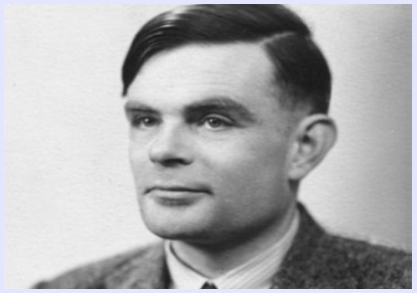
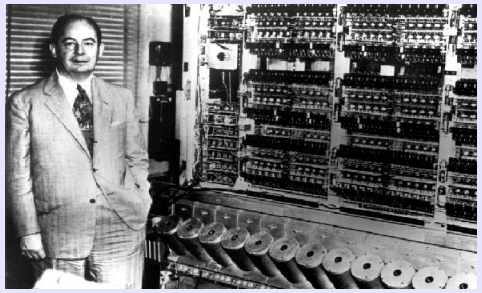




Information, Calcul et Communication

Théorie: Représentation de l'Information (1)

R. Boulic

Par quels moyens peut on représenter les nombres à virgule ?

Est il possible de construire une représentation exacte du monde réel ?

Plan

Lien avec les leçons précédentes
- Rappel des domaines d'applications
- Histoire des conventions de symboles
- Vers l'unité élémentaire d'information (exercices)

Manipulation sur les nombres entiers
- Opérations et domaine couvert

La virgule flottante: Pourquoi ? Comment ?
- Un exemple qui pose problème

Retour à la représentation des symboles
- De l’alphabet aux idéogrammes

Un exemple qui pose problème

La mise en oeuvre d'un algorithme doit aussi tenir compte du choix de représentation des nombres.

Exemple: $a.x^2 + b.x + c$
avec les valeurs décimales **a=0.25**, **b=0.1**, **c=0.01**

Discriminant $\Delta = b^2 - 4ac$

en théorie Δ est nul pour cet exemple

Mais $\Delta = 9.02\ 10^{-19}$ (représentation et calculs sur 64 bits)

Pourquoi ?

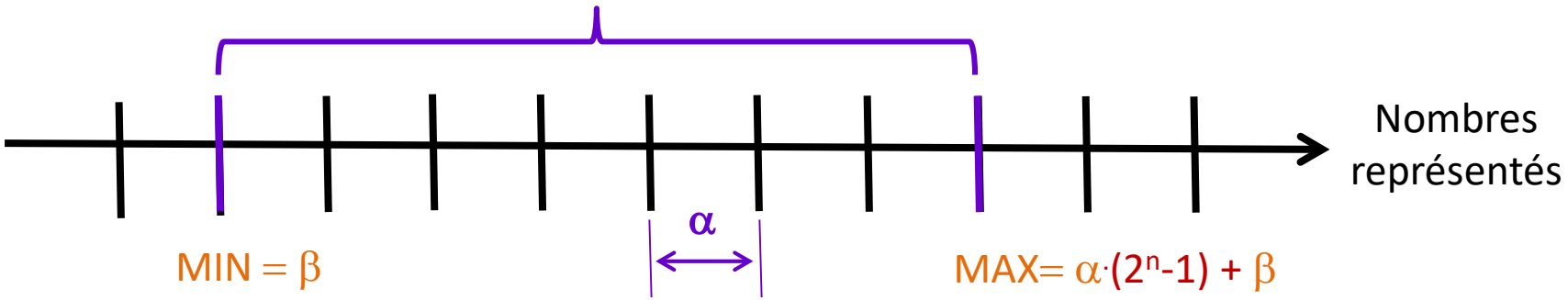
Représentation des nombres à virgule

Représentation à virgule fixe :

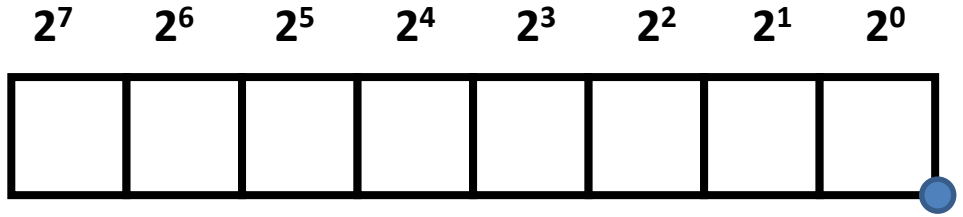
- le domaine couvert des entiers positifs avec n bits est [0, 2^n-1]
- ce domaine peut être mis à une échelle plus fine que l'unité (2^0) en multipliant le domaine couvert [0, 2^n-1] par un facteur $\alpha = 2^{-k}$
- si nécessaire, le domaine peut être décalé de β .

Ex: pour représenter la température du corps humain, on a besoin généralement d'une précision du *dixième de degré*, entre 35 et 43 degrés.

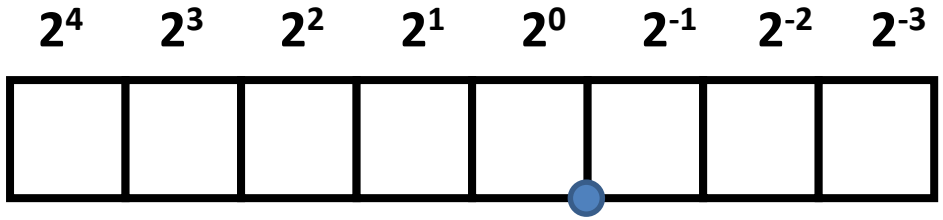
Pour n bits, les 2^n valeurs représentées sont **uniformément réparties** dans le nouvel intervalle [MIN,MAX] et séparée par la quantité α**.**



Exemple1 : avec $\alpha = 2^0$ et $\beta = 0$ le domaine est $[0, 2^8 - 2^0]$



Exemple2 : avec $\alpha = 2^{-3}$ et $\beta = 0$ le domaine est $[0, 2^5 - 2^{-3}]$

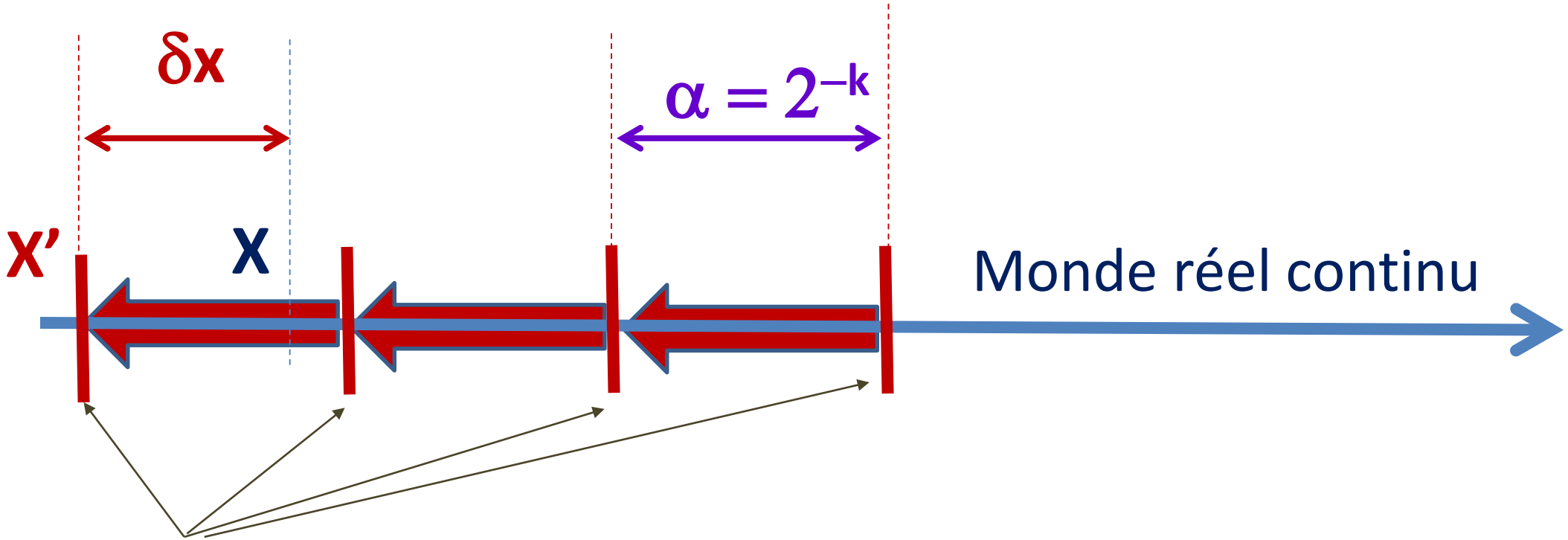


Notion d'erreur absolue : il existe un nombre infini de nombres à virgule à l'intérieur de [MIN, MAX] qui sont au mieux approchées par l'une des 2^n valeurs représentées.

-> production d'une **erreur absolue** de *discrétisation* (ou de *quantification*) au maximum égale à α sur tout l'intervalle.

Définition: soit une valeur **x** qui est représentée (par *troncation*) avec la valeur **x'**.
on appelle **erreur absolue** δx la quantité: $\delta x = | x - x' |$.

L'approximation par troncation est très utilisée dans les systèmes de mesure



Valeurs représentées numériquement

L'erreur absolue $\delta x = | X - X' |$ vaut au maximum α

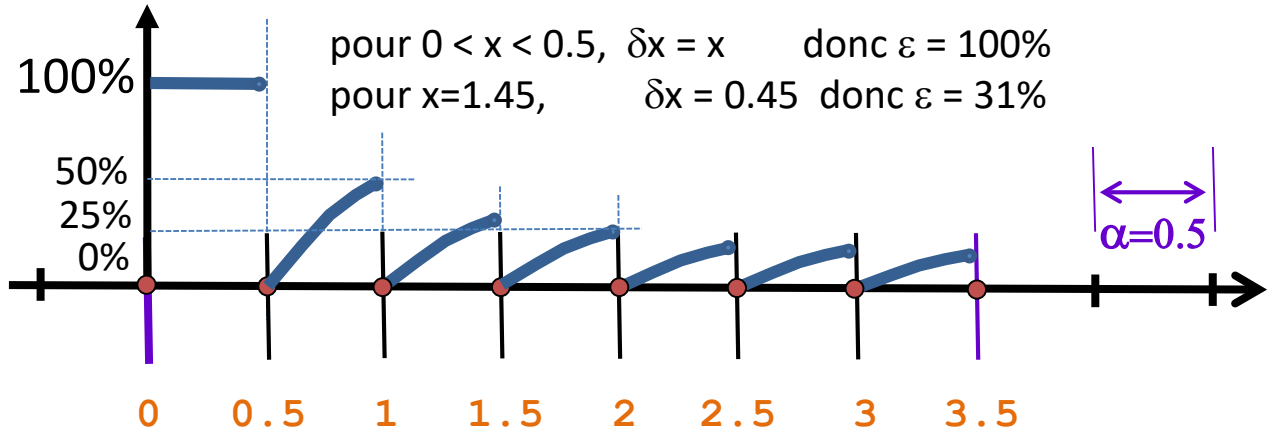
Erreur relative sur le domaine couvert

L'erreur relative décrit l'importance relative de **l'erreur absolue δx** par rapport au nombre à représenter **x** :

|δx|/|x| n'est pas uniforme sur le domaine couvert.

Exemple: avec 3 bits et α =0.5, le domaine couvert est [0, 3.5]. Seules 8 valeurs sont représentées exactement: 0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5 .
L'erreur relative (en %) est importante lorsque **δx** est grand par rapport à **x**.

Erreur relative en %
pour une approximation
par troncation



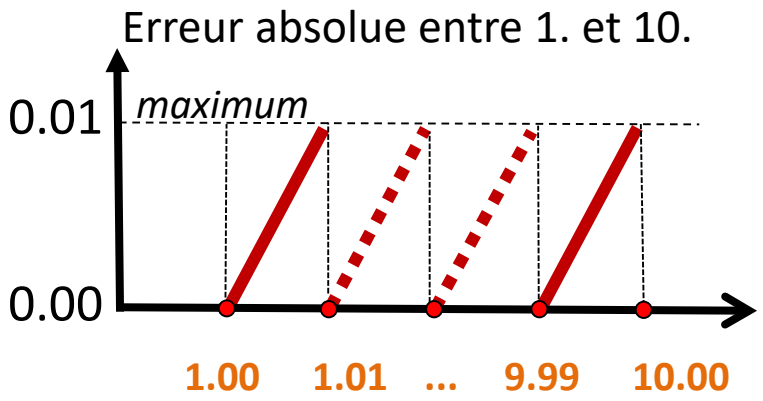
Cette répartition hétérogène de l'erreur relative n'est pas acceptable pour de nombreux problèmes

Erreur relative uniforme : comment ?

Inspiration: notation scientifique en base 10
avec un nombre fixe de chiffres significatifs.

Exemples avec 3 chiffres significatifs :

3.1415 s'écrit $3.14\ 10^{0}$
0.0125 s'écrit $1.25\ 10^{-2}$
7354 s'écrit $7.35\ 10^{3}$

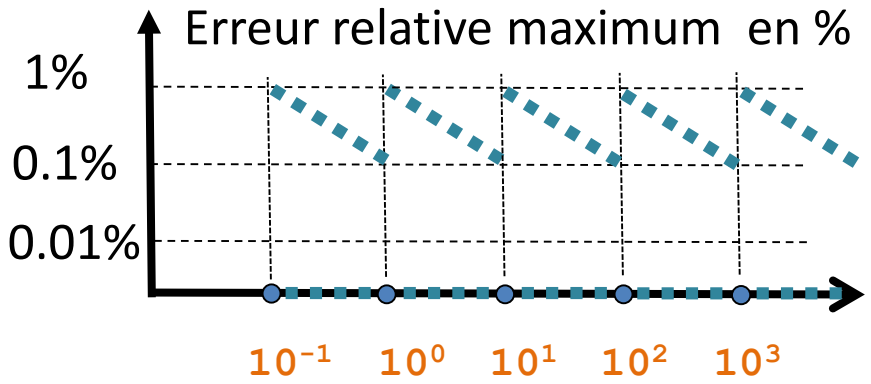


Erreur relative uniforme : comment ?

Inspiration: notation scientifique en base 10
avec un nombre fixe de chiffres significatifs.

Exemples avec 3 chiffres significatifs :

l'erreur relative est (presque) uniforme.
le *pire des cas* est en début d'intervalle
$[10^k, 10^{k+1}[$ lorsque $\delta x = 0.01 * 10^k$
pour $x = 1.00999..* 10^k$
ce qui donne **l'erreur relative** : $\delta x/x \approx 0.01 = 1\%$



La **précision** ε de cette représentation est son poids faible = 1% = plus petit majorant de **l'erreur relative**

La représentation en virgule flottante

= une notation scientifique en base 2

Représentation flottante en base 2: comporte 3 parties qui se partagent le nombre de bits à disposition: le **signe**, **l'exposant** de la base 2 et le **nombre normalisé** en base 2. La partie fractionnaire du nombre normalisé est appelée la **mantisse**.

Particularité de la base 2: le chiffre le plus significatif du nombre normalisé est constant et toujours égal à 1. Il est donc implicite.

$$\textbf{signe} \cdot 2^{\textbf{exposant}} \cdot 1,\textbf{mantisse}$$

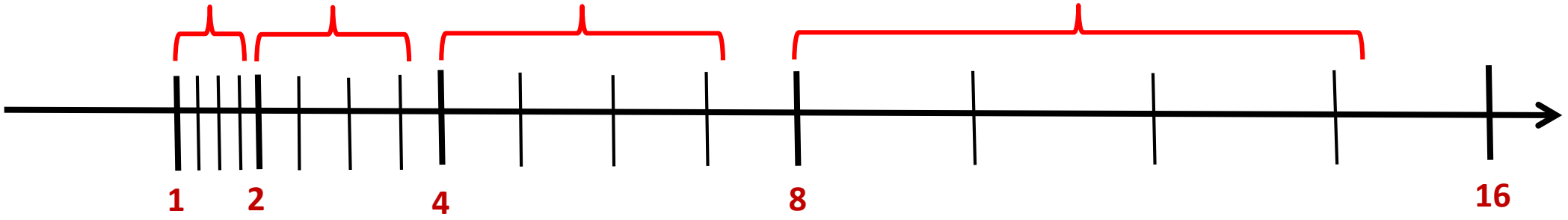
Comme pour la notation scientifique, la **précision ε** est donnée par :
=> la plus petite valeur *majorante* de **l'erreur relative**
=> que l'on approche avec la **puissance la plus faible de la mantisse**

La représentation en virgule flottante

Exemple avec 2 bits pour l'exposant et 2 bits pour la mantisse. On a la forme: $2^{\square\square} \cdot 1,\square\square$. L'erreur relative est majorée par $\mathbf{2^{-2}} = ¼$.

Forme normalisée

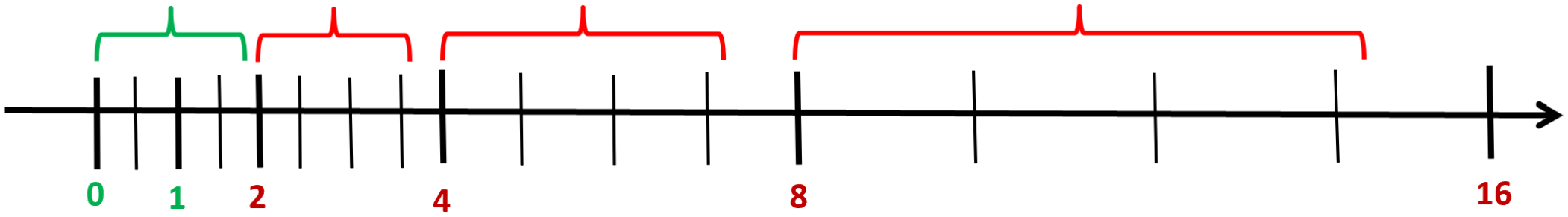
exp :	00	01	10	11
mantisse	00 01 10 11	00 01 10 11	00 01 10 11	00 01 10 11



La représentation de 0 en virgule flottante

Pour **inclure zéro** on traite ***la plus petite puissance de la base***, notée ***P***, **comme un cas particulier** qui permet de couvrir l'intervalle **[0, 2^{P+1}[**
avec la formule: 2^{P+1} . **0**,□□ *Forme dénormalisée*

exp :	**00**	01	10	11
mantisse	**00**	00	00	00
	01	01	01	01
	10	10	10	10
	11	11	11	11



*Pour les puissances de la base supérieures à **P**, on utilise la forme normalisée (intervalles en rouge)*

L’erreur d’arrondi est la règle

On appelle aussi **erreur d’arrondi** l’écart entre un nombre **x** et sa représentation approchée en virgule flottante.

Exercice: quel est le motif binaire de « un dixième » ?

0.0001100110011001100110011001100110011001100...

« un dixième » n’est pas représenté de manière exacte avec un nombre fini de bits

Remarques:

- ce problème existe dans toutes les bases, pensez à 1/3 en base 10 (définition des nombres rationnels)

- Même si l'erreur d'arrondi est inévitable pour la majorité des nombres, on peut garantir qu'elle se situe en dessous d'un seuil défini par les besoins du problème traité, en choisissant une représentation adaptée.

Un exemple qui pose problème (le retour)

Exemple: $a.x^2 + b.x + c$
avec les valeurs décimales **a=0.25**, **b=0.1**, **c=0.01**

Discriminant $\Delta = b^2 - 4ac$

Mais Δ = **$9.02\ 10^{-19}$** (représentation et calculs sur 64 bits)

- **4** et **0.25** sont représentés exactement, leur produit donne **1**.
- Par contre, approximations faites sur les décimaux **0.1** et **0.01**
- Conséquence sur les calculs en binaire : **0.1** * **0.1** ≠ **0.01**

Conséquence des erreurs d'arrondi

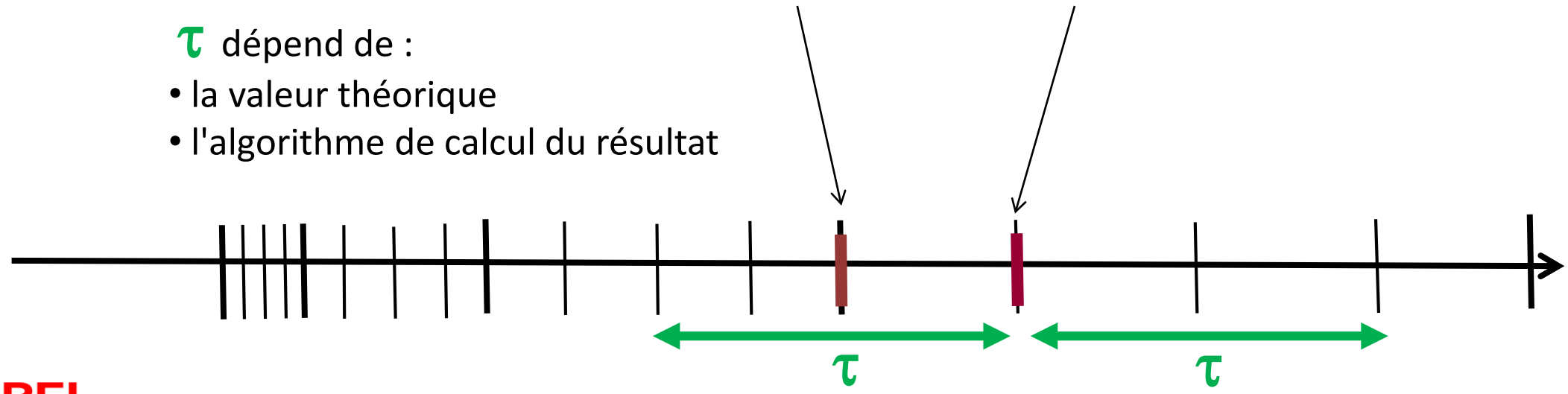
Tester l'EGALITE d'un résultat en virgule flottante vis-à-vis d'UNE SEULE valeur théorique **est une absurdité.** Le résultat est en général différent de la valeur théorique.

Le test d'égalité doit être redéfini à l'aide d'une tolérance variable τ AUTOUR de la valeur_théorique. Il y a *égalité* si le résultat appartient à cet intervalle:

$$| \text{résultat} - \text{valeur\_théorique} | < \tau$$

τ dépend de :

- la valeur théorique
- l'algorithme de calcul du résultat

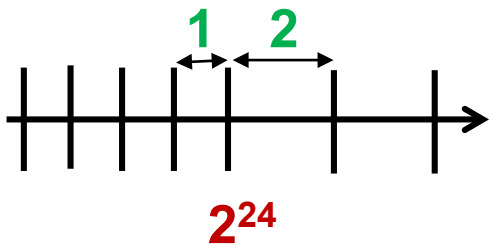


Conséquence des erreurs d'arrondi (2)

Le résultat est différent selon l'ordre des opérations. l'addition n'est plus associative:

Il existe des valeurs a, b, c telles que $(a + b) + c \neq a + (b + c)$

Exemple avec le standard IEEE 754 sur 32 bits possédant 23 bits de mantisse:



$(2^{24} + 1) + 1 \rightarrow 2^{24} + 1 \rightarrow 2^{24}$

$2^{24} + (1 + 1) = 2^{24} + 2$ qui est représenté

<u>Bonne pratique</u>: d'abord additionner les petits nombres entre eux avant de les additionner aux plus grands

La maîtrise de la précision est possible

Pour un problème donné et son algorithme de résolution, il est important de se poser les questions suivantes:

- De quelle précision ai-je besoin pour mes résultats ?
- Quelle est l'influence de l'algorithme sur la précision des résultats ?
- Quelle est la *précision maximum* disponible sur la machine cible ?

En cas de précision insuffisante, il faut reconsidérer l'algorithme de résolution et/ou adapter la représentation pour obtenir une *précision désirée.*

Compromis Précision / Coûts calcul et mémoire

Résumé

Est il possible de construire une représentation exacte du monde réel ?

Pour la **virgule flottante**, il faut d'abord se poser la question de la **précision** dont on a **besoin**. la représentation peut être adaptée pour garantir une précision désirée.

Parfois le **domaine couvert** est plus important que la précision comme dans le domaine des calculs avec les réseaux de neurones. C'est pourquoi il existe une réprésentation spécifique des nombres à virgule pour ce type de calculs (sera vue en série)

De plus il convient de distinguer la résolution mathématique d'un problème de sa mise en œuvre par le calcul numérique ; par exemple, *les erreurs causées par les approximations nous interdisent d'effectuer des tests d'égalité sur le résultat de calculs avec des nombres à virgule.*