Codage des nombres décimaux

1/ Codage des « réels »

On a vu qu'il était possible de coder des nombres à virgules en utilisant des puissances de 2 négatives :

$5{,}25 = 1x4 + 0x2 + 1x1 + 0x0{,}5 + 1x0{,}25 = 1x2^0 + 0x2^1 + 1x2^2 + 0x2^{-1} + 1x2^{-2} = (101{,}01)_2$

Il ne sera donc pas possible de coder exactement en binaire tous les réels, comme par exemple π ou $\sqrt{3}$. Coder un réel implique forcément des erreurs d'arrondis.

Le standard IEEE 754 (1985) définit un format de représentation des nombres à virgule flottante (float) en binaire. C'est ce codage qui sera présenté ici. Parmi les formats proposés, nous ne retiendrons que la représentation binaire simple précision (32 bits) et double précision (64 bits), utilisé par Python (le principe reste le même pour des mots de tailles différentes).

Le principe s'expose facilement en système décimal, par ce qui est appelée la notation scientifique d'un nombre x sous la forme $x = \pm m.10^e$, où :

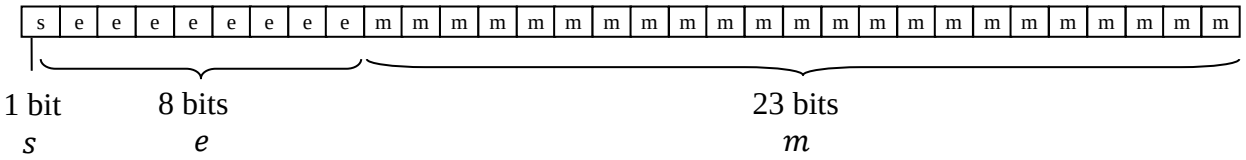
- $m \in [1\,;10[$ est appelé la mantisse ou significande ;
- e, entier relatif, est appelé l'exposant.

En représentation binaire simple précision (sur 32 bits) : $x = \pm m.2^e$

- $m \in [1\,;2[$, codé sur 23 bits : $m = 1 + \sum m_i.2^{-i}$
- e : entier relatif, codé sur 8 bits : $e = \sum e_i.2^i - 127$
- s : un bit pour le signe

$$x = (-1)^s \cdot m \cdot 2^e = (-1)^s \cdot \left(1 + \sum_{i=1}^{23} m_i \cdot \frac{1}{2^i}\right) \cdot 2^{\sum_0^7 e_i.2^i - 127}$$

Le code simple précision va se mettre sous cette forme :

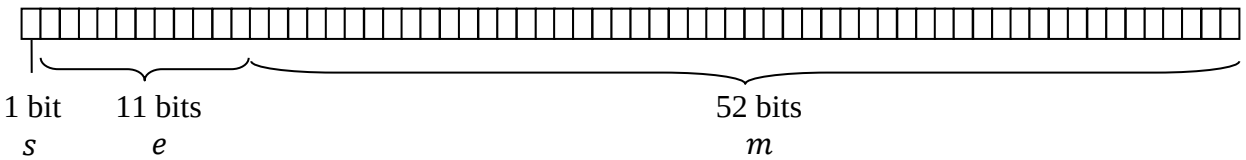


On remarque que l'exposant a été décalé de 127. C'est pour permettre les exposants négatifs. L'exposant peut ainsi varier de -127 à +128. En réalité, l'exposant est compris entre -126 et 127, les valeurs -127 et 128 étant réservées aux codages de cas particuliers.

Le seul cas particulier à connaitre est le 0, qui ne peut être codé avec la convention précédente. On considère alors que si $e = -127$ ($e_i = 0$ pour tout i) :

$$x = (-1)^s \sum_{i=1}^{23} m_i \cdot \frac{1}{2^i}$$

De même, en représentation binaire double précision (sur 64 bits) : $x = \pm m.2^e$



- $m \in [1\,;2[$, codé sur 52 bits : $m = 1 + \sum m_i.2^{-i}$
- e entier relatif, codé sur 11 bits : $e = \sum e_i.2^i - 1023$
- un bit pour le signe

Application 1 :

a) Donner la valeur décimale de 0010 1110 1101 0100 0000 0000 0000 0000
b) Ecrire -10,125 en utilisant la norme IEEE 754 en simple précision.
c) Ecrire 0,1 en utilisant la norme IEEE 754 en simple précision.

2/ Limites du codage des flottants

2.1/ *Plus grand, plus petit nombre*

$$x = 2^{1023}\left(1 + \sum_{1}^{52} 2^{-i}\right) = 2^{1023} + 2^{970}.\sum_{0}^{51} 2^{i} = 2^{1023} + 2^{970}(2^{52} - 1) = 2^{1023} + 2^{1023} - 2^{970}$$
$$= (2^{1024} - 2^{970}) \approx \pm 1{,}7976931348623157 \times 10^{308}$$

2.2/ *Nombres proches de 0*

Selon la règle générale :
Le plus petit nombre positif peut être obtenu, selon la règle, pour l'exposant minimum (-1022) et la mantisse minimale (tous les m_i à 0). Alors :

$$x = 2^{-1022} \approx 2{,}2250738585072020 \times 10^{-308}$$

Si $e = -1023$, la règle ne s'applique plus. On considère que le nombre est nul.
L'exception est basée sur le fait que x est suffisamment proche de 0 pour être codé par :

$$x = (-1)^s \cdot 2^{-1022} \sum m_i . 2^{-i}$$

2.3/ *Arrondi intrinsèque*

Il est rare que le résultat d′ un calcul faisant intervenir deux nombres à virgule flottante donne un résultat représentable exactement sur 64 bits. Même sans effectuer de calcul, la plupart des nombres décimaux ne sont pas représentables exactement dans ce format.
Ainsi, par exemple, le nombre 0*,* 4 admet pour développement en base 2 :

$$\frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^6} + \frac{1}{2^7} + \frac{1}{2^{10}} + \frac{1}{2^{11}} + \frac{1}{2^{14}} + \frac{1}{2^{15}} + \ldots = \underline{0,011001100110011}\ldots$$

Il faudrait donc un nombre de bits infini pour le représenter.

Si nous utilisons 32 bits, une valeur approchée serait : **0-01111101-10011001100110011001101** ou
$2^{-2}\left(1 + \frac{1}{2} + \frac{1}{16} + \frac{1}{32} + \frac{1}{256} + \frac{1}{512} + \frac{1}{4096} + \frac{1}{8192} + \frac{1}{65536} + \frac{1}{131072} + + \frac{1}{2097152} + \frac{1}{4194304} + \frac{1}{16777216}\right)$
$\approx 0{,}39970703423023224$
Il n'existe pas de représentation flottante plus proche de 0,4.

Le codage implique donc une imprécision sur le nombre codé.
Il ne pourra être représenté qu'avec :

- la précision des 54 chiffres binaires significatifs de la mantisse ;
- soit une erreur d'arrondi relative de $2^{-55} = 2{,}8.10^{-17} \approx 10^{-16}$;
- soit 16 chiffres décimaux significatifs.

2.4/ ***Figures explicatives***