Méthodes de calcul numérique et limites de la machine

Coordinateur de projet : Luxel HAMOUCHE
Secrétaire : Allan DENOCE
Programmeurs : Joachim ROBERT & Joris ROUSERE & Maximilien VIDIANI

Semaine 1 :
Tandem 1 : Joris ROUSERE & Allan DENOCE
Tandem 2 : Joachim ROBERT & Luxel HAMOUCHE & Maximilien VIDIANI

Semaine 2 :
Tandem 1 : Joris ROUSERE & Joachim ROBERT
Tandem 2 : Allan DENOCE & Luxel HAMOUCHE & Maximilien VIDIANI

1er février 2023

Table des matières

1 Représentation des nombres en machine

1.1 Introduction

2 Exemples d'algorithmes numériques

2.1 Introduction

Dans cette partie, on cherche à écrire un algorithme permettant de calculer une valeur approchée de log(2). Pour se faire, on utilisera l'écriture du logarithme de 2 sous la forme de la somme suivante : $log(2) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}$.

On étudiera aussi plusieurs algorithmes CORDIC qui sont notamment utilisé pour les calculatrices. Les fonctions de cette partie sont présentes dans le fichier *partie2.py*.

2.2 Logarithme népérien de 2

On va s'interesser à un algorithme en python afin de pouvoir calculer une valeur approché de log(2) sur p décimales. On a décidé de décomposer l'algorithme en 2 algorithmes différents.

Le premier qui s'appelle calclog prend en argument un nombre m et calcul la valeur de la somme correspondant au logarithme de 2. On a en argument le nombre m car il nous est impossible de calculer une somme infini en python d'où l'utilisation d'un nombre m que l'on prend très grand. On se retrouve avec la formule suivante : $log(2) = \sum_{n=1}^{m} \frac{(-1)^{n+1}}{n}$.
On a effectué deux boucles for, une allant de 1 à m et une allant de m à 1. On va comparer l'efficacité des 2 fonctions dans le tableau ci-dessous.

La deuxième fonction logdec prend p en argument et à l'aide de la fonction round, elle nous renvoie le nombre de décimale attendu.

m/sens	1 à m	m à 1
10	0.74563	0.64563
100	0.69817	0.68817
1000	0.69365	0.69265
10000	0.69320	0.69310
100000	0.69315	0.69314

La valeur réelle de log(2) est 0,6931471805599. On peut donc voir que les 2 façons de faire se rapproche bien de la valeur attendu.

2.3 Algorithmes CORDIC

Dans cette partie, on va s'interesser à une FAQ du groupe de discussion *fr.sci.maths* qui explique l'utilisation des algorithmes CORDIC dans les calculatrices.

a. Dans une calculatrice typique un nombre 'flottant' occupe 8 octets de mémoire et se décompose en :
   - une mantisse constituée de 13 chiffres codés indépendamment sur 4 chiffres binaires (on parle de Binaire Code Decimal ou BCD)
   - un exposant (puissance de 10) éventuellement signé
   - le signe du nombre (et de l'exposant s'il n'est pas signé)

   AVANTAGES : On ne stocke que sur 8 octets et nous n'avons pas besoin d'une grosse précision sur des calculatrices de base.
   INCONVENIENTS : On ne peut pas faire de gros calculs à cause de la mentisse de 13 chiffres et ca nous apporte une perte de précision.

b. La technique générale pour réaliser les 4 algorithmes est d'utiliser une interpolation linéaire voir un développement en série.
   On réduit au fur et à mesure l'intervalle de valeur servant à l'interpolation qui se rapproche de plus en plus de la valeur réelle de x.
   Pour se faire, on dispose de 2 tableaux contenants des valeurs précalculées de l'inverse de la fonction exponentielle (Logartihme népérien) et un tableau de fonction atan.
   Cette technique est efficace car elle ne necessite qu'un tableau avec quelques valeurs de la fonction.