La précision numérique de calculs

Ou

Pourquoi il ne faut pas trop faire confiance à une machine

Sleipner A

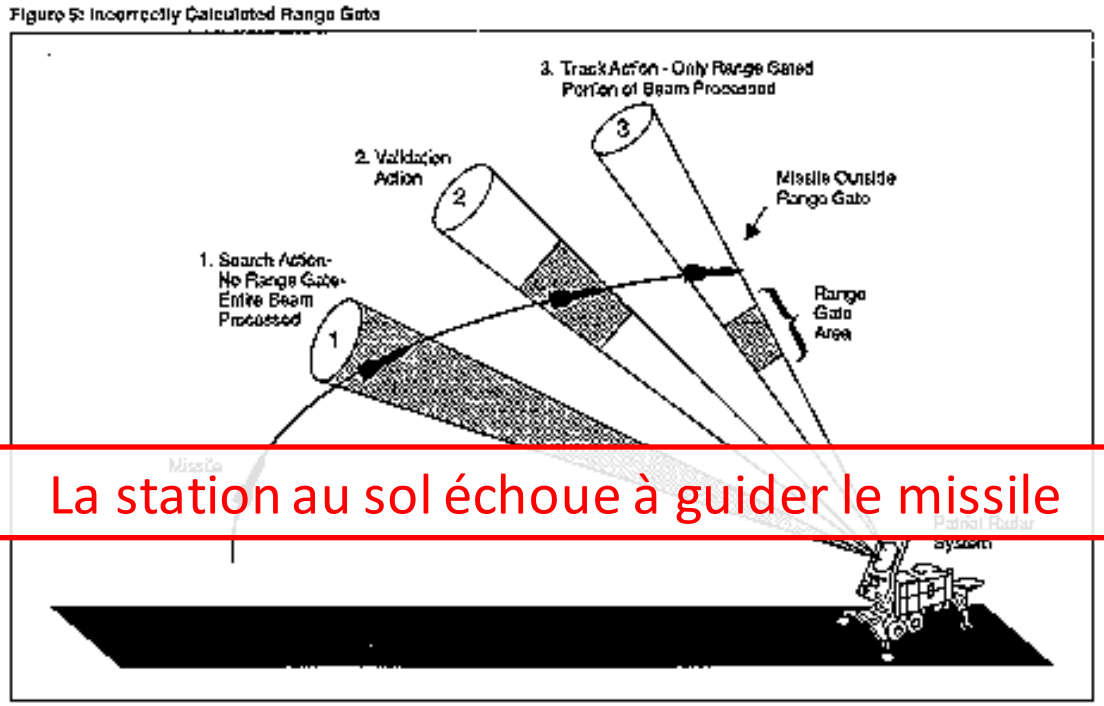
Patriot

Ariane 5

1991 : le Missile Patriot américain

- Première guerre du Golfe
- Missile pour contrer les SCUD irakiens
- 3 M$ par unité

1991 : le Missile Patriot américain



La station au sol échoue à guider le missile

- Première guerre du Golfe
- Missile pour contrer les SCUD irakiens
- 3 M$ par unité

- Un Patriot rate un SCUD
- 28 soldats tués
- 98 soldats blessés

Raison ?

- Un ordinateur ne sait pas vraiment calculer

Exemples

- x = 1 + 1 ?

Raison ?

- Un ordinateur ne sait pas vraiment calculer

Exemples

- x = 1 + 1 => $> 2
- x = 1 + 100000000 ?

Raison ?

- Un ordinateur ne sait pas vraiment calculer

Exemples

- x = 1 + 1 => $> 2
- x = 1 + 100000000 => $> 100000001

Raison ?

- Un ordinateur ne sait pas vraiment calculer

Exemples

- x = 1 + 1 => $> 2
- x = 1 + 100000000 => $> 100000001
- x = 1.0 + 100000000.0 ?

Raison ?

- Un ordinateur ne sait pas vraiment calculer

Exemples

- x = 1 + 1 => $> 2
- x = 1 + 100000000 => $> 100000001
- x = 1.0 + 100000000.0 => $> 1000000000.0

Un autre exemple

- Un incrément

```
float compteur  = 0.0f;
float increment = 1.0f;
for (int i = 0 ; i < 100000000 ; i++){
  compteur = compteur + increment;
}
print(compteur);
```

Un autre exemple

- Un incrément

```
float compteur  = 0.0f;
float increment = 1.0f;
for (int i = 0 ; i < 100000000 ; i++){
  compteur = compteur + increment;
}
print(compteur);
```

```
$> 16777216.0
```

16 Millions au lieu de 100 Millions !

Calculer ou ne pas calculer

- Le système de numération utilisé
- Machine finie vs. infinité de nombre

- Représentation binaire sur ordinateur

- Format = notation scientifique

$$s \times m \times 2^e$$

s = {+, -}

m = mantisse

e = exposant

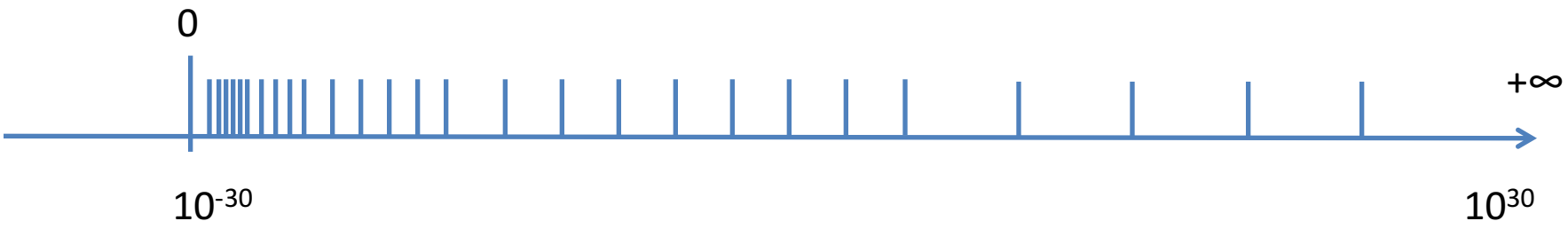
- Les nombres flottants
  - base 2
  - 32 bits (**float**), 64 bits (**double**)

64 bits		
signe	mantisse	exposant
1 bit	52 bits	11 bits

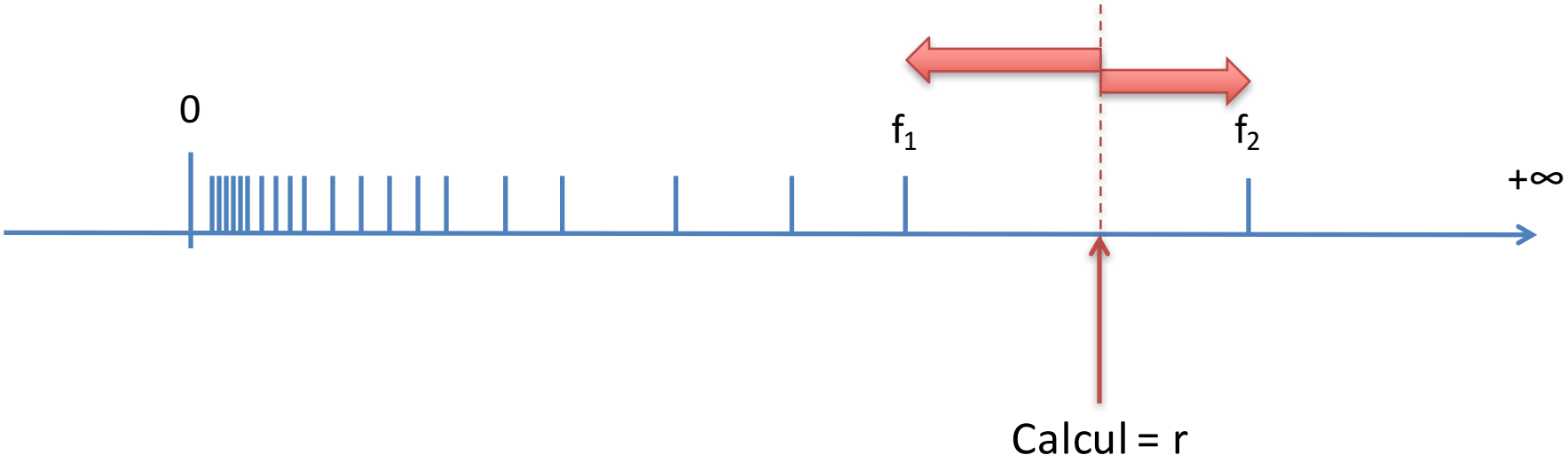
32 bits		
signe	mantisse	exposant
1 bit	23 bits	8 bits

- Idée sous-jacente
  - Infinité de nombre à représenter
  - Étaler les flottants sur l'ensemble du spectre

- Répartition
  - Très petit nombre, ex: 10^{-30}
  - Très grand nombre, ex: 10^{30}



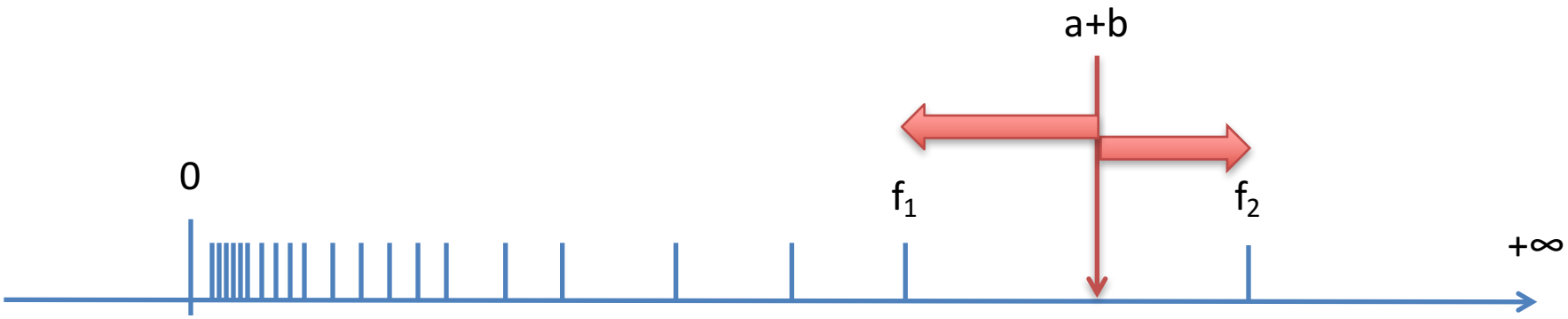
- Arrondis



Mode d'arrondi

- Plus proche — arrondi(r) = f_2
- Vers zéro — arrondi(r) = f_1
- Vers +∞ — arrondi(r) = f_2
- Vers -∞ — arrondi(r) = f_1

Notion d'arrondi correct

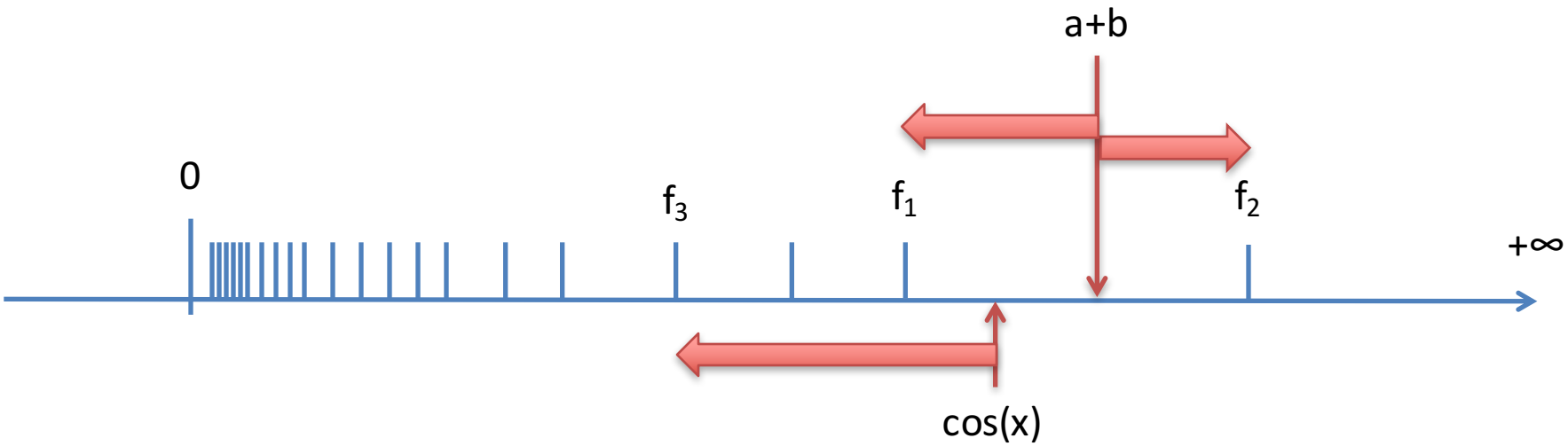


Si mon calcul est à l'arrondi correct alors arrondi(calcul) vaut soit f_1 soit f_2

Les opérateurs à arrondis correctes sont :

- +
- -
- ×
- ÷
- √

- Notion d'arrondi pas toujours correct



Si mon calcul est à l'arrondi correct alors arrondi(calcul) vaut soit f_1 soit f_2

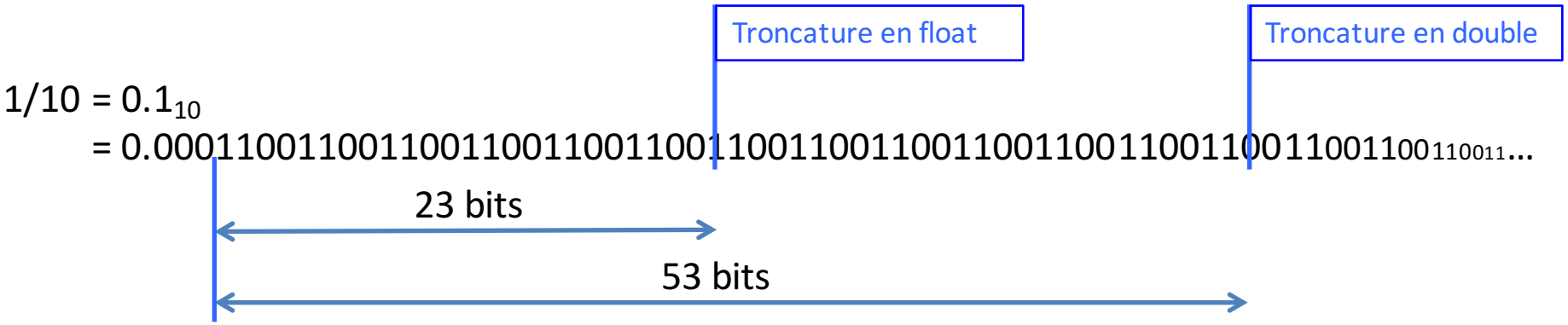
Les opérateurs à arrondis correctes sont :

- +
- -
- ×
- ÷
- √

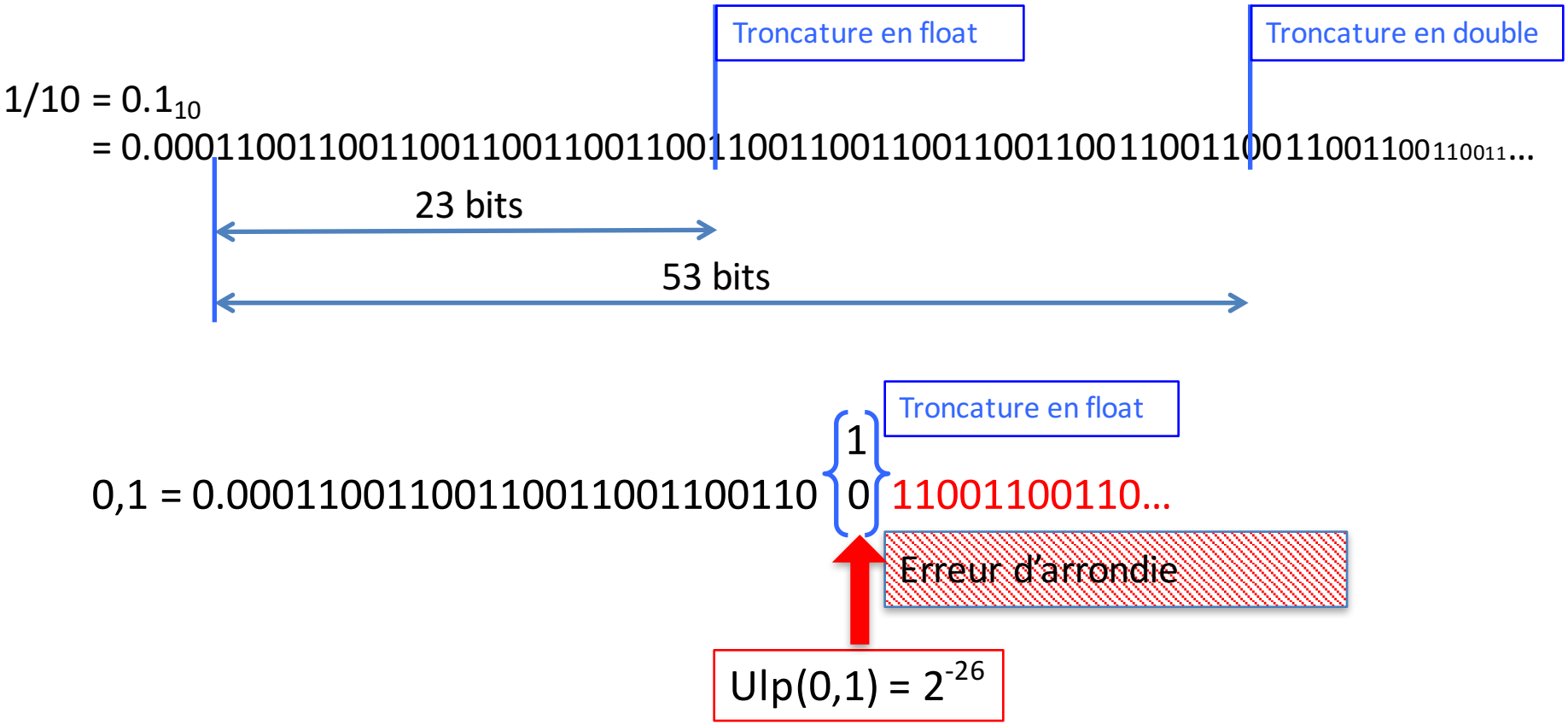
Ne sont **pas** arrondis correctement :

- cos
- sin
- tan
- log
- exp
- …

Exemples typiques d'arrondi qui tuent



Exemples typiques d'arrondi qui tuent

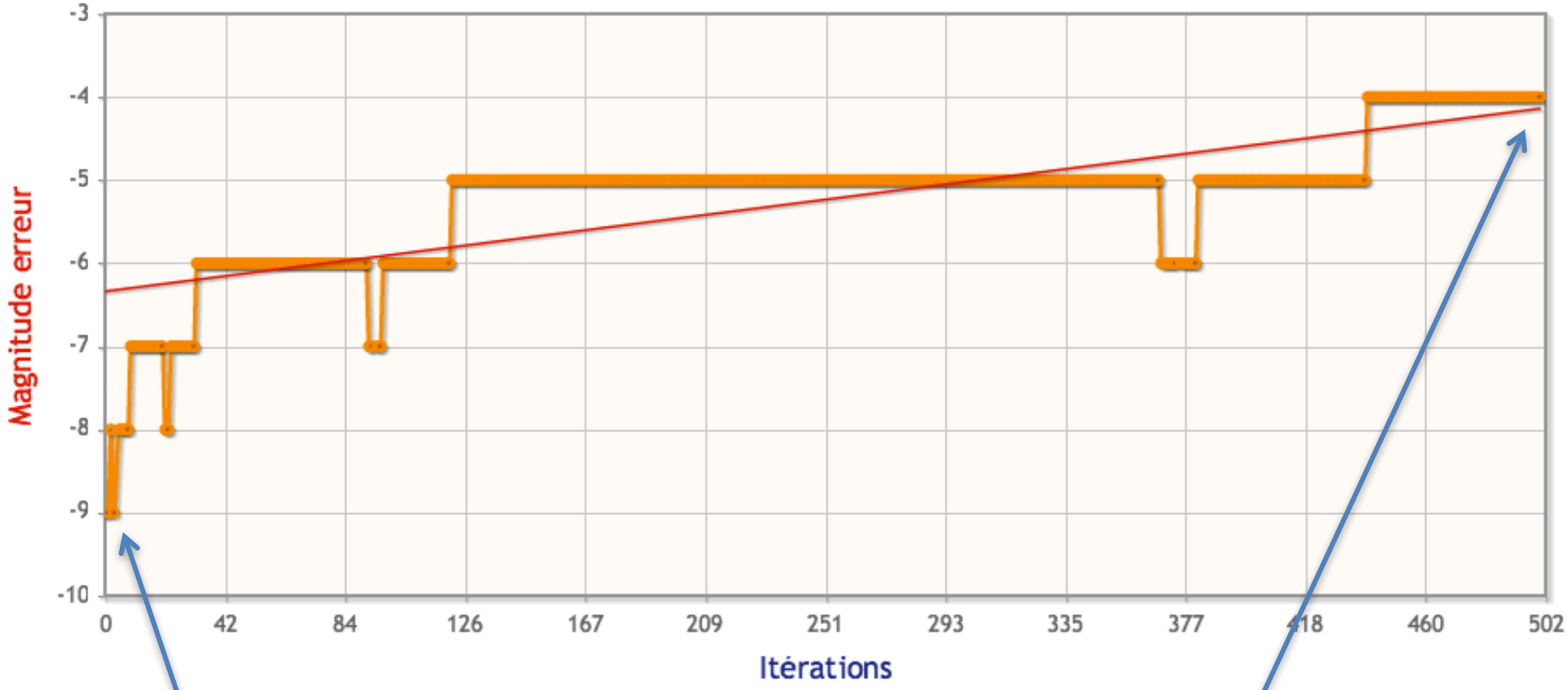


0,1 ≈ 0.1000000014901161193847656 25 en arrondi au plus près

Missile Patriot

Le missile a une horloge en 10ème de seconde
+ 0.1 en binaire c'est 0.10000000149011611938476562 5
+ 100h de fonctionnement où l'écart s'accumule
= 600 mètres de déviation

```
float compteur  = 0.0f;
float increment = 0.1f;
while (missile is on){
  compteur = compteur + increment;
}
```



0.1

0.1000000149011611938476562 5

0.00000000149011611938476562 5

50.1

50.0998077392578125

0.0001922607421875

Dérive du calcul au fil du temps

- Exemples typiques d'arrondi qui tuent
  - L'absorption

Pourquoi 1.0 + 100 000 000.0 ≠ 100 000 001.0 ?

- Exemples typiques d'arrondi qui tuent
  - L'absorption

Pourquoi 1.0 + 100 000 000.0 ≠ 100 000 001.0 ?

- Les petits et les grands s'additionnent mal

- Pourquoi ?

Et si j'utilise le type **double** ?

Mantisse est trop petite

- Exemples typiques d'arrondi qui tuent
  - L'annulation catastrophique

Comment deux calculs un peu faux font un résultat aléatoire

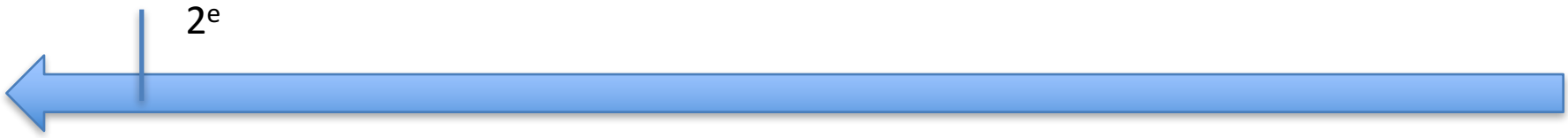
- Cas concret
  - 2 capteurs redondants
  - Calcul de la moyenne
  - Calcul de l'écart à la moyenne

- Valeurs bruités mais proches entre elles
  - Les valeurs (x_1 et x_2) sont entachés d'erreurs
  - Mais leurs valeurs sont proches

$$x_{moy} = (x_1 + x_2)/2 \approx x_1 \approx x_2$$

- Écart type
  - $e_1 = (x_{moy} - x_1)^2$
  - $e_2 = (x_{moy} - x_2)^2$
  - $\sigma = (e_1 - e_2) / 2$

- Pourquoi ce calcul risque de mal se terminer ?



1	,	0	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	?	?	?	?	?

x_{moy}

−

1	,	0	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	?	?	?	?	?

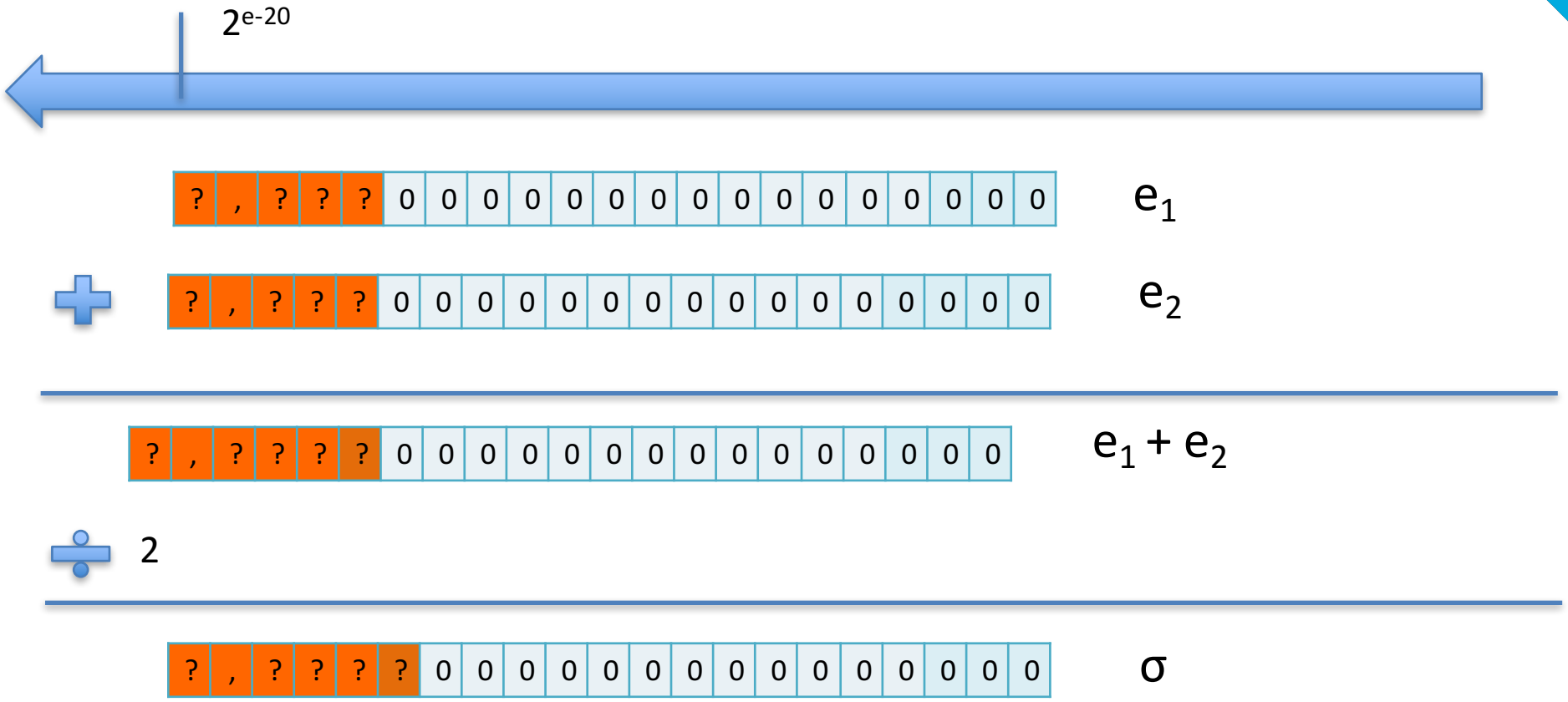
x_1

0	,	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	?	?	?	?	?

e_1

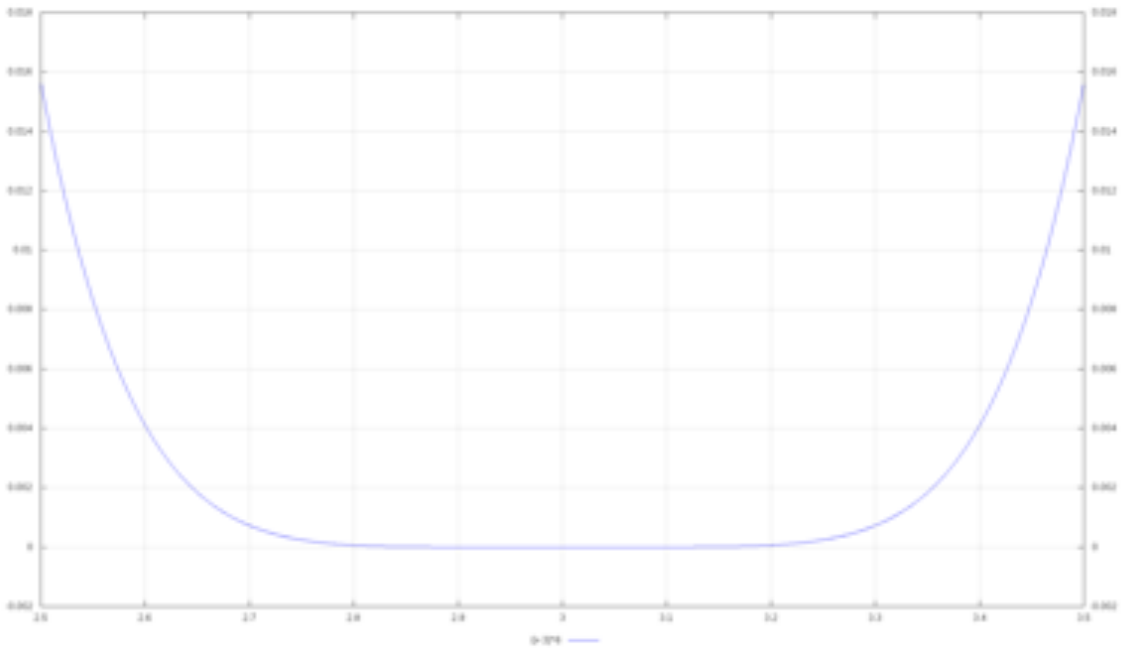
?	,	?	?	?	0	0	0	0	0	0	0	0	0	0	0	0	0	…

On a la même chose pour $\mathbf{e}_2$



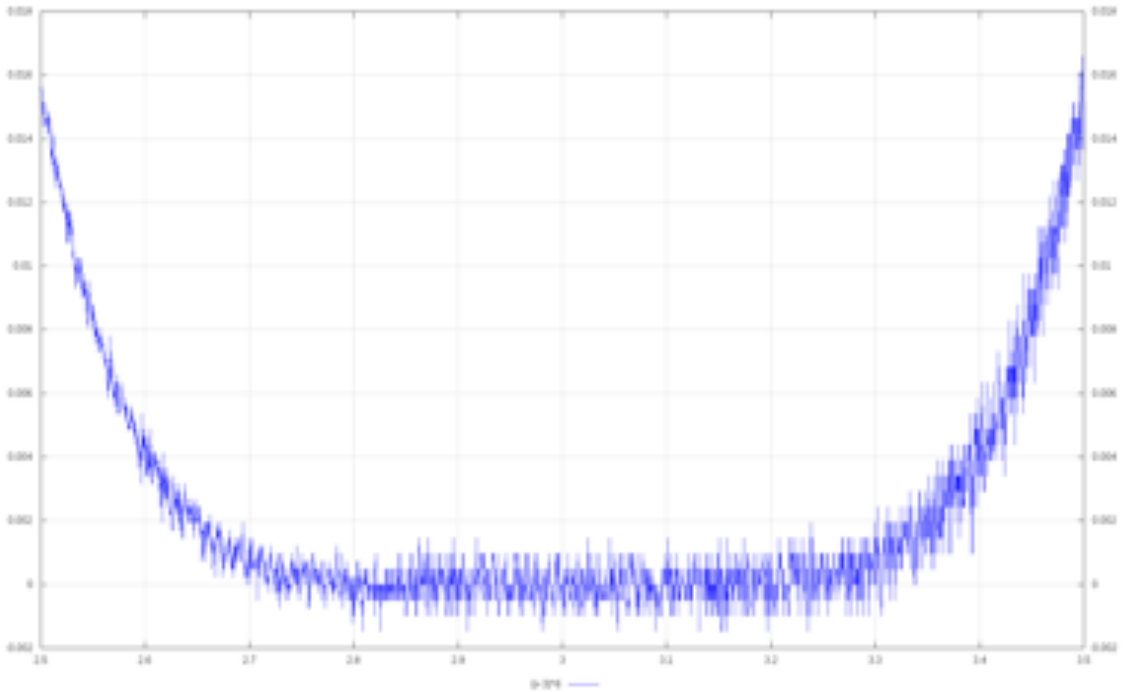
Et si je multiplie σ par un **très gros** coefficient ?

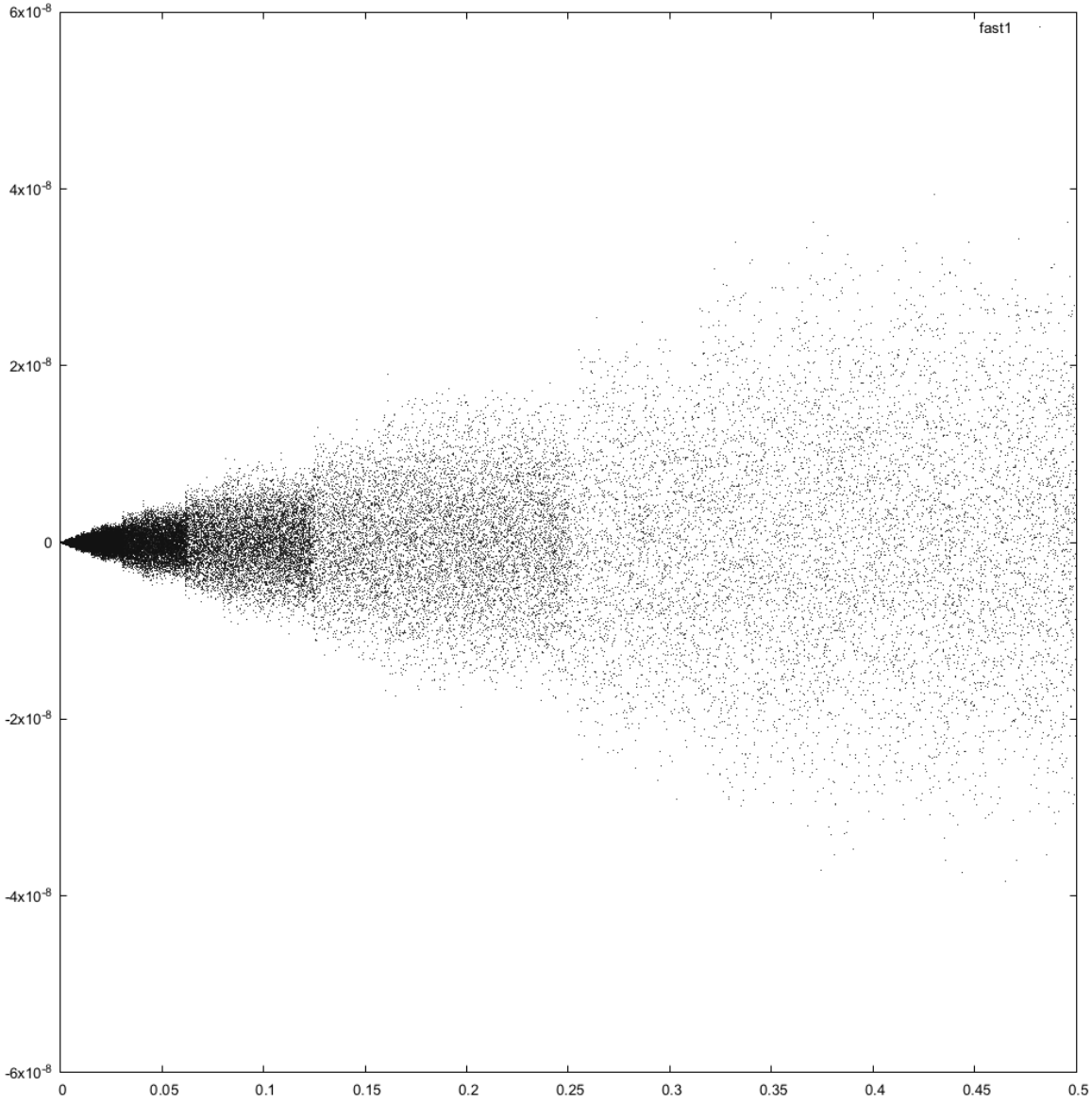
Réel



$F(x) = (x-3)^6$

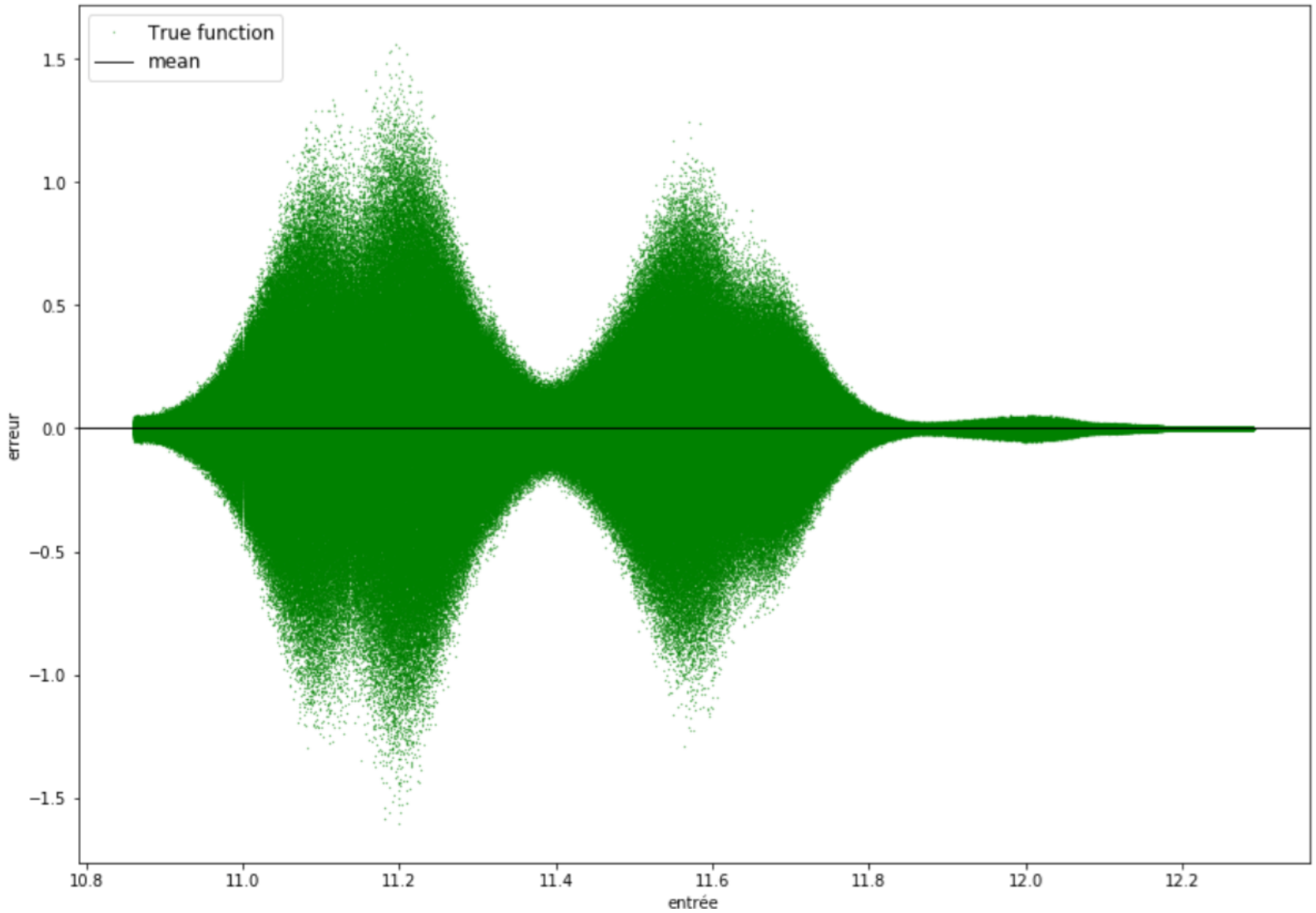
Flottant





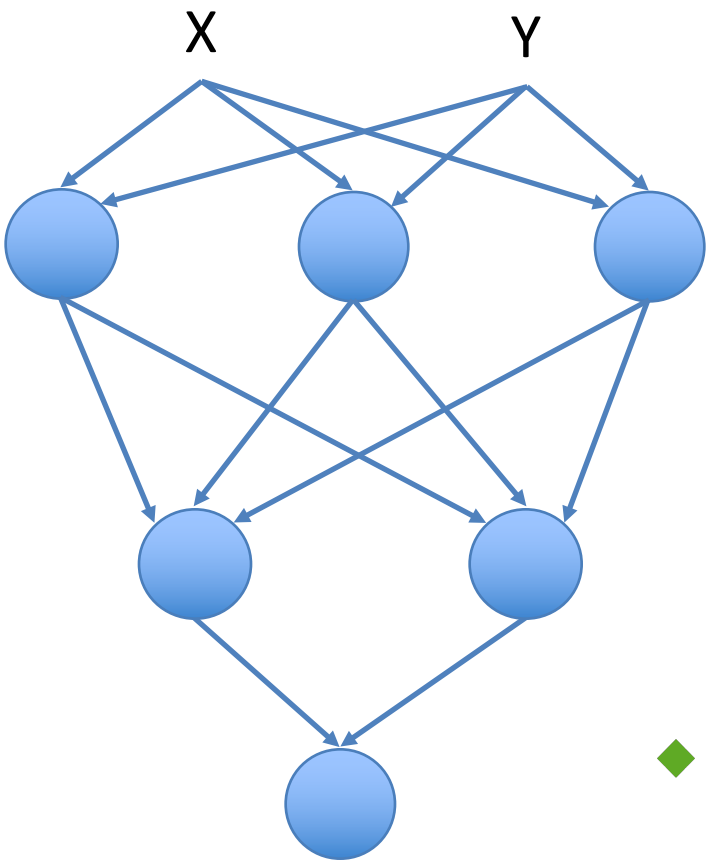
$$F(x) = \frac{\pi}{4}x + x \times (|x| - 1) \times (0.2447 + 0.0663\,|x|)$$

Erreur d'arrondi d'une fonction



$$f(x) = e^{\cos\left((x-5)^2\right)+\frac{\sin(x+3)\times x^2}{10}}$$

Erreur d'arrondi d'une fonction



$$\tanh(p_{13} * \tanh(p_7 * \tanh(p_1 * X + p_2 * Y) + p_8 * \tanh(p_3 * X + p_4 * Y) + p_9 * \tanh(p_5 * X + p_6 * Y)) + p_{14} * \tanh (p_{10} * \tanh(p_1 * X + p_2 * Y) + p_{11} * \tanh(p_3 * X + p_4 * Y) + p_{12} * \tanh(p_5 * X + p_6 * Y)))$$

- Architecte non connu (GPU, CPU, PFGA ? Mix ?)
- Précision non connue (16, 32, 64 bits ? Mix ?)
- Poids non connus (online learning ?)

- Architecture variable
- Librairie avec performances différentes
- Arrondis des valeurs dû aux calculs

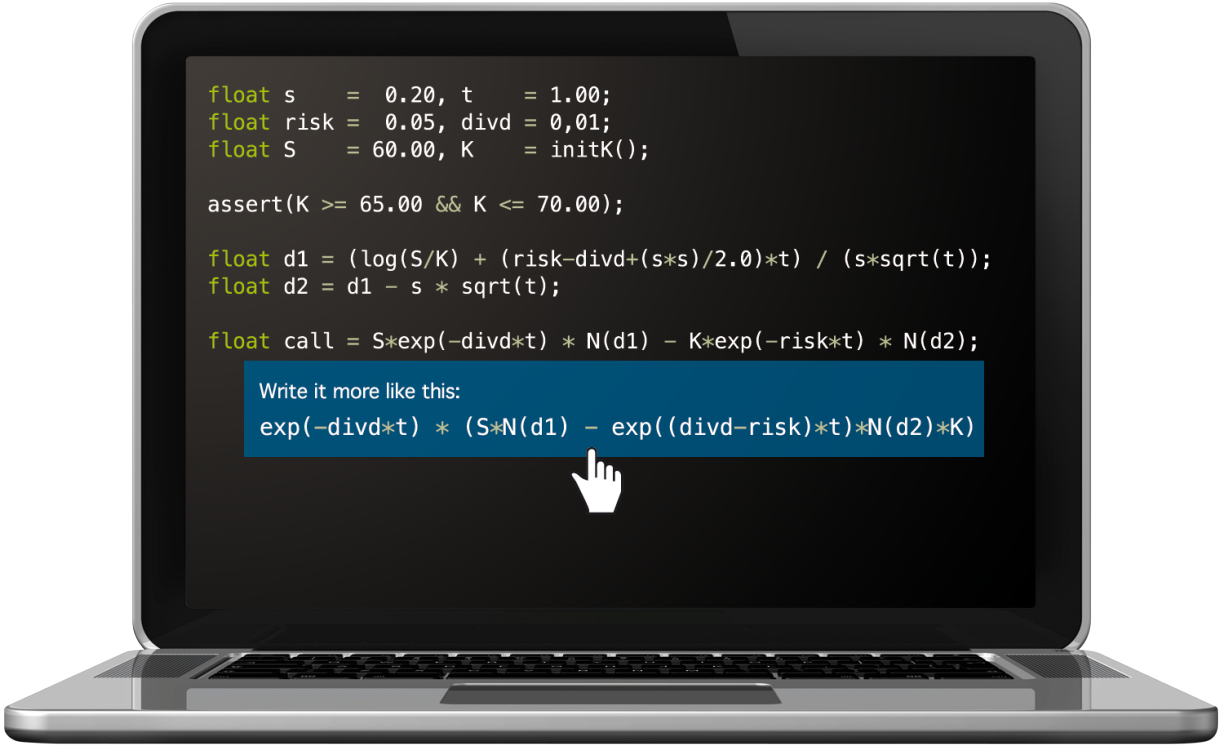
Reproduire des résultats n’est pas simple !

- L'industrie commence à regarder tout ça
  - Aéronautique, aérospatiale, énergie, défense
  - Finance
  - IA
- Notion de précision garantie
  - Type des variables et manière d'écrire le code
  - Entrées plus précises
- Equilibre entre
  - Performance (temps, consommation, occupation)
  - Précision (stabilité, durabilité)

Alors que faire ?

- Utiliser des types les plus grands ?
  - Perte importante de performances
- Faire des simulations ?
  - Nombre exponentiel de cas à traiter
  - Très peu de cas pathologiques
- Avis d'expert ?
  - Très rares
- Outils de preuves de programmes ?
  - Pas forcément très simples et user-friendly

numalis


float s = 0.20, t = 1.00;
float risk = 0.05, divd = 0,01;
float S = 60.00, K = initK();
assert(K >= 65.00 && K <= 70.00);
float d1 = (log(S/K) + (risk-divd+(s*s)/2.0)*t) / (s*sqrt(t));
float d2 = d1 - s * sqrt(t);
float call = S*exp(-divd*t) * N(d1) - K*exp(-risk*t) * N(d2);
Write it more like this:
exp(-divd*t) * (S*N(d1) - exp((divd-risk)*t)*N(d2)*K)