} + \frac{1}{12} = \frac{3 \times 1 + 1}{12}$$

$$\frac{1}{3} = \frac{4}{12}$$

On a donc bien vérifié la validité de la représentation binaire.

Jouons un peu

Comment s'écrit $\frac{2}{3}$ en binaire ?

indices

- On pourrait faire comme pour $\frac{1}{3}$...mais non ...
- $\overline{0.1111111111\cdots}_2$, avec une suite infinie de '1', est une autre écriture du nombre '1'
- Conclusion ?
- Vérification ?

Ou bien...

$$\frac{2}{3} = 2 \times \frac{1}{3}$$

- Multiplier par 2 un nombre fractionnaire exprimé en binaire, c'est déplacer la virgule d'un chiffre vers la droite ...

En résumé

Traduction des réels en binaire

- On a un algorithme pour transformer un nombre réel en binaire :
  - Traduire la partie entière (algorithmes du cours précédent).
  - Traduire la partie décimale (voir ci-dessus).
  - Regrouper les deux morceaux.

Cependant...

- Le format n'est pas très homogène :
  - $11010101011110,1011101_2$
  - $0,00000000000001011101_2$
    Beaucoup de 0 inutiles, perte de précision.
  - $0,0001011101_2$ contient *à peu près* la même information que $0,00000000000001011101_2$
    Combien de chiffres significatifs ?

Vers une normalisation

Souhaits

- Garder un maximum de chiffres significatifs.
- Avoir une mesure directe de l'ordre de grandeur.

Améliorations

- Cadrer les nombres de telle façon que le premier chiffre significatif soit immédiatement à droite de la virgule.
- Définir en conséquence un *exposant*, pour recadrer le nombre.
- L'ensemble des chiffres significatifs s'appelle la mantisse.
- On parle de représentation en virgule flottante.

Vers une normalisation

Exemples

$$\overline{11010101011110,1011101_2} = \overline{0,1101010101111010111101_2} \times 2^{14}$$

$$\overline{0,00000000000001011101_2} = \overline{0,1011101_2} \times 2^{-13}$$

$$\overline{0,0001011101_2} = \overline{0,1011101_2} \times 2^{-3}$$

Le cas de l'exposant

- On doit aussi coder l'exposant en base 2...
- ...et on retrouve deux préoccupations :
  - Coder des nombres positifs ou négatifs de façon pratique.
  - Contrainte de place : combien de chiffres ?

La norme IEEE 754

l'IEEE

L'IEEE (Institute of Electrical and Electronics Engineers) est une association professionnelle, dont le but est de propager la connaissance dans son domaine. Elle édite des revues scientifiques, organise des colloques, et définit des normes, dont la norme IEEE 754, qui définit des formats de nombres en virgule flottante.

La norme IEEE 754

Principe général

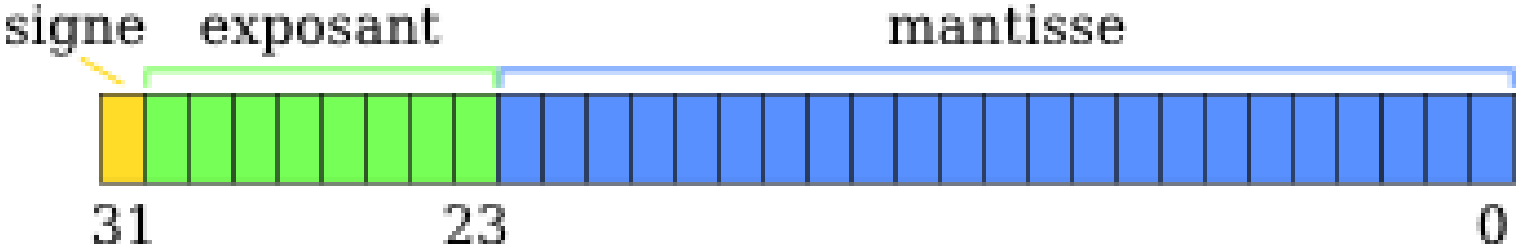
Les nombres réels sont représentés sur un format fixe d'un total de n bits :

- Un bit de signe.
- e bits pour l'exposant.
- m bits pour la mantisse.

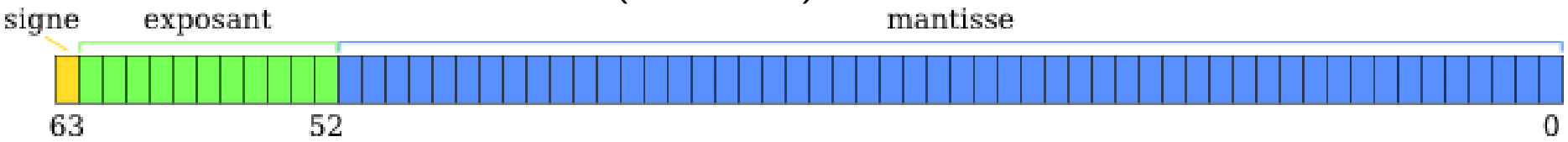
Crédits : wikipedia

La norme IEEE 754 : Les principaux formats

- Simple précision :
  1 bit de signe, 8 bits d'exposant, 23 bits de mantisse, pour un total de 32 bits (4 octets).



- Double précision :
  1 bit de signe, 11 bits d'exposant, 52 bits de mantisse, pour un total de 64 bits (8 octets).



- Double précision étendue :
  1 bit de signe, 15 bits d'exposant, 64 bits de mantisse, pour un total de 80 bits (10 octets).

Crédits : wikipedia

IEEE 754 : la mantisse

Conventions

- La mantisse est alignée de telle façon que son bit de poids fort soit égal à 1.
- De ce fait, il n'est pas nécessaire de coder ce bit, il est implicite.
- Donc la mantisse sur 23 bits (respectivement 52) est en fait de 24 bits (resp 53).

IEEE 754 : la mantisse

On ne s'intéresse qu'à la valeur de la mantisse, on reprendra l'exemple complet après l'étude de l'exposant

$$0\ 00000101\ 00110011000000000000000$$

Signe : 0
Exposant :00000101
Mantisse : $00110011000000000000000 \longrightarrow 2^{-3} + 2^{-4} + 2^{-7} + 2^{-8}$

Valeur de la mantisse : $\frac{1}{8} + \frac{1}{16} + \frac{1}{128} + \frac{1}{256}$

Ajouter le '1' implicite : $1 + \frac{1}{8} + \frac{1}{16} + \frac{1}{128} + \frac{1}{256}$

Valeur représentée : $1 + \frac{32+16+2+1}{256} = 1 + 0.1953125$

Illustration

- Considérons le nombre 21.725
- Partie entière : 21 codée 10101
- Partie décimale : 0.725 codée 0.10111001100110011001...
- En réunissant les deux parties : 10101.10111001100110011001...
- Ramener entre 1 et 2 : $1.01011011100110011001100\cdots \times 2^4$
- **Mantisse**, garder 24 bits : 1.01011011100110011001100
  - Le premier 1. est implicite, on ne le fait pas figurer : 01011011100110011001100
- **Exposant** 4, avec un biais de 127
  - $127+4=131=\overline{10000011}_2$
- On recolle signe, exposant et mantisse : 01000001101011011100110011001100

Illustration

- On recolle signe, exposant et mantisse :
  0100000110101101110011001100
- Pour plus de lisibilité, et pour gagner de la place, on peut le réécrire en hexadécimal.

$$\underbrace{0100}_{4}\overbrace{0001}^{1}\underbrace{1010}_{A}\overbrace{1101}^{D}\underbrace{1100}_{C}\overbrace{1100}^{C}\underbrace{1100}_{C}\overbrace{1100}^{C}$$

$$0x41ADCCCC$$

Mantisses : cas particuliers

Exceptions

- Les nombres dont l'exposant ne contient que des 0 sont les nombres dénormalisés.
- Il n'y a pas de 1 implicite, et leur valeur est :

$$m \times 2^{-126} \text{ en simple précision}$$

$$m \times 2^{-1022} \text{ en double précision}$$

Nombres dénormalisés

Exemple

- $N = \overline{0003\mathtt{F}000_{16}} = \overline{00000000000001111110000000000_2}$
- $\underbrace{0}_{\text{signe}}\ \overbrace{00000000}^{\text{exposant}}\ \underbrace{00000111111000000000000}_{\text{mantisse}}$
- $\mathit{mantisse} = 2^{-6} + 2^{-7} + 2^{-8} + 2^{-9} + 2^{-10} + 2^{-11}$
- $\mathit{mantisse} = 63 \times 2^{-11} = 63/2048 = 0.03076171875$
- $N = 0.03076171875 \times 2^{-126} \approx 3.616\,022\,7\mathrm{e}{-40}$

La norme IEEE 754 : l'exposant

Implémentation

- Il faut pouvoir décrire des exposants négatifs ou positifs.
- Il faut donc un signe ou équivalent :
  - Signe + valeur ?
  - Complément à deux ?
- On a choisit une autre représentation : le *biais*

Le biais

Si la représentation de l'exposant est sur n bits, l'exposant de valeur k sera codé $k - (2^{n-1} - 1)$

Taille de l'exposant	biais	exposant minimal	exposant maximal
8 bits	127	-127	128
11 bits	1023	-1023	1024

Norme IEEE : codage de l'exposant

Exemples

Simple précision

Exposant codé	Valeur décimale	Application du biais	Valeur
$\overline{00110101_2}$	53	53-127	-74
$\overline{11010101_2}$	213	213-127	86

Double précision

Exposant codé	Valeur décimale	Application du biais	Valeur
$\overline{11000110101_2}$	1589	1589-1023	566
$\overline{00111010001_2}$	465	465-1023	-558

Comparaison des codages

On a vu trois façons de coder des entiers relatifs.

binaire	signe/valeur absolue	complément à 2	biais (7)
0000	0	0	-7
0001	1	1	-6
0010	2	2	-5
0011	3	3	-4
0100	4	4	-3
0101	5	5	-2
0110	6	6	-1
0111	7	7	0
1000	-0	-8	1
1001	-1	-7	2
1010	-2	-6	3
1011	-3	-5	4
1100	-4	-4	5
1101	-5	-3	6
1110	-6	-2	7
1111	-7	-1	8

Pourquoi un biais ?

Le rôle limité de l'exposant

- Les opérations qu'on devra effectuer sur l'exposant sont limitées :
  - Comparer deux exposants (pour l'addition...)
  - Additionner deux exposants (pour la multiplication...)
- La comparaison est plus simple avec un biais qu'avec les autres représentations :
  - Biais : comparaison simple (ordre croissant dans le tableau)
  - Autres : il faut d'abord s'occuper du signe, puis comparer les magnitudes.

Norme IEEE : exceptions

Exposants spéciaux

Les deux exposants extrêmes ont un rôle particulier :

- Les nombres dont l'exposant vaut 255 (simple précision) ou 1023 (double précision) (i.e. tous les bits de l'exposant sont à 1) représentent un infini ou un objet qui n'est pas un nombre (NaN : Not A Number)
- Les nombres dont l'exposant a tous ses bits à 0 sont des nombres dénormalisés : voir plus haut.

NaN et infini

Not a Number

- On peut obtenir un résultat numérique qui n'est pas un nombre en effectuant un calcul illicite :
  - Racine carrée d'un nombre négatif
  - $0/0$, $\infty - \infty$ ...
- Dans la norme IEEE754, les NaN ont un exposant composé uniquement de 1, et une mantisse non nulle.
- On peut se servir de la mantisse non nulle pour indiquer le type d'erreur.

Vers l'infini et au-delà !

- Les infinis ont un exposant composé uniquement de 1, et une mantisse nulle.
- Il y a deux infinis, selon le signe.

Quelques valeurs

Zéro

On n'a pas fait figurer le signe

Exposant	Mantisse	Valeur
0000 0000	000 0000 0000 0000 0000 0000	0

Plus petit nombre positif dénormalisé

On n'a pas fait figurer le signe

Exposant	Mantisse	Valeur
0000 0000	000 0000 0000 0000 0000 0001	$\approx 1.4 \times 10^{-45}$

$$m = 2^{-23}$$

$$e = -126$$

$$N = 2^{-149} \approx 1.401\,298\,5\mathrm{e}{-45}$$

Quelques valeurs

Deuxième plus petit nombre positif dénormalisé

On n'a pas fait figurer le signe

Exposant	Mantisse	Valeur
0000 0000	000 0000 0000 0000 0000 0010	$\approx 2.80 \times 10^{-45}$

$$m = 2^{-22}$$

$$e = -126$$

$$N = 2^{-148} \approx 2.802\,596\,9\mathrm{e}{-45}$$

Ecart avec le plus petit nombre dénormalisé (i.e. e nombre précédent) : $1.401\,298\,5\mathrm{e}{-45}$

Quelques valeurs

Plus grand nombre dénormalisé

On n'a pas fait figurer le signe

Exposant	Mantisse	Valeur
0000 0000	111 1111 1111 1111 1111 1111	$\approx 1.175\,494\,2\mathrm{e}{-38}$

$$m = 2^{-1} + 2^{-2} + \cdots + 2^{-23} = 2^{-1} \times (1 + 2^{-1} + 2^{-2} + \cdots + 2^{-22})$$

$$m = 2^{-1} \times \frac{1 - 2^{-23}}{1 - 2^{-1}} = 1 - 2^{-23}$$

$$e = -126$$

$$N = (1 - 2^{-23}) \times 2^{-126} \approx 1.175\,494\,2\mathrm{e}{-38}$$

Quelques valeurs

Plus petit nombre positif normalisé

On n'a pas fait figurer le signe

Exposant	Mantisse	Valeur
0000 0001	000 0000 0000 0000 0000 0000	$\approx 1.175\,494\,4\mathrm{e}{-38}$

$$m = 1$$

$$e = 1 - 127$$

$$N = 2^{-126} \approx 1.175\,494\,4\mathrm{e}{-38}$$

Ecart avec le plus grand nombre dénormalisé (i.e. le nombre précédent) :

$$2^{-126} - (1 - 2^{-23}) \times 2^{-126}$$

$$2^{-23} \times 2^{-126} = 2^{-149} \approx 1.401\,298\,5\mathrm{e}{-45}$$

Quelques valeurs

Un

Exposant	Mantisse	Valeur
0111 1111	000 0000 0000 0000 0000 0000	1

- Exposant réel : 0
- Exposant biaisé : 0+127
- Mantisse : 1 (implicite)
- Valeur : 1×2^0

Quelques valeurs

Plus petit nombre juste après 1

Exposant	Mantisse	Valeur
0111 1111	000 0000 0000 0000 0000 0001	≈ 1.00000011921

- Exposant réel : 0
- Exposant biaisé : 0+127
- Mantisse : $1 + 2^{-23}$
- Valeur : $(1 + 2^{-23}) \times 2^0 \approx 1.000\,000\,119\,21$
- Ecart avec 1 : $2^{-23} \approx 1.192\,092\,9\mathrm{e}{-7}$

Quelques valeurs

Plus grand nombre normalisé

Exposant	Mantisse	Valeur
1111 1110	111 1111 1111 1111 1111 1111	$\approx 3.402\,823\,5\mathrm{e}38$

Exposant biaisé : 254

Exposant réel : 254-127=127

Mantisse :

$$1 + 2^{-1} + 2^{-2} + \cdots + 2^{-23} = \frac{1 - 2^{-24}}{1 - 2^{-1}} = 2 \times (1 - 2^{-24})$$

Valeur :

$$2 \times (1 - 2^{-24}) \times 2^{127} = 2^{128} - 2^{104} \approx 3.402\,823\,5\mathrm{e}38$$

Quelques valeurs

Presque plus grand nombre normalisé

Exposant	Mantisse	Valeur
1111 1110	111 1111 1111 1111 1111 1110	$\approx 3.402\,823\,3\mathrm{e}38$

Exposant biaisé : 254

Exposant réel : 254-127=127

Mantisse :

$$1 + 2^{-1} + 2^{-2} + \cdots + 2^{-22} = \frac{1 - 2^{-23}}{1 - 2^{-1}} = 2 \times (1 - 2^{-23})$$

Valeur :

$$2 \times (1 - 2^{-23}) \times 2^{127} = 2^{128} - 2^{105} \approx 3.402\,823\,3\mathrm{e}38$$

Ecart avec le plus grand nombre :

$$2^{128} - 2^{104} - (2^{128} - 2^{105}) = 2^{105} - 2^{104} = 2^{104} = 2.028\,241\mathrm{e}31$$

Conclusions

Répartition des nombres

- L'écart entre deux nombres consécutifs n'est pas constant.
- Ce n'était pas le cas pour les entiers.
- Il varie de $1.401\,298\,5\mathrm{e}{-45}$ aux alentours de zéro à $2.028\,241\mathrm{e}31$ autour du plus grand nombre représentable en double précision.
- Par contre le rapport écart/valeur semble de l'ordre de 10^{-7} (à vérifier ...)

Conclusions

Remarques

Les valeurs indiquées en base 10 (case de droite des tableaux) sont approximatives et à prendre avec des pincettes : j'ai utilisé un programme de calcul sur mon ordinateur, qui utilise la norme IEEE pour représenter les flottants.
Pour obtenir des résultats indépendants, il aurait fallu calculer les puissances négatives de 2 à la main : faisable mais très fastidieux.

Opérations de base sur les flottants

Addition

- Ramener les deux nombres au même exposant.
- Additionner les mantisses.
- Recadrer la mantisse du résultat et modifier l'exposant en conséquence.

Addition : exemple en simple précision

- Effectuons l'addition 0.75+0.09375
- Codage de 0.75 :
  - $0.75 = 0.5 + 0.25$ soit $\overline{0,11}_2$
  - Mantisse : $10000000000000000000000_2$
  - Exposant : -1. Avec un biais de 127 : $-1 + 127 = 126$ soit 01111110_2
- Codage de 0.09375 :
  - $0.09375 = 0.0625 + 0.03125 = \frac{1}{16} + \frac{1}{32}$ soit $\overline{0,00011}_2$
  - Mantisse : $10000000000000000000000_2$
  - Exposant : -4. Avec un biais de 127 : $-4 + 127 = -123$ soit 01111011_2

Remarque : les deux mantisses sont égales !

Addition : exemple

- Effectuons l'addition 0.75+0.09375
- Les deux exposants sont différents : il faut se ramener au même exposant.
- On modifie le nombre dont l'exposant est le plus petit : $\overline{1,1_2} \times 2^{-4} = \overline{0,0011_2} \times 2^{-1}$
- Les deux nombres ont le même exposant, on fait la somme des mantisses
- $\overline{1,1_2} + \overline{0,0011_2} = \overline{1,1011_2}$
- Nouvelle mantisse : $\overline{10110000000000000000000_2}$
- Exposant : -1 (biaisé en 126)

Addition : exemple

- Si le résultat de l'addition des mantisses est plus grand que 1, il faudra modifier l'exposant et décaler la mantisse.
- Exemple : additionner 0.75 et 0.75.
- $0.75 = 0.5 + 0.25$ soit $\overline{0,11}_2$
- Mantisse : $\overline{10000000000000000000000}_2$
- Même exposant, pas de décalage.
- Addition des deux mantisses réelles : $\overline{1,1}_2 + \overline{1,1}_2 = \overline{11,0}_2$
- Recadrer la mantisse entre 1 et 2 : $\overline{1,10}_2$
- Ajouter 1 à l'exposant.

Opérations de base sur les flottants

Multiplication

- Multiplier les mantisses.
- Recadrer la mantisse du résultat (voyez-vous pourquoi ?) et modifier les exposants.
- Additionner les exposants.
- Ajuster l'exposant (pourquoi ?)

Multiplication : un exemple

- Multiplier 0.75 par 0.09375=0.0703125 ($\overline{0,0001001_2}$)
- Extraire les mantisses complètes : $\overline{1,1_2} \times \overline{1,1_2} = \overline{10,01_2}$
- Ajouter les exposants : -1+(-4)=-5. Mais attention, ils sont biaisés.
- On compte un biais en trop lors de l'addition : 126+123=249-127=122, soit -5
- recadrer la mantisse entre 1 et 2 : $\overline{10,01_2} = \overline{1,001_2} \times 2^1$
- Ajouter 1 à l'exposant.
- $0.75 \times 0.09375 = \overline{1,001_2} \times 2^{-4}$

Miscellanées

Connaître le format sous python

```
>>> sys.float_info
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308, min=2.2250738585072014e-308,
min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53, epsilon=2.220446049250313e-16, radix=2, rounds=1)
```

Toutes les machines n'utilisent pas la norme IEEE 754 : testez votre calculatrice scientifique !