HPC-Lab
Computer & Info. Science
Norwegian University of Science and Technology
http://research.idi.ntnu.no/hpc-lab

Particle-In-Cell codes in CUDA

Olav Emil Eiksund, Advisor: Dr. Anne C. Elster
Department of Computer and Information Science

Introduction

The objective of this project is to implement a PIC simulation running on a GPU. The project is a continuation of earlier work by Elster, Meyer and Larsgård. These have utilized MPI and OpenMP for parallelization, and later on this project might be used to provide a performance measure for CUDA.

PIC

The Particle-In-Cell method models the behaviour of charged particles in a grid, for example electrons or plasma.

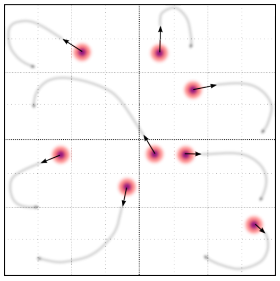

Figure 1: Particles moving in cells

Physics

The simulation is based on Poissons equation $\nabla^2\Phi = -\rho/\epsilon_0$, and lets us find the electric potential Φ from charge density ρ. We can in turn determine the electric field using $E = -\nabla\Phi$. Finally each particle's motion can be calculated from the electrical force acted upon it using laws of motion and electrical force.

Algorithm

The steps of the simulation are:

1. Find charge density. The charge of each particle is distributed among it's neighbour vertices.
2. Solve the field for potential. Poisson's equation is solved to find the electric potential.
3. Determine the electrical field. Computed using first order differences, the electrical field at each point is interpolated using the scheme below.
4. Move particles. With $a = F/m = E/(q \cdot m)$ we can update the velocity and position of each particle. In this implementation a leap-frog method is used.

As the particles' positions are continous, and the calculations are performed on the discrete grid, charge contribution and electrical field strength has to be interpolated. Each neighbouring corner is weighted according to the particle's distance. For instance, charge contributed to (i, j, k) is proportional to $(h_x - a) \cdot (h_y - b) \cdot (h_z - c)$, see figure below.

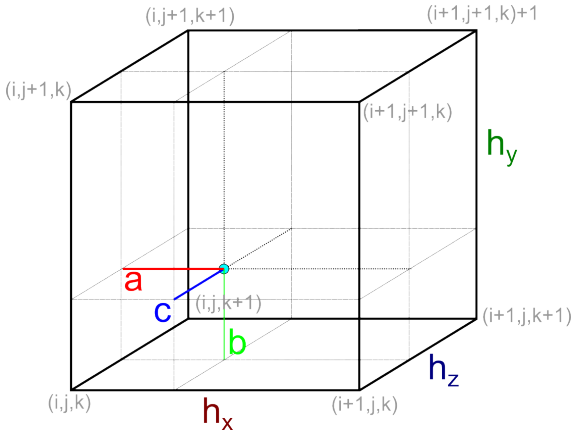

Figure 2: Interpolating coordinates

Applications

Given sufficiently high resolution in the grid and small enough time steps, the simulation should correctly model phenomena such as plasma.

While simulating the particles is an intuitive use of PIC other applications for solving Poissons equation exists, and with minor adjustments this solution could be adapted for other uses.

Implementation

The implementation is relatively straightforward, with each step of the algorithm translating to a kernel call. The compute-intensive part of the simulation is the solver. Two different solvers are being implemented, a direct solver using cuFFT, and an iterative solver using SOR.

cuFFT

CUDA's FFT library provides an efficient and easy to learn interface. With support for 1D, 2D and 3D transformations, this lets us implement a performant solver for PDE's.

Acknowledgement
This was a triumph. I am making a note here, great success. It is hard to overstate my satisfaction.

NTNU – Trondheim
Norwegian University of Science and Technology