Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

The $\mathrm{P^3M}$ Model on Emerging Computer Architectures With Application to Microbunching

Master's Thesis

Benjamin Ulmer

30th June, 2016

Supervisor: Andreas Adelmann (PSI)
Advisors: Phillip Colella (LBNL), John Shalf (LBNL)

D-INFK, ETH Zürich

Abstract

Space charge simulations play a major role in particle accelerator modeling. In this thesis, the parallel particle-particle, particle-mesh (P^3M) algorithm is studied towards its applicability for particle accelerator simulations. The P^3M method combines the efficiency of a grid based solver with the accuracy obtained by the direct solution of an N-body problem. The applicability of the P^3M model to beam dynamics simulations is evaluated by studying two common space charge phenomena. In particular, we use the P^3M algorithm to study microbunching effects in high relativistic electron beams and evaluate the influence of Coulomb collisions on the disorder induced heating process in low energy beams. The results of these two experiments are evaluated with regard to an integration of our implementation into a framework for general particle accelerator simulations which paves the way for start-to-end simulations for new accelerators such as free electron lasers. We show that the P^3M method provides the ability to perform a start-to-end simulation with dynamical adjustment of the contribution of Coulomb collisions to the space charge computations by a single parameter. The presented implementation is parallelized for distributed memory and we give an outlook towards further optimizations for new hardware solution such as the Intel many core architecture.

Contents

Chapter 1

Introduction

The manifold types of particle accelerators for applications reaching from research in the basic and applied sciences to medical applications such as proton therapy for cancer patients [1] have in common that they are highly complex and expensive machines. In order to design and efficiently operate such a machine, large scale computer simulations are widely used. The simulation helps the engineers and physicists to build and optimize machines for different target applications. Comparing experimental data with simulation results can help to detect undesired behavior in the accelerators and even prevent the machine from major damage.
Due to the complexity of particle accelerators and the naturally involved large number of particles in such a machine a simulation framework has to be flexible, extendable and highly efficient. With the end of Moore's law for single processor architectures, the performance of a simulation can only be increased by making use of more efficient algorithms, parallel hardware and parallel programming.
In this thesis we study the applicability of the particle-particle, particle-mesh (P^3M) method [2], which is widely used in cosmological simulations [3, 4], to particle accelerator simulations. The P^3M algorithm combines the efficiency of a mesh based method with the high resolution obtained by the particle-particle method. The implementation of the method is going to be part of the 'Object Oriented Parallel Accelerator Library' (OPAL) [5] and hence applicable to start-to-end simulations of particle beams from the gun to the target. The code is parallelized based on a domain decomposition scheme and the targeted hardware in mind is Intel's new many core (MIC) platform XeonPhi Knights Landing (KNL) [6].

1.1 Motivation

In high brightness electron beams, for example in free electron lasers, space charged induced microbunching effects can degrade the beam quality and lead to unexpected physical effects [7]. Marinelli and Rosenzweig derived an analytical formula for the microbunching gain based on the beam plasmas dielectric function [8]. To validate their theory they used a "high resolution three dimensional molecular dynamics simulation", to be able to provide the high resolution needed for the structures in the suboptical regime. Since common molecular dynamics simulations scale with $\mathcal{O}(N^2)$ [9] where N is the number of particles this is a computationally very demanding approach. However, space charge fields can be very well simulated using grid based methods such as particle in cell (PIC) codes [10] with limited resolution based on the choice of the mesh size. A molecular dynamics simulation for the microbunching effects seems to be a good candidate to profit from a hybrid P^3M method combining the efficiency of grid based methods with the high resolution offered by N^2 molecular dynamics simulation. The particle-particle part of the P^3M algorithm will account for the small scale dynamics and the high resolution needed, while the particle-mesh part will add the long range contributions of the Coulomb interactions by an efficient grid based method. Furthermore, the P^3M code allows us to study the effect of Coulomb collisions between particles in the beam which can influence the beam temperature [11, 12]. This effect is of special interest right behind the electron gun, where the beam is rather slow and the shot-noise within the particle distribution is high. With the integration of the P^3M implementation into the OPAL framework it will be possible to simulate a particle beam throughout the whole machine. The contribution of Coulomb collisions can be dynamically adjusted for the needs of every simulated element of the accelerator.

1.2 Related Work

The microbunching studies in this thesis are based on the analytical formula for the microbunching gain in high relativistic electron beams elaborated by Marinelli and Rosenzweig [8]. They base their theory on one-dimensional plasma oscillations in warm electron plasmas described by the Vlasov equation. Furthermore, they present their simulation results and compare them with the theoretical microbunching gain formula obtained. The results of their molecular dynamics simulation shall serve as a reference for the P^3M simulation, which is presented in this thesis. The P^3M algorithm was developed by Hockney and Eastwood [2] and to a large extend, our implementation follows their original approach.
The parallelization techniques presented in this thesis are inspired by the

parallelization strategies presented in [3, 4] in the context of cosmological simulations and are applied with a hybrid shared and distributed memory architecture in mind.
The only appearance of the P^3M method in the context of space charge simulations for particle accelerator found in published literature is the work by Mitchell and Qiang [12]. They study the influence of Coulomb collisions in low energy electron beams. We use this simulation as an example for the application of our P^3M impelementation to intra beam scattering effects. The results of their simulations for disorder induced beam heating can be successfully reproduced with our implementation.

1.3 Thesis Outline

After a presentation of the original serial P^3M algorithm in chapter 2 the extensions to introduce periodic boundary conditions and parallelization strategies are presented in chapter 3. Since the charge decomposition scheme is a major part of the particle-mesh algorithm, we dedicated section 2.5 to an a priori error analysis for the required order of the applied charge decomposition scheme. Emphasis was placed on regression tests, to ensure correct results given by the implementation for the serial, as well as for the parallel execution case. The regression test results are presented in chapter 5. In chapter 6 the analytic solution for the microbunching gain [8] is presented followed by the results of the numerical experiments performed in 7. To take a closer look at the resolution of Coulomb collisions, another numerical experiment on disorder-induced heating [12] is presented in chapter 8. In chapter 9 we show a performance analysis of the code presented which serves as a basis for the outlook towards an efficient many core implementation given in chapter 10. In the last chapter a summary of the thesis work is given. The relevance and the applicability of the results obtained in this thesis for start-to-end particle accelerator simulations are discussed.

Chapter 2

The Serial $\mathrm{P^3M}$ Algorithm

In this chapter we describe the working horse of the presented thesis, the $\mathrm{P^3M}$ algorithm. The derivation given here is based on the original algorithm presented in [2]. We start with a short derivation of the potentials, charges and fields present in a system of charged particles. After an outline of the $\mathrm{P^3M}$ algorithm itself, a closer look is taken at the various splitting methods for the long range and short range interaction force contributions. This is followed by a discussion of the different choices of charge deposition schemes.

2.1 Interactions in a System of Charged Particles

In this section we derive the forces, potentials and energies present in a system of charged particles. In the following derivation we assume a system of N charged particles with positions $\vec{x}_i$ and charge q_i. The distance vector pointing from particles i to particle j is denoted $\vec{r}_{ij} = \vec{x}_j - \vec{x}_i$. The absolute distance between the two particles is denoted as $r_{ij} = |\vec{x}_j - \vec{x}_i|$ and $\hat{\vec{r}}_{ij} = \frac{\vec{r}_{ij}}{r_{ij}}$ is the unit vector pointing from i to j.

Coulomb's Law. The electrostatic interaction between charged particles is governed by Coulomb's law. It states that the magnitude of the electrostatic force between two charged particles is inversely proportional to the squared distance between the particles

$$\vec{F}_{ij} = k_e \frac{q_i q_j}{r_{ji}^2} \hat{\vec{r}}_{ji} \ . \tag{2.1}$$

$\vec{F}_{ij}$ denotes the force on particle i with respect to a particle j. Coulomb's constant is given by $k_e = \frac{1}{4\pi\varepsilon_0}$ with the electric permittivity ε_0. For a system

of discrete point charges it follows for the force on particle i that

$$\vec{F}(\vec{x}_i) = k_e q_i \sum_{j=1, j\neq i}^{N} q_j \frac{\hat{\vec{r}}_{ji}}{r_{ji}^2} \,. \tag{2.2}$$

The Coulomb potential for a single particle i in a system of discrete point charges is given by

$$\Phi_{coul}(\vec{x}_i) = k_e \sum_{i=1, i\neq j}^{N} \frac{q_j}{r_{ij}} \,. \tag{2.3}$$

The corresponding potential energy $U_{coul}(\vec{x}_i)$ of particle i is

$$U_{coul}(\vec{x}_i) = q_i \Phi_{coul}(\vec{x}_i) = k_e q_i \sum_{i=1, i\neq j}^{N} \frac{q_j}{r_{ij}} \tag{2.4}$$

and hence the electrostatic potential energy U stored in the system is

$$U = \frac{1}{2} \sum_{i=1}^{N} q_i \Phi(\vec{x}_i) = \frac{1}{2} k_e \sum_{i=1}^{N} q_i \sum_{i=1, i\neq j}^{N} \frac{q_j}{r_{ij}} \,. \tag{2.5}$$

Finally the electrostatic electric field felt by particle i becomes

$$\vec{E}(\vec{x}_i) = k_e \sum_{j=1}^{N} \frac{q_j}{r_{ij}^2} \hat{\vec{r}}_{ji} = \sum_{j=1}^{N} \frac{\vec{F}_{ij}}{q_i} \,. \tag{2.6}$$

Poisson's Equation. The electric potential Φ and the charge distribution ρ are related as

$$\nabla^2 \Phi = -\frac{\rho}{\varepsilon_0} \tag{2.7}$$

known as *Poisson's equation*. From equation 2.3 we know the electric potential of the charged particle system. If we assume a continuous charge distribution instead of the discrete particle charges we can write

$$\Phi(\vec{x}) = k_e \int \frac{\rho(\vec{x}')}{|\vec{x} - \vec{x}'|} \mathrm{d}V' \tag{2.8}$$

with volume element $\mathrm{d}V'$. In other words the Green's function solution of Poisson's equation is given by the convolution

$$\Phi(\vec{x}) = k_e \int G(\vec{x}) \rho(\vec{x}') \mathrm{d}V' \tag{2.9}$$

with Green's function $G(\vec{x}) = \frac{1}{|\vec{x} - \vec{x}'|} = \frac{1}{r}$ which is the interaction function between two charged particles at positions $\vec{x}$ and $\vec{x}'$ separated by distance r. The convolution in equation 2.9 becomes a multiplication in Fourier space, indicated by a small hat,

$$\hat{\Phi}(\vec{\xi}) = k_e \hat{G}(\vec{\xi}) \hat{\rho}(\vec{\xi}), \quad \vec{\xi} \in \mathbb{R}^3 \tag{2.10}$$

and can be solved efficiently using the fast Fourier transform (FFT) method.

2.2 The Hockney and Eastwood $\mathrm{P^3M}$ Algorithm

Hockney and Eastwood introduced in [2, chapter 8] a hybrid algorithm for the efficient computation of short and long range interaction between charged particles. The central idea of the algorithm is splitting the Coulomb interaction force F, as introduced in equation (2.1), into a short range and a long range part. With *short range* contribution $F_{SR}(r)$ and *long range* contribution $F_{LR}(r)$, where r denotes the distance between the particles, one can write the magnitude of the Coulomb force $F(r)$ as

$$F(r) = F_{SR}(r) + F_{LR}(r) \ . \tag{2.11}$$

The *long range forces* in the system are computed by solving Poisson's equation in Fourier space using a grid based method such as a particle in cell (PIC) method [13]. This reduces the $\mathcal{O}(N^2)$ computations needed to compute all pairwise interactions in real space to $\mathcal{O}(N_{\mathcal{M}} \log N_{\mathcal{M}})$ computations in Fourier space by making use of the fast Fourier transform (FFT) method. Here $N_{\mathcal{M}}$ denotes the number of mesh points used for the FFT.
The *short range interaction* is computed directly in real space by computing all pairwise interaction as in equation (2.2). For the computation of the short range interaction only particles within a certain cutoff radius r_c are considered. The Newtonian equations of motion are integrated in time by making use of a numerical integration scheme such as the leapfrog algorithm or Runge-Kutta integration schemes.

The Serial Algorithm. The original algorithm presented in [2] consists of three major steps presented below. Implementational details and parallelization strategies are presented in chapter 4. Initially we are given $i = 1, ..., N$ particles with positions $\vec{x}_i^0$ and momenta $\vec{p}_i^0$. For the particle-mesh solver we assume that we are provided with a uniform Cartesian mesh $\mathcal{M}_{PM}$. The following steps are repeated until the desired simulation time $T_{end} = n\Delta t$ is reached after n time steps. For time integration the leap-frog scheme (2.12) and (2.13) is applied for every particle i,

$$\vec{x}_i^{n+1} = \vec{x}_i^n + \frac{\vec{p}_i^{n+1/2}}{m_i}\Delta t \ , \tag{2.12}$$

$$\vec{p}_i^{n+1/2} = \vec{p}_i^{n-1/2} + \vec{F}\Delta t \ . \tag{2.13}$$

The P^{3}M algorithm reads as follows:

1. Particles-mesh (PM) force calculations: Firstly a charge deposition scheme of order p is used to assign the discrete particle charges q_i to the grid $\mathcal{M}_{PM}$. Then we compute the discrete long-range potential on the grid by making use of fast Fourier transformations (FFT) to

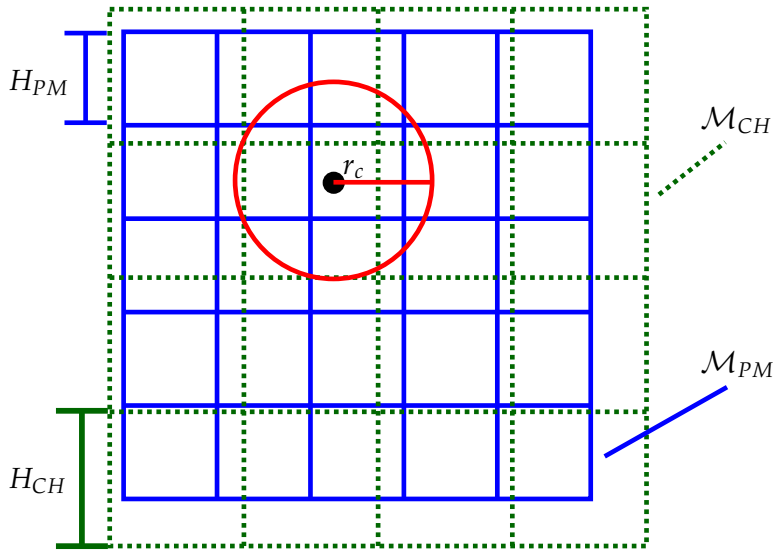

Figure 2.1: Chainingmesh $\mathcal{M}_{CH}$ in dotted green and computational mesh for PM solver in $\mathcal{M}_{\mathcal{PM}}$ in solid blue lines.

solve Poisson's equation. The long-range forces are interpolated back to the particles by using the same deposition scheme as used for the charge assignment to the grid. Finally we increment the momenta of every particle.

2. Particle-particle (PP) force calculation: For every particle in the system search for all particles within a cutoff radius r_c and compute all pairwise interactions with these particles using the short range interaction function $F_{SR}(r)$ and update the momenta of each particle by the short-range contribution. Hockney and Eastwood suggest the introduction of a chaining mesh $\mathcal{M}_{\text{CH}}$ with mesh width $H_{\text{CH}} > r_c$ as shown in Figure 2.1. In each time step, the particles are bin sorted into one of these chaining cells. The neighbors within the cutoff r_c can then be found by looking at every neighboring chaining cell (and the chaining cell itself) of the cell containing the particle of interest. This can be implemented using a double linked list.

3. Update the positions of every particle under consideration of the desired boundary conditions. We arrive at the new positions $\vec{x}_i^{n+1}$ and momenta $\vec{p}_i^{n+1/2}$.

Note on Periodicity. Naturally fast Fourier transform methods are periodic in all dimensions. If open boundary conditions are desired, one has to double the computational domain in each dimension for the FFT solver as

described in [14, 15]. However, for the simulations used in this thesis we assume periodic boundary conditions. Hence, we only have to make sure that the discretized Green's functions provides the correct periodicity and that the particle-particle interactions respect the periodic boundary conditions as well. The treatment of periodic boundary conditions is presented in more detail in chapter 3.

Interaction Function Splitting. Equivalent to splitting the inter particle forces, as shown in section 2.1, we can also split the interaction potential function $G(r)$ in a short range contribution $\psi(r)$ and a long range contribution $\varphi(r)$ such that

$$\frac{1}{r} = G(r) = \psi(r) + \varphi(r) \; . \tag{2.14}$$

The requirements to the interaction splitting 2.14 are that the short range part $\psi(r)$ goes smoothly to zero at a certain cutoff r_c such that the particle-particle computations can be truncated at this cutoff and the long range part $\varphi(r)$ should be smooth and nonsingular. Infinitely many of these interaction splittings exists. However, since a smoothing of the Coulomb reference force is equivalent to ascribing a finite size radius to the charged particle, a commonly used approach is to assume the particles take on the shape of a uniformly charged sphere (shape S1), a sphere with uniformly decreasing density (shape S2) or a Gaussian charge distribution (shape S3) [2]. The advantages and disadvantages of the different interaction splitting techniques are discussed in [16]. In this thesis we follow the most popular approach using Gaussian shaped charges (S3), because the choice of the splitting has only minor impact on the accuracy of the method [16, 17]. The interaction splitting reads

$$G(r) = \psi(r) + \varphi(r) = \frac{1 - f(\alpha r)}{r} + \frac{f(\alpha r)}{r} \; . \tag{2.15}$$

The short range part $\psi(r) = \frac{1-f(\alpha r)}{r}$ is supposed to be equal to zero beyond a certain cutoff radius r_c and can therefore easily computed in real space. The long range part $\varphi(r) = \frac{f(r)}{r}$ is supposed to decay rapidly in Fourier space. The parameter α controls the influence of long and short range parts.
If $f(\alpha r)$ is chosen to be $\operatorname{erf}(\alpha r) = \frac{2}{\sqrt{\pi}} \int_0^{\alpha r} e^{-\tau^2} \mathrm{d}\tau$ the splitting becomes equivalent to Gaussian shaped (S3) screening charges. In Figure 2.2 we show the Green's function splitting based on Gaussian screening for $\alpha = 2.0$ Compared to that a splitting based on a polynomial fit through the origin [18, sec. 1.4] resulting in long range contribution

$$\varphi(r) = \begin{cases} G(r), & r > r_c \\ -\frac{3r^2}{r_c^4} + \frac{4r}{r_c^3}, & r \leq r_c \end{cases} \tag{2.16}$$

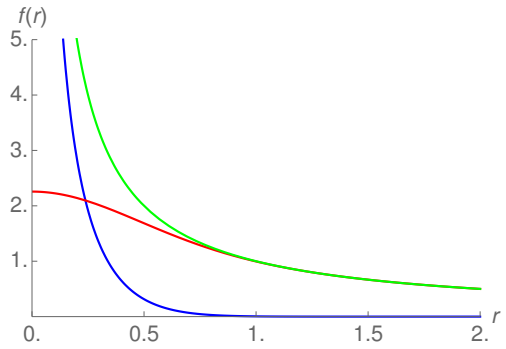

Figure 2.2: Greens function splitting based on Gaussian charge screening for $r_c = 1$. $G(r)$ in green, $\varphi(r)$ in red and $\psi(r)$ in blue.

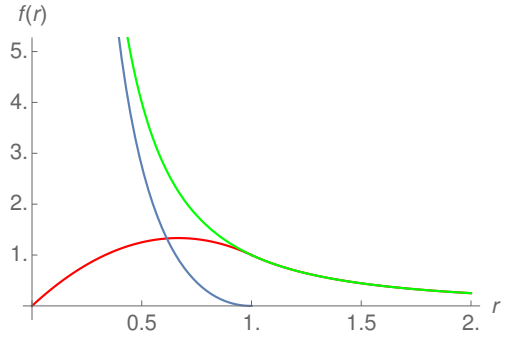

Figure 2.3: Greens function splitting based on polynomials for $r_c = 1$. $G(r)$ in green, $\varphi(r)$ in red and $\psi(r)$ in blue.

and short range contribution

$$\psi(r) = G(r) - \varphi(r) = \begin{cases} 0, & r > r_c \\ \frac{1}{r^2} + \frac{3r^2}{r_c^4} - \frac{4r}{r_c^3}, & r \leq r_c \end{cases} \tag{2.17}$$

is shown in Figure 2.3.

Remark on the Choice of the Lattice Green's Function. The simplest way of choosing the lattice Green's function G_{PM} is to set it equal to the continuous long range part of the interaction $\varphi(\vec{r})$. However, this can result in poor accuracy due to assigning the particles a finite size during the discretization on the mesh [17]. An analytic formula for the optimal lattice Green's function is given in [2]. For the computations performed within this thesis, we are satisfied to the accuracy provided by the non-optimized Green's function giving good results.

2.3 Computation of the Lattice Green's Function

There are several ways to compute the lattice Green's function for the particle-mesh part of P^3M in order to solve Poisson's equation

$$\nabla^2\Phi = -\frac{\rho}{\varepsilon_0}. \tag{2.18}$$

In this section we outline various approaches for the Green's function solution of (2.18) in Fourier space and relate them to each other. The advantages and disadvantages of the different approaches are also discussed.

2.3.1 Method 1: Continuous Space Green's Function

Following the standard Green's function approach, we search for a solution $G(\vec{x}, \vec{x}')$ of equation (2.18) with a delta function $\delta(\vec{x} - \vec{x}')$ as source term reading

$$\nabla^2 G(\vec{x}, \vec{x}') = -\delta(\vec{x} - \vec{x}') \ . \tag{2.19}$$

Making use of Green's identity [19] a general solution for (2.18) is given by

$$\Phi(\vec{x}) = \int G(\vec{x}, \vec{x}')\rho(\vec{x}')\mathrm{d}V', \tag{2.20}$$

with continuous space Green's function

$$G(\vec{x}, \vec{x}') = k_e \frac{1}{|\vec{x} - \vec{x}'|} \ . \tag{2.21}$$

Again, $k_e = \frac{1}{4\pi\varepsilon_0}$ denotes Coulomb's constant. The convolution in equation (2.20) can efficiently be solved by transformation to Fourier space where the convolution becomes a multiplication

$$\hat{\Phi}(\vec{\xi}) = -\hat{\rho}(\vec{\xi})\hat{G}(\vec{\xi}), \quad \vec{\xi} \in \mathbb{R}^3 \ . \tag{2.22}$$

For the implementation in the P^3M algorithm this approach is equivalent to computing the discretized real space Green's function and transforming it to Fourier space.

2.3.2 Method 2: Apply Laplace Operator in Fourier Space

Another approach for solving (2.18) is given by transforming Φ and ρ to Fourier space and apply the Laplace operator in Fourier space. Equation (2.18) becomes

$$\nabla_{x^2} \int \Phi(x) e^{-ix\xi} \mathrm{d}x = -\int \rho(x) e^{-ix\xi} \mathrm{d}x, \tag{2.23}$$

here only shown for the x direction. The y and z direction solutions are obtained analogously. Since the Laplace operator ∇_x^2 only operates on the

spatial coordinate x and can be pulled into the integral and applied on $e^{-i\xi x}$ we can write

$$-\xi^2 \int \Phi(x) e^{-ix\xi} \mathrm{d}x = -\int \rho(x) e^{-ix\xi} \mathrm{d}x \; . \tag{2.24}$$

With $\hat{f}$ denoting the Fourier transformation of a function f we arrive at

$$\hat{\Phi}(\xi) = \frac{1}{\xi^2} \frac{\hat{\rho}(\xi)}{\varepsilon_0} \; . \tag{2.25}$$

In the $\mathrm{P^3M}$ implementation this would be equivalent to computing $\hat{G} = \frac{1}{\xi^2 \varepsilon_0}$ directly in Fourier space and multiply with $\hat{\rho}$.

2.3.3 Method 3: Use Finite Differences in Real Space

The third approach approximates ∇_x^2 in real space by subsequent application of Fourier transformation on the approximated quantities. This approach leads to the solution in Fourier space given by

$$\hat{\Phi}(\xi_1, \xi_2, \xi_3) = \frac{0.5 H^2}{3 - \cos\left(\frac{2\pi}{N}\xi_1\right) - \cos\left(\frac{2\pi}{N}\xi_2\right) - \cos\left(\frac{2\pi}{N}\xi_2\right)} \frac{\hat{\rho}(\xi_1, \xi_2, \xi_3)}{\varepsilon_0} \tag{2.26}$$

with $\vec{\xi} = (\xi_1, \xi_2, \xi_3)^T \in \mathbb{R}^3$. Method 3 is equivalent to method 2. This can easily be seen by looking at the Taylor expansion of $\cos\left(\frac{2\pi}{N}\xi\right)$

$$\cos\left(\frac{2\pi}{N}\xi\right) = 1 - \frac{1}{2}\left(\frac{2\pi}{N}\xi\right)^2 + \mathcal{O}\left(\xi^4\right), \tag{2.27}$$

up to second order. The detailed derivation of equation 2.26 can be found in appendix A.

2.3.4 Choice of Green's Function Approximation for the $\mathrm{P^3M}$ Solver

Since we do the splitting of the interaction function G in real space the most obvious implementation computes the Green's function in real space and transforms it to Fourier space following 2.3.1. The costs of the additional Fourier transform needed are negligible since the Green's function can be computed once in the very beginning of the simulation and has to be updated only if the computational grid changes. However, for the simulations performed in the present work a change of grid is not required.

2.4 Computational Complexity

With the interaction splitting introduced in the previous section, the short range forces for each particle i can be approximated by the direct sum

$$\vec{F}_{SR}(\vec{x}_i) = -\sum_{j=1}^{n} q_i (\nabla\psi)(\vec{x}_j - \vec{x}_i) \tag{2.28}$$

using N-body calculations. For the long range forces we find

$$\vec{F}_{SR}(\vec{x}_i) = -\sum_{j=1}^{n} q_i(\nabla\varphi)(\vec{x}_j - \vec{x}_i) \tag{2.29}$$

which we approximate using a grid based particle in cell method.
To the costs of the particle mesh calculations contribute the charge deposition which can be done in $\mathcal{O}(N)$ and the fast Fourier transform which has complexity $\mathcal{O}(N_{\mathcal{M}} \lg N_{\mathcal{M}})$, where $N_{\mathcal{M}}$ denotes the number of mesh points. Hence we find a total complexity for the particle mesh calculation of

$$\text{Cost(PM)} = \mathcal{O}(N) + (N_{\mathcal{M}} \lg N_{\mathcal{M}}) \ . \tag{2.30}$$

Since the short range potential is chosen in a way such that the short range forces vanish for distances larger than the cutoff r_c, the costs of the N^2 solver reduce to

$$\text{Cost(PP)} = \mathcal{O}(N\delta^3)N = \mathcal{O}(N^2\delta^3) \ , \tag{2.31}$$

where δ denotes relative cutoff radius $\delta = \frac{r_c}{L}$ with domain length L in all three spatial dimensions for a uniform charge distribution.

2.5 A priori Error Analysis

In this section we analyze the influence of the order of the charge deposition scheme to the convergence behavior of the PM part of the solver. The PP part of the solver is an exact method without any approximations. Therefore no error analysis for the PP part is needed.
We are interested in the solution of Poisson's equation, which can be written as

$$\nabla^2\Phi = -\rho \ , \tag{2.32}$$

where Φ denotes the electrostatic potential and ρ is the charge density. We firstly look at the solution for two particles at position $\vec{y}_0$ and $\vec{x}$. The particle at position $\vec{y}_0$ is carrying charge q_0. If $G(\vec{y}) = G(\vec{x} - \vec{y})$ is the Green's function solving equation (2.32) the potential for the two particle system can be written as

$$\Phi(\vec{x}) = G(\vec{x} - \vec{y}_0)q_0 \ , \tag{2.33}$$

as shown in section 2.1. The discrete charge density $\rho_{\vec{i}}$ on a grid with mesh width h can be expressed in terms of the deposition function $\Psi(\vec{i}h - \vec{y}_0)$ as

$$\rho_{\vec{i}} = \sum_k q_k \Psi(\vec{i}h - \vec{y}_0) \ , \tag{2.34}$$

where $\vec{i} \in \mathbb{C}^3$ is the index vector of the grid. The charge deposition function $\Psi(\vec{i}h - \vec{y}_0)$ assigns the point charge at $\vec{y}_0$ to the computational grid with

edge length h. The sum over k runs over all particles in the system. With the Green's function evaluated at the grid points $\vec{i}h$ reading $G(\vec{x} - \vec{i}h)$ we can approximate the right hand side of (2.33) for the two particle system as

$$G(\vec{x} - \vec{y}_0)q_0 \approx \sum_{\vec{i}} G(\vec{x} - \vec{i}h)q_0\Psi(\vec{i}h - \vec{y}_0) \tag{2.35}$$

$$\Rightarrow G(\vec{x} - \vec{y}_0) \approx \sum_{\vec{i}} G(\vec{x} - \vec{i}h)\Psi(\vec{i}h - \vec{y}_0)\ . \tag{2.36}$$

In the following we will give an analytic estimate for the error introduced by this approximation. For this error analysis we will assume that the Green's function for the PM part of the solver agrees with the analytic Green's function $G(\vec{r}) = \frac{1}{|\vec{r}|}$ for particle distances $r = |\vec{x} - \vec{y}|$ beyond the cutoff $r_c = \delta$ and approaches smoothly a constant value at $r = 0$.

Using a Taylor expansion of the Green's function $G(\vec{y}) = G(\vec{x} - \vec{y})$ about $\vec{y} = \vec{y}_0$ up to order P, we can express the right hand side of equation (2.36) as

$$\begin{aligned}\sum_{\vec{i}} G(\vec{x} - \vec{i}h)\Psi(\vec{i}h - \vec{y}_0) \approx\ & G(\vec{x} - \vec{y}_0)\sum_{\vec{i}}\Psi(\vec{i}h - \vec{y}_0) \\ &+ \sum_{0<|p|<P} \frac{(-1)^{|p|}}{p!}\left(\nabla^p G\right)(\vec{x} - \vec{y}_0)\sum_{\vec{i}}(\vec{i}h - \vec{y}_0)^p\Psi(\vec{i}h - \vec{y}_0) \\ &+ \mathcal{O}\left(\frac{h^P}{\max\left(|\vec{x} - \vec{y}_0|, \delta\right)^{P+1}}\right)\ .\end{aligned} \tag{2.37}$$

The factor $(-1)^{|p|}$ in the second summand arises from the inner derivative of $G(\vec{x} - \vec{y})$ after application of the chain rule. In the third summand we made use of the fact, that the Green's function within the cutoff r_c is proportional to $\frac{1}{\delta}$ by construction of the Green's function for the particle mesh part. The *conservation of charge* condition for the deposition function reads

$$\sum_{\vec{i}}\Psi(\vec{i}h - \vec{y}_0) = 1, \quad \forall\vec{y}_0\ . \tag{2.38}$$

And the *moment conditions* for the $p-$th moment with $0 < |p|_1 < P$ reads

$$\sum_{\vec{i}}(\vec{i}h - \vec{y}_0)^p\Psi(\vec{i}h - \vec{y}_0) = 0, \quad \forall\vec{y}_0\ . \tag{2.39}$$

Hence, the error introduced by a single particle due to charge deposition onto the grid reads

$$\varepsilon_{\text{single}} = \mathcal{O}\left(\frac{h^P}{\max\left(|\vec{x} - \vec{y}_0|, \delta\right)^{P+1}}\right)\ . \tag{2.40}$$

Collection of Particles. Let us assume that the deposition function Ψ is charge conserving and fulfills the moment conditions up to order $P-1$. If we look at a collection of k particles $\{\vec{y}_k, q_k\}$ equation (2.35) becomes

$$\sum_k G(\vec{x}-\vec{y}_k)q_k = \sum_k G(\vec{x}-\vec{y}_k)q_k + \sum_k q_k \mathcal{O}\left(\frac{h^P}{\max\left(|\vec{x}-\vec{y}_k|,\delta\right)^{P+1}}\right) . \tag{2.41}$$

The error for n particles reads

$$\varepsilon_n = \sum_{k=1}^{n} q_k \mathcal{O}\left(\frac{h^P}{\max\left(|\vec{x}-\vec{y}_k|,\delta\right)^{P+1}}\right) . \tag{2.42}$$

Error for Constant Charges. Let the particles be uniformly distributed with uniform charge density $\bar{\rho}$. We can now express the sum over the particles k in (2.35) as a sum over spherical shells of thickness δ. The volume of the r-th shell is

$$\begin{aligned} V_{\text{shell}} &= \frac{4}{3}\pi\left[((r+1)\delta)^3 - (r\delta)^3\right] \\ &= \frac{4}{3}\pi\left[((r\delta+\delta)^3 - (r\delta)^3\right] \\ &= \frac{4}{3}\pi\left[(r\delta)^3 + 3r^2\delta^3 + 3r\delta^3 + \delta^3 - (r\delta)^3\right] \\ &= \frac{4}{3}\pi\left[3r^2\delta^3 + 3r\delta^3 + \delta^3\right] \\ &= \mathcal{O}(r^2\delta^3) . \end{aligned} \tag{2.43}$$

With the above assumptions the charge per shell is proportional to $\rho r^2\delta^3$ and the maximum distance of a particle in shell r to a reference particle is $\mathcal{O}(r\delta)$. For the n-particle error we can now write

$$\begin{aligned} \varepsilon_n = \sum_k q_k \mathcal{O}\left(\frac{h^P}{\max\left(|\vec{x}-\vec{y}_k|,\delta\right)^{P+1}}\right) &= \sum_{r>0}^{\infty} \mathcal{O}\left(\frac{h^P}{(r\delta)^{P+1}}\bar{\rho}r^2\delta^3\right) \\ &= \mathcal{O}\left(\frac{h^P}{\delta^{P-2}}\bar{\rho}\sum_{r>0}^{\infty}\frac{1}{r^{P-1}}\right) . \end{aligned} \tag{2.44}$$

To obtain h^2 convergences the following conditions have to hold

$$P > 2 \tag{2.45}$$

$$\delta = \Theta(h) . \tag{2.46}$$

This implies that a *quadratic deposition function* is needed.

Chapter 3

Periodic Boundary Conditions

In this chapter periodic boundary conditions are introduced. The original P^{3}M algorithm presented in [15] assumes isolated boundary conditions. For periodic boundaries one has to adapt both the particle-mesh solver as well as the particle-particle solver. We present our implementation of periodic boundary conditions integrated in the IPPL (Independent Parallel Particle Layer) Framework [18].

3.1 Particle-Mesh Solver

When periodic boundary conditions are applied, one can exploit the periodicity contained naturally in the discrete Fourier transformation. The discrete Fourier transformation computed by the fast Fourier transformation algorithm reads

$$\hat{x}_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i k n/N}, \quad k = 0, ..., N-1 \tag{3.1}$$

which is $N-$periodic. Thus the Fourier transformation of the charge density field mimics a periodic extension of the charge density in all three dimensions. Periodicity is also given for the backward transformation which reads

$$x_n = \sum_{k=0}^{N-1} \hat{x}_k e^{2\pi i k n/N}, \quad n = 0, ..., N-1\ . \tag{3.2}$$

In the isolated system solution a doubling of the domains in all three dimensions can be used [14, 15], which becomes obsolete in the periodic case. However, attention has to be paid on the symmetry of the lattice Green's function with respect to the summation order provided by the used fast Fourier transform implementation.

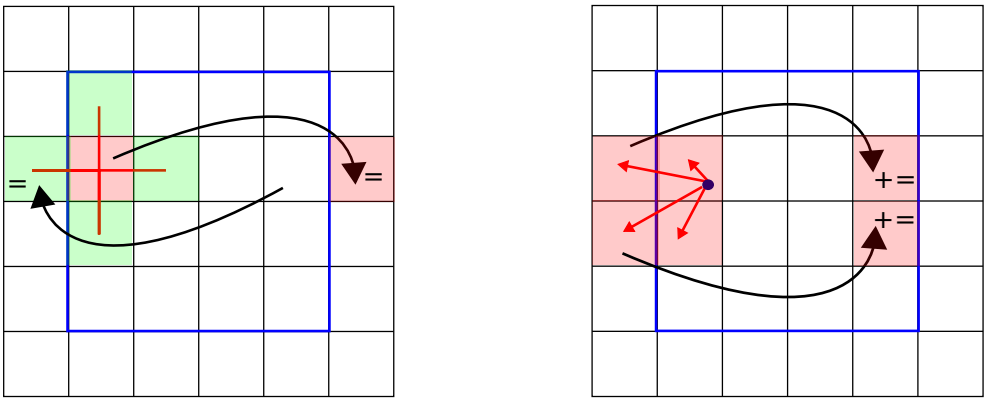

Figure 3.1: Five point stencil to approximate gradient (left) and cloud in cell interpolation stencil (right). The computational domain is inside the blue frame. The stencils need one ghost layer around the actual computational domain. Red cells indicate 'dirty' cells where an update happened, green cells are needed to read from. The black arrows indicate the ghost cell exchange. Cumulative updates are indicated by '$+=$' and overwriting updates by '$=$'.

3.1.1 Periodic Grid Interpolation

In a periodic implementation additional precaution has to be taken on the boundary conditions used for e.g. the charge deposition scheme or the finite difference operations on the grid. While finite different stencils only need ghost cell information to compute the stencil update on a computational cell, interpolation kernels also update the ghost cell values. Therefore the periodic boundary conditions can look different for the various grid operations. The electric field is obtained by applying a second order finite difference stencil on the potential grid as depicted in Figure 3.1 (left). For this operation the standard periodic boundary conditions implemented in the IPPL framework can be used. They ensure that access of a cell with index $-I$ returns the corresponding periodic cell image with index $-I + N$, where N is the number of cells in the corresponding direction. Those ghost cell values have to be updated to their corresponding actual cell value whenever a value in a boundary cell changes.
In the case of an interpolation operation like the cloud in cell (CIC) scheme we used for charge deposition on the grid. Charge is assigned to neighboring cells as depicted in Figure 3.1 (right). However no information from the ghost cells is needed. After the deposition of all point charges to the grid, the values in the ghost cell layers have to be accumulated to the corresponding cells in the actual domain. Additional precaution has to be taken in parallel implementations. Periodic boundary conditions are non local and in the parallel case the source and destination domains for the ghost cells

```
for (unsigned int i = 0; i < 2 * Dim; ++i) {
//use periodic boundary conditions for the particles
  this->getBConds()[i] = ParticlePeriodicBCond;
  //boundary conditions used for interpolation kernels allow writes to ghost cells
  if (Ippl::getNodes()>1) {
    bc_m[i] = new ParallelInterpolationFace<double, Dim, Mesh_t, Center_t>(i);
    //std periodic boundary conditions for gradient computations etc.
    bc_m[i] = new ParallelPeriodicFace<Vector_t, Dim, Mesh_t, Center_t>(i);
    bcp_m[i] = new ParallelPeriodicFace<double, Dim, Mesh_t, Center_t>(i);
  }
  else {
    bc_m[i] = new InterpolationFace<double, Dim, Mesh_t, Center_t>(i);
    //std periodic boundary conditions for gradient computations etc.
    vbc_m[i] = new PeriodicFace<Vector_t, Dim, Mesh_t, Center_t>(i);
    bcp_m[i] = new PeriodicFace<double, Dim, Mesh_t, Center_t>(i);
  }
}
rho_m.initialize(getMesh(), getFieldLayout(), GuardCellSizes<Dim>(1),bc_m);
phi_m.initialize(getMesh(), getFieldLayout(), GuardCellSizes<Dim>(1),bcp_m);
eg_m.initialize(getMesh(), getFieldLayout(), GuardCellSizes<Dim>(1), vbc_m);
```

Listing 3.1: Application of boundary conditions.

are not overlapping but offset to each other. This additional functionality for interpolation boundary conditions has been implemented in the IPPL framework for the serial as well as for the parallel case. A new interpolation boundary condition class is provided for the serial as well as for the parallel case. The previously described new behavior at the domain boundaries are handled analogously to the standard periodic boundary conditions. The application of the two different types of periodic boundary conditions are shown in listing 3.1.

3.2 Particle-Particle Solver

Following an object oriented code design, IPPL implements a `Hashpairbuilder` class which provides neighbor lists of close particles for every particle in the system. In analogy to the `Hashpairbuilder` class, we implemented a new periodic `HashpairbuilderPeriodic` class in order to introduce periodic boundaries in the particle-particle solver. The particles are bin sorted into the chaining mesh $\mathcal{M}_{CH}$ based on their spatial position similar to the non periodic approach. In order to compute the direct sum of particle-particle interactions, we traverse the chaining mesh cell by cell and consider only particles in neighboring cells as potential candidates which are within the cutoff r_c. Since particle-particle forces are symmetric, only one half of the chaining mesh has to be traversed if the forces are updated symmetrically. If a neighboring chaining cell under consideration lays outside the computational domain its periodic image is considered to return all the particles which are potentially within the interaction radius. If a periodic chaining cell is considered, the periodic new `Hashpairbuilder` class returns a non-zero shift factor which is applied to the particle position of all particles in the periodic neighboring cell to retrieve the physical coordinate

of the particle for radius condition checks and force calculation.

3.3 Periodicity of the Lattice Green's Function

In order to combine the particle-particle calculations and the particle mesh calculations based on the previously introduced interaction splitting, we split the real space Green's function $G(r)$ into a particle mesh part G_{PM} responsible for long range interactions and a particle particle part G_{PP} computing the short range interactions:

$$G(r) = \frac{1}{r} = \underbrace{\frac{1 - erf(\alpha r)}{r}}_{G_{PP}} + \underbrace{\frac{erf(\alpha r)}{r}}_{G_{PM}}, \quad \vec{k} \in \mathbb{R}^3 \,. \tag{3.3}$$

The long range interaction function G_{PM} has to be Fourier transformed analytically or numerically. The Fourier transform $\hat{G}_{PM}$ reads

$$\hat{G}_{PM}(\vec{k}) = \int G_{PM}(\vec{x}) e^{-i\vec{k}\vec{x}} d^3\vec{x} = \frac{4\pi}{k^2} e^{\frac{k^2}{4\alpha^2}} \,. \tag{3.4}$$

In the limit $\alpha \to \infty$ we obtain $\hat{G}_{PM} = \frac{4\pi}{k^2}$ which corresponds to the Fourier transform of the original Coulomb interaction term $\frac{1}{\vec{r}}$ and for $\alpha \to 0$ the long range interaction term vanishes: $\hat{G}_{PM} = 0$. The discrete Fourier transform is periodic in N and hence $\hat{G}_{PM}(\vec{k}) = \hat{G}_{PM}(\vec{k} + \vec{i}N)$. Equivalently, one can also compute the periodic Green's function in real space and use the fast Fourier transform to obtain the reciprocal lattice Green's function. The equivalence of the two approaches has been outlined in section 2.3.
If the lattice Green's function, a function of the distance between particles, is computed in real space additional care has to be taken in order to apply the periodic boundary conditions correctly. As shown in section 2.1 we need to compute the convolution

$$\Phi(\vec{x}) = \int G(\vec{x} - \vec{y}')\rho(\vec{y}')\mathrm{d}\vec{y}', \tag{3.5}$$

where the constant k_e has been neglected in order to improve readability. In order to understand the periodicity of the real space Green's function we look at a two dimensional problem, discretized on a $M \times N$grid with indices i and j. The discretized form of equation (3.5) can be written as

$$\Phi_{i,j} = \sum_m \sum_n G_{i-m,j-n}\rho_{m,n}, \tag{3.6}$$

with $i, m \in 0, 1, ..., M-1$ and $j, n \in 0, 1, ..., N-1$. In other words $G_{i-m,j-n}$ is a function of the distance r between the grid points (i, j) and (m, n). If $i = m$ and $j = n$ the distance between the corresponding two grid points is $r = 0$

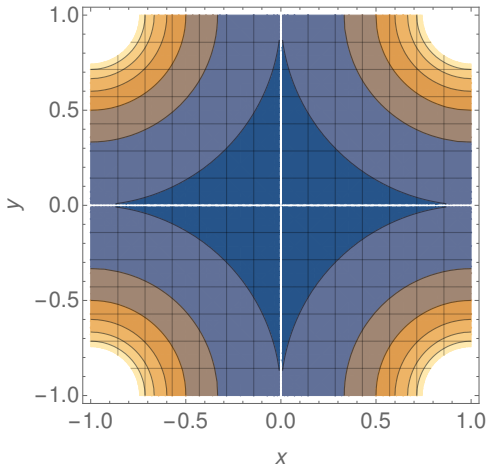

Figure 3.2: Sketch of two dimensional symmetric Green's function on computational domain $\Omega = [-1,1] \times [1,1]$. White color corresponds to high values and dark blue color represents low values of $G(r)$.

and hence $G_{0,0} = G(0)$. Due to the periodicity of the computational domain we find $G_{0,0} = G_{0,N-1} = G_{M-1,0} = G_{M-1,N-1} = G(0)$. Keeping this in mind we find that the two dimensional discrete Green's function is point symmetric around the center of the computational domain with singularities on all four corners of the computational domain. The two dimensional Green's function is visualized in figure 3.2, where the white color corresponds to high values of the Green's function and blue color represents low values. The three dimensional case follows analogously.

Chapter 4

Parallelization

The present code is parallelized for distributed memory machines using the message passing interface (MPI). We follow the for N-body algorithms common strategy of assigning the particles to processing unites (PUs) according to their spatial position in three dimensional configuration space. This increases the amount of data locality per PU. The key points of the applied parallelization strategies are outlined in this chapter.

4.1 Domain Decomposition

Each process is assigned to a part of the computational domain. Every process owns the corresponding parts of the grids for charge density and Green's function. Based on the same spatial decomposition, each process owns particles local to its corresponding domain. The domain decomposition is handled by the Ippl framework and the application can select which of the spatial dimensions are parallelized.

4.2 Parallel Initialization

Particles can either be created locally on one (the main) process and distributed over all processes according to the particle positions or each process can create its own particles. For the first regression tests, we adhere to a single process creating the particles.

4.3 Parallel Particle-Mesh

The parallelization of the particle-mesh (PM) part using the parallel distribution explained in section 4.1 needs special treatment of the periodic boundary conditions. Ghost cells have to be sent to the correct neighboring process under consideration of periodicity. All the communication is handled in an

update procedure, which is explicitly called after every change that affects particle positions or ghost cell updates. The workhorse of the particle-mesh part is the FFT solver for Poisson's equation. The involved Fourier transformations are parallelized over all processes using the existing Ippl implementation. After ghost cell exchange, every process computes the gradient of the electric potential locally. Scatter and gather procedures to and from the particles are also performed locally with the appropriate boundary conditions as discussed in section 3.1.1.

4.4 Parallel Particle-Particle

The parallel particle-particle (PP) implementation is a bit more sophisticated. As described in section 2.2, we follow the chaining cell approach to handle particle-particle interactions. Our implementation allows arbitrary choices of process grids and interaction radii which influence the chaining cells. However, for the parallel periodic case with the current implementation we are limited to domain decompositions which result in only one neighbor domain per dimension. A layout of three processes with two parallel dimensions for example is not supported. The overall layout of domain decomposition and chaining mesh splitting is shown in figure 4.1. Notice that this figure only shows the chaining mesh as introduced in section 2.2 and not the computational mesh for the PM part of the solver. The main purpose of the chaining mesh is to sort the particles into buckets with edge length $h_{CH} > r_{cut}$ such that for the PP computations only particles within neighboring buckets have to be considered. Prior to the inter particle force calculations, the processes exchange their boundary particles which are within distance r_{cut} from the local boundary with the neighboring processes. These particles are from now on called ghost particles. The range of ghost particles received by process 0 is highlighted in yellow in Figure 4.1. Each process then calculates the unique indeces for the particle bins in the extended local chaining mesh, such that every particle (ghost or real) belongs to a single chaining cell. For this purpose, every process stores its own part of the chaining mesh limited by its local domain boundaries plus the ghost layer. The mesh width of the local chaining meshes is chosen as the closest real value greater or equal to r_{cut} such that the chaining mesh boundaries fall on local domain boundaries. Similar to the serial case we perform a symmetric update of particle forces in order to avoid double computation of the same force. Every process loops through its local chaining cells and increments the forces on each particle within the symmetric cell update stencil. By construction of the chaining mesh only particles within neighboring chaining cells have to be considered. The red cross in the stencil indicates the active chaining cell for which the force updates are computed. All the updates affecting ghost particles are discarded. This means that for example process

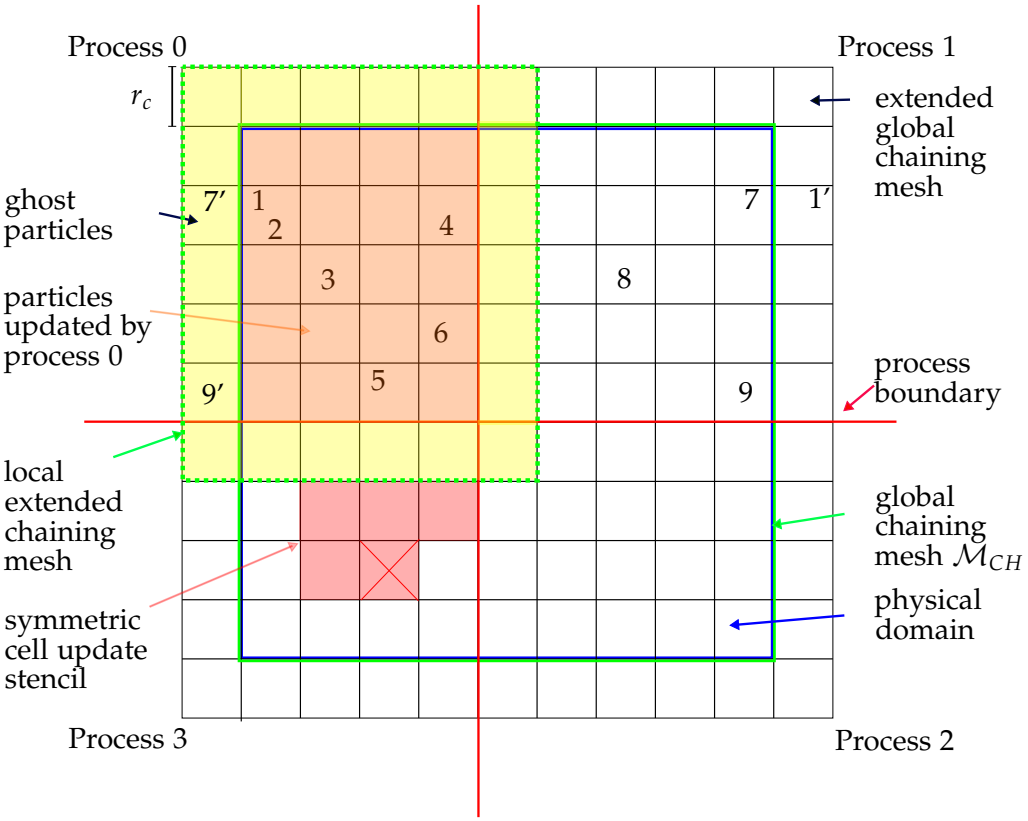

Figure 4.1: Parallel particle-particle chaining mesh. Some example particles are visualized by algebraic numbers. The corresponding ghost particles are denoted by the same number with a superscript prime.

zero computes the force between particle 7 and 1 but only updates particle 1. Process 1 computes the same force but updates only particle 7. In other words we give up the symmetric updates for particles close to the boundaries.

Remark: In the case of a serial dimension no ghost particle exchange is performed in this dimension, since the affected particles are already local to the considered process. This means that periodicity has to be applied manually. Instead of copying the particles to create ghost particles with shifted coordinates, the boundary cells are mapped to their periodically corresponding cell and the particle positions are shifted by an offset. This is how periodic particle updates are done in the serial computations.

Chapter 5

Regression Tests

In this chapter five independent experiments with known analytic solutions are presented to show the correctness of the P^{3}M code.

5.1 Serial P^{3}M

In this section the results obtained by serial execution of the P^{3}M code are presented. The serial results will be used as a reference for the parallel tests shown in section 5.2.

5.1.1 Static Uniformly Charged Sphere

The sphere tests are performed with 20×10^3 particles. We consider a uniformly charged sphere of radius $R = 1$ and total charge $Q = 1$ in arbitrary units centered at the origin in a computational domain $\Omega = [-4, 4]^3$ with periodic boundary conditions. Figure 5.1 shows the electric field obtained by a high resolution PM simulation with 256^3 grid points. This agrees very well with the analytic solution for the electric field

$$E(r) = \begin{cases} \frac{Q}{4\pi\varepsilon_0 r^2} & r > R \\ \frac{Qr}{4\pi\varepsilon_0 R^3} & r \leq R \end{cases} \tag{5.1}$$

as shown in Figure 5.2. The analytic solution corresponds to an isolated sphere in an infinite domain. In the periodic case the electric field vanishes on the boundary of the domain due to the symmetry in charge distribution. Taking this into account, the simulation results agree with the analytic solution.

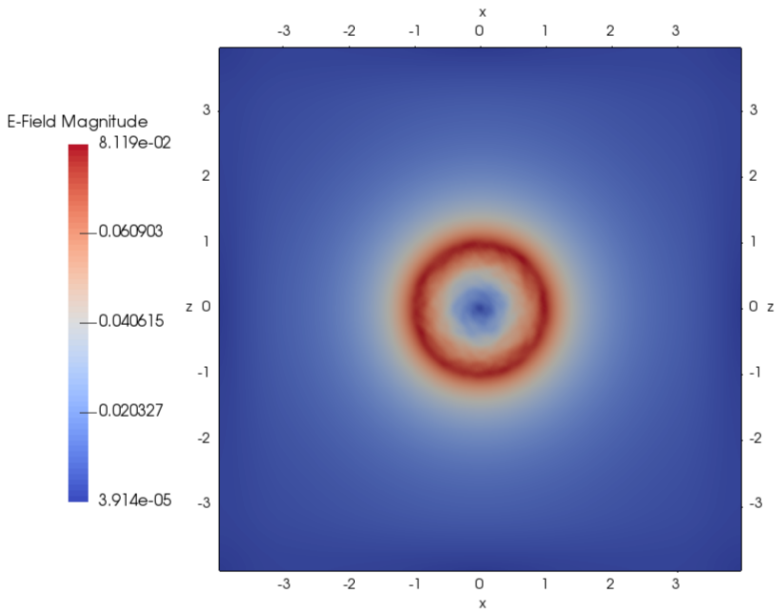

Figure 5.1: xz plane of electric field on a 256^3 grid in arbitrary units.

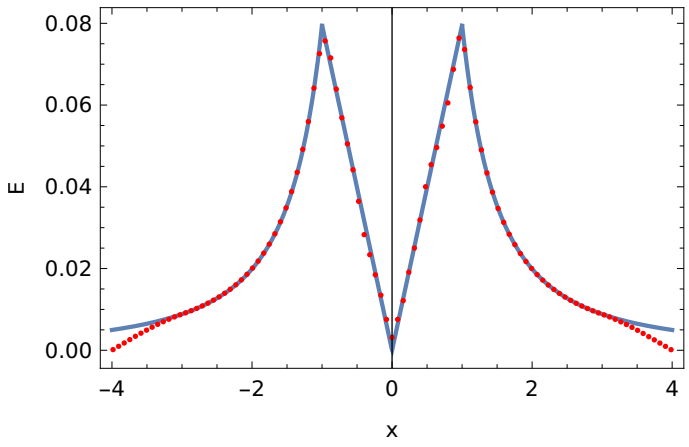

Figure 5.2: Blue line represents analytic solution for the electric field on the $x-$axis of the computational domain, crossing the sphere center. Red dots indicate the numerical values obtained on a 256^3 grid with the particle mesh solver. The results are given in arbitrary units.

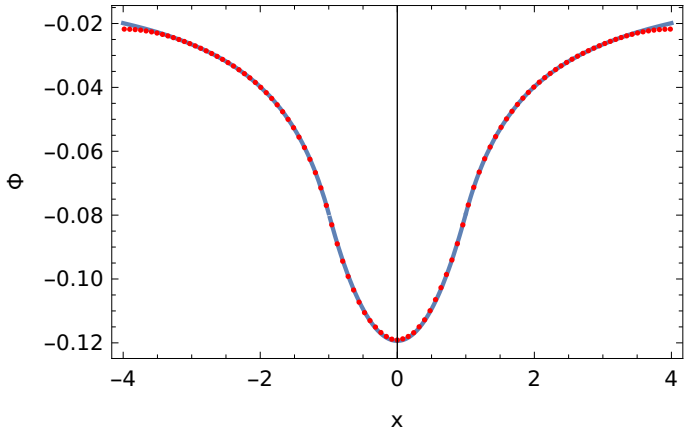

Figure 5.3: Blue line represents analytic solution for the electric potential on the $x-$axis of the computational domain, crossing the sphere center. Red dots indicate the numerical values obtained on a 256^3 grid with the particle mesh solver. The results are given in arbitrary units.

The potential obtained by the simulation shown in figure 5.3 agrees with the analytic solution

$$\Phi(r) = \begin{cases} \frac{-Q}{4\pi\varepsilon_0 r} & r > R \\ \frac{-Q}{2R\cdot 4\pi\varepsilon_0}\left(1 - \frac{r^2}{R^2}\right) & r \leq R \end{cases} \tag{5.2}$$

up to the expected potential shift close to the domain boundary due to the periodic boundary conditions.

The Electrostatic Energy. In a system of charges the electrostatic energy is given by

$$E = \frac{1}{2}\varepsilon_0 \int |\vec{E}|^2 \mathrm{d}V = \frac{1}{2}\int \rho\Phi \mathrm{d}V \tag{5.3}$$

where ε_0 is the permittivity of free space (chosen to be 1 for this test), $\vec{E}$ is the electric field vector, ρ is the charge density and Φ the electrostatic potential. For our uniformly charged sphere we find for the finite volume of $V = \Omega = [-4, 4]^3$ an electrostatic energy of $E_\Omega = 0.0394785$ and for an infinite domain $E_\infty = 0.0477465$. If we compute the electrostatic energy obtained by our numerical simulation by summing over $|\vec{E}|^2$ we obtain the solution of the finite domain (simulation: $E = 0.03827$ analytic: $E_\Omega = 0.03947$). However, if we sum over $\Phi\rho$ the numerical solution answers to E_∞ (simulation: $E = 0.04763$ analytic: $E_\infty = 0.04774$). This can be explained by the fact that $\rho = 0$ outside the charged sphere in our simulation and the influence of the finite simulation domain on the potential field within the sphere is negligible. To agree with the analytic solution for the finite domain we have to subtract

a uniform background charge $Q_{bg} = -Q$ equal to the total charge of the sphere. This results in a shift of the potential and overcomes the issue of $\rho = 0$ outside the sphere. For the simulations performed in this thesis, we are mainly interested in the electric field which is independent of the background charge because it is the gradient of the potential. Hence, the issue of background charge will not be discussed in detail here.

Dependence on Interaction Splitting Parameter α. The following results show the influence of the short range interactions to the previously introduced static sphere experiment. The influence of the PP part of the solver is controlled by the splitting parameter $\alpha \in [0, 20]$ and cutoff r_{cut}. Studied are the percentage mean square error in the electric field

$$\varepsilon_{rms} = \frac{\sqrt{\sum (E_{num} - E_{exact})^2}}{\sqrt{\sum (E_{exact})^2}} \tag{5.4}$$

and the percentage error in the total electric field

$$\varepsilon_{tot} = \frac{|\sum E_{num} - \sum E_{exact}|}{|\sum E_{exact}|}. \tag{5.5}$$

The summations in formulae (5.4) and (5.4) are summations over the computed quantities at the particle positions itself. The results for ε_{rms} are presented in figure 5.4 and the results for ε_{tot} are shown in figure 5.5. One has to keep in mind, that the reference solution is the analytic solution for an *isolated sphere*. The shape of the error functions qualitatively matches the results obtained in [17] for a uniform distribution in a cubical domain very well.

One can see that for a given cutoff and mesh size the optimal value for the parameter α has to be determined individually. For $\alpha \to 0$ the results converge to a pure PP simulation and for $\alpha \to \infty$ the results converge to a PM only simulation. The larger the cutoff r_c is chosen, the more accurate are the PP results. The more mesh points used, the more accurate are the PM results.

5.1.2 Moving Sphere

In order to verify the correctness of the periodic boundary condition implementation the following test was performed. A sphere of 20000 particles and charge $Q = 1$ is centered at the origin. Every particle gets constant velocity $\vec{v} = (1, 1, 1)$. The charges are moving with this constant speed and a timestep of $dt = 0.5$. This enforces a worst-case scenario for the periodic boundary conditions as the sphere leaves and reenters the computational domain over a corner. Table 5.1 shows the deviation of electric field, potential

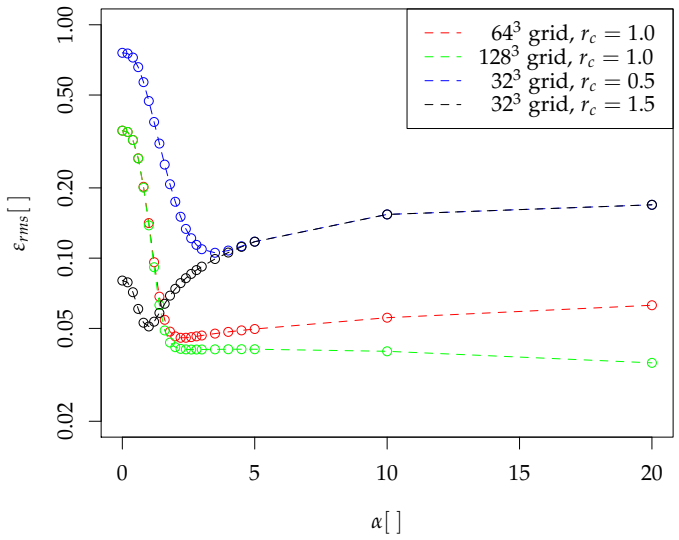

Figure 5.4: Percentage root mean squared error in electric field for a uniformly charged sphere with total charge $Q = 1$ and radius $R = 1$.

and charge stored in the grid and at the particles over 40 time steps. This means that the sphere of charge is crossing a corner of the periodic domain two times. The interaction splitting parameter α is chosen to be $\alpha = 1$ and the interaction radius is set to $r_{cut} = 0.75$ this ensures that particle-mesh computations as well as particle-particle interactions influence the tracked quantities. The computational grid for the PM part is chosen to be 64^3. All measured quantities are conserved as shown in table 5.1. This also holds for the values obtained with a parallel simulation with up to 32 MPI threads.

statistics	$\sum \lvert\vec{E}\rvert_{particle}$	$\sum \Phi_{particle}$	$\sum \rho_{grid}$	$\sum \vec{E}_{grid}$	$\sum \Phi_{grid}$
μ (64^3 grid)	40.56	-1759.69	-512.0	17.88	-6077.52
σ (serial)	2.17×10^{-7}	5.61×10^{-6}	0.0	1.65×10^{-7}	0.0
σ (2 threads)	2.16×10^{-7}	5.56×10^{-6}	0.0	1.64×10^{-7}	0.0
σ (4 threads)	2.17×10^{-7}	5.62×10^{-6}	0.0	1.66×10^{-7}	0.0
μ (128^3 grid)	41.04	-1763.05	-4096.0	143.93	-48624.79
σ (32 threads)	2.84×10^{-7}	3.07×10^{-6}	0.0	4.64×10^{-7}	0.0

Table 5.1: Mean μ and standard deviation σ of conserved quantities in arbitrary units for a moving charged sphere over 40 time steps.

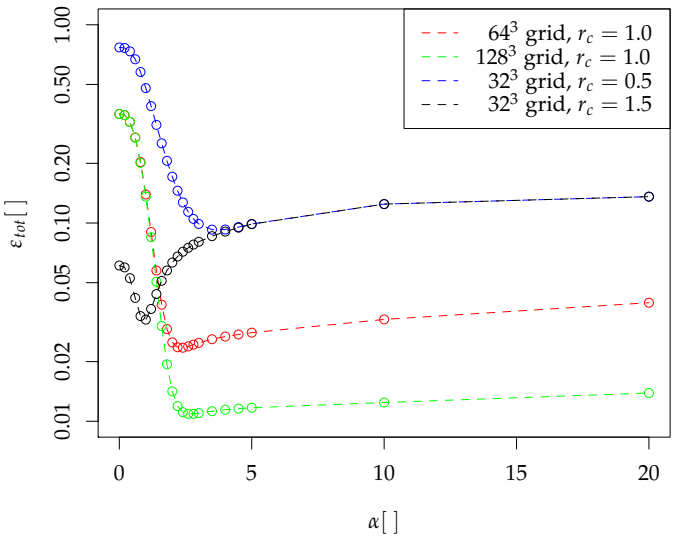

Figure 5.5: Percentage total error in electric field.

5.1.3 Summary of Uniformly Charged Sphere Results

The numerical experiments performed on the uniformly charged sphere show that our code provides us with the correct electric and potential field. Conservation of charge, potential field and electric field is guaranteed for particles crossing the periodic domain boundaries. The accuracy of the P^{3}M method is related to the splitting parameter α as expected from the design of the algorithm based on Gaussian charge screening.

5.1.4 Recurrence Time

In order to study the dynamic behavior of the three dimensional code, the free stream experiment of [20] has been extended to three dimensions. Studied is a particle beam with Gaussian velocity distribution in all three dimensions and a uniform spatial distribution disturbed by a cosine term. The initial three dimensional distribution function for the particle positions $\vec{x}$ and velocities $\vec{v}$ reads

$$f(\vec{x},\vec{v},0) = 0.1\frac{1}{(2\pi)^{3/2}}e^{-|v|^2/2}\sum_i(\cos(kx_i)) \; . \tag{5.6}$$

With this Gaussian velocity distribution and cosine perturbation in the spatial coordinate the particle density $\rho(\vec{x},t)$ is a periodic function in time with

a theoretical recurrence time of

$$T_R = \frac{2\pi}{k\Delta v} = 33.5\ . \tag{5.7}$$

This can be easily seen by integration of equation (5.6) over the velocity coordinate [20, section 4]. In equation (5.8) the periodic density formula for the x coordinate is derived. The y and z direction follow analogously.

$$\begin{aligned}\rho(x,t) &= \sum_{j=-N_{v_x}/2}^{N_{v_x}/2} f(x - v_{x_j}t, v_{x_j}, 0)\Delta v_x \\ &= 0.1\frac{1}{(2\pi)^{3/2}} \sum_{j=-N_{v_x}/2}^{N_{v_x}/2} e^{-v_{x_j}^2/2} \cos\left(kx - \left(j + \frac{1}{2}\right)\Delta v_x t\right)\ .\end{aligned} \tag{5.8}$$

The particles are distributed on a six dimensional phase space grid with $N_x = N_y = N_z = 8$, $N_{v_x} = N_{v_y} = N_{v_z} = 32$ and $k = 0.5$ which leads to $\Delta v = \frac{2V_{max}}{N_v} = \frac{1}{32}$ being the discrete velocity spacing corresponding to a velocity distribution within $v_{x,y,z} \in [-5.0, 5.0]$ in all three dimensions. The computational grid in configuration space was chosen to be 8^3. Figure 5.6 shows the maximum value of the charge density in the computational grid obtained by a numerical PM simulation ($\alpha = \infty$). The recurrence time fits the theoretical value very well. Figure 5.7 shows a similar behavior for the recurrence time of the electric field. This gives evidence that the time integration and periodic boundary conditions are implemented correctly.

5.1.5 Landau Damping

The study of Landau damping is an often performed experiment in plasma simulation. Due to the relatively high resolution required, most of the experiments found in literature are restricted to two spatial dimensions, but here we show our attempt to extend the Landau damping experiment to three dimensions using the initial distribution function

$$f(\vec{x}, \vec{v}, 0) = \frac{1}{(2\pi)^{3/2}} e^{\left(-\frac{|v|^2}{2}\right)} \left(1 + \alpha \sum_{l=1}^{3} \cos(kx_l)\right) \tag{5.9}$$

as suggested in [21], where a semi-Lagrangian method based on a phase space grid is used to solve the Vlasov-Poisson system. We set $\alpha = 0.05, k = 0.5$ and $\Delta t = 0.25$. The initial particle deposition uses a 6D phase space grid with $Nx = 8$ and $Nv = 32$. The computational grid in configuration space is chosen to be 8^3. The theoretical damping rate for the electric energy obtained from linear theory [22] is -0.3066 and the maximum of the electric field is expected to be damped with rate -0.1533. The experimental results for the field electric energy obtained by a PM only simulation are

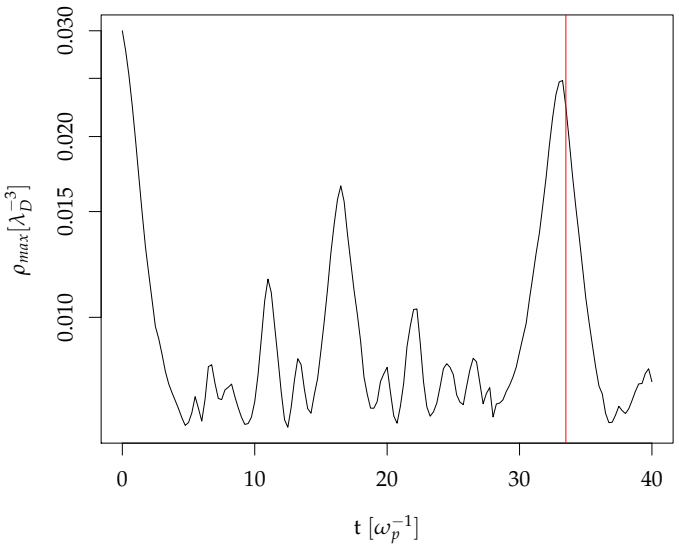

Figure 5.6: Maximum charge density for PM simulation on 8^3 grid. Analytic recurrence time $T_R = 33.5$ showed as vertical red line. The units are the same as in [20].

shown in figure 5.8. The analytic damping rate is depicted with a red line. The experimental results indicate a damping for the first time steps close to the analytical prediction however, we observe severe noise in the numerical solution. One has to take into account that the number of particles in our simulation is much lower than the tensor grid with 32^3 grid points for the spatial grid and 128^3 grid points for the velocity grid used in [21].

A major source of error was found to be the truncation of the Green's function at $r = 0$. The simulation results shown in Figure 5.8 are based on a Greens function which sets the value at the origin to the next neighbor value on the grid. Since we are using a very coarse computational grid (meshwidth ≈ 1.5) to be consistent with the literature [23, 24] this approximation has a significant impact. Better results are obtained for keeping $G(0,0,0) = \frac{1}{r+\varepsilon}$ with a softening parameter $\varepsilon = 1 \times 10^{-10}$. The simulation results for the damping rate of the electric field are shown in figure 5.9.

5.1.6 Twostream Instability

Twostream instability is another popular regression test performed in plasma simulations. Similar to the Landau damping simulations very high resolution is required and hence the PIC codes found in literature are restricted

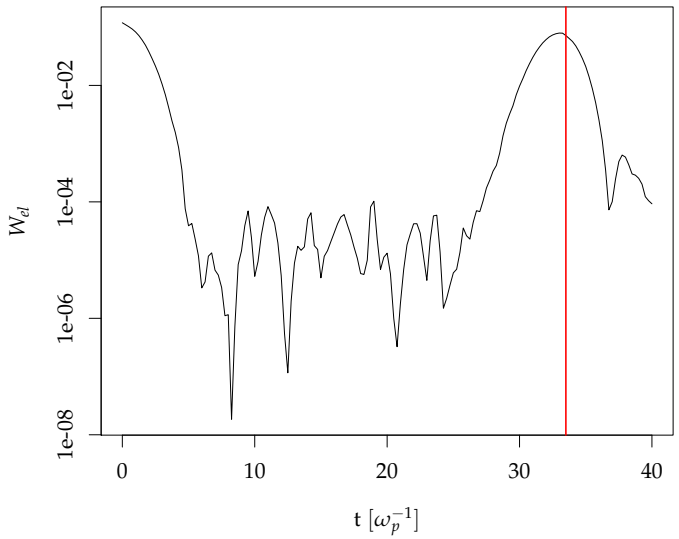

Figure 5.7: Field energy over time for PM simulation on 8^3 grid. Analytic recurrence time $T_R = 33.5$ showed as vertical red line.The units are the same as in [20].

to two dimensions [23, 24]. The simulations presented here are performed in three dimensions. The three dimensional particle distribution function reads

$$f(\vec{x},\vec{v},0) = \frac{1}{(30\pi)} e^{\left(-\frac{|v|^2}{2}\right)} (1+\alpha\cos(kz)) \cdot (1+\frac{1}{2}v_z^2)$$
$$(\vec{x},\vec{v}) \in [0,4\pi] \times [-6,6] \tag{5.10}$$

and was evaluated on the 6d phase space grid $N_{\vec{x}} = [4,4,32]$, $N_{\vec{v}} = [8,8,128]$. With this initial distribution function we simulate to counter streaming beams in $z-$direction carrying the same amount of charge. The initial separation of the two species in $z - v_z$ phase space is expected to vanish by forming a vortex in phase space.
The time step was chosen to be $dt = 1/8$ and the computational grid was 16^3. The amplitude of the initial disturbance was $\alpha = 0.01$. Figure 5.10 shows the expected formation of a vortex at $t = 20$. The coarse resolution and artifacts are again based on the coarse computational grid.

5.1.7 Summary Plasma Tests

The in this chapter performed Landau damping and Twostream instability simulations indicate qualitatively the correctness of our PM solver and time

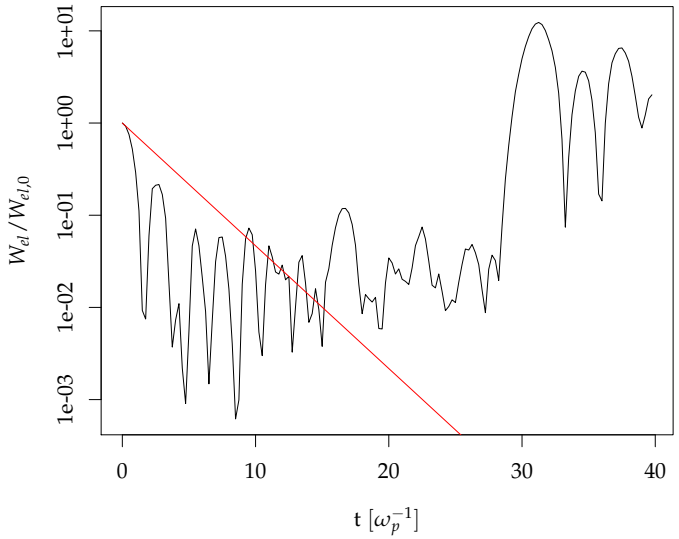

Figure 5.8: Field energy normalized to the initial field energy over time. The red line shows the analytic damping rate of -0.3066.

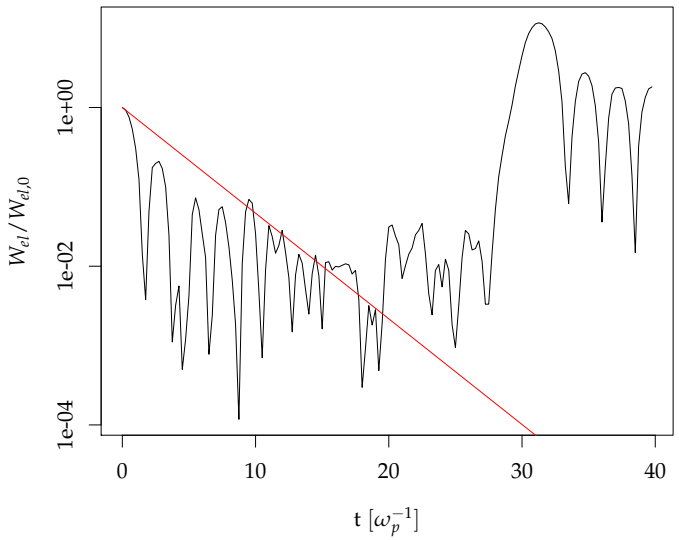

Figure 5.9: Field energy normalized to the initial field energy with softened Greens function. The red line shows the analytic damping rate of -0.3066.

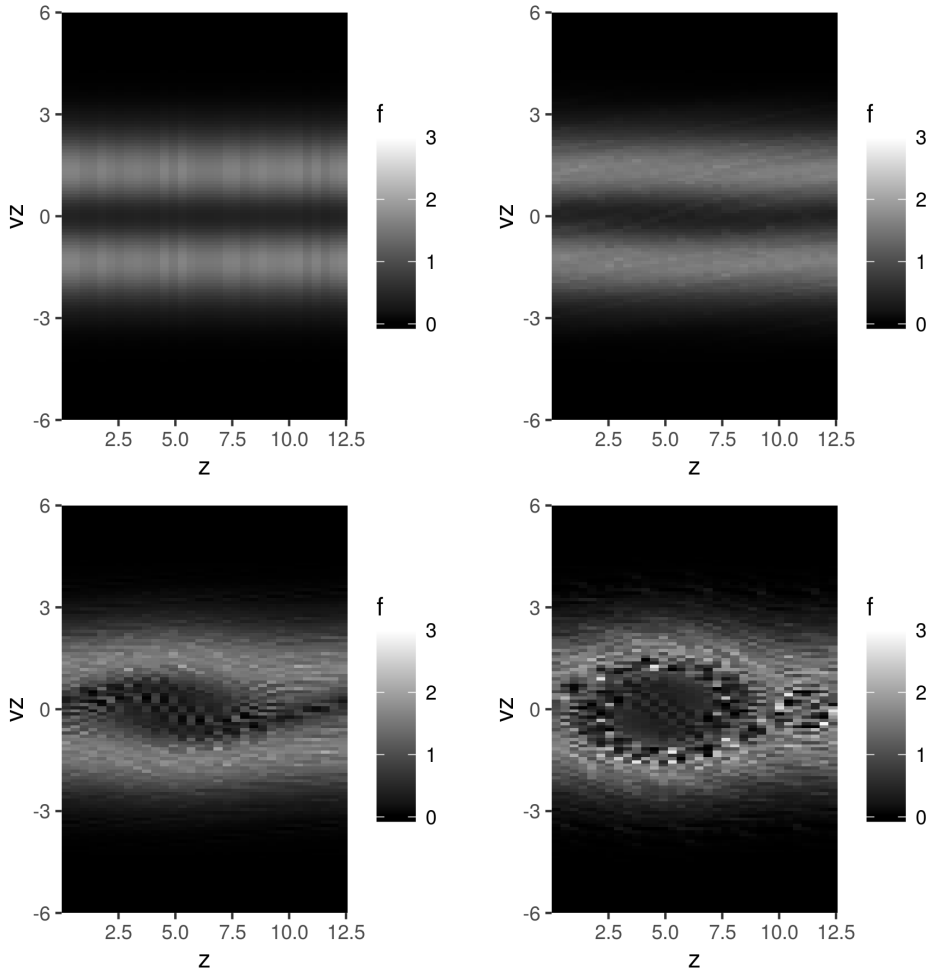

Figure 5.10: Two dimensional longitudinal phase space at times $t = 0, 10, 20, 30$ from left to right, top to bottom. The units are the same as in [24]

integrator. However, we observed that the particle distribution on a fine grained phase space grid as commonly used for this kind of simulations brings us quickly to the computational limit for simulations in three dimensions. It is not the resolution of the computational grid but the huge number of particles, quickly leading to memory constraints and large execution times which are not appreciated for the purpose of simple regression tests. Hence the study of Landau damping and two stream instability was concluded at this point and the focus is set towards the electron beam problems which are supposed to be the core application of this thesis.

S	r_{cut}	1	2	3	4	5	6	7	8
$(0,0,0)$	2	$75.5221(-1903.58)$	x	x	x	x	x	x	x
$(3.5,3.6,3.7)$	2	$75.5221(-1903.58)$	x	o	x	o	o	o	x
$(0,0,0)$	0.6	$8.74116(-600.978)$	x	x	x	x	x	x	x
$(3.5,3.6,3.7)$	0.6	$8.74116(-600.978)$	x	-	x	-	-	-	x

Table 5.2: Sum of electric field (potential) felt by the particles obtained by isolated particle-particle solver. The table shows the results for the center of the sphere S, the used cutoff radius r_{cut} and 1 to 8 MPI threads. "x" denotes the expected reference value was exactly hit, "o" denotes a deviation in the result and "-" denotes that the experiment has not been performed because the outcome is irrelevant.

5.2 Parallel P^{3}M

In this section we study the correctness of our parallel implementation based on the experiments already presented for the serial regression testing.

5.2.1 Parallel Field Solver for a Static Sphere

Particle-particle and particle-mesh parts of the field solver are evaluated separately. We evaluate the sum of the electric field magnitude and the potential energies felt by the particles for a uniformly charged unit sphere centered at the origin as a reference solution and a sphere centered at $S = (3.5, 3.6, 3.7)$. The computational domain is $[-4, 4]^3$. The results in Table 5.2 show that the PP code gives correct results for 1 to 8 processes in the centered sphere case. These 8 cases allow the study of the influence of different domain decompositions. Since, all three dimensions are parallelized, rather pathologic decompositions can occur such as a decomposition of two dimensions over three processes. If the sphere is placed over the periodic boundaries, correct results are only obtained for "simple" process grids. In the case of $3, 5, 6, 7$ processes over 3 parallel dimensions some local domain boundary faces cover the boundary faces of two other local domains. In this situation, the implemented particle caching omits some ghost particles on the corner of the overlapping domains; leading to incorrect results. For now we will restrict ourselves to the "simple" process grids, meaning that each process is allowed to have only one neighboring process per dimension.

For the particle-mesh part the results are correct for all tested scenarios and shown in table 5.3. It is worth to mentioned, that the fine grained grid gives almost the same results for potential and electric field per particle as the large cutoff PP simulation. This is another indication, that our P^{3}M gives correct results.

S	grid size	1	2	3	4	5	6	7	8
$(0,0,0)$	256^3	$74.934(-1904.51)$	-	-	-	-	-	-	-
$(3.5,3.6,3.7)$	256^3	$74.9336(-1904.51)$	-	-	-	-	-	-	-
$(0,0,0)$	64^3	$68.0559(-1878.78)$	x	x	x	x	x	x	x
$(3.5,3.6,3.7)$	64^3	$68.0594(-1878.79)$	x	x	x	x	x	x	x

Table 5.3: Sum of electric field (potential) felt by the particles obtained by isolated particle-mesh solver. The table shows the results for the center of the sphere S, the used grid size and 1 to 8 MPI threads. "x" denotes the expected reference value was exactly hit and "-" denotes that the experiment has not been performed.

5.2.2 Moving Sphere

Periodic boundary conditions are a critical part of the parallel implementation. To verify the correctness we performed the moving sphere experiment as described in section 5.1.2 with two MPI processes. The results can be found in Table 5.1 of section 5.1.2 and are in agreement with the serial simulation.

5.2.3 Dependence on the Interaction Splitting Parameter α

Since the standalone particle-mesh and particle-particle solvers have been tested extensively we only performed one sample test for the α dependence experiment presented in section 5.1.1. Figure 5.11 shows the results for the root mean squared error in the electric field obtained by the serial implementation augmented by simulation results using 2 and 32 MPI processes. The serial and parallel results are in perfect agreement.

5.2.4 Summary Parallel Tests

The parallel tests presented in this section make use of up to 32 processes, corresponding to a full node on the used supercomputer, and are considered sufficient to move on to the real particle beam problems we are interested to simulate. If any doubts about the correctness or scaling of the parallel implementation will appear during these simulations one can always come back to the regression tests and obtain results for more processes involved.

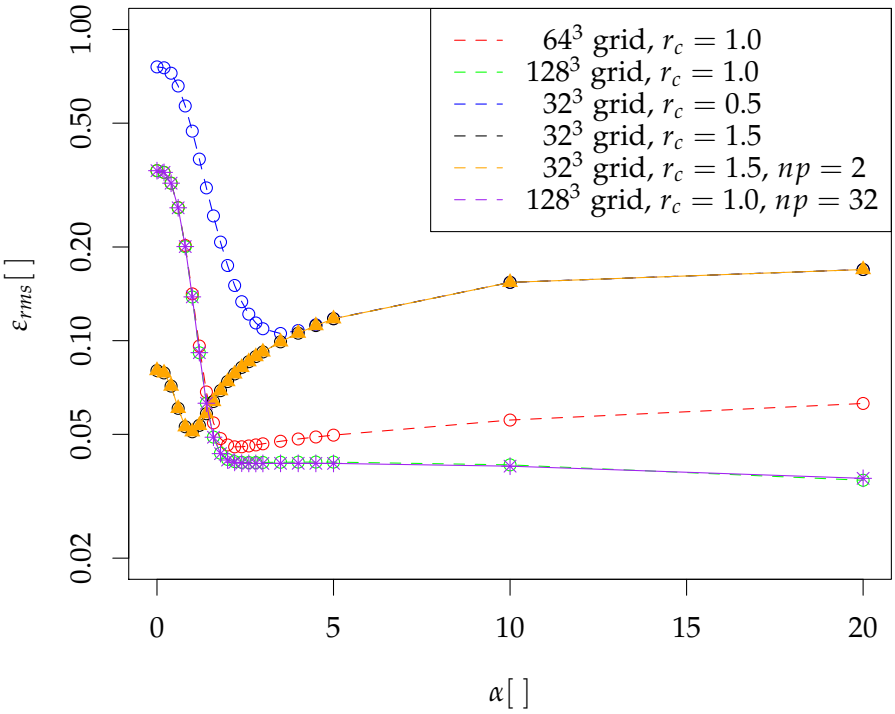

Figure 5.11: Percentage rms error in electric field. The orange triangles are obtained with a parallel simulation using 2 MPI processes and the results depicted in purple are obtained by a simulation using 32 MPI processes.

Chapter 6

Space-Charge Induced Optical Microbunching

With microbunching, the formation of micro structures in a relativstic electron beam is denoted. This in general undesired effect has the potential to degrade the beam quality. An understanding of the origin of the microbunching instability is therefore necessary to control this effect. This understanding will help to reduce any unwanted side effects or allow the use of it e.g. as an amplifier in free-electron lasers. Longitudinal space charge is a possible source for microbunching growth at moderate energies. Marinelli and Rosenzweig [8] derived an analaytic formula for the microbunching gain in a moderate energy beam. This chapter will sketch the derivation of the microbunching gain formula to provide a solid theoretical background to compare the numerical results with. The detailed derivations can be found in [8]. In this chapter the focus is on the approximations and assumptions made during the derivation in order to put our numerical results into context.

6.1 The Bunching Factor

The bunching factor $b(k, \theta, \varphi)$ with wave number $k = \frac{2\pi}{\lambda}$ is the three dimensional Fourier transform of the electron density. It is a number that quantifies the formation of micro structures at a wave length λ for given polar and azimuthal angles θ and φ with respect to the beam propagation axis z

$$b(\theta, \varphi, k) = \frac{1}{N} \sum_{n=1}^{N} e^{-ik[z_n + \sin\theta(x_n \cos\varphi + y_n \sin\varphi)]} \,. \tag{6.1}$$

In the numerical simulation and the further analysis we will assume a symmetric beam in transverse directions x and y which corresponds to $\varphi = 0$. For small angles θ we approximate $\sin(\theta) \approx \theta$ and the formula for the bunch-

ing factor becomes

$$b(\theta, k) = \frac{1}{N} \sum_{n=1}^{N} e^{ik(z_n + \theta x_n)}. \tag{6.2}$$

High Frequency Limit In the analysis performed in [8] only the high frequency regime $\sigma_x \gg \gamma\lambda/2\pi$ is considered. If the transverse beam size σ_x is much larger than the wavelength of interest λ scaled by the relativistic factor γ, space charge effects play a major role and the collective physics can be treated in analogy to an unbounded uniform plasma. In this case a fully three dimensional treatment becomes necessary.

6.1.1 Longitudinal Space-Charge Induced Microbunching

In [25] the evolution of micobunching is described as follows:

1. Point of departure is a random distribution of particles with noise e.g. from the particle gun.
2. The induced longitudinal space charge fields will cause *energy modulations*.
3. Under the influence of a *dispersive element* with non-zero momentum compaction, this energy modulation gets converted to density modulation which results in longitudinal position changes.
4. The longitudinal density distribution shows enhancement at some frequency range. We call this effect mircobunching at these frequencies.

Vlasov-Poisson Equation. The following analysis is based on the Vlasov-Poisson equation which describes the kinetics of a warm plasma. In order to apply the Vlasov-Poisson theory to a beam dynamics problem the condition

$$\gamma\lambda \gg (\gamma/n_0)^{1/3} \tag{6.3}$$

has to be satisfied. Here, $\gamma\lambda$ is the wavelength of interest in the beam rest frame which is required to be much bigger than the mean interparticle distance in the beam rest frame $(\gamma/n_0)^{1/3}$ with particle density n_0. The Vlasov equation for zero magnetic field reads

$$\frac{\partial f_\alpha}{\partial t} + \vec{v}_\alpha \frac{\partial f_\alpha}{\partial \vec{x}} + \frac{q_\alpha \vec{E}}{m_\alpha} \frac{\partial f_\alpha}{\partial \vec{v}} = 0 \tag{6.4}$$

together with Poisson's equation

$$\nabla^2 \Phi + \rho = 0. \tag{6.5}$$

Here n_0 denotes the particle number density, γ is the relativistic factor and f_α is the continuous six dimensional distribution function of a species α with

particle mass m_α and velocity vector $\vec{v}_\alpha$. The electric field is denoted by $\vec{E}$. The electric potential is denoted by Φ and ρ is the charge density.

Remark on Collisions. The Vlasov-Poisson equation ignores binary collisions and hence does not account for particle-particle collisions. This issue is discussed in [8, sec. II] and will be taken into account for the evaluation of the numerical results in chapter 7.

Setup. The following analysis is based on an electron beam with particle distribution function

$$f(\vec{x}_\perp, z, \vec{\beta}_\perp, p, \tau) \,, \tag{6.6}$$

with relative energy deviation $p = \Delta\gamma/\gamma$ and $\tau = ct$, where c is the speed of light. The ratio of the transverse velocity to the speed of light is denoted by $\vec{\beta}_\perp$. The transverse and longitudinal positions of a particle are denoted by $\vec{x}_\perp$ and z.
The beam is undergoing an external force-free drift followed by a series of optical elements modeled by transformation matrix R_{ij}.
The distribution function can be expanded to first order in perturbation theory and can be written as

$$f = f_o + f_1 \,, \tag{6.7}$$

with $|f_1| \ll |f_0|$ and $f_0 = n_0 f_v(\vec{\beta}_\perp, p)$, where n_0 is the local average particle density and f_v is assumed to be isotropic in transverse velocity.

Derivation of the Bunching Factor b. The Vlasov-Poisson equation (6.4), (6.5) applied to the expanded distribution function reads

$$\frac{\partial f_1}{\partial \tau} + \vec{\beta}_\perp \cdot \vec{\Delta}_\perp f_1 + \frac{p}{\gamma^2}\frac{\partial f_1}{\partial z} + \frac{F_z}{\gamma m c^2} n_0 \frac{\partial f_v}{\partial p} + \frac{\vec{F}_\perp}{\gamma m c^2} n_0 \frac{\partial f_v}{\partial \vec{\beta}_\perp} = 0, \tag{6.8}$$

where the independence of $\tau, \vec{x}$ and z for f_0 was used and $\frac{dz}{d\tau} \approx p/\gamma^2$ was applied. The electric Field has been substituted by the electric force, given by $\vec{F} = e\vec{E}$, with electron charge e. Poisson's equation (6.5) can be written as

$$\left(\nabla_\perp^2 + \frac{1}{\gamma^2}\frac{\partial}{\partial z^2}\right)\Phi = \frac{e}{\gamma\varepsilon_0}\int f_1 dp d^2\beta \tag{6.9}$$

Equations (6.8), (6.9) can be solved in the Laplace-Fourier domain [8, sec. II]. The following notation is introduced. With a hat we denote the Fourier transform of a quantity and the Laplace transform respectively is denoted by a tilde. The consecutive application of a Fourier transform followed by a Laplace transform is denoted by $\tilde{\hat{(.)}}$. With this notation the Laplace-Fourier transform of the distribution function f and the forces $F_z, F_\perp$ can be written in Laplace-Fourier space as $\tilde{\hat{f}}, \tilde{\hat{F}}_z, \tilde{\hat{F}}_\perp$. Making use of equations (6.8) and (6.9)

we can write the first order term of the distribution function in Laplace-Fourier space as

$$\begin{aligned}\tilde{\hat{f}}_1 =& \frac{1}{s+ik(\theta\beta_x + p/\gamma^2}\left[\hat{f}_1\mid_{\tau=0} - \frac{1}{\varepsilon_p}\frac{\omega_p^2}{c^2[1+(\gamma\theta^2)]}\times\right.\\ &\left.\left(\frac{\partial f_v}{\partial p}+\theta\frac{\partial f_v}{\partial\beta x}\right)\frac{\gamma^2}{ik}\int\frac{\hat{f}_1\mid_{\tau=0}}{s+ik(\theta\beta_x'+p'/\gamma^2)}dp'd^2\vec{\beta}'\right],\end{aligned} \tag{6.10}$$

where ω_p^2 is the relativistic beam plasma frequency and ε_p is the plasma dielectric function

$$\varepsilon_p = 1+\frac{\omega_p^2}{c^2[1+(\gamma\theta)^2]}\frac{\gamma^2}{ik}\int\frac{\frac{\partial f_v}{\partial p}+\theta\frac{\partial f_v}{\partial\beta_x}}{s+ik\left(\theta\beta_x+\frac{p}{\gamma^2}\right)}\mathrm{d}p\mathrm{d}^2\vec{\beta}\,. \tag{6.11}$$

To obtain the bunching factor b which is defined as the *Fourier transform of the density perturbation* f_1 we have to compute the inverse Laplace transform of $\tilde{\hat{f}}_1$. Making use of Cauchys residual theorem this can be expressed as a sum over the residuals at the poles s_j which correspond to the roots of ε_p. The effect of longitudinal dispersion in the optical elements is modeled by the matrix elements R_{56} and yields to

$$\hat{f}_1 \to \hat{f}_1 e^{-ikpR_{56}}. \tag{6.12}$$

Taking all of this into consideration and representing the particle positions in terms of Dirac $\delta-$functions, one obtains for the bunching factor $b_{R_{56}}$ after the drift

$$\begin{aligned}b_{R_{56}} =& -\frac{1}{N}\sum_j e^{s_j L_d}\frac{1}{\frac{\partial\varepsilon_p}{\partial s}\mid_{s=s_j}\frac{\omega_p^2 R_{56}\gamma^2}{c^2[1+(\gamma\theta)^2}}\times\\ &\int\frac{e^{-ikpR_{56}}f_v}{s_j+ik(\theta\beta_x+\frac{p}{\gamma^2})}dpd^2\beta\times\sum_{n=1}^{N}\frac{e^{-i(kz_n+k\theta x_n)}}{s+ik(\theta\beta_{x,n}+\frac{p_n}{\gamma^2}}\Bigg|_{\tau=0}.\end{aligned} \tag{6.13}$$

Remark. A key step in this derivation is the application of the Cauchy residual theorem which requires the knowledge of the roots of the plasma dielectric function ε_p which can in general not be obtained numerically. This issue is discussed in more detail in the following sections and in [8, sec. IV].

6.1.2 The Laminar Beam Case

The easiest case to evaluate the bunching factor is the laminar beam case where the particle positions are frozen in the co-moving beam frame. For the laminar beam case the following holds. The particles are fixed in beam-frame position with respect to each other and their velocity distribution is

given by a delta function $f_v = \delta(p)\delta^2(\vec{\beta})$. The plasma dielectric function for the laminar beam case can be obtained analytically and reads [8, sec. IV]

$$\varepsilon_p = 1 + \frac{\omega_p^2}{c^2}\frac{1}{s^2} \,. \tag{6.14}$$

The roots of ε_p and hence poles of $b_{R_{56}}$ can easily be found to be

$$s_\pm = \pm i\omega_p/c \,. \tag{6.15}$$

Plugging in $s_\pm$ into (6.13) leads us to the bunching factor after the drift

$$b_{R_{56}} \approx -b_0 \left[\left(\frac{\gamma\omega_p}{c} \right)^2 \frac{1}{1+(\gamma\theta)^2} R_{56} L_d \right], \tag{6.16}$$

with

$$b_0 = \frac{1}{N} \sum_{n=1}^{N} e^{-i(kz_n + k\theta x_n)} \bigg|_{\tau=0} \,. \tag{6.17}$$

In [8, sec. IV] the assumption

$$\frac{\omega_p L_d}{c} \ll 1 \tag{6.18}$$

is made which allows the approximation

$$\sin\left(\frac{\omega_p L_d}{c}\right) \approx \frac{\omega_p L_d}{c} \,. \tag{6.19}$$

The limits of assumption (6.18) will be discussed in section 6.3.

6.1.3 The Quasilaminar Beam Case

In the quasilaminar beam case we assume a Gaussian velocity distribution for the particles in all three spatial dimensions with longitudinal energy spread σ_p and transverse energy spread σ_β. The electron displacement due to thermal motion in a plasma period $\tau_p = 2\pi/\omega_p$ is assumed to be much smaller than $\lambda = 2\pi/k$ longitudinally and λ/θ transversely. In this scenario thermal effects become negligible and we find a plasma dielectric function similar to the laminar beam case. The following assumptions on the transverse velocity spread σ_β and the longitudinal energy spread σ_p are made

$$k\sigma_p/\gamma^2 \ll \omega_p/c, \tag{6.20}$$

and

$$k\theta\sigma_\beta \ll \omega_p/c. \tag{6.21}$$

The Microbunching Gain

The microbunching gain g is the quotient of the statistical averages of the bunching factors squared before and after the drift

$$g = \left\langle |b_{R_{56}}|^2 \right\rangle / \left\langle |b_0|^2 \right\rangle \ . \tag{6.22}$$

With the above given assumptions we obtain for the quasi laminar beam case

$$g_{ql} = \left[\left(\frac{\gamma \omega_p}{c} \right)^2 \frac{1}{1 + (\gamma\theta)^2} R_{56} L_d \right]^2 e^{-(k\sigma_p R_{56})^2}. \tag{6.23}$$

6.2 The Transversely Warm Beam Case

The most interesting case is the transversely warm beam case, where the assumptions on the velocity spread (6.21) are dropped. In this section we look at a transversely warm but longitudinal quasilaminar beam which means we stick to the condition $k\sigma_p/\gamma^2 \ll \omega_p/c$ but make no assumptions on σ_β. After integration by parts of the right hand side of equation (6.11) one obtains

$$\varepsilon_p = 1 + \frac{\omega_p^2}{c^2} \frac{1}{ik\theta} \int_{\tilde{c}} \frac{\frac{\partial f_v}{\partial \beta_x}}{s + ik(\theta\beta_x)} d\beta_x \ , \tag{6.24}$$

with singularity $\beta_x = -s/ik\theta$ and corresponding Landau contour $\tilde{c}$. In equation (6.24) no closed form expression for the roots of ε_p exists, and therefore the roots have to be found numerically. In **??** a dimensionless form of equation (6.24) is given as

$$\varepsilon_p = 1 - \frac{1}{K^2} \int_{\tilde{c}} \frac{\frac{\partial F}{\partial B}}{B - \frac{\Omega}{K}} dB \ , \tag{6.25}$$

depending on only one external dimensionless parameter K. All the roots Ω_j have a negative imaginary part which results in an exponential decay of the microbunching as a function of the drift length. This corresponds to the well known *Landau Damping* which can be easily seen by plugging in the roots $s_j = \Omega$ of the dielectric function into the definition of the bunching factor given in equation (6.13). The damping constant $-\Im(\Omega)$ is a growing function of $K = \frac{k\theta}{k_D}$, with $k_D = \omega_p/c\sigma_\beta$ and hence negligible for small K. However, for $K > 1$ the damping term is bigger than 1 and hence we have a strong damping for angles $\theta > k_D/k$. One can observe that the Landau damping is stronger for bigger K. Since K on the other hand is bigger for larger σ_β corresponding to higher emittance we remain with a smaller angle θ for the microbunching gain for higher emittances. For the microbunching gain in the transversely warm beam case Marinelli and Rosenzweig [8,

equation (25)] derived

$$g = 2\left(\frac{\omega_p}{c\left[1+(\gamma\theta)^2\right]}\gamma^2 R_{56}\right)^2 e^{-(k\sigma_p R_{56})^2}\left[\left|\frac{Ke^{-i\Omega_+(\omega_p/c)L_d}}{1-\frac{\Omega_+^2}{1+K^2}}\right|^2\right.$$
$$\times\left(\left|\frac{K^2\left(1+K^2\right)}{\Omega_+^2}\right| - \sqrt{2\pi}K\frac{\Re\left\{e^{-(\Omega_+^2/2K^2)}\right\}}{\Im\{\Omega_+\}}\right) \tag{6.26}$$
$$\left.-\Re\left\{\left(\frac{Ke^{-\Omega_+(\omega_p/c)L_d}}{1-\frac{\Omega_+^2}{1+K^2}}\right)^2\left(\frac{K^2\left(1+K^2\right)}{\Omega_+^2} - \sqrt{s\pi}K\frac{e^{-(\Omega_+^2/2K^2)}}{\Omega_+}\right)\right\}\right],$$

where Ω_+ denotes the root of the dielectric function (6.25) with positive real part and the smallest damping constant.

6.3 Analytic Solution for a Realistic Beam Scenario

In this section the beam parameters for the following numerical studies in chapter 7 are given and the corresponding analytic solution for the microbunching gain is shown. We try to mimic the beam and simulation parameters used in [8] as far as possible with the given information. In order to achieve the assumptions made in section 6.2 a uniform beam with moderate energy of 135 MeV, a current of $I = 40\,\mathrm{A}$ and an rms envelope size of $\sigma_x = 85\,\mu\mathrm{m}$ is studied. The drift length is chosen to be $Ld = 4\,\mathrm{m}$ and the wavelength of interest is $\lambda = 0.5\,\mu\mathrm{m}$. The transverse beam temperature is varied by studying the different normalized transverse emittances

$$\varepsilon_n \in \{0\,\mathrm{mm\ mrad},\ 0.1\,\mathrm{mm\ mrad},\ 0.5\,\mathrm{mm\ mrad},\ 1\,\mathrm{mm\ mrad}\} \tag{6.27}$$

For the coupling dispersion, we assume $R_{56} = 1\,\mathrm{mm}$. The longitudinal energy spread is given by $\Delta E = 1\,\mathrm{keV}$.

Derived Quantities. From these beam parameters we can derive the quantities needed to solve (6.26). For the relativistic factor γ we find

$$\gamma = \frac{E}{m_0 c} = \frac{135\,\mathrm{MeV}}{0.51\,\mathrm{MeV/c^2}\cdot c^2} \approx 270 \tag{6.28}$$

The number of particles N can be derived from the beam current I by

$$N = \frac{I\cdot l_z}{q\beta c} \approx 2.5\times 10^6 \tag{6.29}$$

where $\beta = \sqrt{1-\frac{1}{\gamma^2}}$ and the longitudinal length of the beam slice is assumed to be $l_z = 3\,\mu\mathrm{m}$. The charge per particle is denoted by q. For the longitudinal

relative energy spread one can find

$$\sigma_p = \frac{\Delta E}{\gamma m_0 c^2} \approx 7.1 \times 10^{-6}. \tag{6.30}$$

The transverse momentum spread is

$$\sigma_{p_x} = \sigma_{p_y} = \frac{\varepsilon_n m_0 c}{\sigma_x} \tag{6.31}$$

and for the given emittances between $6 \times 10^{-6} MeV$ and $6 \times 10^{-3} MeV$. The particle number density n_0 used in [8] leaves room for interpretation. If one assumes that the simulated domain ($l_x \times l_y = 160\,\mu\text{m} \times 160\,\mu\text{m}$) covers the whole transverse extension of the beam the number density reads $n_0 = \frac{N}{l_x \cdot l_y \cdot l_z} \approx 32.5 \frac{1}{(\mu m)^3}$. For the relativstic beam plasma frequency we then obtain

$$\omega_p = \sqrt{\frac{q^2 n_0}{\varepsilon_0 m_0 \gamma^3}} \approx 70\,\text{MHz}\ . \tag{6.32}$$

Another possible interpretation of the setup would be that the particle density in the simulated beam slice corresponds to the maximum density in the Gaussian beam. Under these assumption, we obtain $n_0 = \frac{N}{l_z} \frac{1}{2\pi\sigma_x^2} = 18.35$, corresponding to a plasma frequency of

$$\omega_p = 5.6 \times 10^7 \frac{1}{s} = 56\,\text{MHz}. \tag{6.33}$$

Validity of Assumptions. With the parameters given in section 6.3 most of the assumptions made previously are met, while some are violated or approach a limit. The assumption, that the beam is relatively cold in longitudinal direction, given by $k\sigma_p/\gamma^2 \ll \omega_p/c$ reads for the given parameters $0.00127 \frac{1}{m} \ll 0.242 \frac{1}{m}$ and is well met. The additional condition (6.20) for the quasilaminar beam case is only met for the $\varepsilon_n = 0\,\text{mm mrad}$ case, which agrees with the theory. We expect space charge effects to be important if $k\sigma_\beta \geq \gamma\omega_p/c$, which is the case for all ε_n given above except for the zero emittance case.
For $|K| \ll 1$ equation (6.26) should converge to equation (6.23) which corresponds to the quasilaminar beam case. For the limit for $|K| \to 0$ for equation (6.26) one can find

$$\lim_{K\to 0} g_{warm} = 2 g_{ql} \frac{c^2}{\omega_p^2 L_d^2} \cdot \frac{1}{2} \sin^2\left(\frac{\omega_p L_d}{c}\right), \tag{6.34}$$

which only equals the quasilaminar solution (6.23) for $\frac{\omega_p L_d}{c} \ll 1$. However, for the given parameters above we find $\frac{\omega_p L_d}{c} = 0.967$ or $\frac{\omega_p L_d}{c} = 0.75$ for the

maximum particle density case. Both cases do not not allow the approximation $\sin(x) \approx x$.
Fig. 6.1 shows the analytic solution for the microbunching gain g for the given beam parameters. The plot is not normalized and as a result one can see a discrepancy between the grey line showing the quasilaminar beam case ($\varepsilon_n = 0\,\text{mm mrad}$) obtained by (6.23) and the violet line showing the result obtained by equation (6.26). The difference between the two solutions is the factor 1.4 lost by the approximation $\sin(\frac{\omega_p L_d}{c}) = \frac{\omega_p L_d}{c}$. This factor is also missing for the other solutions obtained by (6.23). The non-smooth behavior close to $\theta = 0$, comes from the numerical approximation of the roots of the plasma dielectric function and could be eliminated by putting more effort on the numerical approximation.

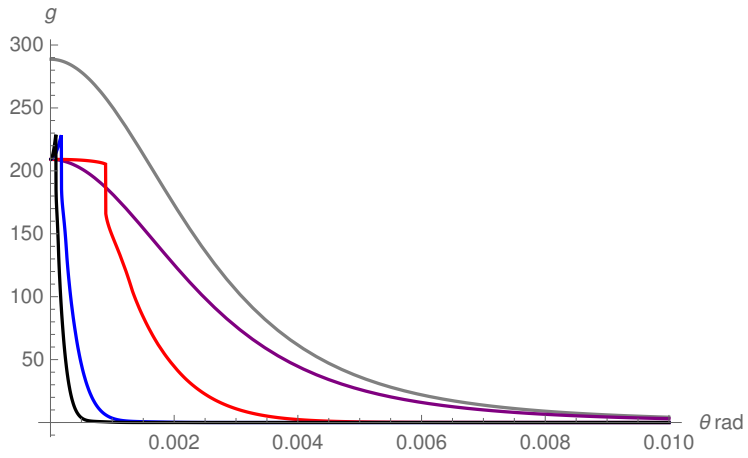

Figure 6.1: Analytic solution for the microbunching gain without normalization for $\omega_p = 70\,\text{MHz}$. The gray line is the solution for $\varepsilon_n = 0$ obtained by the quasilaminar beam approximation (6.23). The other lines are obtained using equation (6.26) for the transversely warm beam case. The violet, red, blue and black lines correspond to the $0, 0.1, 0.5$ and $1\,\text{mm mrad}$ normalized emittance case.

If the solutions for the microbunching gain are normalized to their corresponding value at $\theta = 0$ the obtained results match the ones obtained in [8]. In Figure 6.2 the normalized values are shown and compared to the solution obtained by [8] shown in green.

If we assume the maximum particle density of the beam throughout the simulated beam slice the numerical approximation of the roots of the plasma dielectric function results in oscillatory behavior of the analytic solution shown in Figure 6.3

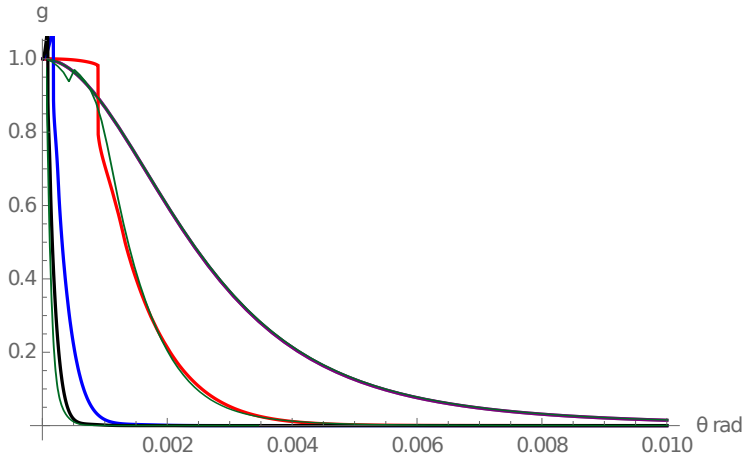

Figure 6.2: Comparison of normalized microbunching gain for $\omega_p = 70\,\text{MHz}$ with [8]

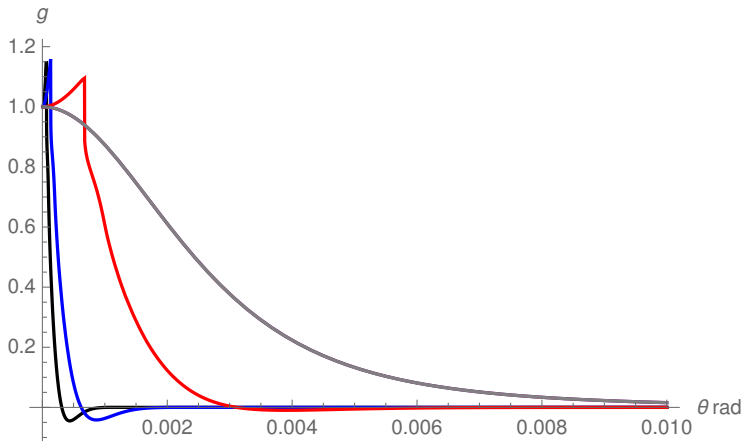

Figure 6.3: Normalized microbunching gain for $\omega_p = 56\,\text{MHz}$.

Side Note. Figures 3 and 6 of [8] which are supposed to show the same analytic solution do not exactly match, which is illustrated in Figure 6.4. This shows that the analytic solution of the microbunching gain is sensitive to minor details and has to be evaluated carefully. The numerical solution for the roots of the plasma dielectric function are a major error source for deviations in the analytic solution.

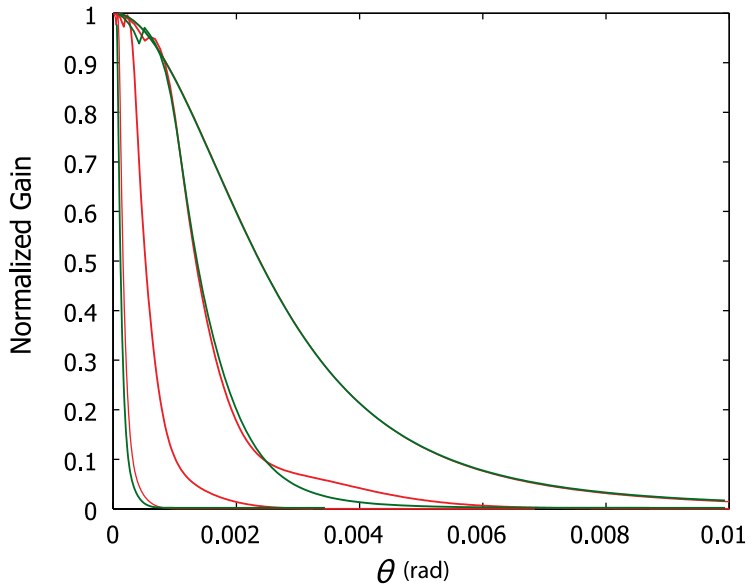

Figure 6.4: Comparison of analytic solutions presented in [8].

Chapter 7

Numerical Simulation of Microbunching

7.1 Experimental Setup

The experimental setup for the modeling of microbunching effects following [8] is a drift without external force fields. Subsequent rearrangement of the longitudinal coordinates by the energy-position coupling constant R_{56} ($\approx 1mm$) accounts for the influence of a dispersive element. The modeled beam has a beam current of $I = 40\,\mathrm{A}$. The simulation is performed in three dimensions with periodic boundary conditions in all three dimensions. This allows us to model only a small fraction of the beam of 165 µm in the transverse directions and 3 µm in longitudinal direction as suggested in [8]. The energy of the beam is 135 MeV. From $E = \gamma m_0 c^2$ where m_0 corresponds to the rest mass of an electron we obtain

$$\gamma_0 = \frac{E}{m_0 c^2} = \frac{135\,\mathrm{MeV}}{0.510\,998\,910\,\mathrm{MeV/c^2} \cdot c^2} \approx 264. \tag{7.1}$$

To be consistent with [8] we will assume $\gamma_0 = 270$. For the longitudinal velocity of the beam relative to the speed of light we obtain

$$\beta_0 = \sqrt{1 - \frac{1}{\gamma_0^2}} = \sqrt{1 - \frac{1}{270^2}} = 0.999993 \tag{7.2}$$

From the definition of current as total charge Q over time and $t = \frac{l}{\beta c}$ we obtain

$$I = \frac{Q}{t} = \frac{eN}{t} = \frac{eN}{l}\beta c \tag{7.3}$$

with elementary charge e. This gives us a number of $N \approx 2.5 \times 10^6$ real particles in the simulated beam slice of $l = 3\,\mu\mathrm{m}$.

Transverse Momentum Distribution. The transverse rms envelope size of the beam is given by $\sigma_x = 85\,\mu\text{m}$ and we will perform simulations for normalized emittance values $\varepsilon_{nx} = \beta\gamma_0\varepsilon_x$ for

$$\varepsilon_{nx} \in [0\,\text{mm mrad}, 0.1\,\text{mm mrad}, 0.5\,\text{mm mrad}, 1\,\text{mm mrad}].$$

The corresponding transverse spread in the second order momentum $x' = \frac{dx}{dz}$ is $\sigma_{x'}$. The normalized transverse emittance reads

$$\varepsilon_{nx} = \beta\gamma\sigma_x\sigma_{x'} \,. \tag{7.4}$$

For the $\varepsilon_{nx} = 1\,\text{mm mrad}$ case we find $\sigma_{x'} = \sigma_{y'} = 0.0436\,\text{mrad}$. The second order moment is related to the transverse momentum via

$$p_x = m_0\beta\gamma c x' \,, \tag{7.5}$$

and hence we can find the transverse momenta spread σ_{p_x} for given $\sigma_{x'}$. For the initial transverse momenta distribution we therefore assume

$$f(p_x, p_y) = \frac{1}{2\pi\sigma_{p_x}\sigma_{p_y}} e^{-\frac{p_x^2}{2\sigma_{p_x}^2} - \frac{p_y^2}{2\sigma_{p_y}^2}} \tag{7.6}$$

The transverse momenta in the beam frame are the same as in the lab frame.

Longitudinal Momentum Distribution. In longitudinal direction we assume an energy spread of 1 keV which corresponds to $\gamma = 1.9 \times 10^{-3}$
The energy deviation of a particle in the laboratory frame is assumed to follow a normal distribution with

$$f(\Delta\gamma) = \mathcal{N}(0, 1.9 \times 10^{-3}) \tag{7.7}$$

The energy deviation from a reference particle (with reference momentum p_{z_0}) in the lab frame is defined as

$$\delta = \frac{\Delta E}{E} = \frac{\Delta\gamma}{\gamma} = \frac{\Delta p_z}{p_{z_0}} \tag{7.8}$$

and hence the momentum deviation from the reference particle is found to be $\Delta p_z = p_{z_0}\delta$. With $p_{z_0} = \beta\gamma_0 m_0 c$ we find for the distribution of the momentum deviation in the lab frame

$$f(\Delta p_z) = f(\Delta\gamma)\beta m_0 c \tag{7.9}$$

Coordinates corresponding to the *beam frame* are from now on denoted with a tilde. The longitudinal momentum in the *beam frame* reads after Lorentz transformation

$$\tilde{p}_z = \frac{1}{\gamma}\Delta p_z \tag{7.10}$$

Spatial Particle Distribution. Since the simulated beam slice has periodic boundary conditions in all three dimensions, we distribute the N particles uniformly over the simulated domain in the laboratory frame. The Lorentz transformation to the beam frame reads

$$\begin{aligned}\tilde{x} &= x \\ \tilde{y} &= y \\ \tilde{z} &= \gamma z\end{aligned}$$

Equations of Motion and Time Integration. The equations of motion read

$$\frac{d\vec{p}}{dt} = \frac{d}{dt}\left(\vec{\beta}\gamma m_0 c\right) = q\vec{E} \tag{7.11}$$

The integration of motion is done in the *beam frame* using the leapfrog algorithm which reads

$$\begin{aligned}\vec{\tilde{x}}_i &= \vec{\tilde{x}}_{i-1} + \frac{\vec{\tilde{p}}_{i-1/2}}{m_0}\Delta\tilde{t} \\ \vec{\tilde{p}}_{i+1/2} &= \vec{\tilde{p}}_{i-1/2} + q\vec{\tilde{E}}(\vec{\tilde{x}}_i)\Delta\tilde{t}\end{aligned}$$

The time in the laboratory frame needed to simulate a drift of length $L_d = 4\,\mathrm{m}$ is

$$t_{end} = \frac{L_d}{\beta_0 c} \approx \frac{4\,\mathrm{m}}{3\times 10^8\,\mathrm{m/s}} \approx 1.33\times 10^{-8}\,\mathrm{s} \approx 13.3\,\mathrm{ns} \tag{7.12}$$

This corresponds to a time t' in co-moving frame of

$$\begin{aligned}\tilde{t}_{end} &= \gamma\left(t_{end} - \frac{\beta_0 L_d}{c}\right) \\ &= \gamma\left(\frac{L_d}{\beta_0 c} - \frac{\beta_0 L_d}{c}\right) \\ &= \frac{\gamma L_d}{c\beta_0}\left(1-\beta_0^2\right) \\ &= \frac{\gamma L_d}{c\beta_0}\frac{1}{\gamma^2} \\ &= \frac{1}{\gamma}t_{end}\end{aligned}$$

Energy-Position Coupling. The influence of longitudinal dispersion is modeled by applying the coupling constant $R_{56} \approx 1\,\mathrm{mm}$ to rearrange the particles in longitudinal direction relative to their energy deviation in every time step. In the lab frame this rearrangement reads

$$z \to z + \frac{\Delta\gamma}{\gamma_0}R_{56} = z + \frac{\Delta p_z}{p_{z_0}}R_{56} \tag{7.13}$$

quantity	unit	order of magnitude
length	m	$\sim 1 \times 10^{-6}$
time	mc^{-1}	$t'_{end} \sim 1 \times 10^{-2}$
mass	MeV c^{-2}	$\sim 1 \times 10^{-1}$
particle rest mass m_0	MeV c^{-2}	$\equiv 0.510\,998\,9$
momentum	MeV c^{-1}	$\sim 1 \times 10^{-3}$
charge	e	~ 1
energy	MeV	~ 1
electric field strength	MeV $\mathrm{e}^{-1}\mathrm{m}^{-1} = \mathrm{MVm}^{-1}$	$\sim 1 \times 10^{-3}$
coulomb constant k_e	m MeV e^{-2}	$\equiv 1.439\,964 \times 10^{-15}$

Table 7.1: Physical units and constants used.

The momentum deviation and longitudinal position of a particle in the *lab frame* reads after Lorentz transformation

$$\Delta p_z = \gamma_0 \tilde{p}_z; \quad z = \frac{1}{\gamma_0}\tilde{z}. \tag{7.14}$$

With the reference momentum $p_{z_0} = \beta\gamma_0 m_0 c$ we find for the longitudinal particle position after dispersion

$$z = \frac{1}{\gamma_0}\tilde{z} + \frac{\tilde{p}_z}{\beta m_0 c} R_{56} \tag{7.15}$$

Choice of Physical Units. Table 7.1 shows the choice of physical units and constants used in the simulation

7.2 Simulation Results

In this section the experimental results for the microbunching simulation are presented. As in [8], the transverse domain is chosen to be $l_x \times l_y = 160\,\mu\mathrm{m} \times 160\,\mu\mathrm{m}$ and the longitudinal extension in the laboratory frame is $l_z = 3\,\mu\mathrm{m}$. The beam parameters are presented in section 7.1 and correspond to the analytical solution presented in chapter 6. We evaluate the microbunching gain for the two different interpretations of the particle density in the simulated domain as presented in chapter 6.

7.2.1 Particle-mesh simulation

It turned out that a particle-mesh simulation with 64^3 grid cells is suitable for studying the microbunching gain in the given beam setup. This number of grid cells corresponds to approximately ten particles per grid cell. In the language of a P^3M simulation introduced in chapter 2 the simulations are

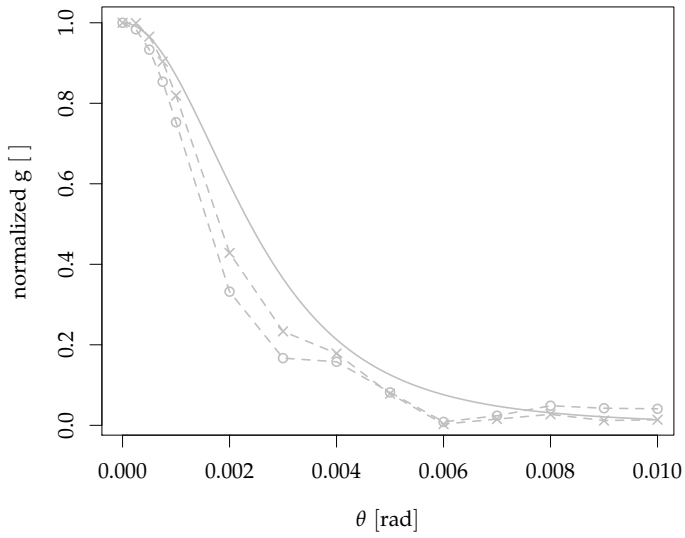

Figure 7.1: Comparison between the analytic solution of the microbunching gain depicted in the solid gray line and the numerical results shown as dashed lines for the 0 mm mrad emittance case. The circles correspond to $\omega_p = 56\,\text{MHz}$ and the crosses correspond to $\omega_p = 70\,\text{MHz}$. The simulation results are averaged over 50 independent runs.

performed with $r_{cut} = 0$ and $\alpha = 4 \times 10^4$ which corresponds to $\alpha = \frac{1}{2h}$ where h is the cell width in the longitudinal dimension in the beam frame, where the mesh solver is applied. In the following sections the simulation results for the microbunching gain for various transverse emittance values are studied.

The quasilaminar beam case ($\varepsilon_n = 0\,\text{mm mrad}$)

For the quasilaminar beam with $\varepsilon_n = 0\,\text{mm mrad}$ one observes microbunching gain similar to the one predicted by the analytic solution (6.23). Figure 7.1 compares the simulation results averaged over 50 independent runs to the analytic solution. The solution varies slightly for the different particle densities simulated. Qualitateively one can see a strong microbunching at $\theta = 0$ decreasing for wider opening angles.

In Figure 7.2 the evolution of the longitudinal momentum of the beam in the laboratory frame is shown. A bunching in the longitudinal momenta can be observed. However, the direction of the bunching varies arbitrarily. This explains the broad width of the microbunching gain as predicted by the analytical solution.

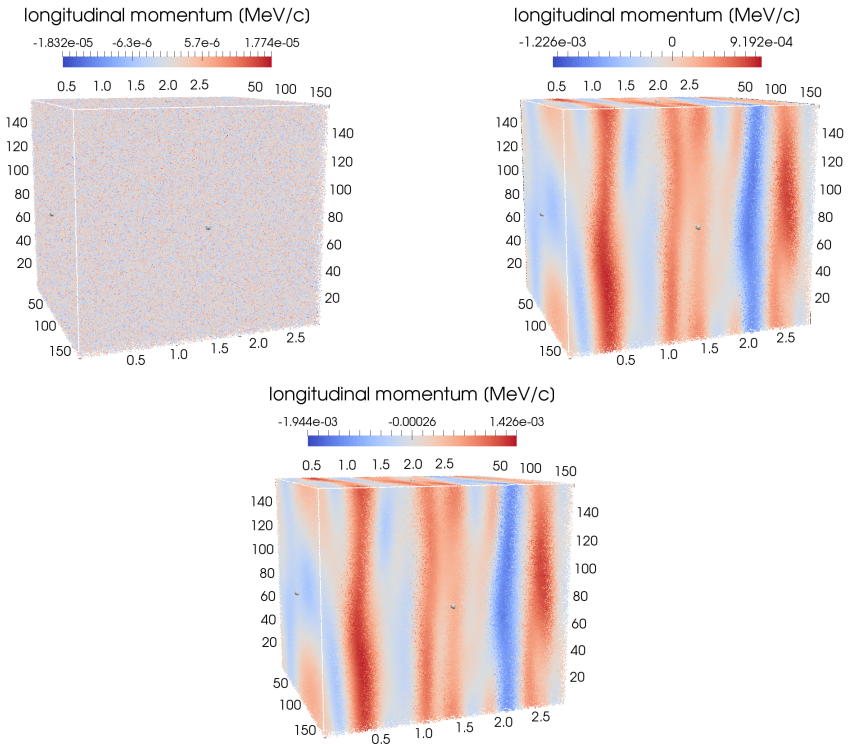

Figure 7.2: Evolution of the longitudinal momenta. From top to bottom: the initial state of the beam, the state after 2 m drift and the final state after 4 m drift. Longitudinal coordinate z pointing from left to right, x-axis pointing from bottom to top and y-axis pointing from back to front.

One can clearly see, that the bunching effect leads to a formation of slices in the transverse dimension and hence a projection to one of the transverse plains allows a better visualization of the bunching behavior. The projection of the three dimensional beam shown in Figure 7.2 to the two dimensional xz configuration space is shown in Figure 7.3

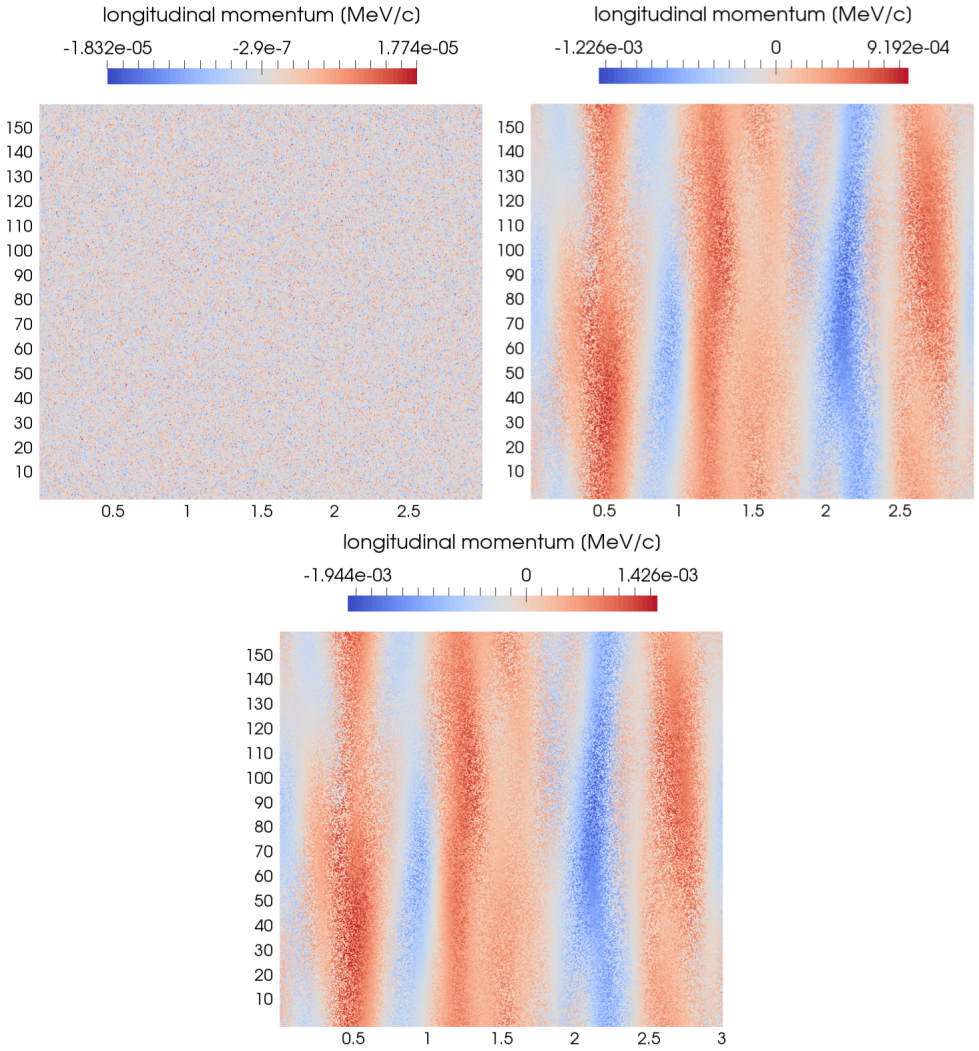

Figure 7.3: xz-configuration space of the beam in the laboratory frame. From left to right, top to bottom, the initial state of the beam, the state after 2 m drift and the final state after 4 m drift are shown. Longitudinal coordinate z pointing from left to right and x-axis pointing from bottom to top.

Emittance $\varepsilon_n = 0.1\,\text{mm mrad}$

For an increasing transverse temperature, the analytic solution predicts a narrowing in the microbunching gain. Figure 7.4 shows the comparison of the numerical results and the predicted analytic solution. For the higher emittance, the simulation with the higher particle density is closer to the analytic solution. However, the narrowing of the microbunching gain is not as distinctive as predicted.

The projection of the longitudinal momenta to the x-z plane shows slightly more distinct microbunches than for the zero emittance case. However, in agreement with the computed microbunching gain, the directions of the bunches still vary. Figure 7.5 shows the microbucnhing for a single simulation run after 4 m drift in the $\varepsilon_n = 0.1\,\text{mm mrad}$ case.

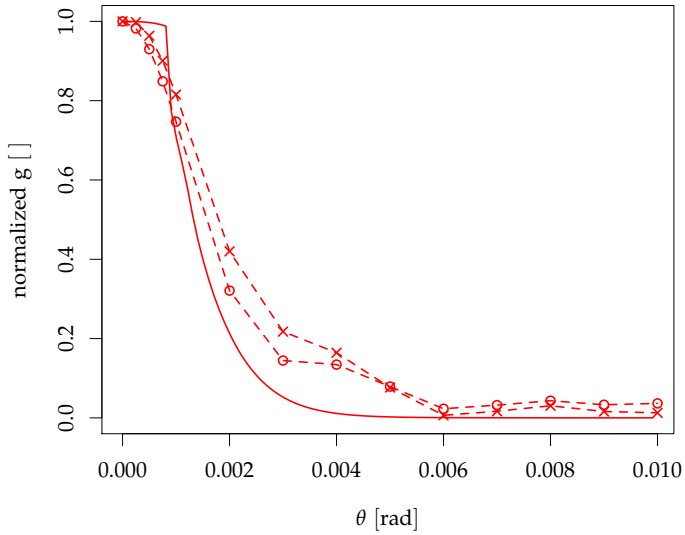

Figure 7.4: Comparison between the analytic solution of the microbunching gain depicted in the solid red line and the numerical results shown as dashed lines for the 0.1 mm mrad emittance case. The circles correspond to $\omega_p = 56\,\mathrm{MHz}$ and the crosses correspond to $\omega_p = 70\,\mathrm{MHz}$. The simulation results are averaged over 50 independent runs.

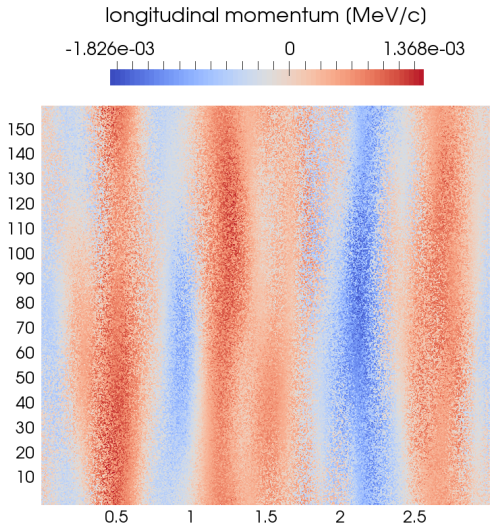

Figure 7.5: xz-configuration space of the beam in the laboratory frame. Final state after 4 m drift for $\varepsilon_n = 0.1\,\mathrm{mm\ mrad}$.Longitudinal coordinate z pointing from left to right and x-axis pointing from bottom to top.

Emittance $\varepsilon_n = 1\,\text{mm mrad}$

For the $\varepsilon_n = 1\,\text{mm mrad}$ the microbunching is pronounced in the $\theta = 0$ direction and hence the microbunching gain narrows compared to the colder beam cases. However, the numerical simulation is off the analytic prediction, which can be seen in Figure 7.6. One has to consider, that in the analytic solution for the $\varepsilon_n = 1\,\text{mm mrad}$ case the analytic results become inaccurate [8]. For this case the particle density in the simulation does not play a big role towards the microbunching formation.

The projection of the longitudinal momenta to the x-z plane shows very distinct microbunches along the longitudinal axis as shown in Figure 7.7.

The microbunching also becomes visible by looking at the histogram of the longitudinal beam coordinate after application of the dispersion factor R_{56}. For higher dispersion factors the amplitude of the bunching is enhanced. This has no influence on the microbunching gain, since we normalize to the value at $\theta = 0$. Figure 7.8 shows the histograms of the longitudinal beam coordinate for different coupling constants R_{56} ranging from 0 mm to 100 mm.

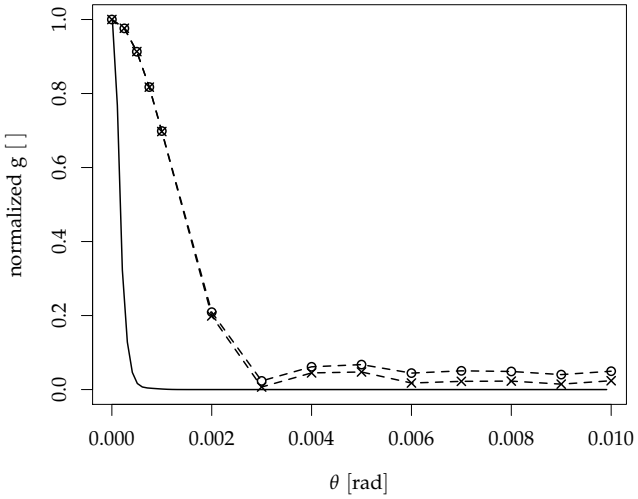

Figure 7.6: A comparison between the analytic solution of the microbunching gain depicted in the solid red line and the numerical results shown as dashed lines for the 1 mm mrad emittance case. The circles correspond to $\omega_p = 56\,\text{MHz}$ and the crosses correspond to $\omega_p = 70\,\text{MHz}$. The simulation results are averaged over 50 independent runs.

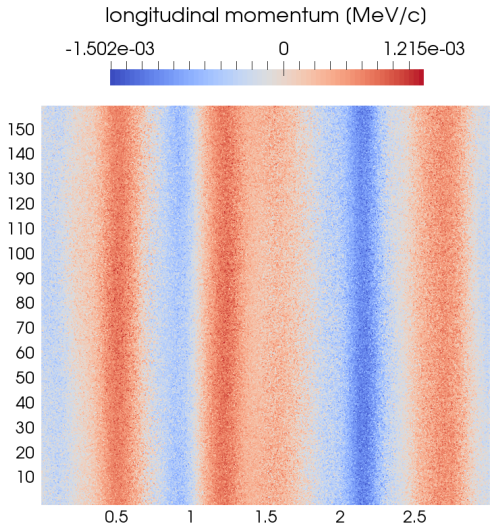

Figure 7.7: xz-configuration space of the beam in the laboratory frame. This is the final state after 4 m drift for $\varepsilon_n = 1$ mm mrad. Longitudinal coordinate z pointing from left to right and x-axis pointing from bottom to top.

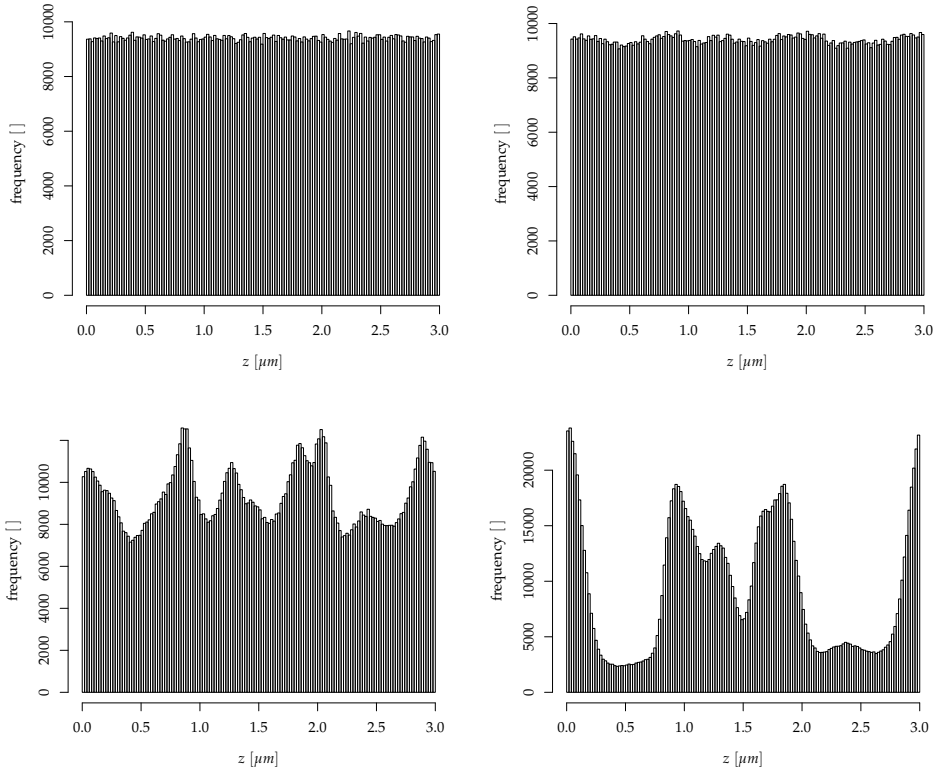

Figure 7.8: Histograms of the z coordinate after $4m$ drift. From left to right, top to bottom the coupling constant is $R_{56} \in \{0\,\text{mm}, 1\,\text{mm}, 10\,\text{mm}, 100\,\text{mm}\}$.

Summary of PM results

Our simulation can qualitatively reproduce the microbunching effects and a narrowing of the microbunching gain with increasing normalized transverse emittance ε_n can be observed. However the computed microbunching gain is not as distinctive as predicted by the analytic solution. This needs further investigation of the initial beam parameters. Figures 7.10 and 7.11 show a summary of the simulated and analytic microbunching gain for different particle densities. The density has only minor effects on the result.

We also performed simulations using the same beam parameters and particle density but doubled the transverse domain size to $l_x = l_y = 320$ which results in a simulation of 5643165 real particles. For this simulation, we obtained more destinctive results for the microbunching gains for the various emittance values. Although, the 0.0 mm mrad case and the 0.1 mm mrad still show the same bunching gain. The results are summarized in Figure 7.11

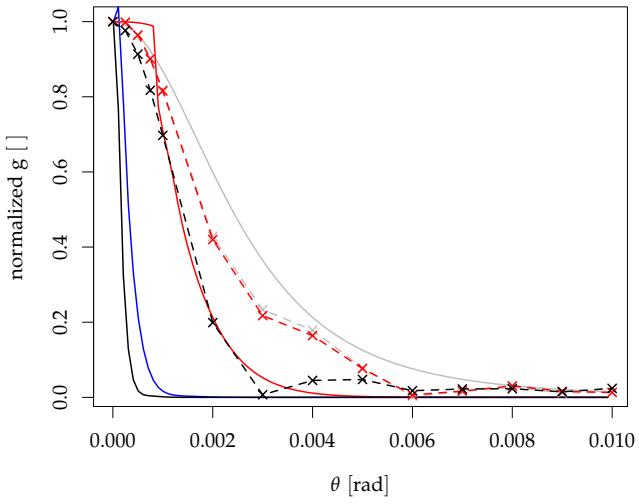

Figure 7.9: Simulation results and analytic solution for the microbunching gain for $\omega_p = 70\,\text{MHz}$. The gray line corresponds to $\varepsilon_n = 0$. The red, blue and black lines correspond to the $0.1, 0.5$ and $1\,\text{mm mrad}$ normalized emittance case.

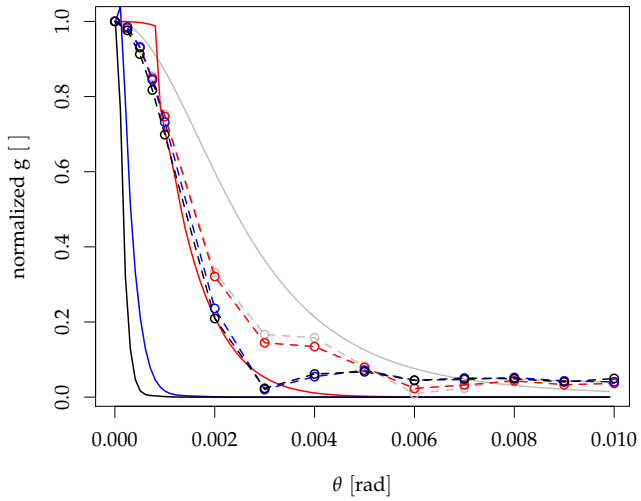

Figure 7.10: Simulation results and analytic solution for the microbunching gain for $\omega_p = 56\,\text{MHz}$. The gray line corresponds to $\varepsilon_n = 0$. The red, blue and black lines correspond to the $0.1, 0.5$ and $1\,\text{mm mrad}$ normalized emittance case.

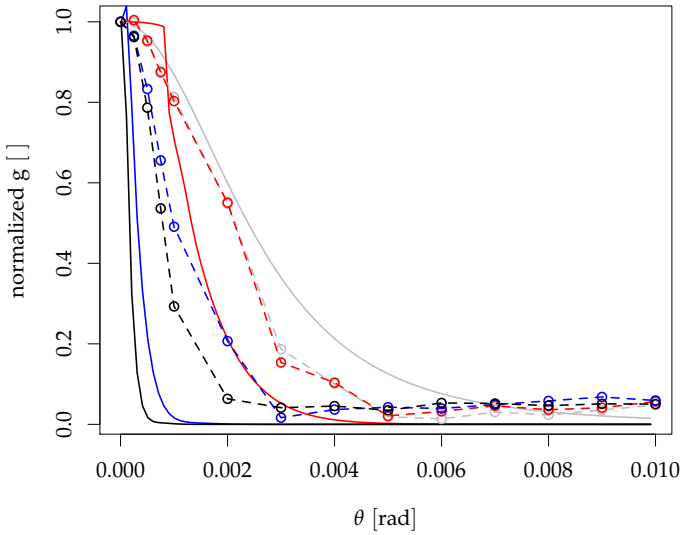

Figure 7.11: Simulation results and analytic solution for the microbunching gain for $\omega_p = 56\,\text{MHz}$ and $l_x = l_y = 320$. The gray line corresponds to $\varepsilon_n = 0$. The red, blue and black lines correspond to the $0.1, 0.5$ and $1\,\text{mm mrad}$ normalized emittance case.

7.2.2 Influence of Coulomb Collisions

In this section we study the influence of particle particle collisions on the example of the 1 mm mrad emittance beam. As a reference case, we consider the simulation from section 7.2.1 with a transverse domain size of 320 µm and a 64^3 computational mesh without particle particle interactions ($r_{\text{cut}} = 0$). The resulting microbunching is compared to a high resolution PM simulation using a $256 \times 256 \times 1024$ computational mesh corresponding to 12 mesh cells per particle (!). Another simulation is performed using a 64^3 computational mesh with additional particle particle interactions within a cutoff equal to one grid cell in longitudinal direction. The xz trace space plots for all three experiments are shown in Figure 7.12. The additional contribution of short range forces result in a significant increase of transverse momenta, which makes it difficult to visually judge the influence on the microbunching in configuration space. Although, it looks like the overal bunching structure at $\lambda = 0.5\,\mu\text{m}$ seems to be similar with superimposed high momenta obtained from Coulomb collisions. Figure 7.13 showing a comparison of the obtained microbunching gain of all three simulations confirms the suspicion that the particle particle collisions have no significant influence on the microbunching gain at $\lambda = 0.5\,\mu\text{m}$.

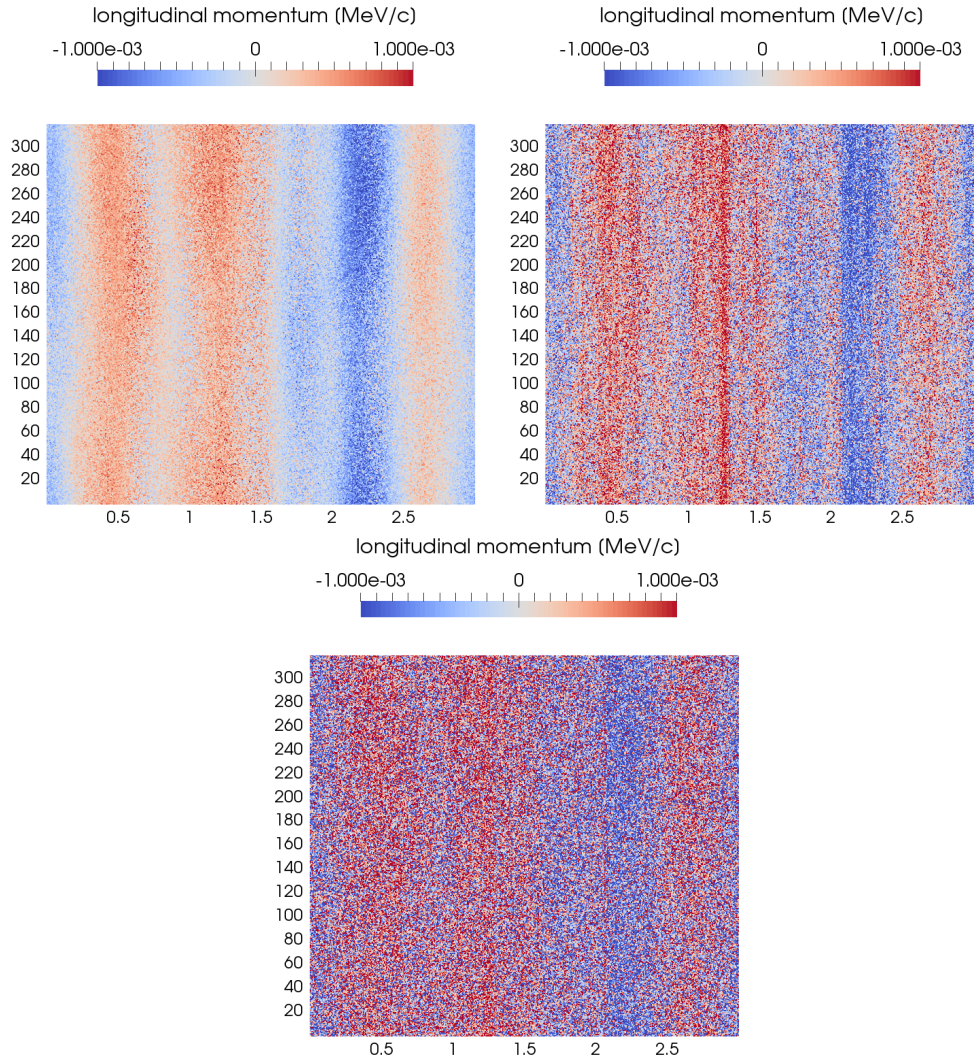

Figure 7.12: Influence of coulomb collisions onto xz configuration space distribution for $\varepsilon_n = 1\,\text{mm mrad}$. Base experiment with 64^3 PM solver (top left), high resolution PM simulation (top right), P^3M simulation with $r_{\text{cut}} = h$ (bottom). Longitudinal coordinate z pointing from left to right and x-axis pointing from bottom to top.

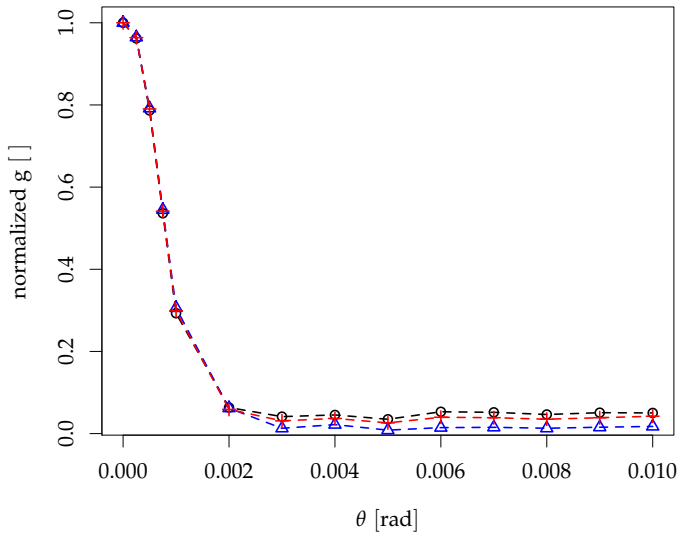

Figure 7.13: Comparison of normalized microbunching gain for $\varepsilon_n = 1\,\text{mm mrad}$ with different amount of coulomb collision influence. The black line corresponds to a PM only simulation on 64^3 grid. The blue line represents a PM only simulation on a $256 \times 256 \times 1024$ grid and the red line shows a P^3M simulation with $r_c = 1h$

7.2.3 Conclusion

The microbunching gain in a high relativistic electron beam was studied for different normalized emittances. The results presented in [8] have been reproduced qualitatively within the the available information about their simulation setup. Our experiments show that Coulomb collisions play no significant role in the formation of microbunching in high relativistic beams which agrees with the theory based on Vlasov-Poisson equation, which does not account for Coulomb collisions as well [8]. The negligibility of Coulomb collisions for the study of microbunching phenomena also agrees with results obtained by Mitri [26], who studied intra beam scattering in high brightness electron linacs. The requirement of very high resolution simulation as claimed by Marinelli and Rosenzweig has not been proven true within our numerical studies, based on the P^3M method. Using more grid cells than particles available in the PM simulation as presented in [27] (here called Molecular Dynamics Simulation), did not show enhanced microbunching effects in our numerical experiments. Additional studies are needed to evaluate the gap between analytic solution and numerical simulation using our P^3M implementation. If it turns out that an oversampling of 260 grid points per particle [27] is necessary to obtain the required resolution for microbunch-

ing studies, then the P^3M method is an efficient alternative to resolve for short range force contribution as will be shown in chapter 8.

Chapter 8

Disorder Induced Heating

As shown in chapter 7 the influence of Coulomb collisions to the microbunching is rather small. It is worth to mention that the microbunching effect is only one space charge effect, which can be observed in an all-embracing particle accelerator simulation. Especially in areas with a rather cold beam the influence of Coulomb collisions becomes significant and has to be resolved. The P^3M algorithm allows to vary the influence of short range forces throughout the whole simulation. This allows us to study the influence of Coulomb collisions in every simulated element of the accelerator. For example, Coulomb collisions play a major role for beam dynamics simulation right after the electron gun where the beam has relatively low energy and the particle distribution is influenced by shot noise from the electron gun. In the experiment presented in this chapter we therefore study the influence of particle collisions to the beam heating in a coasting beam.

8.1 The Disorder Induced Heating Process

The phenomenon of disorder induced heating is known from ultracold plasmas [28]. It has been observed that in ultracold plasmas with initial random distribution of the ions, the ions relax into a more structured state. During this transition from the initially disordered state to a state structured by Coulomb forces energy gets transferred from potential to kinetic energy of the ions satisfying the conservation of energy. The plasma gets heated.
The phenomenon of disorder induced heating has also been experienced in cold electron beams generated by a photoemission source [29]. Here the disorder induced heating limits the achievable beam brightness. An important parameter is the plasma coupling parameter $\Gamma_{eq} = \frac{E_{kin}}{E_{pot}}$ which is defined as the ratio of the thermal energy to the potential energy [29]. Knowledge of this parameter gives us an estimate of the amount of heating to be expected and a lower bound of the minimum electron temperature in a cold electron

beam. A higher minimum temperature on the other hand means a lower beam brightness which is usually desired to be kept as high as possible. In order to resolve the disorder induced heating process in a computer simulation it is unavoidable to account for Coulomb collisions since they are the driving forces of the ordering process. The applicability of the P^3M algorithm to the simulation of this phenomenon is shown in the following sections.

8.2 Experimental Setup and Simulation Parameters

The experiment presented in this section follows to a large extent the simulations performed in [12]. The simulation parameters are chosen to correspond to a realistic beam close to the electron gun and are a realistic scenario for the planned ultrafast electron diffraction experiment at Lawrence Berkeley Lab [30]. A spherical coasting beam of radius $R = 17.74\,\mu\text{m}$ carrying 25 fC of real electron charges which corresponds to $N = 156055$ electrons is simulated over the time of 5 plasma periods. The beam is under constant focusing implemented by a linear force field towards the beam center, with magnitude in the order of the average space charge force field. The beam is initially at rest and the particles have zero momentum in each direction. The particles are distributed uniformly within a sphere of radius R. With electron charge e, electron mass m_0 and electric constant ε_0 the plasma frequency in our simulation is

$$\omega_p = \sqrt{\frac{Ne^2}{m_0\varepsilon_0}} \approx 1.45 \times 10^{11}\frac{1}{s} = 145\,\text{GHz}\,. \tag{8.1}$$

Hence, one plasma period has a duration of

$$\tau = \frac{2\pi}{\omega_p} \approx 43\,\text{fs} \tag{8.2}$$

and 5 plasma periods correspond to a total simulation time of

$$T_{\text{end}} = 215\,\text{fs}\,. \tag{8.3}$$

Since the particle collisions can lead to very large momenta between two colliding particles a relatively small time step of $\Delta t = T_{\text{end}}/1000 = 2.15\,\text{fs}$ has been chosen. In contrast to the simulation presented in [12] we adhere to the periodic boundary conditions. Open boundaries are approximated by placing the beam into the center of a cubic computational domain with edge length $L = 100\,\mu\text{m}$. Furthermore, our P^3M implementation uses an interaction splitting based on Gaussian screening charges as described in chapter 2 in contrast to the polynomial splitting approach used in [12]. The computational mesh width is chosen to be $h = 0.39\,\mu\text{m}$ which corresponds to a grid

with 256^3 cells. In order to study the influence of the particle-particle part of the solver corresponding to the Coulomb collisions between particles, the cutoff r_c has been varied from $r_c = 0\,\mu\text{m} = 0h$ to $r_c = 3.125\,\mu\text{m} = 8.0h$. The splitting parameter α was chosen to be $\alpha = 2/r_{cut}$.

8.3 Simulation Results

From the plasma coupling parameter $\Gamma_{eq} = 2.23$ obtained from molecular dynamics simulation [29] one can compute an estimate for the normalized $x-$emittance in equilibrium state which is found to be $\varepsilon_x = 0.491\,\text{nm}$ [12]. Figure 8.1 shows the emittance growth over 5 plasma periods. With increasing cutoff of particle-particle interactions, the emittance converges to the analytical solution for the thermal equilibrium. The oscillations have a period of one half of a plasma period which is a characteristic of the disorder induced heating process [12]. Compared to the results obtain by Mitchel and Qiang the periodic boundary conditions and the different interaction function splitting appears to have no significant influence on the numerical results.

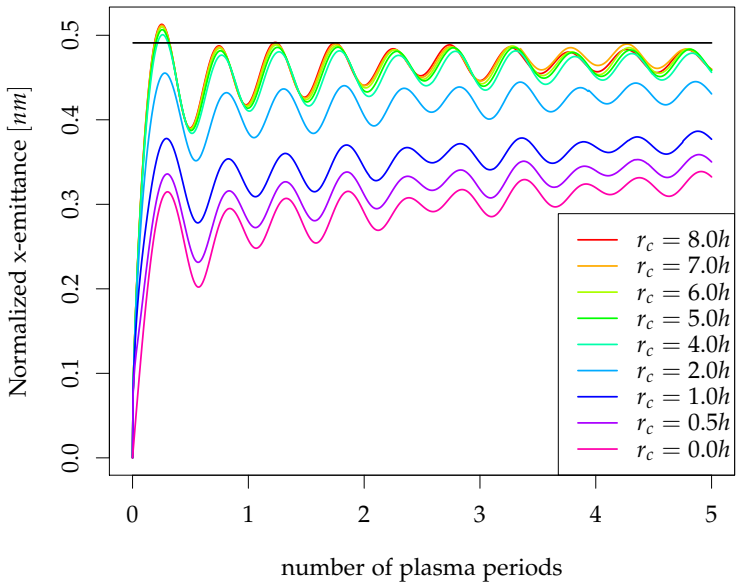

Figure 8.1: Emittance growth for the disorder induced heating process for various values of the cutoff radius r_c. The equilibrium solution for the normalized $x-$emittance $\varepsilon_x = 0.491\,\text{nm}$ is represented by the horizontal black line.

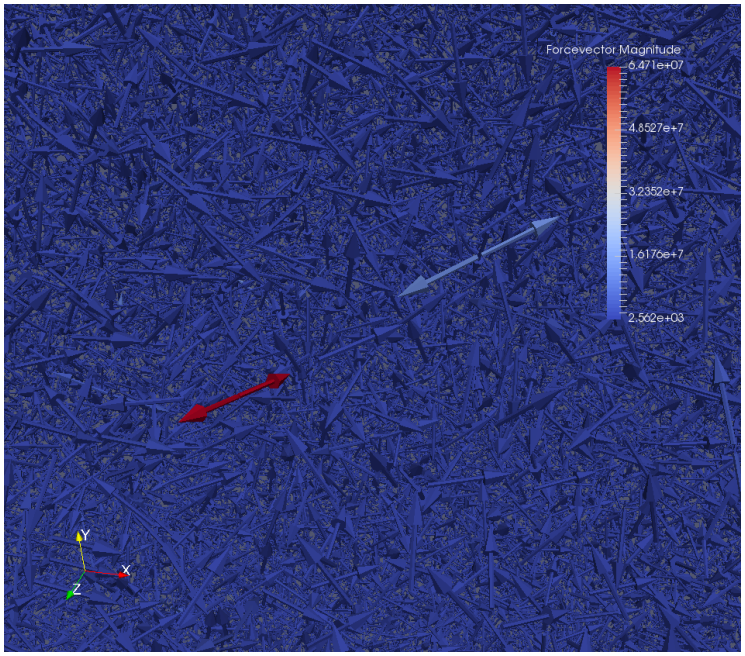

Figure 8.2: Force vectors acting on the particles with particle-particle interactions.

8.4 Particle Collisions

Coulomb collisions become apparent looking at the force vector distribution within the particle beam. Figure 8.2 shows a fraction of the initial particle distribution where two particle collisions appear with turned on particle-particle interactions. The arrows correspond to the forces acting on each particle and the magnitude of the forces is represented by the coloring of the glyphs. For two particles very close to each other the repulsive forces between the colliding particles dominate and act in opposite directions. Figure 8.3 shows the very same excerpt of the beam without particle-particle interactions contributing to the field solver. We do not see any explicit particle collisions and no opposite forces can be spotted by simply looking at the force vectors. The range of the magnitude of the forces is shorter since we are looking at an average force field computation instead of the single real electron-electron interactions.

8.5 Conclusion

The disorder induced heating simulation presented in this chapter shows that our P^3M implementation can resolve coulomb collisions within a particle beam. The additional resolution to the particle-mesh approach can

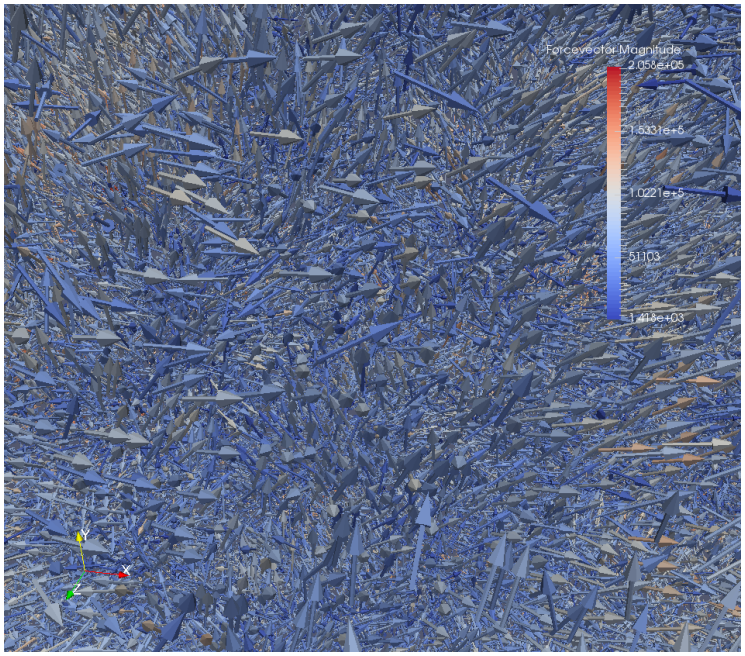

Figure 8.3: Force vectors acting on the particles without particle-particle interactions.

be used for those parts of a beam dynamics simulation of a real machine, where collisions play a significant role. This paves the way for a start-to-end beam dynamics simulation using the P^3M algorithm to efficiently study the effect of Coulomb collisions where necessary and turning of the PP part for those parts of the simulation where no influence of Coulomb collisions is expected.

Chapter 9

Performance Analysis

In this chapter some basic performance tests are presented. Since the focus of the present thesis is on the applicability of the P^3M algorithm to particle accelerator simulations no effort was made towards parallelization beyond the distributed memory implementation based on a spatial domain decomposition. The implementation presented was sufficient to perform the previously described simulations within less than an hour of computation time, using up to 32 processing units. Keeping this in mind, the presented performance analysis in this chapter highlights the bottlenecks of the standard P^3M implementation to show the path to a more efficient implementation, which will be outlined in chapter 10.

9.1 Time Dependence on Cutoff Radius

The overall execution time of the P^3M algorithm is by design strongly dependent on the amount of particle particle interactions involved in the computations. The number of pair interactions directly computed is controlled by the cutoff radius r_{cut}, which is usually related to the mesh width h used for the particle-mesh solver. Figure 9.1 shows the computation time spent per particle particle and particle mesh step for different values of r_{cut}. The underlying experiment is the disorder induced heating presented in chapter 8. The results are averaged over the 1000 time steps required for the simulation of 5 plasma periods. The simulation was performed on a single node of Cori providing two 2.3 GHz 16-core Haswell$^{\text{TM}}$ processors. For very small cutoff radii the chaining mesh used for the particle particle solver becomes very fine and the overhead of looping through all mesh cells slows down the computation. Starting from a cutoff radius equivalent to the double mesh width for the particle mesh solver we see the computation time for the PP part increasing proportional to r_{cut}^3 which agrees with the theoretical computational complexity shown in chapter 2. Since the mesh for the PM solver

is kept constant throughout the simulations the execution time for the PM solver doesn't change with the cutoff radius.

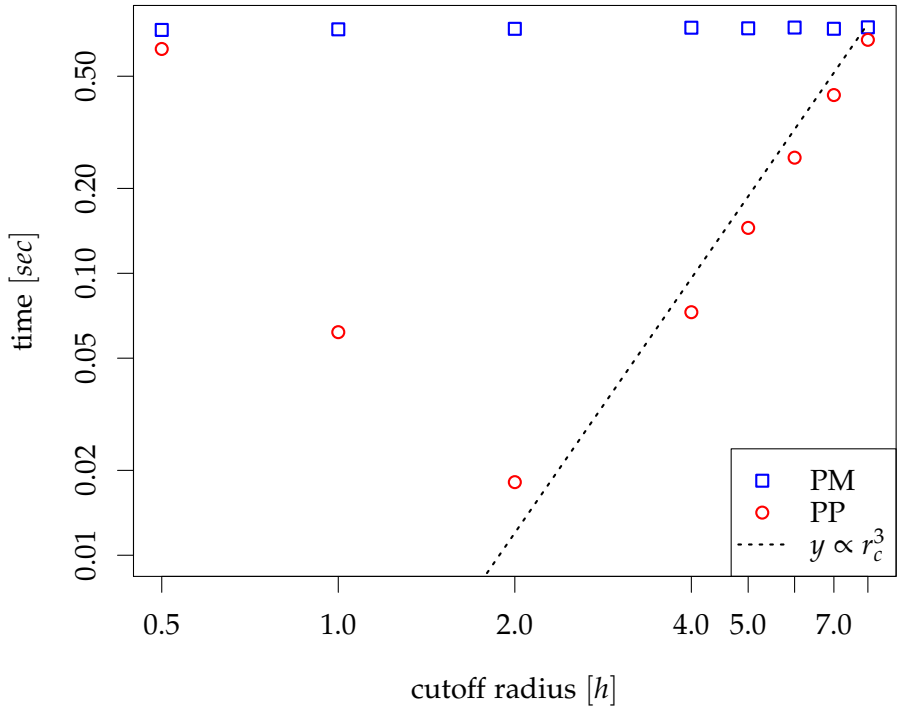

Figure 9.1: Time spent per time step for particle particle solver and particle mesh solver using a 256^3 grid and a splitting parameter $\alpha = 1/(2h)$. The results are averaged over 1000 time steps on one node of Cori.

9.2 Scaling Experiments

In this section the weak and strong scaling behavior of our implementation is presented an analyzed.

9.2.1 Experimental Setup

To study the weak and strong scaling behavior of our implementation the disorder induced heating experiment has been modified. The particle density is kept fix throughout all simulations to $\rho_0 = 6.67 \times 10^{12} cm^{-3}$, in order to simulate a realistic particle density. The particles are uniformly distributed in a cubic box of length $L_x \times L_y \times L_z$ with periodic boundary conditions in

all three dimensions. Charge and mass per particle correspond to real electrons and an initial uniform velocity distribution within $\pm 2.85 \times 10^{-20}$eV ensures that particles extensively cross the periodic boundaries. The cutoff radius is fixed to $r_{\text{cut}} = 2.5\,\mu\text{m}$ corresponding to a doubled mesh width of $h = 1.25\,\mu\text{m}$ which is also kept fix.

9.2.2 Weak Scaling

For the weak scaling experiment the number of particles per processing unit (PU) as well as the number of grid cells per PU is kept constant. We use $N_0 = 6670$ particles per PU and a constant mesh width of $h = 1.25\,\mu\text{m}$ which corresponds to 13 particles per grid cell. Beginning with $L_x = L_y = L_z = 0.001\,\text{cm}$ the computational domain is doubled in one dimension whilst the number of PUs is doubled as well. The timestep $\Delta t = 1\,\text{ps}$ is kept fix over all simulation runs. Figure 9.2 shows the parallel efficiency $\frac{t_1}{t_N}$ with the subscripts denoting the number of PUs used. The simulation is averaged over 100 time steps and PM and PP efficiency is plotted separately. For the PM part of the solver we do not expect a perfect scaling behavior since the dominant part is the FFT solver which suffers from communication overhead for more PUs involved in the computation. For the PP part however there is no obvious reason for a not close to optimal weak scaling behavior. The observed decreasing in computational efficiency indicates load balancing issues or an unintended non local behavior of parallel particle particle interaction computations. This issue must be examined in further studies. Going from 32 to 64 PUs we observe another decrease in the efficiency. This can be explained by slower communication as soon as a second compute node gets involved.

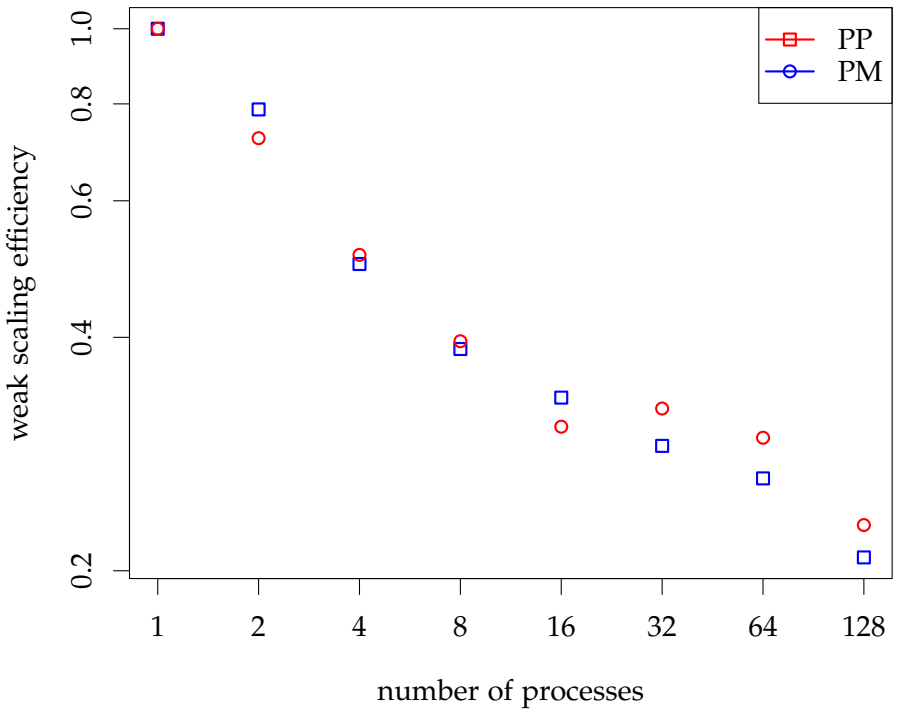

Figure 9.2: Weak scaling efficiency averaged over 100 time steps in log-log plot. The number of particles per PU is 6670 and the number of grid cells per PU is 512.

9.2.3 Strong Scaling

For the strong scaling experiment the number of particles is fixed to $N_0 = 853760$ on a computational domain $L_x = L_y = 0.004\,\text{cm}$ and $L_z = 0.008\,\text{cm}$. Figure 9.3 shows the time spent for 100 timesteps for a constant problem size and increasing number of MPI processes. In analogy to the weak scaling experiment that additional effort has to be spend on the parallelization overhead of the PP part. The black triangles show the timing of the total simulation indicating that the PP part is the dominating factor. However one has to take into account the relatively high cutoff radius used which influence the performance cubically as shown earlier. Even with the not ideal strong scaling behavior the overall execution time for the simulation can be reduced from 48 minutes to 2 minutes on 4 nodes compared to a single core execution. We observe a speedup of $7x$ going from 1 to 16 cores which is half the speedup obtained by miniMD, a lightweight molecular dynamics code

for performance studies [31]. Since the strong scaling behavior is much better than the weak scaling this indicates a hidden dependency on non local particles within the PP part.

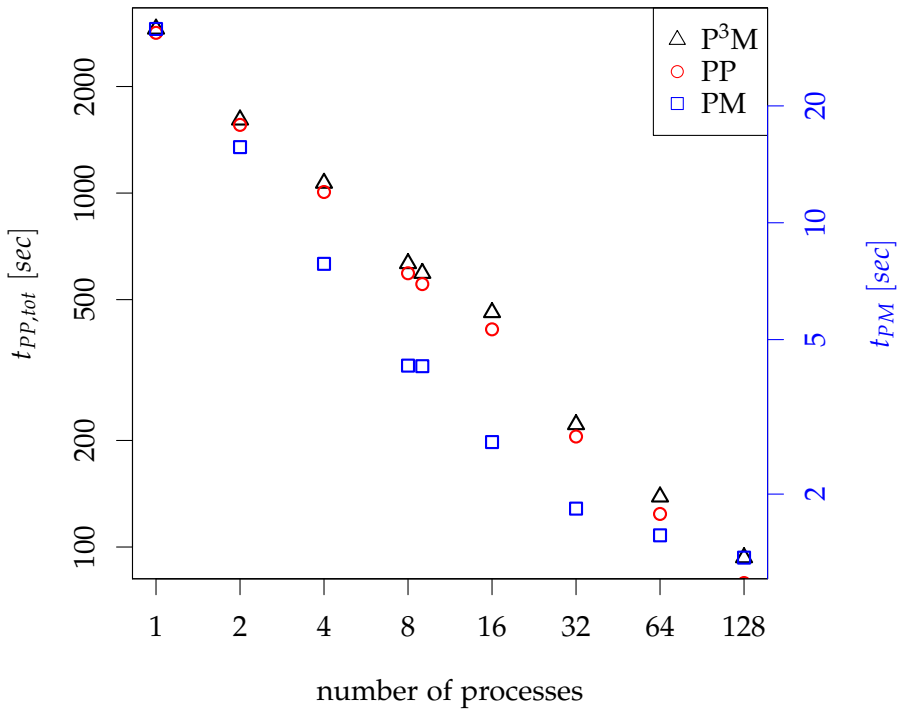

Figure 9.3: Strong scaling timings for 100 time steps in semilogarithmic plot. The number of particles is 853760, the computational mesh is of size $\mathcal{M}_{PM} = 32 \times 32 \times 64$ and the cutoff radius is $r_c = 2h = 2.5\,\mu\text{m}$.

9.2.4 Conclusion

This chapter was not meant to show an optimized performance study but its goal was to show the bottlenecks and potential of our P^{3}M implementation. No effort was made towards assisting the compiler with optimization flags or shared memory parallelization. Keeping that in mind the speedup of $7x$ going from one core to 16 gave us the possibility to perform all the simulations required for this thesis in reasonable time. The potential of our implementation to profit from shared memory parallelization on future manycore architectures will be outlined in the next chapter.

Chapter 10

Many Core Implementation

In the previous chapters we have shown that our P^3M implementation is suitable to study the effect of short range interactions between beam particles. Since short range interactions can influence the beam in several parts of the accelerator, a start-to-end simulation including the P^3M solver is desirable. The OPAL [5] library provides the required framework to track the beam through every single element of an accelerator. By its design the presented code is easily integrable into OPAL. However, since the P^3M algorithm is computationally demanding a start-to-end simulation can only be performed in reasonable time if efficient parallelization techniques are applied. With the end of Moore's law for single processing units multi- and many core architectures are used to accelerate scientific codes. Two promising hardware solutions are general purpose graphics processing units (GPGPUs) and Intel's manycore architecture XeonPhi. The next-generation supercomputer 'Cori' at the National Energy Research Scientific Computing (NERSC) Center will be equipped with the latest generation of Intel's XeonPhi processor named Knights Landing (KNL). Besides its 512 bit wide vector units and 60+ cache coherent cores, the heavier weight cores of the KNL chip can be used in a more flexible way than the extremely lightweight GPU cores. Hence, KNL is an up-and-coming hardware solution for the various computationally expensive tasks within a particle accelerator simulation. Another advantage of the XeonPhi is its better ratio between single precision and double precision floating point operations compared to GPUs which are in general optimized for 32bit floating point operations used in graphics and gaming applications. High precision computations are often needed in particle accelerator codes; making this feature highly appreciated. In this chapter our implementation is analyzed towards optimization for Intel's manycore (MIC) architecture.

10.1 The Point of Departure

The present implementation is based on a domain decomposition scheme using the message passing interface (MPI) to distribute the particles and meshes over the processes. Currently, no use of shared memory parallelism has been made. However, due to the modular design of the implementation, the code can easily be extended to make use of shared memory parallelism. In [32] the performance of the fast Fourier transformation (FFT) on the XeonPhi was studied with the result that with the KNL, Intel will also improve on their FFT implementation in their Math Kernel Library (MKL). Significant speedup has been promised. Under this assumption the particle-mesh part of the algorithm will gain some speedup from KNL for free. Intel MKL also provides a cluster version of the FFT implementation which is optimized for the use of multiple XeonPhi. Besides that improving on data locality can give an additional speed up for the charge deposition computations. As shown in chapter 9 the bottleneck of P^3M is the particle particle part of the algorithm which can strongly profit from shared memory parallelization and vectorization. In addition to the two field solvers the particle push to integrate the equations of motion in time can naturally profit from single instruction multiple data (SIMD) optimizations due to its locality properties.

10.2 Improving on Particle Mesh Performance

Besides the field solver which is heavily dependent on the FFT performance, the applied charge deposition scheme influences the overall performance of the PM part of P^3M. For the parallel performance of the charge deposition the data layout plays an important role. By its design, based on the Ippl framework, our code assigns the particles to different MPI processes based on the particles spatial positions. After every computation influencing the particle positions, the code reassigns particles to other processes if necessary. The particles local to each process can be stored as an Array-of-Structures (AoS) or a Structure-of-Arrays (SoA). The present implementation uses a mix of both strategies by applying the SoA scheme on the particles attributes like charge, mass and position, where the position attribute itself is represented as a structure. The data layout can easily be changed in order to optimize locality on the targeted hardware. The computational grid needed for the field computations is distributed over the processes in the same manner as the particles are distributed. Each process only stores the field data belonging to its assigned spatial part of the computational domain. In the present distributed memory parallelization based on this spatial decomposition, charge deposition is local to each process. Additional care has to be taken to treat the domain boundaries correctly. This is achieved by exchanging ghost cell information after the local charge decomposition is done. If however such a

spatial part of the computational domain is assigned to a XeonPhi processor with hundreds of threads, it is obvious to make additional use of shared memory parallelism. The challenge here is that race conditions occur as soon as two threads write to the same grid cell. Since the interpolation stencil is finite, e.g. in the cloud in cell approximation charges are only assigned to the 27 nearest neighbor cells, we can traverse the particles 27 times in checkerboard order. Each traversal can be done in parallel without any race conditions. In order to include shared memory charge deposition functionality in our code, we only have to add the checkerboard fashioned charge decomposition to the present scattering method. Further improvements can be made on cache locality by processing the particles cell by cell. This allows caching of the surrounding field data of the 27 neighbor cells for the force scattering back to the particles. In a similar way one can also store the charge density created by particles sharing a cell in a small local array before accumulating it to the global charge density field. The use of the here described techniques was recently studied in [33] and a speedp of 6.21x was obtained on the previous generation of XeonPhi named Knights Corner (KNC). The authors also showed that vectorization of the charge decomposition is more challenging [33, 34] but can give an additional speedup of roughly 1.3x.

10.3 Improving on Particle Particle Performance

Molecular dynamics codes, to solve the N-body problem are very popular and a lot of ongoing research frequently improves on the performance of these codes on multi- and many core implementations. The scientific interest in this problem becomes apparent by the fact that 5 chapters of the 'High Performance Parallelism Pearls' books [35, 36] are related to direct N-body problems. In this section we highlight the most promising approaches towards shared memory and SIMD parallelization. Shared Memory and SIMD parallelization profits from the spatial locality of close particle interactions. The already existing chaining mesh method to limit the search space for neighbor particles can be extended in order to improve on data locality. The idea is to use the chaining mesh to build up neighbor lists for each particle instead of computing the direct sum for particle interactions by directly traversing the chaining mesh. This has the great advantage that the force computation can be vectorized more easily and the neighbor particles processed in the direct sum can be found in consecutive cache lines. The neighbor list building process can profit from SIMD vectorization. However automatic vectorization comes with the implicit conversion from an AoS data layout used for the particle positions to SoA by loading them into vector units. Pennycook et al. showed in [31] that these gather operations are a bottleneck. For the force computation loop we face a similar problem with an additional scatter operation which converts the SoA obtained

by the SIMD operations back into an AoS. The authors were able to obtain a speedup of 1.5x for the force calculation by using dedicated scatter and gather operations on KNL instead of auto vectorization. With a manual vectorization they could get another 10% speedup. Making use of the previously described improvements and some additional effort on hierarchical domain decomposition from nodes over sockets to threads for a 2.048M particle simulation a speedup of roughly 6x has been achieved on KNC [31]. The lessons learned from this study is that a shared memory parallelization and SIMD vectorization of the PP part of P^3M is a promising candidate for significant performance increase. It is worth to mention that the results in [31] have been obtained using the offload model for the KNC co-processor. With KNL, only native mode will be available and we can hope for an additional speedup. Furthermore, KNL comes with the AVX512 instruction set supporting wide vectors and improved gather scatter operations which will be benefitial, too. Another aspect of shared memory parallelization for the direct force computation are race conditions due to shared resources. In general, race conditions are avoided by using memory locking mechanisms or thread local memory spaces. In [36, chapter 6] it was shown that locks should be avoided whenever possible. The hierarchical space decomposition proved to be beneficial again. Another technique to improve on data locality is studied in [35, chapter 9]: the direct force computation loop can improve on cache locality by tiling the loop according to the particles position. The loop over the particles i and their neighbors j is tiled in squared tiles of particles. This will improve on data locality when the sum over the i, j loop is parallelized using openMP. Besides that and other optimizations regarding data alignment for vectorization, the authors claim that they obtain 89% of the upper performance on KNC.

10.4 Conclusion

In this chapter we briefly sketched various approaches to increase the efficiency of our P^3M code on a many core architecture. The literature shows that there is huge potential in shared memory parallelization of the PM as well as the PP part of P^3M. The modular design of our code based on a spatial domain decomposition is well suited for the presented parallelization strategies and can be extended easily. It is worth to mention that all the optimizations presented in this chapter have been performed using the offload model to program the co-processor. This will change with KNL and we can make direct use of our particle and field distribution mechanisms using MPI. The shared memory and SIMD parallelization will then be added as additional hierarchical layers. Since both the PP as well as the PM part of our code are of major scientific relevance, often known as PIC codes or Molecular Dynamics respectively, it is to be expected that further optimiza-

tion of these techniques in terms of soft- and hardware acceleration will be studied in the future.

Chapter 11

Conclusions and Future Work

11.1 Conclusions

In the present thesis the particle-particle, particle-mesh algorithm P^3M was studied with focus towards its applicability on beam dynamics simulations in the context of particle accelerator modeling. We showed the benefits of the P^3M method on improving the resolution of the particle in cell method used for space-charge simulations. As an example of beam dynamics simulations, two common beam scenarios based on space-charge effects have been studied. First, we studied the applicability of the P^3M method to the problem of microbunching instabilities in high relativistic electron beams as an alternative to a computationally demanding molecular dynamics simulation proposed in [8]. We showed that the influence of particle collisions to the formation of microbunching is negligible and a rather small resolution in the particle mesh solver already shows the formation of microbunching. Second, we successfully studied the influence of particle collisions in a low energy beam with noise in the particle spatial distribution, a common state for the particle beam after the electron gun, using our P^3M implementation. We showed that the P^3M method accurately reproduces the physics predicted by the analytical theory. These results provide a basis for future cold beam simulations which are sensitive to Coulomb collisions. An example for such an application could be the planned ultra fast electron diffraction experiment [30] at LBL.
Our P^3M implementation can easily be integrated into a framework for general particle accelerator simulations such as OPAL [5] and optimized for shared memory parallelism. Starting points for possible optimizations for Intel manycore architectures have been given. With the integration of P^3M into OPAL we can perform start-to-end particle accelerator simulations sensitive to Coulomb collisions. The amount of collisions taken into account can be adjusted by a single parameter. Turning off the direct particle interaction in our implementation leaves us with the well known particle in cell

code without significant overhead. This offers the possibility of performing space-charge simulations throughout the whole particle accelerator, using the same code. The influence of Coulomb collisions for every part of the machine can be adjusted according to the requirements given by physical theory or experimental data. To our knowledge this is a novelty for general particle accelerator simulation frameworks and paves the way for start-to-end simulations of new machines with dynamically adjustable sensitivity to Coulomb collisions.

11.2 Future Work

Microbunching. As shown in chapter 7 the required resolution for the simulation of microbunching phenomena observed in our simulations is not as high as claimed in [8]. Furthermore, the influence of Coulomb collisions to the evolution of microbunching has to be studied in greater detail. Here an indepth-discussion with the authors of [8] is desirable in order to figure out the differences of their simulations and our P^3M simulation and to determine possible future steps. A simulation of a real beam without periodic boundary conditions would be an interesting milestone to achieve.

Ultrafast Electron Diffraction. The *disorder induced heating* experiment presented in chapter 8 already uses the beam parameters from the planned *ultrafast electron diffraction* experiment planned at LBL [30]. The new experiment is planned to use the Advanced Photon Injector Experiment (APEX) for electron diffraction studies. Here our P^3M could can be applied for start-to-end simulations under consideration of Coulomb collisions which are of major importance for this cold type of electron beams as shwon in chapter 8.

SwissFEL. For the newest accelerator at Paul Scherrer Institute (PSI), the *Swiss Free Electron Laser* (SwissFEL) our P^3M code can be applied to simulate the particle beam from the electron gun to the target with adjustable influence of Coulomb interactions.

Code Optimization. As shown in chapter 9 and 10 work has to be done to optimize the P^3M for shared memory parallelism and Cori.

Acknowledgments

I am very grateful for the opportunity of working in the inspirational environment of the Lawrence Berkeley Laboratory (LBL). I would also like to thank Andreas Adelmann for initiating the collaboration with LBL and the countless support and video chats during the thesis work. For the helpful discussions on site regarding numerical theory and computational aspects I would like to acknowledge John Shalf, Phil Colella and Brian van Straalen. My special thanks goes to Marco Venturini for his extraordinary interest in our work and his numerous explanations and suggestions based on his experience in particle accelerator physics and modelling.

Appendix A

Approximation of Laplace Operator in Real Space

We want solve three dimensional Poisson equation discretized on a $L \times M \times N$ grid. In order to improve readability we will assume that the number of discretization points in each dimension are equal and hence $L = M = N$. Poisson's equation can then be written as

$$\nabla^2 \Phi_{n_1,n_2,n_3} = -\rho_{n_1,n_2,n_3}, \quad n_1, n_2, n_3 \in [0, N-1], \tag{A.1}$$

with electric potential denoted by Φ and charge density ρ. The forward discrete Fourier transform of the charge density and electric potential can be written as

$$\hat{\Phi}_{k_1,k_2,k_3} = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \sum_{n_3=0}^{N-1} e^{\frac{-2\Pi i}{N} k_1 n_1} e^{\frac{-2\Pi i}{N} k_2 n_2} e^{\frac{-2\Pi k}{N} k_3 n_3} \Phi_{n_1,n_2,n_3}, \quad k_1, k_2, k_3 \in \mathbb{Z} \tag{A.2}$$

$$\hat{\rho}_{k_1,k_2,k_3} = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \sum_{n_3=0}^{N-1} e^{\frac{-2\Pi i}{N} k_1 n_1} e^{\frac{-2\Pi i}{N} k_2 n_2} e^{\frac{-2\Pi k}{N} k_3 n_3} \rho_{n_1,n_2,n_3}, \quad k_1, k_2, k_3 \in \mathbb{Z}\,. \tag{A.3}$$

To simplify this expression we introduce

$$\omega = e^{-\frac{2\Pi i}{N}}\,. \tag{A.4}$$

The Fourier transformed quantities $\hat{\Phi}$ and $\hat{\rho}$ can now be written as

$$\hat{\Phi}_{k_1,k_2,k_3} = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \sum_{n_3=0}^{N-1} \omega^{k_1 n_1} \omega^{k_2 n_2} \omega^{k_3 n_3} \Phi_{n_1,n_2,n_3}, \quad k_1, k_2, k_3 \in \mathbb{Z} \tag{A.5}$$

$$\hat{\rho}_{k_1,k_2,k_3} = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} \sum_{n_3=0}^{N-1} \omega^{k_1 n_1} \omega^{k_2 n_2} \omega^{k_3 n_3} \rho_{n_1,n_2,n_3}, \quad k_1, k_2, k_3 \in \mathbb{Z}\,. \tag{A.6}$$

The inverse Fourier transform reads

$$\Phi_{k_1,k_2,k_3} = \frac{1}{N^3} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \sum_{k_3=0}^{N-1} \omega^{-k_1 n_1} \omega^{-k_2 n_2} \omega^{-k_3 n_2} \hat{\Phi}_{k_1,k_2,k_3}, \quad k_1, k_2, k_3 \in \mathbb{Z} \tag{A.7}$$

$$\rho_{k_1,k_2,k_3} = \frac{1}{N^3} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \sum_{k_4=0}^{N-1} \omega^{-k_1 n_1} \omega^{-k_2 n_2} \omega^{-k_3 n_3} \hat{\rho}_{k_1,k_2,k_3}, \quad k_1, k_2, k_3 \in \mathbb{Z} \, . \tag{A.8}$$

If we discretize equation (A.1) on a $N \times N \times N$ mesh with meshwidth h, we can approximate the Laplace operator ∇^2 in real space by a finite difference approach leading to

$$\begin{aligned} \nabla^2 \Phi_{n_1,n_2,n_3} = \frac{1}{h^2} \, (& \Phi_{n_1+1,n_2,n_2} + \Phi_{n_1-1,n_2,n_3} + \Phi_{n_1,n_2+1,n_3} + \Phi_{n_1,n_2-1,n_3} + \\ & \Phi_{n_1,n_2,n_3+1} + \Phi_{n_1,n_2,n_3-1} - 6\Phi_{n_1,n_2,n_3}) \end{aligned} \tag{A.9}$$

If we plug in equation (A.5) into the right hand side of equation (A.9) we find

$$\begin{aligned} \nabla^2 \Phi_{n_1,n_2,n_3} = \frac{1}{h^2} \frac{1}{N^3} \Bigg(& \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \sum_{k_3=0}^{N-1} \omega^{-k_1 (n_1+1)} \omega^{-k_2 n_2} \omega^{-k_3 n_2} + \\ & \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \sum_{k_3=0}^{N-1} \omega^{-k_1 (n_1-1)} \omega^{-k_2 n_2} \omega^{-k_3 n_2} \ldots \\ & -6 \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \sum_{k_3=0}^{N-1} \omega^{-k_1 n_1} \omega^{-k_2 n_2} \omega^{-k_3 n_2} \Bigg) \, . \end{aligned} \tag{A.10}$$

If we plug in equation (A.10) and equation (A.8) into equation (A.1) we can write Poisson's equation in Fourier space as

$$\begin{aligned} & \frac{1}{h^2} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \sum_{k_3=0}^{N-1} \left[\underbrace{\left(\omega^{-k_1 n_1} \omega^{-k_2 n_2} \omega^{-k_3 n_3} \right)}_{:=A} \left(\omega^{-k_1} + \omega^{+k_1} + \omega^{-k_2} + \omega^{+k_2} + \omega^{-k_3} + \omega^{+k_3} - 6 \right) \hat{\Phi}_{k_1,k_2,k_3} \right] \\ & = - \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \sum_{k_3=0}^{N-1} \left[\underbrace{\left(\omega^{-k_1 n_1} \omega^{-k_2 n_2} \omega^{-k_3 n_3} \right)}_{:=A} \hat{\rho}_{k_1,k_2,k_3} \right] . \end{aligned} \tag{A.11}$$

After equating the coefficients of the quantity A appearing in the sums on the left and right hand side of equation (A.11) we find

$$\frac{1}{h^2} \left(\omega^{-k_1} + \omega^{+k_1} + \omega^{-k_2} + \omega^{+k_2} + \omega^{-k_3} + \omega^{+k_3} - 6 \right) \hat{\Phi}_{k_1,k_2,k_3} = -\hat{\rho}_{k_1,k_2,k_3}, \tag{A.12}$$

which leads to the well known Green's function solution in Fourier space for Poisson's equation (A.1)

$$\hat{\Phi}_{k_1,k_2,k_3} = \frac{h^2}{6 - \omega^{-k_1} - \omega^{+k_1} - \omega^{-k_2} - \omega^{+k_2} - \omega^{-k_3} - \omega^{+k_3}} \hat{\rho}_{k_1,k_2,k_3} \tag{A.13}$$

$$= \frac{h^2}{6 - \underbrace{\left(e^{\frac{2\Pi i}{N}k_1} + e^{\frac{-2\Pi i}{N}k_1}\right)}_{2\cos\left(\frac{2\Pi}{N}k_1\right)} - 2\cos\left(\frac{2\Pi}{N}k_2\right) - 2\cos\left(\frac{2\Pi}{N}k_3\right)} \hat{\rho}_{k_1,k_2,k_3} \tag{A.14}$$

$$= \frac{0.5h^2}{3 - \cos\left(\frac{2\Pi}{N}k_1\right) - \cos\left(\frac{2\Pi}{N}k_2\right) - \cos\left(\frac{2\Pi}{N}k_3\right)} \hat{\rho}_{k_1,k_2,k_3} \, . \tag{A.15}$$

With Fourier space, discretized Green's function

$$\hat{G}_{k_1,k_2,k_3} = \frac{0.5h^2}{3 - \cos\left(\frac{2\Pi}{N}k_1\right) - \cos\left(\frac{2\Pi}{N}k_2\right) - \cos\left(\frac{2\Pi}{N}k_3\right)} \tag{A.16}$$

we can write

$$\hat{\Phi} = \hat{G}\hat{\rho} \, . \tag{A.17}$$

Appendix B

Execution of Regression Tests

B.1 Microbunching

In this section we provide all the relevant information to reproduce the microbunching simulations presented in chapter 7.

B.1.1 Run the Simulation

In order to run a microbunching simulation execute `p3m3dMicrobunching` with the following input parameters

`# mpirun -np <num threads> ./p3m3dMicrobunching <`N_x`> <`N_y`> <`N_z`> <`r_c`> <`α`> <`ε`> <num steps> <seed> <print every>`

and the parameters presented in Table B.1. Example values for a possible test run are provided.

parameter	meaning	example value
num threads	number of MPI threads	8
N_x, N_y, N_z	number of grid points per dimension	$64, 64, 64$
r_c	cutoff radius for PP interactions	0
α	Green's function splitting parameter	40000
ε	regularization for PP interaction	0
num steps	number of time-steps	100
seed	integer seed value between 0 and 49*	0
print every	prints output every $n-$th time-step	10

Table B.1: Meaning of input parameters and example values.

(*) we provide a sequence of 50 integers used as seed values from the whole 32-bit range according to http://en.cppreference.com/w/cpp/ numeric/random/seed_seq in order to avoid poorly distributed initial seeds while providing reproducibility.

Make sure that an empty `./data` directory is available in the directory you are running the example as well as a `./BeamParams.in` file specifying the beam parameters. An example input file is shown in Listing B.1.

BeamParams.in

```
270                 //gamma                     [1]
0.001956951         //energy spread             [1]
40                  //current I                 [A]
3e-6                //longitudinal length       [m]
320e-6              //transverse length         [m]
4                   //drift length              [m]
85e-6               //rms envelope size         [m]
0.1e-6              //emittance                 [m rad]
1e-3                //R56                       [m]
1                   //charge per particle       [e]
0.5109989           //particle rest mass m0     [MeV/c^2]
1.439964e-15        //coulomb constant ke       [m MeV/e^2]
299792458           //speed of light            [m/s]
```

Listing B.1: Input file to specify beam parameters.

B.1.2 Postprocessing

Beam Statistics. After successful execution of the program the programs output can be found in `./data`. For every seed value there will exist a `BeamStatistics\_seedID\_xx.csv` file providing the second order moments and emittances for every time-step. In order to remove the brackets contained in the file (will be removed in future releases) the following command comes handy,

```
# cat BeamSTatistics_nod_00.csv | tr -d '()' > Moments.csv .
```

A plotting tool like R or gnuplot can be used to plot the comma separted values in the resulting `Moments.csv` file.

Particle Data. Additionally you will find the particle data for every time-step specified by the 'print every' parameter in the h5part files `particleData\_seedID\_xx.h5part`. you will find the particle coordinates, specified by the parameter names 'x,y,z'; the particles momenta in the beam frame, specified by 'px,py,pz'; the particles longitudinal momentum in the lab frame 'pz_lab' and the particles longitudinal position in the lab frame after application of the dispersion denoted by 'zR56'.
The graphics in 7 have been created using ParaView. Any other software capable of reading hdf5 files, such as ROOT or Visit can be used as well.

B.2 Disorder Induced Heating

In this section we provide all the relevant information to reproduce the disorder induced heating simulations presented in chapter 8.

B.2.1 Run the Simulation

In order to run a disorder induced heating simulation execute `p3m3dHeating` with the following input parameters

`# mpirun -np <num threads> ./p3m3dHeating <`N_x`> <`N_y`> <`N_z`> <beam radius> <boxlength> <num particles> <`r_c`> <`α`> <dt> <`ε`> <num steps> <`m_e`> <`q_e`> <focus strength> <print every>`

and the parameters presented in Table B.2. Example values for a possible test run are provided.

parameter	meaning	example value
num threads	number of MPI threads	32
N_x, N_y, N_z	number of grid points per dimension	$256, 256, 256$
beam radius	radius of the beam	0.001774
boxlength	cubical domain extension in all 3 dimensions	0.01
num particles	number of particles	156055
r_c	cutoff radius for PP interactions	0.0003125
α	Green's function splitting parameter	6400
dt	time-step	2.15623e–13
ε	regularization for PP interaction	0
num steps	number of time-steps	100
m_e	electron mass	1
q_e	electron charge	1
focus strength	factor to scale the focusing strength	1.5
print every	prints output every $n-$th time-step	10

Table B.2: Meaning of input parameters and example values.

B.2.2 Postprocessing

The relevant output files written to the `data` directory are `BeamStatistics.csv` containing the second order moments and emittances for every time-step and the `particleData.h5part` file containing the particle attributes. Hints on evaluating the collected data can be found in section B.1.2.

B.3 Regression Tests

B.3.1 Run the Tests

All regression tests presented in chapter 5 can be executed using the same executable `p3m3dRegressiontests`. To run a regression test execute the program as follows,

`# mpirun -np <num threads> ./p3m3dRegressionTests <testcase> <`N_x`> <`N_y`> <`N_z`> <`r_c`> <`α`> <dt> <`ε`> <num steps> <`S_x`> <`S_y`> `S_z

and the parameters presented in Table B.3.

parameter	meaning	example value
num threads	number of MPI threads	1
testcase	choose a testcase from the list (sphere, recurrence, landau, twostream)	sphere
N_x, N_y, N_z	number of grid points per dimension	$64, 64, 64$
r_c	cutoff radius for PP interactions	1.0
α	Green's function splitting parameter	1.5
dt	time-step	0.01
ε **	regularization for PP interaction	0
num steps	number of time-steps	100
S_x, S_y, S_z	center of charged sphere*	$0, 0, 0$

Table B.3: Meaning of input parameters and example values. *Center of charged sphere is only required for the sphere experiment. **For the PM only tests like Landau damping, twostream instability and the free stream experiment, the parameter ε is used for regularization of the PM Green's function at the origin.

B.3.2 Postprocessing

For all regression tests you can find in the `data` directory the electric field on the grid stored in a vtk file for every iteration `EFieldAfterPMandPP\_nod\_0\_it\_\#.vtk`. The kinetic and potential energies as well as the electric field energy for every time-step are stored in `energy\_nod\_\#.csv`. The particle data per MPI process for every iteration is given in `charges\_nod\_\#\_it\_\#.csv`.

Sphere Tests. For the tests involving the uniformly charged sphere additionally the quantities supposed to be conserved (see Table 5.1) are stored in `conservedQuantities.csv` and the `statistics.txt` provides various error calculations along with the relevant parameters. This file gets extended by every simulation run in order to study the error dependence on different input parameters.

Twostream Instability. For the twostrem instability an additional file `f\_mesh\_\#.csv` provides the charge density data for the two dimensional phase space in longitudinal direction.

B.3.3 Reproduction of Figures:

Figures of section 5.1.1. To reproduce the static sphere experiments run

```
# mpirun -np 1 ./p3m3dRegressionTests sphere 256 256 256 0 10000 0 1e-10 1 0 0 0
```

with Mathematica notebook `ChargedSPhereCompareSimul.nb`.

Figures of section 5.1.2. To reproduce the moving sphere experiments run

```
# mpirun -np 4 ./p3m3dRegressionTests sphere 64 64 64 0.75 1 0.5 1e-10 40 0 0 0
```

and vary the number of cores. For the 32 core case we had to increase the computational domain and executed as follows

```
# srun -n 32 ./p3m3dRegressionTests sphere 128 128 128 0.75 1 0.5 1e-10 40 0 0 0
```

Figures of section 5.1.4 To reproduce the free stream experiments run

```
# mpirun -np 1 ./p3m3dRegressionTests recurrence 8 8 8 0 77777 0.25 1e-10 160 0
0 0
```

Figures of section 5.1.5 To reproduce the Landau damping experiments run

```
# mpirun -np 1 ./p3m3dRegressionTests landau 8 8 8 0 7777777 0.25 1e-10 160 3.5
3.6 3.7
```

Figures of section 5.2.3 To reproduce the plots showing the error dependencies on the splitting parameter α one can use the the bash script presented in Listing B.1. The obtained numerical errors of interest and the used relevant parameters are stored in `statistics.txt`.

```
#!/bin/bash
echo "start alpha dependence experiment"
NP=1
N=32
dt=0
eps=0.000001
Sx=0
Sy=0
Sz=0
it=1
for rc in 1.5
do
  for alpha in 0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 3.5 4.0
     4.5 5.0 10.0 20.0
  do
    mpirun -np ${NP} ./p3m3dRegressionTests sphere ${N} ${N} ${N} ${rc} ${alpha} ${dt}
     ${eps} ${it} ${Sx} ${Sy} ${Sz}
  done
done
echo "experiment done!"
```

Listing B.1: Script to run the α-dependence tests.

Figures of section 5.1.6 To reproduce the twostream instability simulation run

```
# srun -n 1 ./p3m3dRegressionTests twostream 16 16 16 0 7777777 0.125 0.0000000001
400 0 0 0
```

Appendix C

List of Mathematica Scripts

During the Thesis work several Mathematica scripts emerged in order to produce and reproduce graphics and to perform analytical computations. In this chapter the most relevant Mathematica notebooks are listed including a short description of the functionality.

C.1 Charged Sphere

In `ChargedSphereCompareSimul.nb` the analytic solution for the electric field and potential of a uniformly charged sphere is computed. This script was also used to read in the obtained values from the numerical simulations in order to compare with the analytic solution. The corresponding experiment is described in section 5.1.1.

C.2 Green's Function Splitting

The different options of Green's function splitting as described in chapter 2 and the influence of a softening parameter can be studied and visualized with the Mathematica notebook `InteractionSplittingSoftening.nb`.

C.3 Analytic Formula of the Microbunching Gain

The main purpose of `AnalyticFormulaMBGainVariousEmittance.nb` is the visualization of formula (6.26). Different emittance values can be studied and formula (6.26) is subdivided into small pieces which can be studied separately. The influence of the made approximations can be made visible. In a separate Mathematica notebook `RootsOfDispEq.nb` the numerical roots for the plasma dielectric function (6.11) are computed.

Bibliography

1. Pedroni, E. *et al.* The 200-MeV proton therapy project at the Paul Scherrer Institute: Conceptual design and practical realization. *Medical physics* **22,** 37–53 (1995).
2. Hockney, R. W. & Eastwood, J. W. *Computer simulation using particles* doi:10.1002/phbl.19880441113 (CRC Press, 1988).
3. Brieu, P. P. & Evrard, A. E. P4M: a parallel version of P3M. *New Astronomy* **5,** 163–180 (2000).
4. MacFarland, T., Couchman, H., Pearce, F. & Pichlmeier, J. A new parallel P3M code for very large-scale cosmological simulations. *New Astronomy* **3,** 687–705 (1998).
5. Adelmann, A. *et al. The OPAL (Object Oriented Parallel Accelerator Library) Framework* tech. rep. PSI-PR-08-02 (Paul Scherrer Institut, 2008-2016).
6. Anthony, S. *Intel unveils 72-core x86 Knights Landing CPU for exascale supercomputing* 2013.
7. Saldin, E., Schneidmiller, E. & Yurkov, M. Longitudinal space charge-driven microbunching instability in the TESLA Test Facility linac. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **528,** 355–359 (2004).
8. Marinelli, A. & Rosenzweig, J. B. Microscopic kinetic analysis of space-charge induced optical microbunching in a relativistic electron beam. *Physical Review Special Topics-Accelerators and Beams* **13,** 110703 (2010).
9. Rapaport, D. C. *The art of molecular dynamics simulation* (Cambridge university press, 2004).
10. Qiang, J., Ryne, R. D., Habib, S. & Decyk, V. *An object-oriented parallel particle-in-cell code for beam dynamics simulation in linear accelerators* in *Proceedings of the 1999 ACM/IEEE conference on Supercomputing* (1999), 55.

11. Qiang, J., Ryne, R. D. & Habib, S. *Self-consistent Langevin simulation of Coulomb collisions in charged-particle beams* in *Proceedings of the 2000 ACM/IEEE conference on Supercomputing* (2000), 27.

12. Mitchell, C. & Qiang, J. *A Parallel Particle-Particle, Particle-Mesh Solver for Studying Coulomb Collisions in the Code IMPACT-T* in *Proceedings of IPAC2015* Richmond, VA, USA (2015).

13. Birdsall, C. K. & Langdon, A. B. *Plasma physics via computer simulation* (CRC Press, 2004).

14. Eastwood, J. & Brownrigg, D. Remarks on the solution of Poisson's equation for isolated systems. *Journal of Computational Physics* **32,** 24–38 (1979).

15. Hockney, R. Methods in Computational Physics. *Alder, B,* 136–211 (1970).

16. Hünenberger, P. H. Optimal charge-shaping functions for the particle–particle—particle–mesh (P3M) method for computing electrostatic interactions in molecular simulations. *The Journal of Chemical Physics* **113,** 10464–10476 (2000).

17. Ballenegger, V., Cerdà, J. J. & Holm, C. How to convert SPME to P3M: Influence functions and error estimates. *Journal of Chemical Theory and Computation* **8,** 936–947 (2012).

18. Adelmann, A. *The IPPL (Independent Parallel Particle Layer) Framework* tech. rep. PSI-PR-09-05 (Paul Scherrer Institut, 2009).

19. Griffiths, D. J. & College, R. *Introduction to electrodynamics* (prentice Hall Upper Saddle River, NJ, 1999).

20. Cheng, C.-Z. & Knorr, G. The integration of the Vlasov equation in configuration space. *Journal of Computational Physics* **22,** 330–351 (1976).

21. Kormann, K. A semi-Lagrangian Vlasov solver in tensor train format. *SIAM Journal on Scientific Computing* **37,** B613–B632 (2015).

22. Sonnendrücker, E. Approximation numérique des équations de Vlasov-Maxwell, 2010. notes de cours de M2. *Université de Strasbourg*.

23. Wang, B., Miller, G. H. & Colella, P. A particle-in-cell method with adaptive phase-space remapping for kinetic plasmas. *SIAM Journal on Scientific Computing* **33,** 3509–3537 (2011).

24. Myers, A., Colella, P. & Van Straalen, B. A 4th-Order Particle-in-Cell Method with Phase-Space Remapping for the Vlasov-Poisson Equation. *arXiv preprint arXiv:1602.00747* (2016).

25. Ratner, D., Huang, Z. & Chao, A. *Three-dimensional analysis of longitudinal space charge microbunching starting from shot noise* in *Presented at* (2008).

26. Di Mitri, S. Intrabeam scattering in high brightness electron linacs. *Physical Review Special Topics-Accelerators and Beams* **17,** 074401 (2014).
27. Marinelli, A., Rosenzweig, J. & Pham, A. *Molecular Dynamics Simulation of Longitudinal Space-Charge Induced Optical Microbunching* `https://accelconf.web.cern.ch/accelconf/FEL2009/talks/weob03_talk.pdf`. 2009.
28. Gericke, D. & Murillo, M. Disorder-induced heating of ultracold plasmas. *Contributions to Plasma Physics* **43,** 298–301 (2003).
29. Maxson, J., Bazarov, I., Wan, W., Padmore, H. & Coleman-Smith, C. Fundamental photoemission brightness limit from disorder induced heating. *New Journal of Physics* **15,** 103024 (2013).
30. Filippetto, D. *et al.* High repetition rate ultrafast electron diffraction at LBNL. *Proceedings of IPAC2014, Dresden, Germany,* 724 (2014).
31. Pennycook, S. J., Hughes, C. J., Smelyanskiy, M. & Jarvis, S. A. *Exploring SIMD for Molecular Dynamics, Using Intel® Xeon® Processors and Intel® Xeon Phi Coprocessors* in *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on* (2013), 1085–1097.
32. Ulmer, B. *Performance Aanalysis of MKL Fast Fourier Transform and FFT-based Poisson Solver on Intel Xeon Phi* `http://amas.web.psi.ch/people/aadelmann/ETH-Accel-Lecture-1/projectscompleted/`. 2015.
33. Surmin, I. *et al.* Particle-in-Cell Laser-Plasma Simulation on Xeon Phi Coprocessors. *arXiv preprint arXiv:1505.07271* (2015).
34. Nakashima, H. Manycore challenge in particle-in-cell simulation: How to exploit 1 TFlops peak performance for simulation codes with irregular computation. *Computers & Electrical Engineering* **46,** 81–94 (2015).
35. Reinders, J. & Jeffers, J. *High Performance Parallelism Pearls Volume One: Multicore and Many-core Programming Approaches* (Morgan Kaufmann, 2014).
36. Jeffers, J. & Reinders, J. *High Performance Parallelism Pearls Volume Two: Multicore and Many-core Programming Approaches* (Morgan Kaufmann, 2015).