Les calculs sur les flottants

Projet GL

Ensimag
Grenoble INP

22 décembre 2022

Pourquoi s'intéresser aux flottants ?

- Un objectif fort du projet GL est de comprendre comment les machines traduisent nos algorithmes
- Les flottants ne sont qu'un type, mais très utilisé dans toutes sortes d'applications
  - Codage (audio et vidéo), jeux vidéos, graphismes
  - Localisation, planification de trajets
  - Contrôle de processus, analyse de capteurs, logiciels embarqués,
  - Calcul scientifique, etc.
- Le calcul flottant est plein de difficultés et de bizarreries
- Besoin évident pour l'extension TRIGO, mais aussi TAB, OPTIM etc.

Pour le développement durable, il est important de réduire le calcul flottant qui représente un élément important de la consommation d'énergie

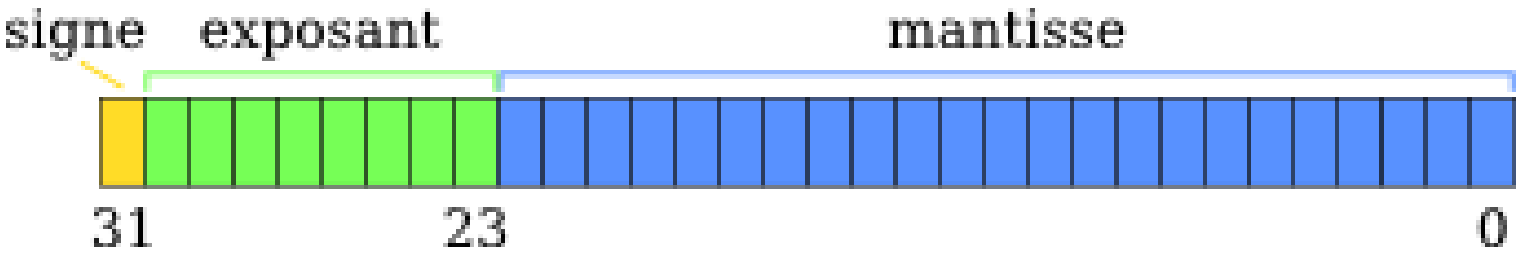
Un projet digne des Ensimag

Ensimag = Maths + Info

- Héritage de Jean Kuntzmann : l'informatique grenobloise fondée sur le calcul numérique
- Témoignage d'industriels : *"Un Ensimag, c'est un ingénieur qui sait que 1 plus 1 ne fait pas forcément 2"*
  - 1.1 + 1.1 n'est PAS égal à 2.2
- Différence infime ? Mais la conséquence peut être "catastrophique"
- Exemple de test incorrect :

```
float x, y = 1.1; float z = 2.2;
if (x + y == z) {
  ...
```

Les flottants Deca



- 1.0
  - Signe : $s = 0$
  - Exposant+127 : $e = 0111\ 1111$
  - Mantisse : $m = 00000000000000000000000$
  - $1.0 = (-1)^s \times 2^{e-127} \times (1 + m \times 2^{-23})$
- Précision sur 1.0 : dernier bit significatif
  $ulp(1.0) = 2^{-23} \approx 1.19209 \times 10^{-7}$
- Mais attention : le flottant juste en dessous de 1.0 vaut $x = 1 - 2^{-24}$
  $ulp(x) = 2^{-24}$

Répartition des flottants parmi les réels

- Tous les flottants sont des rationnels $\Rightarrow \pi$ n'est pas un flottant.
- >40% des flottants sont des entiers : tous ceux supérieurs (en valeur absolue) à 2^{23}
- >40% des flottants sont inférieurs (en valeur absolue) à 2^{-23}

- Il n'y a que 18% entre ces deux "extrêmes" 2^{-23} et 2^{23} ($\approx 8.4E6$)
- Densité très variable
  - Très forte autour de 0 (jusqu'à 2^{-149})
  - Et dans les grands entiers (eux-mêmes très espacés)

Importance de raisonner en binaire

- Seule la représentation binaire des flottants est exacte
  - Et l'affichage par ima en décimal est tronqué.
- La précision de vos résultats doit s'exprimer selon ulp
- ulp = Unit of Least Precision
  - La précision relative est 2^{-23}
  - ulp = précision absolue
- Exemples
  - $ulp(1) = 2^{-23}$
  - $ulp(2^{-23}) = 2^{-46}$
  - $ulp(10,000,000) = 1 \Rightarrow$ le flottant suivant est l'entier $10,000,001$
  - $ulp(2^{30}) = 128$: autour de 1 milliard, l'écart entre flottants successifs est de l'ordre de la centaine. Et on est encore très loin des plus grands flottants représentables (il en reste encore 38%).

Exemple $sin(\pi)$

- π n'est pas rationnel, donc n'est pas un flottant. Or vos fonctions trigonométriques ne peuvent être appliquées qu'à des arguments flottants.
- Soit
  - π^- le flottant immédiatement inférieur à π.
  - $\Delta = \pi - \pi^-$.
- On a
  - $\Delta < 2^{-22} = ulp(\pi^-)$ (car $2 < \pi < 4$ donc exposant 1)
  - $sin(\pi^-) < 2^{-22}$ (car $sin(\pi - \Delta) = sin(\Delta) \leq \Delta$)
  - Votre calcul devrait être précis :
    - ★ à 2^{-23} en valeur RELATIVE si possible,
    - ★ donc à 2^{-45} (puisque $2^{-23}\Delta < 2^{-45}$).

Consignes pour l'extension TRIGO

- Votre bibliothèque sera testée séparément par nos tests : ne vous basez pas sur des particularités de votre compilateur.
- *Vos tests* ne doivent pas utiliser de particularités de votre bibliothèque, car ils seront testés avec une bibliothèque standard.
- Respectez bien les domaines et codomaines des fonctions trigonométriques.
- Documentation attendue pour la classe Math : 10 à 20 pages décrivant vos *choix d'algorithmes*, analysant leur *précision* (cf vos cours de Méthodes Numériques), décrivant l'implantation et la validation.

Recommandations

Extension :

- $\approx$ 20 à 25% de l'effort du projet... Et 20% de la note :
  - $\Rightarrow$ presqu'autant que le compilateur (25%) et que les tests (22,5%)
- TRIGO : Spécifiée, mais pas guidée ; analyse et conception à faire intégralement.

Organisation

- Commencer dès le début du projet.
- Choix d'algorithmes adaptés au contexte (flottants 32 bits, capacités IMA...)
- Gros travail en phase amont : algorithmes, analyse de convergence et précision etc. Vous ne pouvez pas attendre que votre compilateur compile du code objet (rush de la dernière semaine).
- Impliquer plusieurs étudiants (comme la plupart des autres tâches) : validation croisée etc.