High performance collisional PIC plasma simulation with modern GPUs

Alexey A Romanenko 1 and Alexey V Snytnikov 1,2,4 and Galina G Lazareva1,2,3

1 Novosibirsk State University
2 Institute of Computational Mathematics and Mathematical Geophysics SB RAS
3 Peoples’ Friendship University of Russia (RUDN University), Moscow, Russian Federation
4 Institute of Automation and Electrometry SB RAS

E-mail: lazareva@ssd.sscc.ru, snytav@ssd.sscc.ru, arom@ccfit.nsu.ru

Abstract. GPU implementation of the collisional Particle-In-Cell plasma model is proposed. The goal of this model is the simulation of Plasma Enhanced Chemical Vapour Deposition (PECVD) reactors with glow discharge plasma and also for simulation of fusion plasma with sufficient role of collisions. Vlasov equation is being solved by the Particle-In-Cell (PIC) method. Collision are simulated with the null collision technique. The important difference from the collisionless approach is that major time is taken by collision evaluation and not by particle push.

GPU performance is increased by storing the particles in cells and also by the exclusion of synchronization of threads at the stage of current and density evaluation. Electromagnetic field is given by either the Maxwell equations or be the Poisson equation for electrostatic case. The Maxwell equations are solved by the FDTD method. The Poisson equation is solved by the hybrid method. Performance obtained at the moment is 0.5 TFLOPS for particle push with the Nvidia Tesla V100, and the parallels efficiency is 92% for a cluster with 250 nodes with 2 GPUs each resulting in 500 GPUs.

Acknowledgments
The publication has been prepared with the support of the ”RUDN University Program 5-100”, code development was supported by RFBR grant 18-07-00364, the numerical model was designed by RFBR grant 19-07-0085, computational experiments were performed under support of RFBR grant 19-07-00446, development of the collision model was supported by RFBR grant 19-07-00446.

1. Introduction
The main idea of this work is to achieve high performance with supercomputers and GPUs in order to conduct high resolution 3D simulations of the fusion plasma in different experimental facilities and also in astrophysical applications like supernovae expansions and solar bursts of II and III type. Although fusion plasma is geneally treated as being collisionless, there are cases when collisions might play a role. Moreover, we intend to use the same Particle-In-Cell engine for low temperature plasma problems, too. An important example of such a problem is the Plasma Enhanced Chemical Vapour Deposition (PECVD) reactor.
Since two problems: fusion reactor and PECVD reactor are treated with the same code, there are two filed solvers, one for Maxwell equations and the other one for Poisson equation. Exactly the same particle pusher is used for both cases. Like other plasma physics problems with non-Maxwellian velocity distribution this problem requires kinetic treatment.

It means either direct finite-difference solution of Vlasov equation or Particle-In-Cell[1] method. While PIC method is noisy and time-consuming, still it is in many cases the only method that is able to treat the problem with no non-physical simplifications.

There are a lot of PIC packages, like OSIRIS[2], LCODE[3] or VLPL[4], including those adopted for GPU, e.g. PICconGPU[5], ALAdyn[6], and also some works aimed to particular questions of GPU implementation of PIC method, like [7] and [8], but since they are all problem-specific, we make an implementation on our own.

PIC method consists of two stages: field evaluation and particle pushing. Computational experiments with general purpose computers (e.g. the Lomonosov supercomputer at MSU) show that for the collisionless case most time is spent in particle pushing (from 60% to 90%). Since the total time is very big even for a small test case: about a week with 128 Intel Xeon cores for a 3D problem with just 100 × 4 × 4 mesh nodes and 10000 particles for each cell [9], it requires hiring GPUs to solve real-size problem.

2. Basic equations
The basic equations are Vlasov equations for ion and electron components of the plasma and also of the Maxwell equation system. These equations in the usual notation have the following form:

\[ \frac{\partial f_{i,e}}{\partial t} + \bar{v} \frac{\partial f_{i,e}}{\partial \bar{r}} + \bar{F}_{i,e} \frac{\partial f_{i,e}}{\partial p} = St\{f\}, \quad \bar{F}_{i,e} = q_{i,e} \left( \bar{E} + \frac{1}{c} [\bar{v}, \bar{B}] \right) \]

\[ rot \bar{B} = \frac{4\pi}{c} \bar{j} + \frac{1}{c} \frac{\partial \bar{E}}{\partial t} \]

\[ rot \bar{E} = -\frac{1}{c} \frac{\partial \bar{B}}{\partial t} \]

\[ \nabla^2 \varphi = 4\pi \rho \]

\[ div \bar{B} = 0 \]

Here \( f_{i,e} \) is the distribution function for ions and electrons, respectively, \( t \) is time, \( \bar{v} \) is the velocity vector, \( \bar{r} \) is the coordinate vector, \( \bar{F} \) is the Lorentz force, \( \bar{p} \) is the impulse, \( \bar{E} \) is the electric field, \( \bar{B} \) is the magnetic field, \( \varphi \) is the electric potential, \( \rho \) is the charge density. Finally, \( St\{f\} \) is the collision term.

All the equations below are given in non-dimensional units. The following quantities are used for transition to non-dimensional form:

- velocity of light \( c = 3 \times 10^{10} \) cm/s
- mass of electron \( m_e = 9.1 \times 10^{-28} \) g
- plasma density \( n_0 = 10^{14} \)
- time \( t = \omega_{pe}^{-1} \), where plasma electron frequency \( \omega_{pe} = 5.6 \times 10^{11} \) s\(^{-1}\).

3. Computational domain
The 3D computational domain has the shape of a cube with the following dimensions:

\[ 0 \leq x \leq L_X, \quad 0 \leq y \leq L_Y, \quad 0 \leq z \leq L_Z \]
Within this domain there is the model plasma that consists from electrons and ions. The model plasma particles are distributed uniformly within the domain. The density of plasma is set by the user as well as the electron temperature. The temperature of ions is considered to be zero. Initial distribution of particles by velocities is Maxwellian:

\[ f(v) = \frac{1}{\Delta v \sqrt{2\pi}} \exp\left(\frac{-(v - v_0)^2}{2\Delta v^2}\right) \]

here \( \Delta v \) - electron velocity dispersion, \( v_0 \) - average velocity (\( v_0 = 0 \) for plasma particles. Up to this moment it is the same for fusion plasma and for PECVD reactor plasma. But there are also differences. It is about the composition of plasma and the boundary conditions.

3.1. Fusion reactor plasma simulation
It this case the boundary conditions are periodical, and plasma consists of three sorts of particles:

- Hydrogen ions,
- plasma electrons,
- beam electrons.

Beam electrons are also distributed uniformly, but they have the same impulse directed along X axis and low temperature along Y and Z.

3.2. PECVD reactor simulation
In this case there is a periodic electric potential applied at \( z = L_z \) and \( \phi = 0 \) at all the other boundaries. At \( z = 0 \) there is substrate with amorphous silicon hydrid film deposed from plasma. In this case plasma consists of

- Hydrogen ions
- electrons
- \( \text{SiH}_2^+ \), \( \text{SiH}_3^- \) ions

4. Fusion reactor plasma simulation
The physical statement of the problem exactly follows the simulations in [9], the difference is in simulation time which in the present case is by orders of magnitude smaller. As it is stated in [9] beam relaxation may proceed in three modes, depending on the instability increment \( \gamma \) and beam velocity dispersion \( \Delta v \): kinetic (\( k\Delta v \gg \gamma \)), hydrodynamical (\( k\Delta v \ll \gamma \)) and transit mode (\( k\Delta v \sim \gamma \)). Here \( k \) is the wavenumber.

The goal of the computations is not to simulate the facility as a whole - since no supercomputer is able to do it now or in the future. Instead, we are going to reveal the underlying instabilities and reproduce the turbulence observed in experiments.

In such a way, the computational domain corresponds to a very small (characteristic size \( \sim 10^{-3} \text{ cm} \)) volume of plasma.

5. Description of the GPU implementation
The Vlasov equation is solved by the Particle-In-Cell method [1]. In order to understand the ideas and bottlenecks of GPU implementation of PIC method let us first consider the underlying numerical techniques.
5.1. Numerical method
The essential of the PIC method is the substitution of the Vlasov equation by the set of characteristic equations. The characteristic equations exactly resemble the Newton motion equations for plasma particles, and due to this reason they are called the motion equations for model particles.

One must note that these model particles are just mathematical tool for solving the Vlasov equation. This is not a replacement of very big number ($N \sim 10^{23}$) of small particles in real plasma by a small number ($N \sim 10^{10}$) of big particles in model plasma.

5.1.1. Particle push
In such a way, the equations of motion for model particles are the following:

\[
\frac{d\vec{p}_{i,e}}{dt} = q_{i,e} (\vec{E} + \vec{v} \times \vec{B}), \quad \frac{d\vec{r}_{i,e}}{dt} = \vec{v}_{i,e}, \quad \vec{p}_{i,e} = \frac{\vec{v}_{i,e}}{\sqrt{1 - v_{i,e}^2}} \tag{2}
\]

here \(i,e\) are the indices for ions and electrons, respectively, \(\vec{p}\) is impulse, \(\vec{r}_{i,e}\) is the coordinate vector, \(\vec{E}\) - electric field, \(\vec{v}\) is the velocity and \(\vec{B}\) - magnetic field. Equations 2 are solved by the leapfrog scheme:

\[
\frac{\vec{p}_{\alpha}^{m+1/2} - \vec{p}_{\alpha}^{m-1/2}}{\tau} = q_{i,e} \left( \vec{E}_{\alpha}^{m} + \frac{\vec{v}_{\alpha}^{m+1/2} - \vec{v}_{\alpha}^{m-1/2}}{2} \times \vec{B}_{\alpha}^{m} \right)
\]

\[
\frac{\vec{r}_{\alpha}^{m+1} - \vec{r}_{\alpha}^{m}}{\tau} = \vec{v}_{i,e}^{m+1/2}
\]

here \(\tau\) is the timestep.

5.1.2. Field solver

**Maxwell equations** For the evaluation of electromagnetic field the Langdon-Lasinski [1] method is employed:

\[
\frac{\vec{B}_{m+1/2}^{m+1/2} - \vec{B}_{m-1/2}^{m-1/2}}{\tau} = \text{rot}_h \vec{E}_{\alpha}^{m}
\]

\[
\frac{\vec{E}_{m+1}^{m+1} - \vec{E}_{m}^{m}}{\tau} = \text{rot}_h \vec{B}_{m+1/2}^{m+1/2} - \vec{J}_{m+1/2}
\]

**Poisson solver** The method for solving the 3D Poisson equation is a mixture of FFT along \(Y\), the Thomas elimination along \(X\) and Block Successive Over-Relaxation (BSOR) along \(Z\). Is is shown that BSOR iteration works faster than even the FFT if the right hand side (charge density) varies not very much. One can see in figure 1 that when BSOR makes just one or two iterations, BSOR solution time is less than FFT solution time. The algebraic multigrid method was also applied in the Poisson solver [10] instead of BSOR.

5.1.3. Collision simulation
The collisions of electrons with gas molecules are treated with the null collision technique [11]. First, for the given electron, the gas is defined (considering two gases with the densities \(n_1\) and \(n_2\)) by means of the random number \(t\), \(0 < t < 1\). If the condition

\[
t < \frac{n_1}{n_1 + n_2}
\]

is valid, then the first gas is taken for collision, otherwise the second. In the same way, one of the possible collisional processes is selected (the processes are listed in Section 4). Since the
Figure 1. The number of iterations of the BSOR method depending on the simulation timestep number. The horizontal line shows the FFT solution time divided by a single iterations time.

Type of the process is known, it is possible to find its cross-section $\sigma$. The cross-section is used to compute the collision probability

$$p = 1 - \exp(\nu \tau n).$$

Here $v$ is the electron velocity, $\tau$ is the timestep and $n$ is the gas density. By means of a random number, the collision probability defines either this electron superparticles suffer a collision or not.

5.2. GPU implementation principles

The following main principles are used to achieve high GPU performance:

- shared memory is used for computations with particles;
- particles are stored in arrays attached to cells in order to use the cache more efficiently;
- field and current values (8 of them for each field and current component) are computed and stored in cells;
- particle reordering is used to speedup particle pusher.

The computational algorithm consists of the following stages performed at each time step:

- particle push, that is the computation of new coordinates and impulses of each particle. At present, the evaluation of current is included in the push, [12, 13];
- electromagnetic field evaluation, [14];
- reordering of particle, which means the transfer of particles to the cells they belong to [15];
- copying of currents from cells to the global arrays [14];
- copying of fields from global arrays to cells [14].

For each of the stages a special kernel is devoted in GPU implementation. The implementation of each kernel is given in more detail in the above-listed papers.

Random values needed for PIC-MC simulation were evaluated with cuRand library.
6. Performance measurements with different GPUs

The results of profiling the code execution are presented in the table 1.

Performance is evaluated in the following way: the computational domain has 6.4 million particles, and near 500 floating point operation are required to push a particle at a time step. The total number of operations is divided to the measured pushing time. As pushing takes the most time (from 60% to 95%), only push performance is given. Field solution time is given for both Poisson equation (solved by hybrid method) and for the Maxwell equations solved by FDTD. This does not mean that both solvers were used in one simulation. The Poisson equation was solved for PECVD simulation and The Maxwell equation – for fusion simulation, and the same pusher was used for both of them.

Table 1. Worktime, measured by the Nvidia Visual Profiler and the resulting performance in flops for different GPUs, time in microseconds. Grid size $100 \times 6 \times 6$ with 1000 particles in cell.

| Device name | Field evaluation time | Pushing time | Collision time | Push performance |
|-------------|-----------------------|--------------|----------------|------------------|
|              | Poisson               | Maxwell      |                |                  |
| Geforce GTX 1050 Ti | 166.2 ms            | 20 ms        | 225.4 ms       | 20.1 ms          | 14.1 GFLOPS     |
| Tesla K40m   | 280 ms                | 36 ms        | 434.4 ms       | 38.3 ms          | 7.3 GFLOPS      |
| Tesla V100   | 4 ms                  | 0.5 ms       | 6.5 ms         | 2 ms             | 0.492 TFLOPS    |

A part of the profile for the NVidia Tesla V100 is shown in figure 2. One can see the device name (Tesla V100 SXM2) as well as kernel duration (6.58132 ms). It can be noticed also that there is no big pause between the kernels, so the device is busy most of the time.

7. Conclusion

The performance achievement of nearly 0.5 TFLOPS by a single NVidia Tesla V100 for 3D Particle-In-Cell simulation is presented. It means that now it is possible to perform the simulation of plasma of the size mentioned in Introduction in 4 hours with one V100 instead of a week with a ordinary cluster of 128 cores (the cluster at Siberian Supercomputer Center, based on rather old Intel Xeon 5540 4-core processors). It is also shown that the Plasma Enhanced Chemical Vapour Deposition reactor could be also simulated within reasonable time.
References

[1] Vshivkov V A, Grigoryev Y N and Fedoruk M P 2002 Numerical Particle-in-Cell Methods. Theory and applications (Utrecht-Boston: VSP)

[2] Fonseca R A, Silva L O, Tsung F S, Decyk V K, Lu W, Ren C, Mori W B, Deng S, Lee S, Katsouleas T and Adam J C 2002 Computational Science — ICCS 2002 ed Sloat P M A, Hoekstra A G, Tan C J K and Dongarra J J (Berlin, Heidelberg: Springer Berlin Heidelberg) pp 342–351 ISBN 978-3-540-47789-1

[3] Lotov 2012 (accessed August 11, 2018) LCODE main site URL http://www.inp.nsk.su/~lotov/lcode/

[4] Tuckmantel T, Pukhov A, Liljo J and Hochbruck M 2010 IEEE Transactions on Plasma Science 38 2383–2389

[5] Burau H, Widera R, Honig W, Juckeland G, Debus A, Kluge T, Schramm U, Cowan T E, Sauerbrey R and Bussmann M 2010 IEEE Transactions on Plasma Science 38 2831–2839

[6] Rossi F, Londrillo P, Sgattoni A, Sinigardi S and Turchetti G 2012 AIP Conference Proceedings 1507(1) 184–192 (Preprint https://doi.org/10.1063/1.4773692) URL https://doi.org/10.1063/1.4773692

[7] Kong X, C Huang M, Ren C and K Decyk V 2011 230 1676–1685

[8] Rieke M, Trost T and Grauer R 2015 Journal of Computational Physics 283 436–452 (Preprint https://doi.org/10.1016/j.jcp.2014.12.016) URL https://doi.org/10.1016/j.jcp.2014.12.016

[9] Lotov K V, Timofeev I V, Mesyats E, Snytnikov A V and Vshivkov V A 2015 Physics of Plasmas 22(2) (Preprint https://doi.org/10.1063/1.4907223) URL https://doi.org/10.1063/1.4907223

[10] Snytnikov A V and Vshivkov V A 2005 Proceedings of the 8th International Conference on Parallel Computing Technologies PaCT’05 (Berlin, Heidelberg: Springer-Verlag) pp 457–467 ISBN 3-540-28126-6, 978-3-540-28126-9

[11] Birdsall C K 1991 IEEE Transactions on Plasma Science 19 65–85 ISSN 0093-3813

[12] Snytnikov A and Romanenko A 2014 Computer Science 36 79–88

[13] Snytnikov A and Romanenko A 2013 Numerical Analysis 16 81–92

[14] Snytnikov A V, Boronina M A, Mesyats E A and Romanenko A A 2015 Russian Supercomputing Days 2015 (MSU)

[15] Romanenko A A and Snytnikov A V 2018 Vestnik Udmurtskogo Universiteta