Projet n°1
Méthodes de calcul numérique / Limites de la machine

Groupe n°2 - Équipe n°1

Responsable : tmichel003
Secrétaire : alamhamdi001
Codeurs : kdieu, matguedon, mkasour

Résumé : Ce projet consiste à évaluer les problèmes qui peuvent apparaître lors de l'utilisation d'opérations élémentaires, voire d'algorithmes plus poussés, sur des nombres flottants. La première partie s'intéresse à trouver des exemples dans lesquels les opérations élémentaires sont insuffisamment précises. La seconde partie donne des exemples d'algorithmes utilisés dans des conditions de calcul en basse précision.

1 Représentation des nombres en machine

1. Le but de cet exercice était de créer une fonction qui calcule la représentation décimale réduite d'un nombre x pour une précision p. Nous allons présenter l'algorithme réalisé sous forme d'étapes.

   Etape 1 Tout d'abord, on transforme le nombre d'origine de manière à ce que sa valeur soit comprise entre 0 et 1 exclus et que son premier chiffre des décimales soit strictement supérieur à 0. On retiendra le nombre d'opérations nécessaires pour obtenir ce résultat dans une variable nommée n.

   Etape 2 Puis, on effectue une nouvelle opération pour obtenir une partie entière dont la taille correspond à la précision p demandée. Pour cela on multiplie le nombre obtenu précédemment par 10^p.

   Etape 3 Ensuite, on réalise un arrondi sur le chiffre des unités et on ne récupère que la partie entière.

   Etape 4 Enfin, on effectue une dernière multiplication sur le nombre obtenu pour revenir au format d'origine du nombre :
   — Si le nombre d'origine est inférieur à 1 : on multiplie par 10^{n-p}
   — Sinon : on multiplie par 10^{1-n-p}

   Pour présenter les opérations effectuées, deux exemples sont dévelopés en Figure 1 pour une précision p = 6. On voit que, pour la première étape, des divisions sont nécessaires (n=6 divisions par 10) sur le premier exemple tandis que des multiplications (n=3 multiplications par 10) sont effectuées sur le deuxième.

Exemples	**Etape 1**	**Etape 2**	**Etape 3**	**Etape 4**
10507.1823	0.105071823	105071.823	105072	10507.2
0.0001857563	0.1857563	185756.3	185756	0.000185756

FIGURE 1 – Deux exemples pour une précision de 6

Avec cet algorithme, nous obtenons les mêmes résultats que dans les exemples du sujet sauf pour le nombre π à la précision 6 où nous obtenons 3.1415900000000003. Cela est dû à la représentation des nombres décimaux par un ordinateur. Ce phénomène est évoqué brièvement dans la deuxième partie de ce rapport.

Nous avions d'abord un algorithme qui calculait l'arrondi en manipulant des chaînes de caractères et nous obtenions les bons résultats mais il était demandé de ne pas passer par des chaînes de caractères.

2. Pour l'addition de deux nombres x et y, en représentation décimale réduite (RDR) à une précision p, la fonction `add_rp(x, y, p)` retourne la somme de RDR de x et de y à la précision p. Pour la multiplication, la fonction `mul_rp(x, y, p)` renvoie le produit de RDR de x et de y à la précision p.
3. Les deux fonctions `relative_err_add(x, y, p)` et `relative_err_mul(x, y, p)` calculent l'erreur relative pour la somme et la multiplication de x et y à la précision p.
4. Pour observer l'erreur relative maximale de l'addition et de la multiplication, nous avons choisi de regarder le résultat des deux fonctions précédentes pour x = π car il a un nombre infini de décimales. Par ailleurs, nous avons pris une précision de 1 car plus la précision est petite, plus l'erreur relative est élevée. Enfin, nous avons fait varier y sur l'intervalle [0,100] pour l'addition et [1,100] pour la multiplication.

Que ce soit sur l'addition (Figure 2) ou la multiplication (Figure 3), on observe que l'erreur relative maximale est supérieure à 0.30 pour l'addition et à 0.35 pour la multiplication. Cela est confirmé par le calcul de la valeur maximale précise en Figure 4 où l'on voit que l'erreur relative de la somme est d'environ 0.32 et celle de la multiplication est d'environ 0.36. Nous avons donc une erreur relative maximale de plus de 30% ce qui est plutôt élevée.

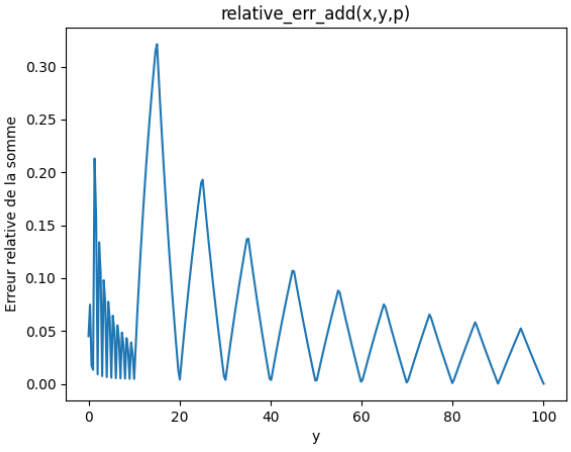

FIGURE 2 – Erreur relative pour l'addition

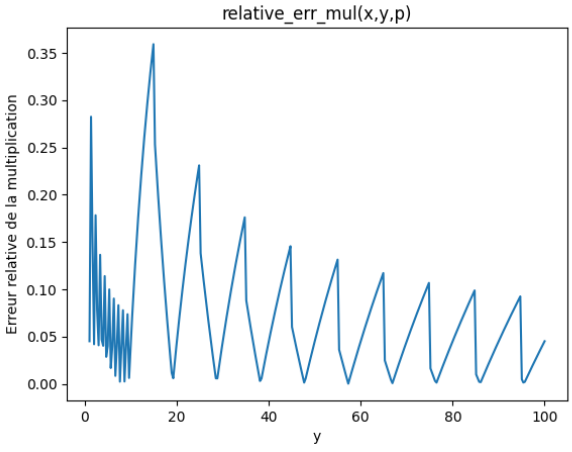

FIGURE 3 – Erreur relative pour la multiplication

```
Erreur relative maximale de la somme pour
x =  0.3141592653589793  et p =  1  :
 0.32124242569001094

Erreur relative maximale de la multiplication pour
x =  0.3141592653589793  et p =  1  :
 0.35938082138914
```

FIGURE 4 – Erreur relative maximale pour l'addition et la multiplication

5. Pour calculer nous même une valeur approchée de log(2) à la précision p, nous effectuons la somme (1) en incrémentant n tant que l'erreur relative est supérieure à 10^{-p}. Nous avons donc, en parrallèle, un calcul de l'erreur relative qui s'inspire de celui de l'addition :
$|log(2) - rp(res, p)|/log(2)$ où res est le résultat courant de la somme.

$$log(2) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \tag{1}$$

2 Algorithmes CORDIC

1. L'algorithme CORDIC (pour COordinate Rotation DIgital Computer, « calcul numérique par rotation de coordonnées ») est un algorithme créé en 1957 par un ingénieur américain, Jack Volder, et qui permet de calculer des valeurs approchées des fonctions trigonométriques. Cet algorithme a notamment été utilisé sur les premières calculatrices, au vu des rescources limitées à notre disposition (mémoire, puissance calculatoire) pour effectuer les calculs.

2. La représentation des nombres flottants utilisés sur une calculatrice est définie comme suit : un nombre occupe 8 octets soit 64 bits de mémoire et se décompose en plusieurs parties :
   — le bit de signe
   — la mantisse constituée de 13 chiffres codés indépendamment sur 4 chiffres binaires, soit codée sur 52 bits
   — l'exposant, éventuellement signé, qui est une puissance de 10.
   Cette représentation a pour avantage la facilité des opérations simples telles que l'addition, la soustraction et la multiplication ou division par une puissance de 10. Mais elle risque d'introduire des erreurs sur les flottants et un risque d'overflow.

3. Soit f la fonction par laquelle on cherche à déterminer l'image d'un entier relatif x, l'algorithme de CORDIC est applicable sur des valeurs d'un intervalle fixé. On ramène donc la valeur à évaluer dans l'intervalle où la méthode est applicable. Ensuite, on exprime f(x) sur une base formée de valeurs sauvegardées en mémoire. On obtient enfin la valeur de f(x) à une erreur près. Dans l'algorithme décrit dans la FAQ, on obtient un résultat avec plus de 12 chiffres de précision.

4. Les quatre algorithmes décrits dans la FAQ ont été implémentés sous Python dans le fichier algo_cordic.py .

5. Les algorithmes ont été vérifiés et testés par le calcul de l'erreur relative défini comme suit :

$$e_r = \frac{fonction_{CORDIC} - fonction_{python}}{fonction_{python}}$$

Les graphes en Figures 5, 6, 7 et 8 illustrent l'erreur relative pour le calcul des fonctions logarithme népérien, exponentielle, arctangente et tangente.

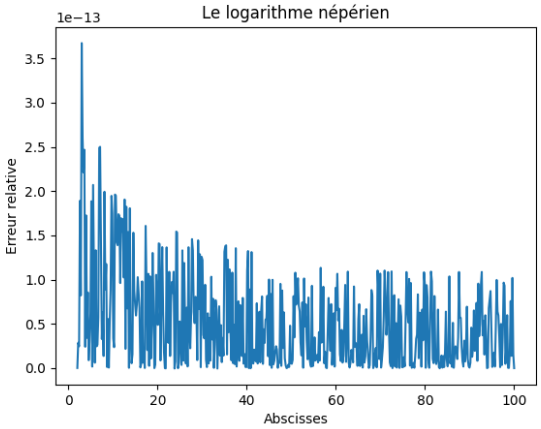

FIGURE 5 – Représentation de l'erreur relative pour le calcul du logarithme népérien

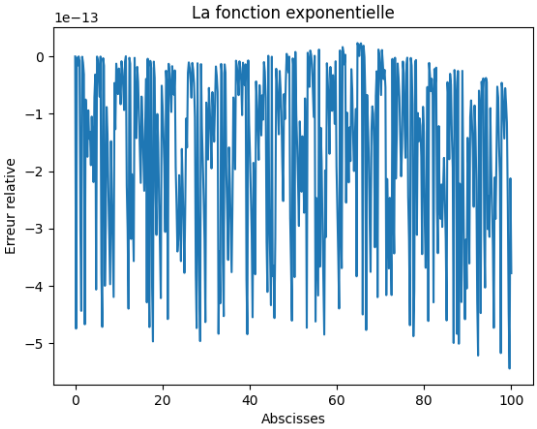

FIGURE 6 – Représentation de l'erreur relative pour le calcul de l'exponentielle

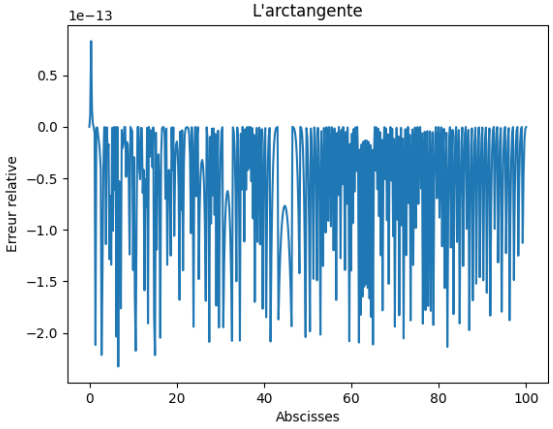

FIGURE 7 – Représentation de l'erreur relative pour le calcul de l'arctangente

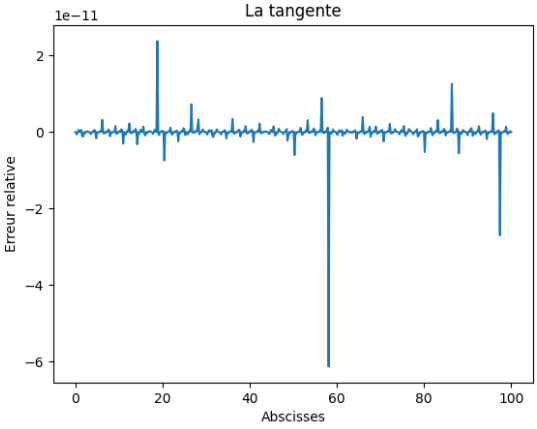

FIGURE 8 – Représentation de l'erreur relative pour le calcul de la tangente

6. (a) Le premier problème traite l'évaluation des polynômes de degré 3 au moins. Ces derniers peuvent être évalués en moins de n multiplications. Soit $P \in \mathbb{R}_n[X]$ avec $n > 3$.
Pour $n = 4$, P est naturellement de la forme :
$P(x) = a + b * x + c * x * x + d * x * x * x + e * x * x * x * x$ avec $(a, b, c, d, e) \in \mathbb{R}^5$ et $e \neq 0$
En décomposant P en la somme d'un produit de deux polynômes de degré 2 et d'un réel tel que :
$P(x) = [(A * x + B)^2 + A * x + C] * [(A * x + B)^2 + D] + E$ où A,B,C,D,E sont des réels précalculés par :
$A = e^{(1/4)}$
$B = (d - A^3)/(4A^3)$
$D = 3B^2 + 8B^3 + (bA - 2cB)/A^2$
$C = (c/A)^2 - 2B - 6B^2 - D$
$E = a - B^4 - B^2(C + D) - CD$
On passe de 10 multiplications à 3 et de 4 additions à 5. Comme les additions sont des opérations plus simples et moins complexes que les multiplications, on peut améliorer la méthode de calcul des polynômes en utilisant cette forme de décomposition.

   (b) D'une manière similaire, le deuxième problème traite la multiplication de deux nombres complexes.
Soit $(x, y) \in \mathbb{C}^2$ $\exists!(a, b, c, d) \in \mathbb{R}^4$ tel que $x = a + ib$ et $y = c + id$
On a donc $x * y = (ac - bd) + i(bc + ad)$ *(E)*
On peut réduire le nombre de multiplications en calculant le produit de la façon suivante :
$x * y = (ac - bd) + i[(a + b)(c + d) - ac - bd]$ *(F)*
(E) et *(F)* sont bien évidemment équivalentes, mais on a réduit le nombre de multiplications de 4 à 3. Comme la multiplication est une opération lente sur certaines machines, le calcul devient plus performant.

   (c) Le troisième problème concerne le module d'un nombre complexe. Soit $x \in \mathbb{C}$ tel que $x = a + ib$ avec $(a, b) \in \mathbb{R}^2$. Son module est défini par : $|x| = \sqrt{a^2 + b^2}$. Si a ou b est aussi grand que la racine carrée du plus grand nombre représentable sur machine, on risque d'obtenir un overflow malgré que le module soit représentable. Pour éviter cela, on peut calculer le module d'un nombre complexe de la façon suivante :
Si $|a| >= |b|$ alors $|a + ib| = |a|\sqrt{1 + (b/a)^2}$, sinon $|a + ib| = |b|\sqrt{1 + (a/b)^2}$

3 Conclusion

Ce projet nous a permis de comprendre les conséquences de la représentation limitée en précision des nombres dans les machines, et également dans quels cas ces imprécisions sont à prendre en compte. Dans la seconde partie du projet, nous avons vu que dans les cas où ces imprécisions dérangent il est possible de les contenir en utilisant des algorithmes astucieux, l'algorithme de CORDIC dans notre cas. Globalement ce projet fut une réussite tant par l'entente de notre équipe, que par les résultats trouvés. Une suite intéressante à ce projet serait de voir les effets de ces imprécisions sur des cas concrets (simulation physique, financières, ML, ...), voir les cas où elles deviennent gênantes et s'intéresser à leur réduction par différentes techniques.