The numerical solution of the Vlasov-Poisson-Fokker-Planck equation in the context of accelerator physics

Master thesis for the UU Mathematical Sciences Programme

Linda Stoel, 3345653
Under supervision of Jason Frank

February 13, 2015

Abstract

We discuss the numerical solution of the Vlasov-Poisson-Fokker-Planck (VPFP) equation in the context of accelerator physics. Our experiments focus on the single particle type case, which is useful in studying intrabeam scattering. The theory and algorithms, however, are derived for the more general case with multiple species of particles, so that applications in the simulation of electron cooling, for example, are also possible.
We first review the derivation of the VPFP equation and the associated stochastic differential equations, after which we formulate a particle-in-cell algorithm to solve them. Our experiments are carried out with a one-dimensional restriction of the model, and their results are promissing enough to warrant implementation and investigation of a three dimensional version. We show that the choice of boundary conditions is of particular interest.

Contents

Introduction

In this thesis we will discuss the numerical solution of the Vlasov-Poisson-Fokker-Planck (VPFP) equation in the context of accelerator physics, using stochastic differential equations.

There are approximately 30000 particle accelerators in use around the world today. Most of these accelerators are used for medical and industrial purposes, for example as radiotherapy accelerators or as ion implanters, which are used for semiconductor device fabrication. The most famous accelerators however are the high energy accelerators used for high energy physics research, such as the Large Hadron Collider (LHC) at CERN and the Relativistic Heavy Ion Collider (RHIC) at Brookhaven National Laboratory. These high energy accelerators are built to discover more about particle physics, but in doing so they push back the frontiers of science and technology, leading to advances in various areas.

One of the effects present in accelerators is the so-called intrabeam scattering (IBS), which is the name for the effects of Coulomb scattering within the beam. Since the beam in an accelerator typically consists of many charged particles with the same charge there will be a Coulomb repulsion between the particles. This interaction can be modelled as collisions between the particles, which mostly happen at large impact parameters[1], causing only a small deflection per collision. These collisions will however add up and cause the beam to widen over time, which decreases the intensity of the beam, which is disadvantageous for the experiments that typically occur at large accelerators.

However, we need not only consider the collisions, but also the mean force field caused by all the other particles. An equation that describes both of these will be derived in the first chapter, which we will use to simulate IBS. Coulomb collisions are also important in various other areas of physics, most notably plasma physics. In this paper we will look at the presented equations relating to accelerator physics, but it may be interesting to bear in mind that it may be relevant in other fields as well.

We will mostly consider IBS in our simulation, but we will also briefly discuss the application of the presented work to electron cooling. Electron cooling is a method to decrease velocity deviations in an ion beam, by sending an electron beam though it. Since it is quite easy to make an electron beam with low velocity deviations, one can send such a beam though a section of the ion ring, keeping new electrons coming in and continually extracting the old ones out. The electrons and the ions will then interact through the Coulomb interaction, and tend towards a near-thermal equilibrium state, in which the velocity deviations of the ions are lowered. So even though electron cooling and IBS have opposite effects on the ion beam, the underlying physics is similar, and both can be described by the equations used in this thesis.

Calculating the behaviour of a charged particle beam is essentially a many-body problem with Coulomb forces between the particles. This problem is hard to solve exactly, because the number of forces grows as the square of the number of particles. Hence an approximation or computational approach is in order for the high particle numbers associated with beam physics.

Some exact approximations for the growth rate of the beam size due to IBS have been developed, however most of these assume that it is independent of position and that the beam has a certain distribution. These conditions are often not fulfilled in the actual situation, so that these exact methods can only provide estimates. To gain a better understanding of IBS, and in the future possibly electron cooling, it is therefore useful to examine the general case. Due to the computational intensity of this problem one needs to use a computer to do so.

For a charged particle beam we do not have the full conditions (the exact location and velocity of each particle) available, but rather an initial distribution for the positions and velocities of the particles.

[1]The impact parameter in scattering theory signifies the distance from the scattering target to the line through the incoming particle spanned by its initial velocity.

Moreover, one is also typically interested in the final distribution of the particles, and not in the exact position or velocity of any particular particle. Therefore it is natural to make a continuum approximation of the beam and look at the evolution of this distribution.

In Chapter 1 we will introduce such a description of the beam and show the derivation of the VPFP equation which describes the evolution of the beam. Since this equation is hard to solve in its original form we will formulate a system of stochastic differential equations (SDEs) whose solution will lead to the solution of the original equation. These SDEs describe the evolution of the position and velocity of a test particle in our beam. By tracking the evolution of many test particles and looking at their distribution, we will find the distribution we were looking for. The advantage of the SDE description compared to the original many-body problem is that the number of test particles need not be as big as the physical number of particles, and that it can be implemented such that the calculation time is linear in the number of test particles, instead of quadratic in the number of particles.

In Chapter 2 we describe an algorithm, which is a so-called particle-in-cell method, that solves the system of SDEs. This algorithm keeps track of the positions and velocities of the particles, but also uses a grid to represent continuous quantities such as the density. Forces at particle positions are extrapolated from the continuous force fields and using these forces we change the particle coordinates and thus the particle distribution, after which the continuous quantities are re-calculated.

In Chapter 3 we test a one-dimensional version of the algorithm we described. We notice that the algorithm converges to a steady state, given the right boundary conditions, and we investigate the influence of several parameters on the precise final state. These experiments show good convergence upon increasing the numerical resolution. Finally we suggest a few adaptations to this algorithm for further research into the numerical simulation of ion beams.

Chapter 1

Theory

When we look at an ion beam, we will make the approximation that all particles (of the same type) are equivalent and independent. Since particles in an ion beam typically undergo a lot of small angle deflections the correlations between particles is small and this approximation is indeed valid. This allows us to describe the (perceived continuous) density of the ion beam in terms of the single particle probability density function, and hence to describe the evolution of the beam by calculating the evolution of the single particle distributions.

In order to describe the evolution of the single particle distribution, we need not only take the external forces into account, but also the forces of the other particles on the particle of interest. This can be achieved by using the Vlasov-Poisson-Fokker-Planck (VPFP) equation derived in [2]. We will review the derivation of this equation below, in a slightly more general form than we will use, allowing for multiple species of particles. Even though we will only describe an algorithm for a single particle type, the existence of the equations for multiple particle types gives us the option of expanding the algorithm in the future. Such an extended algorithm would then not only be able to describe an ion beam, but also electron cooling, for example.

From the single particle type Fokker-Planck equation we proceed to a system of stochastic differential equations (SDE's) as in [3]. It is this system of SDE's upon which we will base the algorithm that we develop in the next chapter.

1.1 Basic equations

Following [2], we define a phase-space domain $\mathcal{D}$ in which particles of several species move and we assume their motions to be uncorrelated. The particles of species a have mass m_a and charge q_a. Further we introduce a probability density function f_a, normalized such that

$$N_a = \int_{\mathcal{D}} f_a(\boldsymbol{r}, \boldsymbol{v}) \, \mathrm{d}^3\boldsymbol{r} \, \mathrm{d}^3\boldsymbol{v},$$

where N_a is the number of particles of species a in the domain, $\boldsymbol{r}$ is the spatial vector and $\boldsymbol{v}$ is the velocity vector, so that f_a represents the number of particles of type a per unit phase-space volume.

The Boltzmann equation, which describes the statistical behaviour of a thermodynamic system not in thermodynamic equilibrium, states that the change of the particle distribution functions is given by

$$\frac{\partial f_a}{\partial t} + \boldsymbol{v} \cdot \frac{\partial f_a}{\partial \boldsymbol{r}} + \frac{\boldsymbol{F}}{m_a} \cdot \frac{\partial f_a}{\partial \boldsymbol{v}} = \left(\frac{\partial f_a}{\partial t} \right)_{\text{coll}}, \tag{1.1}$$

where the force $\boldsymbol{F} = \boldsymbol{F}_{\text{ext}} + \boldsymbol{F}_{\text{mf}}$ includes both the external force $\boldsymbol{F}_{\text{ext}}$ and the self generated mean field space charge force $\boldsymbol{F}_{\text{mf}} = -\boldsymbol{\nabla}\phi$ which can be obtained from the Poisson equation

$$\boldsymbol{\nabla}^2\phi(\boldsymbol{r}) = -\frac{\rho(r)}{\epsilon_0}, \quad \rho(\boldsymbol{r}) = \sum_a q_a \int f_a(\boldsymbol{r}, \boldsymbol{v}) \, \mathrm{d}^3\boldsymbol{v}.$$

The quantity $(\partial f_a/\partial t)_{\text{coll}}$ represents the change in the distribution function due to collisions and needs to be specified further.

The appropriate Fokker-Planck equation for the rate of change of f_a due to collisions (in Cartesian coordinates) is [4]

$$\left(\frac{\partial f_a}{\partial t}\right)_{\text{coll}} = -\frac{\partial}{\partial \boldsymbol{v}} \cdot (f_a \langle \boldsymbol{\Delta v} \rangle_a) + \frac{1}{2}\frac{\partial^2}{\partial \boldsymbol{v}\partial \boldsymbol{v}} : (f_a \langle \boldsymbol{\Delta v}\boldsymbol{\Delta v}^T \rangle_a), \tag{1.2}$$

where $\langle \boldsymbol{\Delta v} \rangle_a$ is the average increment per unit time of the velocity of a particle of type a. This equation is actually an approximation in which terms of higher order in $\boldsymbol{\Delta v}$ are neglected, but we will justify this approximation in Section 1.3.

1.2 Collision kinematics

We will now take a look at the kinematics of an elastic collision between a particle of type a with velocity $\boldsymbol{v_a}$ before the collision and $\tilde{\boldsymbol{v}}_{\boldsymbol{a}}$ after the collision and a particle of type b with velocity $\boldsymbol{v_b}$ before the collision and $\tilde{\boldsymbol{v}}_{\boldsymbol{b}}$ after the collision. Due to conservation of momentum we can express the velocity $\boldsymbol{V}$ of the center of mass in two ways

$$\boldsymbol{V} = \frac{m_a \boldsymbol{v_a} + m_b \boldsymbol{v_b}}{m_a + m_b} = \frac{m_a \tilde{\boldsymbol{v}}_{\boldsymbol{a}} + m_b \tilde{\boldsymbol{v}}_{\boldsymbol{b}}}{m_a + m_b}.$$

We now introduce the relative velocities $\boldsymbol{u} = \boldsymbol{v_a} - \boldsymbol{v_b}$ and $\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{v}}_{\boldsymbol{a}} - \tilde{\boldsymbol{v}}_{\boldsymbol{b}}$ and notice that we may write

$$\boldsymbol{v_a} = \boldsymbol{V} + \frac{m_b}{m_a + m_b}\boldsymbol{u} \quad \text{and} \quad \tilde{\boldsymbol{v}}_{\boldsymbol{a}} = \boldsymbol{V} + \frac{m_b}{m_a + m_b}\tilde{\boldsymbol{u}},$$

which means that the change in particle a's velocity $\boldsymbol{\Delta v} = \tilde{\boldsymbol{v}}_{\boldsymbol{a}} - \boldsymbol{v_a}$ is related to the change in relative velocity $\boldsymbol{\Delta u} = \tilde{\boldsymbol{u}} - \boldsymbol{u}$ by

$$\boldsymbol{\Delta v} = \frac{m_b}{m_a + m_b}\boldsymbol{\Delta u}. \tag{1.3}$$

We will make use of this fact in order to calculate $\langle \boldsymbol{\Delta v} \rangle_a$ and $\langle \boldsymbol{\Delta v}\boldsymbol{\Delta v}^T \rangle_a$.

If we now look at energy conservation in the frame moving with velocity $\boldsymbol{V}$ we see that

$$\begin{aligned} m_a\|\boldsymbol{v_a} - \boldsymbol{V}\|^2 + m_b\|\boldsymbol{v_b} - \boldsymbol{V}\|^2 &= m_a\|\tilde{\boldsymbol{v}}_{\boldsymbol{a}} - \boldsymbol{V}\|^2 + m_b\|\tilde{\boldsymbol{v}}_{\boldsymbol{b}} - \boldsymbol{V}\|^2 \\ m_a\left\|\frac{m_b}{m_a+m_b}\boldsymbol{u}\right\|^2 + m_b\left\|\frac{m_a}{m_a+m_b}\boldsymbol{u}\right\|^2 &= m_a\left\|\frac{m_b}{m_a+m_b}\tilde{\boldsymbol{u}}\right\|^2 + m_b\left\|\frac{m_a}{m_a+m_b}\tilde{\boldsymbol{u}}\right\|^2 \\ \|\boldsymbol{u}\|^2 &= \|\tilde{\boldsymbol{u}}\|^2, \end{aligned}$$

Which means that the relative velocity may only be rotated, not rescaled. In order to simplify the upcomming equations we will now introduce a local Cartesian coordinate system with basis vectors $\boldsymbol{e_1'}$, $\boldsymbol{e_2'}$, $\boldsymbol{e_3'}$ which is related to the fixed system $\boldsymbol{e_1}$, $\boldsymbol{e_2}$, $\boldsymbol{e_3}$ by

$$\boldsymbol{e_1'} = \frac{\boldsymbol{u}}{u}, \quad \boldsymbol{e_2'} = \frac{\boldsymbol{e_3} \times \boldsymbol{u}}{\|\boldsymbol{e_3} \times \boldsymbol{u}\|} = \frac{\boldsymbol{e_3} \times \boldsymbol{u}}{(u_1^2 + u_2^2)^{1/2}}, \quad \boldsymbol{e_3'} = \boldsymbol{e_1'} \times \boldsymbol{e_2'}. \tag{1.4}$$

In the local system we see that there must exist angles $\theta \in [0, 2\pi)$ and $\phi \in [0, \pi)$ such that

$$\boldsymbol{\Delta u} = \tilde{\boldsymbol{u}} - \boldsymbol{u} = \begin{pmatrix} u\cos(\theta) \\ u\sin(\theta)\cos(\phi) \\ u\sin(\theta)\sin(\phi) \end{pmatrix}_L - \begin{pmatrix} u \\ 0 \\ 0 \end{pmatrix}_L = \begin{pmatrix} -2u\sin^2(\theta/2) \\ u\sin(\theta)\cos(\phi) \\ u\sin(\theta)\sin(\phi) \end{pmatrix}_L.$$

This expression can be used to calculate $\langle \boldsymbol{\Delta v} \rangle_a$ and $\langle \boldsymbol{\Delta v} \boldsymbol{\Delta v} \rangle_a$, since they are by definition[1] equal to

$$\langle \boldsymbol{\Delta v} \rangle_a = \sum_b \int f_b(\boldsymbol{r}, \boldsymbol{v_b}) \int u \, \boldsymbol{\Delta v} \, \sigma_{ab}(u, \Omega) \, \mathrm{d}\Omega \, \mathrm{d}\boldsymbol{v_b}$$
$$\langle \boldsymbol{\Delta v} \boldsymbol{\Delta v}^T \rangle_a = \sum_b \int f_b(\boldsymbol{r}, \boldsymbol{v_b}) \int u \, \boldsymbol{\Delta v} \boldsymbol{\Delta v}^T \, \sigma_{ab}(u, \Omega) \, \mathrm{d}\Omega \, \mathrm{d}\boldsymbol{v_b},$$

where σ denotes the differential scattering cross section for the Coulomb force

$$\sigma_{ab}(u, \Omega) = \left(\frac{q_a q_b}{8\pi\epsilon_0 m_{ab} u^2} \right)^2 \frac{1}{\sin^4(\theta/2)},$$

in which $m_{ab} = m_a m_b / (m_a + m_b)$ is the reduced mass of the particles. We will perform the calculation by first calculating

$$\{\boldsymbol{\Delta u}\}_{ab} = \int u \, \boldsymbol{\Delta u} \, \sigma_{ab}(u, \Omega) \, \mathrm{d}\Omega$$
$$\{\boldsymbol{\Delta u} \boldsymbol{\Delta u}^T\}_{ab} = \int u \, \boldsymbol{\Delta u} \boldsymbol{\Delta u}^T \, \sigma_{ab}(u, \Omega) \, \mathrm{d}\Omega,$$

in the local coordinate system. We can then perform the coordinate change and use Equation (1.3), after which only the velocity integral remains.

1.3 Calculation of the Fokker-Planck coefficients

We now see that in the local system

$$\{\boldsymbol{\Delta u}\}_{ab} = \left(\frac{q_a q_b}{8\pi\epsilon_0 m_{ab} u} \right)^2 \int_0^\pi \int_0^{2\pi} \frac{1}{\sin^4(\theta/2)} \begin{pmatrix} -2\sin^2(\theta/2) \\ \sin(\theta)\cos(\phi) \\ \sin(\theta)\sin(\phi) \end{pmatrix}_L \sin(\theta) \, \mathrm{d}\phi \, \mathrm{d}\theta,$$

so $\{\Delta u_{2L}\}_{ab} = \{\Delta u_{3L}\}_{ab} = 0$ and

$$\{\Delta u_{1L}\}_{ab} = -\pi \left(\frac{q_a q_b}{4\pi\epsilon_0 m_{ab} u} \right)^2 \int_0^\pi \frac{\sin^2(\theta/2)}{\sin^4(\theta/2)} \sin(\theta) \, \mathrm{d}\theta. \tag{1.5}$$

The integral in (1.5) diverges logarithmically at small angles. This divergence is caused by the long-range nature of the Coulomb force, since small angles correspond to large impact parameters, but it can be eliminated when we take shielding[2] into account. Once we introduce a minimal angle θ_{min} we get

$$\{\Delta u_{1L}\}_{ab} = 4\pi \left(\frac{q_a q_b}{4\pi\epsilon_0 m_{ab} u} \right)^2 \log(\sin(\theta_{\text{min}}/2)) \approx -4\pi \left(\frac{q_a q_b}{4\pi\epsilon_0 m_{ab} u} \right)^2 \ln(2/\theta_{\text{min}}).$$

The natural cutoff on the maximal impact parameter provided by shielding is of the order of a Debye length

$$\lambda_D = \left(\frac{\epsilon_0 k_B}{\sum_a q_a^2 n_a / T_a} \right)^{1/2},$$

where k_B is Boltzmann's constant and T_a is the temperature of species a. This impact parameter corresponds to an angle such that $\ln(2/\theta_{\text{min}}) = \ln(\Lambda)$. The quantity $\ln \Lambda$ is called the Coulomb logarithm and

[1] We will not go into the details of this definition, but some intuition for what it means can be gained from the following. For any scattering process $\int \sigma \mathrm{d}\Omega$ is an area, that corresponds to the area of a disk centered at the target particle such that if a particle with relative velocity u enters this disk it will be scattered. The differential scattering cross section itself also contains information about which part of the disk will cause which scattering angle. The quantity $f_b u$ determines how many particles of type b the particle of type a will encounter, so that $\int f_b \, u \, \sigma_{ab} \, \mathrm{d}\Omega$ will give the scattering rate.

[2] The long-range effects of the Coulomb force are suppressed by the flow of the other particles in response to the electric field of the particle of interest. This reduces the effective interaction at long distances to the mean field interaction, which we had already taken into account. Hence we get a maximal impact parameter, and thus a minimal scattering angle, for the collisions.

its insensitivity to the precise value of u (it scales as $\ln(u)$) allows us to use the common approximation $\ln(\Lambda) \approx 10$. Note that since $\ln(\Lambda) \gg 1$ and since any term higher than second order in $\boldsymbol{\Delta v}$ will not contain $\ln(\Lambda)$, the neglect of those terms in the Fokker-Planck Equation (1.2) was indeed justified.

Now for the second order terms we find that

$$\{\Delta u_{iL}\Delta u_{jL}\}_{ab} = 0 \quad \text{for } i \neq j,$$

$$\{(\Delta u_{1L})^2\}_{ab} = 4u^3\left(\frac{q_a q_b}{8\pi\epsilon_0 m_{ab}u^2}\right)^2 \int_0^{\pi}\int_0^{2\pi} \frac{\sin^4(\theta/2)}{\sin^4(\theta/2)}\sin(\theta)\,\mathrm{d}\phi\,\mathrm{d}\theta = 4\pi u\left(\frac{q_a q_b}{4\pi\epsilon_0 m_{ab}u}\right)^2,$$

$$\begin{aligned}\{(\Delta u_{2L})^2\}_{ab} &= u^3\left(\frac{q_a q_b}{8\pi\epsilon_0 m_{ab}u^2}\right)^2 \int_0^{\pi}\int_0^{2\pi} \frac{\sin^2(\theta)\cos^2(\phi)}{\sin^4(\theta/2)}\sin(\theta)\,\mathrm{d}\phi\,\mathrm{d}\theta \\ &= 8\pi u\left(\frac{q_a q_b}{8\pi\epsilon_0 m_{ab}u}\right)^2 \int_0^{\pi} \frac{\sin^3(\theta/2)\cos^3(\theta/2))}{\sin^4(\theta/2)}\,\mathrm{d}\theta \\ &= \pi u\left(\frac{q_a q_b}{4\pi\epsilon_0 m_{ab}u}\right)^2 \left[\cos(\theta) + 4\log(\sin(\theta/2))\right]_0^{\pi}.\end{aligned}$$

We make use of the minimal scattering angle once more to find

$$\{(\Delta u_{2L})^2\}_{ab} = -4\pi u\left(\frac{q_a q_b}{4\pi\epsilon_0 m_{ab}u}\right)^2 \log(\sin(\theta_{\min}/2)) \approx 4\pi u\left(\frac{q_a q_b}{4\pi\epsilon_0 m_{ab}u}\right)^2 \ln(\Lambda).$$

And similarly we find

$$\{(\Delta u_{3L})^2\}_{ab} \approx 4\pi u\left(\frac{q_a q_b}{4\pi\epsilon_0 m_{ab}u}\right)^2 \ln(\Lambda).$$

Since these last two contain a factor $\ln(\Lambda)$, the size of $\{(\Delta u_{1L})^2\}_{ab}$ is an order of magnitude smaller, motivating the approximation of Rosenbluth et al.

$$\{(\Delta u_{1L})^2\}_{ab} = 0.$$

If we now combine these with equations (1.3) and (1.4) we find that

$$\begin{aligned}\{\Delta v_i\}_{ab} &= \frac{m_b}{m_a + m_b}\sum_j \boldsymbol{e_i}\cdot\boldsymbol{e'_j}\{\Delta u_{jL}\}_{ab} \\ &= -\frac{m_{ab}}{m_a}\boldsymbol{e_i}\cdot\frac{\boldsymbol{u}}{u}4\pi\left(\frac{q_a q_b}{4\pi\epsilon_0 m_{ab}u}\right)^2\ln(\Lambda) \\ &= -\frac{(q_a q_b)^2}{4\pi\epsilon_0^2 m_a m_{ab}u^3}\ln(\Lambda)u_i,\end{aligned}$$

and that

$$\{\Delta v_i \Delta v_j\}_{ab} = \frac{q_a^2 q_b^2}{4\pi\epsilon_0^2 m^2}\ln(\Lambda)\frac{1}{u}\left(\delta_{ij} - \frac{u_i u_j}{u^2}\right).$$

Obtaining the latter expression requires a long calculation, which can be found in Appendix B.1.

If we now introduce the shorthand

$$\Gamma_a \equiv \frac{q_a^4}{4\pi\epsilon_0^2 m^2}\ln(\Lambda),$$

and we recall that $u = (\sum_i (v_{ai} - v_{bi})^2)^{1/2}$, we see that we may write

$$\{\boldsymbol{\Delta v}\}_{ab} = \Gamma_a\frac{m_a}{m_{ab}}\left(\frac{q_b}{q_a}\right)^2\frac{\partial}{\partial \boldsymbol{v_a}}\frac{1}{u}, \quad \{\boldsymbol{\Delta v}\,\boldsymbol{\Delta v}^T\} = \Gamma_a\left(\frac{q_b}{q_a}\right)^2\frac{\partial^2}{\partial \boldsymbol{v_a}\partial \boldsymbol{v_a}}u.$$

1.4 Resulting equations

Now it is clear that

$$\langle \boldsymbol{\Delta v} \rangle_a = \sum_b \int f_b(\boldsymbol{r}, \boldsymbol{v_b}) \{\boldsymbol{\Delta v}\}_{ab} \, d^3\boldsymbol{v_b} = \Gamma_a \frac{\partial h_a}{\partial \boldsymbol{v_a}}, \tag{1.6}$$

and

$$\langle \boldsymbol{\Delta v} \, \boldsymbol{\Delta v}^T \rangle_a = \Gamma_a \frac{\partial^2 g_a}{\partial \boldsymbol{v_a} \partial \boldsymbol{v_a}}, \tag{1.7}$$

where

$$h_a(\boldsymbol{r}, \boldsymbol{v}) = \sum_b \frac{m_a + m_b}{m_b} \left(\frac{q_b}{q_a}\right)^2 \int \frac{f_b(\boldsymbol{r}, \tilde{\boldsymbol{v}})}{\|\boldsymbol{v} - \tilde{\boldsymbol{v}}\|} \, d^3\tilde{\boldsymbol{v}},$$

$$g_a(\boldsymbol{r}, \boldsymbol{v}) = \sum_b \left(\frac{q_b}{q_a}\right)^2 \int f_b(\boldsymbol{r}, \tilde{\boldsymbol{v}}) \, \|\boldsymbol{v} - \tilde{\boldsymbol{v}}\| \, d^3\tilde{\boldsymbol{v}}.$$

Lastly we define

$$\boldsymbol{F_{da}} \equiv \Gamma_a \frac{\partial h_a}{\partial \boldsymbol{v}},$$

$$\boldsymbol{D_a} \equiv \Gamma_a \frac{\partial^2 g_a}{\partial \boldsymbol{v} \partial \boldsymbol{v}},$$

which are known as the dynamic friction coefficient and the diffusion coefficient respectively. Now we can combine equations (1.1) and (1.2) with equations (1.6) and (1.7) to get the complete Fokker-Planck equation for the evolution of our system

$$\frac{\partial f_a}{\partial t} + \boldsymbol{v} \cdot \frac{\partial f_a}{\partial \boldsymbol{r}} + \frac{\boldsymbol{F}}{m_a} \cdot \frac{\partial f_a}{\partial \boldsymbol{v}} = -\frac{\partial}{\partial \boldsymbol{v}} \cdot (f_a \boldsymbol{F_{da}}) + \frac{1}{2} \frac{\partial^2}{\partial \boldsymbol{v} \partial \boldsymbol{v}} : (f_a \boldsymbol{D_a}).$$

In this thesis we will only treat the case where there is a single species of particles, with distribution f, mass m and charge q. In this case the equations simplify to the VPFP equation we will use,

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \frac{\partial f}{\partial \boldsymbol{r}} + \frac{\boldsymbol{F}}{m} \cdot \frac{\partial f}{\partial \boldsymbol{v}} = -\frac{\partial}{\partial \boldsymbol{v}} \cdot (f \boldsymbol{F_d}) + \frac{1}{2} \frac{\partial^2}{\partial \boldsymbol{v} \partial \boldsymbol{v}} : (f \boldsymbol{D}), \tag{1.8}$$

where

$$\boldsymbol{F}(\boldsymbol{r}, \boldsymbol{v}) = \boldsymbol{F_{ext}} + \boldsymbol{F_{mf}} = \boldsymbol{F_{ext}} - q \boldsymbol{\nabla} \phi, \tag{1.9}$$

$$\boldsymbol{F_d}(\boldsymbol{r}, \boldsymbol{v}) = \Gamma \frac{\partial h}{\partial \boldsymbol{v}}, \tag{1.10}$$

$$\boldsymbol{D}(\boldsymbol{r}, \boldsymbol{v}) = \Gamma \frac{\partial^2 g}{\partial \boldsymbol{v} \partial \boldsymbol{v}}, \tag{1.11}$$

in which

$$\Gamma = \frac{q^4}{4\pi\epsilon_0^2 m^2} \ln(\Lambda), \quad \ln(\Lambda) \approx 10, \tag{1.12}$$

$$h(\boldsymbol{r}, \boldsymbol{v}) = 2 \int \frac{f(\boldsymbol{r}, \tilde{\boldsymbol{v}})}{\|\boldsymbol{v} - \tilde{\boldsymbol{v}}\|} \, \boldsymbol{d}^3\tilde{\boldsymbol{v}}, \tag{1.13}$$

$$g(\boldsymbol{r}, \boldsymbol{v}) = \int f(\boldsymbol{r}, \tilde{\boldsymbol{v}}) \|\boldsymbol{v} - \tilde{\boldsymbol{v}}\| \, \boldsymbol{d}^3\tilde{\boldsymbol{v}}, \tag{1.14}$$

$$\boldsymbol{\nabla}^2 \phi(\boldsymbol{r}) = -\frac{\rho(\boldsymbol{r})}{\epsilon_0}, \tag{1.15}$$

$$\rho(\boldsymbol{r}) = q \int f(\boldsymbol{r}, \boldsymbol{v}) \, \boldsymbol{d}^3\boldsymbol{v}. \tag{1.16}$$

Lastly we note that the integrals h and g do in fact converge, even though that may not seem obvious at first glance. The convergence of g becomes clear if we keep in mind that in any situation of physical relevance f is a bounded function which will either have a bounded support in $\tilde{\boldsymbol{v}}$ or it will decay exponentially at large values of $\tilde{\boldsymbol{v}}$. The convergence of h (it looks like there might be a problem around $\boldsymbol{v} = \tilde{\boldsymbol{v}}$) becomes more clear if we first perform a change of variables to $\boldsymbol{w} = \boldsymbol{v} - \tilde{\boldsymbol{v}}$ and then change to polar coordinates:

$$\begin{aligned} h(\boldsymbol{r}, \boldsymbol{v}) &= 2 \int \frac{f(\boldsymbol{r}, \tilde{\boldsymbol{v}})}{\|\boldsymbol{v} - \tilde{\boldsymbol{v}}\|} \, \boldsymbol{d}^3 \tilde{\boldsymbol{v}} \\ &= 2 \int \frac{f(\boldsymbol{r}, \boldsymbol{v} - \boldsymbol{w})}{\|\boldsymbol{w}\|} \, \boldsymbol{d}^3 \boldsymbol{w} \\ &= 2 \int_0^\infty \int_0^\pi \int_0^{2\pi} \frac{f(\boldsymbol{r}, \boldsymbol{v} - \boldsymbol{w})}{w} w^2 \sin(\phi) \, \mathrm{d}\theta \, \mathrm{d}\phi \, \mathrm{d}w \end{aligned}$$

If we take into account the properties for physically relevant functions f, as we did for g, we do indeed see that h will converge as well.

1.5 Elliptic description

It is interesting to note that h and g can also be described by elliptic equations. Calculating the derivatives of equations (1.13) and (1.14) will show that

$$\boldsymbol{\nabla}_{\boldsymbol{v}}^2 h_a = -4\pi \sum_b \frac{m_a + m_b}{m_b} \left(\frac{q_b}{q_a} \right)^2 f_b(\boldsymbol{r}, \boldsymbol{v}),$$

$$\boldsymbol{\nabla}_{\boldsymbol{v}}^2 \boldsymbol{\nabla}_{\boldsymbol{v}}^2 g_a = -8\pi \sum_b \left(\frac{q_b}{q_a} \right)^2 f_b(\boldsymbol{r}, \boldsymbol{v}).$$

These equations give us an alternative way to calculate h and g, namely through a Poisson equation. These equations are also the reason that h and g are often referred to as the Rosenbluth potentials.

In the next chapter we shall formulate two versions of our algorithm, one using this elliptic description and the other using the integro-differential description of the previous section. Note that in the special case where there is only one type of charged particle the equations simplify somewhat, and the expressions are related by

$$\boldsymbol{\nabla}_{\boldsymbol{v}}^2 h = \boldsymbol{\nabla}_{\boldsymbol{v}}^2 \boldsymbol{\nabla}_{\boldsymbol{v}}^2 g,$$

so that, with the right boundary conditions, g satisfies the Poisson equation

$$\boldsymbol{\nabla}_{\boldsymbol{v}}^2 g = h,$$

which reduces the computational cost of this approach.

1.6 Recasting the problem

Using Itô's Lemma it can be shown that the VPFP equation (1.8) describes the evolution of the probability density function of the following system of stochastic (multiplicative noise) differential equations (SDEs)

$$\begin{cases} \mathrm{d}\boldsymbol{r} = \boldsymbol{v} \, \mathrm{d}t \, , \\ \mathrm{d}\boldsymbol{v} = \left(\dfrac{\boldsymbol{F}}{m} + \boldsymbol{F}_{\mathrm{d}} \right) \mathrm{d}t + \boldsymbol{Q} \cdot \mathrm{d}\boldsymbol{W}_{\boldsymbol{t}} \, , \end{cases} \tag{1.17}$$

where $\mathrm{d}\boldsymbol{W}(t)$ are Gaussian random variables with

$$\langle \mathrm{d}W_i(t) \rangle = 0 \, ,$$

$$\langle \mathrm{d}W_i(t)\, \mathrm{d}W_j(t') \rangle = \delta_{ij}\, \delta(t-t')\ ,$$

and the matrix $\boldsymbol{Q}$ is such that $\boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{D}$. The system (1.17) will often be a family of systems, indexed by a set of matrices $\boldsymbol{Q}$ with the desired property. Theorem 1.6.2 proves that such a matrix $\boldsymbol{Q}$ always exists and that it does not matter which one we choose if we look at weak convergence only. For more information about SDEs we refer the reader to Appendix A.

By equivalence we mean here that when we would take an infinite collection of particles with positions and velocities initialized according to the distribution $\frac{1}{N} f_0(\boldsymbol{r}, \boldsymbol{v})$ and let those particles evolve according to (1.17) then the distribution will evolve according to (1.8). It is important to note that, even though the statistical properties of the particle ensemble would correspond to the statistical properties of the physical system, the particles themselves are not guaranteed to behave in the same way as the physical particles.

The fact that we are only interested in the statistical behaviour, and not in the paths themselves, implies for our numerical experiments that we are only interested in weak convergence, and not strong convergence. This in turn means that we are indeed free to choose any matrix $\boldsymbol{Q}$ such that $\boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{D}$.

Lemma 1.6.1 *The matrix $\boldsymbol{D}$ defined in Equation* (1.11) *is positive semi-definite.*

Proof First note that $\frac{ne^4}{4\pi\epsilon_0^2 m^2}\lambda$ is a non-negative scalar, that f is a non-negative scalar function and that

$$\frac{\partial^2}{\partial \boldsymbol{v}\partial \boldsymbol{v}} \|\boldsymbol{v} - \tilde{\boldsymbol{v}}\| = \frac{1}{\|\boldsymbol{v} - \tilde{\boldsymbol{v}}\|} \left(\boldsymbol{I} - \frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T \boldsymbol{v}} \right)\ ,$$

where $\boldsymbol{I}$ denotes the 3×3 identity matrix. Being a positive scalar multiple of a projection matrix, this matrix is positive semi-definite. Now we see that

$$\begin{aligned}
\min_{\{\boldsymbol{x}\in\mathbb{R}^3 \mid \|\boldsymbol{x}\|=1\}} \boldsymbol{x}^T \boldsymbol{D} \boldsymbol{x} &= 2\frac{ne^4}{4\pi\epsilon_0^2 m^2}\lambda \min_{\{\boldsymbol{x}\in\mathbb{R}^3 \mid \|\boldsymbol{x}\|=1\}} \boldsymbol{x}^T \int f(\boldsymbol{r}, \tilde{\boldsymbol{v}}) \frac{\partial^2}{\partial \boldsymbol{v}\partial \boldsymbol{v}} \|\boldsymbol{v} - \tilde{\boldsymbol{v}}\|\, \boldsymbol{d}^3\tilde{\boldsymbol{v}}\, \boldsymbol{x} \\
&= 2\frac{ne^4}{4\pi\epsilon_0^2 m^2}\lambda \min_{\{\boldsymbol{x}\in\mathbb{R}^3 \mid \|\boldsymbol{x}\|=1\}} \int f(\boldsymbol{r}, \tilde{\boldsymbol{v}})\, \boldsymbol{x}^T \frac{\partial^2}{\partial \boldsymbol{v}\partial \boldsymbol{v}} \|\boldsymbol{v} - \tilde{\boldsymbol{v}}\|\, \boldsymbol{x}\, \boldsymbol{d}^3\tilde{\boldsymbol{v}} \\
&\geq 2\frac{ne^4}{4\pi\epsilon_0^2 m^2}\lambda \int f(\boldsymbol{r}, \tilde{\boldsymbol{v}}) \min_{\{\boldsymbol{x}\in\mathbb{R}^3 \mid \|\boldsymbol{x}\|=1\}} \boldsymbol{x}^T \frac{\partial^2}{\partial \boldsymbol{v}\partial \boldsymbol{v}} \|\boldsymbol{v} - \tilde{\boldsymbol{v}}\|\, \boldsymbol{x}\, \boldsymbol{d}^3\tilde{\boldsymbol{v}} \\
&\geq 0.
\end{aligned}$$

So we see that $\boldsymbol{D}$ is indeed positive semi-definite.

⊠

Theorem 1.6.2 *There always exists a matrix $\boldsymbol{Q}$ such that $\boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{D}$ and if there exists more than one such matrix, any choice will yield the same solution of* (1.17)*, where 'same' should be interpreted in the weak convergence sense.*

Proof Since $\boldsymbol{D}$ is a real symmetric matrix and thus there exists a decomposition $\boldsymbol{D} = \boldsymbol{U}^T \boldsymbol{\Sigma} \boldsymbol{U}$ with $\boldsymbol{U}$ unitary and $\boldsymbol{\Sigma}$ diagonal. The matrix Σ will contain the eigenvalues of $\boldsymbol{D}$, which are non-negative by Lemma 1.6.1, so we can define $\boldsymbol{\Sigma}^{\frac{1}{2}}$ by taking the square root of the entries. It is easy to see that the matrix $\boldsymbol{A} = \boldsymbol{U}^T \boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{U}$ does indeed satisfy $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{D}$, proving the existence of such a matrix.

Now assume that there are $\boldsymbol{Q_1},\ \boldsymbol{Q_2}$ such that $\boldsymbol{Q_1}\boldsymbol{Q_1}^T = \boldsymbol{Q_2}\boldsymbol{Q_2}^T = \boldsymbol{D}$. Then there must exist a unitary matrix $\boldsymbol{U}$ such that $\boldsymbol{Q_1}\boldsymbol{U} = \boldsymbol{Q_2}$. So we see that $\boldsymbol{Q_2} \cdot \boldsymbol{dW}(t) = \boldsymbol{Q_1}\boldsymbol{U} \cdot \boldsymbol{dW}(t)$ and since the coordinates of $\boldsymbol{dW}(t)$ are independent and identically distributed (iid) Gaussian random variables with mean zero, $\boldsymbol{U} \cdot \boldsymbol{dW}(t)$ has the same probability density function as $\boldsymbol{dW}(t)$ (as proven in Lemma B.2.1), which implies that $\boldsymbol{Q_2} \cdot \boldsymbol{dW}(t)$ has the same probability density function as $\boldsymbol{Q_1} \cdot \boldsymbol{dW}(t)$. So both $\boldsymbol{Q_1}$ and $\boldsymbol{Q_2}$ yield the same solution of (1.17) with regard to weak convergence, proving the theorem.

⊠

1.7 Nondimensionalization

We will use a technique called nondimensionalization in order to get a better idea of the scales involved in this problem. More about this technique can be found in [5]. We will introduce dimensional constants that represent the characteristic scales of the problem, denoted with a subscript c, and nondimensional dynamical quantities which are denoted with a subscript asterisk. We introduce the following quantities

$$
\begin{aligned}
&f = f_c f_* \\
&t = t_c t_* \\
&\boldsymbol{v} = v_c \boldsymbol{v}_* \\
&\boldsymbol{r} = r_c \boldsymbol{r}_* \\
&\boldsymbol{W_t} = t_c^{1/2} \boldsymbol{W_{t*}} \\
&\boldsymbol{E_{ext}} = E_c \boldsymbol{E_{ext*}} \\
&\boldsymbol{B_{ext}} = B_c \boldsymbol{B_{ext*}}
\end{aligned}
$$

Which allows us to write the system (1.17) as

$$
\begin{cases}
\mathrm{d}\boldsymbol{r_*} = & \dfrac{v_c t_c}{r_c} \boldsymbol{v_*} \, \mathrm{d}t_* \\
\mathrm{d}\boldsymbol{v_*} = & \left(\dfrac{q E_c t_c}{m v_c} \boldsymbol{E_{ext*}} + \dfrac{q B_c t_c}{m} \boldsymbol{v_*} \times \boldsymbol{B_{ext*}} + \dfrac{q^2 r_c v_c^2 f_c t_c}{\epsilon_0 m} \boldsymbol{F_{mf*}} + \dfrac{\ln(\Lambda)}{2\pi} \dfrac{q^4 f_c t_c}{\epsilon_0^2 m^2} \boldsymbol{F_{d*}} \right) \mathrm{d}t_* \\
& + \sqrt{\dfrac{\ln(\Lambda)}{4\pi}} \dfrac{q^2 f_c^{1/2} t_c^{1/2}}{\epsilon_0 m} Q_* \mathrm{d}\boldsymbol{W_{t*}}
\end{cases}
\tag{1.18}
$$

where

$$
\begin{aligned}
&\boldsymbol{F_{mf*}} = \nabla_* \phi_* \\
&\nabla_*^2 \phi_* = \rho_* = \int f_* \, \mathrm{d}^3 \boldsymbol{v_*} \\
&\boldsymbol{F_{d*}} = \frac{\partial h_*}{\partial \boldsymbol{v_*}} \\
&h_* = \int \frac{f_*}{\|\boldsymbol{v_*} - \tilde{\boldsymbol{v_*}}\|} \, \mathrm{d}^3 \tilde{\boldsymbol{v_*}} \\
&Q_* Q_*^T = D_* = \frac{\partial^2 g}{\partial \boldsymbol{v_*}} \\
&g_* = \int f_* \, \|\boldsymbol{v_*} - \tilde{\boldsymbol{v_*}}\| \, \mathrm{d}\boldsymbol{v_*}.
\end{aligned}
$$

These equations allow us to make calculations using quantities that are scaled such that their representation in a computer will be more accurate. In the numerical experiments reported in Chapter 3, we make no use of this, but we can still use the nondimensionalized equations to our advantage. By choosing the characteristic values, except for the characteristic timescale, to match the test case for our numerical experiments we will see that the natural timescales for the different processes in our equation vary a lot. We will take a closer look at this in Section 3.1.

Chapter 2

The algorithm

In order to solve the system of SDEs (1.17) we use a so-called particle-in-cell (PIC) method, in which we simulate a collection of particles that indirectly represent the phase-space distribution, but we solve the equations for the coefficients we need on a grid (the particles are not confined to gridpoints). More about PIC methods can be found in [6, 7] and its application in plasma physics is treated in [8, 9, 10, 11, 12, 13], while some alternative methods are discussed in [14, 15, 16, 17]. An outline of this procedure is given in Algorithm 1.

Initialize: Set the positions and velocities of the particles according to an appropriate initial distribution.

`for` $t = 1 : T$ `do`
    Calculate $f(\boldsymbol{r}, \boldsymbol{v})$ on the grid, using the coordinates of the particles
    Calculate $\boldsymbol{F}$, $\boldsymbol{F_d}$ and $\boldsymbol{D}$ on the grid
    Calculate $\boldsymbol{F}$, $\boldsymbol{F_d}$ and $\boldsymbol{Q}$ at particle positions, using the values on the grid
    Update the positions and velocities of the particles

Algorithm 1: A brief outline of the main program

So to summarize our approach of solving the original problem: We have gone from a particle description to a continuous one by assuming the particles are independent and identical and then considering probabilities, resulting in Equation (1.8). We then went back to a single particle equation, with a Wiener process defined to approximate the collision statistics (Equation (1.17)). This allows us to now go back to a particle description through discretization by particle-in-cell method, combined with employing Monte-Carlo sampling of the diffusion process. This chapter will give a more detailed description of that final step.

Even though nature is three dimensional, and the derivation of the equations in the previous chapter has made explicit use of this, we will formulate our algorithm in d dimensions. We should note that this means that the d-dimensional model is not a model of a d-dimensional physical world, but rather a d-dimensional mathematical extension of a physical 3-dimensional model. We do this so that we can run our initial experiments in only one dimension, in order to have low computation times.

2.1 Nomenclature

In order to give a clear description of the algorithm we will first introduce some of the notation used in this thesis.

We take our $2d$-dimensional phase-space domain to be

$$\mathcal{D} = [0, L_{r_1}] \times \cdots \times [0, L_{r_d}] \times [-L_{v_1}, L_{v_1}] \times \cdots \times [-L_{v_d}, L_{v_d}],$$

with periodic boundary conditions, for ease of computation. Periodicity in the velocity directions would lead to nonphysical behaviour if a change between $-L_{v_i}$ and L_{v_i} occurs. If we however take care that the particle density tends to zero at the edges of the domain, then the particles will not experience such transitions. In our simulations we will take care that this condition is indeed fulfilled, and since it is a reasonable condition for physical problems it will not limit the generality of the algorithm by much.

On the domain we use a Cartesian phase-space grid, on which there are $\prod_{i=1}^{d} N_{r_i} \cdot N_{v_i}$ gridpoints. The grid spacing is

$$\Delta r_i = \frac{L_{r_i}}{N_{r_i}}, \quad \Delta v_i = \frac{L_{v_i}}{N_{v_i}}, \quad \text{for } i \in \{1, \dots, d\},$$

and the gridpoints are at coordinates

$$(n_1 \Delta r_1, \dots, n_3 \Delta r_3, m_1 \Delta v_1 - L_{v_1}, \dots, m_3 \Delta v_3 - L_{v_3}),$$

for $n_i \in \{0, 1, \dots, N_{r_i} - 1\}$ and $m_i \in \{0, 1, \dots, N_{v_i} - 1\}$ for each $i \in \{1, \dots, d\}$. In cases where all of the gridpoint-indices appear together we will use vector notation to avoid highly space-consuming notation, so

$$\zeta_{\boldsymbol{n},\boldsymbol{m}} \equiv \zeta_{n_1,\dots,n_3,m_1,\dots,m_3}.$$

We will use a similar notation in sums, so for example

$$\sum_{\boldsymbol{n},\boldsymbol{m}} \equiv \sum_{n_1=0}^{N_{r_1}-1} \cdots \sum_{n_d=0}^{N_{r_d}-1} \sum_{m_1=0}^{N_{v_1}-1} \cdots \sum_{m_d=0}^{N_{v_d}-1}.$$

In cases where we only need a spatial grid we use the natural choice of gridpoints

$$(n_1 \Delta r_1, n_2 \Delta r_2, n_3 \Delta r_3) \quad \text{for } n_i \in \{0, 1, \dots, N_{r_i} - 1\},$$

and we make a similar choice in the case where only a velocity grid is needed.

Lastly, we denote the phase-space coordinates of our N_{sim} simulated particles by

$$\left(r_1^{(k)}, r_2^{(k)}, r_3^{(k)}, v_1^{(k)}, v_2^{(k)}, v_3^{(k)}\right) \in \mathcal{D} \quad \text{for all } k \in \{1, 2, \dots, N_{\text{sim}}\}.$$

2.2 Translating between quantities on the grid and at particle positions

As can be seen in Algorithm 1 we need to convert information about the particles to the grid and back again several times. We will do so using linear splines which, given the grid-spacing Δx are defined as

$$B(x) = \begin{cases} 1 - \left|\frac{x}{\Delta x}\right| & \text{if } |x| \leq \Delta x \\ 0 & \text{otherwise} \end{cases}$$

in one dimension and as a product of these in multiple dimensions.

This function has the convenient property that it can be used to form a partition of unity of our (periodic) domain, namely the set

$$\left\{ \prod_{i=1}^{d} B(r_i - n_i \Delta r_i) B(v_i - m_i \Delta v_i) \,\middle|\, n_i \in \{0, \dots, N_{r_i} - 1\},\, m_i \in \{0, \dots, N_{v_i} - 1\} \right\}.$$

This allows us to approximate any function $\zeta(\boldsymbol{r}, \boldsymbol{v})$ as

$$\zeta(\boldsymbol{r}, \boldsymbol{v}) = \sum_{\boldsymbol{n},\boldsymbol{m}} \left(\zeta_{\boldsymbol{n},\boldsymbol{m}} \prod_{i=1}^{d} B(r_i - n_i \Delta r_i) B(v_i - m_i \Delta v_i) \right),$$

with $\{\zeta_{\boldsymbol{n},\boldsymbol{m}}\}$ some appropriate set of coefficients. Note that in this case, with linear splines, this is equivalent to linear interpolation between the values $\{\zeta_{\boldsymbol{n},\boldsymbol{m}}\}$.

If we have such a function defined on our grid we can evaluate it at the particle positions by summing the spline values at the particle positions with the right coefficients from the grid. Note that due to the fact that $B(x)$ has a small support we only need to sum over the 2^{2d} gridpoints surrounding our particle, instead of all $\prod_{i=1}^{d} N_{r_i} N_{v_i}$. This results in Algorithm 2 for the evaluation of functions on the grid at particle positions. The algorithm is given for $2d$-dimensional phase-space functions, but it extends naturally to one on a d-dimensional (spatial or velocity) grid. In both cases the execution time of this algorithm is $\mathcal{O}(4^d d N_{\text{sim}})$.

Function: `GridToPart_2dD`
Input: The particle positions $\{(\boldsymbol{r}^{(k)}, \boldsymbol{v}^{(k)})\}$ and grid coefficients $\{\zeta_{\boldsymbol{n},\boldsymbol{m}}\}$.
Initialize: Set the function values at particle positions $\{\zeta(\boldsymbol{r}^{(k)}, \boldsymbol{v}^{(k)})\}$ to zero.

`for` $k = 0 : N_{\text{sim}} - 1$ `do`
  Find the set G of 2^{2d} gridpoints surrounding $(\boldsymbol{r}^{(k)}, \boldsymbol{v}^{(k)})$.
  `for` $(\boldsymbol{m}, \boldsymbol{n}) \in G$ `do`
    $\zeta(\boldsymbol{r}^{(k)}, \boldsymbol{v}^{(k)}) += \zeta_{\boldsymbol{n},\boldsymbol{m}} \prod_{i=1}^{d} B(r_i^{(k)} - n_i \Delta r_i) B(v_i^{(k)} - m_i \Delta v_i)$

Algorithm 2: Algorithm for evaluating a function defined by coefficients of linear splines on the $2d$-dimensional grid at particle positions.

Going the other way around we can use the splines as well, provided there is some quantity $\zeta_{\text{part}}^{(i)}$ ($i \in 0, \ldots, N_{\text{sim}} - 1$) that we know each of the particles to contribute to $\zeta(\boldsymbol{r}, \boldsymbol{v})$. Examples for which such a quantity can be found are the number density (for which we can use the number of physical particles represented, divided by the volume of a grid cell) and the charge density (for which we can use the amount of charge represented, divided by the volume of a grid cell). The calculation of the coefficients on the grid is then as follows

$$\zeta_{\boldsymbol{n},\boldsymbol{m}} = \sum_{k=0}^{N_{\text{sim}}-1} \zeta_{\text{part}}^{(k)} \prod_{i=1}^{d} B(r_i^{(k)} - n_i \Delta r_i) B(v_i^{(k)} - m_i \Delta v_i).$$

We have one function, $\rho(\boldsymbol{r})$, which is defined on a d-dimensional grid instead of the $2d$-dimensional one. However, an exact integration of the splines will yield the same coefficients as a direct calculation of the coefficients on the d dimensional grid, at a lower computational cost. The verification of this equivalence can be found in Appendix B.3.

We can once again make use of the fact that the splines have a small support to construct an efficient algorithm, like Algorithm 3, which again extends naturally to the d-dimensional case. The reason that we still look at the surrounding gridpoints per particle, just as in Algorithm 2, instead of finding the surrounding particles for each gridpoint, is that the former can be implemented much more efficiently. The execution time is again $\mathcal{O}(4^d d N_{\text{sim}})$.

This way of translating between grid and particles is similar to the method used in [18] to solve the rotating shallow water equations, which was proven lead to a convergent PIC algorithm (under appropriate assumptions, which are slightly more strict than the conditions we are using in this thesis) in [19].

2.3 Calculations on the grid

The quantities $\boldsymbol{F}_{\mathbf{d}}$, $\boldsymbol{D}$ and $\boldsymbol{F}_{\mathbf{mf}}$, which are defined by equations (1.9) through (1.16), will be calculated directly on the grid before evaluating them at the particle positions. We will describe two different methods for these calculations.

Function: `PartToGrid_2dD`
Input: The particle positions $\{(\boldsymbol{r}^{(k)}, \boldsymbol{v}^{(k)})\}$ and particle contributions $\{\zeta_{\text{part}}^{(k)}\}$.
Initialize: Set the grid coefficients $\{\zeta_{\boldsymbol{n},\boldsymbol{m}}\}$ to zero.

`for` $k = 0 : N_{\text{sim}} - 1$ `do`
  Find the set G of 2^{2d} gridpoints surrounding $(\boldsymbol{r}^{(k)}, \boldsymbol{v}^{(k)})$.
  `for` $(\boldsymbol{m}, \boldsymbol{n}) \in G$ `do`
    $\zeta_{\boldsymbol{n},\boldsymbol{m}} += \zeta_{\text{part}}^{(k)} \prod_{i=1}^{d} B(r_i^{(k)} - n_i \Delta r_i) B(v_i^{(k)} - m_i \Delta v_i)$

Algorithm 3: Algorithm for calculating the coefficients of linear splines on the $2d$-dimensional grid from the particle positions.

2.3.1 Using Fourier transformations

The first method relies on the definition of $\boldsymbol{F}_{\mathrm{d}}$ and $\boldsymbol{D}$ as a derivative of a convolution integral, which can be written as a convolution integral itself

$$\boldsymbol{F_d}(\boldsymbol{r}, \boldsymbol{v}) = 2\Gamma \int f(\boldsymbol{r}, \boldsymbol{v}) \frac{\partial \|\boldsymbol{v} - \tilde{\boldsymbol{v}}\|^{-1}}{\partial \boldsymbol{v}} \, \mathrm{d}^3\tilde{\boldsymbol{v}},$$

$$\boldsymbol{D}(\boldsymbol{r}, \boldsymbol{v}) = \Gamma \int f(\boldsymbol{r}, \boldsymbol{v}) \frac{\partial \|\boldsymbol{v} - \tilde{\boldsymbol{v}}\|}{\partial \boldsymbol{v}} \, \mathrm{d}^3\tilde{\boldsymbol{v}}.$$

This formulation shows that we can use FFT[1]-based convolution in velocity-space to compute them, which means we will compute them as

$$\{(F_{\mathrm{d}\,i})_{\boldsymbol{n},\boldsymbol{m}}\} = 2\Gamma \mathcal{F}^{-1}\left(\mathcal{F}(\{f_{\boldsymbol{n},\boldsymbol{m}}\}) \cdot \mathcal{F}\left(\left\{\left(\frac{\partial}{\partial v_i}\frac{1}{\|\boldsymbol{v}\|}\right)_{\boldsymbol{m}}\right\}\right)\right) \quad i \in \{1,2,3\} \quad \forall \boldsymbol{n}$$

$$\{(D_{ij})_{\boldsymbol{n},\boldsymbol{m}}\} = \Gamma \mathcal{F}^{-1}\left(\mathcal{F}(\{f_{\boldsymbol{n},\boldsymbol{m}}\}) \cdot \mathcal{F}\left(\left\{\left(\frac{\partial^2}{\partial v_i \partial v_j}\|\boldsymbol{v}\|\right)_{\boldsymbol{m}}\right\}\right)\right) \quad i,j \in \{1,2,3\} \quad \forall \boldsymbol{n}$$

(Note that D_{ij} needs only be calculated for $i \geq j$, because of symmetry.) However, the functions $\partial/\partial v_i(1/\|\boldsymbol{v}\|)$ and $\partial^2/\partial v_i \partial v_j(\|\boldsymbol{v}\|)$ are ill-behaved around $\boldsymbol{v} = 0$. Therefore they may not be well represented by the splines, which is important for accuracy.

We will take a closer look at these functions. For any open neighborhood around $\boldsymbol{v} = 0$ the function

$$\frac{\partial}{\partial v_i}\frac{1}{\|\boldsymbol{v}\|} = -\frac{v_i}{\|\boldsymbol{v}\|^3.}$$

takes all values between positive and negative infinity, and the function

$$\frac{\partial^2}{\partial v_i \partial v_j}\|\boldsymbol{v}\| = \delta_{ij} - \frac{v_i v_j}{\|\boldsymbol{v}\|^2}$$

takes values in $[-\frac{1}{2}, \frac{1}{2}]$ for $i \neq j$ and values in $[0, 1]$ for $i = j$. Therefore both functions cannot be assigned a value at $\boldsymbol{v} = 0$ such that they are continuous, and especially the former function will be hard to describe using our splines. In the code we will define the function values at $\boldsymbol{v} = 0$ to be equal to their average over a sphere centered at that point, which is zero for the former functions and 0 or $\frac{2}{3}$ for the latter, for $i \neq j$ and $i = j$ respectively.

For the calculation of the (inverse) Fourier transforms, we suggest using the FFTW library. We will not elaborate on the theory behind those calculations here, but more information can be found in [20] and via the FFTW homepage. Using this library we can carry out a Fourier transform on a grid with N points in $\mathcal{O}(N \log(N))$ flops, regardless of the number of dimensions.

[1] Fast Fourier Transform.

The calculation of $\boldsymbol{F}_{\mathbf{mf}} = -q\boldsymbol{\nabla}\phi$ is somewhat simpler. As can be seen from the definitions, we need to solve a Poisson equation for ϕ. This can efficiently be done using a Fourier transform in position-space

$$\{\phi_{\boldsymbol{n}}\} = -\frac{1}{\epsilon_0}\mathcal{F}^{-1}\left(\left\{\frac{1}{n_1^2+n_2^2+n_3^2}\right\}\cdot\mathcal{F}(\{\rho_{\boldsymbol{n}}\})\right).$$

After that $\boldsymbol{F}_{\mathbf{mf}}$ can be calculated using central finite differences on the grid.

2.3.2 Using the numerical elliptic description

The second method relies on the elliptic description of h and g given in Section 1.5. This description uses Poisson equations instead of elliptic integrals. Hence we can calculate discretized versions of h and g by solving the discretized Poisson equations. This amounts to constructing a discrete Laplace operator, e.g. using finite differences, (using appropriate boundary conditions) and then solving the resulting matrix-vector equations for h and g. After that $\boldsymbol{F_d}$ and $\boldsymbol{D}$ can simply be calculated by numerical differentiation. Using this formalism it would be more consistent to also calculate $\boldsymbol{F}_{\mathbf{mf}}$ using a discrete Laplacian, instead of the Fourier method used in the previous subsection.

The boundary conditions for the spatial calculations are obvious: since we have a periodic domain, the boundary conditions should be periodic. Finding good boundary conditions for the Laplacian in velocity space is however much more complicated. In our numerical experiment we shall test two different boundary conditions, and, as we shall see, the choice of a boundary condition has a significant impact on the results. We will leave the question of what the ideal boundary conditions for this problem would be unanswered, with the remark that it may be an interesting topic for further research.

The discretized Laplace operator in velocity space is a sparse $\prod_{i=1}^{d} N_{v_i}$ by $\prod_{i=1}^{d} N_{v_i}$ matrix with $2d+1$ nonzero entries in each row/column, so solving a discretized Poisson equation in velocity space will take $\mathcal{O}(d\prod_{i=1}^{d} N_{v_i})$ flops. Similarly solving a spatial Poisson equation will take $\mathcal{O}(d\prod_{i=1}^{d} N_{r_i})$ flops.

2.4 The Cholesky decomposition

Once we know the diffusion coefficient $\boldsymbol{D}$ at the particle positions we still need to find a matrix $\boldsymbol{Q}$ such that $\boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{D}$. As we have seen in Section 1.6 we can make any choice for such a matrix. Here we will choose $\boldsymbol{Q}$ such that it is lower triangular. This is called the Cholesky decomposition of $\boldsymbol{D}$ and it can be calculated efficiently. There are several algorithms available, which can be found in [21] amongst others. We have chosen a version that exploits the positive semi-definiteness of $\boldsymbol{D}$, which will take $\mathcal{O}(d)$ flops per matrix.

2.5 SPDE solution scheme

The simplest approach to solving (1.17) would be to use the Euler-Maruyama method. However, we will use a slightly more sophisticated method. We use a splitting method similar to those in [22], which consists of three types of steps, of which two are repeated to form the following ABCBA structure

A) Update $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{v}\frac{\Delta t}{2}$.

B) Calculate $\boldsymbol{F}(\boldsymbol{r})$, then update $\boldsymbol{v} \leftarrow \boldsymbol{v} + \frac{\boldsymbol{F}}{m}\frac{\Delta t}{2}$.

C) Calculate $\boldsymbol{F}_{\mathbf{d}}(\boldsymbol{r},\boldsymbol{v})$ and $\boldsymbol{Q}(\boldsymbol{r},\boldsymbol{v})$, then update $\boldsymbol{v} \leftarrow \boldsymbol{v} + \boldsymbol{F}_{\mathbf{d}}\,\Delta t + \boldsymbol{Q}\cdot\boldsymbol{\Delta W}_t$.

B) Calculate $\boldsymbol{F}(\boldsymbol{r})$, then update $\boldsymbol{v} \leftarrow \boldsymbol{v} + \frac{\boldsymbol{F}}{m}\frac{\Delta t}{2}$.

A) Update $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{v}\frac{\Delta t}{2}$.

Where $\boldsymbol{\Delta W}_t$ is a vector of iid Gaussians with mean zero and variance Δt.

Apart from the fact that it will allow us to adapt to the varying timescales later on, this scheme has the advantage that without the C step it would be a symmetric and symplectic scheme. This means that without the C step the scheme would be guaranteed to preserve the Hamiltonian of the system and that

it would be time-reversible, which makes it more true to nature and guarantees that the accuracy is at least $\mathcal{O}(\Delta t^2)$. The C step itself is a Euler-Maruyama type step and makes the resulting scheme $\mathcal{O}(\Delta t)$ nevertheless. Substituting step C for a symmetric variant would increase the order of accuracy of the scheme, but it would require a step that is implicit. This would require more computation time and is not feasible with the computing power available for this thesis.

As mentioned in Section 1.7 the time scales for the different parts will differ a lot in our test case, we will see in Section 3.1 that steps A and B involve a much quicker timescale than step C. We could decrease the numerical time step such that it is small enough to accurately simulate the processes in steps A and B. However, since step C is quite expensive, this would not be very efficient. Instead we choose to resolve the C steps with a bigger time step than the A and B steps, by increasing the number of A and B steps per C step. Since the update in step A concerns an update of positions based only on velocity, and vice versa for step B, it would not help to make multiple steps of type A or B consecutively. Thus we come to a scheme of the form $(AB)^{N_{\text{rep}}}C(BA)^{N_{\text{rep}}}$, where N_{rep} is an integer greater than one, which determines how much smaller the A and B time steps are compared to the C time step. More information on such multiple time-stepping methods can be found in [23].

2.6 The complete algorithm

This chapter has shown us the details to be implemented in each of the steps of Algorithm 1, so that we can now extend it to a full pseudocode, which can be found as Algorithm 4. In this algorithm we incorporate both options for the calculations on the grid, with the Fourier formalism displayed in cyan and the discrete elliptic formalism in green.

We see that the type A steps take $\mathcal{O}(N_{\text{sim}})$ flops. Type B steps take $\mathcal{O}(4^d dN_{\text{sim}} + d\prod_{i=1}^{d} N_{r_i})$ using the elliptical description or $\mathcal{O}(4^d dN_{\text{sim}} + (d + \log(\prod_{i=1}^{d} N_{r_i}))\prod_{i=1}^{d} N_{r_i})$ flops using Fourier transforms. The type C steps take $\mathcal{O}(4^d d^3 N_{\text{sim}} + d^2 \prod_{i=1}^{d} N_{v_i})$ using the elliptical description or $\mathcal{O}(4^d d^3 N_{\text{sim}} + d^2 \log(\prod_{i=1}^{d} N_{v_i})\prod_{i=1}^{d} N_{v_i})$ flops using Fourier transforms. Overall that means that the algorithm takes $\mathcal{O}(4^d dTN_{\text{rep}}N_{\text{sim}} + 4^d d^3 TN_{\text{sim}} + dTN_{\text{rep}}\prod_{i=1}^{d} N_{r_i} + d^2 T\prod_{i=1}^{d} N_{v_i})$ flops using the elliptical description or $\mathcal{O}(4^d dTN_{\text{rep}}N_{\text{sim}} + 4^d d^3 TN_{\text{sim}} + (d + \log(\prod_{i=1}^{d} N_{r_i}))TN_{\text{rep}}\prod_{i=1}^{d} N_{r_i} + d^2 \log(\prod_{i=1}^{d} N_{v_i})T\prod_{i=1}^{d} N_{v_i})$ using Fourier transforms.

Initialize: Set the positions and velocities of the particles according to some initial distribution and calculate $\mathcal{F}\left(\left\{\left(\frac{\partial}{\partial v_i}\frac{1}{\|\boldsymbol{v}\|}\right)_{\boldsymbol{m}}\right\}\right)$ for $i \in \{1,2,3\}$ and $\mathcal{F}\left(\left\{\left(\frac{\partial}{\partial v_i \partial v_j}\frac{1}{\|\boldsymbol{v}\|}\right)_{\boldsymbol{m}}\right\}\right)$ for $i,j \in \{1,2,3\}$ or initialize the discretized Laplacians $\boldsymbol{\nabla}^2_{\boldsymbol{r}}$ and $\boldsymbol{\nabla}^2_{\boldsymbol{v}}$.

for $t = 1 : T$ do
  for $r = 1 : N_{\text{rep}}$ do
    $\{\boldsymbol{r}^{(k)}\} \leftarrow \{\boldsymbol{r}^{(k)} + \boldsymbol{v}^{(k)}\frac{\Delta t}{2}\}$

    PartToGrid_dD(ρ)
    Calculate ϕ on the grid as:
      $\{\phi_{\boldsymbol{n}}\} = -\frac{1}{\epsilon_0}\mathcal{F}^{-1}\left(\left\{\frac{1}{n_1^2+n_2^2+n_3^2}\right\}\cdot\mathcal{F}(\{\rho_{\boldsymbol{n}}\})\right)$ or
      solve $\boldsymbol{\nabla}^2_{\boldsymbol{r}}\phi(\boldsymbol{r}) = -\frac{\rho(\boldsymbol{r})}{\epsilon_0}$
    Calculate $\{\boldsymbol{F}_{\mathbf{n}}\}$ using central finite differences on $\{\phi_{\mathbf{n}}\}$
    for $i = 1 : 3$ do
      GridToPart_dD(F_i)

    $\{\boldsymbol{v}^{(k)}\} \leftarrow \{\boldsymbol{v}^{(k)} + \frac{\boldsymbol{F}^{(k)}}{m}\frac{\Delta t}{2}\}$

  PartToGrid_2dD(f)
  Solve $\boldsymbol{\nabla}^2_{\boldsymbol{v}} h = -8\pi\, f(\boldsymbol{r},\boldsymbol{v})$ for h
  Solve $\boldsymbol{\nabla}^2_{\boldsymbol{v}} g = h$ for g

  for $i = 1 : d$ do
    Calculate $F_{\mathrm{d}\,i}$ on the grid as:
      $\{(F_{\mathrm{d}\,i})_{\boldsymbol{n},\boldsymbol{m}}\} = \Gamma\mathcal{F}^{-1}\left(\mathcal{F}(\{f_{\boldsymbol{n},\boldsymbol{m}}\})\cdot\mathcal{F}\left(\left\{\left(\frac{\partial}{\partial v_i}\frac{1}{\|\boldsymbol{v}\|}\right)_{\boldsymbol{m}}\right\}\right)\right)$ or
      $F_{\mathrm{d}\,i}(\boldsymbol{r},\boldsymbol{v}) = \Gamma\frac{\partial h}{\partial v_i}$ (use finite differences)
    GridToPart_2dD($F_{\mathrm{d}\,i}$)
    for $j = 1 : d$ do
      Calculate D_{ij} as:
        $\{(D_{ij})_{\boldsymbol{n},\boldsymbol{m}}\} = \Gamma\mathcal{F}^{-1}\left(\mathcal{F}(\{f_{\boldsymbol{n},\boldsymbol{m}}\})\cdot\mathcal{F}\left(\left\{\left(\frac{\partial^2}{\partial v_i \partial v_j}\|\boldsymbol{v}\|\right)_{\boldsymbol{m}}\right\}\right)\right)$
        $F_{\mathrm{d}\,i}(\boldsymbol{r},\boldsymbol{v}) = \Gamma\frac{\partial g}{\partial v_i \partial v_j}$ (use finite differences)
      GridToPart_2dD(D_{ij})
      $\{Q_{ij}(\boldsymbol{r}^{(k)},\boldsymbol{v}^{(k)})\}$=Cholesky($\{D_{ij}(\boldsymbol{r}^{(k)},\boldsymbol{v}^{(k)})\}$)

  $\{\boldsymbol{v}^{(k)} \leftarrow \boldsymbol{v}^{(k)} + \boldsymbol{F}^{(k)}_{\mathbf{d}}\Delta t + \boldsymbol{Q}^{(k)}\cdot\boldsymbol{\Delta W}_{\mathrm{t}}\}$

  for $r = 1 : N_{\text{rep}}$ do
    PartToGrid_dD(ρ)
    Calculate ϕ on the grid as:
      $\{\phi_{\boldsymbol{n}}\} = -\frac{1}{\epsilon_0}\mathcal{F}^{-1}\left(\left\{\frac{1}{n_1^2+n_2^2+n_3^2}\right\}\cdot\mathcal{F}(\{\rho_{\boldsymbol{n}}\})\right)$ or
      solve $\boldsymbol{\nabla}^2_{\boldsymbol{r}}\phi(\boldsymbol{r}) = -\frac{\rho(\boldsymbol{r})}{\epsilon_0}$
    Calculate $\{\boldsymbol{F}_{\mathbf{n}}\}$ using central finite differences on $\{\phi_{\mathbf{n}}\}$
    for $i = 1 : 3$ do
      GridToPart_dD(F_i)

    $\{\boldsymbol{v}^{(k)}\} \leftarrow \{\boldsymbol{v}^{(k)} + \frac{\boldsymbol{F}^{(k)}}{m}\frac{\Delta t}{2}\}$

    $\{\boldsymbol{r}^{(k)}\} \leftarrow \{\boldsymbol{r}^{(k)} + \boldsymbol{v}^{(k)}\frac{\Delta t}{2}\}$

Algorithm 4: A more detailed pseudocode for solving our system of SDEs.

Chapter 3

Results

In this chapter we will examine the performance of our algorithm for a simple test case. Since the hardware[1] for this project is quite limited compared to the supercomputers that are usually employed for these kinds of simulations we will simplify the problem and the algorithm slightly, and examine whether the performance of the simple algorithm is good enough to warrant investigation of the full algorithm with more powerful hardware.

In these experiments we will use a restriction of the equations to one dimension, without external force, so the equation we will solve is

$$\begin{cases} \mathrm{d}r = v\,\mathrm{d}t \\ \mathrm{d}v = \left(\frac{F}{m} + F_d\right)dt + Q\,\mathrm{d}W_t \end{cases}$$

where dW_t are Gaussian random variables with $\langle \mathrm{d}W_t \rangle = 0$ and $\langle \mathrm{d}W_t\,\mathrm{d}W_{t'} \rangle = \delta(t - t')$ and

$$\begin{aligned} F(r,v) &= F_{mf} = -q\frac{\partial}{\partial r}\phi, \\ F_d(r,v) &= \Gamma\frac{\partial h}{\partial v}, \\ Q(r,v) &= \sqrt{\Gamma\frac{\partial^2 g}{\partial v^2}}, \end{aligned}$$

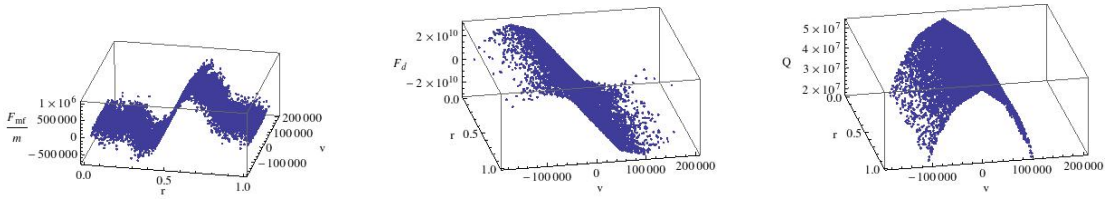
in which

$$\begin{aligned} \Gamma &= \frac{q^4}{4\pi\epsilon_0^2 m^2}\ln(\Lambda), \quad \ln(\Lambda) \approx 10, \\ \frac{\partial^2}{\partial v^2}h(r,v) &= -8\pi f(r,v), \\ \frac{\partial^4}{\partial v^4}g(r,v) &= -8\pi f(r,v), \\ \frac{\partial^2}{\partial r^2}\phi(r) &= -\frac{\rho(\boldsymbol{r})}{\epsilon_0}, \\ \rho(r) &= q\int f(r,v)\,\mathrm{d}v. \end{aligned}$$

Note that we are using the elliptic description of h and g, since a one dimensional restriction of Equation (1.13) does not converge.

We use a Matlab script to perform the calculations for this test, in which we solve the Poisson equations by solving a sparse matrix vector equation using a discretized Laplacian. This allows for a very efficient algorithm whose cost, as we saw in the previous section, is linear in all parameters (except for the dimension d, which we have fixed to 1 in this chapter).

[1] An ASUS X7BS series laptop, with an Core i7-2630QM at 2.0 GHz and 6 GB RAM, using the Ubuntu operating system.



(a) F_{mf} (b) F_d (c) Q

Figure 3.1: The forces working on the particles, at the particle positions, in the initial time step of the experiment of Section 3.2, using Neumann boundary conditions.

3.1 Experimental setup

We will test the algorithm using realistic parameters, based on those of LEIR[2]. However since we are only using a one-dimensional reduction of the equations we do not expect the results to be precisely as in the physically relevant case. As it turns out, the dynamic friction and the diffusion coefficient are larger than in the three dimensional case, removing the need for our parameter N_{rep}.

The spatial domain will be one meter long and we choose a velocity domain with $L_v = 2.5 \cdot 10^5$ meter per second. The initial conditions are random, with a uniform distribution along the spatial direction and a Gaussian distribution of the velocity, with mean 0 and standard deviation $5 \cdot 10^4$ meter per second. The mass and charge of the particles are set to correspond to Pb^{54+} ions and we set the physical particle number to $1 \cdot 10^7$.

In the experiments below we expect the velocity distribution to remain Gaussian, since the solution of the VPFP equation is proven to converge to a Gaussian velocity distribution in [24].

3.2 Influence of the numerical boundary condition

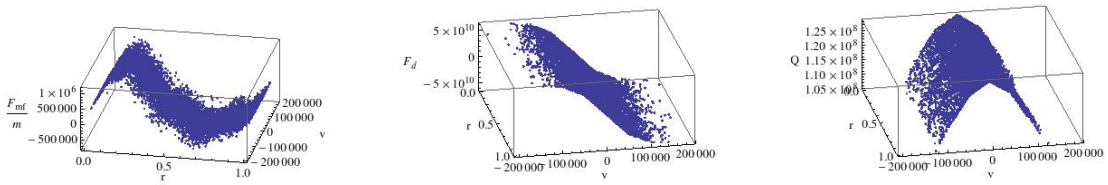
In this first experiment we use 10 gridpoints in both position and velocity space, and thus 100 points in phase space. We use 10^4 numerical particles and simulate for a time of 10^{-5} seconds using 1000 time steps. This simulation takes approximately 1.2 minute on our test system.

The only freedom left for our algorithm now is the choice of boundary conditions in solving the Laplace equations and carrying out finite difference differentiation. In the spatial direction we will choose periodic boundary condition, since this is the logical choice for a (circular) one-dimensional accelerator or storage ring. But the choice for the boundary conditions in the velocity direction is not as straightforward. We will test two different boundary conditions, the first being a Neumann boundary condition, which assumes the velocity derivative of functions to be zero on the edges of the velocity domain and the second is a Dirichlet boundary condition, which assumes the function value itself to be zero on the edges of the velocity domain.

Figures 3.1 and 3.2 show the forces that act on the particles in the initial time step for Neumann and Dirichlet conditions respectively, these are similar to those at later time steps and thus give a good idea of the forces throughout the simulation. The magnitudes of these forces tell us that the velocity updates due to the mean field are approximately $10^6 \cdot \frac{10^{-8}}{2} = 5 \cdot 10^{-3}$ meter per second for both cases, while those due to dynamic friction and diffusion are approximately $2 \cdot 10^{10} \cdot 10^{-8} = 2 \cdot 10^2$ and $5 \cdot 10^7 \cdot \sqrt{10^{-8}} = 5 \cdot 10^3$ meter per second in the Neumann case and $5 \cdot 10^{10} \cdot 10^{-8} = 5 \cdot 10^2$ and $1.25 \cdot 10^8 \cdot \sqrt{10^{-8}} = 1.24 \cdot 10^4$ meter per second in the Dirichlet case. If we compare these to each other and to the standard deviation of the velocity distribution ($5 \cdot 10^5$ meter per second), we notice that the effects of the mean field force are negligible, and that at this time step size the other two velocity updates are large enough to perform accurately and probably small enough to capture the dynamics of the system. Recall that the fact that the mean field force is negligible is by design, since we initialize with a uniform spatial density.

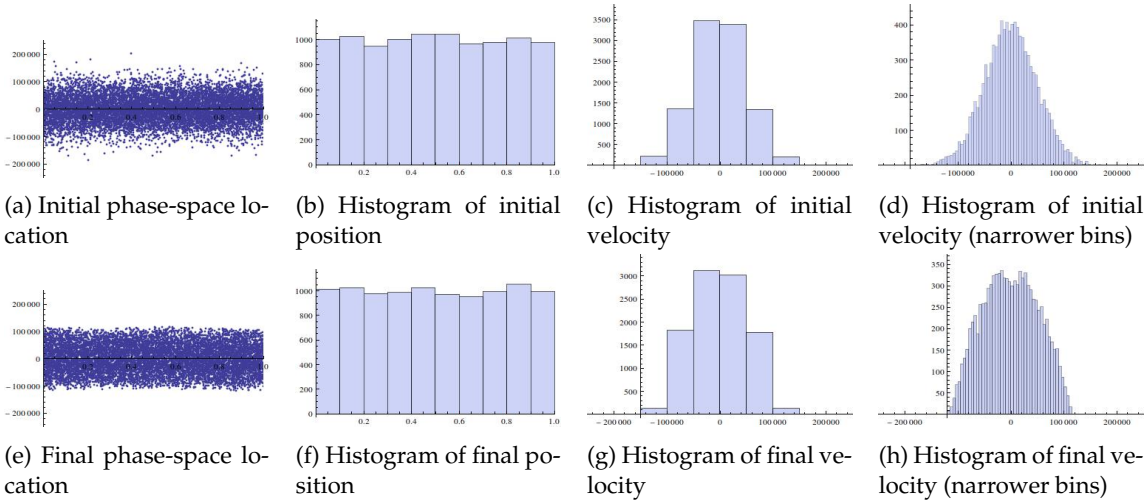
The results of these simulations are shown in figures 3.3 and 3.4, by means of plots of the phase-space locations of all particles and histograms of position and velocity separately, for the initial and final configurations. The final phase-space distribution in figure 3.3e seems to be a stationary state,

[2]The Low Energy Ion Ring (LEIR) is currently in use at CERN, as part of the Large Hadron Collider (LHC) acceleration chain.



(a) F_{mf} (b) F_d (c) Q

Figure 3.2: The forces working on the particles, at the particle positions, in the initial time step of the experiment of Section 3.2, using Dirichlet boundary conditions.



(a) Initial phase-space location (b) Histogram of initial position (c) Histogram of initial velocity (d) Histogram of initial velocity (narrower bins)

(e) Final phase-space location (f) Histogram of final position (g) Histogram of final velocity (h) Histogram of final velocity (narrower bins)

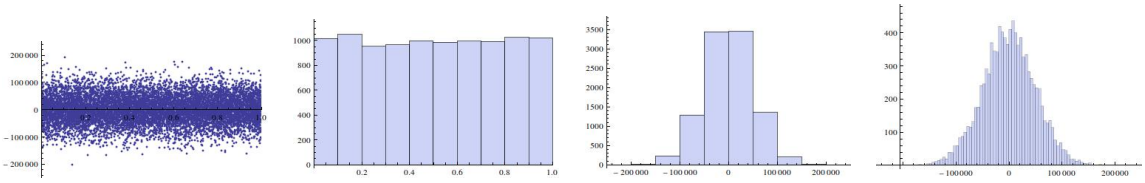
Figure 3.3: Comparision of initial (top row) and final (bottom row) phase space locations for the experiment of Section 3.2 with Neumann boundary conditions, using (from left to right) a plot of the phase space coordinates of all particles, a histogram of the spatial positions with bin width equal to the grid resolution, a similar histogram for velocity and a histogram of the velocities with bin with equal to a tenth of a grid cell.

for the algorithm with Neumann boundary conditions, whereas the algorithm with Dirichlet boundary conditions seems to have no stable state, and the velocity distribution continually widens.

Both algorithms seem to maintain the uniform spatial distribution, but the velocity distribution is not as expected in both cases. The reason we do not obtain a stable Gaussian velocity profile lies mainly with the diffusion coefficients, which are shown in figure 3.5. We see that for the case with Neumann boundary conditions, the diffusion coefficient is negative for large velocities. Since the dynamic friction is a mean-reverting force which increases with increasing deviation from the mean velocity, the absence of diffusion means that particles will not stay in the large speed areas of our phase-space domain. In the case of Dirichlet boundary conditions though, the diffusion coefficient does not go to zero fast enough, and the diffusion coefficient is quite large on the entire domain, which leads to too much diffusion of the velocities.

We have seen that the problem in these algorithms is too little diffusion in one case and too much diffusion in the other case. So even though neither choice of boundary conditoins gives the desired result, we believe that a correct boundary condition will lead to a version of this algorithm that does correctly maintain a Gaussian velocity profile.

In the rest of this chapter we will look at one experiment examining the distribution widening for the Dirichlet boundary conditions and then examine the convergence behaviour of the algorithm with Neumann boundary conditions in more detail. As we have seen, the choice of Neumann boundary conditions for the Rosenbluth potentials leads to an evolution with steady state. We use this test case to examine the convergence of our algorithm as a function of numerical discretization parameters.

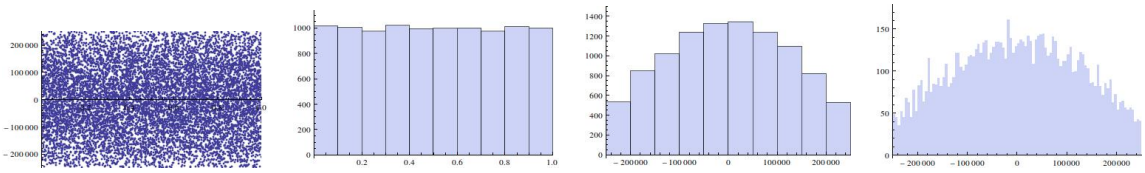


(a) Initial phase-space location

(b) Histogram of initial position

(c) Histogram of initial velocity

(d) Histogram of initial velocity (narrower bins)



(e) Final phase-space location

(f) Histogram of final position

(g) Histogram of final velocity

(h) Histogram of final velocity (narrower bins)

Figure 3.4: Comparision of initial (top row) and final (bottom row) phase space locations for the experiment of Section 3.2 with Dirichlet boundary conditions, using (from left to right) a plot of the phase space coordinates of all particles, a histogram of the spatial positions with bin width equal to the grid resolution, a similar histogram for velocity and a histogram of the velocities with bin with equal to a tenth of a grid cell.

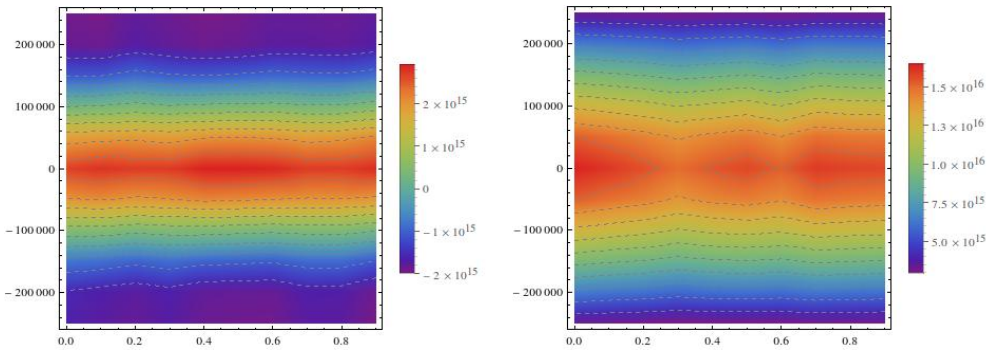


Figure 3.5: Diffusion coefficient on the grid for the experiment of Section 3.2, with Neumann boundary conditions (left) and Dirichlet boundary conditions (right). (Note the different scales.)

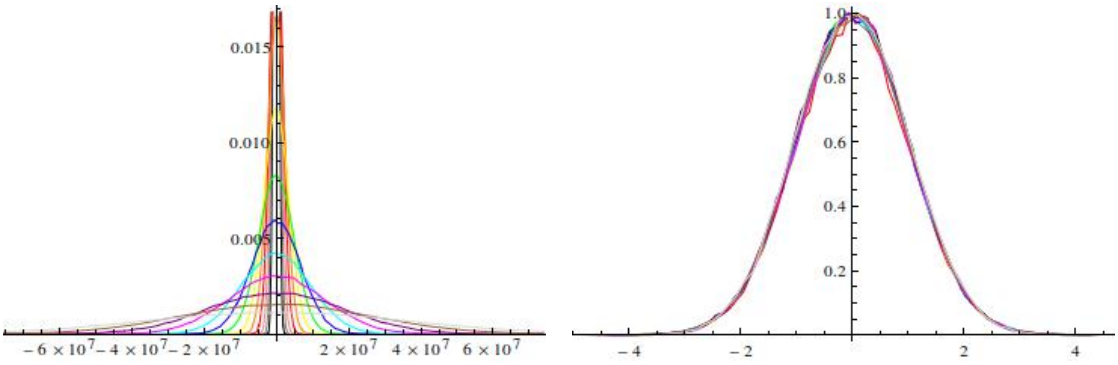


Figure 3.6: The velocity distributions at various times (left) for the experiment of Section 3.3 and normalized versions of those same distributions (right). The distributions are given at $t = 2^n \cdot 10^{-8}$ for $n = 1, 2, \ldots, 13$ and their colors ordered from low times to high times are black, grey, pink, red, orange, yellow, green, blue, cyan, magenta, purple, brown and light grey.

3.3 Distribution widening for Dirichlet boundary conditions

We will now investigate the widening of the distribution for the case of Dirichlet boundary conditions. For this experiment we use a domain that is larger in the spatial and temporal directions, while keeping the resolution the same. We choose a velocity domain with $L_v = 3 \cdot 10^8$ meters per second, still using gridcells of $5 \cdot 10^4$ meters per second wide and the temporal domain runs until $T = 2^{13} \cdot 10^{-8} \approx 8.2 \cdot 10^{-5}$ seconds, still using steps of 10^{-8} seconds. In order to make sure we can accurately draw the velocity distributions we use $N_{\text{sim}} = 10^5$. The results of this experiment can be seen in figure 3.6.

We see that the distribution expands very rapidly at first (note that the initial distribution with a standard deviation of $5 \cdot 10^4$ meters per second would hardly be noticeable in this figure), but but the spreading rate decreases continually as time progresses. It is also interesting to note that even though the distribution widens, it is self-similar. This implies that the Maxwellian distribution is structurally stable but does not appear to be attracting for any finite variance under this boundary condition.

3.4 Convergence with respect to numerical particles for fixed grid

One of the parameters that most significantly influences convergence is the number of numerical particles. In the following experiment we will use the same settings as in the previous experiment, except for the varying number of numerical particles. We then calculate the average and standard deviation of the position and velocity distributions for ten runs of the algorithm for each of the numerical particle numbers. The results of this experiment can be seen in figure 3.7.

First of all we see that, next to the accuracy in the calculation, the particle number also influences the accuracy of the initial condition. This is a concept one usually does not encounter, since initial conditions are fixed. In our case however we have a fixed initial distribution, but the initial particle coordinates are drawn randomly from this distribution, which means that the initial distribution of the particles may not represent the initial distribution well. It is clear that increasing the number of numerical particles greatly improves this.

We see that for small particle numbers we have large errors, especially the standard deviation of the velocity grows too large. For somewhat larger particle numbers the errors in the standard deviation are less pronounced and we mostly see a shift of the mean velocity. This second type of error is to be expected, since there is no mechanism to bring back the average velocity once an error has occurred[3]. However this effect also decreases with increased particle number.

Overall we conclude that with increasing the numerical particle number we see a clear convergence of the mean and average of the phase-space distribution.

[3]To be more precise, the equation we are solving is invariant under a shift of the mean, hence when a numerical error induces a shifted mean it will not be brought back to the original mean.

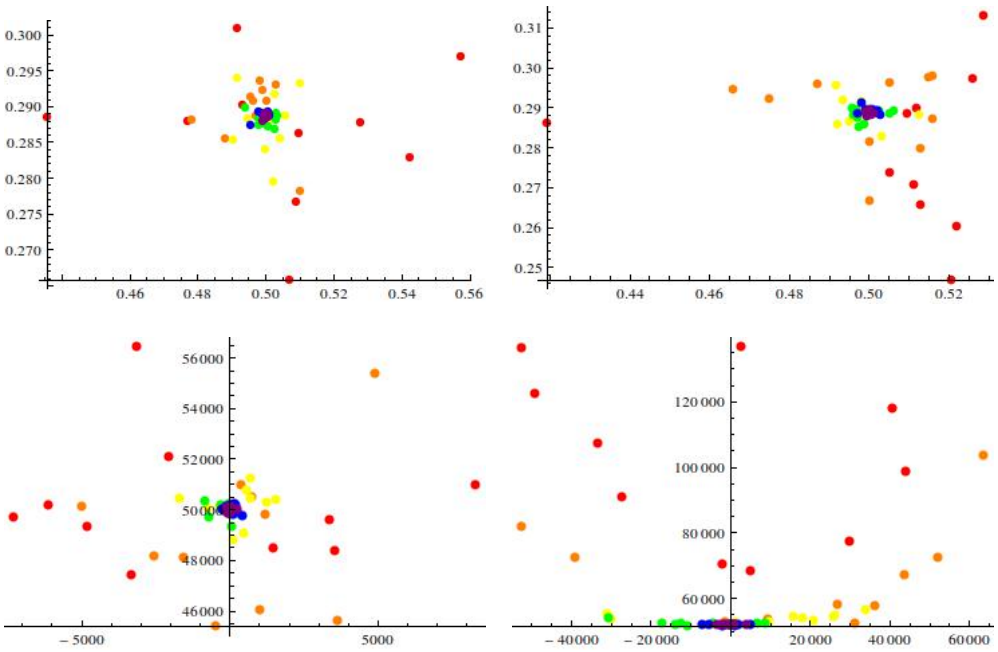


Figure 3.7: Initial (left) and final (right) values for the mean and standard deviation of the position (top) and velocity (bottom) for the experiment of Section 3.4, showing ten instances of the algorithm for each of the choices of numerical particle number: 80 (red), 400 (orange), 2000 (yellow), 10000 (green), 50000 (blue), 250000 (purple).

3.5 Convergence with respect to grid resolution for fixed number of particles

Next we will be looking at the influence of the velocity grid size on convergence. In this experiment we once again use the same settings as in the first experiment, but now with a varying number of gridpoints along the velocity direction. The results can be seen in figure 3.8.

As expected this setting does not influence the initial distributions and does not have a significant influence on the final spatial distribution. The impact on the final velocity distribution however is clear. Even though the drift of the mean does not seem to be influenced, there is a clear convergence of the standard deviation.

The reason the convergence effect is not as strong as one may hope is that while this parameter does decrease one type of error, it increases another. The increase of the number of velocity grid points allows for a more accurate representation of all velocity dependent quantities. However by increasing the number of gridpoints while keeping the number of numerical particles stable we decrease the number of particles per gridpoint, which will increase the errors due to sampling. Therefore it is important to pay attention to the combination of velocity gridpoints and numerical particle number, if one desires to produce a very accurate result.

We have run a similar experiment varying the number of spatial gridpoints. Results of that simulation are not displayed, since no significant difference between the different numbers of gridpoints were observed. This is likely due to the fact that, for the scenario we are looking at, the most significant influence on the error of the final spatial distribution is the error of the initial condition, which does not change with the number of gridpoints.

3.6 Convergence with respect to time step size

We will now examine the the influence of the number of time steps on convergence. The results of an experiment using the same settings as before, but with varying numbers of time steps, can be found in

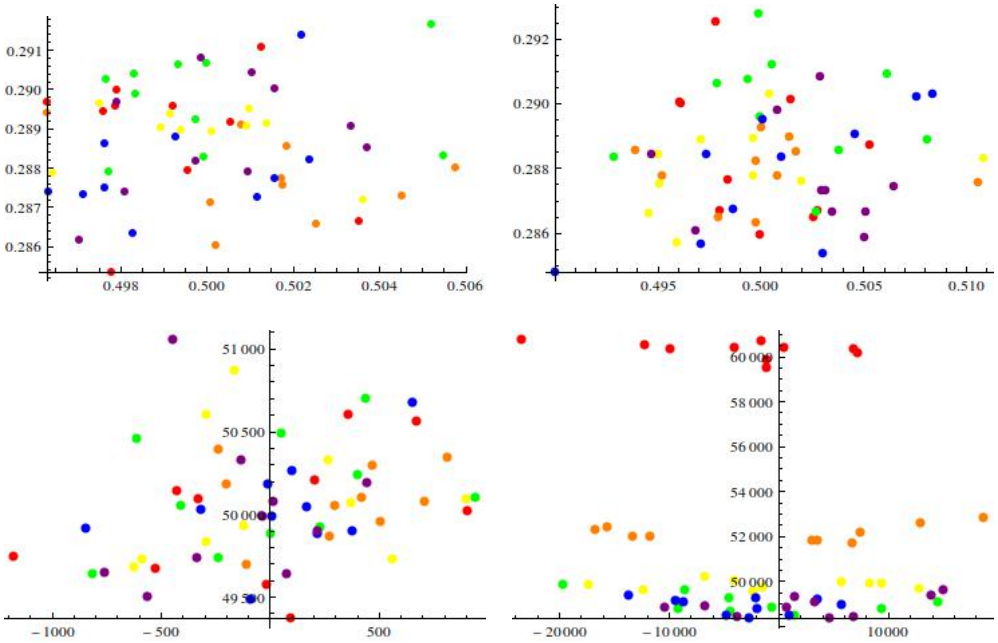


Figure 3.8: Initial (left) and final (right) values for the mean and standard deviation of the position (top) and velocity (bottom) for the experiment of Section 3.5, showing ten instances of the algorithm for each of the choices of the number of velocity gridpoints: 5 (red), 10 (orange), 20 (yellow), 40 (green), 80 (blue), 160 (purple).

figure 3.9. It seems as though the number of time steps does not make a difference, however that just means that the errors from different sources are more significant than those due to the number of time steps. We have therefore repeated the experiment using $N_v = 160$ and $N_{\text{part}} = 320000$, the results of which can be found in figure 3.10.

We see that in the latter experiment there is a clear convergence of the velocity deviations. In comparing the scales between figures 3.9 and 3.10 we see that the changes in the number of velocity gridpoints and the number of simulated particles have indeed reduced the differences between runs, as we would expect from the results of previous experiments.

3.7 Discussion

In the previous paragraphs we have seen that the algorithm does indeed converge to a final state. The results of the experiments were encouraging and warrant a more detailed analysis of the boundary conditions on the elliptic equations for the Rosenbluth potentials. It is quite likely that this would allow us to produce a more realistic final state, at which point a full three dimensional version of the algorithm would be very interesting to look at.

There are also a few other adaptations that are interesting to investigate, most notably different initializations and nonuniform grids. We have seen that, depending on the number of numerical particles, the initial distribution of the numerical particles may differ from the initial distribution we want to use. This particular effect could be almost entirely eliminated by a non-random initialization that is constructed to represent the distribution as well as possible. However, the resulting higher accuracy of such a highly structured initial condition would be misleading, and may be lost over time. Furthermore the induced symmetries of such an initial condition can lead to nongeneric behaviour. So it would be interesting use a compromise between representing the initial distribution well and making sure the numerical particles are not too correlated. Generating an initial condition using pseudorandom sampling would lead to more efficient sampling and a more uniform temporal error growth. A brief discussion of this topic can be found in [25].

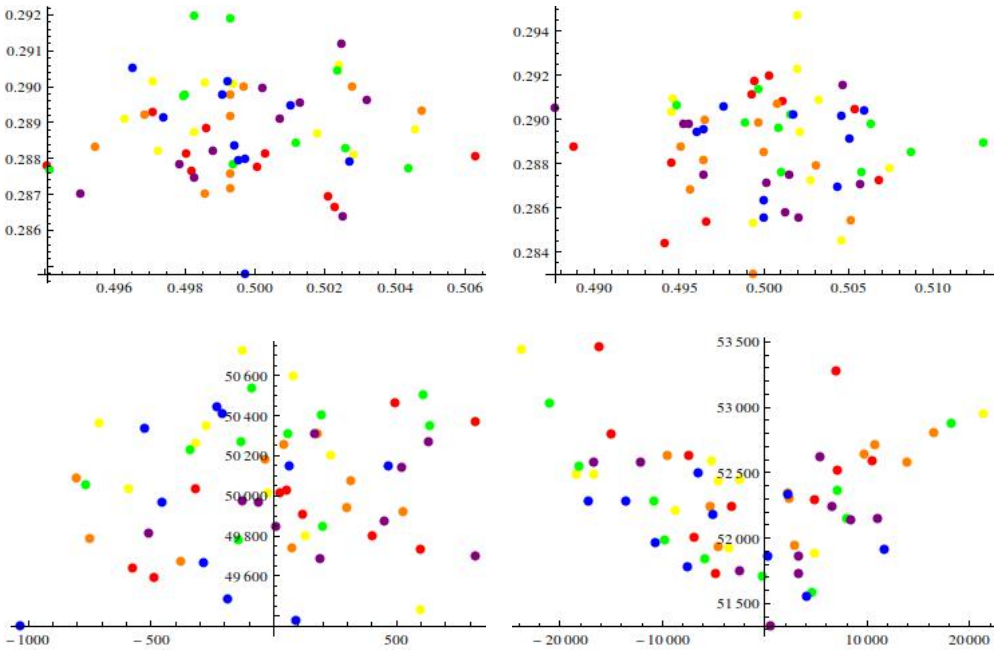


Figure 3.9: Initial (left) and final (right) values for the mean and standard deviation of the position (top) and velocity (bottom) for the experiment of Section 3.6, showing ten instances of the algorithm for each of the choices of the number of time steps: 125 (red), 250 (orange), 500 (yellow), 1000 (green), 2000 (blue), 4000 (purple).

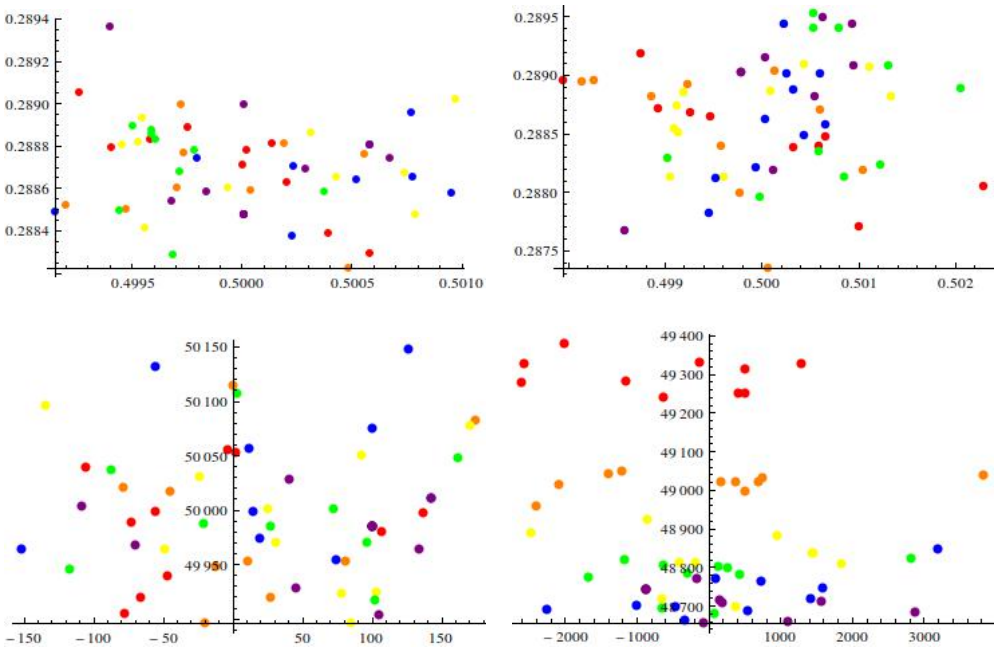


Figure 3.10: Initial (left) and final (right) values for the mean and standard deviation of the position (top) and velocity (bottom) for the modified experiment of Section 3.6, showing ten instances of the algorithm for each of the choices of the number of time steps: 125 (red), 250 (orange), 500 (yellow), 1000 (green), 2000 (blue), 4000 (purple).

Using nonuniform grids may help with better accuracy in representing nonuniform distributions, effectively allowing for the use of fewer gridpoints for the same accuracy. This may be very beneficial for the running time required, especially in a three dimensional version. Hence further research into both of these adaptations could prove most useful.

A final adaptation that will prove to be important for a three-dimensional version of the algorithm is parallelization. It will be of importance in keeping the computation time low enough while using parameters that grant high enough precision. The method of parallelizing described in [3] will work for the algorithm described in this paper and it has been shown to work well.

In summary we have seen that the PIC algorithm we have designed using the SDE description of a plasma works well, and with some adaptation may provide useful results for the simulation of ion beams.

Appendix A

Stochastic differential equations

In order to rigorously define stochastic differential equations (SDEs) and their accompanying theory, one needs concepts from advanced probability that go beyond the scope of this thesis. However one can get a basic understanding of SDEs using little more than the basic concepts of probability and deterministic differential equations. Simply put an SDE is a differential equation in which at least one of the terms is a stochastic process, making the solution a stochastic process as well. We will give a short introduction below, but for readers who want to know more a good introduction to the numerical side of SDEs is given by Higham in [26] and an extensive treatment of the theory can be found in [27].

A.1 The standard Wiener process

Most SDEs (including the one used in this thesis) make use of what is often called the 'derivative' of the standard Wiener process (also known as Brownian motion). The standard Wiener process itself is defined as a stochastic process $\{W_t \,|\, 0 \leq t \leq T\}$ such that

i) $W_0 = 0$,

ii) for any $0 \leq s \leq t \leq T$ the increment $W_t - W_s$ is a Gaussian random variable with mean 0 and variance $t - s$,

iii) for any $0 \leq s < t \leq u < v \leq T$ the increments $W_t - W_s$ and $W_u - W_v$ are independent,

iv) W_t is continuous with respect to t, with probability one.

It can be shown that W_t is non-differentiable almost everywhere, but (somewhat informally) one does often denote the quantity $\mathrm{d}W_t$ in differential equations, since integration with respect to W_t is in fact well defined. For all practical purposes we can regard $\mathrm{d}W_t$ as

$$dW_t \equiv W_{t+\mathrm{d}t} - W_t = \xi_t \,\mathrm{d}t, \quad \text{where } \langle \xi_t \rangle = 0 \text{ and } \langle \xi_t \xi_{t'} \rangle = \delta(t - t').$$

A.2 Stochastic integration

Just as with regular integration, a stochastic integral can be seen as the limit of Riemann sums. However, in the stochastic case one needs to be a bit more careful. The Itō integral (which we will use) defines the stochastic integral $\int_0^T f(t)\,\mathrm{d}W_t$ as the limit as $\Delta t \to 0$ of

$$\sum_{j=0}^{N-1} h(t_j)(W_{t_{j+1}} - W_{t_j}), \quad \text{where } t_j = j\Delta t.$$

The different choice of Riemann sum

$$\sum_{j=0}^{N-1} h\left(\frac{t_j + t_{j+1}}{2}\right)(W_{t_{j+1}} - W_{t_j}), \quad \text{where } t_j = j\Delta t,$$

which for deterministic integrals would yield the same result, gives us the Stratonovich integral, which is generally different from the Itō integral.

Note however that the freedom to 'choose' a specific Riemann sum does not mean that stochastic integrals cannot give meaningful, well defined results. The different choices lead to different versions of stochastic calculus, and given an application these will lead to different differential equations, which are solved using different definitions of integration, but they will ultimately give the same result.

A.3 Numerical solution

Suppose we would like to numerically solve the SDE

$$\mathrm{d}X(t) = f(X(t))\,\mathrm{d}t + g(X(t))\,\mathrm{d}W_t, \quad X(0) = X_0, \quad 0 \leq t \leq T. \tag{A.1}$$

If we look at its integral form

$$X(t) = X_0 + \int_0^t f(X(s))\,\mathrm{d}s + \int_0^t g(X(s))\,\mathrm{d}W_s, \quad 0 \leq t \leq T,$$

then the definition of the integrals as the limiting case of Riemann integrals suggest the following numerical approximation

$$X_j = X_{j-1} + f(X_{j-1})\Delta t + g(X_{j-1})(W_{t_j} - W_{t_{j-1}}), \quad j = 1, 2, \ldots, N$$

where $\Delta t = T/N$ for some positive integer N, $t_j = j\Delta t$ and X_j is our numerical approximation to $X(t_j)$. This method is called the Euler-Maruyama (EM) method, and is similar to the Euler forward method for deterministic differential equations.

A.3.1 Convergence

In order to tell at which rate the EM method converges we must be precise about what we mean by convergence. A method is said to have strong order of convergence γ if there exists a constant C such that

$$\mathbb{E}\|X_n - X(\tau)\| \leq C\Delta t^\gamma \quad \text{for any fixed } \tau = n\Delta t \in [0, T) \text{ and } \Delta t \text{ small enough.}$$

And the method is said to have a weak order of convergence γ if there exists a constant C such that for all polynomials $p(x)$

$$\|\mathbb{E}p(X_n) - \mathbb{E}p(X(\tau))\| \leq C\Delta t^\gamma \quad \text{for any fixed } \tau = n\Delta t \in [0, T) \text{ and } \Delta t \text{ small enough.}$$

The difference between these two is that for strong convergence we require convergence for all paths (different realizations of the Brownian motion), while for weak convergence we only require the moments of the probability distribution to converge.

It can be shown that the EM method for Equation (A.1), for appropriate f and g, has weak order of convergence 1 and strong order of convergence $\frac{1}{2}$. Note that that differs from what we would see in the deterministic case. If we were to make the equation deterministic by setting $g = 0$ we would have both weak and strong order of convergence 1.

Appendix B

Various techniques

B.1 u-v-transformation

Obtaining the final expression for $\{\Delta v_i \Delta v_j\}_{ab}$ in Section 1.3 using the previous results of that section requires a long calculation, which we show here.

$$
\begin{aligned}
\{\Delta v_{ai}\Delta v_{aj}\}_{ab} &= \left(\frac{m_b}{m_a+m_b}\right)^2 \sum_k \sum_l (\boldsymbol{e_i}\cdot\boldsymbol{e'_k})(\boldsymbol{e_j}\cdot\boldsymbol{e'_l})\{\Delta u_{kL}\Delta u_{lL}\}_{ab} \\
&= \left(\frac{m_{ab}}{m_a}\right)^2 ((\boldsymbol{e_i}\cdot\boldsymbol{e'_2})(\boldsymbol{e_j}\cdot\boldsymbol{e'_2}) + (\boldsymbol{e_i}\cdot\boldsymbol{e'_3})(\boldsymbol{e_j}\cdot\boldsymbol{e'_3}))\, 4\pi u \left(\frac{q_a q_b}{4\pi\epsilon_0 m_{ab} u}\right)^2 \ln(\Lambda) \\
&= \Gamma_a \left(\frac{q_b}{q_a}\right)^2 \frac{1}{u}\left(\frac{\boldsymbol{u}\cdot(\boldsymbol{e_i}\times\boldsymbol{e_3})\,\boldsymbol{u}\cdot(\boldsymbol{e_j}\times\boldsymbol{e_3})}{u_1^2+u_2^2} + \boldsymbol{e_i}\cdot\frac{(\boldsymbol{u}\cdot\boldsymbol{u})\boldsymbol{e_3}-(\boldsymbol{u}\cdot\boldsymbol{e_3})\boldsymbol{u})}{u(u_1^2+u_2^2)^{1/2}}\boldsymbol{e_j}\cdot\frac{(\boldsymbol{u}\cdot\boldsymbol{u})\boldsymbol{e_3}-(\boldsymbol{u}\cdot\boldsymbol{e_3})\boldsymbol{u})}{u(u_1^2+u_2^2)^{1/2}}\right) \\
&= \Gamma_a \left(\frac{q_b}{q_a}\right)^2 \frac{1}{u}\frac{1}{u_1^2+u_2^2}\left(\sum_k\sum_l \epsilon_{ki3}u_k\epsilon_{lj3}u_l + \frac{\delta_{i3}\delta_{j3}u^4 - \delta_{i3}u^2u_3u_j - \delta_{j3}u^2u_3u_i + u_3^2u_iu_j}{u^2}\right) \\
&= \Gamma_a \left(\frac{q_b}{q_a}\right)^2 \frac{1}{u}\frac{1}{u_1^2+u_2^2}\Big(\delta_{i2}\delta_{j2}u_1^2 - \delta_{i2}\delta_{j1}u_1u_2 - \delta_{i1}\delta_{j2}u_1u_2 + \delta_{i1}\delta_{j1}u_2^2 \\
&\qquad +\delta_{i3}\delta_{j3}(u_1^2+u_2^2+u_3^2) - \delta_{i3}u_3u_j - \delta_{j3}u_3u_i + \left(1-\frac{u_1^2+u_2^2}{u^2}\right)u_iu_j\Big) \\
&= \Gamma_a \left(\frac{q_b}{q_a}\right)^2 \frac{1}{u}\frac{1}{u_1^2+u_2^2}\Big(\delta_{i1}\delta_{j1}u_2^2 + \delta_{i2}\delta_{j2}u_1^2 + \delta_{i3}\delta_{j3}(u_1^2+u_2^2-u_3^2) - \frac{u_1^2+u_2^2}{u^2}u_iu_j \\
&\qquad +u_iu_j(1-\delta_{i1}\delta_{j2}-\delta_{i2}\delta_{j1}-\delta_{i3}-\delta_{j3}+2\delta_{i3}\delta_{j3})\Big) \\
&= \Gamma_a \left(\frac{q_b}{q_a}\right)^2 \frac{1}{u}\frac{1}{u_1^2+u_2^2}\left(\delta_{i1}\delta_{j1}u_2^2 + \delta_{i2}\delta_{j2}u_1^2 + \delta_{i3}\delta_{j3}(u_1^2+u_2^2-u_3^2) + \delta_{ij}u_iu_j - \frac{u_1^2+u_2^2}{u^2}u_iu_j\right) \\
&= \Gamma_a \left(\frac{q_b}{q_a}\right)^2 \frac{1}{u}\left(\delta_{ij} - \frac{u_iu_j}{u^2}\right)
\end{aligned}
$$

Where

$$\Gamma_a = \frac{q_a^4}{4\pi\epsilon_0^2 m_a^2}\ln(\Lambda)$$

and ϵ_{ijk} is the Levi-Civita symbol.

B.2 Lemma about unitary transformations

The lemma used in Theorem 1.6.2 can be found below.

Lemma B.2.1 *Unitary transformations do not change the probability density function of a vector of independent and identically distributed (iid) Gaussian random variables with mean zero.*

Proof A vector of iid Gaussian random variables with mean zero and standard deviation σ has pdf

$$f(\boldsymbol{x}) = (2\pi\sigma)^{-\frac{k}{2}} \exp(-\frac{1}{2\sigma^2}\boldsymbol{x}^T\boldsymbol{x}),$$

so we see that for any unitary matrix $\boldsymbol{U}$ we have

$$\begin{aligned} f(\boldsymbol{U}\boldsymbol{x}) &= (2\pi\sigma)^{-\frac{k}{2}} \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{U}\boldsymbol{x})^T(\boldsymbol{U}\boldsymbol{x})\right) \\ &= (2\pi\sigma)^{-\frac{k}{2}} \exp\left(-\frac{1}{2\sigma^2}\boldsymbol{x}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{x}\right) \\ &= (2\pi\sigma)^{-\frac{k}{2}} \exp\left(-\frac{1}{2\sigma^2}\boldsymbol{x}^T\boldsymbol{x}\right) \\ &= f(\boldsymbol{x}) \end{aligned}$$

which concludes this proof.

$\boxtimes$

B.3 Spatial density

Below we show the equivalence of calculating the d-dimensional spatial density ρ by integration of the phase-space density f on the $2d$-dimensional grid to direct calculation on a d-dimensional grid.

$$\begin{aligned} \rho(\boldsymbol{r}) &= q\int f(\boldsymbol{r},\boldsymbol{v})\,\mathrm{d}^d\boldsymbol{v} \\ &= q\int \sum_{\boldsymbol{n},\boldsymbol{m}}\left(f_{\boldsymbol{n},\boldsymbol{m}}\prod_{i=1}^{d}B(r_i-n_i\Delta r_i)B(v_i-m_i\Delta v_i)\right)\mathrm{d}^d\boldsymbol{v} \\ &= q\sum_{\boldsymbol{n},\boldsymbol{m}}\left(f_{\boldsymbol{n},\boldsymbol{m}}\prod_{i=1}^{d}B(r_i-n_i\Delta r_i)\int B(v_i-m_i\Delta v_i)dv_i\right) \\ &= q\sum_{\boldsymbol{n},\boldsymbol{m}}\left(f_{\boldsymbol{n},\boldsymbol{m}}\prod_{i=1}^{d}B(r_i-n_i\Delta r_i)\Delta v_i\right) \\ &= q\sum_{\boldsymbol{n},\boldsymbol{m}}\left(\left(\sum_{k=0}^{N_{\text{sim}}-1}f_{\text{part}}^{(k)}\prod_{i=1}^{d}B(r_i^{(k)}-n_i\Delta r_i)B(v_i^{(k)}-m_i\Delta v_i)\right)\prod_{i=1}^{d}B(r_i-n_i\Delta r_i)\Delta v_i\right) \\ &= q\sum_{\boldsymbol{n}}\left(\left(\sum_{k=0}^{N_{\text{sim}}-1}f_{\text{part}}^{(k)}\prod_{i=1}^{d}B(r_i^{(k)}-n_i\Delta r_i)\mathbb{1}(v_i^{(k)})\right)\prod_{i=1}^{d}B(r_i-n_i\Delta r_i)\Delta v_i\right) \\ &= \sum_{\boldsymbol{n}}\left(\left(\sum_{k=0}^{N_{\text{sim}}-1}qf_{\text{part}}^{(k)}\Delta v_1\ldots\Delta v_d\prod_{i=1}^{d}B(r_i^{(k)}-n_i\Delta r_i)\right)\prod_{i=1}^{d}B(r_i-n_i\Delta r_i)\right) \\ &= \sum_{\boldsymbol{n}}\left(\left(\sum_{k=0}^{N_{\text{sim}}-1}\rho_{\text{part}}^{(k)}\prod_{i=1}^{d}B(r_i^{(k)}-n_i\Delta r_i)\right)\prod_{i=1}^{d}B(r_i-n_i\Delta r_i)\right) \\ &= \sum_{\boldsymbol{n}}\left(\rho_{\boldsymbol{n},\boldsymbol{m}}\prod_{i=1}^{d}B(r_i-n_i\Delta r_i)\right) \end{aligned}$$

Bibliography

[1] Suzie Sheehy. *Applications of accelerators*. CERN Accelerator School lecture. Sept. 2014. URL: http://cas.web.cern.ch/cas/CzechRepublic2014/Lectures/SheehyII.pdf.

[2] Marshall N. Rosenbluth, William M. MacDonald, and David L. Judd. "Fokker-Planck Equation for an Inverse-Square Force". In: *Phys. Rev.* 107 (1 July 1957), pp. 1–6. DOI: 10.1103/PhysRev.107.1.

[3] Ji Qiang, Robert D. Ryne, and Salman Habib. "Self-consistent Langevin Simulation of Coulomb Collisions in Charged-particle Beams". In: *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*. SC '00. IEEE Computer Society, 2000. ISBN: 0-7803-9802-5. URL: http://dl.acm.org/citation.cfm?id=370049.370396.

[4] Crispin Gardiner. *Stochastic Methods - A Handbook for the Natural and Social Sciences*. Springer, 2009. ISBN: 978-3-540-70712-7.

[5] Mark H. Holmes. *Introduction to the Foundations of Applied Mathematics*. Springer, 2009. ISBN: 978-0-387-87765-5.

[6] Martha W. Evans, Francis H. Harlow, and Eleazer Bromberg. *The particle-in-cell method for hydrodynamic calculations*. Tech. rep. Los Alamos National Lab NM, 1957.

[7] J.U. Brackbill and H.M. Ruppel. "FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions". In: *Journal of Computational Physics* 65.2 (1986), pp. 314 –343. ISSN: 0021-9991. DOI: 10.1016/0021-9991(86)90211-1.

[8] John M. Dawson. "Particle simulation of plasmas". In: *Rev. Mod. Phys.* 55 (2 1983), pp. 403–447. DOI: 10.1103/RevModPhys.55.403.

[9] S.E. Parker. "A Particle-In-Cell Method for Modeling Small Angle Coulomb Collisions in Plasmas". In: *13th Conference on Numerical Simulation of Plasmas* (1989).

[10] Danilo D'Andrea, Claus-Dieter Munz, and Rudolf Schneider. *Modeling of Electron-Electron Collisions for Particle-In-Cell Simulations*. Forschungszentrum Karlsruhe, 2006.

[11] B.I. Cohen et al. "Time-Step Considerations in Particle Simulation Algorithms for Coulomb Collisions in Plasmas". In: *Plasma Science, IEEE Transactions on* 38.9 (2010), pp. 2394–2406. ISSN: 0093-3813. DOI: 10.1109/TPS.2010.2049589.

[12] A.M. Dimits et al. "Higher-order time integration of Coulomb collisions in a plasma using Langevin equations". In: *Journal of Computational Physics* 242.0 (2013), pp. 561 –580. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2013.01.038.

[13] M.S. Rosin et al. "Multilevel Monte Carlo simulation of Coulomb collisions". In: *Journal of Computational Physics* 274.0 (2014), pp. 140 –157. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2014.05.030.

[14] Charles F.F. Karney. "Fokker-Planck and quasilinear codes". In: *Computer Physics Reports* 4.3–4 (1986), pp. 183 –244. ISSN: 0167-7977. DOI: 10.1016/0167-7977(86)90029-8.

[15] L. Chacón et al. "An Implicit Energy-Conservative 2D Fokker–Planck Algorithm: I. Difference Scheme". In: *Journal of Computational Physics* 157.2 (2000), pp. 618 –653. ISSN: 0021-9991. DOI: 10.1006/jcph.1999.6394.

[16] Giacomo Dimarco, Russel Caflish, and Lorenzo Pareschi. "Direct simulation Monte Carlo schemes for Coulomb interactions in plasmas". In: *Communications in Applied and Industrial Mathematics* 1.1 (2010), pp. 72 –91. ISSN: 2038-0909. DOI: 10.1685/2010CAIM498.

[17] Matt Landreman and Darin R Ernst. "Local and global Fokker–Planck neoclassical calculations showing flow and bootstrap current modification in a pedestal". In: *Plasma Physics and Controlled Fusion* 54.11 (2012), p. 115006. DOI: 10.1088/0741-3335/54/11/115006.

[18] Jason Frank, Georg Gottwald, and Sebastian Reich. "A Hamiltonian Particle-Mesh Method for the Rotating Shallow-Water Equations". English. In: *Meshfree Methods for Partial Differential Equations*. Ed. by Michael Griebel and Marc Alexander Schweitzer. Vol. 26. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2003, pp. 131–142. ISBN: 978-3-540-43891-5. DOI: 10.1007/978-3-642-56103-0_10.

[19] Vladimir Molchanov and Marcel Oliver. "Convergence of the Hamiltonian particle-mesh method for barotropic fluid flow". In: *Math. Comp.* 82 (2013), pp. 861–891. DOI: 10.1090/S0025-5718-2012-02648-2.

[20] Matteo Frigo and Steven G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005), pp. 216–231. ISSN: 0018-9219. DOI: 10.1109/JPROC.2004.840301.

[21] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.)* Johns Hopkins University Press, 1996. ISBN: 0-8018-5414-8.

[22] Benedict Leimkuhler and Charles Matthews. "Rational Construction of Stochastic Numerical Methods for Molecular Sampling". In: *Applied Mathematics Research eXpress* 2013.1 (2013), pp. 34–56. DOI: 10.1093/amrx/abs010.

[23] Ben Leimkuhler and Charles Matthews. *Molecular Dynamics*. Springer, 2015. ISBN: 978-3-319-16374-1.

[24] François Bouchut, J Dolbeault, and et al. "On long time asymptotics of the Vlasov-Fokker-Planck equation and of the Vlasov-Poisson-Fokker-Planck system with Coulombic and Newtonian potentials". In: *Differential and Integral Equations* 8.3 (1995), pp. 487–514.

[25] Georges-Henri Cottet and Petros D. Koumoutsakos. *Vortex Methods - Theory and Practice*. Cambridge University Press, 2000. ISBN: 9780521621861.

[26] Desmond J. Higham. "An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations". In: *SIAM Review* 43.3 (2001), pp. 525–546. DOI: 10.1137/S0036144500378302.

[27] Michael J. Steele. *Stochastic Calculus and Financial Applications*. Springer, 2001. ISBN: 978-1-4419-2862-7.