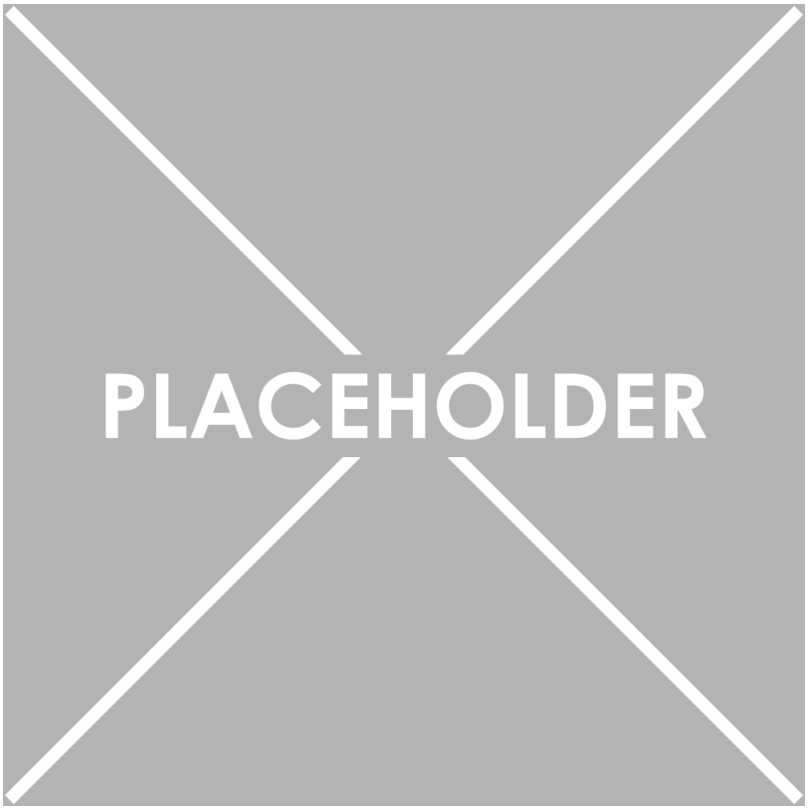
Foo Bar Title

Thèse de master en sciences informatiques

Session 2017



Professeur responsable : Latt Jonas
Diplômant : Python Adrien

Table des matières

Table des figures

Acronymes

LBM Lattice Bolzmann Method

CPU Central Processing Unit

GPU Graphics Processing Unit

FMA Fused Multiply-Add

Avant-propos

Présentation

Conventions typographiques

Ce document suit des règles typographiques afin d'en simplifier la lecture :

- l'*italique* est employé pour les termes anglais ou sur lesquels l'attention est attirée ;
- une police de caractère à `chasse fixe` indique un extrait de code, une commande, un chemin ou la sortie standard d'une console.

Remerciements

Chapitre 1

Introduction

Chapitre 2

Calculs sur nombres flottants

2.1 Norme IEEE 754

2.2 Différences entre calculs sur CPU et GPU

2.2.1 Observations

Cuda supporte la norme IEEE 754 pour les calculs avec nombre flottants [2]. Toutefois, la première implémentation Cuda de la LBM a montré des différences sur certains calculs réalisés par les codes Central Processing Unit (CPU).

2.2.2 Mesures

Pour mesurer cette différence, un code LBM $2D$ CPU et un code Graphics Processing Unit (GPU) ont été implémenté et leurs résultats comparés. La figure 2.1 illustre la différence moyenne entre les valeurs calculées sur CPU et GPU pour l'ensemble des populations f_{in} de chaque itération.

L'écart entre les résultats observés sur CPU et GPU suit les étapes de la simulation réalisée (fluide autour d'un cylindre) illustrée par les figures 2.2 :

- entre 0 et environ 12000 itérations, la queue se forme derrière le cylindre et la l'écart moyen reste faible et stable (quinze zéros après la virgule) ;
- entre 15000 et 20000 itérations, la queue commence à osciller et l'écart augmente brutalement (treize zéros après la virgule) ;
- à la $25792^{\text{ième}}$ itération, l'oscillation commence à se stabiliser et l'écart atteint son maximum (toujours treize chiffres après la virgule) ;
- dès environ 36000 itérations, l'oscillation est stabilisée (comme le montre les figure 2.2(f) et 2.2(j), même motif réapparaît toute les 4000 itérations environ) et les écarts avec.

La différence entre certains résultats sur CPU et GPU est le fruit d'une optimisation du compilateur Cuda.

2.2.3 Fonctions intrinsèques

Sur CPU, lorsqu'une opération du type $r = x \times y + z$ est rencontrée, le processeur calcul d'abord $r_{xy} = x \times y$ puis $r = r_{xy} + z$. Deux opérations sont donc réalisées, ce qui implique deux potentiels arrondis dans le cas des nombres flottants. Sur GPU, le

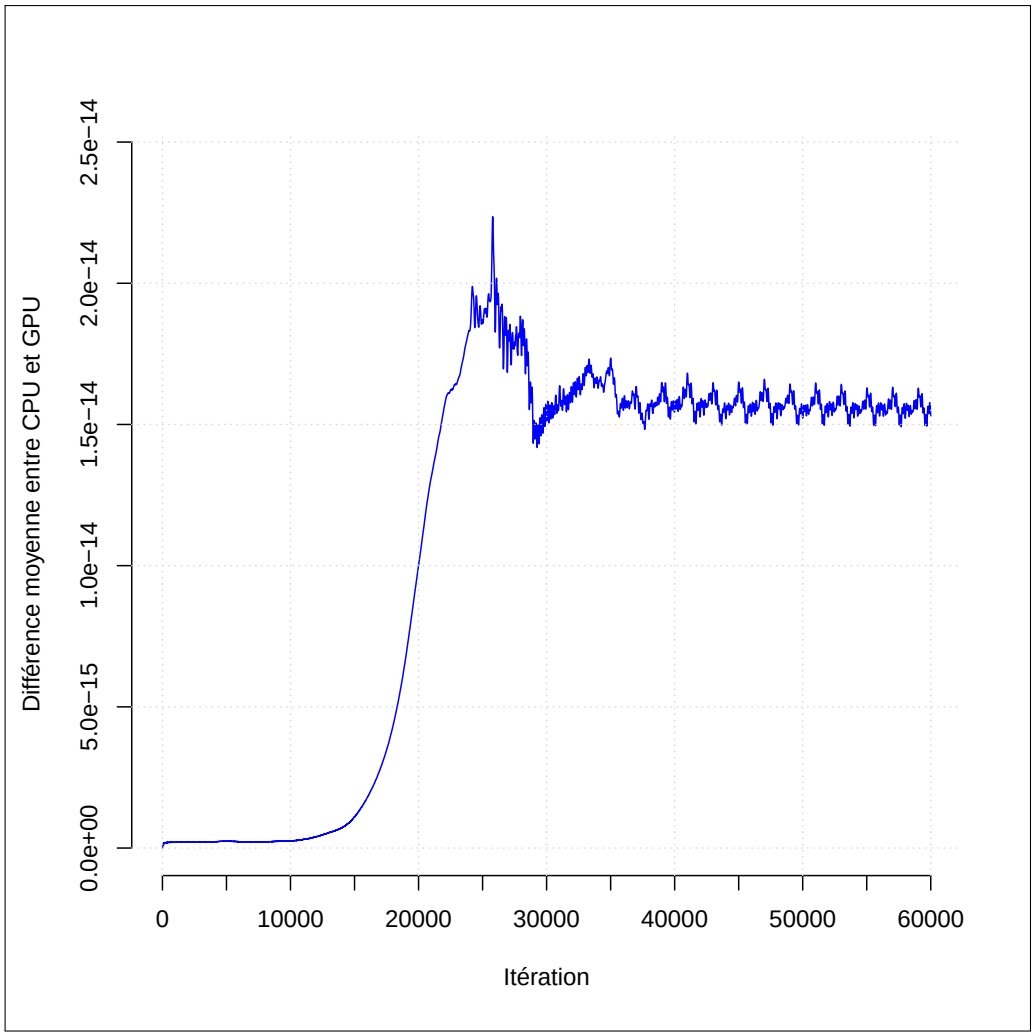
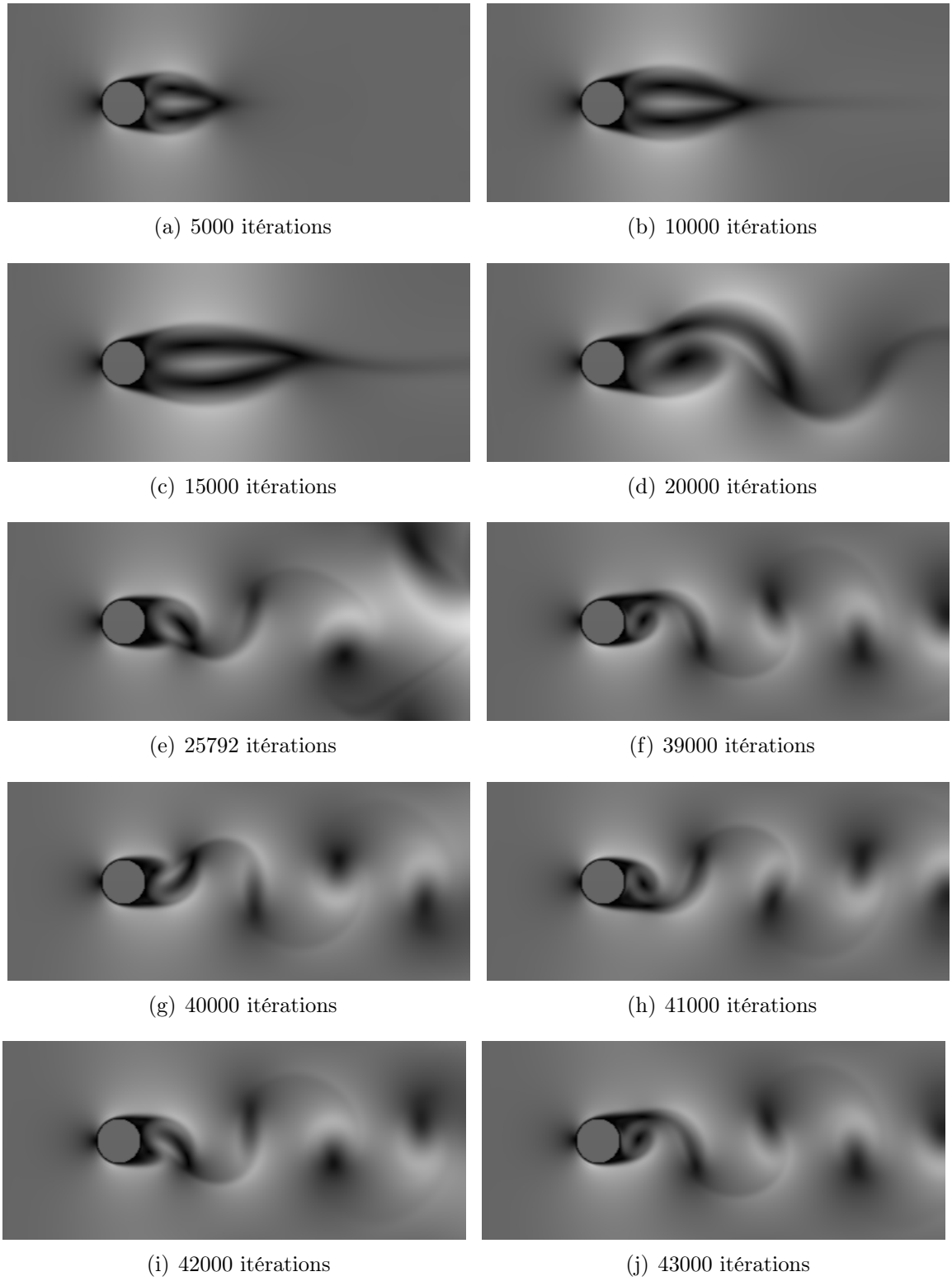


FIGURE 2.1 – Différences moyennes (par itération) entre les calculs sur CPU et GPU

compilateur optimise ce type de calculs [1] avec la fonction Fused Multiply-Add (FMA) qui réalise $x \times y + z$ en une seule opération. Le calcul est ainsi plus rapide et précis (puisqu'un seul arrondi est effectué).

Comme le souligne la documentation de Cuda, cette optimisation n'est pas forcément souhaitable dans certaines circonstances et peut être évitée avec l'utilisation de fonctions intrinsèques [1, 3].

Cette précaution force le GPU à effectuer les multiplications et additions indépendamment et permet de s'assurer que les calculs réalisés sur CPU et GPU produisent les mêmes résultats.



(a) 5000 itérations (b) 10000 itérations

(c) 15000 itérations (d) 20000 itérations

(e) 25792 itérations (f) 39000 itérations

(g) 40000 itérations (h) 41000 itérations

(i) 42000 itérations (j) 43000 itérations

FIGURE 2.2 – Simulation LBM d'un fluide autour d'un cylindre

Chapitre 3

État de l'art

Chapitre 4

Conclusion

Bibliographie

[1] Cuda Toolkit Documentation, Controlling Fused Multiply-add. http://docs.nvidia.com/cuda/floating-point/#controlling-fused-multiply-add (page consultée le 05/03/2017).

[2] Cuda Toolkit Documentation, Cuda and Floating Point. http://docs.nvidia.com/cuda/floating-point/#cuda-and-floating-point (page consultée le 05/03/2017).

[3] Cuda Toolkit Documentation, Double Precision Intrinsics. http://docs.nvidia.com/cuda/cuda-math-api/group__CUDA__MATH__INTRINSIC__DOUBLE.html (page consultée le 05/03/2017).