Matthias Gradaive
Mamadian Diallo

PPAR GPU lab 1 & 2

1.
Voir *log_2_series*

2.
Plus i est grand, plus la valeur de (+-1)/(i+1) est petite et necessite de la précision.
Quand le calcul est effectué avec des valeures de i décroissante, quand les valeurs de i sont grandes, la somme est encore composée uniquement de ces petits nombres necessitant de la précision.
Quand le calcul est effectué avec des valeures de i croissante, la somme contient dejà les premiers résultats, qui sont bien plus grands que les résultats avec i proche de n.
Il y a donc moins de bits pouvant être dédiés à la précision du nombre flottant.

3. 4. 5. 6.
Les deux méthodes sont implémentées dans *summation_kernel.cu*. Voir *summation_kernel* et *summation_kernel2*.
En utilisant le GPU, le calcul est réalisé 6 fois plus performants (on passe de 0.95s à 0.14s). Sur le GPU, les calculs sont parallélisés sur plusieurs dizaines de threads.

7.
Les meilleurs résultats ont été obtenus avec 8 bloc dans la grille et 512 thread par bloc.

8.
Lorsque chaque thread à terminé sa somme, nous pouvons faire communiquer entre eux tous les threads au sein des blocs. On effectue alors la somme des résultats des threads de chaque bloc avec une réduction.
Les threads ne renvoient donc plus de résultats au CPU, seuls les blocs le font. Cela divise le nombre de valeurs retournées par 512.
Voir *reduced_summation_kernel*

9.
Nous executons deux kernels différents. Le premier est similaire à celui de la question 8, le deuxième réalise la somme des résultats retournés à la question 8 et retourne le résultat final en un seul float.
Voir *reduced_array_summation*