Choix d'extension

Projet GL – Groupe 4 GL20

Thématique choisie : TRIGO

Contexte

La thématique trigo consiste à créer une suite de fonctions mathématiques comme sinus, cosinus, arc-sinus, arc-tangente, ulp. Il faudra donc créer une class Math avec toutes les méthodes nécessaires.

Ces fonctions sont à la base de la majorité de calculs que pourraient faire une machine. L'idée de se confronter à des problématique de mémoire, temps de calcul ou encore précision de calcul nous semblent très intéressantes : en effet, cette expérience nous permettra de mieux appréhender les méthodes et les dérives que nous constatons sur certains algorithmes / certaines machines.

Objectifs

L'objectif est de créer des fonctions précises et efficaces. Il ne serait pas acceptable d'avoir une fonction qui calcul un cosinus en plusieurs secondes. A l'inverse, une précision minimum est attendue pour éviter les dérive en cas de fonction récursive qui utiliserait la bibliothèque que nous créons.

Pour atteindre le meilleur compromis entre temps de calcul et précision il nous faudra deux éléments :

- Un état de l'art de l'existant et des performances du marché
- Des solutions d'essais pour analyser et comparer les algorithmes développés

L'état de l'art nous permettra de bien comprendre quels sont les verrous technologiques et les limites des méthodes qui existent. Nous pourrons ensuite chercher à vérifier ces limites avant de réfléchir a certaines solutions pour améliorer cela.

Une discussion avec le client sera nécessaire afin de définir quelles sont les attentes en terme de performance : mieux vaut-il dégrader la précision du calcul ou bien son temps de calcul. Un livrable type tableau d'essais pourrait donner une vision d'ensemble au client afin de lui permettre de faire le meilleur choix.

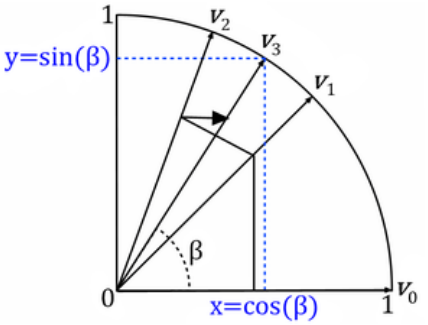
Déroulé

Nos premières recherches nous ont permis d'identifier l'algorithme CORDIC : celui-ci consiste à calculer des valeurs de sinus et cosinus à l'aide de la somme des termes précédents. Cela permet non seulement de contourner une partie de la complexité algorithmique, mais aussi d'avoir une précision intéressante sur chaque valeur. En revanche, le moindre décalage sera conservé et risque

d'être amplifié sur les itérations suivantes. Nous pourrions donc imaginer une solution de contrôle afin de s'assurer que la valeur retournée est bien cohérente par rapport au calcul demandé.

Aujourd'hui il semblerait que l'algorithme CORDIC soit moins utilisé notamment parce qu'il existe aujourd'hui des coprocesseurs arithmétiques dans la majorité des CPU. Des notions comme l'approximant de Padé ou bien la méthode de Homer pourront être creusées afin d'améliorer la complexité algorithmique de notre produit.

Le schéma ci-contre montre plusieurs itérations sur la base de l'algorithme CORDIC. Pour réaliser les calculs il faudra probablement mettre en place du calcul matriciel. Une formule générale pourra être caractérisée afin d'essayer d'anticiper les capacité de l'algorithme à atteindre la précision voulue. L'algorithme nous devrait nous permettre de réaliser une majorité de fonctions trigonométriques classiques.



Tests

Un tableau de ce type pourra être rempli afin de faire un suivi précis des performances

Développeur	Type d'algorithme	Temps d'exécution	Complexité

Les notes pourrons aller de (moins bonne note) 0 à 5 (meilleure note) sur chacune de ces thématiques. Il reviendra ensuite au client de faire son choix par rapport aux performances des essais.

Remarques

Les ressources trouvées sur cette thématique sont variées. On y retrouve des méthodes d'implémentations itératives, d'autres méthodes linéaires. En revanche nous noterons qu'il ne semble pas y avoir de méthode récursive.

Bibliographie

1. https://fr.wikipedia.org/wiki/CORDIC
2. https://irem.univ-reunion.fr/spip.php?article445