Projet 1
Méthodes de calcul numérique / Limites de la machine

Groupe 4 - Equipe 8618

Responsable : DELPEUCH Sebastien
Secrétaire : NADIR Souhail
Codeurs : DECOU Nathan, GIRADET Maxime

Résumé : L'idée de ce projet est d'étudier les méthodes de calcul numérique. La question sous jacente, qui est aussi le coeur du sujet, est d'étudier les limites de la machine lors de calculs numériques simples (addition, multiplication) ou plus complexes, comme le celui du logarithme népérien d'un nombre.

1 Représentation des nombres en machine

Dans cette première partie, nous allons étudier la représentation des nombres en machine Python. Nous allons donc dans un premier temps préciser l'encodage des nombres dans Python. Etant donné que nous allons utiliser des nombres décimaux, nous allons étudier les nombres à virgule flottante.
Au niveau matériel, les nombres à virgule flottante sont représentés en fraction de nombres binaires (en base 2). C'est le principe de la MANTISSE. Cette méthode nous permet alors de représenter les nombres à virgule flottante. Malheuresement, la plupart des fractions décimales ne peuvent pas avoir de représentation exacte en fractions binaires. Par conséquent les nombres à virgule flottante sont souvent approximés par la machine. Nous allons étudier les limites de cette approximation dans cette partie.

Nous voulons tout d'abord pour un nombre x donné obtenir une précision de p chiffres. Ainsi, nous créons une fonction $rp(x,p)$ [1]. Par exemple si l'on donne en entrée $x = \pi$ et $p = 4$, la fonction nous renvoie 3.142.

Une fois que cela est effectué, nous pouvons rentrer dans le vif du sujet. L'idée est d'étudier et de caractériser l'écart qui existe entre la valeur "réelle" et la valeur "machine" après une opération. Pour cela, nous devons définir ce que nous entendons par valeur "réelle". La valeur réelle est la valeur définie par Python. Cette valeur est définie suivant la norme IEEE-754 double précision. C'est à dire que nous avons 53 bits de précision. Nous pouvons alors introduire la notion de $\epsilon_{\text{machine}}$. Ce dernier représente la précision "maximale" que l'on peut atteindre.

D'autre part, nous devons définir la valeur "machine". Pour cela, nous définissons deux fonctions. La première permet de réaliser l'addition réduite (c'est à dire que nous fournissons deux entiers x et y et que la fonction renvoie l'addition à la précision p). Nous pouvons noter que la troncature permettant de nous donner le nombre à la bonne précision est réalisé après l'addition. Cela ne modifie en rien nos calculs. De manière analogue nous définissons la seconde fonction : la multiplication réduite.

1.1 Etude de la somme

Commençons par étudier la somme. Pour cela nous devons définir un critère permettant d'évaluer la "différence" entre la valeur réelle et la valeur machine. Ce critère est fourni et se nomme l'écart relatif

$$\delta_s(x,y) = \frac{|(x+y)_{\text{reel}} - (x+y)_{\text{machine}}|}{|(x+y)_{\text{reel}}|} \tag{1}$$

Pour étudier ce critère, nous allons regarder son conditionnement. Commençons par définir le conditionnement d'une fonction :

$$C_f(x) = |\frac{\delta f(x)}{f(x)} / \frac{\delta x}{x}| \overset{T.Y.}{=} |\frac{f'(x)x}{f(x)}| \tag{2}$$

Dans le cas de l'addition, nous utilisons la fonction $f : y \mapsto y + 1$. Ainsi, le conditionnement de cette dernière est $C_f : y \mapsto \frac{y}{y+1}$. Ce conditionnement nous montre (même si nous pouvons le déduire autrement) que l'erreur relative sur la somme va suivre la courbe $\frac{1}{1+y}$. Nous allons donc tracer les courbes dans le cas général (nous évoquerons le cas critique plus tard). Sur ce graphique nous avons fixé x à 1 et nous faisons varier y entre 0 et 5.

1. la complexité de cette fonction est en $0(\log(x))$

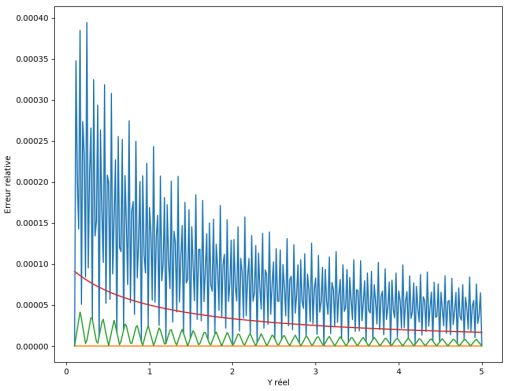

FIGURE 1 – Erreur relative sur la somme de $1+y$ avec y entre 0 et 5

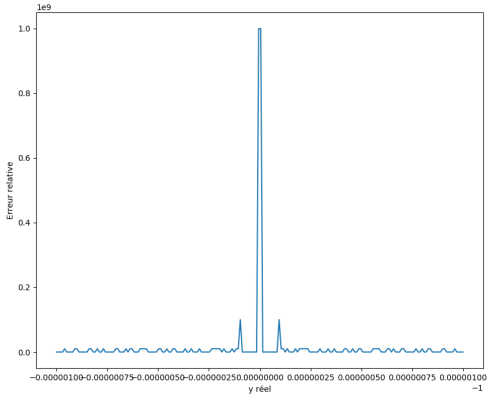

FIGURE 2 – Erreur relative de la somme aux alentours de -1

Dans un premier temps, intéressons nous à la figure 1. La courbe bleue représente l'erreur relative de la somme avec une précision à 4 décimales. Nous pouvons remarquer que la courbe n'est pas stable : elle possède de fortes variations. Cependant, une tendance générale se dégage. Lorsque nous comparons avec la courbe rouge (qui représente la fonction $f : y \mapsto \dfrac{1}{1+y}$), nous pouvons confirmer que l'erreur relative suit bien $\dfrac{1}{1+y}$. De plus, nous avons tracé en vert l'erreur relative pour $p = 5$ et en orange l'erreur relative pour $p = 53$. Nous remarquons dans un premier temps que les erreurs relatives sont plus faibles que pour 4 décimales, elles suivent toujours la courbe $\dfrac{1}{1+y}$. Pour la courbe orange, notre précision dépasse la précision maximale de la machine. Ainsi, sur cet échelle, nous avons l'impression que l'erreur relative est nulle. Ceci est purement artificiel. En effet, si nous nous approchions plus près, nous verrions qu'elle n'a aucun sens car la machine ne peut pas fournir une valeur aussi précise.

Au vu de la fonction caractérisant l'erreur relative et la fonction de conditionnement, nous pouvons soulever rapidement un problème lorsque $y = -1$. En effet, cela va provoquer une division par 0. Nous avons donc fait le graphique 2 en zoomant sur la valeur -1. Nous pouvons noter sur ce graphique une explosion en -1 (l'échelle est un peu spéciale, il faut retirer 1 à chaque valeur de l'axe des ordonnées). En conclusion, nous pouvons voir que le conditionnement de la somme $x + y$ est relativement bon en dehors de la valeur $x = -y$.

En ouverture, nous pouvons regarder l'erreur relative lorsque nous dépassons $\epsilon_{\text{machine}}$. Nous traçons alors le graphique 3 de l'erreur relative à une précision de 17 décimales. Nous avons le résultat qui suit. Nous pouvons directement voir que les résultats ne sont plus cohérents, nous demandons une précision plus grande que celle que peut nous fournir la machine.

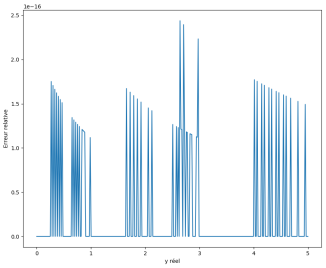

FIGURE 3 – Erreur relative avec une précision de 17 décimales

1.2 Etude de la multiplication

Nous allons maintenant effectuer le même travail pour la multiplication. Nous définissons alors l'erreur relative de la multiplication :

$$\delta_p(x,y) = \frac{|(x \times y)_{\text{reel}} - (x \times y)_{\text{machine}}|}{|(x \times y)_{\text{machine}}|} \tag{3}$$

Pour étudier la multiplication, nous allons nous intéresser à la fonction $f : x \mapsto 2x$. Nous pouvons directement remarquer que son conditionnement vaut 1 : il nous indique donc que l'erreur relative ne va jamais exploser.

Nous pouvons alors tracer la courbe de l'erreur relative de la multiplication pour x variant entre -4 et 4. Nous pouvons voir sur ce graphique deux courbes. Tout d'abord en bleu, la courbe avec une précision à 4 décimales et en orange la courbe avec une précision à 5 décimales. Les deux courbes ont un comportement similaire : nous ne pouvons pas déterminer une tendance comme pour l'addition. Cependant nous pouvons voir des modifications conséquentes en 0, ce qui est normal car nous avons un calcul du type $0/0$. En conclusion, la multiplication possède une erreur relative différente de l'addition. Lorsque l'addition suit $\frac{1}{x+y}$, l'erreur relative de la multiplication s'ajoute.

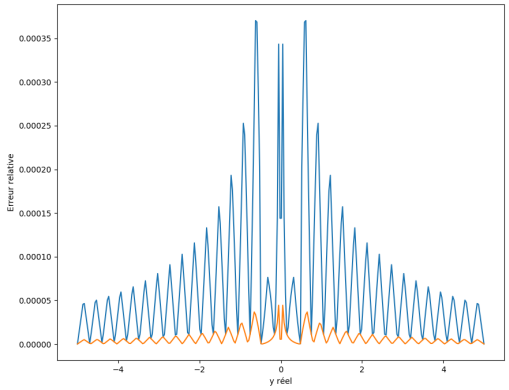

FIGURE 4 – Erreur relative de la multiplication entre -4 et 4

1.3 Calcul du log(2)

Finalement nous allons tenter d'approximer une fonction plus complexe : le logarithme népérien. Plus précisément, il s'agit de sa valeur en 2, à une précision p. Pour ce faire nous allons utiliser la formule suivante :

$$log(2) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \tag{4}$$

Nous définissons alors la fonction $log\_2(p)$ permettant de calculer le logarithme de 2 à la précision p. L'idée est d'étudier la précision du résultat final par rapport au logarithme de 2 enregistré par Python (c'est à dire à la précison $\epsilon_{\text{machine}}$). Nous réalisons le graphe qui suit. Nous pouvons voir que l'erreur relative s'approche rapidement de 0. le modèle proposé est donc relativement fiable. Comme pour les opérations simples, si nous dépassons l'$\epsilon_{\text{machine}}$, l'erreur relative n'a plus de sens.

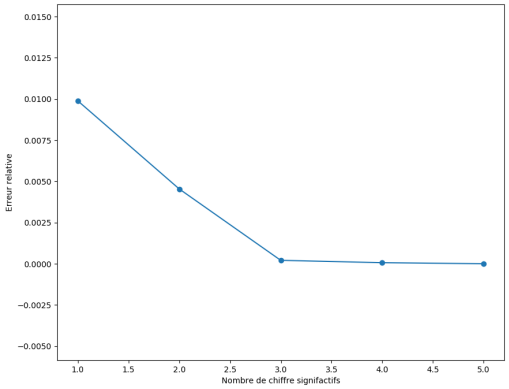

FIGURE 5 – Erreur relative entre $log(2)_{\text{reel}}$ et $log(2)_{\text{calcule}}$ pour des valeurs de p de 1 à 5

2 Algorithmes CORDIC

2.1 Fonctionnement général

Sur une calculatrice, un nombre flottant occupe 8 octets de mémoire. Le nombre se décompose en une mantisse de 13 chiffres dont chacun est codé sur 4 chiffres binaires, un exposant sous la forme d'une puissance de 10 qui peut être signé et un bit de signe.

Les algorithmes CORDIC reposent sur la technique d'interpolation linéaire. Nous allons l'exploiter pour quatre fonctions : ln , exp , tan et arctan .

Pour chacune d'entre elles, il est nécessaire de disposer d'une base d'interpolation. Ici, ce seront deux tableaux (L et A). Le tableau L est utilisé pour les fonctions ln et exp , tandis que le tableau A est utilisé pour les fonctions tan et arctan .

Le principe est aisé : grâce à des formules caractéristiques de la fonction considérée, nous découpons le problème en un ensemble de sous problèmes qui peuvent être résolus grâce à la base d'interpolation.

Celle-ci est obtenue en exploitant les propriétés des fonctions : pour les fonctions ln et exp , l'élément i du tableau L est $L[i] = \ln(1 + 10^{-i})$.

Pour les fonctions tan et arctan, l'élément i du tableau A est $A[i] = \arctan(10^{-k})$
La longueur du tableau dépend de la précision désirée pour le calcul des images par les fonctions. Par exemple, pour les fonctions ln et exp, afin d'obtenir une précision p, il est nécessaire que la longueur du tableau L soit de $p/2 + 1$.

Afin de rendre effectifs les algorithmes correspondant aux 4 fonctions, il fut nécessaire d'y apporter quelques modifications. En effet, pour les fonctions ln et arctan, il fut nécessaire d'ajouter la condition $k \leq 6$ et $k \leq 4$ respectivement, dans la boucle *while* la plus intérieure.
Sans ces ajouts, un débordement de tableau se produisait.

Les sous problèmes évoqués plus haut sont obtenus après l'application d'une série de transformations simples qui permettent de se rapprocher des valeurs de la base d'interpolation. C'est pourquoi il est nécessaire de disposer de ces quelques valeurs qui correspondent aux images de nombres bien choisis par la fonction considérée ou sa réciproque.
En effet, pour les fonction exp et ln, la même base d'interpolation est utilisée. Il en est de même pour tan et arctan.

Une transformation typique est par exemple : $ln(x) = ln(x \times 10^{-n}) + n \times ln(10)$, ce qui permet de trouver m entiers tels que $n_0 < n_1 < ... < n_{m-1} = 1$ et $ln(x) = n_0 \times \ln(1+1) + n_1 \times \ln(1+1/10) + ... + \ln(1+10^{-m+1})$.
C'est cette décomposition qui permet ici, dans l'exemple de la fonction ln, de se rapprocher peu à peu des valeurs de référence.

2.2 Etude des erreurs relatives

Maintenant que les quatre fonctions sont programmées, nous allons étudier les erreurs relatives commises par les algorithmes pour plusieurs précisions. Sur tous les graphiques, nous représentons toujours la variation de l'erreur relative en fonction de x.

Logarithme. Nous avons tracé le graphe de l'erreur relative, c'est à dire de la fonction f définie ainsi, sur [2 ;10] :

$$f : x \longmapsto \frac{|ln\_cordic(x) - ln(x)|}{|ln(x)|} \tag{5}$$

La fonction f est représentée par la courbe bleue dans le graphique 6. La courbe orange représente uniquement le dénominateur de la fonction f, ie $x \longmapsto \frac{1}{|ln(x)|}$. Nous pouvons voir que l'erreur relative suit $\frac{1}{|ln(x)|}$. Nous allons maintenant regarder aux alentours du cas critique, c'est à dire au voisinage de 1. Nous traçons alors la figure 7 qui se concentre sur l'interalle $[0.5; 1.5]$ avec une précision de 12 décimales. Nous constatons l'existance d'un point de discontinuité en $x = 1$. Ceci est normal car le dénominateur de la fonction f s'annule en $x = 1$. (courbe en vert).
Maintenant, si l'on applique une précision de seulement 2 décimales, nous remarquons des oscillations, accompagnées d'un décalage de la discontinuité. (courbe bleue).
La courbe orange représente le dénominateur de f et comporte un point de rebroussement, toujours en $x = 1$ (courbe orange). Nous pouvons donc voir sur l'exemple du logarithme que le CORDIC nous permet un calcul précis pour des fonctions plus complexes.

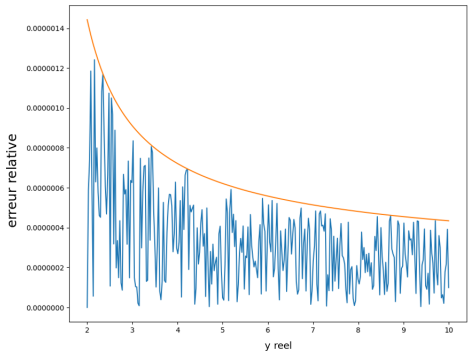

FIGURE 6 – Erreur relative de la fonction logarithmique sur 2-10

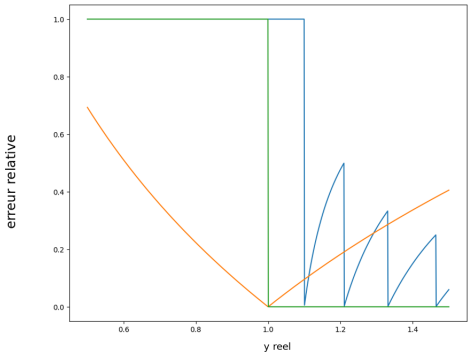

FIGURE 7 – Erreur relative de la fonction logarithme proche de 1

La même expérience a été réalisée avec les autres fonctions : pour la fonction *exponentielle*, il n'y a pas de singularité sur la courbe de l'erreur relative. Pour une précision de 12 décimales, celle-ci est au maximum de 5×10^{-13}, ce qui montre que l'algorithme CORDIC réussit, avec seulement un tableau de longueur 7, à obtenir une valeur très précise.

Il en est de même pour la fonction *tangente* dont la courbe (figure 8) de l'erreur relative présente une discontinuité en $x = \pi/2$ (annulation du dénomminateur) : mise à part l'explosion engendrée par celle-ci, la valeur de l'erreur relative (sur $[0.2\,;1.5]$) est au maximum de 2×10^{-14} pour une précision de 12 décimales.

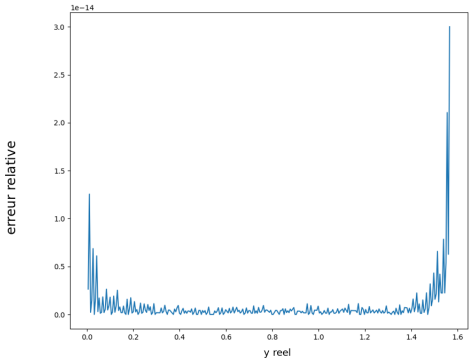

FIGURE 8 – Erreur relative de la fonction tangeante montrant l'explosion en $\pi/2$

Ainsi, les algorithmes CORDIC peuvent être considérés comme très performants : ils permettent, grâce à un nombre très limité de valeurs issues de tables de calculer une valeur très approchée de l'image d'un nombre par une fonction.
Les calculatrices modernes affichant un nombre limité de décimales (12 par exemple), les deux tableaux L et A présentés précédemment leur permettent de réaliser tous les calculs afin d'afficher le résultat.

2.3 Evaluation des fonctions usuelles

La lecture du "Numerical Recipes in C" nous fait réfléchir sur plusieurs points qui peuvent poser problème lors de l'évaluation de fonctions. Nous en avons sélectionné 3 :

1. Le calcul du module d'un nombre complexe
2. Le calcul des racines d'un polynome du troisième degré
3. Le calcul de la dérivée d'une fonction en un point

1. La formule usuelle $|a+ib| = \sqrt{a^2+b^2}$ peut mener à un OVERFLOW si a ou b est plus grand que le plus grand nombre représentable en machine. La solution est de factoriser par $max(|a|,|b|)$, ce qui permet de limiter le risque d'OVERFLOW. Nous avons donc

$$|a+ib| = \begin{cases} |a|\sqrt{1+(b/a)^2} & \text{si } |a| > |b| \\ |b|\sqrt{1+(a/b)^2} & \text{sinon} \end{cases} \tag{6}$$

2. Il est possible de déterminer les racines d'un polynôme du troisième degré en calculant Q et R définis grâce aux coefficients du polynôme. Si Q et R sont réels et si $R^2 < Q^2$, il y a trois racines réelles qui se calculent grâce à la fonction cos. Le problème majeur est l'erreur d'arrondi qui peut apparaître lors de l'évaluation de la fonction cos. Afin de solutionner ce problème, il convient de passer par l'ensemble des nombres complexes. On calcule $A = -\sqrt[3]{R+\sqrt{R^2-Q^3}}$ et $B = Q/A$ si A est non nul, et 0 sinon. Nous pouvons alors en déduire les trois solutions. Par exemple, $x_2 = -\frac{1}{2}(A+B) - \frac{a}{3} + i\frac{\sqrt{3}}{2}(A-B)$. Le calcul s'effectue donc uniquement grâce à des multiplications, des additions, et le calcul de racines carrées ou cubiques.

3. Pour calculer la dérivée en un point d'une fonction f, nous utlisons la formule : $\lim_{h\to 0} \frac{f(a+h)-f(a)}{h}$. Cependant, la valeur de h choisie pour le calcul peut être plus petite que $\epsilon_{\text{machine}}$ ce qui entraîne une troncature et une erreur d'arrondi. La solution est de choisir h tel qu'il soit plus grand que $\epsilon_{\text{machine}}$, mais aussi qu'il soit représentable exactement par la machine.