Projet 1
Limites de la représentation des nombres flottants

Groupe 4 - Equipe 3

Responsable : Mélissa Colin
Secrétaire : Mohamed Aziz Migaou
Codeurs : Younes Bamouss, Anna Gouedard

Résumé : Ce travail étudie les limites de la représentation des nombres flottants en machine, en mettant en évidence les erreurs d'arrondi et les imprécisions associées. Nous analysons les erreurs relatives dans les opérations arithmétiques fondamentales et implémentons des algorithmes numériques pour le calcul du logarithme népérien et des fonctions trigonométriques en utilisant la méthode CORDIC. Les résultats montrent l'importance de comprendre ces erreurs pour garantir la fiabilité des calculs en informatique.

1 Introduction

1.1 Contexte

Dans les ordinateurs, les nombres réels ne peuvent pas être représentés avec une précision infinie. Ils sont stockés sous forme de nombres *flottants*, ce qui peut entraîner des erreurs d'arrondi et des imprécisions. La norme IEEE 754 définit les formats et méthodes pour l'arithmétique des nombres à virgule flottante en environnement de programmation informatique [1]. La Figure 1 illustre la représentation des nombres flottants selon cette norme.

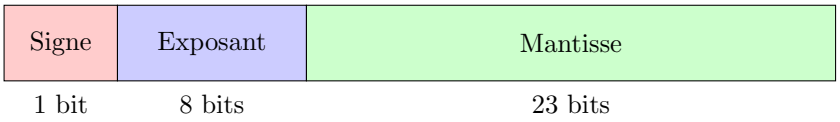

Figure 1: Schéma de la représentation des nombres flottants (32 bits) selon la norme IEEE 754.

1.2 Motivation

L'étude de la représentation des nombres et des erreurs qu'elle engendre est essentielle pour garantir la fiabilité des calculs en informatique. Dans de nombreux domaines comme la simulation numérique, le calcul haute performances ou encore l'intelligence artificielle, il est crucial de comprendre comment les erreurs de représentation peuvent affecter les résultats.
Ce projet vise donc à anticiper les limites des nombres flottants et à fournir des outils pour analyser ces erreurs.

2 Méthodologie

Nous étudions dans un premier temps la représentation décimale réduite des nombres ainsi que l'impact de ses arrondis sur les opérations arithmétiques élémentaires. Puis afin de mieux comprendre les erreurs d'arrondi et de précision, nous avons implémenté des algorithmes numériques pour le calcul du logarithme népérien et des fonctions trigonométriques.

2.1 Représentation décimale réduite

Afin d'étudier cette représentation, nous avons réalisé une fonction $rp(x, p)$ en python qui prend un nombre réel x et le représente avec p chiffres significatifs.
Pour ce faire, nous avons choisi d'utiliser le module *decimal* en définissant la précision avec `getcontext().prec = p`.

Nous avons en suite simulée l'addition et la multiplication avec ses représentations réduites en calculant

l'erreur relative pour chaque opération avec les formules (1) et (2).

$$\delta_s(x,y) = \frac{|(x+y) - (x_m +_m y_m)|}{|x+y|} \qquad (1) \qquad \delta_p(x,y) = \frac{|(x \times y) - (x_m \times_m y_m)|}{|x \times y|} \qquad (2)$$

Si on note respectivement x et y les nombres réels et x_m et y_m leurs représentations machine ainsi que $+_m$ et $\times_m$ les opératio ns machine.

Pour visualiser l'impact de la précision sur les opérations arithmétiques, nous avons représenté les erreurs relatives des opérations d'addition et de multiplication en fixant $x = 1$ et en faisant varier y en puissance de 10 négatives. L'objectif étant de faire croître la différence d'échelle entre x et y.

2.2 Algorithmes Numériques

Pour éprouver ses représentations, nous proposons deux algorithmes pour le calcul : l'un utilisant la série entière pour calculer log(2), et l'autre utilisant l'algorithme CORDIC sur quatre fonctions ln, exp, tan et arctan.

2.2.1 Série entière

Nous tentons de calculer une valeur approchée de log(2) en utilisant une série alternée définie par l'équation 3.

$$S_n = \sum_{k=1}^{n} \frac{(-1)^{k+1}}{k} \qquad (3)$$

Cette série est connue pour converger vers log(2) lorsque $n \to \infty$ [2].

On utilise donc cette approche pour calculer log(2) à p décimales: nous calculons la somme partielle S_n de la série jusqu'à ce que le terme suivant $T_{n+1} = \frac{(-1)^{n+2}}{n+1}$ soit inférieur à la précision souhaitée. C'est à dire tant que $|T_n| = \frac{1}{n} > \epsilon, \quad \text{où} \quad \epsilon = 10^{-p-2}$.

2.2.2 Algorithme CORDIC

L'algorithme CORDIC est une méthode de calcul utilisée principalement dans les calculatrices [4]. Celui-ci utilise pour un nombre flottant 8 octets de mémoire en utilisant une mantisse codée sur 4 chiffres binaires, un exposant signé ou non sous forme d'une puissance de 10 et du signe du nombre. [3]. Cette méthode permet d'effectuer efficacement diverses opérations mathématiques telles que les fonctions trigonométriques, logarithmiques, et exponentielles, en utilisant uniquement des opérations arithmétiques de base qui réduisent progressivement la valeur de x à une valeur très faible tout en élaborant le résultat, ce qui la rend particulièrement adaptée aux systèmes à ressources limitées. Chacune de ces transformations nécessite une valeur précalculée de la fonction f (ou de son inverse). Cet algorithme possède comme variant la valeur x, un réel qu'on souhaiterait calculer avec une précision d'environ 12 décimales. Dans ce travail, on l'évalue par les fonctions ln, tan et leur bijection réciproque.

Logarithme népérien et exponentielle
Pour calculer la valeur des fonctions exponentielle et logarithme naturel, nous avons écrit x sous les deux formes suivantes 4 et 5 pour les fonctions ln et exp respectivement.

$$x = \prod_{k=0}^{6} \left(1 + \frac{1}{10^k}\right)^{n_k} \cdot (1+\varepsilon) \qquad (4)$$

$$x = \sum_{k=0}^{6} n_k \ln\left(1 + \frac{1}{10^k}\right) + \varepsilon \qquad (5)$$

Avec $n_0, n_1, \ldots, n_6$ des coefficients maximaux et $\varepsilon \leq 10^{-7}$ (Cela est dû aux coefficients maximaux).

Le logarithme vérifie la propriété suivante $\forall x, y \in \mathbf{R}^{+*} \ln(xy) = \ln(x) + \ln(y)$. On a donc :

$$\ln(x) = \ln\left(\prod_{k=0}^{6}\left(1+\frac{1}{10^k}\right)^{n_k}\right) + \ln(1+\varepsilon) = \sum_{k=0}^{6} n_k \ln\left(1+\frac{1}{10^k}\right) + \ln(1+\varepsilon) = \sum_{k=0}^{6} n_k L[k] + \ln(1+\varepsilon) \quad (6)$$

Or $\varepsilon \leq 10^{-7}$, et $\ln(1+x) \sim x$ et $e^x \sim x+1$ quand $x \to 0$. donc :

$$\boxed{\ln(x) = \sum_{k=0}^{6} n_k L[k] + \varepsilon} \qquad \boxed{e^x = (1+\varepsilon)\prod_{k=0}^{6}\left(1+\frac{1}{10^k}\right)^{n_k}} \quad (7)$$

Tangente et Arctangente
Pour calculer les valeurs des fonctions trigonométriques tan et arctan, nous utilisons les relations suivantes. La relation pour arctan est donnée par l'équation 8 et pour tan par l'équation 10.

$$\arctan\left(\frac{x}{y}\right) = \arctan\left(\frac{x - y\cdot 10^{-k}}{y + x\cdot 10^{-k}}\right) + \arctan(10^{-k}) \quad (8)$$

Avec initialement $x = x$, $y = 1$, et $k \leq 4$. En appliquant la relation d'équivalence $\arctan(x) \sim x$ *lorsque* $x \to 0$*, on obtient l'équation 9.*

$$\arctan(x) = \sum_{k=0}^{4} n_k \cdot A[k] + \varepsilon \quad (9)$$

où $0 \leq \varepsilon \leq 10^{-4}$.

$$\tan\left(s - \arctan(10^{-k})\right) = \frac{n}{d} \quad \Rightarrow \quad \tan(s) = \frac{n + d\cdot 10^{-k}}{d - n\cdot 10^{-k}} \quad (10)$$

car $\tan(a-b) = \frac{\tan a - \tan b}{1+\tan a \tan b}$*. En utilisant le fait que* tan *est impaire et* $\tan(x) \sim x$ *lorsque* $x \sim 0$*, on considère "*$0 \leq s$*" est un invariant de l'algorithme donc s est diminue (terminaison) dans:*

$$\tan(s) = \frac{n + d\cdot 10^{-k}}{d - n\cdot 10^{-k}} \quad (11)$$

3 Résultats et discussions

Dans cette section, nous présentons et analysons les résultats obtenus à partir des différentes méthodes et algorithmes implémentés.
Nous commencerons par discuter des résultats relatifs à la représentation décimale réduite, en mettant en évidence les erreurs relatives observées lors des opérations d'addition et de multiplication. Ensuite, nous analyserons les performances des algorithmes numériques, en évaluant leur précision et leur efficacité pour les calculs de fonctions mathématiques spécifiques.

3.1 Représentation décimale réduite

En traçant le graphique, visible en figure 2, on observe d'importantes différences entre l'addition et la multiplication.

- L'erreur relative dans l'addition augmente progressivement avec y. Pour des valeurs très petites de y (ex. $y < 10^{-20}$), l'erreur est négligeable car y est trop petit pour influencer la somme. Cependant, à partir de $y \approx 10^{-17}$, elle croît de manière quasi linéaire en échelle logarithmique jusqu'à un maximum, ce qui correspond à un phénomène de perte de précision dû à l'arrondi numérique. Ce phénomène est particulièrement visible lorsque deux nombres de grandeurs très différentes sont additionnés. À partir de $y \approx 10^{-5}$, l'erreur atteint un plateau proche de 10^{-4}, montrant que la précision est limitée par la représentation décimale réduite.

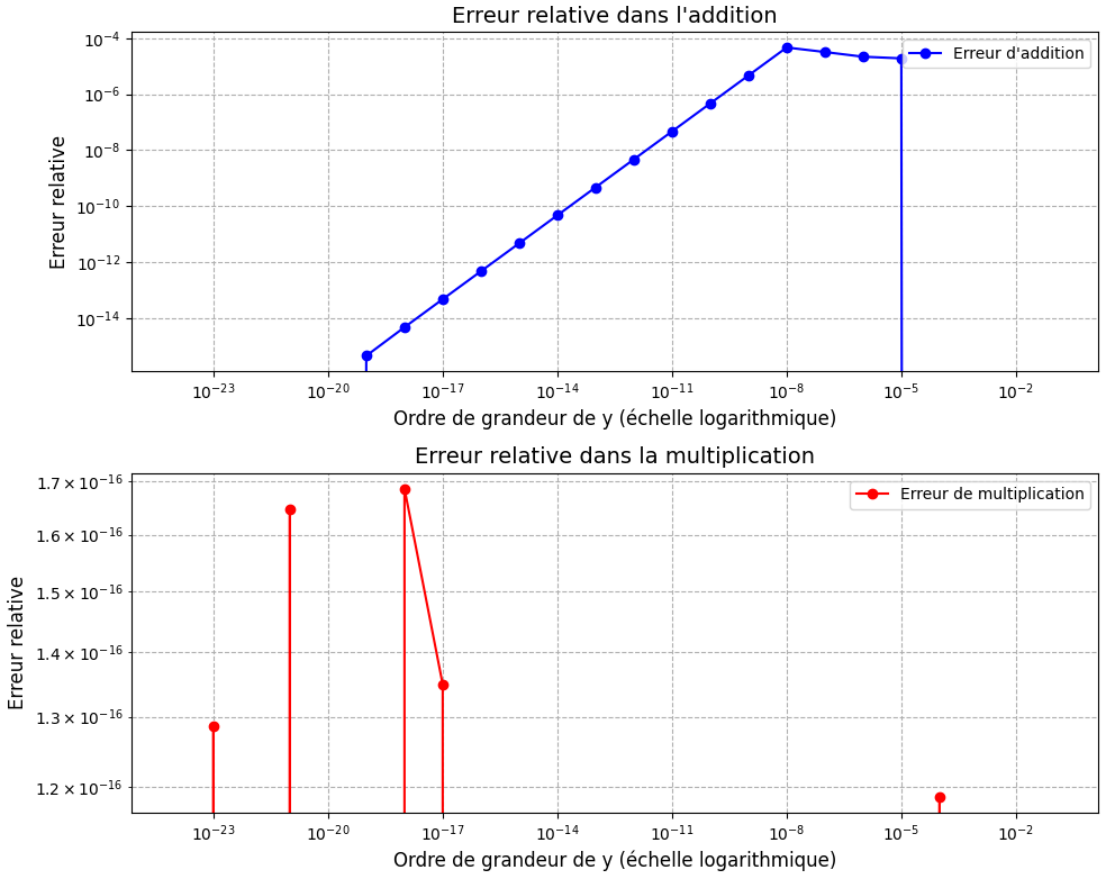

Figure 2: Erreurs relatives sur l'addition (en bleu) et la multiplication (en rouge) avec une précision $p = 5$ pour $x = 1$ et y décroissant en puissances de 10 sur une échelle logarithmique.

- L'erreur relative dans la multiplication reste globalement stable autour de 1.6×10^{-16}, ce qui correspond aux limites de précision imposées par la norme IEEE 754 pour les nombres flottants en double précision. Contrairement à l'addition, la multiplication est moins affectée par la différence d'ordre de grandeur entre x et y, expliquant cette stabilité. On observe toutefois de légères fluctuations, qui sont dues aux effets de l'arrondi numérique.

L'analyse des erreurs relatives montre que l'addition est plus sujette à des pertes de précision que la multiplication, en particulier lorsque les nombres manipulés diffèrent fortement en ordre de grandeur.

3.2 Algorithmes Numériques

Ces résultats montrent l'importance de comprendre les erreurs d'arrondi et de précision dans les calculs numériques. Pour approfondir cette analyse, nous avons mis en œuvre des algorithmes spécifiques afin d'évaluer leur impact sur des fonctions mathématiques couramment utilisées. Les sections suivantes présentent les détails des résultats obtenus pour chaque algorithme.

3.2.1 Série entière

Lors des tests de l'algorithme de la série entière pour le calcul de $\log(2)$, nous avons remarqué que plus la valeur de p augmente, plus l'algorithme met de temps à s'exécuter en raison de la convergence particulièrement lente de la série harmonique alternée. Par exemple, pour $p = 6$, la somme doit inclure jusqu'au terme $n = 100000001$, ce qui entraîne un temps d'attente considérable.

3.2.2 Algorithme CORDIC

En testant les quatre fonctions sur les entiers de 1 à 300, on observe que les fonctions ln et arctan offrent une bonne précision pour les valeurs comprises entre 1 et 300 comme présenté dans le tableau 1). Alors que exp et tan peuvent atteindre jusqu'à 9 chiffres significatifs.

Fonction	**Précision**	**Nombre de valeurs**	**Pourcentage de valeurs**
ln	11 décimales	2	0,67%
	≥12 décimales	298	99,33%
exp	9 décimales	2	0,67%
	10 décimales	4	1,33%
	11 décimales	59	19,67%
	≥12 décimales	235	78,33%
tan	10 décimales	10	3,33%
	11 décimales	41	13,67%
	≥12 décimales	249	83,00%
arctan	11 décimales	1	0,33%
	≥12 décimales	299	99,67%

Table 1: Pourcentage de valeurs ayant une certaine précision pour ln, exp, tan et arctan entre 1 et 300

Pour exp, avec une précision d'ordre 6 ($\ln(1+10^{-6}) = 0.00000099$) et la relation d'équivalence, la précision est d'environ 12 chiffres après la virgule $\frac{\exp(10^{-6})}{10^{-6}+1} = 1.0000000000004999$, tandis que pour tan, avec une précision d'ordre 4 ($\arctan(10^{-4}) = 0.0001$), la relation d'équivalence donne une précision d'environ 9 chiffres après la virgule $\frac{\tan(10^{-4})}{10^{-4}} = 1.0000000033333334$.

4 Conclusion et perspectives

Ce travail a permis de mettre en évidence les limites de la représentation des nombres flottants en machine. Les erreurs d'arrondi deviennent problématiques lorsque les nombres manipulés ont des ordres de grandeur très différents. D'autre part, elles sont plus marquées dans l'addition que dans la multiplication, ce qui met en évidence la sensibilité accrue de cette opération aux effets de la précision flottante. Les résultats sont cohérents avec la mise en évidence des limites de la précision flottante IEEE 754 étudiée en classe.
En mettant en œuvre des algorithmes numériques, nous avons pu calculer des valeurs approchées de fonctions mathématiques en utilisant des méthodes de calcul alternatives. Les résultats obtenus montrent que ces algorithmes peuvent être efficaces pour obtenir des approximations précises de fonctions mathématiques. Cependant, des séries à convergence plus rapide pourraient être utilisées pour surmonter ce problème et réduire significativement le temps d'exécution.

References

[1] Ieee standard for floating-point arithmetic, 2019.

[2] S. Friedli. Analyse 1, 2025. Consulté le 2 février 2025.

[3] Usenet France. Faq mathématiques – partie 3 : Arithmétique des ordinateurs, 2004. Consulté le 2 février 2025.

[4] Jack E. Volder. The CORDIC Trigonometric Computing Technique. *IRE Transactions on Electronic Computers*, EC-8(3):330–334, 1959.