Nombres, machines, calculs

Roland Denis (Thierry Dumont)
Institut Camille Jordan, Lyon.

8 Février 2018

Nombres

Avec quels nombres peut-on calculer ?

- $\mathbb{N}$, entiers naturels : $\{0, 1, \ldots\}$,
- $\mathbb{Z}$, entiers relatifs : $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$,
- $\mathbb{Q}$, rationnels : a/b avec $a \in \mathbb{Z}$, $b \in \mathbb{Z}^*$,
- $\mathbb{R}$, nombres réels.
- $\mathbb{C}$, nombres complexes.

Quelles représentations en machine ? Que peut-on représenter ?

- $\mathbb{N}$ et $\mathbb{Z}$: $p = 32$ ou $p = 64$ bits avec un bit de signe ($\mathbb{Z}$) ou non signés $\mathbb{N}$.

- $\mathbb{N}$ et $\mathbb{Z}$: $p = 32$ ou $p = 64$ bits avec un bit de signe ($\mathbb{Z}$) ou non signés $\mathbb{N}$.

  Seuls les nombres compris entre $-2^{p-1} + 1$ et $2^{p-1} - 1$ (cas signé) ou $-2^p + 1$ et $2^p - 1$ (cas non signé) sont représentables.

- $\mathbb{N}$ et $\mathbb{Z}$: $p = 32$ ou $p = 64$ bits avec un bit de signe ($\mathbb{Z}$) ou non signés $\mathbb{N}$.

  Seuls les nombres compris entre $-2^{p-1} + 1$ et $2^{p-1} - 1$ (cas signé) ou $-2^p + 1$ et $2^p - 1$ (cas non signé) sont représentables.

  Comment faire pour représenter de grands entiers ?

- $\mathbb{N}$ et $\mathbb{Z}$: $p = 32$ ou $p = 64$ bits avec un bit de signe ($\mathbb{Z}$) ou non signés $\mathbb{N}$.

  Seuls les nombres compris entre $-2^{p-1} + 1$ et $2^{p-1} - 1$ (cas signé) ou $-2^p + 1$ et $2^p - 1$ (cas non signé) sont représentables.

  Comment faire pour représenter de grands entiers ?
  Représenter les entiers sur plusieurs mots (sur plusieurs entiers machine). $=>$ lent ! (exemple : bibliothèque gmp, beaucoup utilisée en cryptographie et théorie des nombres).

- $\mathbb{N}$ et $\mathbb{Z}$: $p = 32$ ou $p = 64$ bits avec un bit de signe ($\mathbb{Z}$) ou non signés $\mathbb{N}$.

  Seuls les nombres compris entre $-2^{p-1} + 1$ et $2^{p-1} - 1$ (cas signé) ou $-2^p + 1$ et $2^p - 1$ (cas non signé) sont représentables.

  Comment faire pour représenter de grands entiers ?
  Représenter les entiers sur plusieurs mots (sur plusieurs entiers machine). $=>$ lent ! (exemple : bibliothèque gmp, beaucoup utilisée en cryptographie et théorie des nombres).

- $\mathbb{Q}$, rationnels. Représentation par paires d'entiers signés (réduction).

Calculer avec les nombres rationnels ?

Pour :

- Tout nombre réel peut être approché d'aussi près qu'on veut par des nombres rationnels.

Calculer avec les nombres rationnels ?

Pour :

- Tout nombre réel peut être approché d'aussi près qu'on veut par des nombres rationnels.

Contre :

- On ne peut pas borner la taille (nombre de chiffres) des numérateurs et dénominateurs.

Calculer avec les nombres rationnels ?

Pour :

- Tout nombre réel peut être approché d'aussi près qu'on veut par des nombres rationnels.

Contre :

- On ne peut pas borner la taille (nombre de chiffres) des numérateurs et dénominateurs.

Conséquence : le coût des opérations élémentaires n'est pas constant.

Coût des algorithmes

Coût des algorithmes :

nombre d'opérations élémentaires pour résoudre un problème.

Coût des algorithmes

Coût des algorithmes :

nombre d'opérations élémentaires pour résoudre un problème.

Il faut s'entendre sur ce qu'est une *opération élémentaire !*

Coût des algorithmes

Coût des algorithmes :

nombre d'opérations élémentaires pour résoudre un problème.

Il faut s'entendre sur ce qu'est une *opération élémentaire !*

Exemple : système linéaire de n équations à n inconnues $Ax = B$, par la méthode de Gauss :

- même coût pour toutes les opérations : $C\ n^3$,
- dans $\mathbb{Q}$: coût non polynomial.

Vers les nombres flottants

Il est impossible de représenter exactement tous les nombres réels avec une quantité d'information finie !

Vers les nombres flottants

Il est impossible de représenter exactement tous les nombres réels avec une quantité d'information finie !

Exemple : Un nombre normal est un nombre réel tel que la fréquence d'apparition de tout n-uplet dans la suite de ses « décimales » dans toute base est équirépartie.

Vers les nombres flottants

Comment représenter les nombres réels *de manière approchée* avec une quantité d'information fixe ?
Consensus pour utiliser les nombres à virgule flottante.

Nombres à virgule flottante

Exemple : flottants en base 10.

Nombres à virgule flottante

Exemple : flottants en base 10.

$$\begin{array}{r|c|c} 0.0121 & => & +0.121\ 10^{-1} \\ -5837.25 & => & -0.583725\ 10^{4} \end{array}$$

Nombres à virgule flottante

Exemple : flottants en base 10.

0.0121	$=>$	$+0.121\ 10^{-1}$
-5837.25	$=>$	$-0.583725\ 10^{4}$

(*signe*)	(*mantisse*)	(*exposant*)
$+$	121	-1
$-$	583725	4

La mantisse commence par le premier chiffre non nul, en commençant par la gauche.

Nombres à virgule flottante

Ensemble $F(\beta, r, m, M)$ de nombres à virgule flottante :

- une base $\beta \geq 2$,
- un nombre de chiffres r,
- deux entiers relatifs m et M.

$$x = (-1)^s \; 0.d_1 d_2 \ldots d_r \cdot \beta^j,$$

Nombres à virgule flottante

Ensemble $F(\beta, r, m, M)$ de nombres à virgule flottante :

- une base $\beta \geq 2$,
- un nombre de chiffres r,
- deux entiers relatifs m et M.

$$x = (-1)^s \ 0.d_1 d_2 \ldots d_r \cdot \beta^j,$$

- les chiffres d_i sont des nombres entiers qui vérifient $0 \leq d_i < \beta$ pour $i > 1$ et $0 < d_1 < \beta$.

Nombres à virgule flottante

Ensemble $F(\beta, r, m, M)$ de nombres à virgule flottante :

- une base $\beta \geq 2$,
- un nombre de chiffres r,
- deux entiers relatifs m et M.

$$x = (-1)^s \ 0.d_1 d_2 \ldots d_r \cdot \beta^j,$$

- les chiffres d_i sont des nombres entiers qui vérifient $0 \leq d_i < \beta$ pour $i > 1$ et $0 < d_1 < \beta$.
- Le nombre de chiffres r est la *précision*.

Nombres à virgule flottante

Ensemble $F(\beta, r, m, M)$ de nombres à virgule flottante :

- une base $\beta \geq 2$,
- un nombre de chiffres r,
- deux entiers relatifs m et M.

$$x = (-1)^s \ 0.d_1 d_2 \ldots d_r \cdot \beta^j,$$

- les chiffres d_i sont des nombres entiers qui vérifient $0 \leq d_i < \beta$ pour $i > 1$ et $0 < d_1 < \beta$.
- Le nombre de chiffres r est la *précision*.
- l'indicateur de signe s vaut 0 ou 1.

Nombres à virgule flottante

Ensemble $F(\beta, r, m, M)$ de nombres à virgule flottante :

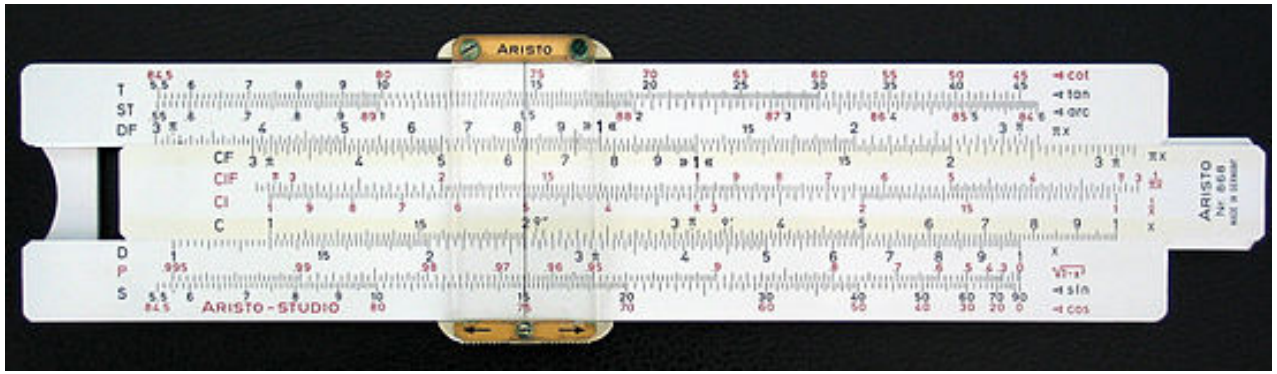
- une base $\beta \geq 2$,
- un nombre de chiffres r,
- deux entiers relatifs m et M.

$$x = (-1)^s \; 0.d_1 d_2 \ldots d_r \cdot \beta^j,$$

- les chiffres d_i sont des nombres entiers qui vérifient $0 \leq d_i < \beta$ pour $i > 1$ et $0 < d_1 < \beta$.
- Le nombre de chiffres r est la *précision*.
- l'indicateur de signe s vaut 0 ou 1.
- L'*exposant* j est compris entre les deux entiers m et M.
- $0.d_1 d_2 \ldots d_r$ est la *mantisse*.

Virgule flottante

Pas si neuf...



Selon D. Knuth, les Babyloniens utilisaient un système de virgule flottante en base 60 !

Nombres à virgule flottante, et norme IEE

Bien sûr on ne stocke que s, $d_1 d_2 \dots d_r$ et j.
Cas *particulier* de la base 2 : d_1 vaut toujours 1 et n'a pas besoin d'être stocké.

Nombres à virgule flottante, et norme IEE

Bien sûr on ne stocke que s, $d_1 d_2 \dots d_r$ et j.
Cas *particulier* de la base 2 : d_1 vaut toujours 1 et n'a pas besoin d'être stocké.

Norme IEE 754

1985.
Définit plusieurs types de flottants.
Exemple : nombres stockés sur 64 bits (« double précision ») :

- le signe s est codé sur 1 bit,
- la mantisse sur 53 bits (dont 52 seulement sont stockés),
- l'exposant sur 11 bits.

Nombres à virgule flottante, et norme IEE

Bien sûr on ne stocke que s, $d_1 d_2 \dots d_r$ et j.
Cas *particulier* de la base 2 : d_1 vaut toujours 1 et n'a pas besoin d'être stocké.

Norme IEE 754

1985.
Définit plusieurs types de flottants.
Exemple : nombres stockés sur 64 bits (« double précision ») :

- le signe s est codé sur 1 bit,
- la mantisse sur 53 bits (dont 52 seulement sont stockés),
- l'exposant sur 11 bits.

Les nombres sont donc de la forme :

$$(-1)^s \; 0.d_1 d_2 \dots d_{53} \cdot 2^{j-1023}.$$

Ils correspondent au type `double` du langage `C`.

Nombres à virgule flottante : normaux et sous-normaux

Nombres normaux

Les nombres normaux sont ceux dont la représentation est celle vu précédemment :

- En base 10, la mantisse est de la forme $m = 0, n_1 n_2 ... n_m$ où n_i est compris entre 0 et 9 avec n_1 non nul.
- En base 2, c'est un peu différent, la mantisse est de la forme $m = 1, n_1 n_2 ... n_m$, où les n_i sont dans $\{0, 1\}$. Le 1 avant la virgule n'est pas codé, il est implicite.

Nombres sous-normaux

Afin d'avoir une meilleure répartition des nombres proches de 0, on définit les nombres sous normaux. Lorsque l'exposant est minimal ($e = e_{\min}$), on autorise la mantisse à ne plus suivre le modèle des nombres normaux :

- en base 10, n_1 peut-être égal à 0,
- en base 2, la mantisse est de la forme $m = 0, n_1 n_2 ... n_m$.

Nombres à virgule flottant : limites

Propriétés et définitions

- Plus petit nombre normal représentable : $\epsilon = \pm 0.1\beta^{e_{min}}$
- Plus petit nombre sous normal représentable : $\epsilon = \pm 0.0 \ldots 1\beta^{e_{min}}$
- Plus grand nombre représentable : $M = \pm 0.(\beta - 1) \ldots (\beta - 1)\beta^{e_{max}}$
- Dépassement de capacité : nombre plus petit que $\epsilon \Longrightarrow$ débordement par valeur inférieure (underflow) ; Nombre plus grand que $M \longrightarrow$ débordement par valeur supérieure (overflow).

Limites

- Certains réels sont par définition impossibles à représenter en numération classique : $1/3$, π ...
- La représentation en un nombre fixe d'octets oblige le processeur à faire appel à des approximations afin de représenter les réels.
- Le degré de précision de la représentation par virgule flottante des réels est directement proportionnel au nombre de bits alloués à la mantisse, alors que le nombre de bits alloués à l'exposant conditionnera l'amplitude de l'intervalle des nombres représentables.

Nombres à virgule flottante

Et si on veut plus de précision ?

- type `long double` pas très normalisé.
- *Bibliothèques :*
  `GNU MPFR` (Gnu Multiple Precision) `http://www.mpfr.org/`
  (Paul Zimmermann, Inria Nancy).
  Mais c'est forcément *très* lent.

Nombres à virgule flottante

Et si on veut plus de précision ?

- type `long double` pas très normalisé.
- *Bibliothèques :*
  `GNU MPFR` (Gnu Multiple Precision) `http://www.mpfr.org/`
  (Paul Zimmermann, Inria Nancy).
  Mais c'est forcément *très* lent.

Note : si vous compilez `GCC` il vous faut `mpfr`. Sert à évaluer des expressions constantes à la compilation
(Exemple : `double pi=4.0 * atan(1.0)`).

Quelques propriétés des nombres à virgule flottante

Les ensembles $F(\beta, r, m, M)$ décrivent seulement un sous-ensemble fini des nombres réels.

Quelques propriétés des nombres à virgule flottante

Les ensembles $F(\beta, r, m, M)$ décrivent seulement un sous-ensemble fini des nombres réels.

Arrondi :

- Si $x \in F(\beta, r, m, M)$, $Arrondi(x) = x$.
- Si $x \notin F(\beta, r, m, M)$:
  - $Arrondi(x) =$ nombre de $F(\beta, r, m, M)$ le plus proche de x.
  - $Arrondi(x) =$ nombre de $F(\beta, r, m, M)$ immédiatement supérieur.
  - $Arrondi(x) =$ nombre de $F(\beta, r, m, M)$ immédiatement inférieur.
  - $Arrondi(x) =$ nombre de $F(\beta, r, m, M)$ le plus proche en direction de zéro.

Quelques propriétés des nombres à virgule flottante

Les ensembles $F(\beta, r, m, M)$ décrivent seulement un sous-ensemble fini des nombres réels.

Arrondi :

- Si $x \in F(\beta, r, m, M)$, $Arrondi(x) = x$.
- Si $x \notin F(\beta, r, m, M)$:
  - $Arrondi(x) =$ nombre de $F(\beta, r, m, M)$ le plus proche de x.
  - $Arrondi(x) =$ nombre de $F(\beta, r, m, M)$ immédiatement supérieur.
  - $Arrondi(x) =$ nombre de $F(\beta, r, m, M)$ immédiatement inférieur.
  - $Arrondi(x) =$ nombre de $F(\beta, r, m, M)$ le plus proche en direction de zéro.

ULP

Unit in the last place.
Taille de l'intervalle séparant chaque nombre du nombre représentable le plus proche (dans la direction opposée de celle de zéro.demo

Quelques propriétés des nombres à virgule flottante

L'annulation catastrophique. (Catastrophic cancellation)

C'est le diable !..

Quelques propriétés des nombres à virgule flottante

L'annulation catastrophique. (Catastrophic cancellation)

C'est le diable !..

Perte de précision qui résulte de la soustraction de deux nombres voisins. démo

L'annulation catastrophique. (Catastrophic cancellation)

C'est le diable !..

Perte de précision qui résulte de la soustraction de deux nombres voisins. démo

Êtes vous sûrs de savoir calculer les racines d'un trinôme du second degré ? démo

L'annulation catastrophique. (Catastrophic cancellation)

C'est le diable !..

Perte de précision qui résulte de la soustraction de deux nombres voisins. démo

Êtes vous sûrs de savoir calculer les racines d'un trinôme du second degré ? démo

Exercice :
Envisager tous les cas possibles pour le choix de `a`, `b` et `c` : l'écriture d'un programme numériquement robuste pour le calcul des racines d'un trinôme du second degré est loin d'être simple.

Quelques propriétés des nombres à virgule flottante

Les ensembles de nombres flottants ne sont pas des groupes pour l'addition.

Groupe additif :

1. élément neutre : $a + 0 = a$
2. élément symétrique : pour tout a il existe $-a$ tel que $a + (-a) = 0$.
3. **Associativité :** $a + (b + c) = (a + b) + c$.

Quelques propriétés des nombres à virgule flottante

Les ensembles de nombres flottants ne sont pas des groupes pour l'addition.

Groupe additif :

1. élément neutre : $a + 0 = a$
2. élément symétrique : pour tout a il existe $-a$ tel que $a + (-a) = 0$.
3. **Associativité :** $a + (b + c) = (a + b) + c$.

Dans les ensembles de nombres flottants, l'addition n'est pas associative !

demo

Quelques propriétés des nombres à virgule flottante

Les ensembles de nombres flottants ne sont pas des groupes pour l'addition.

Groupe additif :

1. élément neutre : $a + 0 = a$
2. élément symétrique : pour tout a il existe $-a$ tel que $a + (-a) = 0$.
3. **Associativité :** $a + (b + c) = (a + b) + c$.

Dans les ensembles de nombres flottants, l'addition n'est pas associative !

demo

Conséquence : le même programme, compilé par deux compilateurs (optimiseurs) différents ne donne pas toujours *exactement* le même résultat.

Quelques propriétés des nombres à virgule flottante

Calcul de récurrences.

$$u_{n+1} = 4u_n - 1$$

avec $u_0 = 1/3$.

demo

Quelques propriétés des nombres à virgule flottante

Calcul de récurrences.

$$u_{n+1} = 4u_n - 1$$

avec $u_0 = 1/3$.

demo

Réaction du calculateur malin : *comportement normal : puisqu'on ne calcule pas exactement les u_i, les erreurs sont multipliées par 4 à chaque itération.*

Quelques propriétés des nombres à virgule flottante

Calcul de récurrences.

$$u_{n+1} = 4u_n - 1$$

avec $u_0 = 1/3$.

demo

Réaction du calculateur malin : *comportement normal : puisqu'on ne calcule pas exactement les u_i, les erreurs sont multipliées par 4 à chaque itération.*

On recommence avec :

$$u_{n+1} = 3u_n - 1$$

avec $u_0 = 1/2$. demo

Quelques propriétés des nombres à virgule flottante

Mais qu'est ce que cette histoire ? ? ?

Quelques propriétés des nombres à virgule flottante

Mais qu'est ce que cette histoire ? ? ?

Explication :

1. $u_{n+1} = 4u_n - 1$ avec $u_0 = 1/3$:

$$\frac{1}{3} = \frac{1}{4}\sum_{i=0}^{\infty}\frac{1}{4^i} = \frac{1}{4}\sum_{i=0}^{\infty}\frac{1}{2^{2i}},$$

et donc $u_0 = 1/3$ ne peut pas être représenté exactement en flottant. L'erreur initiale est amplifiée...

Quelques propriétés des nombres à virgule flottante

Mais qu'est ce que cette histoire ? ? ?

Explication :

1. $u_{n+1} = 4u_n - 1$ avec $u_0 = 1/3$:

$$\frac{1}{3} = \frac{1}{4}\sum_{i=0}^{\infty}\frac{1}{4^i} = \frac{1}{4}\sum_{i=0}^{\infty}\frac{1}{2^{2i}},$$

   et donc $u_0 = 1/3$ ne peut pas être représenté exactement en flottant. L'erreur initiale est amplifiée...

2. $u_{n+1} = 3u_n - 1$ avec $u_0 = 1/2$.

Quelques propriétés des nombres à virgule flottante

Mais qu'est ce que cette histoire ? ? ?

Explication :

1. $u_{n+1} = 4u_n - 1$ avec $u_0 = 1/3$:

$$\frac{1}{3} = \frac{1}{4}\sum_{i=0}^{\infty}\frac{1}{4^i} = \frac{1}{4}\sum_{i=0}^{\infty}\frac{1}{2^{2i}},$$

   et donc $u_0 = 1/3$ ne peut pas être représenté exactement en flottant. L'erreur initiale est amplifiée...

2. $u_{n+1} = 3u_n - 1$ avec $u_0 = 1/2$.
   En base 2 :
   - $1/2$ s'écrit 0.1,
   - $3/2$ s'écrit 1.1

   donc le calcul est exact .

Finalement, que peut-on calculer avec les nombres à virgule flottante ?

On ne peut effectuer que des calculs pour lesquels la solution dépend *gentiment* des données (*problèmes bien posés*).

- Les nombres à virgule flottante doivent être regardés avec méfiance, mais ils n'ont pas empêché le développement du calcul et de ses applications : ce ne sont pas les erreurs d'arrondi qui limitent la validité de la prévision météorologique, pour ne citer que cet exemple.

Finalement, que peut-on calculer avec les nombres à virgule flottante ?

On ne peut effectuer que des calculs pour lesquels la solution dépend *gentiment* des données (*problèmes bien posés*).

- Les nombres à virgule flottante doivent être regardés avec méfiance, mais ils n'ont pas empêché le développement du calcul et de ses applications : ce ne sont pas les erreurs d'arrondi qui limitent la validité de la prévision météorologique, pour ne citer que cet exemple.
- La stabilité des algorithmes vis-à-vis des petites perturbations doit être étudiée.

Nombres à virgule flottante : un exemple de problème mal conditionné

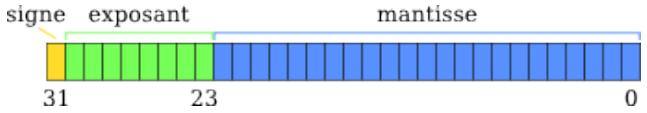
Résoudre de système linéaire $Ax = B$ en prenant pour A une matrice de taille n, avec $A_{i,j} = 1/(i + j)$.

- La solution est dans $\mathbb{Q}$: calcul exact si on a les moyens de calculer dans $\mathbb{Q}$ (Exemple : bibliothèque Linbox)
- On peut alors comparer la solution dans $\mathbb{Q}$ à la solution « flottante ».

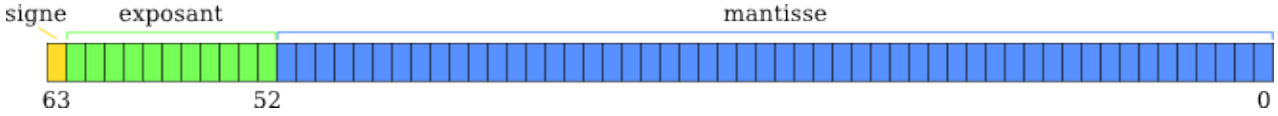
démo.

Nombres à virgule flottante : retour sur la norme

32 bits

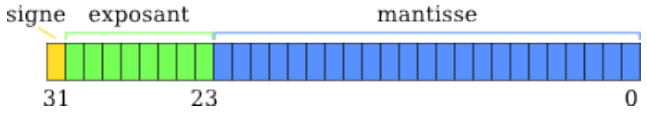


64 bits

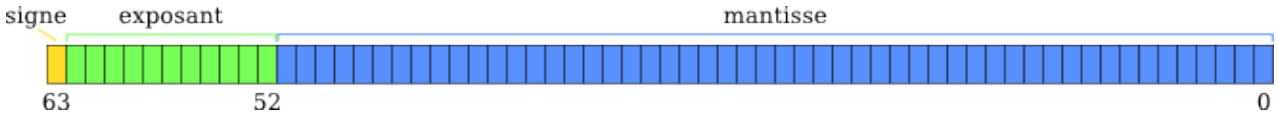


Nombres à virgule flottante : retour sur la norme

32 bits



64 bits



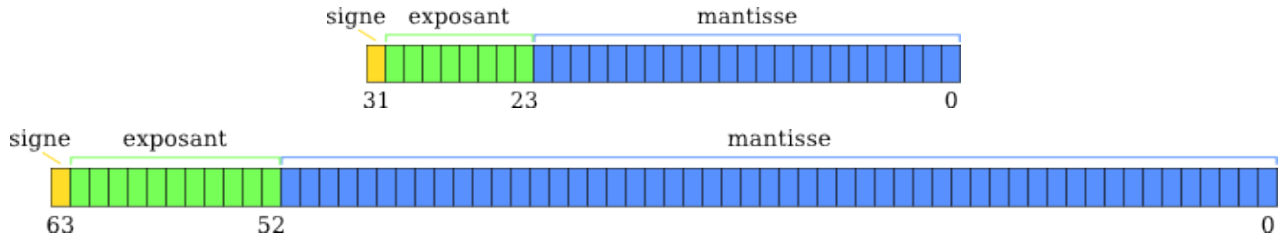
Les valeurs nombres ayant tous les bits de leur exposant à 1 sont réservés pour des valeurs exceptionnelles :

- `NaN` mantisse différente de 0. Pour indiquer des résultats comme 0/0.
- `Infty` mantisse égale à 0 (le signe compte ($\pm\infty$).

Nombres à virgule flottante : retour sur la norme

32 bits

64 bits



Les valeurs nombres ayant tous les bits de leur exposant à 1 sont réservés pour des valeurs exceptionnelles :

- `NaN` mantisse différente de 0. Pour indiquer des résultats comme 0/0.
- `Infty` mantisse égale à 0 (le signe compte ($\pm\infty$).
- 0 est codé en mettant 0 dans la mantisse et l'exposant. Le signe compte ($+0$ et -0).
- les nombres dénormalisés : l'exposant est mis à 0. Permettent de coder des valeurs en dehors de l'intervalle « normalisé ».

Nombres à virgule flottante : références bibliographiques

- *What every scientist should know about floating-point arithmetic*. David Goldberg.
  Texte disponible à de nombreux endroits, entre autres à : `http://perso.ens-lyon.fr/jean-michel.muller/goldberg.pdf`.
- *Handbook of Floating-Point Arithmetic*, Muller et collaborateurs (ENS Lyon).
- *Calcul mathématique avec Sage*. Casamayou, Alexandre et Connan, Guillaume et Dumont, Thierry et Fousse, Laurent et Maltey, François et Meulien, Matthias et Mezzarobba, Marc et Pernet, Clément et Thiéry, Nicolas et Zimmermann, Paul.