Particle-In-Cell Monte Carlo Collision model on GPU (Graphics Processing Units) - Application to a low temperature magnetized plasma

J. Claustre [1,*], M. Paulin[2], B. Chaudhury[1], G. Fubiani[1], J.P. Boeuf[1]

[1] *Université de Toulouse; UPS, INPT, CNRS; LAPLACE, F-31062 Toulouse, France.*
[2] *Université de Toulouse; UPS, INPT, CNRS; IRIT, F-31062 Toulouse, France.*

(*) claustre@laplace.univ-tlse.fr

The particle in cell algorithm for the simulation of charged particle kinetics in plasmas is a very resource consuming method. In order to improve the computation time we have developed a PIC MCC (Particle-In-Cell Monte Carlo Collisions) model on GPU (Graphics Processing Units) where speedups reach from 10 to 30 compared with a sequential code. We describe how each part of the PIC MCC model is implemented on the GPU and show how particles are dynamically managed. The computational costs of each part of the PIC MCC model on the GPU are compared with those of a standard PIC MCC model running on a single CPU. The results are illustrated with the example of plasma transport across a magnetic filter similar to that of a negative ion source.

The PIC MCC method[1] is a very powerful tool to study low temperature plasmas since it can provide the space and time evolution of the charged particles velocity distribution functions under the effect of self-consistent fields and collisions. In an electrostatic problem, the method consists in following the trajectories of a representative number of charged particles, electrons and ions, in phase space, and to describe the collective interaction of the particles by solving Poisson's equation as the particles move. In a low temperature plasma, the trajectories in phase space are determined by the self-consistent electric field and by collisions with neutral atoms or molecules and, for large enough plasma densities, by collisions between charged particles. The computational cost of the method is very high in terms of CPU and memory resources, especially when multidimensional geometry must be taken into account. This is because of the constraints (at least in explicit PIC simulations) on the time step (smaller than a fraction of the plasma period or inverse of the electron gyrofrequency), on the grid spacing (on the order of the Debye length), and on the number of particles per Debye length in the simulation (larger than several tens).
The PIC MCC algorithm can be parallelized on CPU clusters (the treatment of particles trajectories is easy to parallelize, but the parallelization of Poisson's equation is not straightforward). The emergence of GPGPUs (General Purpose computing on Graphics Processing Units) in scientific computing opens the way to low cost massive parallelization by using the large number of processors of a graphics card to perform elementary calculations (eg computation of electron trajectories) in parallel. A number of numerical tools for GPU computing have been developed in the last 10 years. Furthermore, NVIDIA has developed a programming environment called CUDA (Compute Unified Device Architecture) that allows to create efficient GPU codes. A few recent papers[2-4] report on the development of PIC MCC simulation codes on GPUs.
In this paper we describe the implementation of a PIC MCC simulation code on GPU in the CUDA environment. CUDA is a programming environment that has been developped to facilitate scientific computation on GPUs without taking into account the complex details of GPUs (a basic knowledge of the GPU architecture is however necessary for effective programming). The GPU can be seen as a device composed of a few hundred thread processors. Each of them can manage a set of concurrent threads grouped in so called thread multiprocessor. Threads are simultaneously launched for parallel execution by group of 32 threads called warp. From a function, called a kernel, these threads will execute independently the same operations on different data. Another important concern is memory management. Several different memory spaces can be accessed by threads on the device, but the access time can differ by one or two orders of magnitudes. Typically, to access the global memory, which is equivalent to the RAM memory on the CPU, a thread will take around $400 - 600$ clock-cycles while it will take about 4 clock-cycles to access the shared memory which is a 32 kB size memory located in each thread multiprocessor. In addition, because many threads access the device

memory, the memory is divided into banks to achieve high bandwidth. It is important to note that the way in which threads access the memory has a non-negligible part in kernel execution and latency.
We will describe in this paper how each part of the PIC MCC model (particle mover, charge assignement, Monte Carlo collisions, Poisson solver) has been implemented in the CUDA environment.
With this implementation, the simulation is 10 to 30 times faster than on a single CPU.

Figure 1 is an example of simulation result showing the plasma density and electron temperature in a 10 cm x 10 cm chamber in H_2 for a gas density of 5. 10^{19} m^{-3}. Electron are supposed to be heated uniformly in a region of 3 cm x 10 cm on the left of the chamber. The plasma expands to the right side of the chamber through a magnetic filter (Gaussian shape, 3 mT maximum, centered at x=8 cm and uniform along the y direction). The non uniformity of the plasma along the y direction after the magnetic filter is due to the presence of a combination of diamagnetic and ExB current perpendicular to the walls, as described in [5].

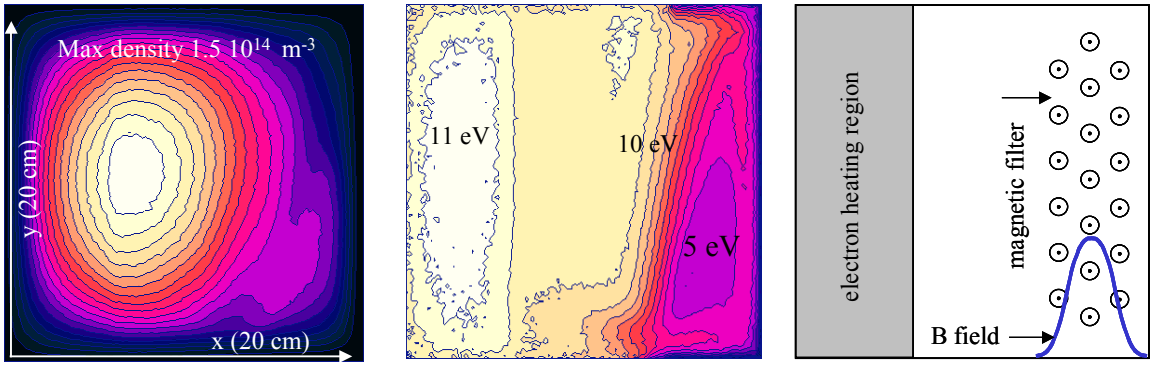

Fig. 1: (a) Plasma density, and (b), electron temperature distribution in a low temperature plasma source in H_2 (gas density 5 10^{19} m^{-3}); the 2D Cartesian geometry, electron heating region and magnetic field distribution are shown in (c). Electron heating is assumed to be uniform in the left part of the source (input power 10 W/m) . The magnetic field is perpendicular to the simulation domain (maximum 3 mT). The chamber walls are grounded. The computational grid is 128x128.

Acknowledgments

This work has been perfomed in the frame of the MOSITER project of the University of Toulouse and of the FR-FCM (Fédération nationale de Recherche Fusion par Confinement Magnétique – ITER)

References

[1] C.K. Birdsall, Particle-In-Cell charged-particle simulation, plus Monte Carlo collisions with neutral atoms, PIC-MCC, IEEE Trans. Plasma Sci. **19**, 65 (1991)
[2] P. Abreu, R. A. Fonseca, J. M. Pereira et al., PIC codes in new processors: A full relativistic PIC code in CUDA-enabled hardware with direct visualization, IEEE Trans. Plasma Sci. **39**, 675 (2011)
[3] V.K. Decyk and T.V. Singh, *Adaptable Particle-In-Cell algorithm for graphical processing units*, Computer Physics Communications **182**, 641 (2011)
[4] G.Stantchev, W.Dorland, and N.Gumerov, *Fast parallel particle-to-grid interpolation for plasma PIC simulation on GPU*, Journal of Parallel and Distributed Computing **10**, 1339 (2008)
[5] St. Kolev, G. Hagelaar, G Fubiani and J.P. Boeuf, Physics of magnetic barrier in low temperature bounded plasmas - insight from particle-in-cell simulations, to appear in PSST (2012)