Monte Carlo Collision method for Particle-In-Cell plasma simulation: PyTorch implementation

Alexey Romanenko¹ and Alexey Snytnikov²,³ and Thibault Lemaire⁴ and Pierre Masson⁴

¹ Novosibirsk State University
² Institute of Automation and Electrometry SB RAS
³ Novosibirsk State Technical University
⁴ CY Cergy Paris University

E-mail: arom@nsu.ru, snytav@gmail.com, lemaireth@cy-tech.fr, massonpie@cy-tech.fr

Abstract. The glow discharge in silane-hydrogene plasma is simulated by the Particle-In-Cell method. For the sake of simplicity just two collisional processes are considered. The implementation of null collision technique with PyTorch library is presented. PyTorch provides portability, flexibility and high precision of computations. The mathematical basics of the technique are given, and the Python code is listed. The diagnostics that help to understand the physical process under study as well as the correctness of the simulation are also given: energy distribution function, the number of electron collision events and the number of model particles.

1. Introduction
The simulation of collision-driven plasma, in particular, various discharges, is a wide and dynamic area of computational physics. Let’s mention just a few recent works. In [1] inductively coupled radio-frequency plasma in chlorine is investigated via a global model, [2] considers two-temperature, multi-fluid model of a plasma in stagnation flow against a cooled, electrically biased surface. An extremely important question is the comparing of simulation and experimental results. This is done in [3]. The work [4] characterizes N₂O fed plasma devices working in various configurations, as plasma reac-tors, hollow cathodes, and plasma thrusters, by means of a Global (volume averaged) Model (GM).

The interest to silane glow discharge [5], [6] is due to the widespread use of amorphous silicon films [7], [8] in microelectronics. An electric glow discharge is a type of plasma formed by passing a current at 100 V to several kV through a gas, usually argon or another noble gas [9].

Due to the non-Maxwell character of the electron and the ion energy distribution, the Particle-in-Cell plus Monte Carlo (PIC/MCC) method is frequently used to simulate these discharges.

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
Figure 1. Velocity distribution for model particles in nondimensional units.

form:

$$
\frac{\partial f_{i,e}}{\partial t} + \vec{v} \frac{\partial f_{i,e}}{\partial \vec{r}} + \vec{F}_{i,e} \frac{\partial f_{i,e}}{\partial \vec{p}} = St\{f\},
$$

$$
\vec{F}_{i,e} = q_{i,e} \vec{E}.
$$

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

We use the Poisson solver to evaluate electric field.

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.

The distribution of velocities of model particles is shown in figure 2 in non-dimensional units. Here velocity value 1.0 corresponds to $10^4$ cm/sec. For the convinience of future use it is necessary to have this distribution in energy units, namely, in electron-volts. This is because the energy of model particles is used to compute collision cross sections, and the cross section data are given usually in electron-volts, e.g. [10]. The distribution in electron-volts is shown in figure 2.
Figure 2. Velocity distribution for model particles in nondimensional units.

4. Particle pusher
The Vlasov-Liouville equation is solved by Particle-in-Cell method [11, 12, 13]. Particle push was described in [14], the implementation generally follows the scheme given in [12].

All the computations are performed in real physical units. Using dimensional units we might loose some precision compared to computing in non-dimensional units, though it is not very important with modern computers. On the other hand, using dimensional units makes the computation much more transparent and greatly facilitates the analysis of the results.

The PyTorch implementation of the pusher is available as Push bitbucket repository.

5. Monte Carlo collision simulation
5.1. Numerical technique
The collisions of electrons with gas molecules are treated with the null collision technique [15]. 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

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

then the first gas is take for collision simulation, otherwise the second. This formula is easily generalized for the case of more than two gases. Here we work with the mixture of silan and hidrogene.

In the same way, one of the possible collisional processes is selected. Here, for the sake of simplicity, we consider just two collision processes: ionization (cross section \( \sigma_{\text{ion}} \)) and elastic scattering ((cross section \( \sigma_{\text{el}} \))). If the random number \( t \):

\[
t < \frac{\sigma_{\text{ion}}}{\sigma_{\text{ion}} + \sigma_{\text{el}}}
\]

Then ionization happens, if this condition is false, then elastic scattering. This formula is also easily generalized for a set of collision processes.

Such a small section of collision processes, or plasma-chemical reactions is, of course, unrealistic. We use these two reactions just as an example. The set of reactions considered in the model could be easily extended for the simulation of some particular plasma device or plasma process in real nature.
Since the type of the process is known, it is possible to find its cross-section. The cross-section is used to compute the collision probability

\[ p = 1 - \exp(-v\tau\sigma n) \]  

(4)

here \( v \) is the velocity of the model particle, \( \tau \) is the timestep, \( \sigma \) is the cross section and \( n \) is the gas density. The cross sections are taken from [10].

For both ionization and elastic scattering new values of the model particles velocity are evaluated.

Let the velocity of a model particle be \( \vec{v} = (v_x, v_y, v_z) \) before collision. Then the absolute value of the velocity vector as well as its components have the following form:

\[
|v^1| = \begin{cases} 
1 + (1 - \cos \theta), & t < 0.5 \\
1 - (1 - \cos \theta), & t > 0.5 
\end{cases}
\]

\[ v_x^1 = |v^1| \left( \cos \theta \cos \alpha + \sin \theta \sin \varphi \sin \alpha \right), \]

\[ v_y^1 = |v^1| \left( \cos \beta \left( -\cos \theta \sin \alpha + \sin \theta \sin \varphi \cos \alpha \right) + \sin \beta \sin \theta \cos \varphi \right), \]

\[ v_z^1 = |v^1| \left( \cos \beta \left( -\cos \theta \sin \alpha + \sin \theta \sin \varphi \cos \alpha \right) + \cos \beta \sin \theta \cos \varphi \right) \]

(5)

here \( t \) is a random number with uniform distribution, \( 0 < t < 1 \), and \( \varphi \) is a random number with the uniform distribution, \( 0 < \varphi < 2\pi \). The scattering angle \( \theta \) is defined as

\[ \theta = \sqrt{1 - \tau \ln(1 - x)} \]  

(6)

here \( x \) is a random number with uniform distribution, \( 0 < x < 1 \). The values \( \alpha \) and \( \beta \) are the angles between the initial velocity vector \( \vec{v} \) and \( XY \) and \( XZ \) planes, respectively:

\[
\cos \alpha = \frac{v_x}{\sqrt{v_x^2 + v_y^2}}, \quad \sin \alpha = \frac{v_y}{\sqrt{v_x^2 + v_y^2}} \\
\cos \beta = \frac{v_y}{\sqrt{v_x^2 + v_y^2}}, \quad \sin \beta = \frac{v_z}{\sqrt{v_x^2 + v_y^2}} 
\]

(7)

Moreover, for ionization, a new model particle (electron) is created.

The described combination of the PIC method with a null collision technique is often referred to as PIC/MCC method (Particle-in-Cell plus Monte Carlo Collision), [10].

5.2. PyTorch implementation

Here let’s consider the PyTorch implementation of the null collision technique. The following listing shows its initial part. The most important is that the code is fully vectorized (it has no loops), and is easily converted to GPU.

```
#PyTorch with different types of collision
import torch

N = 20000 #number of model particles

# Step 1 collision probability
V = torch.cat((torch.ones(N, 1), 1e-3*torch.ones(N, 2)), 1) #contains values of the velocities

v = torch.pow(torch.sum(torch.pow(V, 2), 1), 0.5) #velocity vector norm
```
p = torch.sub(torch.ones(1, N),
torch.exp(torch.mul(v, - tau * sigma * n)))

# collision probability
### Step 2 collision process

#### theta
theta = torch.sqrt(1 - tau * torch.log(torch.rand(N)))

# scattering angle
theta2 = theta * p
cosTheta = torch.cos(theta2)
sinTheta = torch.sin(theta2)

#### varphi
phi = torch.rand(N) * 2

# polar angle of the spherical coordinate system

cosPhi = torch.cos(phi)
sinPhi = torch.sin(phi)

#### v'

### Angle of initial particle speed

cosAlpha = torch.div(V[:, 0],
torch.sqrt(torch.pow(V[:, 0], 2) + torch.pow(V[:, 1], 2)))
sinApha = torch.div(V[:, 1],
torch.sqrt(torch.pow(V[:, 0], 2) + torch.pow(V[:, 1], 2)))
cosBeta = torch.div(V[:, 2],
torch.sqrt(torch.pow(V[:, 2], 2) + torch.pow(V[:, 1], 2)))
sinBeta = torch.div(V[:, 2],
torch.sqrt(torch.pow(V[:, 2], 2) + torch.pow(V[:, 1], 2)))

### Update individual particle speed components

V[:, 0] = v * (cosTheta * cosAlpha + sinTheta * sinPhi * sinApha)
V[:, 1] = v * (cosBeta * (cosTheta * sinApha + sinTheta * sinPhi * cosAlpha) + sinTheta * cosPhi * sinBeta)
V[:, 2] = v * (-sinBeta * (cosTheta * sinApha + sinTheta * sinPhi * cosAlpha) + sinTheta * cosPhi * cosBeta)
Figure 3. The ratio of ionization events and elastic scattering events to the total number of collision events in the simulation, depending on timestep. No particle removal at the boundaries.

Figure 4. The ratio of ionization events and elastic scattering events to the total number of collision events in the simulation, depending on timestep. Particles that reach the boundaries are removed.

6. Simulation Analysis and Diagnostics

Figure 5 shows the number of model particles in the domain depending on the timestep. One can see that the number of particles is growing rapidly, and this is a clear sign of a wrongly going process. In the glow discharge the number of particles must be stable, and such a rapid growth is more like a breakdown. The reason of this wrong behavior is quite clear: there is a source of new particles (electrons): ionization, but there’s no possibility to remove the particles from the computational domain. In reality the electrons are removed due to absorption at the electrodes and the wall of reaction chamber. Another way of removing electrons are the plasma-chemical reactions like recombination and attachment. Such reactions are not considered in the present work, but particle absorption at the boundaries should be introduced. The result is shown in figure 6. It is seen that after some oscillation the number of model particles becomes stable.
7. Conclusion

The model of silane-hydrogen plasma of glow discharge is implemented on the basis of Particle-In-Cell method with Monte-Carlo collisions. The plasma reaction set is simplified and contains just two reactions: ionization and elastic scattering. Still the model can be easily extended. The use of PyTorch library provides high portability - the code is immediately ported to GPUs and also high precision of computations.

The main result of this work is the model of glow discharge plasma that is capable of simulation of real plasma processes, provided the correct set of plasma-chemical processes and corresponding cross sections.

Acknowledgments

The publication has been prepared with the support of RFBR grant 19-07-00446, code development was supported by RFBR grant 19-07-00085.
References

[1] Kemaneci E, Carbone E, Booth J P, Graef W, van Dijk J and Kroesen G 2014 Plasma Sources Science and Technology 23 045002

[2] Meeks E and Cappelli M 1993 IEEE Transactions on Plasma Science 21 768–777

[3] Corr C, Despiau-Pujo E, Chabert P, Graham W, Marro F and Graves D 2008 Journal of Physics D: Applied Physics 41 185202

[4] Katsonis K and Berenguer C 2013 International Journal of Aerospace Engineering 2013

[5] Abolmasov S, Kroely L and Cabarrocas P 2008 Journal of Physics D 41 165203

[6] Aldim M R and Goedheer W J 2003 Journal of Applied Physics 94 104–109 URL https://doi.org/10.1063/1.1578522

[7] Byun J Y, Ji Y J, Kim K H, Kim K S, Tak H W, Ellingboe A R and Yeom G Y 2020 Nanotechnology 32 075706 URL https://doi.org/10.1088/1361-6528/abab9b

[8] Wrobel A M and Uzanski P 2021 Plasma Processes and Polymers 18 2000240 (Preprint https://onlinelibrary.wiley.com/doi/pdf/10.1002/ppap.202000240) URL https://onlinelibrary.wiley.com/doi/abs/10.1002/ppap.202000240

[9] Fridman A and Kennedy L 2004 Plasma Physics and Engineering (Taylor & Francis) ISBN 9781560328483 URL https://books.google.ru/books?id=9sqtYiy_gloC

[10] Perrin J, Leroy O and Bordage M C 1996 Contributions to Plasma Physics 36 3–49 (Preprint https://onlinelibrary.wiley.com/doi/pdf/10.1002/ctpp.2150360102) URL https://onlinelibrary.wiley.com/doi/abs/10.1002/ctpp.2150360102

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

[12] Brieda L 2019 Plasma Simulation by example (CRC Press)

[13] Birdsall C K and Langdon A B 2005 Plasma physics via computer simulation (New York: Taylor and Francis) ISBN 0750310251 9780750310253

[14] Romanenko A A, Snytnikov A V and Boronina M A 2020 Journal of Physics Conference Series (Journal of Physics Conference Series vol 1640) p 012016

[15] Birdsall C 1991 IEEE Transactions on Plasma Science 19 65–85

[16] Vahedi V and Surendra M 1995 Computer Physics Communications 87 179–198 ISSN 0010-4655 particle Simulation Methods URL https://www.sciencedirect.com/science/article/pii/00104655940011w