Manipulations des nombres à virgule flottante

Informatique pour tou(te)s - Semaines 3 & 4

Guillaume CONNAN

Lycée Clemenceau - MP/MP*

Dernière mise à jour : 11 janvier 2015 à 12:58

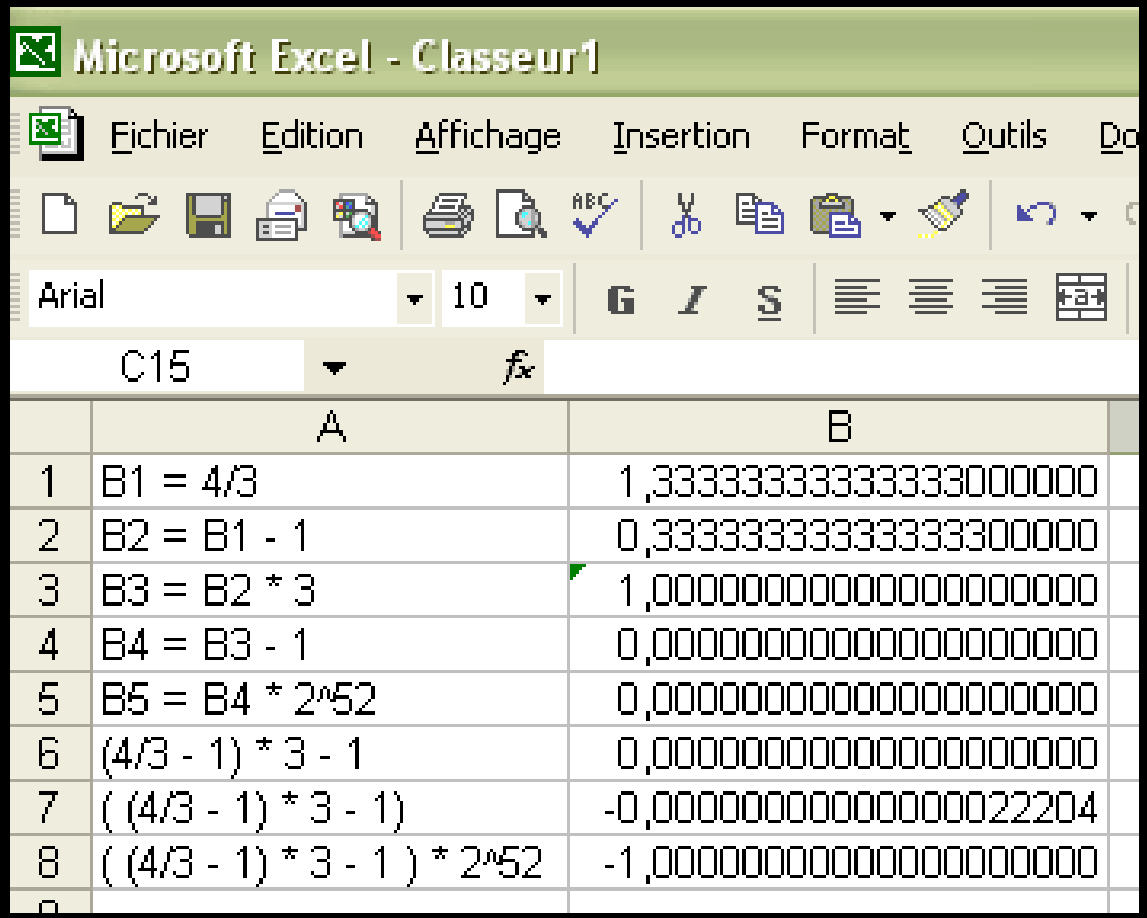
Sommaire

Sommaire


SCUD


OUPS !


LE 20 JANVIER 1992 LE MATELOT CLOTAIRE LEANDRI SE COINÇA LE PIED DANS LA CATAPULTE DU PORTE-AVIONS CLÉMENCEAU. ON VIENT D'ABANDONNER LES RECHERCHES.

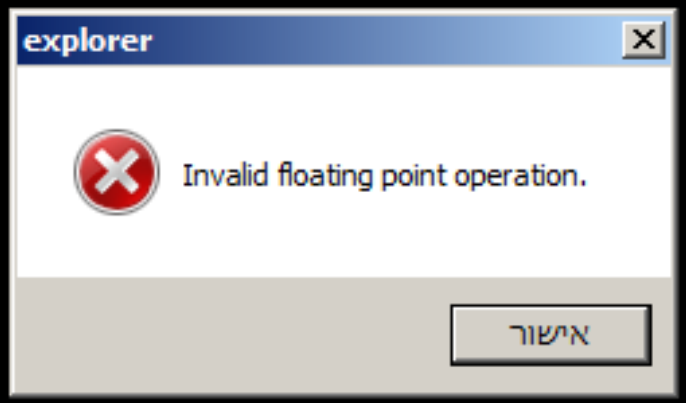

Microsoft Excel - Classeur1

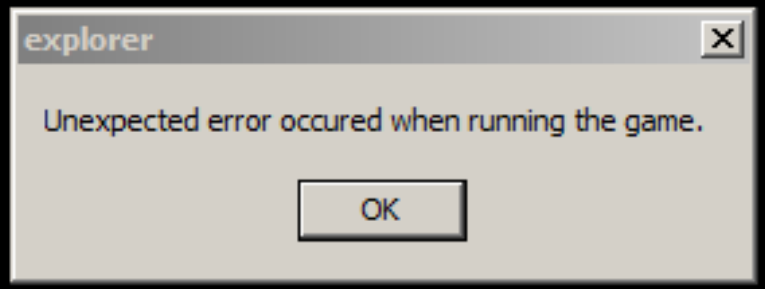
Fichier Edition Affichage Insertion Format Outils Do

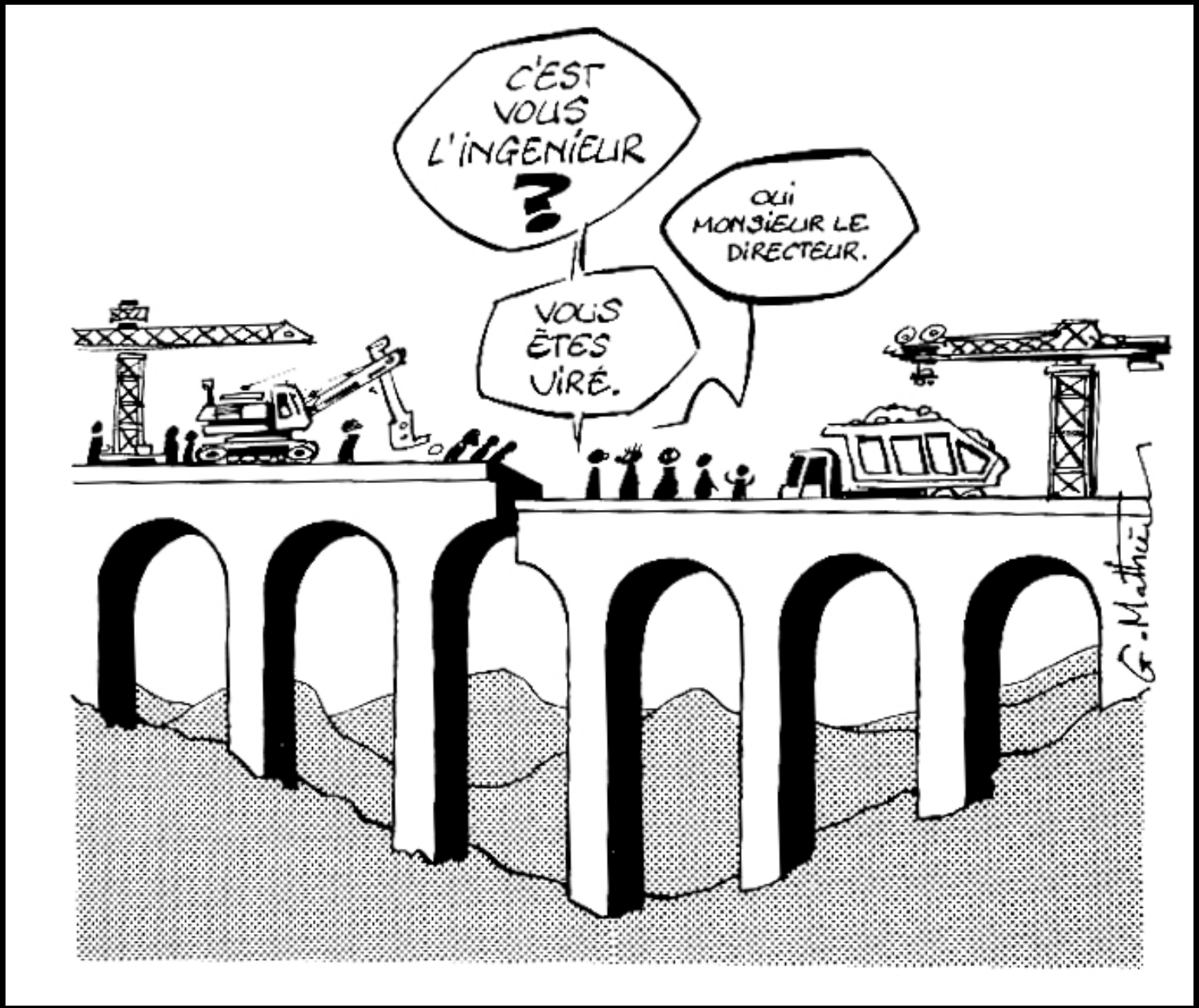
Arial 10

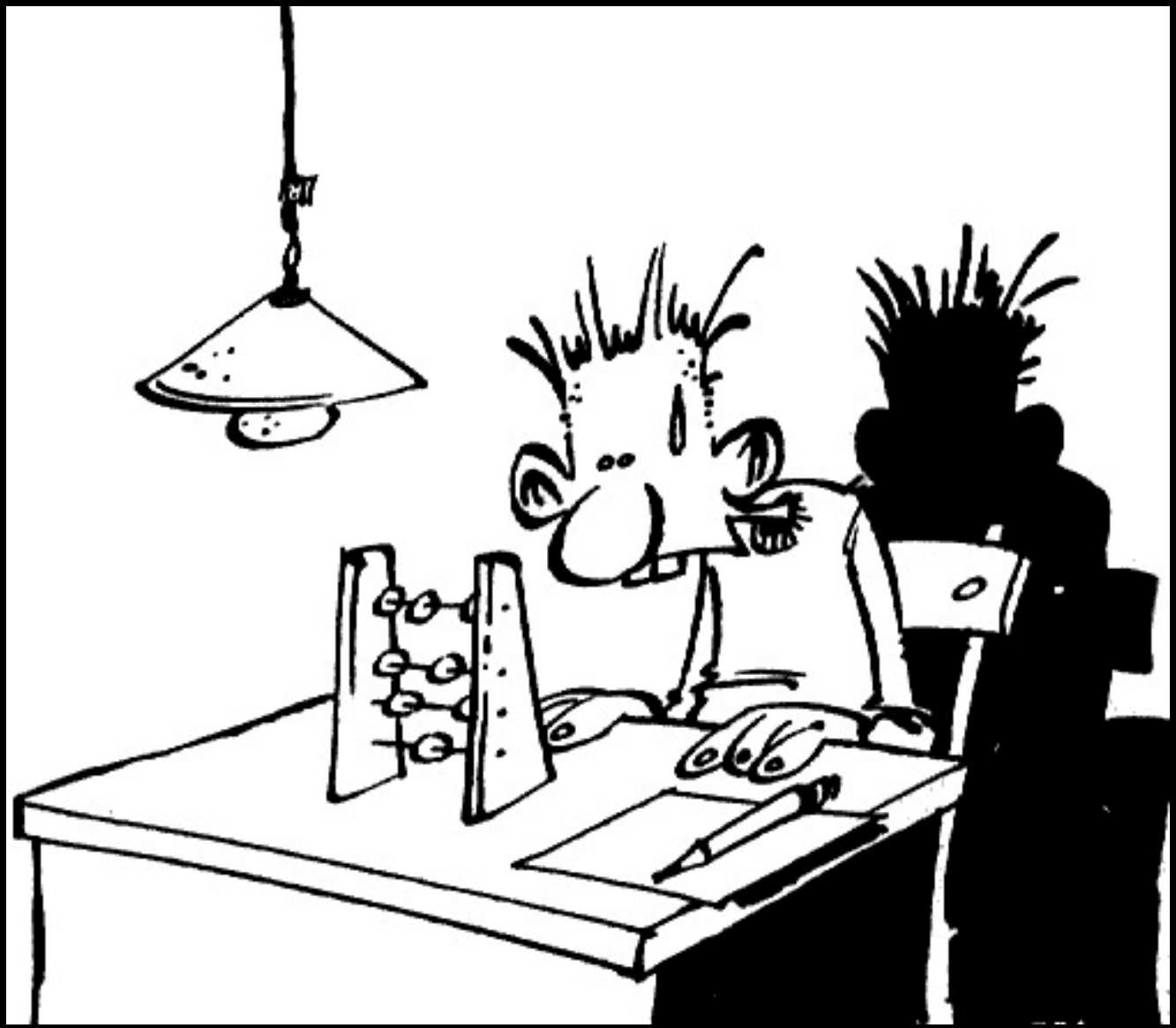
C15

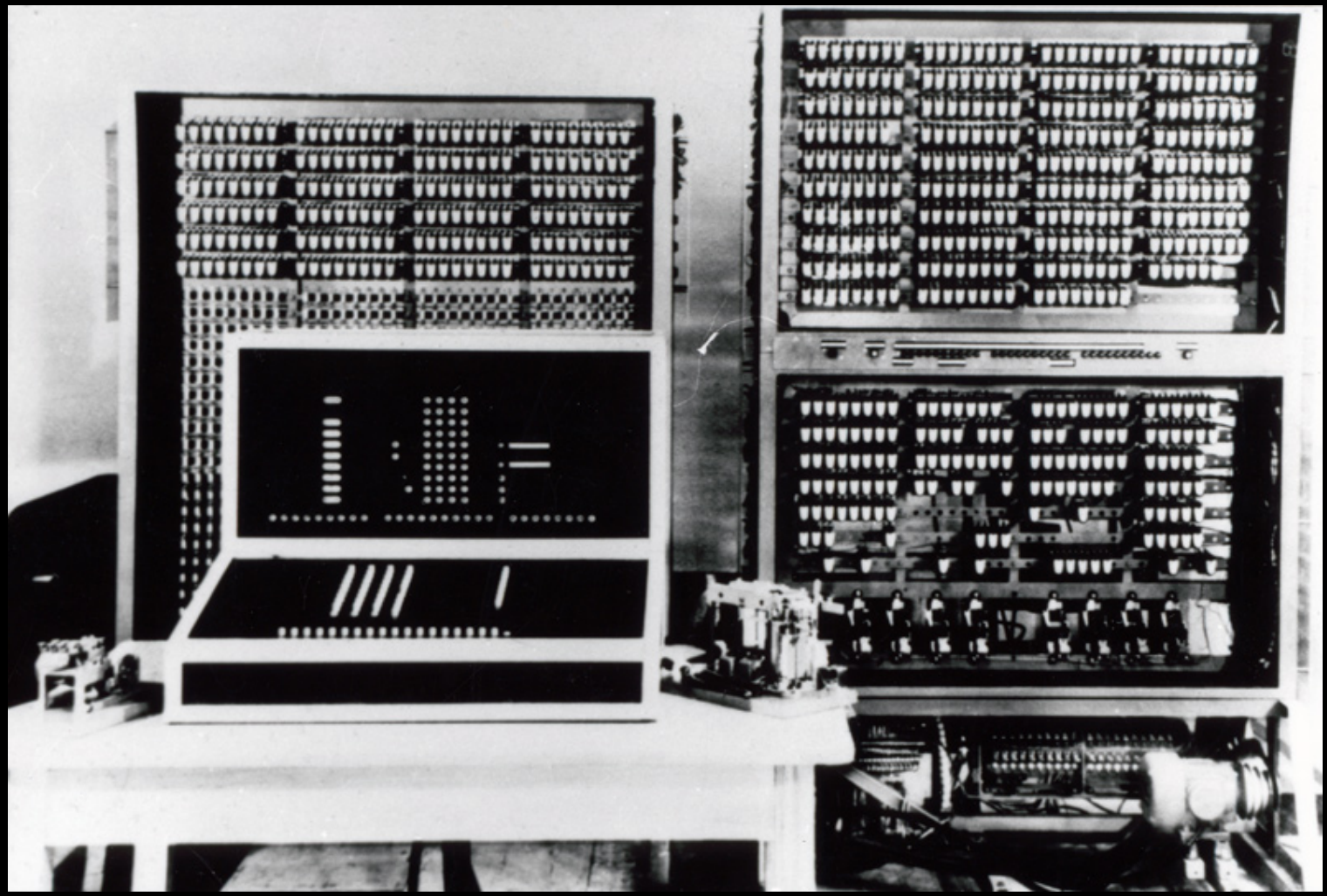
	A	B
1	B1 = 4/3	1,33333333333333000000
2	B2 = B1 - 1	0,33333333333333300000
3	B3 = B2 * 3	1,00000000000000000000
4	B4 = B3 - 1	0,00000000000000000000
5	B5 = B4 * 2^52	0,00000000000000000000
6	(4/3 - 1) * 3 - 1	0,00000000000000000000
7	((4/3 - 1) * 3 - 1)	-0,00000000000000022204
8	((4/3 - 1) * 3 - 1) * 2^52	-1,00000000000000000000


explorer
Invalid floating point operation.
אישור


explorer
Unexpected error occured when running the game.
OK


C'EST VOUS L'INGENIEUR ?
OUI MONSIEUR LE DIRECTEUR.
VOUS ÊTES VIRÉ.
G. Mathieu





- char
- int

- `char`
- `int` $2147483647 + 1 = -2147483648\ldots$

- `char`
- `int` $2147483647 + 1 = -2147483648\ldots$
- `short`

- `char`
- `int` $2147483647 + 1 = -2147483648...$
- `short` `long`

- `char`
- `int` $2147483647 + 1 = -2147483648\ldots$
- `short` `long`

- `char`
- `int` $2147483647 + 1 = -2147483648\ldots$
- `short` `long`
- `float`

- `char`
- `int` $2147483647 + 1 = -2147483648...$
- `short long`
- `float`
- `single`
- `double`

- `char`
- `int` $2147483647 + 1 = -2147483648\ldots$
- `short long`
- `float`
- single
- `double`
- `long double`
- 3

- `char`
- `int` $2147483647 + 1 = -2147483648...$
- `short long`
- `float`
- single
- `double`
- `long double`
- 3

- `char`
- `int` $2147483647 + 1 = -2147483648...$
- `short long`
- `float`
- single
- `double`
- `long double`
- 3

- `char`
- `int` $2147483647 + 1 = -2147483648...$
- `short long`
- `float`
- single
- `double`
- `long double`
- 3 ...

- `char`
- `int` $2147483647 + 1 = -2147483648...$
- `short long`
- `float`
- single
- `double`
- `long double`
- 3 3.0 3.0000 3.0e0 0.3e1

- `char`
- `int` $2147483647 + 1 = -2147483648\ldots$
- `short long`
- `float`
- single
- `double`
- `long double`
- 3 3.0 3.0000 3.0e0 0.3e1

- `char`
- `int` $2147483647 + 1 = -2147483648\ldots$
- `short long`
- `float`
- single
- `double`
- `long double`
- 3 3.0 3.0000 3.0e0 0.3e1

- `char`
- `int` $2147483647 + 1 = -2147483648...$
- `short long`
- `float`
- single
- `double`
- `long double`
- 3 3.0 3.0000 3.0e0 0.3e1
- ...

- `char`
- `int` $2147483647 + 1 = -2147483648...$
- `short long`
- `float`
- single
- `double`
- `long double`
- 3 3.0 3.0000 3.0e0 0.3e1
- ...

- `char`
- `int` $2147483647 + 1 = -2147483648...$
- `short long`
- `float`
- single
- `double`
- `long double`
- 3 3.0 3.0000 3.0e0 0.3e1
- ...

```
In [1]: 3 * 0.1
.
.
.
```

```
In [1]: 3 * 0.1
Out[1]: 0.30000000000000004
.
.
```

```
In [1]: 3 * 0.1
Out[1]: 0.30000000000000004
In [2]: sum([0.1 for k in range(10000000)])
.
```

```
In [1]: 3 * 0.1
Out[1]: 0.30000000000000004
In [2]: sum([0.1 for k in range(10000000)])
Out[2]: 999999.9998389754
```

LES NOMBRES RÉELS N'EXISTENT PAS.

TOUT CE QUE VOUS AVEZ VU EN MATHS N'EST QU'ILLUSION !

LES NOMBRES RÉELS N'EXISTENT PAS.

TOUT CE QUE VOUS AVEZ VU EN MATHS N'EST QU'ILLUSION !

Sommaire

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 52 bits de mantisse (ou « significande » en français) m (de 0 à 51).

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $1 \leqslant m$.

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $1 \leqslant m$.

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

→ *binary32*

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.
- *binary32* (illegible)

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.

→ *toy7*

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 3)$

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.

$$v = (-1)^s \times m \times 2^E$$

- un bit de signe s qui vaut 0 si le nombre est positif, 1 sinon ;
- 11 bits d'exposant décalé e ;
- 53 bits de mantisse (ou « significande » en franglais) m, $\quad 1 \leqslant m$.

- *binary32* $(\#E, \#m) = (8, 24)$
- *binary64* $(\#E, \#m) = (11, 53)$.
- *toy7* $(\#E, \#m) = (3, 4)$.

Oups ! $1 + 11 + 53 = 65$...En fait, on représente un nombre de $\mathbb{V}_{64}$ sous **forme normale** :

$$v = (-1)^s \times 1, f \times 2^E$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant.

Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.

La première moitié va donc de 0 à $2^{\#E-1}-1$.

Elle correspond aux exposants réels de $E_{\min}$ jusqu'à 0.

La deuxième de $2^{\#E-1}$ à $2^{\#E}-1$.

Elle correspond aux exposants de 1 jusqu'à $E_{\max}$.

Il suffit donc de translater les exposants réels de $2^{\#E-1}-1$...

$$e_{\text{stocké}} = E_{\text{réel}} + 2^{\#E-1} - 1$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant.
Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1}-1$.
Elle correspond aux exposants réels de $E_{\min}$ jusqu'à 0.
La deuxième de $2^{\#E-1}$ à $2^{\#E}-1$.
Elle correspond aux exposants de 1 jusqu'à $E_{\max}$.
Il suffit donc de translater les exposants réels de $2^{\#E-1}-1$...

$$e_{\text{stocké}} = E_{\text{réel}} + 2^{\#E-1} - 1$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant.
Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1}-1$.
Elle correspond aux exposants réels de $E_{\min}$ jusqu'à 0.
La deuxième de $2^{\#E-1}$ à $2^{\#E}-1$.
Elle correspond aux exposants de 1 jusqu'à $E_{\max}$.
Il suffit donc de translater les exposants réels de $2^{\#E-1}-1$...

$$e_{\text{stocké}} = E_{\text{réel}} + 2^{\#E-1} - 1$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant.
Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1}-1$.
Elle correspond aux exposants réels de E_{min} jusqu'à 0.
La deuxième de $2^{\#E-1}$ à $2^{\#E}-1$.
Elle correspond aux exposants de 1 jusqu'à E_{max}.
Il suffit donc de translater les exposants réels de $2^{\#E-1}-1$...

$$e_{\text{stocké}} = E_{\text{réel}} + 2^{\#E-1} - 1$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant.
Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1}-1$.
Elle correspond aux exposants réels de E_{min} jusqu'à 0.
La deuxième de $2^{\#E-1}$ à $2^{\#E}-1$.
Elle correspond aux exposants de 1 jusqu'à E_{max}.
Il suffit donc de translater les exposants réels de $2^{\#E-1}-1$...

$$e_{\text{stocké}} = E_{\text{réel}} + 2^{\#E-1} - 1$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant.
Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1}-1$.
Elle correspond aux exposants réels de E_{min} jusqu'à 0.
La deuxième de $2^{\#E-1}$ à $2^{\#E}-1$.
Elle correspond aux exposants de 1 jusqu'à E_{max}.
Il suffit donc de translater les exposants réels de $2^{\#E-1}-1$...

$$e_{\text{stocké}} = E_{\text{réel}} + 2^{\#E-1} - 1$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant.
Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1}-1$.
Elle correspond aux exposants réels de E_{min} jusqu'à 0.
La deuxième de $2^{\#E-1}$ à $2^{\#E}-1$.
Elle correspond aux exposants de 1 jusqu'à E_{max}.
Il suffit donc de translater les exposants réels de $2^{\#E-1}-1$...

$$e_{\text{stocké}} = E_{\text{réel}} + 2^{\#E-1} - 1$$

On peut également gagner de la place en ne stockant pas le signe de l'exposant.
Sur $\#E$ bits on peut coder $2^{\#E}$ nombres.
La première moitié va donc de 0 à $2^{\#E-1}-1$.
Elle correspond aux exposants réels de E_{min} jusqu'à 0.
La deuxième de $2^{\#E-1}$ à $2^{\#E}-1$.
Elle correspond aux exposants de 1 jusqu'à E_{max}.
Il suffit donc de translater les exposants réels de $2^{\#E-1}-1$...

$$e_{\text{stocké}} = E_{\text{réel}} + 2^{\#E-1} - 1$$

0,75 en toy7

$0,75_{10}$ en *toy7*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	1	0	1	0	0

0,75 en toy7

$0,75_{10}$ en *toy7*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	1	0	1	0	0

0,75 en toy7

$0,75_{10}$ en *toy7*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	1	0	1	0	0

0,75 en toy7

$0,75_{10}$ en *toy7*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	1	0	1	0	0

0,75 en toy7

$0,75_{10}$ en *toy7*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	1	0	1	0	0

0,75 en toy7

$0,75_{10}$ en *toy7*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	1	0	1	0	0

0,75 en toy7

$0,75_{10}$ en *toy7*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	1	0	1	0	0

0,75 en toy7

$0,75_{10}$ en *toy7*.

$$0,75_{10} = \frac{3_{10}}{2^2_{10}} = 11_2 \times 2^{-2} = 1,100_2 \times 2^{-1}$$

$$e - \left(2^{3-1} - 1\right) = E = -1 \leftrightarrow e = -1 + 3 = 2 = 10_2$$

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	1	0	1	0	0

0,75 en toy7 : méthode générale

Comment écrire en base 2 la partie fractionnaire d'un nombre en base 10 ?

$$0,75 = b_1 \times 2^{-1} + b_2 \times 2^{-2} + \cdots$$

$$2 \times 0,75 = b_1 + b_2 \times 2^{-1} + \cdots$$

On voit poindre un bel algo...

0,75 en toy7 : méthode générale

Comment écrire en base 2 la partie fractionnaire d'un nombre en base 10 ?

$$0,75 = b_1 \times 2^{-1} + b_2 \times 2^{-2} + \cdots$$

$$2 \times 0,75 = b_1 + b_2 \times 2^{-1} + \cdots$$

On voit poindre un bel algo...

0,75 en toy7 : méthode générale

Comment écrire en base 2 la partie fractionnaire d'un nombre en base 10 ?

$$0,75 = b_1 \times 2^{-1} + b_2 \times 2^{-2} + \cdots$$

$$2 \times 0,75 = b_1 + b_2 \times 2^{-1} + \cdots$$

On voit poindre un bel algo...

0,75 en toy7 : méthode générale

Comment écrire en base 2 la partie fractionnaire d'un nombre en base 10 ?

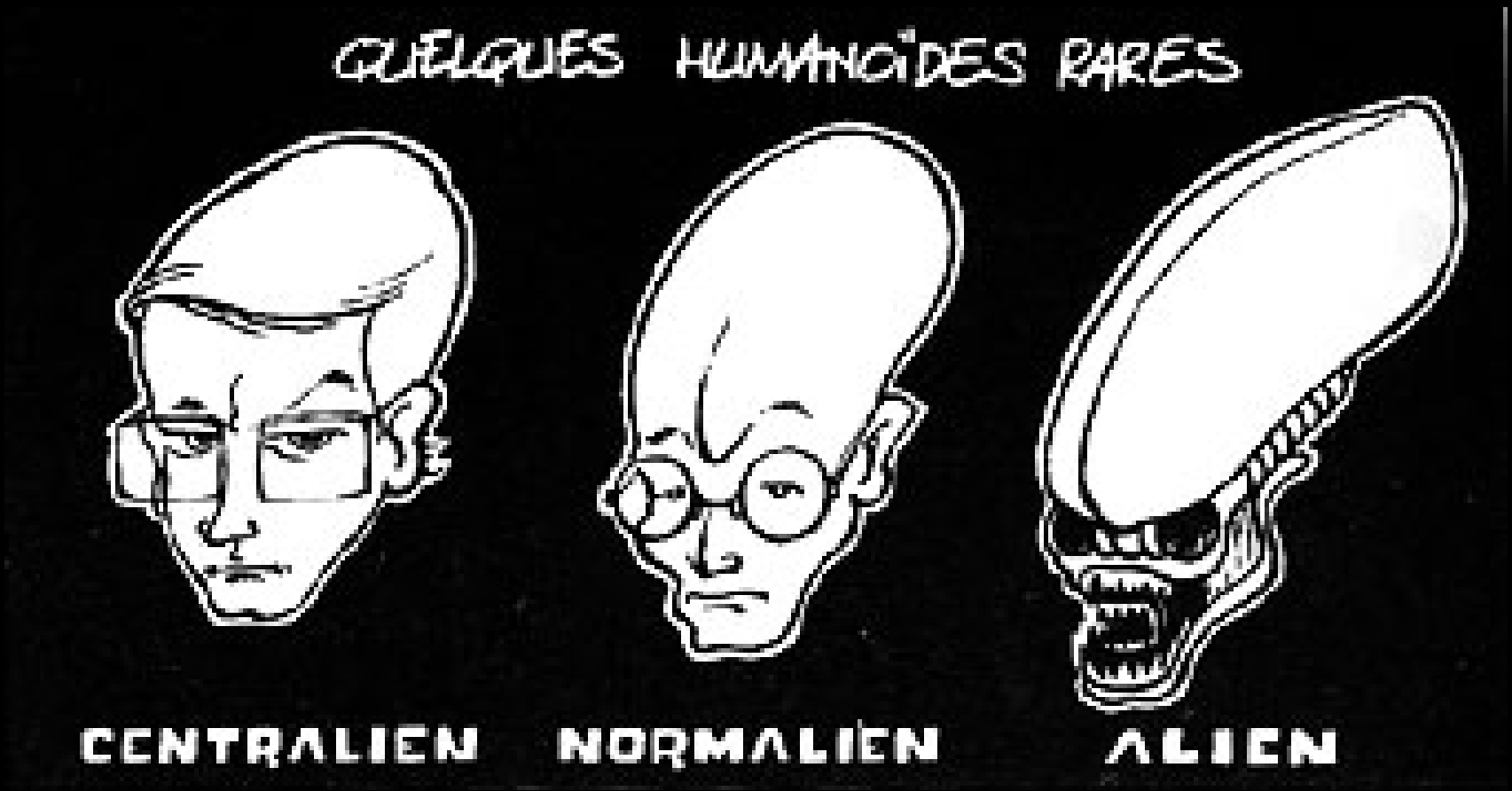
$$0,75 = b_1 \times 2^{-1} + b_2 \times 2^{-2} + \cdots$$

$$2 \times 0,75 = b_1 + b_2 \times 2^{-1} + \cdots$$

On voit poindre un bel algo...

Exercice 1

Pourquoi le Patriot a raté son Scud de 600m ?
Représentez 1/10 avec une mantisse de 24 bits et tronquez le résultat. Calculez l'erreur en seconde puis après 100 heures d'utilisation. Sachant qu'un Scud vole à $1676\,\mathrm{m\,s^{-1}}$*, quelle est environ l'erreur commise en mètres ?*
Et si on avait arrondi au plus proche ?


QUELQUES HUMANOÏDES RARES
CENTRALIEN
NORMALIEN
ALIEN

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = \left(2^3 - 1\right) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = \left(2^3 - 1\right) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

toy7, $\#E = 3$, $e_{\min} = 000$, $e_{\max} = 111 = 1000 - 1$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = \left(2^3 - 1\right) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

toy7, $\#E = 3$, $e_{\min} = 000$, $e_{\max} = 111 = 1000 - 1$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = \left(2^3 - 1\right) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

toy7, $\#E = 3$, $e_{\min} = 000$, $e_{\max} = 111 = 1000 - 1$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = \left(2^3 - 1\right) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = \left(2^3 - 1\right) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = \left(2^3 - 1\right) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

$$e_{\min} = 0$$

$$e_{\max} = 2^{\#E} - 1$$

$$e = E + 2^{(\#E)-1} - 1$$

$$toy7, \#E = 3,\ e_{\min} = 000,\ e_{\max} = 111 = 1000 - 1$$

Les valeurs extrêmes de E sont réservées pour des cas spéciaux

$$E_{\min} = \left(e_{\min} - 2^{(\#E)-1} + 1\right) + 1 = 0 - 2^{3-1} + 1 + 1 = -2$$

$$E_{\max} = \left(e_{\max} - 2^{(\#E)-1} + 1\right) - 1 = (2^3 - 1) - 2^{3-1} + 1 - 1 = 7 - 4 + 1 - 1 = 3$$

- zéro sous forme normale ?
- $0.\underbrace{000\ldots00}_{\#E}11 \times 2^0 = 1.1 \times 2^{-(\#E+1)}$?

Sous-normaux...

- zéro sous forme normale ?
- $0.\underbrace{000...00}_{\#E}11 \times 2^0 = 1.1 \times 2^{-(\#E+1)}$?

Sous-normaux....

- zéro sous forme normale ?
- $0.\underbrace{000\ldots00}_{\#E}11 \times 2^0 = 1.1 \times 2^{-(\#E+1)}$?

Sous-normaux....

- zéro sous forme normale ?
- $0.\underbrace{000\ldots00}_{\#E}11 \times 2^0 = 1.1 \times 2^{-(\#E+1)}$?

Sous-normaux....

- zéro sous forme normale ?
- $0.\underbrace{000\ldots00}_{\#E}11 \times 2^0 = 1.1 \times 2^{-(\#E+1)}$?

Sous-normaux....

Exercice 2

En utilisant vos carreaux, représentez les nombres à virgule flottante normaux en toy7 et rajoutez les sous-normaux.

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	0	0	0	0	0

s (1 bit)	e (3 bits)			f (3 bits)		
1	0	0	0	0	0	0

s (1 bit)	e (3 bits)			f (3 bits)		
0	0	0	0	0	0	0

s (1 bit)	e (3 bits)			f (3 bits)		
1	0	0	0	0	0	0

```
In [1]: x = 1e500
```

.

.

.

.

.

.

.

.

```
In [1]: x = 1e500

In [2]: 1+x
.
.
.
.
.
.
.
```

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
```

.

.

.

.

.

.

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
.
.
.
.
.
```

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
Out[3]: inf
.
.
.
.
```

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
Out[3]: inf
In [4]: 1/x
.
.
.
```

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
Out[3]: inf
In [4]: 1/x
Out[4]: 0.0
.
.
```

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
Out[3]: inf
In [4]: 1/x
Out[4]: 0.0
In [5]: 4 - x
.
```

```
In [1]: x = 1e500

In [2]: 1+x
Out[2]: inf
In [3]: x**3
Out[3]: inf
In [4]: 1/x
Out[4]: 0.0
In [5]: 4 - x
Out[5]: -inf
```

$$\lim_{x \to +\infty} x^2 - x$$

```
In [9]: x**2 - x
```

$$\lim_{x \to +\infty} x^2 - x$$

```
In [9]: x**2 - x
```

.

$$\lim_{x \to +\infty} x^2 - x$$

```
In [9]: x**2 - x
Out[9]: nan
```

$$\lim_{x \to +\infty} x^2 - x$$

```
In [9]: x**2 - x
Out[9]: nan
```


Aladdin
NAN
PITA BREAD
15 oz

```
In [10]: x
Out[10]: inf

In [11]: x - x
Out[11]: nan

In [12]: x / x
Out[12]: nan

In [13]: x - x == x - x
Out[13]: False

In [14]: x**2 - x
Out[14]: nan

In [15]: x * (x - 1)
Out[15]: inf

In [16]: x * (x - 1) == x**2 - x
Out[16]: False
```

Exercice 3

1. *Comment expliquer les résultats suivants :*

```
*Main> let f(x) = x^2 / sqrt(x^3 + 1)
*Main> f(1e100)
1.0e50
*Main> f(1e150)
0.0
*Main> f(1e200)
NaN
```

2. *Comment éviter le dernier NaN ?*

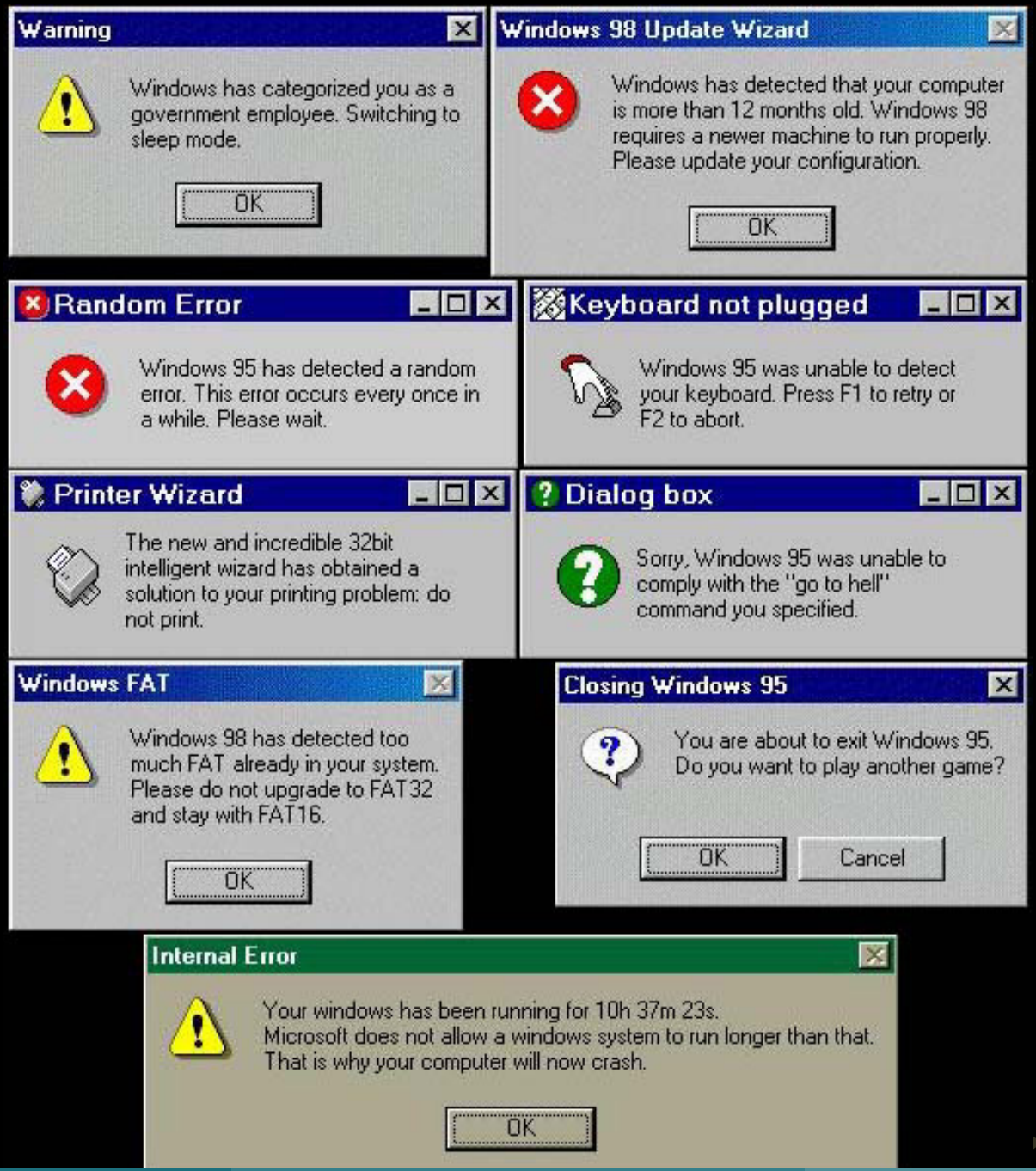
```
In [16]: f = lambda x: x**2 / sqrt(x**3 + 1)

In [17]: f(1e100)
Out[17]: 1e+50

In [18]: f(1e150)
---------------------------------------------------------------------------
OverflowError                                Traceback (most recent
        ↪  call last)
<ipython-input-18-79775d85f31a> in <module>()
----> 1 f(1e150)

<ipython-input-16-eed0b81ef932> in <lambda>(x)
----> 1 f = lambda x: x**2 / sqrt(x**3 + 1)

OverflowError: (34, 'Numerical result out of range')

In [19]: f(1e200)
---------------------------------------------------------------------------
OverflowError                                Traceback (most recent
```

```
In [53]: x = 1e-500

In [54]: x
Out[54]: 0.0

In [55]: x / x
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent
    ↪  call last)
<ipython-input-55-fd52a7f8b5f1> in <module>()
----> 1 x / x

ZeroDivisionError: float division by zero
```

```
In [56]: sqrt(-1.0)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent
        ↪  call last)
<ipython-input-56-d1c09f21b443> in <module>()
----> 1 sqrt(-1.0)

ValueError: math domain error
```

```
*Main> let x = 1e500
*Main> x - x
NaN
*Main> x / x
NaN
*Main> sqrt(-1)
NaN
*Main> 0 / 0
NaN
```

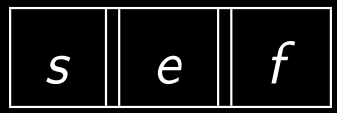
USS Yorktown 1998


Warning
Windows has categorized you as a government employee. Switching to sleep mode.
OK
Windows 98 Update Wizard
Windows has detected that your computer is more than 12 months old. Windows 98 requires a newer machine to run properly. Please update your configuration.
OK
Random Error
Windows 95 has detected a random error. This error occurs every once in a while. Please wait.
Keyboard not plugged
Windows 95 was unable to detect your keyboard. Press F1 to retry or F2 to abort.
Printer Wizard
The new and incredible 32bit intelligent wizard has obtained a solution to your printing problem: do not print.
Dialog box
Sorry, Windows 95 was unable to comply with the "go to hell" command you specified.
Windows FAT
Windows 98 has detected too much FAT already in your system. Please do not upgrade to FAT32 and stay with FAT16.
OK
Closing Windows 95
You are about to exit Windows 95. Do you want to play another game?
OK
Cancel
Internal Error
Your windows has been running for 10h 37m 23s.
Microsoft does not allow a windows system to run longer than that.
That is why your computer will now crash.
OK

Air France 447 - 1er juin 2009

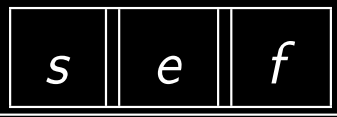
s (1 bit)	e (3 bits)			f (3 bits)		
1	1	1	1	0	1	0

Arbre de lecture



$e = 0\ldots0$	$f = 0$	$\rightarrow$	$v = (-1)^s \times 0.0$
$e = 0\ldots0$	$f \neq 0$	$\rightarrow$	$v = (-1)^s \times 0.f \times 2^{1-E_{\max}}$
$e = 1\ldots1$	$f = 0$	$\rightarrow$	$v = (-1)^s \times \infty$
$e = 1\ldots1$	$f \neq 0$	$\rightarrow$	NaN
$0\ldots0 < e < 1\ldots1$	$f \neq 0$	$\rightarrow$	$v = (-1)^s \times 1.f \times 2^{e-E_{\max}}$

Arbre de lecture



$e = 0\ldots0$	$f = 0$	$\rightarrow$	$v = (-1)^s \times 0.0$
$e = 0\ldots0$	$f \neq 0$	$\rightarrow$	$v = (-1)^s \times 0.f \times 2^{1-E_{\max}}$
$e = 1\ldots1$	$f = 0$	$\rightarrow$	$v = (-1)^s \times \infty$
$e = 1\ldots1$	$f \neq 0$	$\rightarrow$	NaN
$0\ldots0 < e < 1\ldots1$	$f \neq 0$	$\rightarrow$	$v = (-1)^s \times 1.f \times 2^{e-E_{\max}}$

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v)+1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

Successeur

$$n = \#f$$

toy7, $n = 3$

binary32, $n = 23$

binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v)+1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

Successeur

$$n = \#f$$

toy7, $n = 3$

binary32, $n = 23$

binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$succ(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$succ(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$succ(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v)+1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\mathrm{succ}(v) = (M(v)+1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\text{succ}(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

Successeur

$$n = \#f$$

toy7, $n = 3$
binary32, $n = 23$
binary64, $n = 52$

$$v = M(v) \times 2^{E(v)-n}$$

$$\text{succ}(v) = (M(v) + 1) \times 2^{E(v)-n} = v + 2^{E(v)-n}$$

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Tableau récapitulatif

- On notera ε_m l'*epsilon* de la machine, c'est-à-dire le successeur de 1
- On notera λ le plus petit VF normal positif
- On notera μ le plus petit VF sous-normal positif
- On notera Ω le plus grand VF normal.

Quelle relation existe-t-il entre μ, ε_m et λ ?

Nom	$\#E$	$\#f$	$E_{\min}$	$E_{\max}$	ε_m	λ	μ	Ω
toy7	3	3	-2	3	$2^{-3} = 1/8$	$2^{-2} = 1/4$	$2^{-5} = 1/32$	$1,111 \times 2^{3} = 15$
bin32	8	23	-126	127	2^{-23} $\approx 1,2 \times 10^{-7}$	2^{-126} $\approx 1,2 \times 10^{-38}$	$2^{-126-23}$ $\approx 1,4 \times 10^{-45}$	$(2^{24}-1) \times 2^{127-2}$ $\approx 3,4 \times 10^{38}$
bin64	11	52	-1022	1023	2^{-52} $\approx 2,2 \times 10^{-16}$	2^{-1022} $\approx 2,2 \times 10^{-308}$	2^{-1074} $\approx 5 \times 10^{-324}$	$(2^{53}-1)2^{1023-52}$ $\approx 1,8 \times 10^{308}$

```
In [1]: 1 + 1e-16 == 1
```

```
In [1]: 1 + 1e-16 == 1
Out[1]: True
```

.

.

.

.

.

.

.

.

.

.

.

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16
```

.

.

.

.

.

.

.

.

.

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
.
.
.
.
.
.
.
```

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
Out[3]: 0.0

.
.
.
.
.
```

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
Out[3]: 0.0

In [4]: x - 1e-16
```

.

.

.

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
Out[3]: 0.0

In [4]: x - 1e-16
Out[4]: 0.9999999999999999

.
.
```

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
Out[3]: 0.0

In [4]: x - 1e-16
Out[4]: 0.9999999999999999

In [5]: 1e-16 + 1e-18
.
```

```
In [1]: 1 + 1e-16 == 1
Out[1]: True

In [2]: x = 1 + 1e-16

In [3]: x - 1
Out[3]: 0.0

In [4]: x - 1e-16
Out[4]: 0.9999999999999999

In [5]: 1e-16 + 1e-18
Out[5]: 1.01e-16
```

Sommaire

$$\mathbb{V}_b$$

$$\overline{\mathbb{V}_b}$$

$$\mathbb{V}_b$$

$$\overline{\mathbb{V}_b}$$

Comparaison

Il est très simple et rapide de comparer deux VF : comment la machine procède-t-elle ? Quel est l'avantage de ce stockage des VF ?

Addition

1. on commence par ramener les deux nombres au même exposant, en l'occurrence le plus grand des deux ;
2. on ajoute les deux mantisses *complètes* en tenant compte du signe ;
3. on renormalise le nombre obtenu.

Addition

1. on commence par ramener les deux nombres au même exposant, en l'occurrence le plus grand des deux ;
2. on ajoute les deux mantisses *complètes* en tenant compte du signe ;
3. on renormalise le nombre obtenu.

Addition

1. on commence par ramener les deux nombres au même exposant, en l'occurrence le plus grand des deux ;
2. on ajoute les deux mantisses *complètes* en tenant compte du signe ;
3. on renormalise le nombre obtenu.

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^{0} + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

$$\begin{array}{l} 1,100 \\ 0,00111 \\ \hline 1,10111 \end{array}$$

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^0 + 1,11 \times 2^{-2}$

1,1 :	0	0	1	1	1	0	0

0,0011 :	0	0	0	1	1	1	0

1,100
0,00111
1,10111

Addition

En *toy7* : $1,1+0,0111 \quad \rightarrow \quad 1,1\times 2^{0}+1,11\times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,00111
_ _ _ _
1,10111

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^0 + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,001*11*
– – ––
1,101*11*

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^0 + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,001*11*
– – ––
1,101*11*

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^0 + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,001*11*
– – ––
1,101*11*

Addition

En *toy7* : $1,1 + 0,0111 \quad \rightarrow \quad 1,1 \times 2^0 + 1,11 \times 2^{-2}$

1,1 :

0	0	1	1	1	0	0

0,0011 :

0	0	0	1	1	1	0

1,100
0,001*11*
– – ––
1,101*11*

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

Les arrondis

Définition 1

1. l'arrondi au plus proche (RN) qui arrondit au VF...le plus proche. En cas d'égalité, on choisit la valeur paire (donc qui se termine par un 0 en binaire) ;
2. l'arrondi vers 0 (RZ) qui arrondit à la valeur de plus petite valeur absolue : c'est la troncature ;
3. l'arrondi vers $+\infty$ (RU) qui arrondit à la valeur supérieure la plus petite ;
4. l'arrondi vers $-\infty$ (RD) qui arrondit à la valeur inférieure la plus grande.

Le mode d'arrondi par défaut est le premier.

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1,00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1,11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10,00}\,ulp(x)}_{succ(v)}$$

$|x - v| = 0,11 ulp(x)$ $|x - succ(v)| = 0,01 ulp(x)$ donc $RN(x) = succ(v)$

$$1,1 \oplus 0,00111 = 1,110$$

$1,100$
$0,001\mathit{11}$
$---$
$1,101\mathit{11}$

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{succ(v)}$$

$|x - v| = 0,11 ulp(x)$ $|x - succ(v)| = 0,01 ulp(x)$ donc $RN(x) = succ(v)$

$$1,1 \oplus 0,00111 = 1,110$$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{succ(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - succ(v)| = 0,01\,ulp(x)$ donc $RN(x) = succ(v)$

$$1,1 \oplus 0,00111 = 1,110$$

1,100
0,001*11*
– – ––
1,101*11*

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{\mathrm{succ}(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - \mathrm{succ}(v)| = 0,01\,ulp(x)$ donc $RN(x) = \mathrm{succ}(v)$

$$1,1 \oplus 0,00111 = 1,110$$

$1,100$
$0,001\mathit{11}$
$---$
$1,101\mathit{11}$

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{\mathrm{succ}(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - \mathrm{succ}(v)| = 0,01\,ulp(x)$ donc $RN(x) = \mathrm{succ}(v)$

$$1,1 \oplus 0,00111 = 1,110$$

$1,100$
$0,001\mathit{11}$
$---$
$1,101\mathit{11}$

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{\mathrm{succ}(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - \mathrm{succ}(v)| = 0,01\,ulp(x)$ donc $RN(x) = \mathrm{succ}(v)$

$$1,1 \oplus 0,00111 = 1,110$$

$1,100$
$0,001\mathit{11}$
$---\!--$
$1,101\mathit{11}$

$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{\mathrm{succ}(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - \mathrm{succ}(v)| = 0,01\,ulp(x)$ donc $RN(x) = \mathrm{succ}(v)$

$$1,1 \oplus 0,00111 = 1,110$$

$1,100$
$0,001\mathit{11}$
$---\!--$
$1,101\mathit{11}$

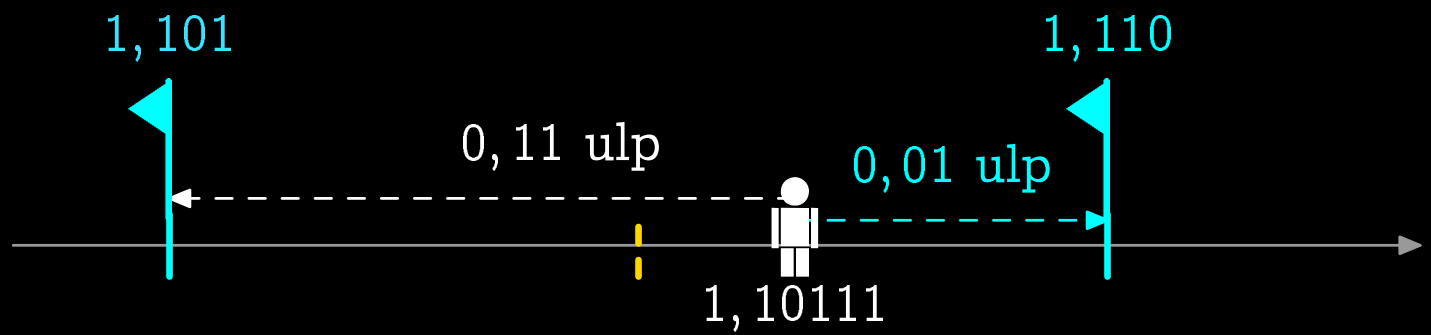
$x = 1,101\mathit{11}$ VF en *toy7* ?

$$1,101\mathit{00} \leqslant \underbrace{1,101\mathit{11}}_{x} \leqslant 1,110\mathit{00}$$

$$\underbrace{1,100 + \mathit{1},\mathit{00}\,ulp(x)}_{v} \leqslant \underbrace{1,100 + \mathit{1},\mathit{11}\,ulp(x)}_{x} \leqslant \underbrace{1,100 + \mathit{10},\mathit{00}\,ulp(x)}_{\mathrm{succ}(v)}$$

$|x - v| = 0,11\,ulp(x)$ $|x - \mathrm{succ}(v)| = 0,01\,ulp(x)$ donc $RN(x) = \mathrm{succ}(v)$

$$1,1 \oplus 0,00111 = 1,110$$


1, 101
1, 110
0, 11 ulp
0, 01 ulp
1, 10111

1. Est-ce que l'addition des VF est associative ?

```
In [14]: (1 + 1e-16) + 1e-16
Out[14]: 1.0

In [15]: 1 + (1e-16 + 1e-16)
Out[15]: 1.0000000000000002
```

2. Est-on sûr de l'ordre dans le quel un compilateur calcule $a + b + c + d$?

1. Est-ce que l'addition des VF est associative ?

```
In [14]: (1 + 1e-16) + 1e-16
Out[14]: 1.0

In [15]: 1 + (1e-16 + 1e-16)
Out[15]: 1.0000000000000002
```

2. Est-on sûr de l'ordre dans le quel un compilateur calcule $a + b + c + d$?

1. Est-ce que l'addition des VF est associative ?

```
In [14]: (1 + 1e-16) + 1e-16
Out[14]: 1.0

In [15]: 1 + (1e-16 + 1e-16)
Out[15]: 1.0000000000000002
```

2. Est-on sûr de l'ordre dans le quel un compilateur calcule $a+b+c+d$?

Multiplication

1. on « xore » les bits de signe ;
2. on additionne les exposants réels et on décale ou plutôt on additionne les exposants décalés et on retire la valeur d'un décalage ;
3. on multiplie les mantisses ;
4. on normalise.

Multiplication

1. on « xore » les bits de signe ;
2. on additionne les exposants réels et on décale ou plutôt on additionne les exposants décalés et on retire la valeur d'un décalage ;
3. on multiplie les mantisses ;
4. on normalise.

Multiplication

1. on « xore » les bits de signe ;
2. on additionne les exposants réels et on décale ou plutôt on additionne les exposants décalés et on retire la valeur d'un décalage ;
3. on multiplie les mantisses ;
4. on normalise.

Multiplication

1. on « xore » les bits de signe ;
2. on additionne les exposants réels et on décale ou plutôt on additionne les exposants décalés et on retire la valeur d'un décalage ;
3. on multiplie les mantisses ;
4. on normalise.

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

$\rightsquigarrow$ 0 xor 1 donne 1 : le bit de signe est 1 ;

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. 101 + 100 − 11 = 1001 − 11 = 110 : l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011 ulp(x)$ et $|x - 1,101| = 0,101 ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011\,ulp(x)$ et $|x - 1,101| = 0,101\,ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011 ulp(x)$ et $|x - 1,101| = 0,101 ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011 ulp(x)$ et $|x - 1,101| = 0,101 ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011 ulp(x)$ et $|x - 1,101| = 0,101 ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011 ulp(x)$ et $|x - 1,101| = 0,101 ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

$101,1 \times (-10,01)$

101,1 :

0	1	0	1	0	1	1

-10,01 :

1	1	0	0	0	0	1

1. 0 xor 1 donne 1 : le bit de signe est 1 ;
2. $101 + 100 - 11 = 1001 - 11 = 110$: l'exposant décalé est 110 donc l'exposant est 3 ;
3. $1,011 \times 1,001 = 1,011 + 1,011 \times 0,001 = 1,011 + 0,001011 = 1,100011$;
4. le produit est donc $1,100011 \times 2^3$. Le passage à la forme normale va arrondir le produit. On a

$$1,100 < x < 1,101$$

avec $|x - 1,100| = 0,011 ulp(x)$ et $|x - 1,101| = 0,101 ulp(x)$ donc

$$RN(101,1 \times (-10,01)) = -1,100 \times 2^3$$

1	1	1	0	1	0	0

1. Est-ce que la multiplication des VF est associative ?

```
In [1]: (1.00000001 * 1e-15) * 1e15
Out[1]: 1.0000000100000002

In [2]: 1.00000001 * (1e-15 * 1e15)
Out[2]: 1.00000001
```

2. Est-ce que la multiplication des VF est distributive sur l'addition ?

```
In [3]: 1e15 * (1e-16 + 1)
Out[3]: 10000000000000000.0

In [4]: (1e15 * 1e-16) + (1e15 * 1)
Out[4]: 10000000000000000.1
```

1. Est-ce que la multiplication des VF est associative ?

```
In [1]: (1.00000001 * 1e-15) * 1e15
Out[1]: 1.0000000100000002

In [2]: 1.00000001 * (1e-15 * 1e15)
Out[2]: 1.00000001
```

2. Est-ce que la multiplication des VF est distributive sur l'addition ?

```
In [3]: 1e15 * (1e-16 + 1)
Out[3]: 1000000000000000.0

In [4]: (1e15 * 1e-16) + (1e15 * 1)
Out[4]: 1000000000000000.1
```

1. Est-ce que la multiplication des VF est associative ?

```
In [1]: (1.00000001 * 1e-15) * 1e15
Out[1]: 1.0000000100000002

In [2]: 1.00000001 * (1e-15 * 1e15)
Out[2]: 1.00000001
```

2. Est-ce que la multiplication des VF est distributive sur l'addition ?

```
In [3]: 1e15 * (1e-16 + 1)
Out[3]: 1000000000000000.0

In [4]: (1e15 * 1e-16) + (1e15 * 1)
Out[4]: 1000000000000000.1
```

1. Est-ce que la multiplication des VF est associative ?

```
In [1]: (1.00000001 * 1e-15) * 1e15
Out[1]: 1.0000000100000002

In [2]: 1.00000001 * (1e-15 * 1e15)
Out[2]: 1.00000001
```

2. Est-ce que la multiplication des VF est distributive sur l'addition ?

```
In [3]: 1e15 * (1e-16 + 1)
Out[3]: 1000000000000000.0

In [4]: (1e15 * 1e-16) + (1e15 * 1)
Out[4]: 1000000000000000.1
```

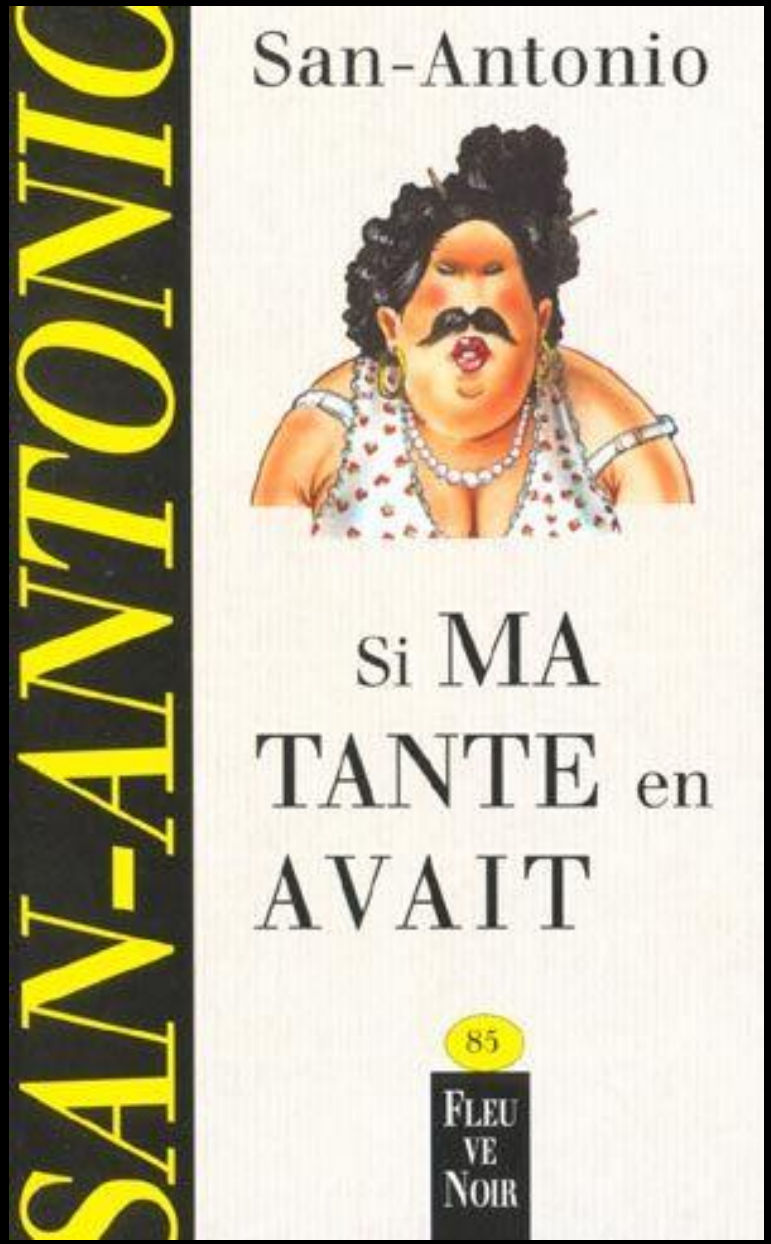
Sommaire

```python
def base1(a):
    if (a + 1.0) - a != 1.0:
        return a
    else:
        return base1(2.0 * a)

def base2(a, b):
    if (a + b) - a == b:
        return b
    else:
        return base2(a, b + 1.0)
```

????!!!????!!???

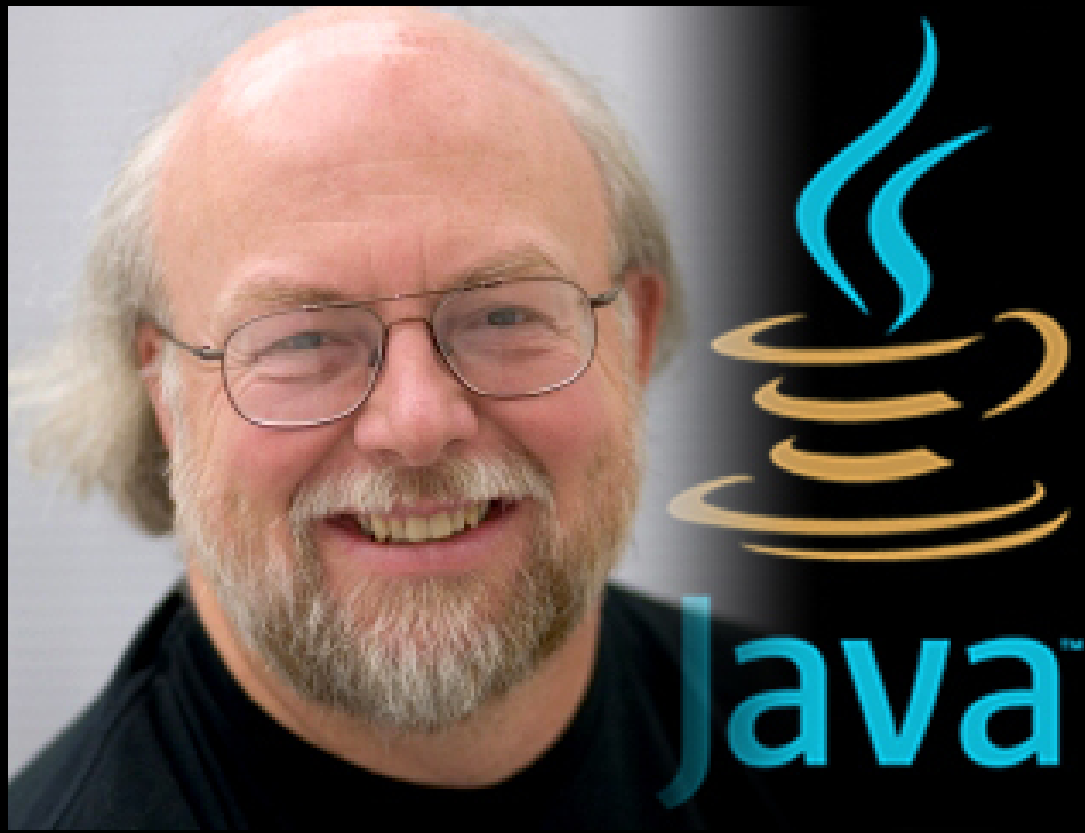
```python
def base1(a):
    if (a + 1.0) - a != 1.0:
        return a
    else:
        return base1(2.0 * a)

def base2(a, b):
    if (a + b) - a == b:
        return b
    else:
        return base2(a, b + 1.0)
```

????!!!!????!!???


SAN-ANTONIO
San-Antonio
Si MA TANTE en AVAIT
85
Fleuve Noir

95 % of the folks out there are completely clueless about floating-point

James Gosling *(M. Java) - 28 février 1998*




LA COLLECTION
RYAN GOSLING
DRIVE // LES MARCHES DU POUVOIR // CRAZY STUPID LOVE
Blu-ray Disc

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple `3`, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

`3.1777777777777777` est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple 3, 3.0, 3.0000, 3.0e0 n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

3,1777777777777777 est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple `3`, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

$3,1777777777777777$ est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple `3`, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

$3,17777777777777$ est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple `3`, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

$3,17777777777777$ est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Précision (*precision*) : c'est le nombre de bits utilisés pour représenter un nombre. La précision concerne donc le format utilisé pour écrire ou stocker ou arrondir un nombre. Par exemple `3`, `3.0`, `3.0000`, `3.0e0` n'ont pas la même précision, précision qui dépend en plus du langage.

Exactitude (*accuracy*) : c'est ce qui relie un nombre au contexte dans lequel il est employé.

$3,177777777777777$ est une approximation plutôt précise (16 décimales) mais inexacte (2 décimales) de π.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \dfrac{|x - \widehat{x}|}{|x|}$

commise en prenant $\widehat{x}$ à la place de x.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1+\eta)$

commise en prenant $\widehat{x}$ à la place de x.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1+\eta)$

commise en prenant $\widehat{x}$ à la place de x.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1+\eta)$

commise en prenant $\widehat{x}$ à la place de x.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1+\eta)$

commise en prenant $\widehat{x}$ à la place de x.

Soit x un nombre et $\widehat{x}$ le nombre qui le représente. On distingue :

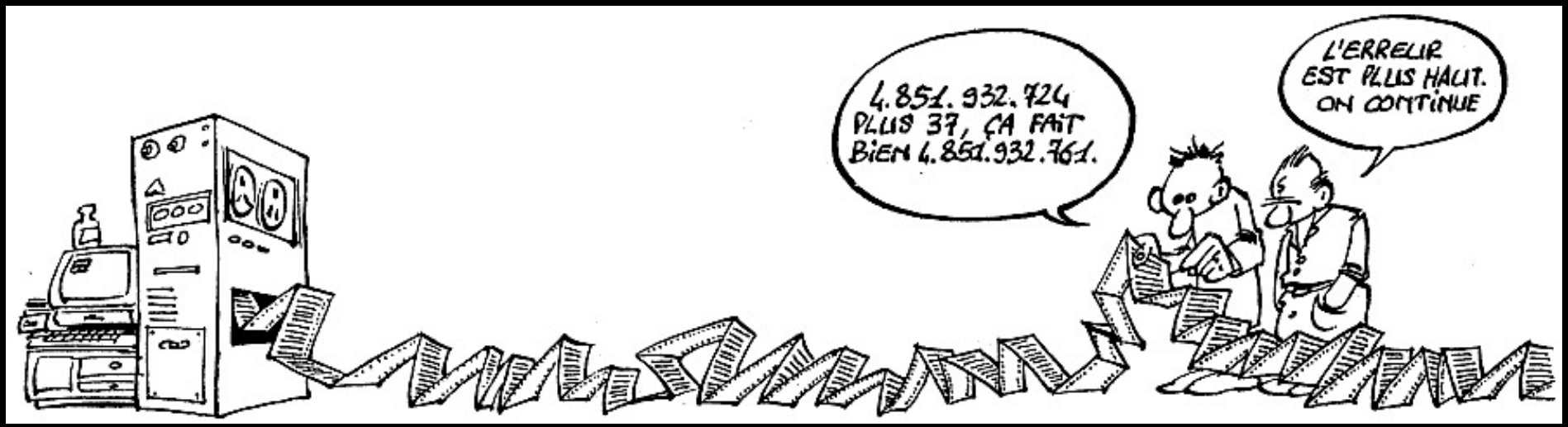
l'erreur absolue $|x - \widehat{x}|$

l'erreur relative $\eta = \frac{x-\widehat{x}}{x}$ alors $\widehat{x} = x(1+\eta)$

commise en prenant $\widehat{x}$ à la place de x.

Feel nervous, but feel in control. It's not dark magic, it's science.

Florent de Dinechin



Sommaire

Précision machine

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} \leqslant \frac{1}{2} \varepsilon_M$$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} \leqslant \frac{1}{2} \frac{\varepsilon_M}{m}$$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} \leqslant \frac{1}{2} \frac{1}{m}$$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

- Si $x \neq 0$ alors l'erreur relative est majorée $\left|\frac{x - \widehat{x}}{x}\right| \leqslant \frac{1}{2} \varepsilon_M$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'erreur relative est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'erreur absolue est majorée

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée par $\frac{1}{2}\varepsilon_M$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2}\frac{2^{E-n}}{m \times 2^E} = \frac{1}{2}\frac{2^{-n}}{m} = \frac{1}{2}\frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

Précision machine

$$M \times 2^{E-n} \leqslant x < (M+1) \times 2^{E-n} = M \times 2^{E-n} + 2^{E-n}$$

$$|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E-n}$$

$$\left|\frac{x - \widehat{x}}{x}\right| \leqslant \left|\frac{x - \widehat{x}}{m \times 2^E}\right| \leqslant \frac{1}{2} \frac{2^{E-n}}{m \times 2^E} = \frac{1}{2} \frac{2^{-n}}{m} = \frac{1}{2} \frac{\varepsilon_M}{m}$$

1. Si $\widehat{x}$ VF alors l'**erreur relative** est majorée $\left|\frac{x-\widehat{x}}{x}\right| \leqslant \frac{1}{2}\varepsilon_M$
2. Si $\widehat{x}$ est sous-normal, alors l'**erreur absolue** est majorée $|x - \widehat{x}| \leqslant \frac{1}{2} 2^{E_{\min}-1-n}$

NE FAITES PAS DES TESTS D'ÉGALITÉ MAIS DES TESTS D'APPARTENANCE À DES INTERVALLES DE LARGEUR L'ε MACHINE !

Précision machine

$$\widehat{x} = x(1+\eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$
- si $\widehat{x}$ est sous-normal $\eta_1 = 0$ et $|\eta_2| \leqslant \frac{1}{2}2^{E_{\min}-1-n}$

Mouais...

Précision machine

$$\widehat{x} = x(1+\eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$
- si $\widehat{x}$ est sous-normal $\eta_1 = 0$ et $|\eta_2| \leqslant \frac{1}{2}2^{E_{\min}-1-n}$

Mouais...

Précision machine

$$\widehat{x} = x(1+\eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$
- si $\widehat{x}$ est sous-normal $\eta_1 = 0$ et $|\eta_2| \leqslant \frac{1}{2}2^{E_{\min}-1-n}$

Mouais...

Précision machine

$$\widehat{x} = x(1+\eta_1) + \eta_2$$

avec :

- si $\widehat{x}$ est normal $|\eta_1| \leqslant \frac{1}{2}\varepsilon_M$ et $\eta_2 = 0$
- si $\widehat{x}$ est sous-normal $\eta_1 = 0$ et $|\eta_2| \leqslant \frac{1}{2}2^{E_{\min}-1-n}$

Mouais...

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Élimination

$\widehat{a} = a(1+\eta_a), \widehat{b} = b(1+\eta_b), x = a-b$ et $\widehat{x} = \widehat{a-b} = \widehat{a} - \widehat{b}$.

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\eta_a + b\eta_b}{a-b}\right| \leqslant \max(|\eta_a|, |\eta_b|)\frac{|a|+|b|}{|a-b|}$$

Catastrophe

```python
def deriv(f,h,x):
    return (f(x+h) - f(x)) / h
```

```python
def derivn(f,x,h,n):
    if n == 0:
        return f(x)
    else:
        return derivn(lambda x: deriv(f,h,x),x,h,n-1)
```

Catastrophe

```python
def deriv(f,h,x):
    return (f(x+h) - f(x)) / h
```

```python
def derivn(f,x,h,n):
    if n == 0:
        return f(x)
    else:
        return derivn(lambda x: deriv(f,h,x),x,h,n-1)
```

Catastrophe

```
In [1]: [derivn(lambda x: x**4,1,1e-7,k) for k in range(5)]
Out[1]:
[1,
 4.000000601855902,
 12.012613126444194,
 -222044.6049250313,
 2220446049250.313]
```

```
In [2]: [derivn(lambda x: x**4,1,1e-8,k) for k in range(5)]
Out[2]: [1, 4.000000042303498, 11.102230246251565, 0.0,
        ↪  2.22044604925031e+16]
```

Catastrophe

```
In [1]: [derivn(lambda x: x**4,1,1e-7,k) for k in range(5)]
Out[1]:
[1,
 4.000000601855902,
 12.012613126444194,
 -222044.6049250313,
 2220446049250.313]
```

```
In [2]: [derivn(lambda x: x**4,1,1e-8,k) for k in range(5)]
Out[2]: [1, 4.000000042303498, 11.102230246251565, 0.0,
        ↪  2.220446049250313e+16]
```

Catastrophe

```
In [3]: [derivn(lambda x: x**4,1,1e-9,k) for k in range(5)]
Out[3]: [1, 4.000000330961484, 0.0, 0.0, 0.0]
```

```
In [4]: [derivn(lambda x: x**4,1,1e-3,k) for k in range(6)]
Out[4]:
[1,
 4.006004000999486,
 12.024014000244776,
 24.0359941303014,
 24.00124543556484,
 -2.442490654175344]
```

Catastrophe

```
In [3]: [derivn(lambda x: x**4,1,1e-9,k) for k in range(5)]
Out[3]: [1, 4.000000330961484, 0.0, 0.0, 0.0]
```

```
In [4]: [derivn(lambda x: x**4,1,1e-3,k) for k in range(6)]
Out[4]:
[1,
 4.006004000999486,
 12.024014000244776,
 24.03599941303014,
 24.001245435556484,
 -2.442490654175344]
```

Catastrophe

```
In [5]: [derivn(lambda x: x**4,1,2**-10,k) for k in range(6)]
Out[5]: [1, 4.005863190628588, 12.02345085144043, 24.03515625,
        ↪  24.0, 0.0]
```

```
In [6]: [derivn(lambda x: x**4,1,1.0/1026.,k) for k in range(6)]
Out[6]:
[1,
 4.005851753982072,
 12.023405112200917,
 24.035087928668187,
 23.999573957547014,
 0.755876563500351]
```

Catastrophe

```
In [5]: [derivn(lambda x: x**4,1,2**-10,k) for k in range(6)]
Out[5]: [1, 4.005863190628588, 12.02345085144043, 24.03515625,
        ↪  24.0, 0.0]
```

```
In [6]: [derivn(lambda x: x**4,1,1.0/1026.,k) for k in range(6)]
Out[6]:
[1,
 4.005851753982072,
 12.023405112200917,
 24.035087928668187,
 23.999573957547014,
 0.755876563500351]
```

Catastrophe

```
In [7]: [derivn(lambda x: x**4,1,2**-20,k) for k in range(6)]
Out[7]: [1, 4.000005722045898, 12.0, 0.0, 268435456.0,
         ↪  -844424930131968.0]
```

Catastrophe

```
In [7]: [derivn(exp,0,2**-5,k) for k in range(10)]
Out[7]:
[1.0,
 1.015789039971288 3,
 1.031827373725491 3,
 1.048118937389517 7,
 1.064667728496715 4,
 1.081477828323841,
 1.098553419113159 2,
 1.115844726562 5,
 1.137695312 5,
 0.953125]
```

Catastrophe

```
In [8]: [derivn(exp,0,2**-20,k) for k in range(10)]
Out[8]: [1.0, 1.0000004768371582, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,
        ↪  0.0, 0.0]
```

TROP DE PRÉCISION TUE LA PRÉCISION !

DIVISEZ PAR DES PUISSANCES DE 2...PAS DES PUISSANCES DE 10 !

TROP DE PRÉCISION TUE LA PRÉCISION !
DIVISEZ PAR DES PUISSANCES DE 2...PAS DES PUISSANCES DE 10 !

TROP DE PRÉCISION TUE LA PRÉCISION !
DIVISEZ PAR DES PUISSANCES DE 2...PAS DES PUISSANCES DE 10 !

Exercice 4

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$?*
4. *Et dans le cas de $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$?*
5. *Moralité ?*

Exercice 4

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$?*
4. *Et dans le cas de $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$?*
5. *Moralité ?*

Exercice 4

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$?*
4. *Et dans le cas de $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$?*
5. *Moralité ?*

Exercice 4

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$?*
4. *Et dans le cas de $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$?*
5. *Moralité ?*

Exercice 4

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$?*
4. *Et dans le cas de $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$?*
5. *Moralité ?*

Exercice 4

Vous savez résoudre une équation du type $ax^2 + bx + c = 0$ avec $a \neq 0$...
Les racines, si elles existent, sont données par une formule bien connue dépendant de a, b et c :

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{avec} \quad \Delta = b^2 - 4ac$$

Où peuvent se cacher d'éventuelles annulations catastrophiques ? Étudiez ces cas avec attention, voyez si vous pouvez éviter les éliminations catastrophiques en réécrivant les formules un peu dans l'esprit de la « levée d'indétermination ».

1. *Que se passe-t-il lorsque $b^2 \gg |4ac|$? En quoi la formule $\frac{-\left(b+\text{signe}(b)\sqrt{\Delta}\right)}{2a}$ peut aider ?*
2. *Que se passe-t-il lorsque $b^2 \approx 4ac$? Peut-on y remédier ? Que peut-on dire de Δ par rapport à b^2 ? Y a-t-il élimination catastrophique ?*
3. *Que se passe-t-il dans le cas de l'équation $10^{200}x^2 - 3 \times 10^{200}x + 2 \times 10^{200} = 0$?*
4. *Et dans le cas de $10^{-200}x^2 - 3x + 2 \times 10^{200} = 0$?*
5. *Moralité ?*

Exercice 5 (Max IEEE)

Fonction max(x, y : *flottants*) : *flottant*

Si $x >= y$ **Alors**

| max = x

Sinon

| max = y

FinSi

```
In [1]: max(1,1+1e-16)
Out[1]: 1

In [2]: max(0.1,1e500/1e500)
Out[2]: 0.1

In [3]: 0.1 <= 1e500/1e500
Out[3]: False

In [4]: 0.1 > 1e500/1e500
Out[4]: False
```

Élimination

```
def expn(n):
    return (1. + 1./n)**n
```

Élimination

```
In [1]: [(exp(1) -
        ↪  expn(10.0**k)) for k
        ↪  in range(20)]
Out[1]:
[0.718281828459045,
 0.1245393683590427 8,
 0.01346799037516 61,
 0.001357896223451 5046,
 0.000135901634119584,
 1.35912667476034 7e-05,
 1.35936329170860 4e-06,
 1.34326963302668 4e-07,
 3.01116873657747 4e-08,
```

```
-2.2355251516614771e-07,
-2.2477574246337895e-07,
-2.2489806505987 53e-07,
-0.0002416675781926 6138,
0.0021717943721442 09,
0.0021717943722 0845,
-0.3167533780902 1675,
1.7182818284590 45,
1.7182818284590 45,
1.7182818284590 45,
1.7182818284590 45]
```

Élimination

```
In [2]: [(exp(1) -
        ↪  expn(2.0**(3*k)))
        ↪  for k in range(20)]
Out[2]:
[0.718281828459045,
 0.152497314508697,
 0.020936875893946105,
 0.002649828290053779 5,
 0.000331747269379345 5,
 4.147652875108321e-05,
 5.184692989274315e-06,
 6.48088607668739 8e-07,
 8.101110671177025e-08,
```

```
 1.012638861652703 8e-08,
 1.265798577065879 7e-09,
 1.582245445774788e-10,
 1.977795704988239e-11,
 2.472244631235298 6e-12,
 3.090860900556436e-13,
 3.863576125695545e-14,
 4.884981308350689e-15,
 4.440892098500626e-16,
 1.718281828459045,
 1.718281828459045]
```

```
In [89]: [2.0**53 + k for k in range(10)]
Out[89]:
[9007199254740992.0,
 9007199254740992.0,
 9007199254740994.0,
 9007199254740996.0,
 9007199254740996.0,
 9007199254740996.0,
 9007199254740998.0,
 9007199254741000.0,
 9007199254741000.0,
 9007199254741000.0]
```

```
In [91]: [2.0**55 + k for k in range(10)]
Out[91]:
[3.602879701896397e+16,
 3.602879701896397e+16,
 3.602879701896397e+16,
 3.602879701896397e+16,
 3.602879701896397e+16,
 3.6028797018963976e+16,
 3.6028797018963976e+16,
 3.6028797018963976e+16,
 3.6028797018963976e+16,
 3.6028797018963976e+16]
```

Attention !

Pour éviter de déduire des lemmes suivants des théorèmes totalement faux, n'oubliez pas que **DANS CE QUI SUIT X ET Y SONT DES NOMBRES À VIRGULE FLOTTANTE** !

Lemme 2 (Majoration de l'erreur d'une somme)

Posons $x \oplus y = x + y + \mathrm{err}(x \oplus y)$. *Alors, s'il n'y a pas de dépassement de capacité,*

$$|\mathrm{err}(x \oplus y)| \leqslant \min(|x|, |y|)$$

On a bien sûr un résultat analogue pour la différence

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

Attention

De l'importance de disposer avec la IEEE 754 de la meilleure approximation !

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

À noter

De l'importance de disposer avec la IEEE 754 de la meilleure approximation !

- $x \oplus y = x + y + \mathrm{err}(x \oplus y)$
- $x \quad = x + y + (-y)$
- x est un VF situé à la distance $|y|$ de $x + y$
- $x \oplus y$ le VF le plus proche de $x + y$
- $|\mathrm{err}(x \oplus y)| \leqslant |y|$
- de même $|\mathrm{err}(x \oplus y)| \leqslant |x|$

À noter

De l'importance de disposer avec la IEEE 754 de la **meilleure approximation** !

Corollaire 3 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

Corollaire 3 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF

Corollaire 3 (Møller, Knuth, Dekker)

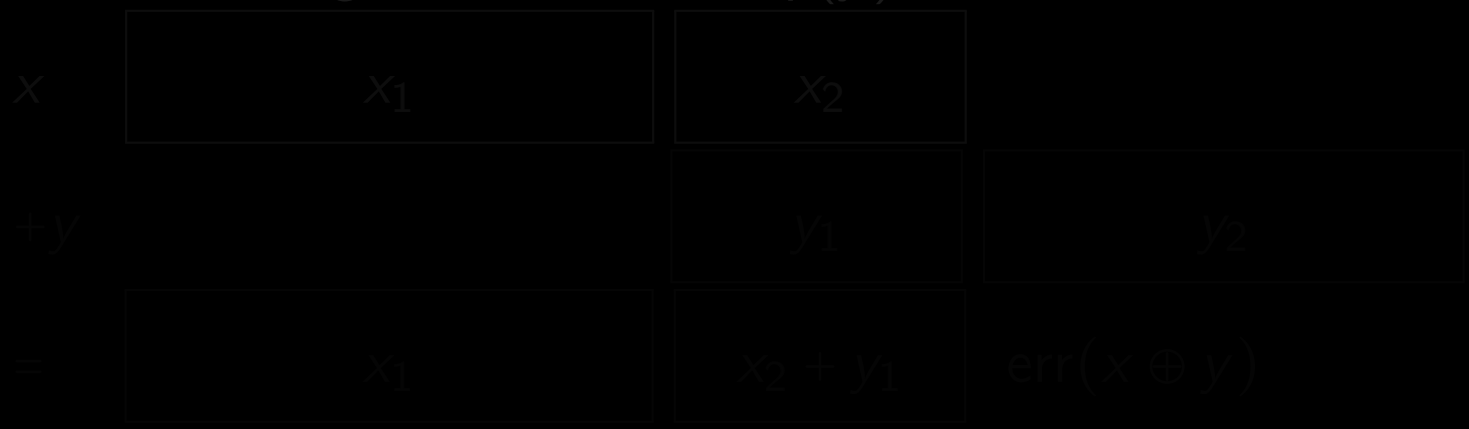
L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF

Corollaire 3 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

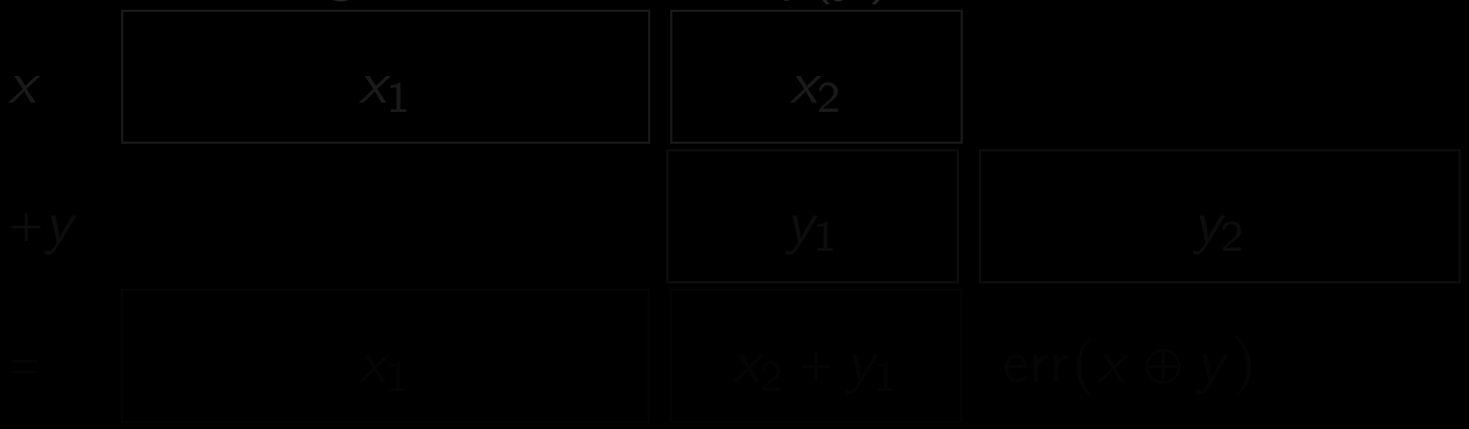
- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$



Corollaire 3 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

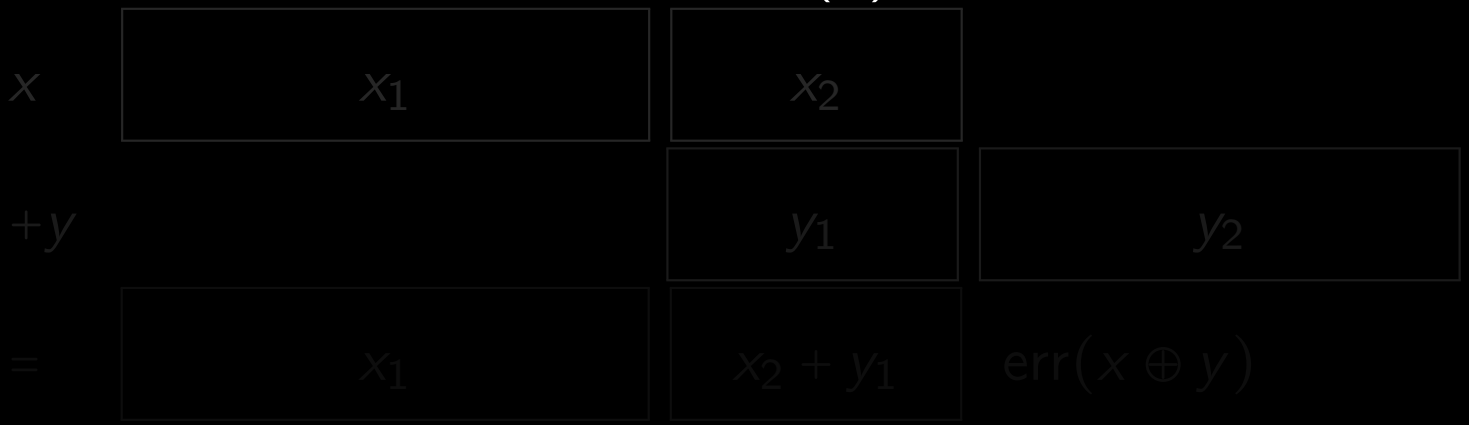
- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$



Corollaire 3 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

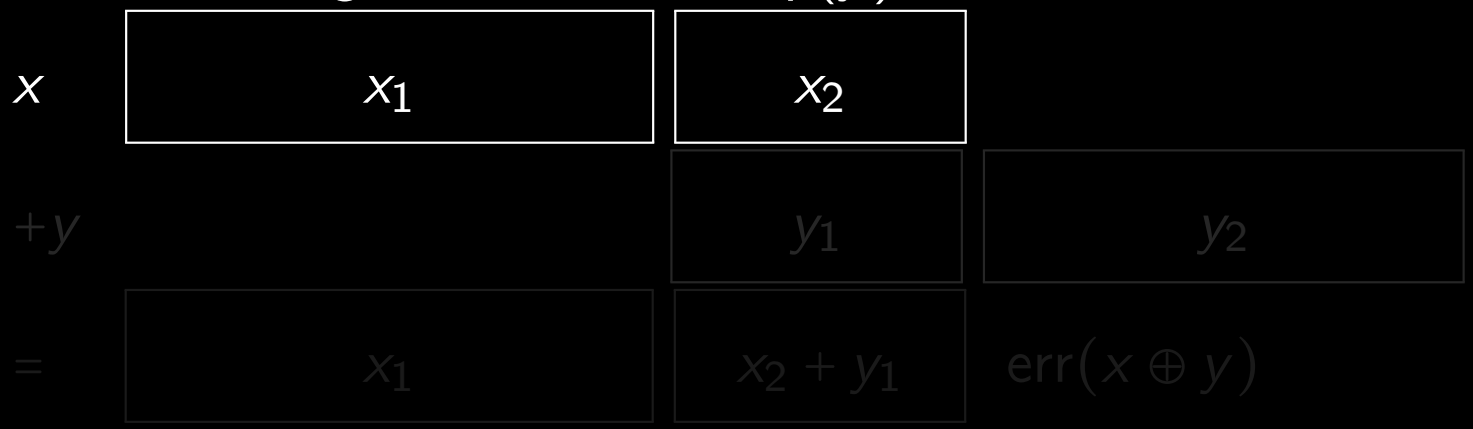
- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$



Corollaire 3 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

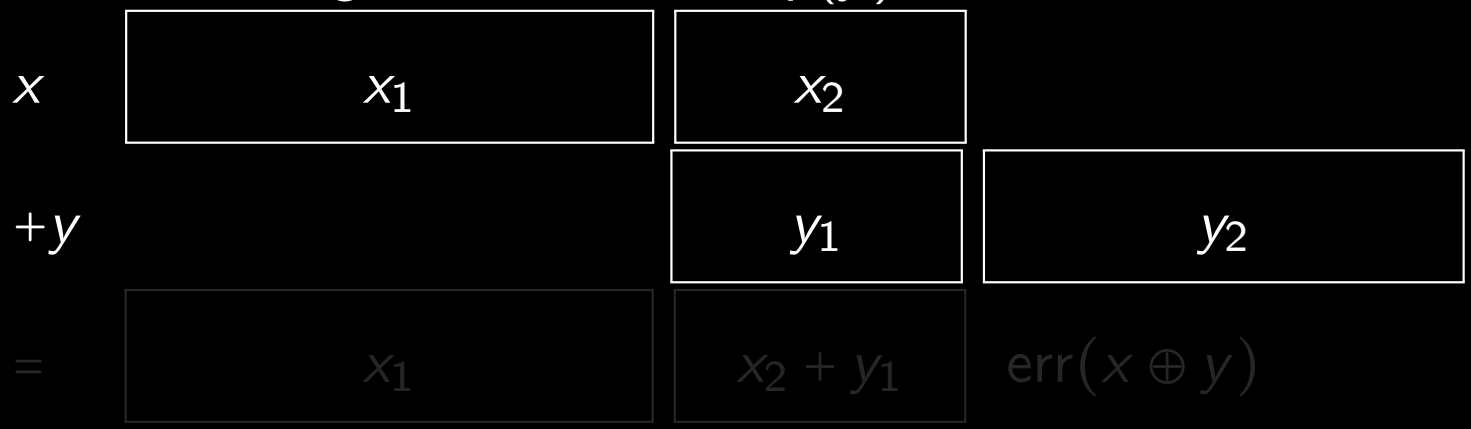
- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$



Corollaire 3 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

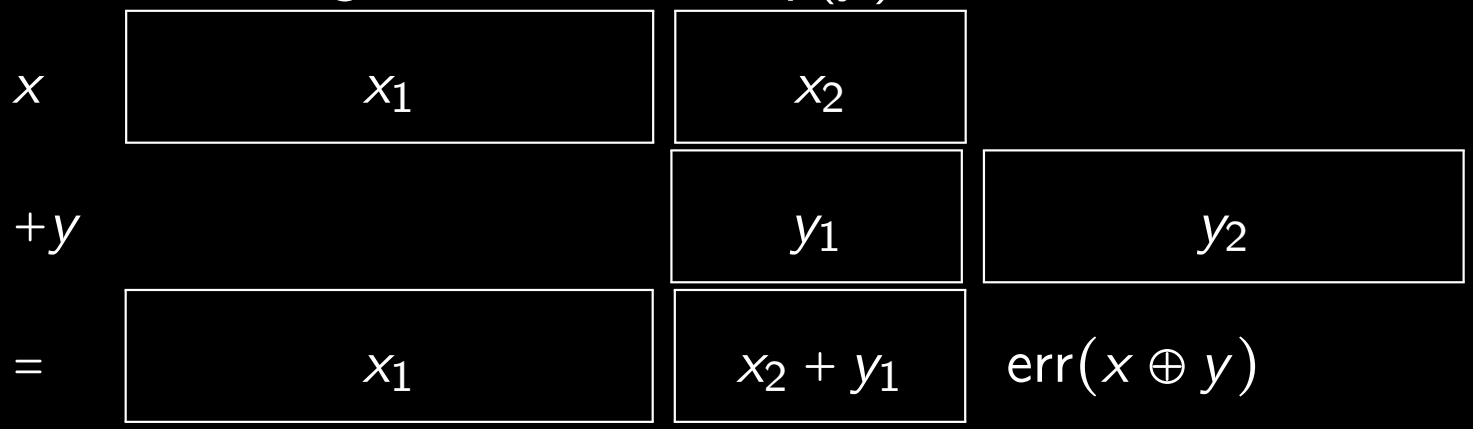


$\Rightarrow |\mathrm{err}(x \oplus y)| \leqslant |y|$

Corollaire 3 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

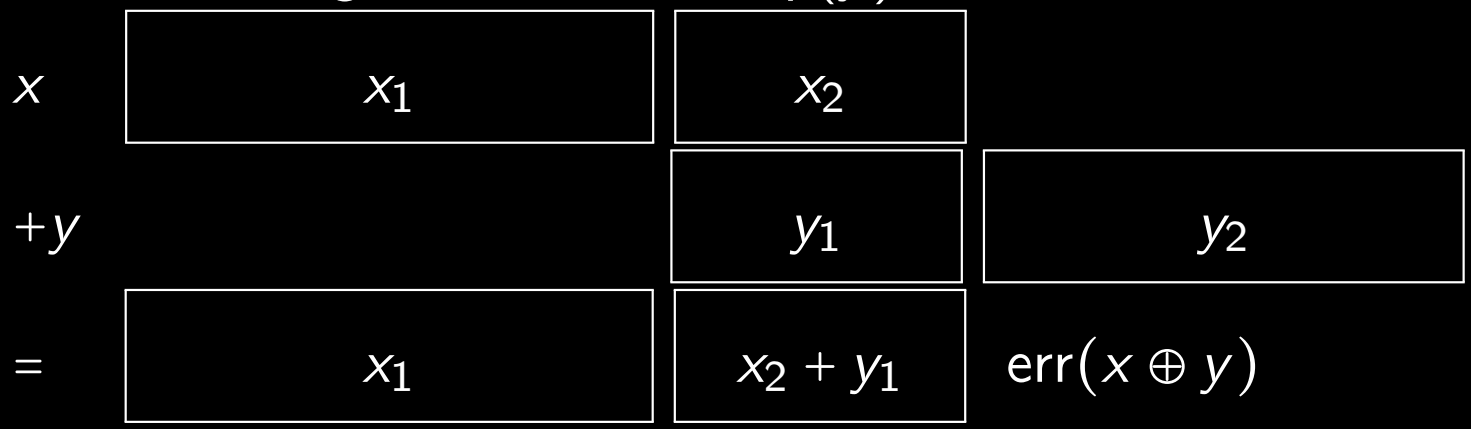


- $|\mathrm{err}(x \oplus y)| \leqslant |y|$ donc la mantisse de $\mathrm{err}(x \oplus y)$ a une longueur inférieure à p bits

Corollaire 3 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

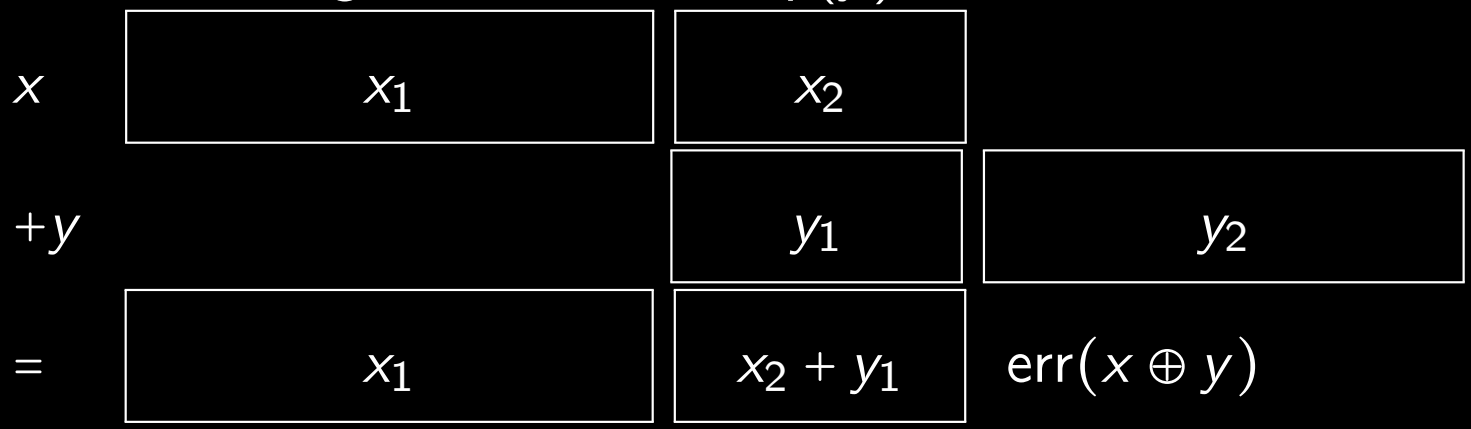


- $|\mathrm{err}(x \oplus y)| \leqslant |y|$ donc la mantisse entière de $\mathrm{err}(x \oplus y)$ a une longueur inférieure à p bits

Corollaire 3 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$

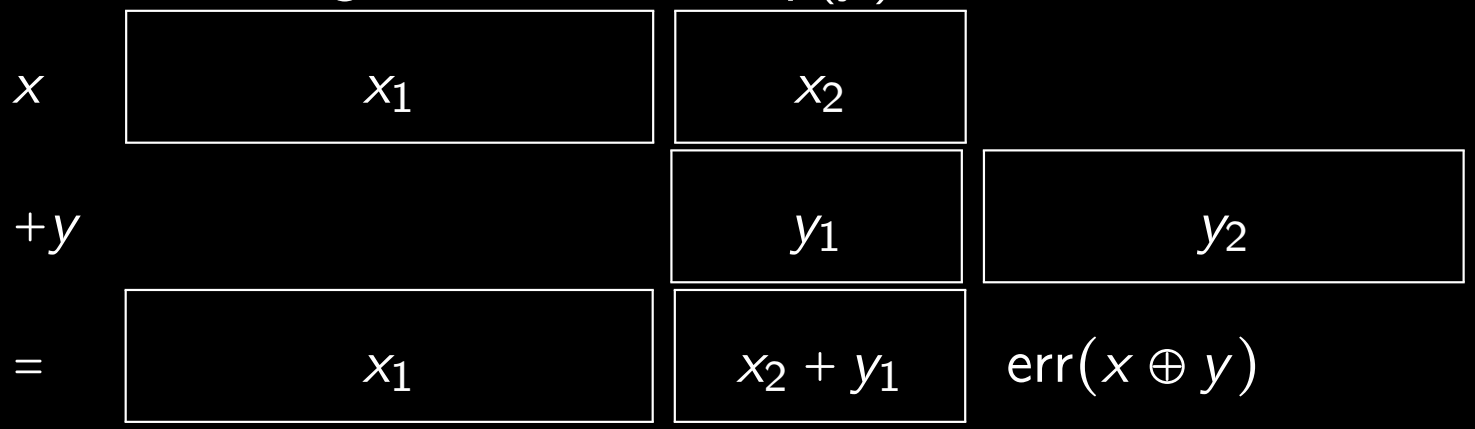


- $|\mathrm{err}(x \oplus y)| \leqslant |y|$donc la mantisse entière de $\mathrm{err}(x \oplus y)$ a une longueur inférieure à p bits

Corollaire 3 (Møller, Knuth, Dekker)

L'erreur commise $|\mathrm{err}(x \oplus y)|$ *peut être exprimée exactement sur* p *bits.*

- $|x| \geqslant |y|$
- x et y sont des VF : le plus petit bit significatif de $\mathrm{err}(x \oplus y)$ est au moins de magnitude celle de $\mathrm{ulp}(y)$



- $|\mathrm{err}(x \oplus y)| \leqslant |y|$donc la mantisse entière de $\mathrm{err}(x \oplus y)$ a une longueur inférieure à p bits

Lemme 4

Supposons que $|x+y| \leqslant \min(|x|,|y|)$, *alors* $x \oplus y = x+y$.
On obtient un résultat analogue pour la soustraction.

1. Supposons, sans perdre de généralité, que $|x| \geqslant |y|$
2. Le plus petit bit significatif de $x+y$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
3. $|x+y| \leqslant |y|$ donc la mantisse entière de $x+y$ a une longueur inférieure à p bits

Lemme 4

Supposons que $|x+y| \leqslant \min(|x|,|y|)$, *alors* $x \oplus y = x + y$.
On obtient un résultat analogue pour la soustraction.

1. Supposons, sans perdre de généralité, que $|x| \geqslant |y|$
2. Le plus petit bit significatif de $x+y$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
3. $|x+y| \leqslant |y|$ donc la mantisse entière de $x+y$ a une longueur inférieure à p bits

Lemme 4

Supposons que $|x+y| \leqslant \min(|x|,|y|)$, *alors* $x \oplus y = x+y$.
On obtient un résultat analogue pour la soustraction.

1. Supposons, sans perdre de généralité, que $|x| \geqslant |y|$
2. Le plus petit bit significatif de $x+y$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
3. $|x+y| \leqslant |y|$ donc la mantisse entière de $x+y$ a une longueur inférieure à p bits

Lemme 4

Supposons que $|x+y| \leqslant \min(|x|,|y|)$, *alors* $x \oplus y = x+y$.
On obtient un résultat analogue pour la soustraction.

1. Supposons, sans perdre de généralité, que $|x| \geqslant |y|$
2. Le plus petit bit significatif de $x+y$ est au moins de magnitude celle de $\mathrm{ulp}(y)$
3. $|x+y| \leqslant |y|$ donc la mantisse entière de $x+y$ a une longueur inférieure à p bits

DANGER

Nous avons démontré nos lemmes en considérant des VF x et y.
Est-ce que le résultat suivant contredit notre dernier lemme ?

```
In [86]: 1 - 0.9
Out[86]: 0.09999999999999998
```

Lemme 5 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

Si $x < y$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$.

Dans tous les cas $|x - y| \leqslant \min(|x|, |y|)$ et on peut utiliser le lemme 4 page 295.

Lemme 5 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $x < y$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$;
2. $x \geqslant y$: alors $\frac{x}{2} \leqslant y \leqslant x$ donc $-\frac{x}{2} \leqslant y - x \leqslant 0$ et par suite $0 \leqslant x - y \leqslant \frac{x}{2} \leqslant y \leqslant x$.

Dans tous les cas $|x - y| \leqslant \min(|x|, |y|)$ et on peut utiliser le lemme 4 page 295.

Lemme 5 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $\boldsymbol{x < y}$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$;
2. $\boldsymbol{x \geqslant y}$: alors $\frac{x}{2} \leqslant y \leqslant x$ donc $-\frac{x}{2} \leqslant y - x \leqslant 0$ et par suite $0 \leqslant x - y \leqslant \frac{x}{2} \leqslant y \leqslant x$.

Dans tous les cas $|x - y| \leqslant \min(|x|, |y|)$ et on peut utiliser le lemme 4 page 295.

Lemme 5 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $\boldsymbol{x < y}$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$;
2. $\boldsymbol{x \geqslant y}$: alors $\frac{x}{2} \leqslant y \leqslant x$ donc $-\frac{x}{2} \leqslant y - x \leqslant 0$ et par suite $0 \leqslant x - y \leqslant \frac{x}{2} \leqslant y \leqslant x$.

Dans tous les cas $|x - y| \leqslant \min(|x|, |y|)$ et on peut utiliser le lemme 4 page 295.

Lemme 5 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $\boldsymbol{x < y}$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$;
2. $\boldsymbol{x \geqslant y}$: alors $\frac{x}{2} \leqslant y \leqslant x$ donc $-\frac{x}{2} \leqslant y - x \leqslant 0$ et par suite $0 \leqslant x - y \leqslant \frac{x}{2} \leqslant y \leqslant x$.

Dans tous les cas $|x - y| \leqslant \min(|x|, |y|)$ et on peut utiliser le lemme 4 page 295.

Lemme 5 (Lemme de Sterbenz (1973))

Soit $(x, y) \in \mathbb{V}^2$ *vérifiant* $\frac{x}{2} \leqslant y \leqslant 2x$ *alors*

$$x \ominus y = x - y$$

La différence de deux VF suffisamment proches est donc exacte.

1. $\boldsymbol{x < y}$: alors $x < y \leqslant 2x$ donc $0 < y - x \leqslant x \leqslant y$;
2. $\boldsymbol{x \geqslant y}$: alors $\frac{x}{2} \leqslant y \leqslant x$ donc $-\frac{x}{2} \leqslant y - x \leqslant 0$ et par suite $0 \leqslant x - y \leqslant \frac{x}{2} \leqslant y \leqslant x$.

Dans tous les cas $|x - y| \leqslant \min(|x|, |y|)$ et on peut utiliser le lemme 4 page 295.

- Concrètement, à quoi correspond cette condition $\frac{x}{2} \leqslant y \leqslant 2x$?
- Demandez-vous ce que l'on peut dire de l'écart maximum entre les exposants de x et y.

- Concrètement, à quoi correspond cette condition $\frac{x}{2} \leqslant y \leqslant 2x$?
- Demandez-vous ce que l'on peut dire de l'écart maximum entre les exposants de x et y.

```
In [9]: 2.**53
Out[9]: 9007199254740992.0

In [10]: 2.**53 + 1
Out[10]: 9007199254740992.0

In [11]: 2.**53 + 2
Out[11]: 9007199254740994.0

In [12]: 2.**53 + 3
Out[12]: 9007199254740996.0

In [13]: 2.**53 + 4
Out[13]: 9007199254740996.0

In [14]: 2.**53 + 5
Out[14]: 9007199254740996.0
```

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53+1023}$$

0 10000110100 0000000000000000000000000000000000000000000000000000

$$1 = (-1)^0 \times 1,000...000 \times 2^{0+1023}$$

0 01111111111 0000000000000000000000000000000000000000000000000000

$$1 = 0,\underbrace{000...000}_{52bits}1 \times 2^{53}$$

1,0000000000000000000000000000000000000000000000000001

0,0000000000000000000000000000000000000000000000000001

1,0000000000000000000000000000000000000000000000000010

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53+1023}$$

0 10000110100 0000000000000000000000000000000000000000000000000000

$$1 = (-1)^0 \times 1,000...000 \times 2^{0+1023}$$

0 01111111111 0000000000000000000000000000000000000000000000000000

$$1 = 0,\underbrace{000...000}_{52bits}1 \times 2^{53}$$

1,0000000000000000000000000000000000000000000000000001

0,0000000000000000000000000000000000000000000000000001

1,0000000000000000000000000000000000000000000000000010

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53+1023}$$

0 10000110100 0000000000000000000000000000000000000000000000000000

$$1 = (-1)^0 \times 1,000...000 \times 2^{0+1023}$$

0 01111111111 0000000000000000000000000000000000000000000000000000

$$1 = 0,\underbrace{000...000}_{52bits}\ 1 \times 2^{53}$$

1,0000000000000000000000000000000000000000000000000000 |
0,0000000000000000000000000000000000000000000000000000 | 1

1,0000000000000000000000000000000000000000000000000000 | 1.

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53+1023}$$

```
0 10000110100 0000000000000000000000000000000000000000000000000000
```

$$1 = (-1)^0 \times 1,000...000 \times 2^{0+1023}$$

```
0 01111111111 0000000000000000000000000000000000000000000000000000
```

$$1 = 0,\underbrace{000...000}_{52bits}\ 1 \times 2^{53}$$

```
1,0000000000000000000000000000000000000000000000000000|
0,0000000000000000000000000000000000000000000000000000|1
----------------------------------------------------------
1,0000000000000000000000000000000000000000000000000000|1
```

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53+1023}$$

```
0 10000110100 0000000000000000000000000000000000000000000000000000
```

$$1 = (-1)^0 \times 1,000...000 \times 2^{0+1023}$$

```
0 01111111111 0000000000000000000000000000000000000000000000000000
```

$$1 = 0,\underbrace{000...000}_{52bits}\ 1 \times 2^{53}$$

```
1,0000000000000000000000000000000000000000000000000|
0,0000000000000000000000000000000000000000000000000|1
---------------------------------------------------
1,0000000000000000000000000000000000000000000000000|1
```

$$2^{53} = (-1)^0 \times 1,000...000 \times 2^{53+1023}$$

```
0 10000110100 0000000000000000000000000000000000000000000000000000
```

$$1 = (-1)^0 \times 1,000...000 \times 2^{0+1023}$$

```
0 01111111111 0000000000000000000000000000000000000000000000000000
```

$$1 = 0,\underbrace{000...000}_{52bits}\ 1 \times 2^{53}$$

```
1,0000000000000000000000000000000000000000000000000000|
0,0000000000000000000000000000000000000000000000000000|1
----------------------------------------------------------
1,0000000000000000000000000000000000000000000000000000|1
```

Unit in the Last Place

Mauvaise nouvelle : une addition entre deux nombres flottants peut donc, dans certains cas, créer une erreur.

Unit in the Last Place

Mauvaise nouvelle : une addition entre deux nombres flottants peut donc, dans certains cas, créer une erreur.

Bonne nouvelle : on peut récupérer cette erreur.

Unit in the Last Place

Mauvaise nouvelle : une addition entre deux nombres flottants peut donc, dans certains cas, créer une erreur.

Bonne nouvelle : on peut récupérer cette erreur.

Unit in the Last Place

Mauvaise nouvelle : une addition entre deux nombres flottants peut donc, dans certains cas, créer une erreur.

Bonne nouvelle : on peut récupérer cette erreur.

Théorème 6 (Fast2Sum - Dekker & Kahan)

On considère deux VF x et y tels que $|x| \geqslant |y|$ et l'algorithme suivant :

```
1  s  ← x ⊕ y
2  y_v ← s ⊖ x
3  d  ← y ⊖ y_v
4  Retourner (s, d)
```

Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = err(x \oplus y)$. De plus s et d ne se chevauchent pas.

- si x et y sont de même signe (OU si $|y| \leqslant \frac{|x|}{2}$), alors $\frac{x}{2} \leqslant s \leqslant 2x$ et on peut appliquer le lemme ;

Théorème 6 (Fast2Sum - Dekker & Kahan)

On considère deux VF x et y tels que $|x| \geqslant |y|$ et l'algorithme suivant :

```
1  s  ← x ⊕ y
2  y_v ← s ⊖ x
3  d  ← y ⊖ y_v
4  Retourner (s, d)
```

Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = err(x \oplus y)$. De plus s et d ne se chevauchent pas.

- si x et y sont de même signe OU si $|y| \leqslant \frac{|x|}{2}$: alors $\frac{x}{2} \leqslant s \leqslant 2x$ et on peut appliquer le lemme ;
- sinon : on a x et y de signes opposés ET $y \geqslant \frac{|x|}{2}$ alors si y est négatif $x/2 \leqslant -y \leqslant x$ et sinon $-x/2 \leqslant y \leqslant -x$. Dans les deux cas, d'après le lemme de Sterbenz, s est calculée exactement et alors $y_v = x$.

Théorème 6 (Fast2Sum - Dekker & Kahan)

On considère deux VF x et y tels que $|x| \geqslant |y|$ et l'algorithme suivant :

```
1  s   ← x ⊕ y
2  y_v ← s ⊖ x
3  d   ← y ⊖ y_v
4  Retourner (s, d)
```

Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = err(x \oplus y)$. De plus s et d ne se chevauchent pas.

- **si x et y sont de même signe OU si $|y| \leqslant \frac{|x|}{2}$** : alors $\frac{x}{2} \leqslant s \leqslant 2x$ et on peut appliquer le lemme ;
- **sinon** : on a x et y de signes opposés ET $y > \frac{|x|}{2}$ alors si y est négatif $x/2 < -y < x$ et sinon $-x/2 < y < -x$. Dans les deux cas, d'après le lemme de Sterbenz, s est calculée exactement et alors $y_v = y$.

Théorème 6 (Fast2Sum - Dekker & Kahan)

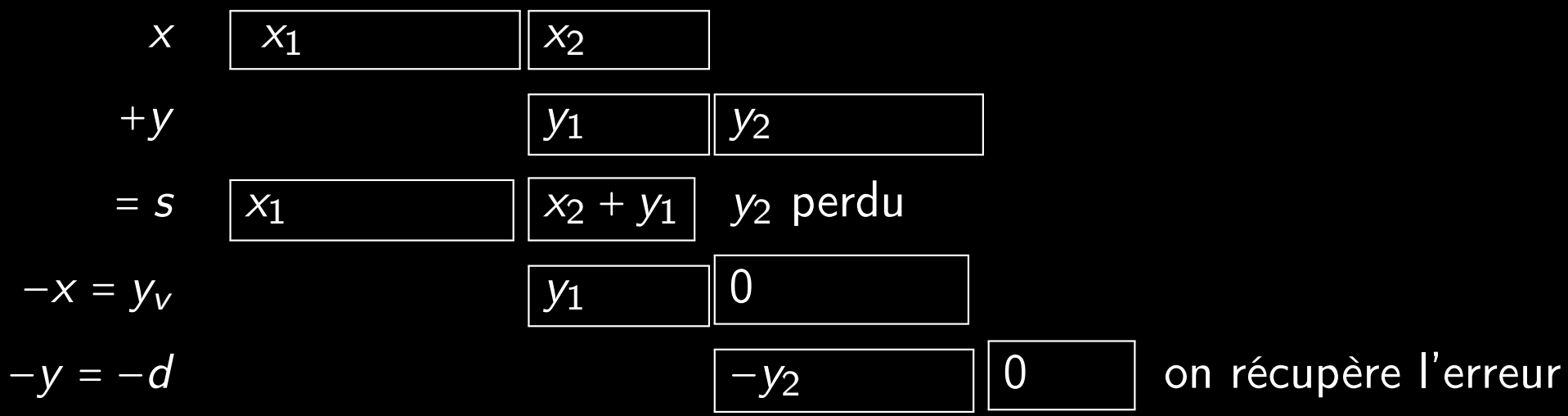
On considère deux VF x et y tels que $|x| \geqslant |y|$ et l'algorithme suivant :

```
1  s  ← x ⊕ y
2  y_v ← s ⊖ x
3  d  ← y ⊖ y_v
4  Retourner (s, d)
```

Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = err(x \oplus y)$. De plus s et d ne se chevauchent pas.

- **si x et y sont de même signe OU si $|y| \leqslant \frac{|x|}{2}$** : alors $\frac{x}{2} \leqslant s \leqslant 2x$ et on peut appliquer le lemme ;
- **sinon** : on a x et y de signes opposés ET $y > \frac{|x|}{2}$ alors si y est négatif $x/2 < -y < x$ et sinon $-x/2 < y < -x$. Dans les deux cas, d'après le lemme de Sterbenz, s est calculée exactement et alors $y_v = y$.

Somme compensée



```python
def fast2sum(x,y):
    if abs(x) >= abs(y):
        s  = x + y
        yv = s - x
        d  = y - yv
        return (s,d)
    else:
        return fast2sum(y,x)
```

```
In [100]: fast2sum(2.0**54,1.0)
Out[100]: (1.8014398509481984e+16, 1.0)
```

```
In [101]: fast2sum(2.0**54,2.0)
Out[101]: (1.8014398509481984e+16, 2.0)
```

```
In [102]: fast2sum(2.0**54,3.0)
Out[102]: (1.8014398509481988e+16, -1.0)
```

```
In [103]: fast2sum(2.0**54,4.0)
Out[103]: (1.8014398509481988e+16, 0.0)
```

```
In [104]: fast2sum(2.0**54,5.0)
Out[104]: (1.8014398509481988e+16, 1.0)
```

```
In [100]: fast2sum(2.0**54,1.0)
Out[100]: (1.8014398509481984e+16, 1.0)
```

```
In [101]: fast2sum(2.0**54,2.0)
Out[101]: (1.8014398509481984e+16, 2.0)
```

```
In [102]: fast2sum(2.0**54,3.0)
Out[102]: (1.8014398509481988e+16, -1.0)
```

```
In [103]: fast2sum(2.0**54,4.0)
Out[103]: (1.8014398509481988e+16, 0.0)
```

```
In [104]: fast2sum(2.0**54,5.0)
Out[104]: (1.8014398509481988e+16, 1.0)
```

```
In [100]: fast2sum(2.0**54,1.0)
Out[100]: (1.8014398509481984e+16, 1.0)
```

```
In [101]: fast2sum(2.0**54,2.0)
Out[101]: (1.8014398509481984e+16, 2.0)
```

```
In [102]: fast2sum(2.0**54,3.0)
Out[102]: (1.8014398509481988e+16, -1.0)
```

```
In [103]: fast2sum(2.0**54,4.0)
Out[103]: (1.8014398509481988e+16, 0.0)
```

```
In [104]: fast2sum(2.0**54,5.0)
Out[104]: (1.8014398509481988e+16, 1.0)
```

```
In [100]: fast2sum(2.0**54,1.0)
Out[100]: (1.8014398509481984e+16, 1.0)
```

```
In [101]: fast2sum(2.0**54,2.0)
Out[101]: (1.8014398509481984e+16, 2.0)
```

```
In [102]: fast2sum(2.0**54,3.0)
Out[102]: (1.8014398509481988e+16, -1.0)
```

```
In [103]: fast2sum(2.0**54,4.0)
Out[103]: (1.8014398509481988e+16, 0.0)
```

```
In [104]: fast2sum(2.0**54,5.0)
Out[104]: (1.8014398509481988e+16, 1.0)
```

```
In [100]: fast2sum(2.0**54,1.0)
Out[100]: (1.8014398509481984e+16, 1.0)
```

```
In [101]: fast2sum(2.0**54,2.0)
Out[101]: (1.8014398509481984e+16, 2.0)
```

```
In [102]: fast2sum(2.0**54,3.0)
Out[102]: (1.8014398509481988e+16, -1.0)
```

```
In [103]: fast2sum(2.0**54,4.0)
Out[103]: (1.8014398509481988e+16, 0.0)
```

```
In [104]: fast2sum(2.0**54,5.0)
Out[104]: (1.8014398509481988e+16, 1.0)
```

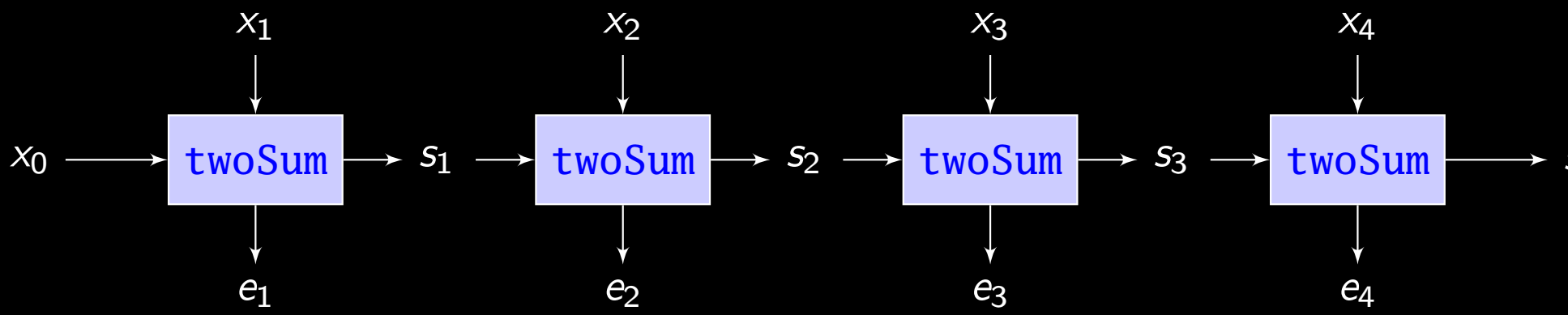
Théorème 7 (2Sum - Knuth)

On considère deux VF x et y et l'algorithme suivant :

1 $s \leftarrow x \oplus y$
2 $y_v \leftarrow s \ominus x$
3 $x_v \leftarrow s \ominus y_v$
4 $y_a \leftarrow y \ominus y_v$
5 $x_a \leftarrow x \ominus x_v$
6 $d \leftarrow x_a \oplus y_a$
7 **Retourner** (s, d)

Alors $x + y = s + d$ avec $s = x \oplus y$ et $d = err(x \oplus y)$. De plus s et d ne se chevauchent pas.

Somme compensée d'un nombre quelconque de flottants



Somme compensée d'un nombre quelconque de flottants

```
def sommeKahan(liste):
    (s,c) = (0.,0.)
    for x in liste:
        (s,c) = fast2sum(s, x + c)
    return s
```

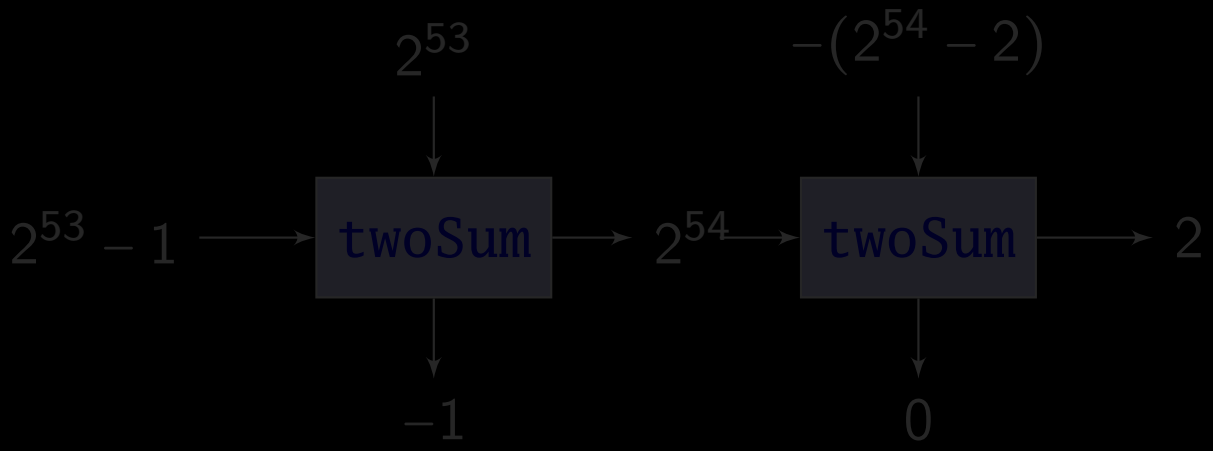
```
def sommePichat(liste):
    s,e = 0.,0.
    for x in liste:
        (s,c) = fast2sum(s, x)
        e += c
    return s + e
```

Somme compensée d'un nombre quelconque de flottants

```
def sommeKahan(liste):
    (s,c) = (0.,0.)
    for x in liste:
        (s,c) = fast2sum(s, x + c)
    return s
```

```
def sommePichat(liste):
    s,e = 0.,0.
    for x in liste:
        (s,c) = fast2sum(s, x)
        e += c
    return s + e
```

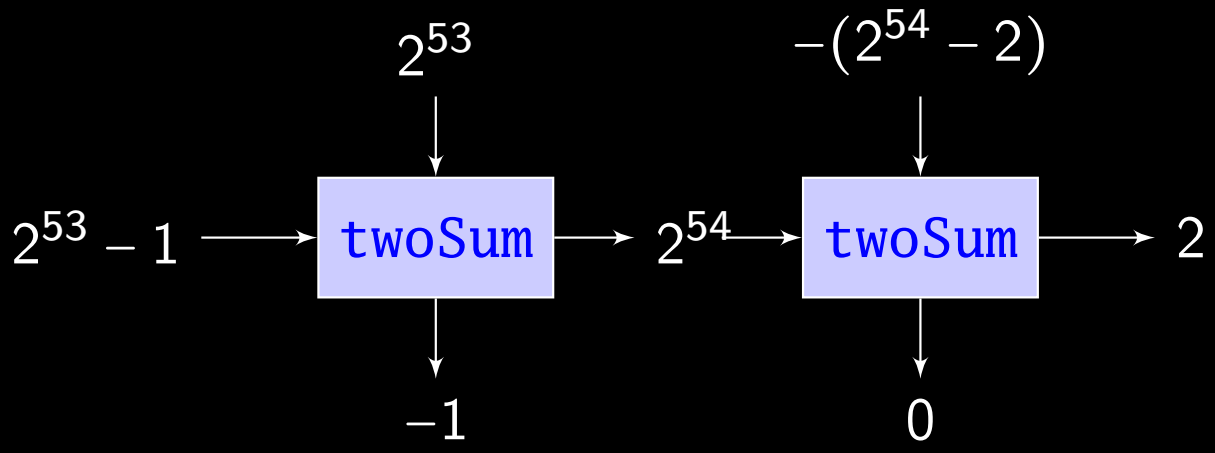
$x_0 = 2^{53} - 1$, $x_1 = 2^{53}$ et $x_2 = -(2^{54} - 2)$



Exercice 6

Expliquez les résultats trouvés. Que va faire l'algorithme de somme compensée ? Et celui de somme en cascade ?

$x_0 = 2^{53} - 1$, $x_1 = 2^{53}$ et $x_2 = -(2^{54} - 2)$



Exercice 6

Expliquez les résultats trouvés. Que va faire l'algorithme de somme compensée ? Et celui de somme en cascade ?

```
In [105]: sommePichat([1. / n for n in range(1,100001)])
Out[105]: 12.090146129863427

In [106]: sommeKahan([1. / n for n in range(1,100001)])
Out[106]: 12.090146129863427

In [107]: sum([1. / n for n in range(1,100001)])
Out[107]: 12.090146129863335

In [108]: sum([1. / n for n in range(100000,0,-1)])
Out[108]: 12.090146129863408
```

```
sage: sum(1. / n, n , 1 , 100000).n(64)
12.090146129863427
```

```
In [105]: sommePichat([1. / n for n in range(1,100001)])
Out[105]: 12.090146129863427

In [106]: sommeKahan([1. / n for n in range(1,100001)])
Out[106]: 12.090146129863427

In [107]: sum([1. / n for n in range(1,100001)])
Out[107]: 12.090146129863335

In [108]: sum([1. / n for n in range(100000,0,-1)])
Out[108]: 12.090146129863408
```

```
sage: sum(1. / n, n , 1 , 100000).n(64)
12.090146129863427
```

Banque chaotique

Exemple dû à Jean-Michel Muller.

M. X a récemment été à sa banque (Chaotic Bank Society), pour connaître les nouvelles offres proposées aux meilleurs clients. Son banquier lui propose l'offre suivante : « vous déposez tout d'abord e - 1 euros sur votre compte (où e = 2.7182818... est la base du logarithme népérien). La première année, nous prenons 1 euro sur votre compte de frais de gestion. Par contre, la deuxième année est plus intéressante pour vous, car nous multiplions votre capital restant par 2 et prenons 1 euro de frais de gestion. La troisième année est encore plus intéressante, car nous multiplions votre capital par 3 et prenons 1 euro de frais de gestion. Et ainsi de suite : la n-ième année, nous multiplions votre capital par n et prenons 1 euro de frais de gestion. Intéressant, non ? » Pour prendre sa décision, M. X décide de demander l'aide d'un informaticien et se retourne vers vous.

Banque chaotique

Exemple dû à Jean-Michel Muller.
M. X a récemment été à sa banque (Chaotic Bank Society), pour connaître les nouvelles offres proposées aux meilleurs clients. Son banquier lui propose l'offre suivante : « vous déposez tout d'abord e - 1 euros sur votre compte (où e = 2.7182818... est la base du logarithme népérien). La première année, nous prenons 1 euro sur votre compte de frais de gestion. Par contre, la deuxième année est plus intéressante pour vous, car nous multiplions votre capital restant par 2 et prenons 1 euro de frais de gestion. La troisième année est encore plus intéressante, car nous multiplions votre capital par 3 et prenons 1 euro de frais de gestion. Et ainsi de suite : la n-ième année, nous multiplions votre capital par n et prenons 1 euro de frais de gestion. Intéressant, non ? » Pour prendre sa décision, M. X décide de demander l'aide d'un informaticien et se retourne vers vous.

```python
def chaotic(n):
    cpt = exp(1) - 1
    for i in range(1,n + 1):
        cpt = i*cpt - 1
    return cpt
```

$$c_n = n! \times \left(c_0 - 1 - \frac{1}{2!} - \frac{1}{3!} - \cdots - \frac{1}{n!} \right)$$

$$= n! \times \left(c_0 - (e-1) + \frac{1}{(n+1)!} + \frac{1}{(n+2)!} + \cdots \right)$$

```python
def chaotic(n):
    cpt = exp(1) - 1
    for i in range(1,n + 1):
        cpt = i*cpt - 1
    return cpt
```

$$c_n = n! \times \left(c_0 - 1 - \frac{1}{2!} - \frac{1}{3!} - \cdots - \frac{1}{n!} \right)$$

$$= n! \times \left(c_0 - (e-1) + \frac{1}{(n+1)!} + \frac{1}{(n+2)!} + \cdots \right)$$