Documentation de l'extension

GUILLAUME Laure
LUU Duc Anh
EL RHATRIF Mohammed Amine
CHAKIR Sami
FARID Othmane

16 janvier 2017

1 Idée globale

Il s'agit d'implémenter les fonctions sin, cos, arcsin et arccos. Il s'agit plus précisément de trouver la meilleure approximation de ces quatre fonctions.

Naturellement, la première idée qui vient est celle du développement limité. Puis, en essayant de trouver des algorithmes d'approximation de fonctions trigonométriques, ou de fonctions tout simplement, deux algorithmes se répétaient souvent : l'algorithme CORDIC, qui est un algorithme de calcul des fonctions trigonométriques et hyperboliques, notamment utilisé dans les calculatrices et l'approximant de Padé, qui est analogue au développement limité, mais qui a recours aux fractions rationnelles au lieu des polynômes. Ce sont donc ces trois pistes qu'on a décidé de développer.

2 Etat courant

Ce qui a été réalisé pour le moment est le développement limité des fonctions à implementer, et ce de deux façons pour chacune.

Pour les fonctions cos et sin, on a calculé leur développement limité en premier lieu naïve, c'est-à-dire en traduisant directement leur développement, la deuxième méthode consistait à exploiter le lien entre le terme du rang $i+1$ et celui du rang i pour coder le développement limité de façon récursive, chose qui améliorait la précision des résultats obtenus.

En effet, on a :

$$\begin{aligned} termeRang_I &= (-1)^n \frac{x^{2n+1}}{(2n+1)!} \\ &= -((-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!}) \frac{x^2}{(2n+1)2n} \end{aligned}$$

$$= -\frac{x^2}{(2n+1)2n} termeRang_{I-1}$$

Pour la fonction arcsin, on a de même calculé son développement directement au début, puis on l'a calculé en passant par la fonction arctan, et puis en exploitant l'identité :

$$arcsin(x) = arctan(\frac{x}{\sqrt{1-x^2)}})$$

On a procédé de façon analogue pour la fonction arccos.

La seconde méthode semble plus exhaustive quand l'argument pris est compris entre 0 et 0.5, pour le reste le développement limité direct est meilleur.

3 Fonctions annexes

On a implémenté de nouvelles fonctions pour que le code des fonctions demandées soit le plus lisible possible, ceci car les calculs demandés sont basées sur des formules assez complexes.

Pour les fonctions cos et sin par exemple, on a implémenté la fonction fact(n), qui prend comme paramètre un entier n et qui retourne la quantité $n!$.

Pour la fonction arcsin, on a implémenté la fonction semifact(n), qui prend comme paramètre un entier n et qui retourne la quantité $n(n-2)...4*2$ si n est pair, et $n(n-2)...3*1$ sinon. On a également introduit la fonction racine(f), qui a été implémentée en utilisant la méthode de Newton-Raphson.

4 Ce qu'il reste à faire

Cette première partie de l'extension nous a pris un temps assez conséquent, or elle peut même ne pas figurer dans notre classe finale. Cependant, elle a été très maîtrisée, et surtout, elle a permis de comprendre les contraintes du problème (précision, rapidité, aberrations ...).

Ce qu'il reste à faire donc est donc de se documenter sur les deux autres méthodes (algorithme CORDIC et approximation de Padé), prendre le temps nécessaire pour les comprendre, la programmation ne nécessitant pas une longue période (d'après ce qu'on a conclu de la programmation du développement limité).