La virgule flottante

Représentation des nombres décimaux

En base 10, les nombres décimaux sont une extension, après la virgule, des nombres entiers, en utilisant des puissances de 10 avec des exposants négatifs :

$$132{,}92 = 1\times10^{2} + 3\times10^{1} + 2\times10^{0} + 9\times10^{-1} + 2\times10^{-2}$$

En binaire ce sont des puissances de 2 avec des exposants négatifs qui sont utilisés :

Binaire		Décimal
$101{,}11$	$=$	$1\times2^{2} + 0\times2^{1} + 1\times2^{0} + 1\times2^{-1} + 1\times2^{-2}$
	$=$	$4 + 1 + 0{,}5 + 0{,}25$
	$=$	$5{,}75$

Exercice 1 : Convertir en base 10 les nombres binaires ci-dessous :

1) 1,1 2) 10,01 3) 0,111 4) 1101,001

Du décimal au binaire

En base 10, pour décaler la virgule, il faut multiplier ou diviser par 10. En multipliant par une puissance assez grande, il est possible d'obtenir un nombre entier. Ainsi $1{,}57\times10^{2} = 157$. De la même manière, en base 2, en multipliant ou en divisant par 2, on déplace la virgule et en multipliant par une puissance de 2 assez grande, on obtient un nombre entier.
Pour convertir un nombre décimal en base 10 en binaire, il faut donc commencer par convertir la partie entière en binaire, puis multiplier progressivement la partie décimale par 2 en notant les unités obtenues au fur et à mesure.

Par exemple, pour convertir $5{,}625_{10}$, on commence par convertir $5_{10} = 101_{2}$. Ensuite, on multiplie la partie décimale par 2 en notant le chiffre des unités obtenu.

$0{,}625\times2 = 1{,}25$
$0{,}25\times2 = 0{,}5$
$0{,}5\times2 = 1{,}0$

On obtient donc $5{,}625_{10} = 101{,}101_{2}$. Malheureusement, si tous les nombres décimaux en binaire correspondent à des nombres décimaux en base 10, la réciproque n'est pas vraie.

Ainsi, il y a des nombres décimaux en base 10 qui ne peuvent pas être écrits avec un nombre finis de chiffres après la virgule en binaire. C'est le cas de 0,2 :
On remarque que $0{,}2_{10} = 0{,}0011001100110011\ldots_{2}$. Les décimales 0011 se répètent à l'infini.

$0{,}2\times2 = 0{,}4$
$0{,}4\times2 = 0{,}8$
$0{,}8\times2 = 1{,}6$
$0{,}6\times2 = 1{,}2$

On peut aussi le noter $0{,}[0011]$. La répétition n'arrive pas forcément juste après la virgule. Par exemple, $0{,}3_{10} = 0{,}0[1001]_{2}$.

Exercice 2 : Convertir en binaire les nombres en base 10 ci-dessous :

1) 0,3 2) 0,35 3) 0,125 4) 0,175

Représentation en virgule flottante

Dans la plupart des langages de programmation, la taille pour la représentation des données est fixe. Il y a donc un nombre fini de bits disponibles pour représenter les décimaux. Il faut représenter la partie entière, la partie décimale et la position de la virgule. Pour cela, on utilise la notation scientifique. En base 10, $253{,}27 = 2{,}5327\times10^{2}$. De la même manière, en binaire, on peut écrire $101{,}101 = 1{,}01101\times2^{2}$. On peut remarquer qu'en binaire, le chiffre avant la virgule dans la **mantisse** est forcément un 1. On peut donc l'omettre. Pour représenter le nombre il suffit de garder les chiffres après la virgule de la mantisse et l'**exposant**.

On peut également rajouter un bit de signe pour les nombres relatifs.
Selon le nombre, on peut avoir besoin d'un exposant positif (110,01) ou d'un exposant négatif (0,001). Il serait possible d'utiliser un complément à 2 pour représenter l'exposant, mais la norme IEEE 754 de 1985 propose plutôt de soustraire une constante fixée à l'avance. Au final, on obtient l'écriture $(-1)^{\text{signe}} \times 1,\text{mantisse} \times 2^{\text{exposant}-\text{constante}}$.
Le nombre de bits prévu pour chaque partie est défini de la manière suivante :

Taille totale	Signe	Exposant	Mantisse	Constante	Précision	Chiffres significatifs
32 bits	1 bit	8 bits	23 bits	127	24 bits	environ 7
64 bits	1 bit	11 bits	52 bits	1023	53 bits	environ 16

En 32 bits (simple précision), l'exposant est codé sur 8 bits et peut donc aller de 0 à 255. Avec la constante, on obtient donc un exposant allant de $0-127=-127$ à $255-127=128$. Cela permet de représenter des nombres très petits et des nombres très grands. La plupart des systèmes modernes utilisent 64 bits (double précision), ce qui est largement suffisant dans la plupart des cas.

Par exemple, la version 32 bits de 211,3 s'obtient en faisant les étapes ci-contre. On obtient :
0 10000110 10100110100110011001101

Décimal		Binaire
211,3	$\approx$	11010011,0100110011001101
	$\approx$	$1,10100110100110011001101 \times 2^7$
$7+127$	$=$	10000110

Ce qui correspond à 211,3000030517578125. On a bien une précision d'au moins 7 chiffres. On peut remarquer que le dernier chiffre de la mantisse ne correspond pas à la periodicité observé sur les bits décimaux. C'est parce que la valeur obtenue est arrondie pour être le plus proche possible de la vraie valeur.

Exercice 3 : Les nombres suivants sont écrits en simple précision.

1) On considère 1 01111110 11110000000000000000000.
   a) Quel est le signe du nombre ?
   b) Convertir $e+127$ en base 10 et en déduire la valeur de e en base 10.
   c) Écrire $1,m \times 2^e$ en binaire, sans écriture scientifique.
   d) Convertir le résultat en base 10 et en déduire la valeur en base 10 du nombre en virgule flottante.
2) Mêmes questions pour 0 10000011 11100000000000000000000.

Exercice 4 : Répondre aux questions pour chacun des nombres suivants :

- 128
- $-32,75$
- $1/3$

1) Quel est le bit de signe ?
2) Convertir le nombre en binaire et l'écrire sous la forme $1,m \times 2^e$.
3) Convertir $e+127$ en binaire.
4) Donner l'écriture en virgule flottante ci-dessous en simple précision.

Erreurs de calcul

Puisque la plupart des nombres réels sont représentés par des valeurs approchées, lorsqu'on enchaîne les calculs, les erreurs peuvent s'ajouter. L'ordre des opérations peut également changer le résultat obtenu. Il est d'autant plus important de faire attention lors des tests d'égalités sur les nombres réels. Il est conseillé de vérifier que la différence entre les deux nombres est inférieure à un certain seuil, comme 10^{-12}.
Le choix de la norme IEEE 754 est un compromis entre précision et rapidité. Depuis sa création, la plupart des processeurs intégrent des unités de calculs dédiées aux nombres en virgule flottante. On appelle FLOPS le nombre d'opérations en virgule flottante que peut faire un processeur peut faire en une seconde. Les ordinateurs modernes ont des capacités de calculs de l'ordre du téra-flops, c'est-à-dire mille milliards d'opérations par seconde.