Extension [TRIG] - GL01

Présentation et engagements de l'extension [TRIG]

L'extension [TRIG] présente une implémentation des fonctions trigonométriques classiques:

- float ulp(float f)
- float sin(float f)
- float cos(float f)
- float asin(float f)
- float atan(float f) ...

L'équipe GL01 s'engage à ce que ces fonctions remplissent les critères suivants:

- Performance: Les fonctions devront être optimisées pour délivrer le résultat avec le minimum de compléxité temporelle et spatiale possible.
- Précision: Pour un coût en performance déterminé, l'utilisateur peut choisir la précision des fonctions implémentées. Il peut aussi choisir les différents modes (Rapide - moins de précision ou Scientifique - Grande précision).
- Fiabilité: Les résultats devront être testés niveau précision et performance(Pas de surprise!).

Différents algorithmes

Pour arriver à remplir les critères précisés plus haut on se verra utiliser plusieurs algorithmes en fonction de l'entrée.

- Série de Taylor: Celle-ci offre une précision déterminée au préalable pour un coût en performance (Factorielles à calculer + Puissances):

$$|f(x) - f(0) - \sum_{k=1}^{n} \frac{x^k}{k!} f^{(k)}(0)| \leq M \frac{x^{n+1}}{(n+1)!}$$

  L'ordre n à choisir est tel que $M\frac{x^{n+1}}{(n+1)!} \leq p$ avec p la précision voulue. On peut simplifier ce test en utilisant une table de factorielles (Avec quelques valeurs suffisantes puisque cette méthode converge rapidement).

  Inconvénients: Cette méthode diverge rapidement si x s'éloigne de 0.

- Lookup Table: Cette méthode retourne une valeur approchée en échange de complexité spatiale. En effet, plusieurs valeurs sont stockées et on peut utiliser une recherche dichotomique pour trouver $v_1 \leq x \leq v_2$. Un test de précision est nécessaire après avoir trouvé une valeur.
- Modulo 2π: Si les valeurs sortent de l'intervalle $[0, 2\pi]$, les méthodes précedentes ne sont plus effectives, donc il faut se rendre à cet intervalle en profitant de la périodicité des fonctions.

Progression

On prend l'algorithme pour la fonction sin:

Si $x \leq 10^{-5}$: return x

Si $x \leq 1$: return Taylor(x, ordre=6)

D'après l'étude des performances et précision pour les ordres 4,5,6 et 7, on en déduit que 6 offre le meilleur compromis précision/performance.

Si $x \leq \pi/2$: return Lookup_Table_sin(n)
La Lookup_Table_Sin est une table contenant un certain nombre de valeurs V_n et leurs $sin(V_n)$ exacte entre 1 et $\pi/2$. Leur distribution uniforme nous garantit une majoration de l'erreur $\leq$ size(Lookup_Table_sin). Mais pour un compromis correcte entre la précision et l'espace de stockage occupé, on a choisit une valeur de 10^4.

Sinon: return sign(x)*sin(principal(x))
La fonction principal(x) retourne la valeur v dans $[0, \pi/2]$ pour laquelle $|sin(x)| = |sin(v)|$. On effectue un traitement de signe par la suite.