Calcul en virgule flottante

Lionel Lacassagne

LIP6
Université Pierre et Marie Curie (Paris 6)

Plan

- Présentation des nombres flottants

- Norme IEEE 754
- problèmes associés à la norme et solutions

Les nombres flottants

- Les nombres flottants (à virgule flottante) sont tellement différents des nombres réels des mathématiques, qu'on a finit par leur donner une lettre pour les différencier :
  - $\mathbb{R}$: l'ensemble des réels,
  - $\mathbb{F}$: l'ensemble des flottants.
- Ils sont moins nombreux (cf Chuck Norris) et n'ont pas les mêmes propriétés (associativité).

  Si ces *petites* différences n'ont pas d'importance dans la très grande majorité des cas, cela peut être catastrophique dans d'autres (simulation et systèmes embarqués).
- Le but de cette présentation est de faire prendre conscience de ces différences et d'apprendre à les gérer lorsque cela est nécessaire.

La norme IEEE 754-1985

Deux formats en simple et double précision (héritage du Fortran)

nom commun	type C	bits	exposant	mantisse
simple précision	`float`	32	8	23 (+1)
double précision	`double`	64	11	52 (+1)

Table: formats IEEE 754-1985

Quatre modes d'arrondi :

- arrondi au plus proche (mode par défaut)
- arrondi vers zéro = troncature (conversion `float -> int`)
- arrondi vers $+\infty$ = arrondi par excès (fonction `ceil`)
- arrondi vers $-\infty$ = arrondi par défaut (fonction `floor`)

Pour en savoir plus :

- `https://en.wikipedia.org/wiki/IEEE_754-1985`
- David Goldberg, "What Every Computer Scientist Should Know about Floating-Point Arithmetic", ACM Computing Surveys, vol. 23, no 1, mars 1991.

La norme IEEE 754-2008

Ajout de deux formats supplémentaires en base 2 et normalisation de 3 formats en base 10.

nom commun	nom normalisé	type C99	bits	exposant	mantisse
demi précision	binary16	`half ?`	16	8	10 (+1)
quadruple précision	binary128	`long double`	128	11	112 (+1)

Table: nouveaux formats IEEE 754-2008

- binary128 : pour la simulation et les super-calculateurs, lorsque beaucoup d'itérations avec très petites variations, pour limiter la propagation d'erreur.
- binary16 : partout où binary32 n'est pas nécessaire : codage couleur cinéma, calcul 3D et texture sur GPU, calcul plus rapide en GP-GPU, sur FPGA et systèmes embarqués, en fonction des domaines applicatifs (traitement du signal, audio, traitement d'image). Les nouveaux GPU intègrent des FPU 16 bits plus rapides que les FPU 32 bits.

Pour en savoir plus :

- `https://en.wikipedia.org/wiki/IEEE_floating_point`

Vocabulaire

En français il n'y a qu'un seul mot pour parler de la précision de codage des nombres et de la précision des calculs : *précision*.

En anglais, il y en a deux :

- *precison* : le nombre de bits utilisés pour coder un nombre (32/64 ou plus précisement 23/52)
- *accuracy* : le nombre de bits corrects d'un calcul

Tout le problème est donc de relier la *precison* initiale à un objectif d'*accuracy*, aux options (d'optimisation bas niveau) du compilateur et aux transformations (algorithmiques de haut niveau) du code.

Quelques problèmes de représentation #1

Certains nombres* de $\mathbb{R}$ n'existent pas (ne sont pas codables) dans $\mathbb{F}$.

En C, la fraction $1/10$ n'est pas *exactement codable*

```
float f = 1.0f / 10.0f;
```

décimal	`f = 0.100000001490116119384765625`
binaire	`f = 00111101110011001100110011001101`
hexadécimal	`f = 0x3dcccccd`
notation mixte	$1.600000023841858 \times 2^{-4}$

Et donc :

- `10 * f` ne vaut pas forcément 1 (ici si grâce au calcul par arrondi)
- Et encore moins `f+f+f+f+f+f+f+f+f+f`
  qui vaut $(1.000000119209289550 78)_{10}$ ou $(0x3f800001)_{16}$
- $\Rightarrow$ Plus on fait d'opérations (9 ADD vs 1 MUL) et plus on propage l'erreur.

(*) : beaucoup de nombres, voire même une infinité de nombres puisque $card(\mathbb{R}) - card(\mathbb{F}) = \infty - 2^{32} = ...\infty$.

Conversion en ligne :
`http://www.h-schmidt.net/FloatConverter/IEEE754.html`

Quelques problèmes de représentation #2

Les options d'optimisation du compilateur (-O2, -O3) sont sources d'erreur :
Le Code

```
f = 1.0f / 10;
for(i = 0; i < n; i++)
  T[i] = i * f;
```

sera remplacé par

```
f = 1.0f / 10;
s = 0.0f;
for(i = 0; i < n; i++)
  T[i] = x;
  x += f;
```

- Car l'addition est plus rapide que la multiplication.
- L'optimisation appliquée s'appelle *strenght reduction*

Propriétés mathématiques perdues : associativité et distributivité

- l'associativité : $(a \star b) \star c = a \star (b \star c)$
  - associativité de l'addition : $(10^{-16} + 1) - 1 = 0 \quad \neq \quad 10^{-16} = 10^{-16} + (1 - 1)$
  - associativité de la multiplication : $10^{-200} \times (10^{200} \times 10^{200}) = \infty \quad \neq \quad 10^{200} = (10^{-200} \times 10^{200}) \times 10^{200}$

- la distributivité (de la multiplication par rapport à l'addition) :
  - distributivité à gauche : $a \times (b + c) = (a \times b) + (a \times c)$
  - distributivité à droite : $(a + b) \times c = (a \times c) + (b \times c)$

Or les compilateurs optimisants peuvent réorganiser (et simplifier) les expressions mathématiques pour que le processeur les calcule plus vite (à partir de -O1)...

Phénomène de cancellation

Lors de la soustraction de deux nombres proches

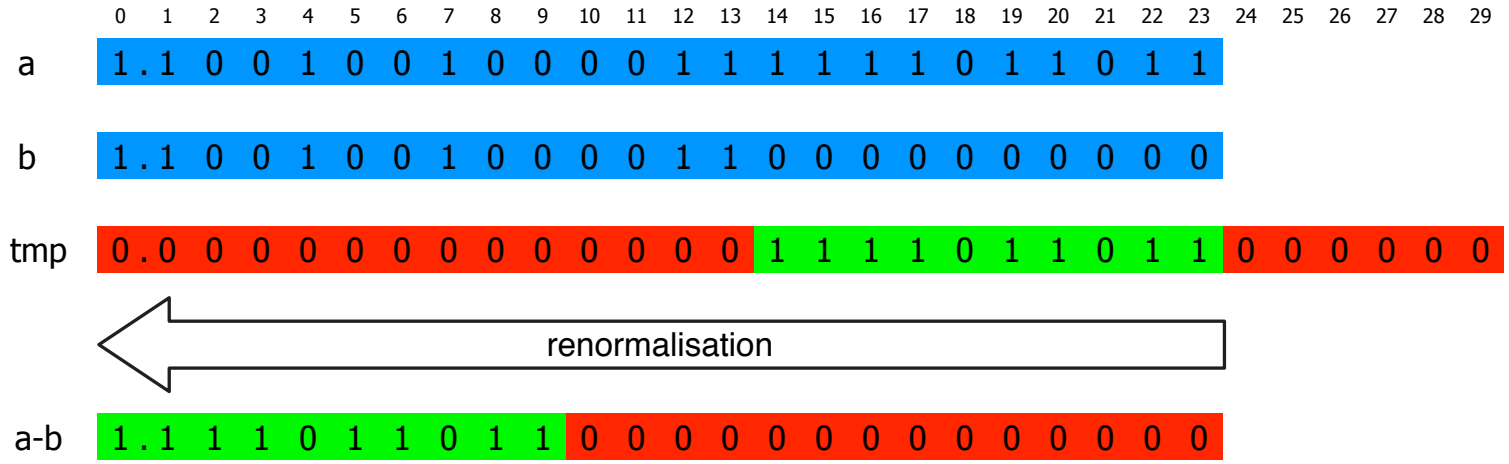

Figure: d'après *arithmétique réelle* - V. Ménissier-Morain LIP6

- perte de chiffres/digits significatifs à gauche (partie rouge),
- après normalisation, il reste peu de chiffres/digits significatif (partie verte),
- ajout de zéros à droite.

Il y a cancellation catastrophique lors qu'il n'y a plus de chiffre significatif.

Phénomène d'absorption

Lors de l'addition de deux nombres très différents

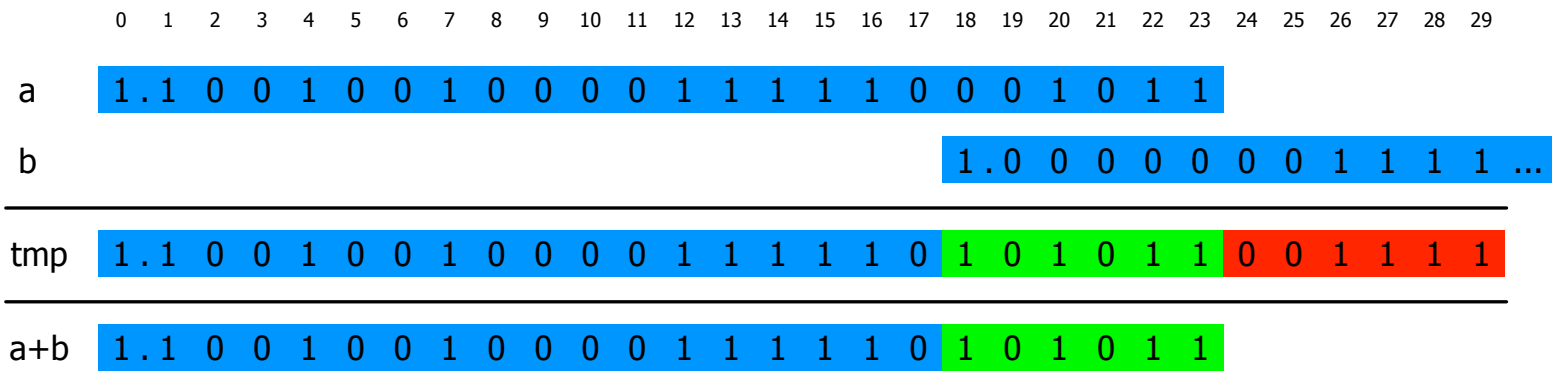

Figure: d'après *arithmétique réelle* - V. Ménissier-Morain LIP6

- la partie où l'addition se fait est petite (vert),
- un grand nombre de chiffres/digits ne sont pas *pris en compte* (rouge) à cause du format en précision finie.

Il y a absoprtion catastrophique lorsqu'aucun chiffre/digit n'est pris en compte : $a + \varepsilon = a$ (avec $\varepsilon \neq 0$).

Exemple du phénomène d'absorption

sommation dans le *mauvais* sens de $\sum_{i=1}^{N} \frac{1}{i}$
(d'après V. Ménissier-Morain LIP6)

N	10^5	10^6	10^7	10^8
exacte	1.209015×10^1	1.439273×10^1	1.439273×10^1	1.899790×10^1
$1 \rightarrow N$	1.209085×10^1	1.435736×10^1	1.540368×10^1	1.540368×10^1
$N \rightarrow 1$	1.209015×10^1	1.439265×10^1	1.668603×10^1	1.880792×10^1

- Commencer par accumuler les plus petits nombres ne suffit pas,
- il faut faire des sommes par paquets, voire faire des sommes avec tris.

Limitation du phénomène d'absorption

Somme d'une matrice :

```
s = 0.f;
for(i = 0; i < n; i++) {
  for(j = 0; j < n; j++) {
    s += T[i][j]; // accumulation de valeurs de magnitude 1 et n^2
  }
}
```

Somme un peu plus précise :

```
s = 0.f;
for(i = 0; i < n; i++) {
  sx = 0.f;
  for(j = 0; j < n; j++) {
    sx += T[i][j]; // accumulation de valeurs de magnitude 1 et n
  }
  s += sx; // accumulation de valeurs de magnitude 1 et n
}
```

Limitation maximale du phénomène d'absorption

Algorithme d'additions d'une liste de nombres avec tris :

- trier l'ensemble des nombres,
- prendre les deux plus petits, les additionner et les insérer dans la liste,
- recommencer jusqu'à ce que la liste ne contienne plus qu'un nombre : la somme.

Exemple (avec des entiers pour faire plus simple) :

$\{1, 2, 3, 4, 5\} \rightarrow 1 + 2 = 3$ à insérer en début
$\{3, 3, 4, 5\} \rightarrow 3 + 3 = 6$ à insérer à la fin
$\{4, 5, 6\} \rightarrow 4 + 5 = 9$ à insérer à la fin
$\{6, 9\} \rightarrow 6 + 9 = 15$
$\{15\}$

Les fonctions mathématiques

Elles sont calculées avec :

- des tables,
- des polynômes,
- des fonctions itératives.

La table donne une valeur initiale (pour un intervalle) qui est raffinée par le polynôme où la fonction itérative. Il existe des logiciels pour fabriquer des tables.

Exemple de fonction tabulée : fonctions trigonométriques

- fonctions tabulées sur $[0, 360[$:
- $C[a] \leftarrow cos(a), S[a] \leftarrow sin(a), T[a] \leftarrow tan(a)$ pour $a \in [0, 359]$
- pour les angles plus grands, on se ramène à l'angle principal par un modulo (coûte une division)
- pour avoir une table plus petite, on utilise les symétries, pour se ramener à $[0, 45]$ (coûte des tests)
- si angle en radian, ajouter une conversion (coûte une multiplication)
- fonctionne avec des valeurs non-entières, renvoie la valeur au plus proche : $a \leftarrow round(\alpha)$
- pour avoir une précision de 0.1 degré : $S[a] \leftarrow sin(a/10)$

Fabriquer une table est complexe. Au delà des cas simples, faire appel à un expert.

Fonction mathématique : sinus

Exemple (N. Revol et V Ménissier-Morain) $sin(10^{22})$ en radian

système	$sin(10^{22})$
valeur exacte	-0.852200849767
HP 48 GX	-0.852200849767
Sharp EL5806	-0.090748172
Casin FX180P, FX8100	Error
VAX VMS	-0.852200849
DEC Station 3100	NaN
HP 700	0.0
IBM 3090/600S-VF AIX 370	0.0
IBM RS/6000 AIX 3005	-0.852200849
Silicon Graphics Indy	0.87402806
Sparc	-0.852200849
PC GCC 2.96	0.462613
PC Borland Turbo C 2.0	4.67734e-240
Matlab 4.2c.1 Sparc	-0.8522
Matlab 4.2c.1 MacIntosh	0.8740

Fonction mathématique sur calculatrices HP

MPFR 200 bits :
-0.85220084976718880177270589375302936826176215041004365625650964

système	$sin(10^{22})$
série *pioneer*	
HP15C (1982)	0.793105679
série *voyager*	
HP20S (1986)	-0.852200850
HP42S (1988)	-0.85220084976
HP32SII (1991)	-0.852200850
série graphique RPL	
HP28S (1988)	-0.852200849762
HP48SX (1990)	-0.852200849762
HP50G (2006)	-0.852200849762
HP Prime (2013)	-0.852200849762

Fonctions mathématique sur calculatrice TI

MPFR 200 bits :
-0.85220084976718880177270589375302936826176215041004365625650964

système	$sin(10^{22})$
Texas Instrument	
TI 30 Galaxy (1982)	Error
TI 92 II (1995)	arg too big for accurate reduction
TI nSpire CAS V1 (2007)	Error
Casio	
appel à contribution	

Fonction mathématique sur téléphones et tablettes

système	$sin(10^{22})$
valeur exacte MPFR	-0.85220084976718880
iOS	
Free Calc 1	-0.68
Calculator X	-0.852200849767189
Big RPN	-0.8522008497671888
Free 42S	-0.85220084977
HP 28S	-0.852200850
RPN28x	-0.8522001151812284
Halcyon Calc lite (28S)	-0.852200849767
i48 / GraphiX48 (ROM HP)	-0.852200849762
MathPACKLite (free)	-0.852200849767189
Calc Apple iOS	0
Android	
Calc Android v ? ? ?	0
BlackBerry 10	
Calc RPN BlackBerry 10	-0.03
Calc BlackBerry 10	-0.984807753012

Fonction mathématique sur ordinateurs

système	$sin(10^{22})$
valeur exacte MPFR	-0.85220084976718880177270589375303029

widgets	
WCalc 2.4 OSX 10.6.8	-0.852200849
Calc PC Laurent	-0.984807753
Calc Apple OSX 10.10.5	0
Calc Red-Hat 6.5	-0.85220085
Calc Red-Hat 6.5	-0.8522008498
Calc Windows 7	-0,852200849767188801772705893753303

Tableurs	
Libre Office 4.0.4.2	#Valeur
Excel 2008 Mac	-0.852200849767189
Number 2.1 (iWorks' 09)	-0.852200849767189

Système de calculs	
Matlab	
Octave 4.0.0 (GNU)	-0.852200849767189
Scilab 5.5.2 on cloud (INRIA)	0.8740281

Compilateurs	
GCC 4.2.1	-0.852200849767189
ICC 12.1	-0.852200849767189

Fonctions mathématiques sur le web

système	$sin(10^{22})$
valeur exacte MPFR	-0.85220084976718880
`http://www.calculator.net`	0.8488608486
`http://www.eeweb.com/toolbox/calculator`	0

EEWeb = Electrical Engineering Community

Logiciels et équipes

Logiciels :

- MPFR : calcul en multiprécision `http://www.mpfr.org`
- MPFI : calcul par intervalles avec MPFR `http://mpfi.gforge.inria.fr`
- FLIP : Floating-point Library for Integer Processors `http://flip.gforge.inria.fr`
- Flopoco : Floating Point Cores for FPGAs (but not only) `http://flopoco.gforge.inria.fr`
- Sollya : génération de fonctions mathématiques `http://sollya.gforge.inria.fr`

Equipes :

- Pequan (LIP6/UPMC) `http://www-pequan.lip6.fr`
- Aric (ENS Lyon) : `www.ens-lyon.fr/LIP/AriC`
- DALI (UPVD/LIRMM) `http://webdali.univ-perp.fr`