Effects and Adaptation of Particle Noise in Large-Scale PIC Simulations

Nils Moschüring

München 2020

Effects and Adaptation of Particle Noise in Large-Scale PIC Simulations

Nils Moschüring

Dissertation
der Fakultät für Physik
der Ludwig-Maximilians-Universität
München

vorgelegt von
Nils Moschüring
aus Wesel

München, den 10. August 2020

Erstgutachter: Prof. Dr. H. Ruhl
Zweitgutachter: Prof. Dr. A. Caldwell
Tag der mündlichen Prüfung: 25. September 2020

Contents

Zusammenfassung

Die Particle-In-Cell (PIC) Technik findet in der Modellbildung einer großen Anzahl von wissenschaftlichen Problemen Anwendung. PIC Simulationen können die nicht-linearen Prozesse der Laser-Plasma-Interaktion abbilden, und benötigen hierfür relativ wenig Rechenleistung. Ein gutes Verständnis dafür, wie verlässlich die Ergebnisse dieser Simulationen sind, und wie nah sie an experimentelle Resultate heran kommen, ist hierbei unerlässlich. Eine der wichtigsten Messgrößen hierfür ist das Teilchenrauschen. Durch die Existenz von diskreten Teilchen, in der Realität wie auch im PIC Modell, zeigen ein Großteil der Messgrößen ein gewisses Rauschen um einen Mittelwert herum. Die Rechenersparnis, die das PIC Modell auszeichnet, wird zu einem großen Teil dadurch erreicht, dass sogenannte Quasiteilchen eingesetzt werden. Diese Stellvertreterteilchen repräsentieren jeweils eine große Zahl von physikalischen Teilchen und reduzieren somit die Anzahl der nötigen mathematischen Operationen um mehrere Größenordnungen. Durch die stark reduzierte Anzahl von interagierenden Teilchen erhöht sich jedoch das Rauschen in den Messgrößen. Diese Dissertation untersucht mehrere Facetten dieses unphysikalischen Quasiteilchenrauschens.

Als erstes wird eine Theorie entwickelt, die die Auswirkungen der Quasiteilchen auf das Teilchenrauschen beschreibt. Zu den Ergebnissen dieser Theorie gehören Gleichungen die das Signal-Rausch-Verhältnis von PIC Simulationen für erste Ordnung Teilchen mit zwei verschiedenen Teilchengewichten beschreiben. Aus den Gleichungen folgt, dass die Anzahl der Quasiteilchen sowie ihre Gewichtsverteilung entscheidend für die Verlässlichkeit der Simulation sind.

Neben der Bedeutung für die Verlässlichkeit einer Simulation, ist die Anzahl der Quasiteilchen auch noch der wichtigste Faktor für ihren Rechenaufwand. Somit ist die dynamische Kontrolle und Anpassung der Quasiteilchenanzahl und deren Gewichte, auch während einer Simulation, von großem Interesse und Vorteil. Ein neuartiger Algorithmus mit der Fähigkeit Quasiteilchen zu splitten (zu zerteilen) und zu mergen (zu vereinigen) wurde entwickelt. Dieser Algorithmus garantiert eine größtmögliche Anzahl an Erhaltungssätzen in möglichst genereller Form. Hierbei ist hervorzuheben, dass keine unphysikalische Divergenz in der Simulation entsteht und durch die Nutzung von Initialmomenta auch die Impulsraumverteilung sehr gut erhalten bleibt. Dies wird mit ausführlichen Benchmarks und Vergleichen, auch mit einem Algorithmus anderer Wissenschaftler, verdeutlicht.

In der zweiten Hälfte werden zwei peer-reviewed Erstautorpublikationen von Nils Moschüring vorgestellt. Diese sind in Zusammenarbeit mit der AWAKE Kollaboration entstanden, die die Möglichkeit eines protonengetriebenen Kielfeld-Beschleunigers untersucht. Die erste Publikation stellt eine neuartige Methode vor, um kontrolliertes Quasiteilchenrauschen in PIC Simulationen zu generieren. Dies erlaubt die verbesserte Extrapolation von Simulationsergebnissen zu Experimenten, mit dem Ziel optimierter Parameter nicht nur für das AWAKE Experiment. Die zweite Publikation enthält die Auswertung von Simulationsergebnissen einer großskaligen PIC Simulationskampagne auf SuperMUC. Die finale Simulation verbrauchte ca. 22 Mch an Rechenleistung und lief auf 32768 Kernen. Sie wurde durch eine Vielzahl von technischen Entwicklungen und Anpassungen ermöglicht. Die Analyse der resultierenden Daten ergab interessante neuartige Effekte während der Teilcheninjektion. Der erfolgreiche Abschluss der Kampagne beweist, dass der PSC nun fähig ist großskalige Simulationen effizient auszuführen.

Introduction and abstract

Particle-In-Cell (PIC) codes are used on a large variety of scientific problems. Their ability to model non-linear processes, combined with a reduced computational load, in comparison to fluid codes, makes them a good tool to investigate all kinds of laser-plasma interaction. It is very important to properly understand the faithfulness of these simulations with regard to actual experiments.

One very important characteristic, in order to adjudicate this faithfulness, is the amount of exhibited particle noise. The computational savings, generated by PIC codes, is mainly due to the employment of so-called quasi-particles. These entities represent a large number of actual physical particles and are therefore able to reduce the amount of necessary computations by several orders of magnitude in many cases. A drawback is the, on average, meaningfully higher particle noise in PIC simulations when compared to experiments. The reduced number of interacting particles results in this increased noise. This thesis will investigate multiple facets of this unphysical quasi-particle noise.

For a brief introduction into the employed unit system, the finite-difference time-domain technique and the PIC method, please refer to chapter 2-4 of my diploma thesis [22].

In chapter 1 of this dissertation, a theory to model the impact of the number of quasi-particles on the particle noise is developed. This chapter gives formulas describing the signal-to-noise ratio of PIC simulations using zeroth-order quasi-particles with a single common weight, first-order quasi-particles with a single common weight and first-order quasi-particles with two different weights. A very general approach has been taken for these derivations and only a small amount of assumptions have been made. This leads to results that are very generally applicable. The two main conclusions from this model are that i) more quasi-particles lead to a higher signal-to-noise ratio and ii) employing particles with different weights leads to a lower signal-to-noise ratio. Therefore, the amount and the weight of the quasi-particles in a simulation are fundamental to the faithfulness of the results.

This conclusion leads directly to the work that is presented in chapter 2. Since the amount and the weight of the quasi-particles is so important for PIC simulations, algorithms to directly control these parameters during a running simulation are valuable. Additionally, the number of quasi-particles is the main driving factor for the computational cost of a simulation.

In order to reduce and increase the number of quasi-particles in a simulation, so-called quasi-particle merging and splitting is performed. During this process, quasi-particles are either merged into a lower number of bigger quasi-particles or split up into a higher number of smaller quasi-particles. This must be done very carefully in order to keep any newly introduced numerical errors to a minimum. It must also be done in a way that tries to respect as many conservation laws in their most general form as possible. A novel algorithm with splitting and merging capabilities has been developed during this thesis and is presented in chapter 2. The number of parameters of this algorithm is intentionally kept to a minimum in order to make a correct usage simpler and more likely.

First, multiple stages of sorting are performed, which are detailed in chapter 2.2. This ensures

that particles undergoing the merging/splitting process are similar to each other, which reduces the introduced errors. It also enables the possibility of selectively merging the smallest particles and splitting the biggest particles, which reduces the variety in weight space.

In the case of merging, the targeted particles then need to be gathered at shared spatial locations, without introducing spurious electromagnetic fields. A special algorithm that performs this movement has been developed. It is detailed in chapter 2.3.1. This algorithm works flawlessly for first-order particles with the effect that no unphysical divergence in the electric fields is generated. For second-order particles it is shown in chapter 2.3.1 that a deterministic flawless algorithm is impossible, as solutions to non-linear equations are required in that case. Splitting quasi-particles does not need any special treatment regarding the spatial location of the targeted quasi-particles.

In the last step, the momenta of the new quasi-particles are computed, using a sophisticated algorithm. This step is explained in chapter 2.3.2. In the case of merging, a localized redistribution of the total energy allows for a re-usage of initial momenta values. This special method is able to, on average, preserve the distribution in momentum space well, including outliers. Computing the momenta of the resulting quasi-particles after a splitting process is much simpler and more straightforward. Nonetheless, a novel technique is presented in chapter 2.4.2, which redistributes energies, and is able produce a more diverse momentum distribution.

Finally, chapter 2.6 describes an additional and optional adaptation step, which reduces the amount of different weights in the simulation, and chapter 2.7 gives the necessary modifications to the algorithm to enable its use on photons.

In order to verify the algorithm and ensure that it fulfills its stated goals, chapter 3 presents a host of benchmarks and various results of applications. To achieve this, the given algorithm was implemented in the PIC code PSC. Chapter 3.1 first shows the repercussions of merging/ splitting on the momentum space for an exhaustive list of parameters for the well-known two-stream problem. After that, a detailed comparison to a previously published algorithm [29] is given in chapter 3.2. This comparison shows that, in relation to the considered error metrics, the algorithm presented in this thesis is able to outperform the previously published algorithm.

This concludes the previously unpublished part of this dissertation. The subsequent chapters include two peer-reviewed publications with Nils Moschüring as the first author. Following the theoretical examination of particle noise and an algorithm to manipulate it, these chapters present an application of the gathered knowledge to simulations for an experiment. This experiment is called AWAKE (”Advanced Proton Driven Plasma Wakefield Acceleration Experiment“). It is an international collaboration that aims at investigating a novel particle acceleration scheme, employing proton-driven plasma-wakefields.

Chapter 4 provides an introduction and the full text of the first peer-reviewed publication. This publication is concerned with the generation of controllable quasi-particle noise for proton-driven plasma-wakefield PIC simulations. In order to achieve this controlled quasi-particle noise, it

investigates the difference between experimental shot-noise (particle noise of actual beam particles) and the quasi-particle noise of a PIC simulation. Finally, it proposes a simulation setup, which makes it possible to control the quasi-particle noise in simulations by employing a special noise generating assembly of quasi-particles. This controlled quasi-particle noise enables an improved extrapolation of simulation results to experiments. The main hindrance in achieving this proper setup was the emergence of numerical Cerenkov radiation. This hindrance was overcome by choosing sinusoidal shapes for the noise generating assembly of quasi-particles.

It was originally planned to use this novel setup scheme to find optimized experimental parameters by running a series of PIC simulations. Due to the complex nature of the experiment, these simulations are quite expensive, and the series of simulations has not been completed. Nonetheless, one large-scale simulation was finished successfully. The results of this run have shown interesting novel effects, which lead to a second peer-reviewed publication.

The final chapter of this thesis, chapter 5, provides an introduction and the full text of this second publication. In this publication the results of the large-scale PIC simulation are presented. The publication describes the complex setup, the special modules and optimizations to the PSC, the actual run of the simulation, and finally analyzes the resulting data. The simulation needed to run on 32768 cores due to its size, consuming about 22 Mch of computing resources, which made it a difficult technical challenge as well. Novel effects were found in the resulting data of the simulation and these are subsequently exposed through a series of elaborate figures and plots. These novel effects happen during the electron injection process, a vital process for AWAKE. The simulations decisively helped in furthering the understanding of this process.

This successful completion proves the capability of the PSC to perform large-scale simulations. Using the developed test case, it is now possible to finish the proposed series of simulations. It is also possible to further investigate the electron injection process. This process is highly nonlinear, three dimensional and is currently not well-understood. The PSC, equipped with the new test case and optimized for this large-scale simulation, is uniquely able to model that scenario, thanks to the work performed for this dissertation.

1 Merging/Splitting Theory

In order to evaluate the effectiveness of a possible merging or splitting algorithm, it is important to develop a theoretical understanding of the importance and the implications of the number of employed quasi-particles. The number of particles per cell determines many statistical properties of a PIC simulation. The process of measuring the particle density at a particular point (or any other physical property that is derived from the quasi-particle distribution), can be viewed as performing a number of measurements equal to the number of quasi-particles in the respective cell. The results of this chapter were presented in the Laser Experiment Theory Seminar at the Max-Planck-Institute for Quantum Optics in Garching in January 2014.

1.1 Zeroth-order form factor

The fraction of the total density in a specific cell at a specific time t is derived from the Vlasov equation to be

$$\frac{\int_{\text{cell}} f(\vec{x},t)d^3x}{\int_V f(\vec{x},t)d^3x}, \tag{1.1}$$

where the momentum integration of f is implicitly assumed. The distribution function is now sampled, using a finite number of quasi-particles with a zeroth-order particle shape. The probability of finding k of the total N quasi-particles in a specific cell is then given by

$$P_{k,\text{cell}} = \binom{N}{k} p_{\text{cell}}^k \left(1-p_{\text{cell}}\right)^{N-k}, \quad \text{where} \quad \binom{N}{k} \tag{1.2}$$

is the binomial coefficient of N and k, and

$$p_{\text{cell}}(t) = \frac{\int_{\text{cell}} f(\vec{x},t)d^3x}{\int_V f(\vec{x},t)d^3x}. \tag{1.3}$$

Equation (1.2) corresponds to a binomial distribution. The expectation value and the standard deviation of the particle number in a specific cell is therefore given by

$$\mu_{\text{ppc}} = N p_{\text{cell}} \quad \text{and} \tag{1.4}$$

$$\sigma_{\text{ppc}} = \sqrt{N p_{\text{cell}} \left(1-p_{\text{cell}}\right)}, \tag{1.5}$$

respectively. The subscript ppc stands for particles per cell. If the total charge is assumed to be equal to Q (i.e. one quasi-particle carries the charge Q/N), and the weight of each particle is equal to one, the average charge density and the standard deviation of the charge density in a cell can be calculated to be

$$\overline{\rho} = \frac{Q}{N}\mu_{\text{ppc}} = Q p_{\text{cell}} \quad \text{and} \tag{1.6}$$

$$\sigma_\rho = \frac{Q}{N}\sigma_{\text{ppc}} = Q\frac{\sqrt{N p_{\text{cell}} \left(1-p_{\text{cell}}\right)}}{N} = Q\frac{\sqrt{p_{\text{cell}} \left(1-p_{\text{cell}}\right)}}{\sqrt{N}}, \tag{1.7}$$

respectively. The quantity (1.7) can be interpreted as the error of the employed sampling. For an infinite amount of quasi-particles it tends to zero. In order to make the interpretation of the numerical results simpler, the relative standard deviation instead of the absolute one (given in (1.7)), will be used. The relative standard deviation is commonly called the coefficient of variation, and is given by

$$c_v = \frac{1}{\mathrm{SNR}_1^{(0)}} = \frac{\sigma_\rho}{|\overline{\rho}|} = \frac{1}{\sqrt{N}}\sqrt{\frac{1-p_{\mathrm{cell}}}{p_{\mathrm{cell}}}}, \tag{1.8}$$

where $\mathrm{SNR}_1^{(0)}$ refers to the signal-to-noise ratio of a distribution of quasi-particles with a zeroth-order form factor ($\mathrm{SNR}_1^{(\mathbf{0})}$, please note the boldface $\mathbf{0}$ that signifies the form factor) and a uniform weight, i.e. the distribution contains only quasi-particles of a single weight species ($\mathrm{SNR}_{\mathbf{1}}^{(0)}$, please note the boldface $\mathbf{1}$ that signifies the number of weight species).

In the case of a constant density $f(\vec{x},t) = c$, $p_{\mathrm{cell}} = \frac{1}{N_{\mathrm{cells}}}$ for all cells, were N_{cells} is the number of cells in the simulation, and $N_{\mathrm{ppc}} = {}^{N}/_{N_{\mathrm{cells}}}$ is the number of particles per cell. This leads to

$$c_v = \frac{1}{\sqrt{N}}\sqrt{N_{\mathrm{cells}}-1} \overset{N_{\mathrm{cells}}\gg 1}{\approx} \frac{1}{\sqrt{N_{\mathrm{ppc}}}}, \quad \text{and} \tag{1.9}$$

$$\mathrm{SNR}_1^{(0)} = \sqrt{N_{\mathrm{ppc}}\frac{N_{\mathrm{cells}}}{N_{\mathrm{cells}}-1}}. \tag{1.10}$$

1.2 Higher-order form factors

The analysis in this chapter is confined to one-dimensional considerations. The density that is deposited on one grid-point with index j and coordinate X_j is defined as

$$\nu_j^N = \sum_{i=1}^{N} \zeta_j^{(n)}\left(x_i\right), \tag{1.11}$$

where x_i is the coordinate of the ith particle and $\zeta_j^{(n)}$ is the density deposition function or integrated shape function in the x-direction of nth order and of grid-point j. The variable $\left(\nu_j^N\right)_s$ is now used as a random variable with the sample number s. Each sample s is constructed via N uniformly distributed particle coordinates x_i. The goal is to determine the structure of the statistical distribution of this random variable. This means determining $\overline{\nu}_j^N$ and $\sigma_{\nu_j}^N$. In order to do this, the probability distribution function $f_{\nu,j}^{(n),N}(\nu)$ must be found. This function returns the probability of getting ν as the deposited density on grid-point j.

In order to simplify the problem, $f_{\nu,j}^{(n)}(\nu)$, the probability distribution function for a single quasi-particle with coordinate x_i (i.e. $N = 1$), will be derived first. The variable $w_\zeta^{(n)}$ is defined by

the equation $\mathrm{supp}(\zeta^{(n)}) = [-w_\zeta^{(n)}, w_\zeta^{(n)}]$, where $\mathrm{supp}(\zeta^{(n)})$ is the support of the integrated shape function $\zeta^{(n)}$ and $w_\zeta^{(n)}$ is its width. The form factor is assumed to be a symmetric, continuous and differentiable function. It is also assumed that this function is strictly monotonic decreasing from its center. In the following, the given particle density distribution $f(x)$ will be transformed into a distribution governed by the integrated shape function $\zeta_j^{(n)}$, which can be related to $f_{\nu,j}^{(n)}(\nu)$. With $N = 1$, and leaving out N, equation (1.11) becomes

$$\nu_j = \zeta_j^{(n)}(x_i)\,. \tag{1.12}$$

If there is a single particle at position x depositing the density ν on grid-point j and $X_j \leq x \leq X_j + w_\zeta^{(n)}$, then the probability of having the charge ν deposited on this grid-point must be equal to the probability of finding that particle at position x:

$$|f_{\nu,j}^{(n)}(\nu)d\nu| = |f(x)dx| \tag{1.13}$$

$$\Rightarrow f_{\nu,j}^{(n)}(\nu) = \left|\frac{dx}{d\nu}\right| f(x) = \left|\frac{d}{d\nu}\left(\zeta_j^{(n)}\right)^{-1}(\nu)\right| f\left(\left(\zeta_j^{(n)}\right)^{-1}(\nu)\right) \tag{1.14}$$

The strict assumptions about x and $\zeta_j^{(n)}$ were necessary in order to have a bijective integrated shape function $\zeta_j^{(n)}$, without which $(\zeta_j^{(n)})^{-1}(\nu)$ would not be well-defined. This condition will be relaxed in the following paragraph.

In order to relax the above conditions for the form factor (symmetric, continuous, differentiable, strictly monotonic decreasing from its center), the form factor needs to be separated into sections for which an inverse function exists. Under that condition equation (1.14) can be applied to each section individually. The resulting values, for each section for which $(\zeta_j^{(n)})^{-1}(\nu)$ is defined, are summed up. The total sum of all contributions gives the probability for this specific ν. The only remaining condition for the form factor is therefore that its invertible sections are differentiable.

Furthermore, the probability that one particle contributes some density > 0 to a specific grid-point X_j is needed. It is given by

$$P_j = P(X_j) = \frac{\int_{X_j - w_\zeta}^{X_j + w_\zeta} f(x)dx}{\int_V f(x)dx}. \tag{1.15}$$

The probability that it will not contribute any density is therefore given by $1 - P_j$. Using all previous results and extending them in a straightforward matter, $f_{\nu,j}^{(n)}(\nu)$ is given by:

$$f_{\nu,j}^{(n)}(\nu) = \begin{cases} 0 & \nu < 0 \\ (1 - P_j)\delta(\nu) & \nu = 0 \\ \sum_{m=1}^{n(\nu)} \left|\left[\frac{d}{d\nu}\left(\zeta_j^{(n)}\right)^{-1}\right]_m\right| f\left(\left(\zeta_j^{(n)}\right)_m^{-1}\right) & 0 < \nu \leq 1 \\ 0 & \nu > 1 \end{cases} \tag{1.16}$$

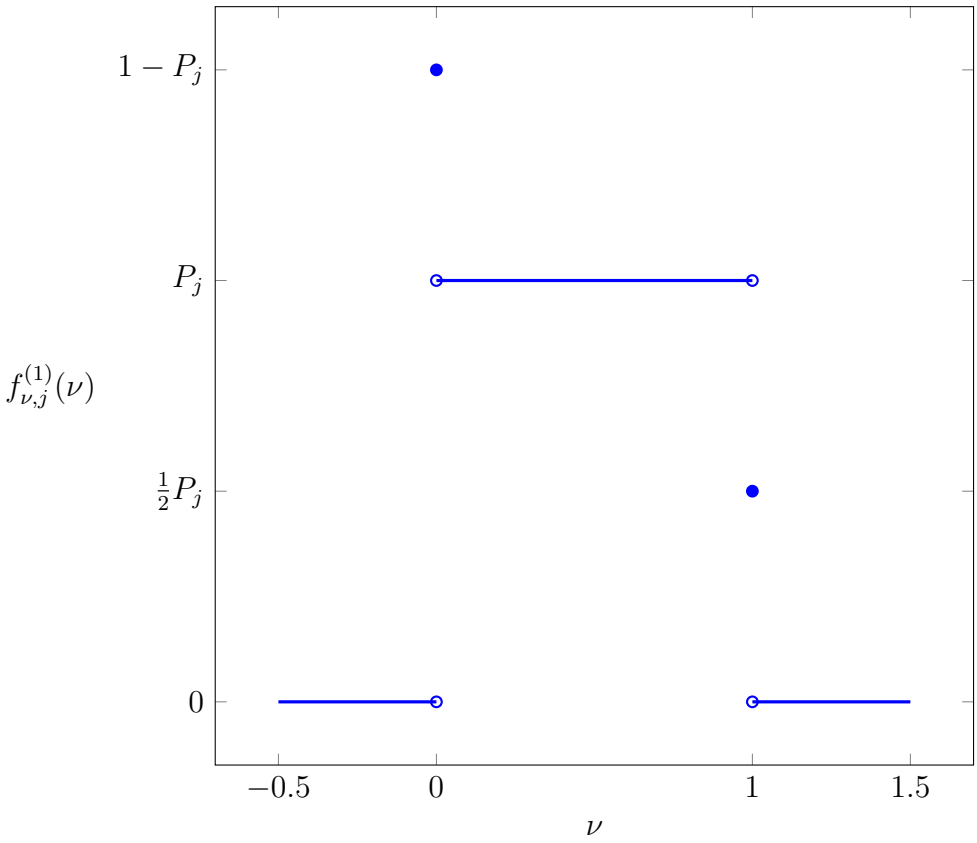

Fig. 1.1 – Probability density distribution of the density deposited on the grid-point X_j by one first-order quasi-particle. This function gives the probability of having the charge ν deposited on grid-point X_j. Variable P_j is defined in (1.15). The value of this function at $\nu = 0$ is given by $(1 - P_j)\delta(\nu)$, which is not represented in the figure.

Here, $n(\nu)$ is the number of invertible sections of $\zeta_j^{(n)}(x)$ that are well-defined for the value ν, $\left(\zeta_j^{(n)}\right)^{-1}_m$ are the different values of the inverse functions at ν and $\left[\frac{d}{d\nu}\left(\zeta_j^{(n)}\right)^{-1}\right]_m$ are the different values of the derivative of the inverse function at ν. The inclusion of the δ-distribution normalizes this function,

$$\begin{aligned}\int_{-\infty}^{\infty} f^{(n)}_{\nu,j}(\nu)d\nu &= \int_{-\infty}^{\leq 0} f^{(n)}_{\nu,j}(\nu)d\nu + \int_{>0}^{\leq 1} f^{(n)}_{\nu,j}(\nu)d\nu + \int_{>1}^{\infty} f^{(n)}_{\nu,j}(\nu)d\nu \\ &= 1 - P_j + P_j + 0 = 1,\end{aligned} \tag{1.17}$$

where the second integral is over the invertible sections. This enables the application of the mathematical properties of probability distribution functions.

In the following, the first-order form factor, given in (2.30), which is bijective for $x > 0$ and symmetric, is examined. For $f(x) = \text{const}$, $f^{(1)}_{\nu,j}(\nu)$ can then be defined as shown in Fig. 1.1.

The values of $\overline{\nu}_j^1$ and $\sigma_{\nu_j}^1$, where 1 signifies the number of quasi-particles $N = 1$, can now be determined:

$$\overline{\nu}_j^1 = \int_{-\infty}^{\infty} \nu f_{\nu,j}^{(1)}(\nu) d\nu = P_j \int_0^1 \nu d\nu = \frac{1}{2} P_j \tag{1.18}$$

$$\begin{aligned}
\sigma_{\nu_j}^1 &= \sqrt{\int_{-\infty}^{\infty} \left(\nu - \overline{\nu}_j^1\right)^2 f_{\nu,j}^{(1)}(\nu) d\nu} = \sqrt{\int_{-\infty}^{\infty} \left(\nu - \frac{1}{2} P_j\right)^2 f_{\nu,j}^{(1)}(\nu) d\nu} \\
&= \sqrt{\int_{-\infty}^{\infty} \left(\nu^2 - 2\nu \frac{1}{2} P_j + \frac{1}{4} P_j^2\right) f_{\nu,j}^{(1)}(\nu) d\nu} \\
&= \sqrt{\int_0^1 \nu^2 P_j d\nu - 2 \frac{1}{2} P_j \int_0^1 \nu P_j d\nu + \frac{1}{4} P_j^2 \int_{-\infty}^{\infty} f_{\nu,j}^{(1)}(\nu) d\nu} \\
&= \sqrt{\mathbb{E}\left[\nu^2\right] - \left(\overline{\nu}_j^1\right)^2} \\
&= \sqrt{\left(\frac{1}{3} - \frac{1}{4} P_j\right) P_j},
\end{aligned} \tag{1.19}$$

where $\mathbb{E}$ is the expected value of a random variable. Using the formulas

$$\mu(kX) = k\mu(X) \quad \text{and} \tag{1.20}$$

$$\sigma\left(kX\right) = |k|\sigma(X), \tag{1.21}$$

the random variable ν_j can be scaled with the charge per particle $\frac{Q}{N}$, resulting in

$$\overline{\rho}_j^1 = \frac{Q}{N} \overline{\nu}_j^1 = \frac{Q}{N} \frac{1}{2} P_j \quad \text{and} \tag{1.22}$$

$$\sigma_{\rho_j}^1 = \frac{|Q|}{N} \sigma_{\nu_j}^1 = \frac{Q}{N} \sqrt{\left(\frac{1}{3} - \frac{1}{4} P_j\right) P_j}. \tag{1.23}$$

In order to extend these derivations to $N > 1$ the following identities for standard deviations and means are employed:

$$\mu\left(\sum_N X\right) = N\mu(X) \tag{1.24}$$

$$\sigma\left(X + Y\right) = \sqrt{\mathrm{Var}\left(X\right) + \mathrm{Var}\left(Y\right) + 2\mathrm{Cov}(X,Y)} \tag{1.25}$$

$$\Rightarrow \sigma\left(\sum_N X\right) = \sqrt{N}\sigma(X) \tag{1.26}$$

The transformations to the standard deviation, done in equation (1.23) and (1.26), can also be understood as a sampling of the probability distribution with quasi-particles and computing the

standard error of the mean of this sampling, given, in general, by

$$\sigma\left(\overline{X}\right) = \sqrt{\mathrm{Var}\left(\overline{X}\right)} = \sqrt{\mathrm{Var}\left(\frac{1}{N}\sum_{i=1}^{N} X_i\right)} = \sqrt{\frac{1}{N^2}\,\mathrm{Var}\left(\sum_{i=1}^{N} X_i\right)} = \frac{1}{\sqrt{N}}\sigma\left(X\right). \quad (1.27)$$

Here, a property of the variance, $\mathrm{Var}(aX) = a^2\,\mathrm{Var}(X)$, as well as the analogue variance property to (1.26), given by $\mathrm{Var}(\sum_N X) = N\,\mathrm{Var}(X)$, was used. Equation (1.21) may seem like it contradicts (1.26). This is not the case. The first examines the random outcome of the random variable X, multiplies every outcome with a constant and computes the standard deviation of this distribution. The second, on the other hand, generates two independent random outcomes of a random variable, sums these outcomes and computes the standard deviation of this new distribution. Essentially, $\sigma(2X) \neq \sigma(X + X)$.

Assuming that the charge distributions from the different quasi-particles are independent, the values of $\overline{\rho}_j^N$ and $\sigma_{\rho_j}^N$ are then given by

$$\overline{\rho}_j^N = \frac{Q}{2}P_j \quad \text{and} \quad (1.28)$$

$$\sigma_{\rho_j}^N = \frac{|Q|}{\sqrt{N}}\sqrt{\left(\frac{1}{3} - \frac{1}{4}P_j\right)P_j}. \quad (1.29)$$

The signal-to-noise ratio on grid-point j for a single weight species and first-order particle shapes is therefore given by

$$\mathrm{SNR}_1^{(1)} = \frac{|\overline{\rho}_j^N|}{\sigma_{\rho_j}^N} = \frac{1}{2}\sqrt{\frac{NP_j}{\frac{1}{3} - \frac{1}{4}P_j}}. \quad (1.30)$$

Again, the subscript 1 in $\mathrm{SNR}_1^{(1)}$ refers to the number of different weight species. In general, if $f(x)$ is constant inside the volume of one quasi-particle (which is a relatively weak condition, as the PIC algorithm already assumes that $f(x) = \mathrm{const}$ inside a grid-cell), it holds that

$$P_j = f_j 2w_\zeta, \quad \text{where} \quad f_j = f(x) \quad \forall \quad X_j - w_\zeta^1 \leq x \leq X_j + w_\zeta^1. \quad (1.31)$$

A plot of (1.30) can be found in Fig. 1.2. Since $P_j \propto f$, c.f. to (1.31), it can be concluded that areas in the simulation with a lower particle density have a strictly worse signal-to-noise ratio in comparison to areas with a higher density. The reason for this is the decreased likelihood of quasi-particles inhabiting that volume. Because of this, it is key to increase the number of quasi-particles in lower density areas. This can be achieved by using the new merging/splitting algorithm.

In order to illustrate (1.30), one-dimensional simulations with $f(x) = \mathrm{const}$ were performed. For the first-order particle shape, it holds that $w_\zeta^1 = \Delta x$, where Δx is the size of one cell, c.f. equation (2.30), and using (1.31),

$$P_j = f_j 2\Delta x = \frac{1}{N_{\text{cells}}\Delta x}2\Delta x = \frac{2}{N_{\text{cells}}}, \quad (1.32)$$

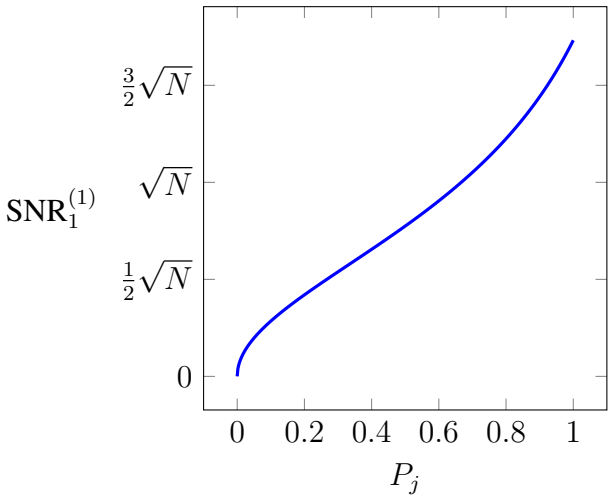

Fig. 1.2 – Plot of $\mathrm{SNR}_1^{(1)}(P_j)$, defined in equation (1.30).

where N_{cells} is the number of cells in the simulation. If equation (1.32) holds for all values of j, periodic boundary conditions are implied, as otherwise the left- and rightmost cells would have a different probability of only $1/N_{\text{cells}}$. This result, in combination with (1.30) and $N = N_{\text{ppc}} N_{\text{cells}}$, where N_{ppc} is the number of particles per cell, gives

$$\mathrm{SNR}_1^{(1)} = \sqrt{\frac{3}{2} N_{\text{ppc}} \frac{N_{\text{cells}}}{N_{\text{cells}} - \frac{3}{2}}}. \tag{1.33}$$

Comparing this formula with the formula for the zeroth-order form factor, given in equation (1.10), it can be seen that the factor k in $\mathrm{SNR}(N_{\text{ppc}}) = k\sqrt{N_{\text{ppc}}}$ has changed. In the first-order case, and for $N_{\text{cells}} \to \infty$, $k = \sqrt{3/2} \approx 1.2247$. In comparison, the zeroth-order form factor has $k = 1$, which is strictly worse. Therefore, higher-order form factors strictly increase the signal-to-noise ratio in a simulation. This modified factor k needs to be taken into account when assessing the advantages of higher particle orders in comparison to their computational performance penalty.

In order to test equation (1.33), simulations using different numbers of particles per cell N_{ppc} and different numbers of cells N_{cells} were conducted. For each configuration, $N_{\text{samples}} = 2048$ simulations were run, computing the mean and standard deviation of the resulting plasma density at the final timestep 1200. Each of these 2048 simulations uses a different initialization of the pseudo-random number generator, leading to different momenta and locations of the initial particles. The PIC code PSC [8] was used for these simulations.

One important assumption of the presented derivations is the random distribution of quasi-particles over the whole simulation domain. This is not supported by the PSC, as this is not especially useful as a feature for PIC codes in general. The PSC will always create an equal amount of quasi-particles in cells with the same prescribed particle density.

Parameter & notation	Value
Initial constant plasma density, n_0	$1 \times 10^{16}\,\mathrm{cm}^{-3}$
Plasma initial temperature	$5\,\mathrm{KeV}$
Number of quasi-particles	Varying
Spatial grid size, Δx	Varying
Courant-Friedrichs-Lewy number, f_c [4]	0.75
Number of timesteps	1200
Timestep size, dt	$\frac{f_\mathrm{c} dx}{c}$
Box size in x-direction	$2.5\,\mu\mathrm{m}$

Table 1.1 – Parameters of the SNR analytics verification simulations. The constant c refers to the velocity of light.

In order to approximate the assumption of a uniform quasi-particle distribution over the whole domain, a one-dimensional simulation, with the parameters given in Tab. 1.1, was used. Importantly, a large number of 1200 timesteps with a very high plasma temperature was simulated. The high plasma temperature of $5\,\mathrm{KeV}$ results in an average of $1200 \cdot {}^{1}\!/_{c} \cdot f_\mathrm{c} \sqrt{2 \cdot 5\,\mathrm{KeV}/m_\mathrm{e}} = 125.90$ cells traveled by each quasi-particle over the course of the simulation. Here, c is the velocity of light and m_e is the mass of an electron. Additionally, the precise starting location inside the cell of each quasi-particles was randomized.

Through these measures, the goal of approximating a randomized distribution of quasi-particles was accomplished and the above theory agrees very well with the results of the PIC simulations, given in Fig. 1.3 and Fig. 1.4. Both parameter dependencies are correctly represented in the resulting data, confirming that the above model is likely correct.

1.3 Multiple weight species

In order to understand the repercussions of the particle merging and splitting algorithm, explained in chapter 2, the above analysis needs to be extended to multiple weight species. When merging or splitting particles, the resulting configuration will necessarily include particles of different weights. Particles with a common weight are designated as belonging to the same weight species. It will be shown in this chapter that multiple weight species will strictly generate more noise than the same amount of particles with only a single weight.

The above analysis assumes a single weight species. Because of this assumption it follows that any process that leaves a single weight species of particles, for example an algorithm that splits all available particles in half, will be governed by (1.10) and (1.33). A process of this kind will not suffer from any increase in noise due to different weight species.

In order to calculate the SNR value for the case of two weight species, the formula to sum up two

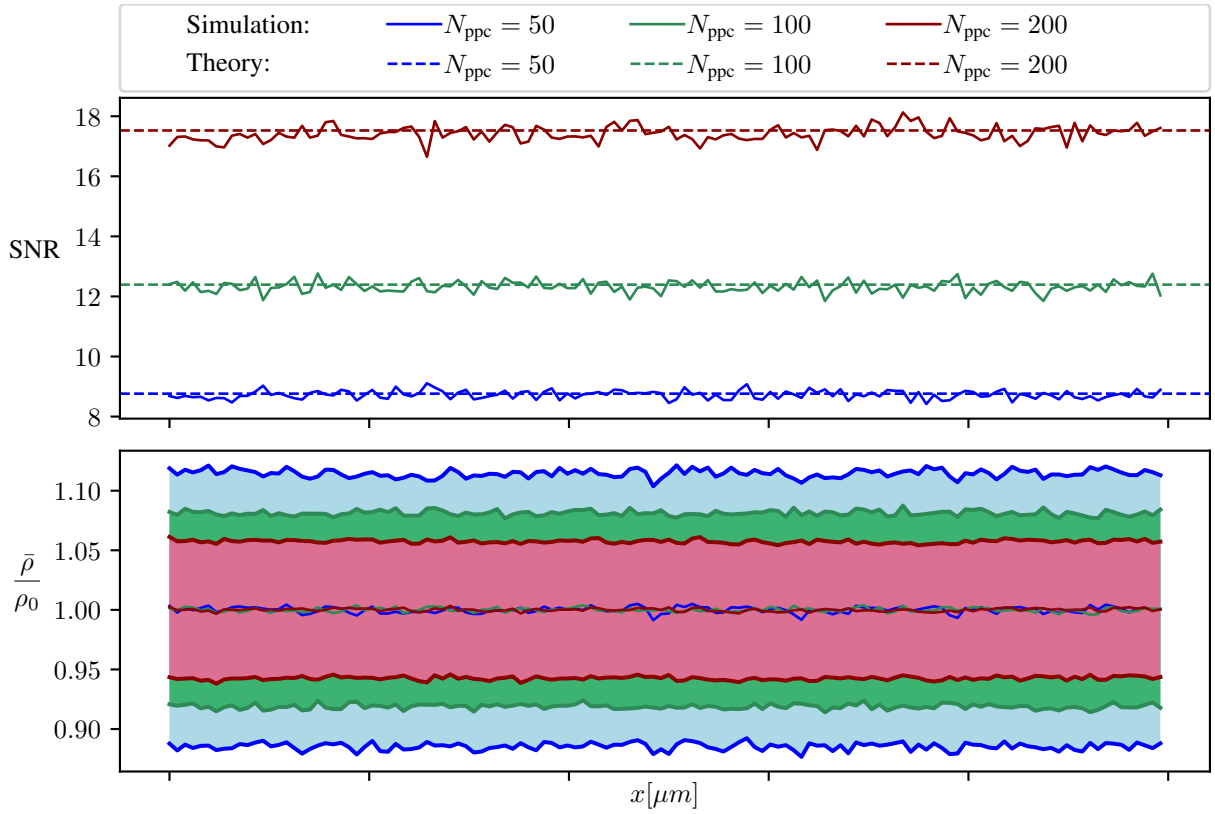

Fig. 1.3 – Comparison of simulation results, using different values for N_{ppc}, with the predictions of equation (1.33). For each case, $N_{\text{samples}} = 2048$ and $N_{\text{cells}} = 128$ holds. The lower plot depicts the normalized mean density and the normalized one-σ error region around it. The solid lines in the upper plot show the SNR values of the simulations, derived from the mean density and σ value given in the lower plot. The dashed lines in the upper plot give the theoretical prediction from equation (1.33).

random variables is used (c.f. equation (1.26), again for independent random variables), as each species produces its own random charge assignment on each grid-point, which is then summed up over all participating weight species:

$$\mu = \mu_1 + \mu_2 \tag{1.34}$$

$$\sigma^2 = \sigma_1^2 + \sigma_2^2 \tag{1.35}$$

$$\rightarrow \text{SNR} = \frac{\mu}{\sigma} = \frac{\mu_1 + \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2}} \tag{1.36}$$

Putting in (1.28) and (1.29), this equation becomes

$$\text{SNR}_2^{(1)} = \frac{Q_1 + Q_2}{\sqrt{\frac{Q_1^2}{N_1} + \frac{Q_2^2}{N_2}}} \frac{1}{2} \sqrt{\frac{P_j}{\frac{1}{3} - \frac{1}{4} P_j}} = \frac{Q_1 + Q_2}{\sqrt{\frac{Q_1^2}{N_1} + \frac{Q_2^2}{N_2}}} \frac{1}{\sqrt{N}} \text{SNR}_1^{(1)}, \tag{1.37}$$

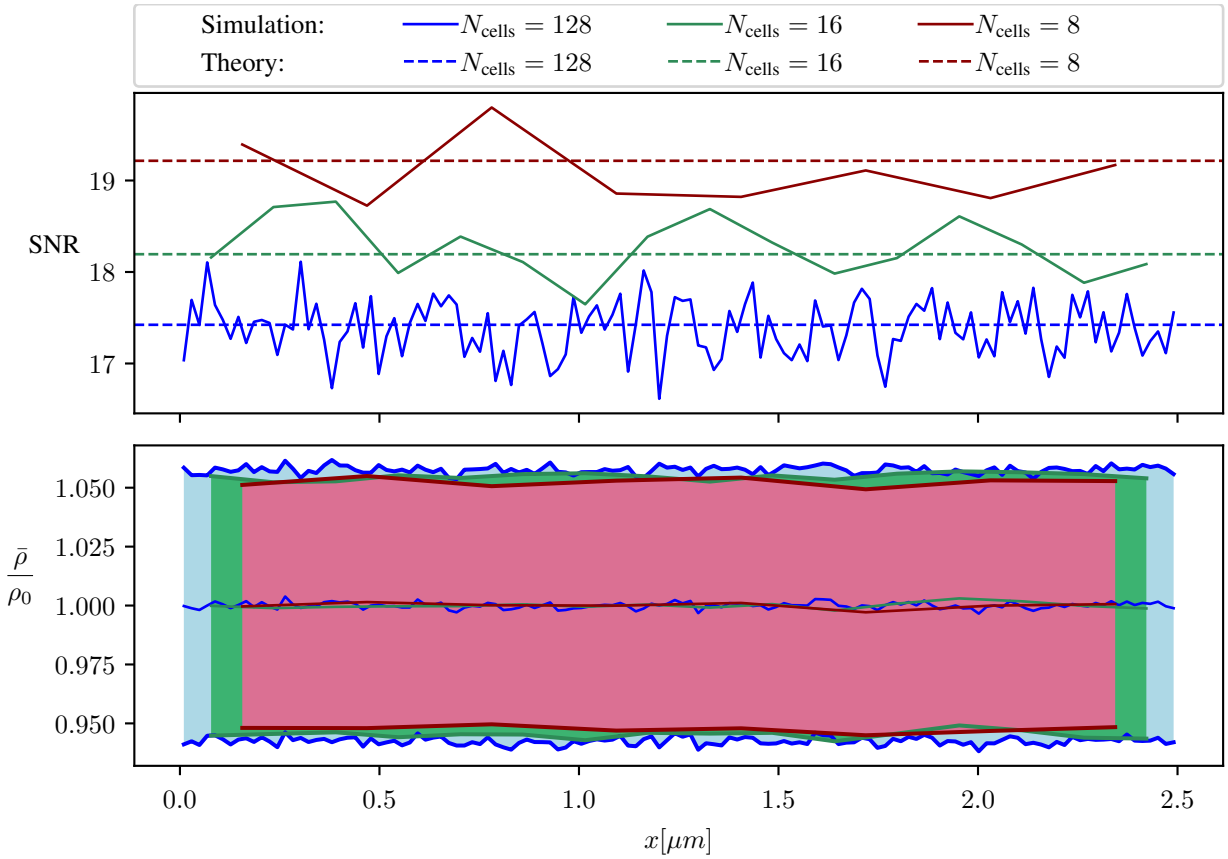

Fig. 1.4 – Comparison of simulation results, using different values for N_{cells}, with the predictions of equation (1.33). For each case, $N_{\text{samples}} = 2048$ and $N_{\text{ppc}} = 200$ holds. The lower plot depicts the normalized mean density and the normalized one-σ error region around it. The solid lines in the upper plot show the SNR values of the simulations, derived from the mean density and σ value given in the lower plot. The dashed lines in the upper plot give the theoretical prediction from equation (1.33).

where $\text{SNR}_1^{(1)}$ is defined in equation (1.30) or (1.33), $Q_1 + Q_2 = Q$ with Q_i the total charge of particles of species i, and $N_1 + N_2 = N$ with $N_i > 0$ the total number of particles of species i. It was also assumed that $P_{1j} = P_{2j} = P_j$. From

$$Q = Q_1 + Q_2 = \frac{Q_1}{N_1} N_1 + \frac{Q_2}{N_2} N_2, \tag{1.38}$$

a suitable definition for the charge of each quasi-particle of species i is derived to be Q_i/N_i. Using this definition, it is straightforward to set the weight of each quasi-particle of species i to

$$w_i = \frac{\frac{Q_i}{N_i}}{\frac{Q}{N}}, \tag{1.39}$$

which is a unit-less quantity. In this model and if there is only one weight species all quasi-particles have $w = 1$. The charge of a single quasi-particle of species i is then given by $w_i \frac{Q}{N} =$

$\frac{Q_i}{N_i}$. From these definitions it follows that

$$w_1 N_1 + w_2 N_2 = N \tag{1.40}$$

$$\Leftrightarrow \quad w_1 N_1 + w_2 N_2 = N_1 + N_2. \tag{1.41}$$

With this definition of w_i, equation (1.37) can be written as

$$\mathrm{SNR}_2^{(1)} = \sqrt{\frac{N_{1\mathrm{ppc}} + N_{2\mathrm{ppc}}}{N_{1\mathrm{ppc}} w_1^2 + N_{2\mathrm{ppc}} w_2^2}}\, \mathrm{SNR}_1^{(1)}, \tag{1.42}$$

where $N_{i\mathrm{ppc}} = \frac{N_i}{N_{\mathrm{cells}}}$.

When comparing (1.33) to (1.42), it is apparent that the existence of two weight species manifests itself in the additional factor

$$\sqrt{\frac{N_{1\mathrm{ppc}} + N_{2\mathrm{ppc}}}{N_{1\mathrm{ppc}} w_1^2 + N_{2\mathrm{ppc}} w_2^2}}. \tag{1.43}$$

This factor vanishes for $w_1 = w_2$, as expected. In order to understand the repercussions of multiple weight species it is helpful to show that

$$\sqrt{\frac{N_{1\mathrm{ppc}} + N_{2\mathrm{ppc}}}{N_{1\mathrm{ppc}} w_1^2 + N_{2\mathrm{ppc}} w_2^2}} \leq 1, \tag{1.44}$$

which means that the SNR value of a simulation that uses two different weight species to represent N particles is strictly worse than the SNR value of a simulation that uses only a single weight species. Starting from (1.44),

$$\sqrt{\frac{N_{1\mathrm{ppc}} + N_{2\mathrm{ppc}}}{N_{1\mathrm{ppc}} w_1^2 + N_{2\mathrm{ppc}} w_2^2}} \leq 1 \tag{1.45}$$

$$\Leftrightarrow \quad N_{1\mathrm{ppc}} + N_{2\mathrm{ppc}} \leq N_{1\mathrm{ppc}} w_1^2 + N_{2\mathrm{ppc}} w_2^2 \tag{1.46}$$

and with $w_1 \overset{(1.41)}{=} 1 + (1 - w_2)\,\frac{N_2}{N_1}$,

$$\Leftrightarrow \quad (1 - w_2)^2 \left(1 + \frac{N_2}{N_1}\right) \geq 0. \tag{1.47}$$

This equation holds true since $N_1 \in \mathbb{N}^+ \wedge N_2 \in \mathbb{N}^+$. The above derivations can be extended to more than two weight species in a straightforward manner. It can therefore be concluded that it is always beneficial to employ the smallest possible amount of different weight species.

It might be of interest to consider some examples in order to better understand the alteration of the SNR value when performing merging or splitting operations. The first example concerns

particle splitting. A fraction $0 \leq f \leq 1$ of all initial particles N will be split and doubled, while conserving the total charge. The different respective quantities of interest are then given by (primed values are values after splitting)

$$N_1' = (1-f)N, \tag{1.48}$$
$$N_2' = 2fN, \tag{1.49}$$
$$N' = (1+f)N, \tag{1.50}$$
$$Q_1' = (1-f)Q, \tag{1.51}$$
$$Q_2' = fQ \quad \text{and} \tag{1.52}$$
$$Q' = Q. \tag{1.53}$$

Using (1.39), the weights of the two final particle groups are then given by

$$w_1' = \frac{N'}{N} = 1 + f \quad \text{and} \tag{1.54}$$
$$w_2' = \frac{1}{2}\frac{N'}{N} = \frac{1}{2}(1+f). \tag{1.55}$$

Using these values, the effect on the particle signal-to-noise ratio can be calculated to be

$$\begin{aligned} \frac{\mathrm{SNR}_2'^{(1)}}{\mathrm{SNR}_1^{(1)}} &= \sqrt{\frac{N_1' + N_2'}{N_1' w_1'^2 + N_2' w_2'^2}} \cdot \sqrt{\frac{3}{2} N' \frac{1}{N_{\text{cells}} - \frac{3}{2}}} \Bigg/ \sqrt{\frac{3}{2} N \frac{1}{N_{\text{cells}} - \frac{3}{2}}} \\ &= \sqrt{\frac{1}{1 - \frac{1}{2}f}}. \end{aligned} \tag{1.56}$$

When increasing f from 0 to 1, this function strictly monotonically increases from 1 to $\sqrt{2}$. For $f = 1$ the number of particles is doubled, resulting in an increase in the SNR value by a factor of $\sqrt{2}$, as expected. For all values between 0 and 1, eq. (1.56) shows that increasing the particle number strictly improves the signal-to-noise value, even though two weight species are used. Of course, the simulation will also need more computational resources as the number of particles increases. The resulting SNR value is still smaller than the SNR value for the case that uses only one weight species for representing the increased particle number N' (which was also shown in general in eq. (1.47)), as

$$\begin{aligned} \frac{\mathrm{SNR}_2'^{(1)}}{\mathrm{SNR}_1'^{(1)}} &= \sqrt{\frac{N_1' + N_2'}{N_1' w_1'^2 + N_2' w_2'^2}} \cdot \sqrt{\frac{3}{2} N' \frac{1}{N_{\text{cells}} - \frac{3}{2}}} \Bigg/ \sqrt{\frac{3}{2} N' \frac{1}{N_{\text{cells}} - \frac{3}{2}}} \\ &= \sqrt{\frac{1}{(1 - \frac{1}{2}f)(1+f)}} < 1 \quad \text{for} \quad 0 < f < 1. \end{aligned} \tag{1.57}$$

Eq. (1.57) gives the general formula for the amount of SNR a simulation is missing out on when employing two weight species instead of only a single weight species. The minimum value of $\mathrm{SNR}_2'^{(1)} / \mathrm{SNR}_1'^{(1)}$ in $[0,1]$ is given by $2/3\sqrt{2} \approx 0.94$ for $f = 1/2$.

The second example concerns particle merging. A fraction $0 \leq f \leq 1$ of all initial particles N will be merged and halved, while conserving the total charge. The different respective quantities of interest are then given by (primed values are values after merging)

$$N_1' = (1-f)N, \tag{1.58}$$
$$N_2' = \frac{1}{2}fN, \tag{1.59}$$
$$N' = (1-\frac{1}{2}f)N, \tag{1.60}$$
$$Q_1' = (1-f)Q, \tag{1.61}$$
$$Q_2' = fQ \quad \text{and} \tag{1.62}$$
$$Q' = Q. \tag{1.63}$$

Using (1.39), the weights of the two final particle groups are then given by

$$w_1' = \frac{N'}{N} = 1 - \frac{1}{2}f \quad \text{and} \tag{1.64}$$
$$w_2' = 2\frac{N'}{N} = 2 - f. \tag{1.65}$$

Taking the same approach as before, the effect on the SNR value can be calculated:

$$\begin{aligned} \frac{\text{SNR}_2'^{(1)}}{\text{SNR}_1^{(1)}} &= \sqrt{\frac{N_1' + N_2'}{N_1' w_1'^2 + N_2' w_2'^2}} \cdot \sqrt{\frac{3}{2}N' \frac{1}{N_{\text{cells}} - \frac{3}{2}}} \Bigg/ \sqrt{\frac{3}{2}N \frac{1}{N_{\text{cells}} - \frac{3}{2}}} \\ &= \sqrt{\frac{1}{1+f}} \end{aligned} \tag{1.66}$$

This function is strictly monotonically decreasing from 1 to $1/\sqrt{2}$, as expected. Merging will therefore make the signal-to-noise ratio strictly worse, but will save computational resources due to the reduction in particles.

2 Merging/Splitting Algorithm

2.1 Introduction

The PIC method makes use of quasi-particles to represent the particle distribution function. One very common occurrence of such a distribution function is in the relativistic Vlasov-Maxwell (RVM) system. The number of quasi-particles is a crucial parameter when performing these simulations. An algorithm that is able to merge and split particles is called an Adaptive-Particle-Refinement (APR) algorithm. It enables the modification of the number of quasi-particles during the run-time of a simulation using an explicit scheme. The number of quasi-particles has important implications for a variety of problems:

1. The number of quasi-particles determines the magnitude of plasma fluctuations and noise ([12], [2], [23], [3]). In current simulations these fluctuations (also called discrete particle noise) are always much larger than those encountered in experiments ([14], [25]). At the same time, it is known that many instabilities in plasmas depend on the level of this noise. The number of quasi-particles is identified to be the singular most important property that determines the signal-to-noise ratio for various derived quantities (c.f. chapter 1).

2. If a process does not conserve the charge density, for example in the case of cascading (e^+e^--pair and photon production) in ultra-intense laser fields, plasma discharges [19] or ionization, the particle load can grow exponentially. The implication for simulations is that they can become unsustainable in terms of their computational costs, unless the particle load is mitigated by a refinement process.

3. For Monte-Carlo simulations, for example Monte-Carlo-Collisional (MCC) or Monte-Carlo self-field simulations, particles involved in the process must have an equal weight in order for the algorithm to be correct. APR can be capable of breaking down distributions composed of quasi-particles of non-equal weights into distributions of quasi-particles of equal weights. For this to work well, the number of different weights has to be restricted. Multiples of the merging and splitting factor of the APR algorithm work well in this case. Furthermore, keeping the amount of different weight species low is important, as multiple particle weights lead to more fluctuations in the simulation, as shown in chapter 1.3, especially equations (1.42) and (1.44).

4. Adaptive-Mesh-Refinement-PIC (AMR-PIC) techniques are considered in order to enable the adaptation of the resolution in configuration space. This method therefore leads to non-constant electromagnetic grids. APR is a candidate to solve the issue of representing particle distributions on non-constant grids.

The number of quasi-particles is often the main contributor to the computational cost of a simulation. The APR method can be used to keep the computational load on sustainable levels, while keeping noise properties inside acceptable limits. After changing the quasi-particles on-the-fly, during the run-time of an explicit scheme, the simulation will not follow the characteristics of the Vlasov equation anymore. Loosing out on this mathematical guarantee is troublesome, but it may be a necessity in order to enable the other important advantages of APR. Nonetheless, it

is very important to limit non-physical repercussions of APR as much as possible. In order to dynamically adapt the number of quasi-particles in a PIC simulation in an optimal fashion, the following 11 topics are therefore of interest:

1. Total mass and charge conservation
2. Total momentum conservation
3. Total energy conservation
4. Obey special relativity
5. Charge density conservation at each grid-point
6. Current density conservation at each grid-point
7. Momentum space distribution function conservation
8. The number of different weight species
9. Conservation of higher orders of the distribution function
10. The supported number of dimensions in phase space
11. The computational performance of the algorithm

Many different approaches have been published, each one addressing certain subsets of this list of topics of interest:

1. One of the most general methods was developed by Assous et al. [1]. It can be applied to different grids (finite difference, finite volume, finite element on unstructured or structured grids) and very different particle shapes. The proposed scheme is non-relativistic and produces a wide array of resulting weights. It can also require the solution of relatively large systems of equations. Moreover, making the method energy conserving demands excessive computational power.

2. Making use of the results from [1], Welch et al. [30] implemented a merging/splitting algorithm for 2D/3D orthogonal grids and first-order particle shapes. This entails the above mentioned problems of a large weight spread, inherent to the approach of Assous. Welch's method requires solving an 8×8 system of equations, where the existence of a solution can not be guaranteed. An expansion leading to relativistically correct merging/splitting is detailed in the paper. It requires the solution of a non-linear equation. In addition, algorithms to preserve the momentum distribution in a better way by incorporating approximate higher-order statistical momenta (e.g. kurtosis) are introduced. These additional algorithms only work for symmetric distributions in momentum space.

3. One of the most well-known publications about particle merging/splitting is the paper by Lapenta [13]. The paper classifies algorithms into two groups: i) binary and ii) ternary schemes. Binary schemes are given for up to three dimensions in configuration space. However, they only conserve the total energy *or* momentum for merging operations and conserve the grid moments only for first-order particle shapes. The ternary scheme conserves both the total energy and total momentum as well as the grid moments. For this scheme, merging is only possible in 1D and approximations have to be made for higher

dimensions. Both groups, the group of binary and the group of ternary schemes, do disturb the velocity distribution and make no special attempts to correct it. They are given for the non-relativistic regime only.

4. One of the most recent publications about particle merging is from Teunissen et al. [29]. It follows the work of Lapenta [13], inheriting most of its properties. It explores the capabilities of the binary scheme, testing different methods and presents a well suited algorithm to find particles that are close to each other in 6D space. One of the presented methods offers very good preservation of the velocity distribution by picking previously existing momenta. However, if that method is used, the total momentum is not preserved.

5. Frignani et al. [9] present an algorithm similar to Lapenta [13] and Shon et al. [27].

This chapter presents a new APR scheme, which combines many ideas from previous papers in a unique way. Referencing the list of the previously established topics of interest above, the method proposed in this dissertation achieves the following:

1. Total mass and charge conservation ✓
2. Total momentum conservation ✓
3. Total energy conservation ✓
4. Obey special relativity ✓
5. Charge density conservation at each grid-point ✓
6. Current density conservation at each grid-point ✓:
   The method is specialized to electromagnetic field solvers that make use of only the charge density via a current-conserving integration scheme [5].
7. Momentum space distribution function conservation:
   In order to enhance the fidelity in momentum space, previous momenta are used and modified slightly in order to guarantee momentum conservation. This method is similar to the approach described in [29], but has the added benefit of total momentum conservation.
8. The number of different weight species:
   The present algorithm favors merging/splitting particles with similar weights and keeps the number of weight species small by presorting in weight space. An additional step that reduces the number of different weight species is described as well.
9. Conservation of higher orders of the distribution function:
   The present algorithm does not preserve any higher orders of the distribution function.
10. The supported number of dimensions in phase space:
    A full description in one, two and three spatial as well as momentum dimensions is given.
11. The computational performance of the algorithm:
    The computational demand is comparable to the demand of the particle pusher. Most of it is generated by sorting the particles in 6D phase space, which could be mitigated by more complex particle structures.

The goal of this new algorithm is to achieve the conservation of the largest possible number of system properties. Properties that can not be preserved perfectly are preserved in an optimal way. Since the errors, which are introduced into the system by a particle merging/splitting algorithm, are very hard to quantify systematically and, in many cases, depend on the specific system under consideration, the best next target to aim at is to try to introduce the least possible amount of those errors.

Additionally, the algorithm requires very few parameters to be set at simulation start. It is therefore not necessary to guess multiple characteristics of the targeted non-linear system. Briefly summarized, the present approach ensures that the merging/splitting process has the least possible impact on the physics, while also being easy to configure and having good performance properties.

The method is only applicable to regular grids. The full preservation properties are reliably achievable only for first-order particles. For second-order particles a non-linear equation must be solved. This dissertation does not give a proof of existence for the solution of this equation, but it has been found to exist in the majority of tested cases. Still, due to the unpredictability of a non-linear equation solver, and the amount of times these equations would need to be solved, the present work focuses on first-order particles.

After this introduction, the algorithmic steps that are used to identify groups of particles as merging or splitting targets are described. Particles in these groups share their particle kind (their mass and charge) as well as their cell. The weight and momentum of all particles in a group is subject to certain similarity conditions, which are used to establish the quality of merging and splitting. These similarity conditions may be controlled using parameters. The impact of these parameters on the faithfulness of the algorithm is investigated subsequently in chapter 3.

Preferential treatment of particles with lower weights, in the case of merging, or higher weights, in the case of splitting, helps lowering the weight diversity of the resulting particles. This is advantageous to the resulting signal-to-noise ratio of the system, as discussed in chapter 1.

After describing the algorithm that identifies viable groups of particles, the process to compute the final locations and final momenta of the resulting after-merge or after-split particles is detailed. Since the process for splitting is very different compared to the process for merging, the two are discussed separately.

The progress of this work has been presented by Nils Moschüring at multiple international conferences. Some first details were presented at the Conference on Computational Physics (CCP) in October 2012 in Kobe, Japan. In the year 2014, Nils Moschüring gave a public update on his progress at the CCP in August in Boston, USA, and at the spring meeting of the Deutsche Physikalische Gesellschaft (DPG) in March in Berlin.

Quasi-particles have the following normalized properties: $\vec{x}_i$ (position), $\vec{p}_i$ (momentum), w_i (weight) and m_i (mass). The normalized particle energy (the energy of every particle inside a quasi-particle) is given by $E_i = \frac{m_i}{m_e} w_i \sqrt{p_i^2 + 1}$. Merging and splitting is restricted to particles of equal mass and therefore the factor m_i/m_e can be omitted.

2.2 Finding groups of target particles

In order to determine groups of particles that can be merged or split, the sorting procedures detailed in 2.2.1, 2.2.2 and 2.2.3 will be used. The original unsorted particle array is only given as an input in step 2.2.1. Each consecutive step is applied to each resulting particle group created by the previous step.

2.2.1 Sort particles by cell and by kind

The merging/splitting algorithm only works on particles that share a cell and that have the same kind. The kind of a particle is an abbreviation for its charge and mass. The sorting algorithm, used for grouping the particles by cell and kind, does not require any particular properties. The simulation code that was used for the examples in this dissertation, for example, uses a standard count sort implementation. Some codes store quasi-particles in different arrays, depending on their kind and cell. These codes do not need to perform this step.

A result of this grouping is N^{cell}, the number of particles, of a specific kind, in each cell. All subsequent formulas assume that the considered particles have the same kind. In order to determine if merging or splitting should happen, the algorithm needs a range $\left[N^{\text{cell}}_{\text{min}}, N^{\text{cell}}_{\text{max}}\right]$. The current amount of particles in each group is compared to this range. The action to be taken by the algorithm is chosen according to the following list:

$$\begin{aligned} N^{\text{cell}} < N^{\text{cell}}_{\text{min}} &\Rightarrow \text{ split} \\ N^{\text{cell}}_{\text{min}} \leq N^{\text{cell}} \leq N^{\text{cell}}_{\text{max}} &\Rightarrow \text{ skip} \\ N^{\text{cell}} > N^{\text{cell}}_{\text{max}} &\Rightarrow \text{ merge} \end{aligned} \tag{2.1}$$

Quasi-particle groups that are not skipped due to this prerequisite are supplied to the next step.

2.2.2 Sort particles by weight

This step starts with the previously determined groups of particles (sorted by kind and cell). Inside each of these groups, subgroups are created, according to the particle weight. It was shown in chapter 1.3, especially (1.42), that it is beneficial to have the smallest possible amount of different particle weight species. Therefore, in order to keep the spread of particle weights in the simulation as small as possible, an algorithm should try to merge light particles and split heavy particles. This necessitates a complete sort of particles according to their weight. The algorithm will then either i) start merging from the lower end of the weight-sorted particle array up until the number is sufficiently reduced, or ii) start splitting from the upper end down until the number has sufficiently increased. In order to save computational load this sort can be relaxed to only sort the particle array according to two weight bins given by $[w_{\text{min}}, w_{\text{avg}}]$ and $[w_{\text{avg}}, w_{\text{max}}]$.

Nonetheless, it was found that a more rigorous presort can be very beneficial. In order to minimize the amount of different weight species in the simulation, rigorous presorting is done using an amount of bins equal to the number of different weights in the cell. Any merging is done exclusively on these bins. Then, if all particles start with the same weight, and merging and splitting procedures are restricted to initial/final ratios of $1/2$ and 2 respectively, the particle weights in the simulation can only become powers of 2. This decreases the variety in weight space and therefore increases the statistical significance of the simulation results, as shown in chapter 1. It also makes it much easier to guarantee certain weights, if that is necessary for certain algorithms, for example for MCC algorithms.

The weight sort is also used to impose a lower limit on the particle weight. All particles below a certain minimal weight are not included in any resulting particle group that is destined for particle splitting. This imposes a minimum resolution for the resulting density distribution.

2.2.3 Sort particles by momentum

Again, this step starts with the previously determined groups of particles (now grouped by kind, cell and weight) and further refines these groups. Before the process of merging or splitting can finally be started the presorted particles are now clustered in momentum space. In order to achieve a high fidelity in momentum space, the momenta of particles that are selected for merging or splitting should be close together. If they are close together, the sum or fraction of the momentum is similar (per mass) to the value of the original particles, which makes the resulting momentum distribution similar as well. For most of the existing schemes the distance in momentum or configuration space is directly correlated to the quality of the merging process ([29], [13]). Closeness is not that vital for the present scheme, since it will choose the momenta of original quasi-particles for most of the resulting quasi-particles. Nonetheless, it will be verified to still be an important element.

A binary tree is used to perform the clustering of the three dimensional momentum-space. This algorithm is shown in Fig. 2.1. This clustering algorithm provides i) a dynamically adapted bucket size and ii) dynamically adapted bucket boundaries. This means that it is possible to choose between sorting into fixed buckets, or sorting as long as there are too many particles left in a bucket by adapting the bucket size.

First, the size of the underlying problem is determined by computing the minimum and maximum momentum value

$$p_{k,\text{min}} = \min_{\text{particles}} p_k \quad \text{and} \tag{2.2}$$

$$p_{k,\text{max}} = \max_{\text{particles}} p_k \quad \forall k \in \{x, y, z\} \tag{2.3}$$

of all particles in a group for every dimension. Second, the recursive algorithm divides one dimension in half, sorting the particles, and then acts on each of the two halves, dividing them

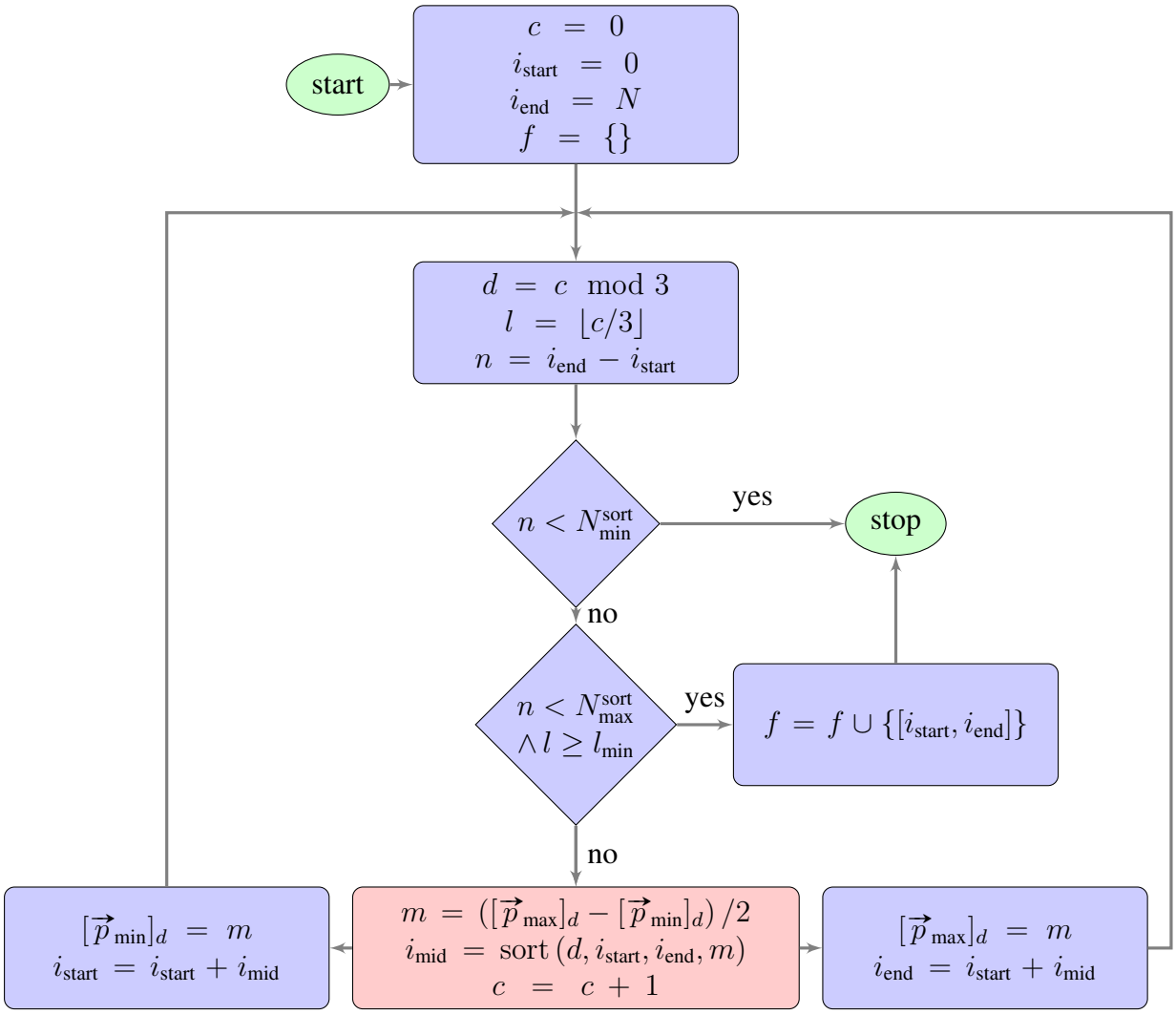

Fig. 2.1 – Flow diagram of the binary tree recursive clustering algorithm. The expression $[\vec{p}]_d$ refers to the d'th element of vector $\vec{p}$. The variable c gives the depth of the recursion, i_{start}, i_{end} and i_{mid} are indexes of particles inside the particle array, d gives the currently treated dimension, l the current level inside the binary tree and n the number of particles for the current recursive depth. The function $\text{sort}(d, i_1, i_2, m)$ sorts the particle array by the momentum dimension d, only considering particles between index i_1 and i_2 and producing two buckets $[i_1, i_{\text{mid}}[$ and $[i_{\text{mid}}, i_2]$, where the first buckets contains the particles with momentum $[\vec{p}]_d < m$ and the second bucket contains the particles with momentum $[\vec{p}]_d \geq m$. The bifurcation point is marked by the light red box. With the exception of f, all variables need to be duplicated for their respective branch at this point. The variable f is the set of buckets that are eligible for further processing. f is updated for each recursive iteration in a global manner.

in a different dimension and so on. This process cycles through the momentum dimensions p_x, p_y and p_z. The recursive binning algorithm in momentum space stops as soon as each of the bins holds a number of particles $N < N_{\text{max}}^{\text{sort}}$, where

$$N_{\text{max}}^{\text{sort}} \tag{2.4}$$

is a predefined number of maximum allowed particles in a momentum bin. If a branch contains fewer particles than needed for the subsequent merging/splitting algorithm $N_{\text{min}}^{\text{sort}}$, the recursion stops as well. The whole process is illustrated in Fig. 2.1.

$N_{\text{max}}^{\text{sort}}$ is a key parameter to control the fidelity to the initial shape of the shape of the distribution function in momentum space after splitting or merging. Setting $N_{\text{max}}^{\text{sort}}$ to a smaller value will provide smaller particle clusters, which lie closer together. The initial number of particles for a merging/splitting process N_{initial} is bounded by $N_{\text{min}}^{\text{sort}}$ and $N_{\text{max}}^{\text{sort}}$. In this way, the merging or splitting ratio

$$Q = \frac{N_{\text{initial}}}{N_{\text{final}}} \tag{2.5}$$

is also influenced by these two values. Furthermore, Q depends on the number of final particles in the merging or splitting algorithm N_{final}, for which some bounds will be defined later on.

When recursion stops in a certain subspace of phase space, particle splitting or merging can be performed on the particles in this subspace. The value of l, which describes the number of divisions of all dimensions during the recursive process, as shown in Fig. 2.1, can be used to determine the current size of the respective subspace in momentum space. Thus, using equations 2.2 and 2.3, a certain maximum distance dp_k, where $k \in \{x, y, z\}$, between to be merged or split particles can be enforced by setting a minimum splitting level

$$l_{\text{min}} = \max_{k=\{x,y,z\}} \left\{ \left\lceil \log_2 \frac{p_{k\text{min}} - p_{k\text{max}}}{dp_k} \right\rceil \right\}. \tag{2.6}$$

The minimum splitting level l_{min} can be revised and enhanced to a vector value. This way, the different momentum dimensions can have their own individual resolution. This change requires some rewiring in the binary tree algorithm and may not always be worthwhile. In many applications it is very helpful to define a minimum value for l_{min} that is independent of dp_k. The two-stream instability, for example, benefits greatly from forcing $l_{\text{min}} \geq 1$, since quasi-particles of the two counter-propagating streams should never be merged.

Since the algorithm acts on a, to a certain degree, random distribution, it is very important to not introduce biases into the system. The following list gives six important additions to the binning algorithm, which help accomplish this goal:

1. The minimum and maximum values for each dimension have been modified. The new values are given by $p_{k\text{min}}^{\text{mod}} = p_{k\text{min}} - r l_k$ and $p_{k\text{max}}^{\text{mod}} = p_{k\text{min}}^{\text{mod}} + 2 l_k$, where r is a random number

in the range $[0, 1[$ and $l_k = p_{k\text{max}} - p_{k\text{min}}$. This ensures that $p^{\text{mod}}_{k\text{min}} < p_{k\text{min}}$, $p^{\text{mod}}_{k\text{max}} > p_{k\text{max}}$ and that

$$\forall p \in [p_{k\text{min}}, p_{k\text{max}}] : P\left(p = \frac{\left(p^{\text{mod}}_{k\text{max}} - p^{\text{mod}}_{k\text{min}}\right)}{2}\right) = \text{const}, \tag{2.7}$$

where P is the probability over the random variable r. This ensures that there is no preferential value in momentum space after splitting and merging. This preferential value exists, if the respective dimension is always divided along the same line (while the momentum distribution in the cell stays approximately the same). Consistently dividing along the same line leads to spikes in the momentum distribution. In general, this modification necessitates an increase in l_{min} by one in order to maintain the same maximum distance in momentum space dp_k. This, in turn, increases the computational demand of the algorithm.

2. The order of the momentum dimensions in each full iteration of all momentum dimensions is chosen randomly. This ensures that there is no preferential momentum dimension with a consistently higher or lower enforced resolution.

3. It is helpful to implement an l_{max} value as well. The binary tree stops at every leaf that reaches this maximum level. It is even better to implement a vector value for it. The algorithm can avoid stack overflows by adhering to l_{max} (if the algorithm is implemented recursively). It is also possible to exclude specific momentum dimensions from merging/ splitting, if a vector valued l_{max} is available. Finally, the current implementation of this algorithm offers the possibility to decide what is done with a bin that is not being divided anymore, due to it being over the l_{max} limit. Either that bin is included in possible merging/ splitting operations, or it is excluded. In the case of merging, it is commonly advantageous to still merge the particles in those bins. They are already too similar with respect to their momentum, so merging them is desired. In the case of splitting the opposite is commonly true. The desired resolution in momentum space is already reached and splitting these particles is therefore not desired. It does still provide more quasi-particles, and if there is another process that increases momentum space coverage it may still be useful.

4. The binary tree produces two branches at each recursive iteration. The algorithm chooses randomly, which one of these two branches is pursued first. In doing this, the order of the recursive calls becomes arbitrary, which ensures that particle groups for merging or splitting are selected randomly. Since the algorithm stops merging and splitting particles in each cell when N^{cell} is within $\left[N^{\text{cell}}_{\text{min}}, N^{\text{cell}}_{\text{max}}\right]$, it is important to make sure that the algorithm does not always refine the same regions in phase space. The same effect can be achieved by randomizing the order of the resulting, after-binning, particle groups.

5. In modifying the previous point, one can opt to sort the resulting particle groups by their depth in the recursive tree, instead of randomizing them. In that case, the merging and splitting algorithm uses the groups that are deepest in the tree first, which will provide the highest resolution in momentum space. When doing this, specific parts of the momentum

space are merged or split preferentially, in comparison to other parts. This will lead to problems if the merging/splitting introduces large errors in the momentum space distribution (i.e. the momentum distribution gets distorted in a meaningful manner by the algorithm). If N^{cell} is large enough, this should not be a problem with the proposed algorithm, since it is very faithful in preserving the momentum space. For the later examples, the algorithm therefore always sorts the resulting particle groups by their depth in momentum space.

6. Finally, an anisotropic binary tree resolution was implemented as well. When activating this modification, each bin in the binary tree can only be further refined and processed, if it shares a boundary with $[p_{k\text{min}}, p_{k\text{max}}]$. This makes the resolution more coarse in the center of phase space, allowing for more merges to happen. In most applications, more particles will reside in this center region and therefore merging will mostly happen there. The outer regions oftentimes need a higher resolution due to having fewer particles representing them. This modification therefore increases the resolution in regions which might need it, while facilitating merging in regions that require a lower resolution. A further extension of this feature allows for setting a threshold region $[p_{k\text{thstart}}, p_{k\text{thend}}]$. In that case, only bins that lie completely outside or overlap with this threshold region are allowed to be refined. This enables specific phase space regions to be resolved coarsely, which may be valuable for certain applications.

This concludes the particle grouping part of the algorithm. The result of this effort is a list of particle groups, where all particles in a specific group share certain properties: i) they are of the same kind ii) they reside in the same cell in configuration space, iii) they have similar weights and finally iv) they fulfill customizable requirements regarding their closeness in momentum space.

2.3 Determination of particle properties after merging

Following the procedure laid out in the previous section, there is now a list of particle groups, with each group consisting of a number of particles between $N^{\text{sort}}_{\text{min}}$ and $N^{\text{sort}}_{\text{max}}$. These particle groups are ordered by weight from smaller to larger, as described in chapter 2.2.2. The groups will now be processed, taking advantage of this order, with the goal of decreasing the number of quasi-particles (merging). In this section, the necessary algorithms to perform a merging operation are detailed. They will decrease the number of particles in each distinct group.

2.3.1 Determination of final locations

This scheme is tailored for PIC algorithms that calculate currents by depositing charge on a grid, followed by solving the continuity equation in order to achieve charge conservation [5]. One example of such an algorithm is detailed in [26].

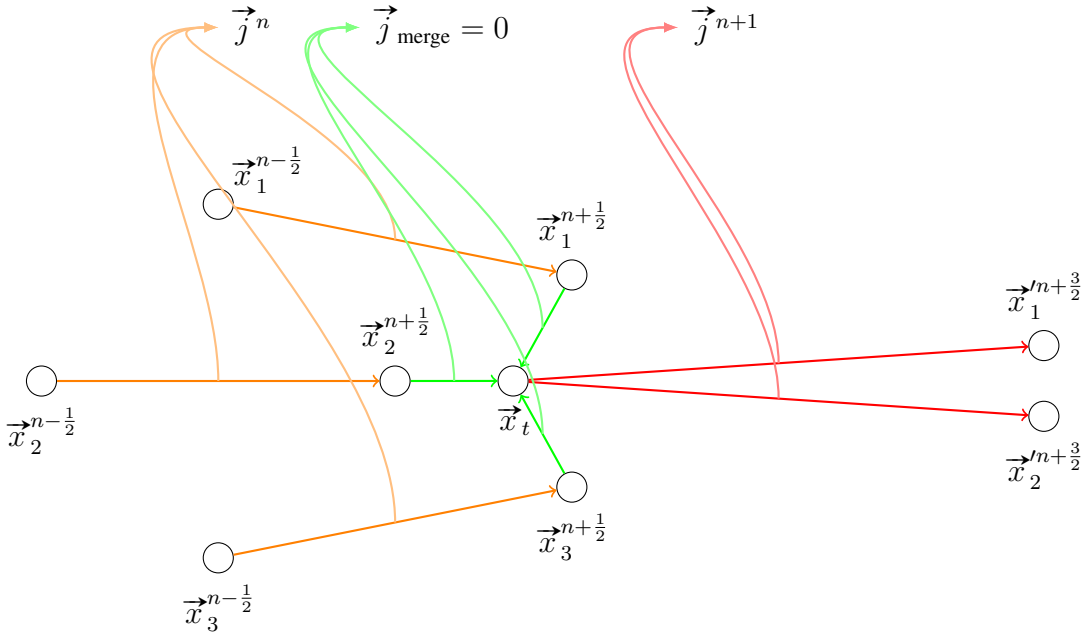

Fig. 2.2 – Process to accumulate 3 particles at a single location and merging them into 2 particles, accompanied by two particle push steps. The diagram depicts two spatial dimensions. The superscript of each quantity describes its respective timestep, with n being an arbitrary timestep of the simulation. The quantity $\vec{x}_1^{\,n-\frac{1}{2}}$ therefore describes the location of the first particle at timestep $n - \frac{1}{2}$, while $\vec{j}^{\,n}$ describes a current density at timestep n. The primed quantities are locations of the final particles after merging. The value $\vec{x}_t$ is a target location of the accumulation process. If a correct merging location $\vec{x}_t$ can be found, the spurious current density $\vec{j}_{\text{merge}}$ will be 0 and the process is free of any spurious divergence in the electric field. It will be shown that each merging process may contain multiple target locations, and t will index these locations.

First, the algorithm needs to accumulate all particles of a group on several distinct locations. The number of these distinct locations must be smaller than the number of initial particles. Each merging operation will result in fewer particles, which means that multiple initial particles must share a location. A simple representation of this process is shown in Fig. 2.2. If the initial particles do not share a location, merging the particles will create spurious divergence in the electric fields. The source of this divergence is found in

$$\frac{\partial}{\partial t}(\rho - \vec{\nabla} \cdot \vec{E}) = 0, \tag{2.8}$$

which can be derived from (2.9) and Maxwell's equations. From this equation, it follows that any change in the charge distribution will result in a change in the divergence of the electric field vector. This change in divergence is equal to the electric fields that would be produced by moving the initial particles to that shared location. In order to avoid this change in divergence, this movement has to be done in a very specific manner, a manner that does not produce any

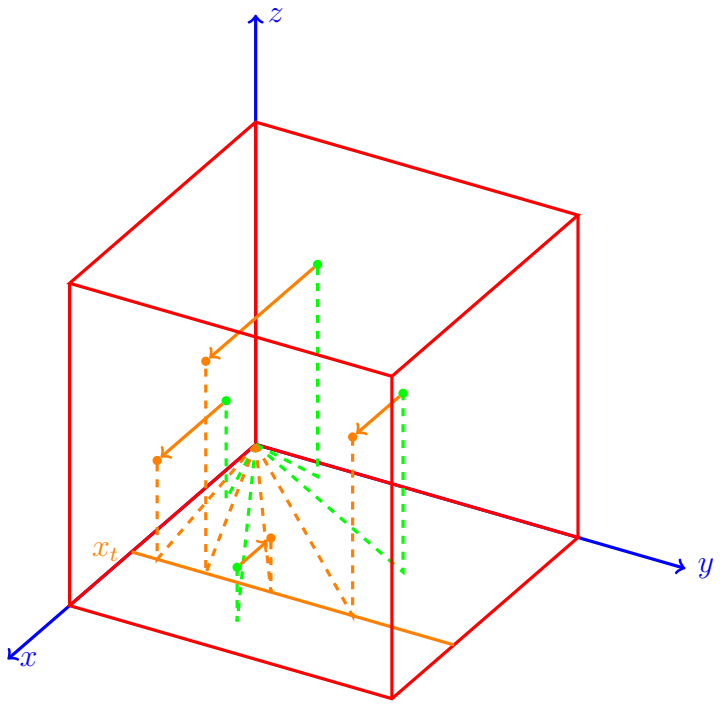

Fig. 2.3 – Decoupled x-direction particle accumulation. The final particles (orange) have a common x-coordinate. The initial particles (green) do not have this property.

additional divergence. From

$$\frac{\partial}{\partial t}\rho + \vec{\nabla} \cdot \vec{j} = 0, \tag{2.9}$$

the continuity equation, it follows that particle movement will not introduce any spurious divergence into the system if it does not entail any current ($\vec{j}_{\text{merge}}$ in Fig. 2.2). Movement without currents will not change the charge deposited on the grid cells, which will, in turn, guarantee that the change in divergence is zero. Achieving this specific movement is detailed in the following section.

In order to conserve the distribution function in configuration space, the charge deposition on each grid-point of each affected cell must stay invariant during a merge operation. From eq. 2.9, it was already deduced that if a scheme does not produce any currents, it will also not produce any change in the deposited charge on the grid. Naturally, movements along a specific axis will only produce currents in that specific direction. As a consequence, the problem of shifting particles to a common location without changing the deposited charge can be reduced to the problem of solving three sets of independent problems. First, the target merge position in the x-direction x_t (Fig. 2.3) must be found. Second, using x_t the target merge position y_t can be found (Fig. 2.4). Third, making use of x_t and y_t, the algorithm finds the target merge position z_t (Fig. 2.5).

This way it is possible to find a target position $\vec{x}_t = \begin{pmatrix} x_t & y_t & z_t \end{pmatrix}$ with the following characteristics:

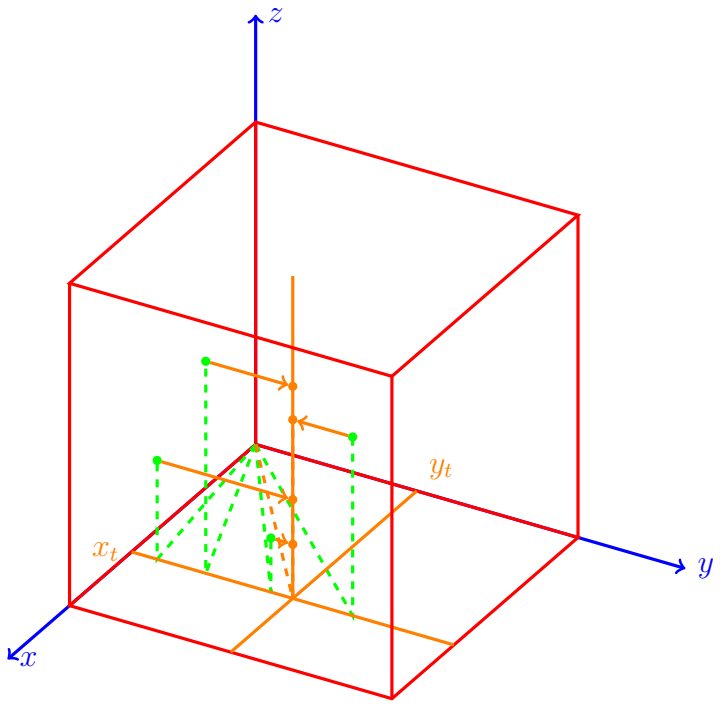

Fig. 2.4 – Decoupled y-direction particle accumulation, to be performed after the x-direction step, shown in Fig. 2.3. The final particles (orange) have a common x- and y-coordinate. The initial particles (green) do not have this property.

- All particles can be moved to $\vec{x}_t$ in three steps. First, going along the x-axis, the y and z coordinates of the particles are left unchanged. Next, the particles move along the y-axis, making use of the position x_t, and leaving the z coordinates of the particles unchanged. Finally, the particles are moved along the z-axis, with fixed positions x_t and y_t (the steps are shown in Fig. 2.3 - Fig. 2.5).
- No current is produced if all initial particles are moved in this manner to $\vec{x}_t$.

Discretizing (2.9), the current density j for the x-direction (the formulas for the y and z directions are derived analogously) can be calculated as follows [26, (4.146)]:

$$j^{n+1}_{j+\frac{1}{2}kl} = j^{n+1}_{j-\frac{1}{2}kl} + \frac{\Delta x}{\Delta t}\left(\rho^{n+\frac{3}{2}}_{jkl} - \rho^{n+\frac{1}{2}}_{jkl}\right), \tag{2.10}$$

where Δx and Δt are the grid size in x-direction and the time step size, respectively. Fig. 2.6 gives the definition of the grid coordinates j, k and l. The total charge contribution of a single quasi-particle located at $\vec{x}$ onto the spatial nodes at $j\Delta x$, $k\Delta y$ and $l\Delta z$ is obtained by the following expression [26, (4.136)]

$$\Sigma_{jkl}\Delta x\Delta y\Delta z\,\zeta_{jkl}\left(\vec{x}\right) = \Sigma_{jkl}\int_{x_j-\frac{\Delta x}{2}}^{x_j+\frac{\Delta x}{2}} dw_1 \int_{y_k-\frac{\Delta y}{2}}^{y_k+\frac{\Delta y}{2}} dw_2 \int_{z_l-\frac{\Delta z}{2}}^{z_l+\frac{\Delta z}{2}} dw_3\, S\left(\vec{x}-\vec{w}\right), \tag{2.11}$$

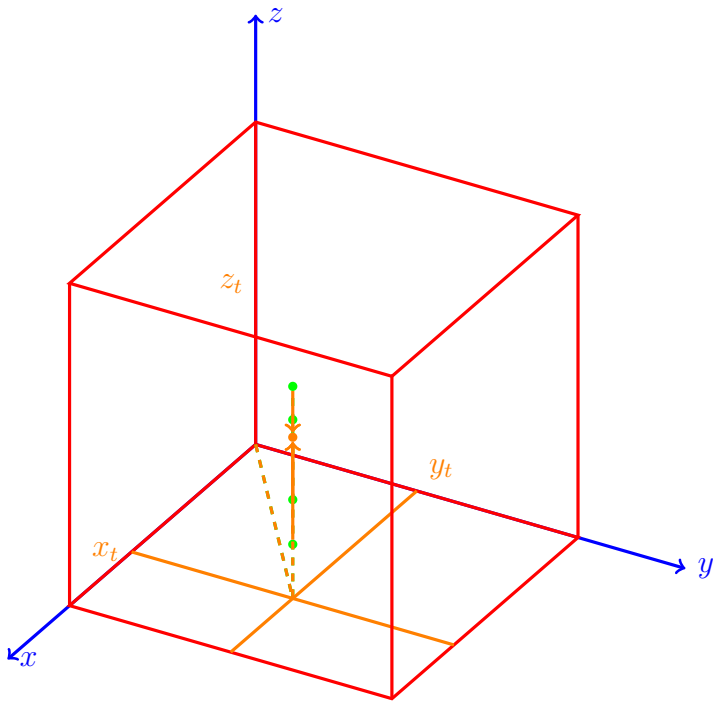

Fig. 2.5 – Decoupled z-direction particle accumulation, to be performed after the y-direction step, shown in Fig. 2.4. The final particles (orange) share the same location.

where $\zeta_{jkl}\left(\vec{x}\right)$ is the charge distribution of a quasi-particle with the shape function

$$S\left(\vec{x}\right) = S_1\left(x\right) S_2\left(y\right) S_3\left(z\right) \tag{2.12}$$

at location $\vec{x}$ over the nodes at $j\Delta x$, $k\Delta y$ and $l\Delta z$ with coordinates (x_j, y_k, z_l) of the grid. The structure of S will be given later. This requires that [26, (4.137)]

$$\zeta_{jkl}\left(\vec{x}\right) = \zeta_{1j}\left(x\right)\zeta_{2k}\left(y\right)\zeta_{3l}\left(z\right) = \frac{1}{\Delta x\Delta y\Delta z}\int_{x_j-\frac{\Delta x}{2}}^{x_j+\frac{\Delta x}{2}} dw_1 \int_{y_k-\frac{\Delta y}{2}}^{y_k+\frac{\Delta y}{2}} dw_2 \int_{z_l-\frac{\Delta z}{2}}^{z_l+\frac{\Delta z}{2}} dw_3\, S\left(\vec{x}-\vec{w}\right). \tag{2.13}$$

Since, instead of performing a full time step, only the current for an arbitrary particle displacement is determined, (2.10) can be written as

$$j_{j+\frac{1}{2}kl} = j_{j-\frac{1}{2}kl} + \frac{\Delta x}{\Delta t}\Delta\rho_{jkl}, \tag{2.14}$$

where, for one dimensional particle movement ([26, (4.145)]),

$$\begin{aligned}\Delta\rho_{jkl} &= w\zeta_{jkl}\left(\vec{x}+\delta x\,\vec{e}_x\right) - w\zeta_{jkl}\left(\vec{x}\right)\\ &= w\left[\zeta_{1j}\left(x+\delta x\right) - \zeta_{1j}\left(x\right)\right]\zeta_{2k}\left(y\right)\zeta_{3l}\left(z\right),\end{aligned} \tag{2.15}$$

here, $\vec{x}$ is the particle position with components x, y and z , $\vec{e}_x$ is the unit vector in x-direction and $\delta x = x_t - x$. Equation (2.15) can now be used to set $\Delta\rho$ to zero at an arbitrary position jkl, using a total number of particles N:

$$\begin{aligned}\Delta\rho_{jkl} &= \sum_{i=0}^{N-1} w_i \left[\zeta_{1j}\left(x_i + \delta x_i\right) - \zeta_{1j}\left(x_i\right)\right] \zeta_{2k}\left(y_i\right) \zeta_{3l}\left(z_i\right) \\ &= 0,\end{aligned} \tag{2.16}$$

where $\delta x_i = x_t - x_i$ is the shift of the ith particle. The value $\Delta\rho_{jkl}$ refers to the total charge difference of all participating particles. T is the number of final particle positions. The final positions of the particles are then given by

$$x_i + \delta x_i = x_t, \quad \text{where} \quad 0 \leq t \leq T-1. \tag{2.17}$$

Since there should be fewer final positions than initial positions, it holds that

$$GT = N, \tag{2.18}$$

with $G > 1, G \in \mathbb{N}$. Factoring out the common form factors as particles start to share the same point in space, equation (2.16) can be written as

$$\sum_{t=0}^{T-1} \zeta_{1j}\left(x_t\right) \sum_{g=0}^{G-1} w_{Gt+g} \zeta_{2k}\left(y_{Gt+g}\right) \zeta_{3l}\left(z_{Gt+g}\right) = \sum_{i=0}^{N-1} w_i \zeta_{1j}\left(x_i\right) \zeta_{2k}\left(y_i\right) \zeta_{3l}\left(z_i\right). \tag{2.19}$$

In this equation, G (the merge group size) is the number of particles that will share a location, and that will therefore be able to be merged into fewer particles. This equation assumes a specific distribution of the initial particles to the final positions. In the following it will be seen that the specific choice of this distribution is of significance. The goal is to find a value for every x_t so that (2.19) holds for every element of a specific group of jkls. This can be impossible for a certain choice of distributing the initial particles to the final positions. Nonetheless, studying the permutations shows that testing out different distributions can lead to a solvable system (see also the later section in this chapter **Permutations**).

For each j, the previous equation (2.19) constitutes a system of linear equations:

$$\mathbf{A}\overrightarrow{\zeta_{1j}\left(x\right)} = \vec{b_j}, \tag{2.20}$$

where $\mathbf{A} \in \mathbb{R}^{C \times T}$, $\overrightarrow{\zeta_{1j}\left(x\right)} \in \mathbb{R}^T$ and $\vec{b_j} \in \mathbb{R}^C$ are given by

$$A_{rt} = \sum_{g=0}^{G-1} w_{Gt+g} \zeta_{2k}\left(y_{Gt+g}\right) \zeta_{3l}\left(z_{Gt+g}\right) \tag{2.21}$$

$$\overrightarrow{\zeta_{1j}\left(x\right)} = \begin{pmatrix}\zeta_{1j}\left(x_0\right) & \dots & \zeta_{1j}\left(x_{T-1}\right)\end{pmatrix} \tag{2.22}$$

$$b_{jr} = \sum_{i=0}^{N-1} w_i \zeta_{1j}\left(x_i\right) \zeta_{2k}\left(y_i\right) \zeta_{3l}\left(z_i\right) \tag{2.23}$$

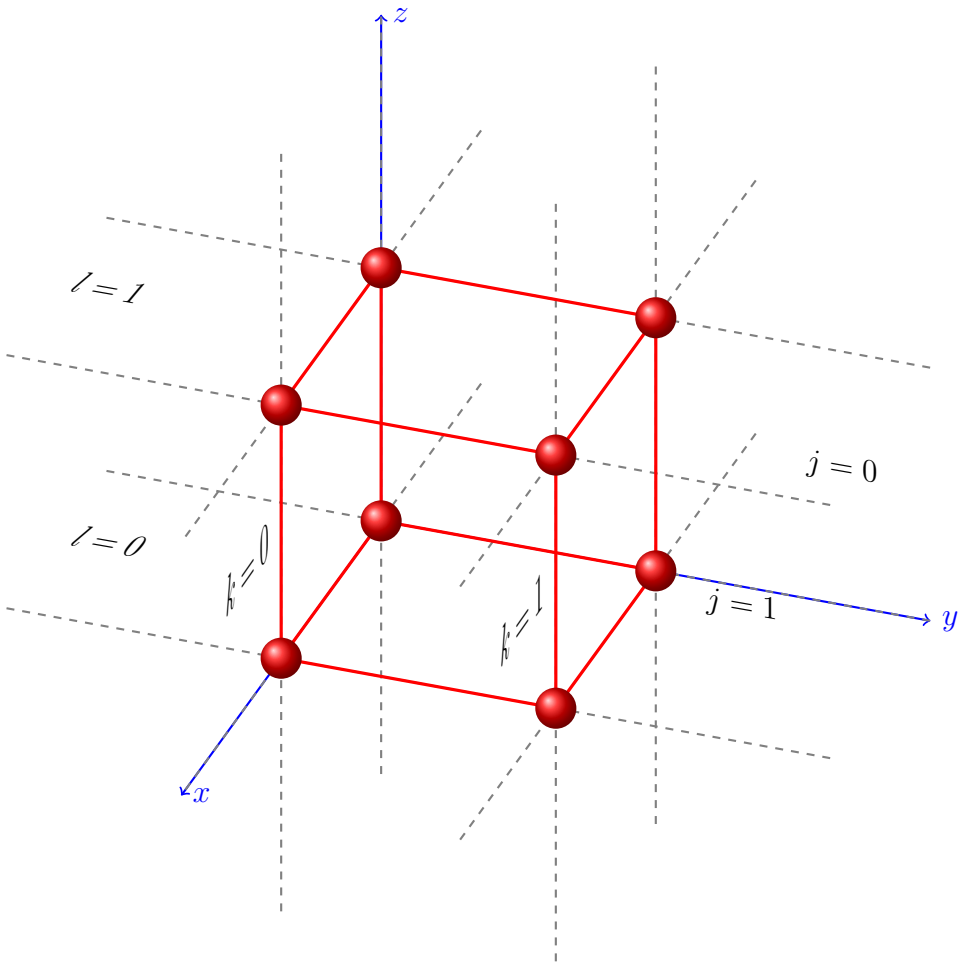

Fig. 2.6 – Illustration of the charge deposition for first-order particle shapes. The eight corners of the red cube, marked with red balls, will experience charge deposition from particles inside the red cube.

with

$$\begin{aligned} r\colon \{1,\ldots,K\cdot L\} &\to \{k_1,\ldots,k_K\}\times\{l_1,\ldots,l_L\},\\ r &\mapsto (k,l), \end{aligned} \tag{2.24}$$

and $C = K \cdot L$. The new variable r maps onto all values of k and l for which $\Delta\rho_{jkl} = 0$ should hold. K and L denote the amount of values for k and l.

First-order form factor

In this chapter, equation (2.19) will be solved for first-order form factors. Since, in this case, there are only two layers of cells that are affected with current by the movement of particles (cf.

Fig. 2.6) equation (2.14) can be written as:

$$j_{\frac{1}{2}kl} = \frac{\Delta x}{\Delta t}\Delta\rho_{0kl} \tag{2.25}$$

$$j_{\frac{3}{2}kl} = j_{\frac{1}{2}kl} + \frac{\Delta x}{\Delta t}\Delta\rho_{1kl} \tag{2.26}$$

$$= \frac{\Delta x}{\Delta t}\left(\Delta\rho_{0kl} + \Delta\rho_{1kl}\right) \tag{2.27}$$

The particles reside in the cube spanned by the coordinates $j = k = l = 0$ and $j = k = l = 1$ (cf. Fig. 2.6). Total charge conservation guarantees that

$$j_{\frac{3}{2}kl} = 0 \quad \Leftrightarrow \quad \Delta\rho_{0kl} + \Delta\rho_{1kl} = 0. \tag{2.28}$$

From this it follows that nullifying (2.25) will result in perfect divergence-free movement. This is equivalent to solving (2.20) with $j = 0$. The shape factor for first-order particle shapes is given by

$$S_1^{(1)}(x) = \rho_0 \begin{cases} 0 & |x| > \frac{\Delta x}{2} \\ \frac{1}{\Delta x} & |x| \leq \frac{\Delta x}{2} \end{cases}, \tag{2.29}$$

where ρ_0 is an arbitrary charge density, which depends on the normalization. Shape factors $S_2^{(1)}$ and $S_3^{(1)}$ are found by substituting Δx for Δy and Δz, respectively. Using (2.13), the integrated particle shape function (i.e. the charge deposition function) for a first-order particle shape is given by

$$\zeta_{1j}^{(1)}(x) = \rho_0 \begin{cases} 0 & |x - x_j| > \Delta x \\ 1 - \frac{|x-x_j|}{\Delta x} & |x - x_j| \leq \Delta x \end{cases}. \tag{2.30}$$

The integrated particle shape functions $\zeta_{2k}^{(1)}$ and $\zeta_{3l}^{(1)}$ are found by substituting Δx for Δy and Δz, respectively, and x_j for y_k and z_l, respectively. Specializing (2.20) for $j = 0$ and $K = L = 2$ (cf. Fig. 2.6), results in a system of linear equations:

$$\mathbf{A}\overrightarrow{\zeta_{10}^{(1)}(x)} = \vec{b_0}, \tag{2.31}$$

where $\mathbf{A} \in \mathbb{R}^{4\times T}$, $\overrightarrow{\zeta_{10}^{(1)}(x)} \in \mathbb{R}^T$ and $\vec{b_0} \in \mathbb{R}^4$ are given by

$$A_{rt} = \sum_{g=0}^{G-1} w_{Gt+g}\zeta_{2k}\left(y_{Gt+g}\right)\zeta_{3l}\left(z_{Gt+g}\right) \tag{2.32}$$

$$\overrightarrow{\zeta_{10}^{(1)}(x)} = \left(\zeta_{10}^{(1)}(x_0) \quad \ldots \quad \zeta_{10}^{(1)}(x_{T-1})\right) \tag{2.33}$$

$$b_{0r} = \sum_{i=0}^{N-1} w_i\zeta_{10}^{(1)}(x_i)\,\zeta_{2k}^{(1)}(y_i)\,\zeta_{3l}^{(1)}(z_i) \tag{2.34}$$

$$\text{with} \quad r\colon \{1,\ldots,4\} \to \{0,1\}^2, r \mapsto (k,l). \tag{2.35}$$

The variable r maps onto the 4 points in the first slab. Therefore, in order to make (2.31) a solvable equation, at least four degrees of freedom are required. Accordingly, this demands that $T \geq 4$. In the presented algorithm and its test cases $T = 4$. The formulas governing energy and momentum conservation, which are derived in chapter 2.4.2, require at least 3 initial particles for each merge operation. Therefore a minimum of

$$N_{\text{min}}^{\text{sort}} = 3 \cdot 4 = 12 \tag{2.36}$$

initial particles is needed. $N_{\text{min}}^{\text{sort}}$ has been introduced in chapter 2.2.3.

The presented algorithm solves the system of equations (2.31) using a standard LU decomposition with simple row wise pivoting. Using this solution and the inverse of $\zeta_{10}^{(1)}(x_t)$, the target positions for each spatial dimension can be found. Since the final particles are not allowed to cross cells, which would deposit current on additional cells, the inverse of $\zeta_{10}^{(1)}(x_t)$ is, using (2.30), given by

$$\left(\zeta_{10}^{(1)}\right)^{-1}(\rho) = \begin{cases} \text{undefined} & \rho > \rho_0 \\ \left(1 - \frac{\rho}{\rho_0}\right)\Delta x + x_j & 0 \leq \rho \leq \rho_0 \end{cases}. \tag{2.37}$$

Unfortunately, a condition for (2.31) to be solvable for the target positions (this includes the case where $(\zeta_{10}^{(1)})^{-1}(\rho)$ is undefined) for a given grouping of initial particles was not found. Therefore, the final algorithm tries each particle group, including a certain amount of its permutations, and simply skips the group if a solution can not be found. The solution for problems with fewer spatial dimensions is straightforward:

- For 2D, the problem reduces to a $\mathbb{R}^{2\times 2}$ matrix that can be solved more easily. This also means that only

  $$T = 2 \rightarrow N_{\text{min}}^{\text{sort}} = 3 \cdot 2 = 6 \tag{2.38}$$

  initial particles are needed.

- For 1D, the problem reduces to a single equation. This means that only

  $$T = 1 \rightarrow N_{\text{min}}^{\text{sort}} = 3 \cdot 1 = 3 \tag{2.39}$$

  initial particles are needed.

$N_{\text{min}}^{\text{sort}}$ was introduced in chapter 2.2.3.

Second-order form factor

In the case of second-order form factors, current will be deposited on the 27 grid-points shown in Fig. 2.7. It will be shown that this inevitably leads to non-linear equations that can not be solved analytically, and that may potentially take some time to be solved numerically. Since there are

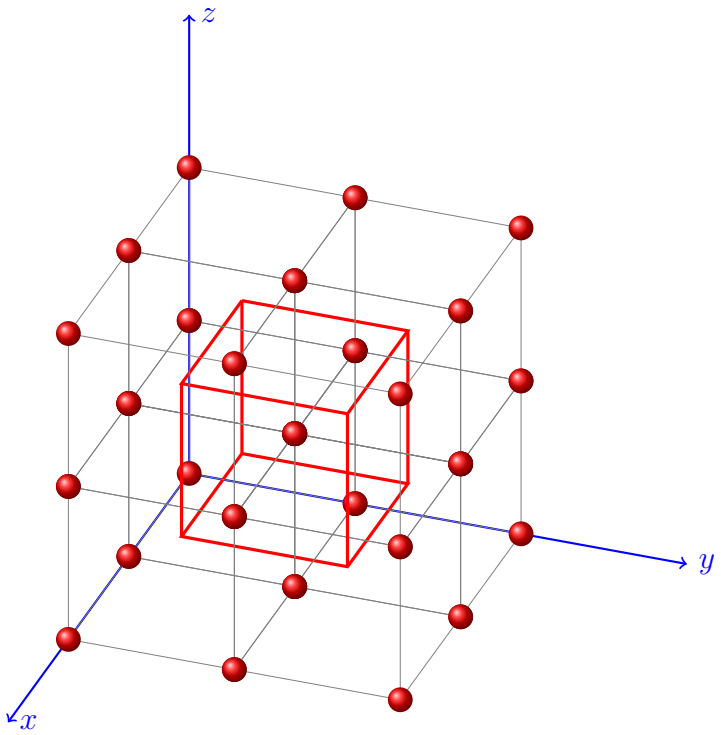

Fig. 2.7 – Illustration of the charge deposition for second-order particle shapes. Particles inside the red cube deposit charge on each of the 27 grid-points marked with red balls. For clarity, the coordinate definitions have been omitted in this figure. They can be found in Fig. 2.6. The smallest coordinate of the red cube is given by $\left(\frac{1}{2}\frac{1}{2}\frac{1}{2}\right)$ and its largest coordinate is given by $\left(\frac{3}{2}\frac{3}{2}\frac{3}{2}\right)$, the center of the red cube is at $\left(111\right)$.

three layers of cells that are affected with current by movement of particles, (2.14) can be written as:

$$j_{-\frac{1}{2}kl} = \frac{\Delta x}{\Delta t}\Delta\rho_{-1kl} \tag{2.40}$$

$$j_{\frac{1}{2}kl} = j_{-\frac{1}{2}kl} + \frac{\Delta x}{\Delta t}\Delta\rho_{0kl} \tag{2.41}$$

$$= \frac{\Delta x}{\Delta t}\left(\Delta\rho_{-1kl} + \Delta\rho_{0kl}\right) \tag{2.42}$$

$$j_{\frac{3}{2}kl} = j_{\frac{1}{2}kl} + \frac{\Delta x}{\Delta t}\Delta\rho_{1kl} \tag{2.43}$$

$$= \frac{\Delta x}{\Delta t}\left(\Delta\rho_{-1kl} + \Delta\rho_{0kl} + \Delta\rho_{1kl}\right) \tag{2.44}$$

Since the current deposition scheme conserves the total charge, the following will automatically be true (as before, no particle will be allowed to leave the initial cell as a result of the merging procedure):

$$j_{\frac{3}{2}kl} = 0 \quad \Leftrightarrow \quad \Delta\rho_{-1kl} + \Delta\rho_{0kl} + \Delta\rho_{1kl} = 0 \tag{2.45}$$

This equation holds for every particle by itself as well. The proposed method needs to ensure that (2.40) - (2.44) are all zero for every $(k,l) \in \{-1,0,1\}^2$. This requires solving 27 equations. These 27 equations correspond to three equations of type (2.20) and are found by considering this type of equation with $(j,k,l) \in \{-1,0,1\}^3$ ($K = L = 3$). Nine of these equations are solved by (2.45). This leaves 18 equations. Nullifying the first slab, $j = -1$, requires solving one system of linear equations of type (2.20). This sets (2.40) to zero and solves another nine equations given by

$$\sum_{t=0}^{T-1} \zeta_{1-1}^{(2)}(x_t) \sum_{g=0}^{G-1} w_{Gt+g} \zeta_{2k}^{(2)}(y_{Gt+g}) \, \zeta_{3l}^{(2)}(z_{Gt+g}) = \sum_{i=0}^{N-1} w_i \zeta_{1-1}^{(2)}(x_i) \, \zeta_{2k}^{(2)}(y_i) \, \zeta_{3l}^{(2)}(z_i), \tag{2.46}$$

which can be written as a system of linear equations

$$\mathbf{A} \overrightarrow{\zeta_{1-1}^{(2)}(x)} = \overrightarrow{b_{-1}}, \tag{2.47}$$

where $\mathbf{A} \in \mathbb{R}^{9 \times T}$, $\overrightarrow{\zeta_{1-1}^{(2)}(x)} \in \mathbb{R}^T$ and $\overrightarrow{b_{-1}} \in \mathbb{R}^9$ are given by

$$A_{rt} = \sum_{g=0}^{G-1} w_{Gt+g} \zeta_{2k}^{(2)}(y_{Gt+g}) \, \zeta_{3l}^{(2)}(z_{Gt+g}) \tag{2.48}$$

$$\overrightarrow{\zeta_{1-1}^{(2)}(x)} = \begin{pmatrix} \zeta_{1-1}^{(2)}(x_0) & \dots & \zeta_{1-1}^{(2)}(x_{T-1}) \end{pmatrix} \tag{2.49}$$

$$b_{-1r} = \sum_{i=0}^{N-1} w_i \zeta_{1-1}^{(2)}(x_i) \, \zeta_{2k}^{(2)}(y_i) \, \zeta_{3l}^{(2)}(z_i) \tag{2.50}$$

$$\text{with} \quad r\colon \{1,\dots,9\} \to \{-1,0,1\}^2, r \mapsto (k,l). \tag{2.51}$$

The variable r maps onto the nine points in the first slab. It follows, that in order to make this a solvable equation it is required that $T \geq 9$ (a minimum of nine degrees of freedom for nine linear equations). As mentioned in the first-order particle shape section, the energy and momentum conservation formulas, derived in chapter 2.4.2, require at least 3 initial particles for each merge operation. Therefore, the full merge operation requires a minimum of $3 \cdot 9 = 27$ initial particles. The shape factor for second-order particle shapes is given by [26, (4.139)]

$$S_1^{(2)}(x) = \rho_0 \begin{cases} 0 & |x| > \Delta x \\ 1 - \frac{|x|}{\Delta x} & |x| \leq \Delta x \end{cases}. \tag{2.52}$$

The shape factors $S_2^{(2)}$ and $S_3^{(2)}$ are defined by substituting Δx for Δy and Δz, respectively. The integrated particle shape function ζ for a second-order particle shape is then given by (using (2.13)) [26, (4.140)]:

$$\zeta_{1j}^{(2)}(x) = \rho_0 \begin{cases} 0 & |x - x_j| > \frac{3\Delta x}{2} \\ \frac{1}{2}\left(\frac{3}{2} - \frac{|x-x_j|}{\Delta x}\right)^2 & \frac{\Delta x}{2} < |x - x_j| \leq \frac{3\Delta x}{2} \\ \frac{3}{4} - \frac{|x-x_j|^2}{\Delta x^2} & |x - x_j| \leq \frac{\Delta x}{2} \end{cases} \tag{2.53}$$

The integrated particle shape functions $\zeta_{2k}^{(2)}$ and $\zeta_{3l}^{(2)}$ are found by substituting Δx for Δy and Δz, respectively, and x_j for y_k and z_l, respectively. Since, in the $j = -1$ case, the second row of this piecewise definition is used, the solution to (2.47) can be used to identify the correct final positions of the particles. Thereby, the first and the last slab is set to zero. Unfortunately, this leaves the middle slab carrying current. Since the form factors in the first and the middle slab are not linearly dependent (cf. to the middle and last row of (2.53)), nullifying the first slab will not guarantee anything about the value of the second slab. Permutation of the nullified $\Delta\rho$ (to a different j) will not ameliorate this situation:

- Setting $\Delta\rho_{-1kl} = 0 \Rightarrow j_{-\frac{1}{2}kl} = 0 \wedge j_{\frac{3}{2}kl} = 0$
- Setting $\Delta\rho_{0kl} = 0 \Rightarrow j_{\frac{3}{2}kl} = 0\,(\wedge \Delta\rho_{-1kl} = -\Delta\rho_{1kl})$
- Setting $\Delta\rho_{1kl} = 0 \Rightarrow j_{\frac{1}{2}kl} = 0 \wedge j_{\frac{3}{2}kl} = 0$

This means that in order to guarantee divergence-free merging in the case of second-order form factors, one would have to solve

$$\mathbf{A}\overrightarrow{\zeta_{1-1}^{(2)}(x)} = \overrightarrow{b_{-1}} \quad \wedge \quad \mathbf{A}\overrightarrow{\zeta_{10}^{(2)}(x)} = \overrightarrow{b_0}, \tag{2.54}$$

with the definitions given in (2.48) - (2.51). The definition of $\zeta_{1j}^{(2)}$, given in (2.53), makes this a system of 18 coupled non-linear equations for the final coordinates x_t. It is very expensive to solve three of these (one per dimension) per single merge operation. There is a high likelihood that an increased number of final and therefore initial particles is needed as well. This would entail more than 27 initial particles. 27 particles per cell is already a very expensive prospect for many applications.

Using a different nonlinear form factor does not remedy the situation, as the charge deposition in neighboring slabs always brings an extra additive constant in the argument of ζ, making the slabs independent. Switching to first-order form factors and using a broader function does not increase the order of the calculation.

Because of these considerations all test cases and results in this thesis use first-order particle form factors. A simple way of getting an approximate solution when using second-order form factors, is to use the algorithm given in the first-order form factor section for second-order particles. This will introduce unphysical divergence in the electric fields, but should nonetheless be a good approximation, keeping these errors minimal.

Permutations

The distribution of initial particles to final positions is not uniquely defined. The initial particles must be grouped together, to form the new and smaller number of particles. The specific choice of grouping can make the underlying equations solvable or unsolvable. Another thing to consider

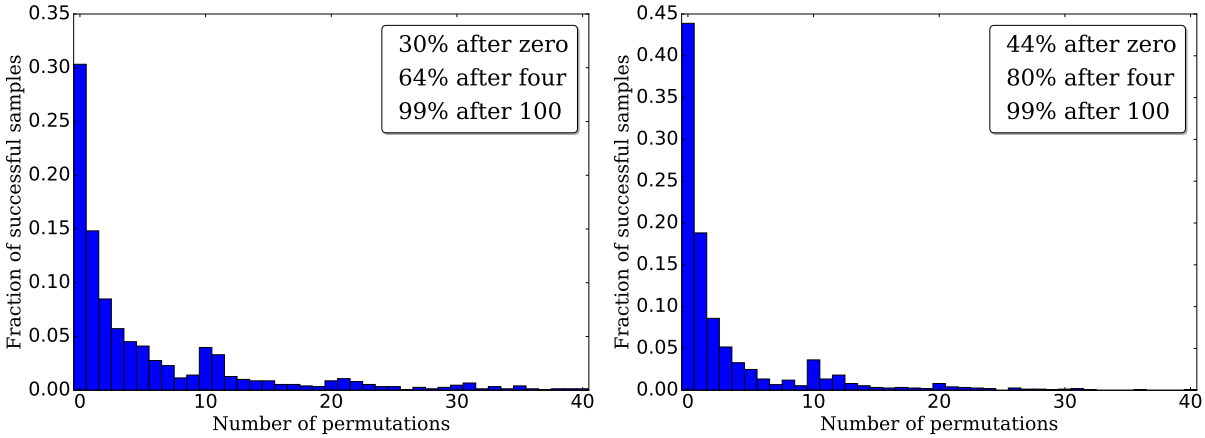

Fig. 2.8 – Relative frequency distribution of required permutations to successfully find a grouping with a viable target location. The search was performed in 3D on randomly distributed particles using a randomized position and weight, occupying 20 % of configuration space and having randomized weights between $0.5 < w < 1.5$. The operation for the right figure applied presorting in the x-direction.

is to presort the particles spatially before assigning them to their final positions. This will bring particles together that are close to each other in a certain direction. A scheme to test out different permutations, with and without sorting, has been implemented.

Multiple different cases where tested and the resulting statistics were evaluated. 12 particles where set up in 3D space, using randomized locations. The accessible space inside the cell for

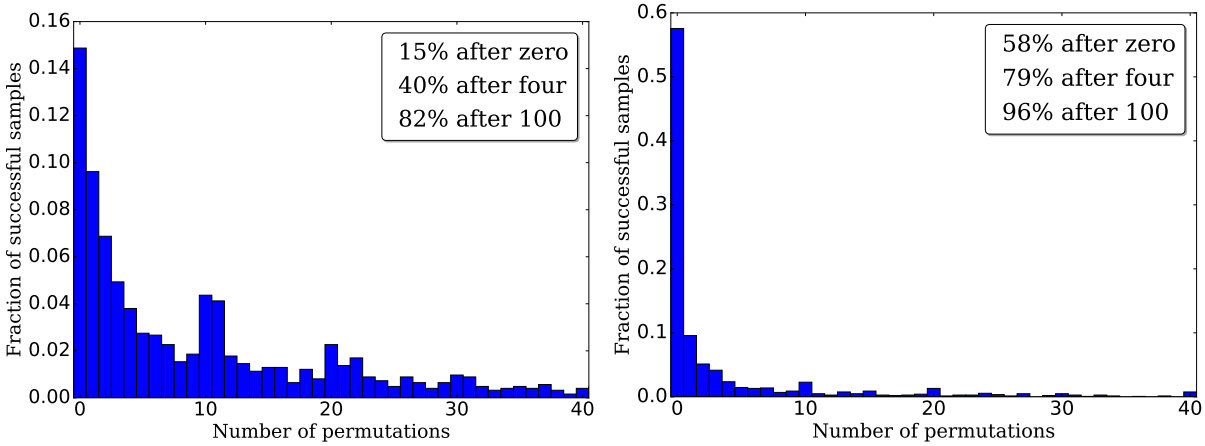

Fig. 2.9 – Relative frequency distribution of required permutations to successfully find a grouping with a viable target location. The search was performed in 3D on randomly distributed particles using a randomized position and weight, occupying 80 % of configuration space and having randomized weights between $0.5 < w < 1.5$. The operation for the right figure applied presorting in the x-direction.

this randomization has been varied between two values, 20 % (Fig. 2.8) and 80 % (Fig. 2.9). The figure on the right in each of these figures uses presorting in the x-direction, while the figure on the left does not. The weights were randomized between $0.5 < w < 1.5$. Using the above first-order form factor algorithm a search for 4 final locations was performed. If a position could not be found, a permutation of the same particle configuration is tested, up to a maximum amount of 100 permutations. This does not mean that it is indeed unsolvable as the total amount of permutations is given by

$$\binom{12}{3} \cdot \binom{9}{3} \cdot \binom{6}{3} \cdot \binom{3}{3} = 369600, \quad \text{where} \quad \binom{\bullet}{\bullet} \quad \text{is the binomial coefficient.} \tag{2.55}$$

In order to increase the statistical significance, the whole process was repeated 1500 times.

The results very clearly show that presorting is very beneficial. In both cases of configuration space accessibility, a solvable configuration was found after fewer permutations. This was even more true for the case that allowed for more configuration space to be accessed (Fig. 2.9), where the amount of successful operations after zero permutations jumped from 15 % to 58 %. In the case of zero permutations only the initial configuration is tested. Additionally, it is found that trying a minimal amount of only four permutations increases the odds of finding a successful particle distribution significantly. In all four cases, independent of the size of the accessible configuration space and whether presorting was applied or not, the odds of successfully finding a solvable distribution increases quite meaningfully. In most cases the odds even double, when trying four permutations instead of zero. Another very promising result is that a solvable permutation is found in almost all cases. Only a very small amount of configurations are abandoned after the maximum of 100 permutations.

Applying the above findings, all test cases in this dissertation use presorting and try five different permutations of each particle group before skipping it.

More sophisticated methods, for example trying to minimize the distance between merge partners, are found to not be particularly effective and need a significant amount of computational resources. From the figures, it is apparent that presorting can be very beneficial in speeding up the merge process. In order to speed-up the algorithm, the matrices and vectors of (2.31) for the permutations can be found using the Steinhaus-Johnson-Trotter-algorithm in conjunction with a numerically stable variation of the Sherman-Morrison-Woodsbury-Formula, instead of calculating them from scratch.

2.3.2 Determination of final momenta

Chapter 2.2 resulted in a list of particle groups with certain properties: all particles in each group i) are of the same kind ii) reside in the same cell in configuration space, iii) have similar weights and finally iv) fulfill customizable requirements regarding their closeness in momentum space. Each group contains an amount of particles $N \in [N_{\text{min}}^{\text{sort}}, N_{\text{max}}^{\text{sort}}]$. Particles in each group represent

merge or split targets, since they fulfill the configured closeness criteria. The goal is to achieve energy and momentum conservation for each distinct group. This way, the parameters governing the previously explained grouping algorithm fulfill their stated purpose of providing a lever for controlling the fidelity of the merge or split operation. The following algorithm conserves energy and momentum and has a low disturbance effect on the momentum distribution.

From now on, each group inside this list is called a cluster of particles. In the previous chapter 2.3.1, the algorithm took this list of clusters and, for each distinct cluster, gathered the particles inside it on a number of locations T. On each of these locations G particles have been gathered (c.f. (2.18)). The initial group of particles with size G on a certain location is called $\mathcal{I}_g$. The final group will consist of exactly two particles and is called $\mathcal{F}_g$.

If a fixed merge factor Q (2.5) is desired, it is helpful to subdivide each cluster into collections of groups. These collections are called ranges. Each range contains T groups and is the result of a single application of the location finding algorithm (chapter 2.3.1). Without the notion of ranges, all particles in a cluster must be merged/split. If a constant merge factor Q is desired, the sorting algorithm needs to be configured in a way as to result in $N_{\text{initial}} = QN_{\text{final}}$ particles. If the sorting process produces more particles, the overhanging particles can not be processed, if it produces fewer particles, the whole cluster can not be processed. Since a binary tree is used to produce the initial particles, it therefore becomes unlikely to find suitable groups of particles. Allowing for ranges relaxes this condition, since it enables larger groups of particles to be eligible as well. If no fixed merge factor is desired, the introduction of ranges is not needed. In this case, the resulting merge factor can then fall between $N^{\text{sort}}_{\text{max}}/N_{\text{final}}$ and $N^{\text{sort}}_{\text{min}}/N_{\text{final}}$. Since a fixed merge factor is essential to minimizing the amount of different weight species, the following considerations will include the notion of ranges.

On the one hand, the more groups in each cluster, the more faithful the final momenta are to the previous momenta in the cluster. This will be made clear in the final energy conservation step of this algorithm. Having more particles in a cluster enables the creation of more particle groups. On the other hand, it is clear that a finer clustering, i.e. fewer particles per cluster, will also contribute to the faithfulness in momentum space, as this corresponds to a less granular particle grouping, i.e. more restricting sorting parameter $N^{\text{sort}}_{\text{max}}$. These two distinct and opposing effects will be investigated after the explanation of the algorithm itself in this chapter.

In the following some examples for the different parameter configurations are given. If, for example, one choses $Q = 2$, i.e. all particle merging processes will decrease the number of particles in a merge cluster by a factor of 2, and if the underlying problem is three dimensional, and the merge cluster contains $N^{\text{sort}}_{\text{min}} = N^{\text{sort}}_{\text{max}} = 16$ (see chapter 2.2.3) particles, 4 final groups would be processed. One could increase the number of groups by setting $N^{\text{sort}}_{\text{max}} = 32$, which could generate up to 8 final groups. As mentioned before, the collection of groups of a particular cluster are called a range. The first of the above cases would constitute one range, the second one comprises two. It is not sensible to set $N^{\text{sort}}_{\text{max}} = k * 16, k \in \mathbb{N}^+$ since this would rarely produce additional groups due to the sorting process, which will most likely not return exact multiples of 16. It is more sensible to set $N^{\text{sort}}_{\text{max}} = k * 16 + 8, k \in \mathbb{N}^+$. The same holds true for the other

dimensions using $N_{\text{min}}^{\text{sort}} = 8, N_{\text{max}}^{\text{sort}} = k*8+4, k \in \mathbb{N}^+$ for 2D and $N_{\text{min}}^{\text{sort}} = 4, N_{\text{max}}^{\text{sort}} = k*4+2, k \in \mathbb{N}^+$ for 1D (for $Q = 2$).

Fig. 2.10 gives an overview of the momentum determining algorithm. The algorithms for "choose from $\mathcal{I}_\bullet$" and "adapt to conserve E" are given in the following under the headings **Random selection of initial momenta values** and **Adaptation of chosen momenta to conserve energy using the original standard deviation**, respectively. This algorithm conserves the total momentum between each initial group $\mathcal{I}_g$ and their final two particles in $\mathcal{F}_g$. In contrast, $\mathcal{I}_g$ and $\mathcal{F}_g$ do not have the same total energy. The subscript g enumerates all groups. The algorithm will redistribute the energy over all groups in a cluster in order to enable the possibility of an increased fidelity in momentum space (c.f. ϵ^{spare} in Fig. 2.10). Nonetheless, the total energy of each cluster is conserved. This fits the description of the sorting parameters, as the result of the sorting process, the clusters, define the degree of energy and momentum fidelity. This is the crucial idea, the key component to enabling an increased re-usage of initial momentum values, which is the best way of preventing a perturbation in the momentum distribution. Not re-using momentum values risks the development of momentum configurations that favor mean values and tend to inhibit non-linear effects. The present algorithm under-girds this effort by a novel method of fixing energy mistakes by adapting the momenta using the original standard deviation of a group.

First, the algorithm needs to calculate certain values for a given list of groups of a cluster (i, j loop over all particles in the cluster, i_g loops over all particles in group g, g loops over all groups, subscript i, j denote values corresponding to single particles, subscript g denotes a value corresponding to a group and bc is short for barycenter and $\hat{e}_k$ are the unit vectors in momentum space):

$$w = \Sigma_i w_i \tag{2.56}$$

$$w_g = \Sigma_{i_g} w_{i_g} \tag{2.57}$$

$$(\vec{\sigma} \cdot \hat{e}_k)^2 = \frac{N}{N-1} \frac{1}{w} \Sigma_i w_i \left((\vec{p}_i - \Sigma_j w_j \vec{p}_j) \cdot \hat{e}_k \right)^2 \tag{2.58}$$

$$\vec{p}_g^{\,\text{bc}} = \frac{1}{w_g} \Sigma_{i_g} w_{i_g} \vec{p}_{i_g} \tag{2.59}$$

$$\epsilon_g^{\text{bc}} = w_g \sqrt{1 + \left| \vec{p}_g^{\,\text{bc}} \right|^2} \tag{2.60}$$

$$\epsilon^{\text{spare}} = \Sigma_i w_i \sqrt{1 + |\vec{p}_i|^2} - \Sigma_g \epsilon_g^{\text{bc}} \tag{2.61}$$

The unbiased sample variance $\vec{\sigma}$ is calculated following [31] in a stable and incremental fashion. The final values are denoted by a prime. Each group $\mathcal{I}_g$ will produce a group $\mathcal{F}_g$ with two final particles denoted with subscripts 1 and 2. Their respective weights are given by:

$$w'_{g1} = \frac{w_g}{2} \tag{2.62}$$

$$w'_{g2} = \frac{w_g}{2} \tag{2.63}$$

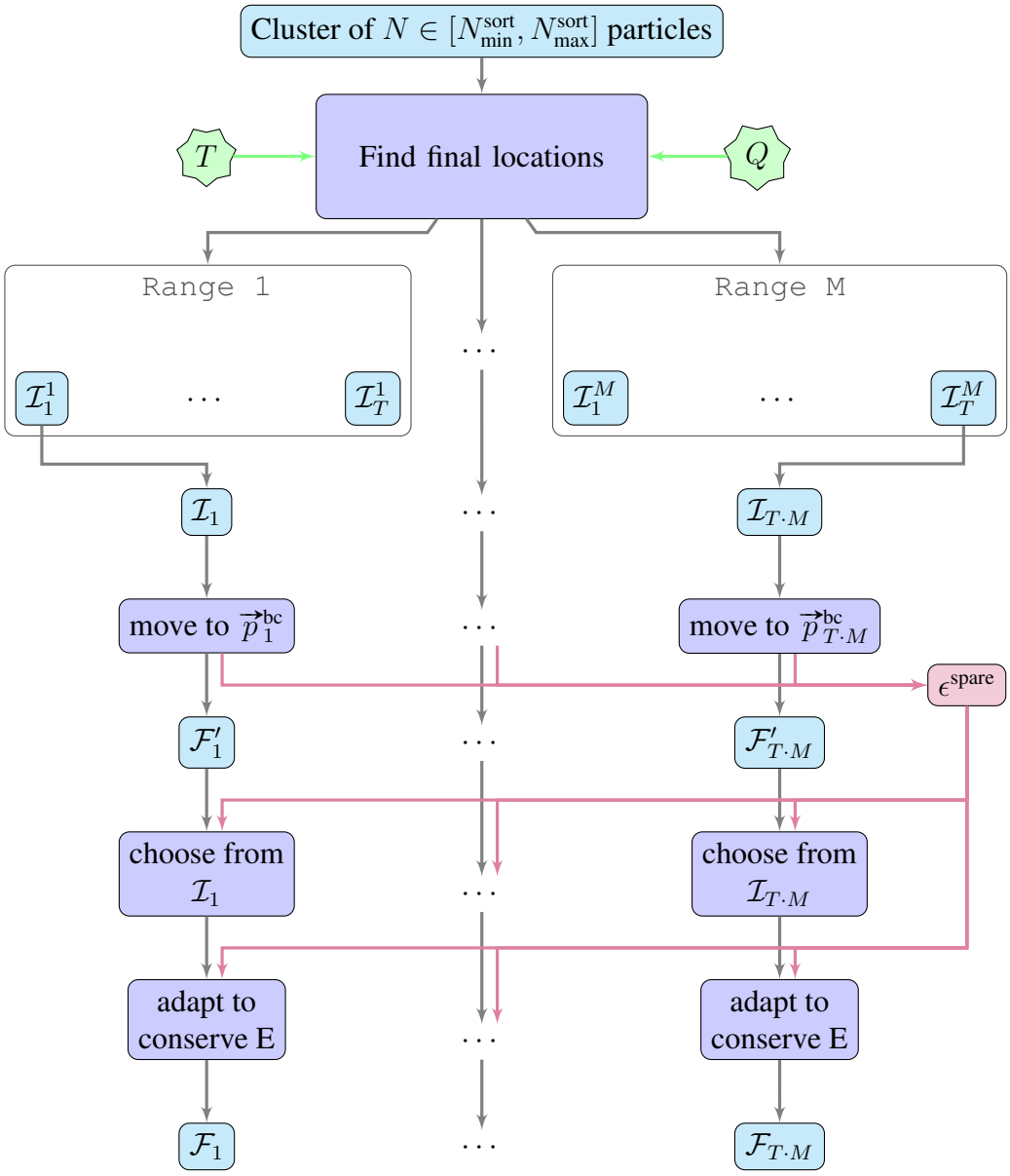

Fig. 2.10 – Flow diagram for the merging algorithm. Light-blue boxes stand for groups of particles. $\mathcal{I}_\bullet^\bullet$ and $\mathcal{I}_\bullet$ are each $T \cdot M$ groups of G initial particles, where for each group, all G particles share a location. $\mathcal{F}'_\bullet$ and $\mathcal{F}_\bullet$ are $T \cdot M$ groups of two final particles, where for each group, all two particles share a location. Light-purple boxes stand for algorithms, which transform the position or momenta of particles. The "Find final locations" algorithm is detailed in chapter 2.3.1. All other algorithms are detailed in chapter 2.3.2. The green star boxes give significant parameters. The light-red box contains a certain amount of energy. Grey arrows signify particle streams from groups or algorithms into other groups or algorithms. Green arrows signify parameters being used in algorithms. Light-red arrows signify streams of energy. M is the number of ranges. T is the number of final locations of each range (c.f. (2.31) and (2.47)). $Q = {}^{N_{\text{initial}}}/_{N_{\text{final}}}$ is the merge factor.

This choice keeps the number of weight species to a minimum. The final momentum for the two final particles in each group $\mathcal{F}_g$ is given by

$$\vec{p}\,'_{g1} = \vec{p}^{\,\text{bc}}_g + \vec{\Pi}_g + \xi_g \vec{\sigma}_{\text{mod}} \quad \text{and} \tag{2.64}$$

$$\vec{p}\,'_{g2} = \vec{p}^{\,\text{bc}}_g - \vec{\Pi}_g - \xi_g \vec{\sigma}_{\text{mod}}, \tag{2.65}$$

where

$$\vec{\sigma}_{\text{mod}} = \begin{pmatrix} \lambda_x \sigma_z & \lambda_y \sigma_y & \lambda_z \sigma_z \end{pmatrix}, \tag{2.66}$$

with $\lambda_i \in \{-1,1\}$ randomly and $\sigma_k = \vec{\sigma} \cdot \hat{e}_k$. The values of $\vec{\Pi}_g$ and ξ_g will be determined in the next two sections. Constructing $\vec{\sigma}_{\text{mod}}$ in this way makes sure that there is no preferential direction in momentum space for the final particles in relation to their standard deviation. From these definitions, the conservation of momentum in each group is guaranteed:

$$w'_{g1}\vec{p}\,'_{g1} + w'_{g2}\vec{p}\,'_{g2} = 2\frac{w_g}{2}\vec{p}^{\,\text{bc}}_g = 2\frac{w_g}{2}\frac{1}{w_g}\Sigma_{i_g} w_{i_g}\vec{p}_{\,i_g} = \Sigma_{i_g} w_{i_g}\vec{p}_{\,i_g} \tag{2.67}$$

Additionally, the construction of this algorithm guarantees that the dimensionality of the momentum space is conserved. Since the final momenta are constructed using either initial momenta or barycenter momenta that are only adapted in the direction of the initial standard deviation, the final momenta occupy the same momentum space as the initial momenta.

Random selection of initial momenta values

This will determine $\vec{\Pi}_g$. The algorithm will randomly chose momenta from the list of initial momenta while conserving the total momentum. This will be done individually for each momentum dimension. The total energy will be conserved in the second step. This first step needs to comply to certain restrictions in order to enable the second step to conserve the total energy. The first thing to calculate is the momentum space range per dimension that is accessible for the final particles. In order to preserve the momentum space distribution every particle should be close to the mean, or barycenter, momentum of each group. Therefore, only the spare energy ϵ^{spare} will be redistributed to the final particles. The accessible momentum range is determined by this excess energy. From all initial momenta, which lie in this accessible range, a momentum is chosen randomly for one of the final particles. The second final particle is set in order to achieve local momentum conservation, c.f. to (2.64) and (2.65). The value

$$\epsilon_g = \frac{2}{w_g}\left(\epsilon^{\text{spare}} + \epsilon^{\text{bc}}_g - \epsilon^{\text{spend}}\right) \tag{2.68}$$

is the normalized energy available to group g with

$$\epsilon^{\text{spend}} = \Sigma_g^{\vec{\Pi}_g \text{defined}} \left(w'_{g1}\sqrt{1 + \left|\vec{p}^{\,\text{bc}}_g + \vec{\Pi}_g\right|^2} + w'_{g2}\sqrt{1 + \left|\vec{p}^{\,\text{bc}}_g - \vec{\Pi}_g\right|^2} \right) \tag{2.69}$$

being the energy spend by randomly selecting initial momenta in already finished groups. The algorithm's goal is to find momenta in the pool of all momenta of all particles in $\mathcal{I}_g$ for which $\epsilon_g \geq 0$. Because of this attribute of the algorithm and since $\epsilon^{\text{spend}} = 0$ for the first group, it holds that $\epsilon_g \geq 0$ at all times. In order to determine the momentum range that is accessible through ϵ_g the solution Δp to

$$\epsilon_g = \sqrt{\epsilon_1^{\text{oc}} + \left(p_g^{\text{bc}} + \Delta p\right)^2} + \sqrt{\epsilon_2^{\text{oc}} + \left(p_g^{\text{bc}} - \Delta p\right)^2} \tag{2.70}$$

is needed. It is given by

$$\begin{aligned} \Delta p_{1,2} = \frac{1}{\epsilon_g^2 - 4\left(p_g^{\text{bc}}\right)^2} \Bigg[& \left(\epsilon_1^{\text{oc}} - \epsilon_2^{\text{oc}}\right) p_g^{\text{bc}} \\ & \pm \frac{1}{2}\epsilon_g \sqrt{\left(\epsilon_g^2 - 4\left(p_g^{\text{bc}}\right)^2 - \left(\epsilon_1^{\text{oc}} + \epsilon_2^{\text{oc}}\right)\right)^2 - 4\epsilon_1^{\text{oc}}\epsilon_2^{\text{oc}}} \Bigg], \end{aligned} \tag{2.71}$$

where p_g^{bc} is a specific component of $\vec{p}_g^{\,\text{bc}}$ and ϵ_i^{oc} are the energies of the other components (oc) plus the rest energy of the two final particles (a formula is given further below). The values $\Delta p_{1,2}$ are ordered in a way so that $\Delta p_1 \leq \Delta p_2$.

The second derivative of (2.70) with respect to Δp is given by

$$\frac{\partial^2 \epsilon_g}{\partial \Delta p^2} = \frac{\epsilon_1^{\text{oc}}}{[\epsilon_1^{\text{oc}} + (p_g^{\text{bc}} + \Delta p)^2]^{\frac{3}{2}}} + \frac{\epsilon_2^{\text{oc}}}{[\epsilon_2^{\text{oc}} + (p_g^{\text{bc}} - \Delta p)^2]^{\frac{3}{2}}}. \tag{2.72}$$

It holds that

$$\frac{\partial^2 \epsilon_g}{\partial \Delta p^2} \geq 0 \quad \forall\, \Delta p \in \mathbb{R}, \tag{2.73}$$

which, coupled with the fact that (2.70) only has a maximum of two solutions (c.f. (2.71)) and that

$$\lim_{\Delta p \to \pm\infty} \left[\sqrt{\epsilon_1^{\text{oc}} + \left(p_g^{\text{bc}} + \Delta p\right)^2} + \sqrt{\epsilon_2^{\text{oc}} + \left(p_g^{\text{bc}} - \Delta p\right)^2} \right] = \infty, \tag{2.74}$$

leads to the conclusion that

$$\sqrt{\epsilon_1^{\text{oc}} + \left(p_g^{\text{bc}} + p\right)^2} + \sqrt{\epsilon_2^{\text{oc}} + \left(p_g^{\text{bc}} - p\right)^2} \leq \epsilon_g \quad \forall\, \Delta p_1 \leq p \leq \Delta p_2. \tag{2.75}$$

Equation (2.70) is a deformed parabola with a lower minimum (i.e. it is opening upwards, in the positive ϵ_g direction).

This defines the accessible momentum ranges for the two final particles. The momentum range needed to be searched for eligible initial particle momenta is given by

$$\mathcal{A} = [p_g^{\text{bc}} - \Delta p_2, p_g^{\text{bc}} - \Delta p_1] \cup [p_g^{\text{bc}} + \Delta p_1, p_g^{\text{bc}} + \Delta p_2], \tag{2.76}$$

for a specific dimension.

Using these formulas, the value of $\vec{\Pi}_g$ can be determined for each group. First, a random momentum dimension order k, l, m is determined. In the following, all scalar values of vector entities are the respective components for the currently processed dimension. Then, the following steps are performed successively for each dimension in a group and for each group $\mathcal{I}_g$ in a cluster:

1. Apply the above formulas to find the accessible momentum range $\mathcal{A}$, using (2.76). The specific values of ϵ_1^{oc}, ϵ_2^{oc}, p_g^{bc} and Π_g for each iteration are given after this list.

2. Find the set of all momenta of particles in $\mathcal{I}_g$, which satisfy

$$\{p \,|\, p \in \mathcal{A}\}. \tag{2.77}$$

   If $\{p \,|\, p \in \mathcal{A}\} = \emptyset$ set $\vec{\Pi}_g = \vec{0}$ (this fixes all components of $\vec{\Pi}_g$ and ends the loop over the momentum dimensions).

3. Choose one particle momentum p_j at random from this set and

   - if $p_j \in \mathcal{A} - [p_g^{\text{bc}} - \Delta p_2, p_g^{\text{bc}} - \Delta p_1]$ set

$$\Pi_g = p_j - p_g^{\text{bc}}, \tag{2.78}$$

     i.e. the first final particle gets this momentum value (refer to (2.64)),

   - or if $p_j \in \mathcal{A} - [p_g^{\text{bc}} + \Delta p_1, p_g^{\text{bc}} + \Delta p_2]$ set

$$\Pi_g = p_g^{\text{bc}} - p_j, \tag{2.79}$$

     i.e. the second final particle gets this momentum value (refer to (2.65)),

   - or if $p_j \in [p_g^{\text{bc}} + \Delta p_1, p_g^{\text{bc}} + \Delta p_2] \cap [p_g^{\text{bc}} - \Delta p_2, p_g^{\text{bc}} - \Delta p_1]$ set

$$\Pi_g = p_g^{\text{bc}} - p_j \quad \text{or} \quad \Pi_g = p_j - p_g^{\text{bc}}, \tag{2.80}$$

     randomly, i.e. the first or second final particle gets this momentum value at random.

The construction of ϵ^{spare} (2.61) and the design of this algorithm make sure that most groups will find a momenta in the set of initial momenta. The likelihood does decrease when moving along all the groups in a cluster, but one of the smallest momenta in each group should stay a viable candidate in most cases. It may still happen that a group can not find an initial momentum, since, if a momentum with a large magnitude is chosen for a previous group, it will consume a lot of ϵ^{spare}, leaving only little energy for the later groups in a cluster. This is by design, as choosing outliers from the initial momenta distribution is an integral part of the algorithm and part of its strength.

The values of ϵ_1^{oc}, ϵ_2^{oc} and p_g^{bc} for each dimension iteration as well as the resulting component of $\vec{\Pi}_g$ are given in the following list:

First dimension k

$$\epsilon_1^{\text{oc}} = 1 + \left(\vec{p}_g^{\,\text{bc}} \cdot \hat{e}_l\right)^2 + \left(\vec{p}_g^{\,\text{bc}} \cdot \hat{e}_m\right)^2 \tag{2.81}$$
$$\epsilon_2^{\text{oc}} = \epsilon_1^{\text{oc}} \tag{2.82}$$
$$p_g^{\text{bc}} = \vec{p}_g^{\,\text{bc}} \cdot \hat{e}_k \tag{2.83}$$
$$\Rightarrow \Pi_g = \vec{\Pi}_g \cdot \hat{e}_k \tag{2.84}$$

Second dimension l

$$\epsilon_1^{\text{oc}} = 1 + \left[\left(\vec{p}_g^{\,\text{bc}} + \vec{\Pi}_g\right) \cdot \hat{e}_k\right]^2 + \left(\vec{p}_g^{\,\text{bc}} \cdot \hat{e}_m\right)^2 \tag{2.85}$$
$$\epsilon_2^{\text{oc}} = 1 + \left[\left(\vec{p}_g^{\,\text{bc}} - \vec{\Pi}_g\right) \cdot \hat{e}_k\right]^2 + \left(\vec{p}_g^{\,\text{bc}} \cdot \hat{e}_m\right)^2 \tag{2.86}$$
$$p_g^{\text{bc}} = \vec{p}_g^{\,\text{bc}} \cdot \hat{e}_l \tag{2.87}$$
$$\Rightarrow \Pi_g = \vec{\Pi}_g \cdot \hat{e}_l \tag{2.88}$$

Third dimension m

$$\epsilon_1^{\text{oc}} = 1 + \left[\left(\vec{p}_g^{\,\text{bc}} + \vec{\Pi}_g\right) \cdot \hat{e}_k\right]^2 + \left[\left(\vec{p}_g^{\,\text{bc}} + \vec{\Pi}_g\right) \cdot \hat{e}_l\right]^2 \tag{2.89}$$
$$\epsilon_2^{\text{oc}} = 1 + \left[\left(\vec{p}_g^{\,\text{bc}} - \vec{\Pi}_g\right) \cdot \hat{e}_k\right]^2 + \left[\left(\vec{p}_g^{\,\text{bc}} - \vec{\Pi}_g\right) \cdot \hat{e}_l\right]^2 \tag{2.90}$$
$$p_g^{\text{bc}} = \vec{p}_g^{\,\text{bc}} \cdot \hat{e}_m \tag{2.91}$$
$$\Rightarrow \Pi_g = \vec{\Pi}_g \cdot \hat{e}_m \tag{2.92}$$

After performing the iterations over all three momentum dimensions all components of $\vec{\Pi}_g$ are fixed for this group. The process is then repeated for all groups $\mathcal{I}_g$, with successively smaller ϵ_g, as the value of ϵ^{spend} increases.

Adaptation of chosen momenta to conserve energy using the original standard deviation

This will determine the value of ξ_g. In order to achieve energy conservation the momenta found in the above algorithm (in section **Random selection of initial momenta values**) need to be adapted. The momentum distribution should not, on average, deteriorate too much through this adaptation. The remaining spare energy will be distributed evenly on the final particles. This distribution is done in a way that ensures an approximation to the initial three dimensional standard deviation in momentum space. Since the above algorithm can fail for specific groups (if there is

not enough remaining energy to find any suitable initial momenta), formulas are needed for two cases: i) particles with a randomly chosen initial momentum and ii) particles without that. Each group gets a specific portion of the remaining energy, given by

$$\kappa = \frac{w_g}{w}. \tag{2.93}$$

The following has to be done for all groups in a cluster after the previous step (the one described in section **Random selection of initial momenta values**) has finished for all groups. The energy that will be distributed on the final two particles in each group is given by

$$\epsilon_g = \frac{2}{w_g}\left(\kappa\left(\epsilon^{\text{spare}} - \epsilon^{\text{spend}}\right) + w'_{g1}\sqrt{1 + \left|\vec{p}_g^{\,\text{bc}} + \vec{\Pi}_g\right|^2} + w'_{g2}\sqrt{1 + \left|\vec{p}_g^{\,\text{bc}} - \vec{\Pi}_g\right|^2}\right), \tag{2.94}$$

where ϵ^{spend} (defined in (2.69)) now contains energy from every group. In order to find the maximum viable shift in the direction of the original standard deviation, the following equation needs to be solved for ξ_g,

$$\begin{aligned}\epsilon_g =& \sqrt{1 + \left|\vec{p}_g^{\,\text{bc}} + \vec{\Pi}_g + \xi_g \vec{\sigma}_{\text{mod}}\right|^2} \\ &+ \sqrt{1 + \left|\vec{p}_g^{\,\text{bc}} - \vec{\Pi}_g - \xi_g \vec{\sigma}_{\text{mod}}\right|^2},\end{aligned} \tag{2.95}$$

which gives

$$\xi_g = \frac{\epsilon_g^2(\vec{\Pi}_g \cdot \vec{\sigma}_{\text{mod}}) - 4(\vec{\Pi}_g \cdot \vec{p}_g^{\,\text{bc}})(\vec{p}_g^{\,\text{bc}} \cdot \vec{\sigma}_{\text{mod}}) \pm \sqrt{S}}{D}, \tag{2.96}$$

where

$$\begin{aligned}S =& \left(\epsilon_g^2(\vec{\Pi}_g \cdot \vec{\sigma}_{\text{mod}}) - 4(\vec{\Pi}_g \cdot \vec{p}_g^{\,\text{bc}})(\vec{p}_g^{\,\text{bc}} \cdot \vec{\sigma}_{\text{mod}})\right)^2 \\ &- D\left[\frac{1}{4}\epsilon_g^4 - \epsilon_g^2\left(\left|\vec{\Pi}_g\right|^2 + \left|\vec{p}_g^{\,\text{bc}}\right|^2 + 1\right)\right. \\ &\left. + 4(\vec{\Pi}_g \cdot \vec{p}_g^{\,\text{bc}})^2\right] \quad \text{and}\end{aligned} \tag{2.97}$$

$$D = 4(\vec{p}_g^{\,\text{bc}} \cdot \vec{\sigma}_{\text{mod}})^2 - \epsilon_g^2\left|\vec{\sigma}_{\text{mod}}\right|^2. \tag{2.98}$$

The $\pm$ sign will be set at random. If the randomized search for an initial momentum failed, the group will still be included in the final energy correction scheme. This ensures, that the final particles are not completely equal. For these groups $\vec{\Pi}_g = \vec{0}$ holds, and (2.96) simplifies to

$$\xi_g = \pm\frac{\epsilon_g}{2}\sqrt{\frac{4\left(1 + \left|\vec{p}_g^{\,\text{bc}}\right|^2\right) - \epsilon_g^2}{D}}, \tag{2.99}$$

with D given in (2.98), and the $\pm$ sign chosen at random again.

This finalizes the determination of the momenta of the two final particles. The complete values can now be computed by replacing $\vec{\Pi}_g$ and ξ_g in (2.64) and (2.65) with the determined values. These momenta will guarantee momentum as well as energy conservation. The method detailed in chapter 2.5 can now be used to increase diversity in the final particle locations.

The most expensive operations in the determining formulas are square roots: one per group in (2.60), (2.71) and (2.96) or (2.99), two per group in (2.94), two per group, except for the first group, in (2.69), and one per initial particle in (2.61). The total number of square roots per initial particle is therefore given by

$$\frac{3TM + 4TM - M + TMG}{TMG} = \frac{7}{G} - \frac{1}{TG} + 1. \tag{2.100}$$

For typical values of $T = 4$ and $G = 4$, this formula gives a total of 2.6875 square roots per initial particle.

Effects of the different parameters controlling the particle grouping

A suite of parameters governing the particle grouping has been introduced. The main parameters from chapter 2.2 are $N_{\text{min}}^{\text{sort}}$ and $N_{\text{max}}^{\text{sort}}$, which set limits for the cluster size. The last sections introduced some more parameters, which further group the particles of a cluster. They are

1. the number of ranges in a cluster M,
2. the number of groups in a range T and
3. the number of particles in a group G.

From these, only the value of M can be modified, which also changes the value of G. The number of groups per range is fixed by the location-finding algorithm. For a fixed merge factor Q (2.5), the number of ranges is determined by the number of particles in a cluster, and can therefore not be changed.

The number of particles in a cluster is the main parameter for the particle sorting algorithms. Increasing this value will decrease the fidelity as more particles, in a larger phase space, will be part of a merge process.

When looking at the algorithm developed above, it is clear that more groups in a cluster will permit more momenta to be chosen from initial ones, as a larger pool of spare energy is available. It can also be assumed that the final distribution of energy, in the adaptation to the original standard deviation step, will be less destructive.

The benefits of smaller groups, i.e. smaller values of G, are given by the fact that the algorithm that choses initial momenta does not assign full three dimensional initial momenta. It choses

the new momenta on a per dimension basis. This makes the new momenta inhabit a momentum space formed by the cube of the minimum and maximum initial momenta in each group. Smaller groups counteract this emergence of wrong momenta.

Unfortunately, assigning three dimensional momenta is a highly nonlinear process. Determining the momentum space from which initial momenta may be assigned becomes highly non-trivial. The fastest solution would be to test all initial momenta for their eligibility, but this would involve calculating the energy of the momentum on the other side of the barycenter. The calculation of energies is one of the most expensive operations in this algorithm as it involves a square root.

Another point to consider is the usage of the cluster wide $\vec{\sigma}$ value (2.58) and the usage of the group-wise barycenter (2.59). These get increasingly less accurate for increasing cluster or group sizes.

The main parameters to control fidelity should therefore still be the parameters introduced in chapter 2.2. In the case of a non-constant Q (2.5), the number of ranges M can be a lever to optimize the performance of the algorithm. It is very hard to construct a well-defined analytic measure to derive an optimal value for M. For a fixed Q there are no additional degrees of freedom, other than the ones introduced in chapter 2.2.

2.4 Determination of particle properties after splitting

The algorithm, laid out in chapter 2.2, results in a list of particle groups, with each group consisting of a number of particles between $N_{\text{min}}^{\text{sort}}$ and $N_{\text{max}}^{\text{sort}}$. These particle groups are ordered by weight from smaller to larger, as described in chapter 2.2.2. The groups will now be processed, taking advantage of this order, with the goal of increasing the number of quasi-particles (splitting). In this section the necessary algorithms to perform a splitting operation are detailed. They will increase the number of particles in each distinct group.

2.4.1 Determination of final locations

This is not as big of an issue as in the case of merging since the final particles can just be created at the location of the former particles. In doing this, the charge and current densities on the grid will remain constant, and there is no unphysical divergence. The method detailed in chapter 2.5 can be used to increase diversity in the final particle locations.

2.4.2 Determination of final momenta

Separating a single initial quasi-particle into multiple final quasi-particles is trivially possible. The weight of a quasi-particle is proportional to the amount of real particles per quasi-particle,

given by

$$\frac{n\Delta x\Delta y\Delta z}{N_{\text{percell}}}, \tag{2.101}$$

with n the physical particle density in the cell, Δx, Δy and Δz the grid size of the PIC code and N_{percell} the number of quasi-particles in each cell. Of course, this quasi-particles-as-bunches-of-real-particles view is not supported by the underlying mathematics, but it sometimes helps in modeling the algorithms.

Splitting up a particle is therefore trivially possible by just generating two particles with the exact properties of the initial particle except for their weight, which is set to $w/2$, with w the weight of the initial particle. This keeps the number of weight species low and conserves the total momentum and energy:

$$\vec{P} = w\vec{p} = \frac{w}{2}\vec{p} + \frac{w}{2}\vec{p}, \tag{2.102}$$

$$E = w\sqrt{\vec{p}^2 + 1} = \frac{w}{2}\sqrt{\vec{p}^2 + 1} + \frac{w}{2}\sqrt{\vec{p}^2 + 1}. \tag{2.103}$$

Unfortunately, this will produce two completely equivalent particles.

Using the method detailed in chapter 2.5, the particle position of the generated particles may be safely modified, if first-order particles are used. This can be deemed a sufficient amount of diversity. Diversifying the momentum space as well may still be advantageous, since it adds more variety to the generated particles.

It can also be deemed detrimental, as it messes with the momentum distribution. In the case of setups with fewer than three momentum dimensions, the below adaptation algorithm will even very likely result in final momenta that are not part of the initial momentum space. It may be possible to improve the algorithm to accommodate for those setups. These improvements are not part of this dissertation.

If the physical properties of the system under investigation allow for broad momentum distributions, like in thermal plasmas, increasing the diversity in momentum space can be very beneficial for the signal-to-noise ratio. In contrast, systems with more strict demands for the momentum distribution, like particle accelerator simulations, may suffer from the introduced perturbations.

The algorithm that is used to determine the final momenta for splitting processes is detailed in the following sections. It will generate additional diversity by modifying the final momenta and energies. First, it will be shown that it is impossible to divide a single quasi-particle into multiple distinct quasi-particles, while preserving the total energy and momentum. From this it is concluded that a minimum of two quasi-particles is needed, which can then be split into four quasi-particles. The solution to the problem of generating four different momenta from two initial ones is shown subsequently.

Energy and momentum conservation for splitting $N \geq 1$ quasi-particles into two quasi-particles

The following quantities are needed for the algorithm in this section and the next section,

$$\epsilon_i = \sqrt{p_i^2 + 1}, \tag{2.104}$$

$$\vec{P} = \sum_{i=0}^{N} w_i \vec{p}_i, \tag{2.105}$$

$$\epsilon = \sum_{i=0}^{N} w_i \sqrt{p_i^2 + 1}, \tag{2.106}$$

$$W = \sum_{i=0}^{N} w_i, \tag{2.107}$$

where ϵ_i is the weightless normalized energy of a single particle, $\vec{P}$ is the total momentum of the initial particles, ϵ is the total energy of the initial particles and W is the total weight of the initial particles. All particle attributes for the two new quasi-particles need to be computed in a way so that the total energy, total momentum, total mass and total weight are conserved. This leaves eight degrees of freedom (three for the momenta plus one for the weight for each of the two particles) in the final particles. The final weights and final energies are given by

$$w_1' = w_2' = \frac{W}{2}, \quad \text{and} \tag{2.108}$$

$$\epsilon_1' = \epsilon_2' = \frac{\epsilon}{W}. \tag{2.109}$$

Again, this choice will keep the number of weight species minimal. The prime always refers to the attributes of the final two particles. Using this definition the energy is conserved:

$$\epsilon' = w_1'\epsilon_1' + w_2'\epsilon_2' = \frac{W}{2}\frac{\epsilon}{W} + \frac{W}{2}\frac{\epsilon}{W} = \epsilon \tag{2.110}$$

For momentum conservation $\vec{P}' = w_1'\vec{p}_1' + w_2'\vec{p}_2' = \vec{P} \overset{(2.108)}{\Leftrightarrow} \frac{W}{2}(\vec{p}_1' + \vec{p}_2') = \vec{P}$ needs to hold. This gives the following three equations,

$$(p_{1x}' + p_{2x}') = \frac{2P_x}{W} = a, \tag{2.111}$$

$$\left(p_{1y}' + p_{2y}'\right) = \frac{2P_y}{W} = b, \tag{2.112}$$

$$(p_{1z}' + p_{2z}') = \frac{2P_z}{W} = c, \tag{2.113}$$

leaving three degrees of freedom. According to (2.104) the following two equations must hold:

$$p_{1x}'^2 + p_{1y}'^2 + p_{1z}'^2 = \epsilon_1'^2 - 1 \overset{(2.109)}{=} \left(\frac{\epsilon}{W}\right)^2 - 1 = d \tag{2.114}$$

$$p_{2x}'^2 + p_{2y}'^2 + p_{2z}'^2 = \epsilon_2'^2 - 1 \overset{(2.109)}{=} \left(\frac{\epsilon}{W}\right)^2 - 1 = d \tag{2.115}$$

Equations (2.111) - (2.115) constitute five equations for the six undetermined momentum values. This leaves one degree of freedom, which can be set arbitrarily. It will be randomized in a later step. It is helpful to rotate the coordinate system so that $\widetilde{a} = 2\,|\vec{P}|/W$, $\widetilde{b} = 0$ and $\widetilde{c} = 0$. Variables with $\widetilde{\ }$ reside in the rotated coordinate system. This corresponds to the coordinate system where $\widetilde{P}_y = 0$ and $\widetilde{P}_z = 0$. It holds that $\widetilde{d} = d$. The solution to (2.111) - (2.115) in that system is given by

$$\vec{\widetilde{p}}\,'_1 = \frac{1}{2}\begin{pmatrix} \widetilde{a} \\ \sqrt{-\widetilde{a}^2 + 4d - k} \\ \sqrt{k} \end{pmatrix} \quad \text{and} \tag{2.116}$$

$$\vec{\widetilde{p}}\,'_2 = \frac{1}{2}\begin{pmatrix} \widetilde{a} \\ -\sqrt{-\widetilde{a}^2 + 4d - k} \\ -\sqrt{k} \end{pmatrix}, \tag{2.117}$$

where $0 \leq k \leq 4d - \widetilde{a}^2$. This expression can be written as

$$4d - \widetilde{a}^2 = 4\left(\frac{\epsilon^2}{W^2} - 1\right) - \left(\frac{2|\vec{P}|}{W}\right)^2 = 4\left(\frac{\epsilon^2 - \vec{P}^2}{W^2} - 1\right). \tag{2.118}$$

This gives a condition for the solubility of the problem:

$$4d - \widetilde{a}^2 < 0 \Rightarrow \text{no solutions} \tag{2.119}$$

$$4d - \widetilde{a}^2 = 0 \Rightarrow \text{one solution: equal particles} \tag{2.120}$$

$$4d - \widetilde{a}^2 > 0 \Rightarrow \text{two solutions: different particles} \tag{2.121}$$

In the case of $4d - \widetilde{a}^2 \geq 0$, the two momentum vectors can rotate around a circular plane with a constant total energy. This rotation is determined by the value of k and is shown in Fig. 2.11.

The value of $|\vec{r}\,'|$ in Fig. 2.11 is determined by an "available energy"

$$E_{\text{avail}} = |\vec{r}\,'| \overset{\text{Fig. 2.11}}{=} \sqrt{\vec{p}_1'^2 - \left(\frac{1}{2}\frac{2}{W}\vec{P}\right)^2} \overset{(2.114)}{=} \sqrt{d - \frac{\vec{P}^2}{W^2}} = \sqrt{\frac{\epsilon^2 - \vec{P}^2}{W^2} - 1}, \tag{2.122}$$

which is a measure of the amount of variation of the momenta in the initial particles. This gives a condition for deciding if the problem is solvable,

$$\sqrt{\frac{\epsilon^2 - \vec{P}^2}{W^2} - 1} > 0, \tag{2.123}$$

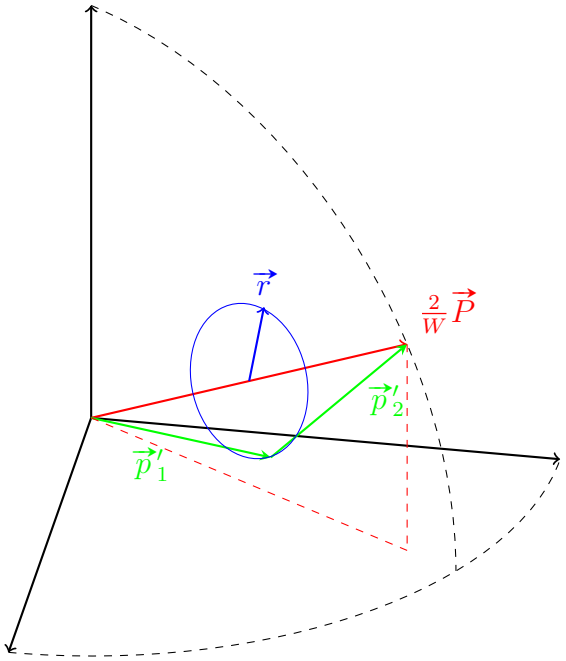

Fig. 2.11 – The two resulting momenta $\vec{p}\,'_1$ and $\vec{p}\,'_2$ in relation to the total initial momentum $\vec{P}$.

which, using (2.118), is easily shown to be the same condition as (2.121). If all initial particles have the same momentum (trivially given in the case of $N = 1$) $E_{\text{avail}} = 0$ and $\vec{p}\,'_1 = \vec{p}\,'_2$. This shows that it is impossible to create two different particles from a single initial particle. Equations (2.116) and (2.117) (with the appropriate back-rotation) and equation (2.108) constitute the solution to the problem of this section. The equations developed here will prove helpful in the next section. The coordinate system rotations can easily be implemented as matrix-vector multiplications, with the original momentum vector determining the transformation matrix.

Energy and momentum conservation for splitting two quasi-particles into four quasi-particles

It was shown in the previous section that a single particle can not be divided into two quasi-particles that are different in momentum and/or energy. This means that, in order to produce new and different quasi-particles (with respect to their momentum), a more sophisticated route must be taken:

1. Choose two particles as initial particles
2. Calculate the total momentum and the total energy
3. Divide the total momentum into two parts, this may be done in a random fashion

4. Produce two particles from each initial particle using the algorithm given in the previous section

The algorithm starts with two initial particles with the attributes w_1, $\vec{p}_1$ and w_2, $\vec{p}_2$. These will be split into four particles, where the first two particles with attributes w_1', $\vec{p}_1'$, w_2' and $\vec{p}_2'$ will be generated at the location of the first initial particle and the second two particles, with attributes w_3', $\vec{p}_3'$, w_4' and $\vec{p}_4'$ will be generated at the location of the second initial particle. This will keep the charge density constant. The weight of the final particles is given by

$$w_1' = w_2' = \frac{w_1}{2}, \tag{2.124}$$

$$w_3' = w_4' = \frac{w_2}{2}, \tag{2.125}$$

keeping the number of weight species to a minimum. As before,

$$\epsilon_1' = \epsilon_2' = \epsilon_3' = \epsilon_4' = \frac{\epsilon}{W}, \tag{2.126}$$

is chosen. The final particles will therefore have the same energy per weight. Getting rid of this assumption will generate more diversity in the final particles. This will not be shown in this dissertation. Instead the algorithm will generate particles with different momenta per weight.

The initial total momentum is then given by $\vec{P} = w_1\vec{p}_1 + w_2\vec{p}_2 = \vec{P}_1 + \vec{P}_2$, with $\vec{P}_1$ and $\vec{P}_2$ the total momentum of the two initial particles. Repeating the analysis of the previous section, but now using these four instead of two final particles, equations (2.111) - (2.115) become

$$w_1'\vec{p}_1' + w_2'\vec{p}_2' + w_3'\vec{p}_3' + w_4'\vec{p}_4' = \vec{P}' = \vec{P}, \tag{2.127}$$

$$|\vec{p}_1'| = \frac{\epsilon^2}{W^2} - 1 = d, \quad |\vec{p}_2'| = \frac{\epsilon^2}{W^2} - 1 = d, \tag{2.128}$$

$$|\vec{p}_3'| = \frac{\epsilon^2}{W^2} - 1 = d, \quad |\vec{p}_4'| = \frac{\epsilon^2}{W^2} - 1 = d. \tag{2.129}$$

From these, equation (2.127) can be written as

$$\frac{w_1}{2}(\vec{p}_1' + \vec{p}_2') + \frac{w_2}{2}(\vec{p}_3' + \vec{p}_4') = \vec{P} \tag{2.130}$$

$$\Leftrightarrow \qquad \vec{P}_1' + \vec{P}_2' = \vec{P}, \tag{2.131}$$

with $\vec{P}_1'$ and $\vec{P}_2'$, the total momentum of the final particles at the first and second location, respectively. This equation can be split up into two problems, which have the same structure as the problem that was solved in the previous section:

$$\vec{p}_1' + \vec{p}_2' = \frac{2\vec{P}_1'}{w_1} \tag{2.132}$$

$$\vec{p}_3' + \vec{p}_4' = \frac{2\vec{P}_2'}{w_2} \tag{2.133}$$

If both of these equations, (2.131), (2.128) and (2.129) hold, the problem is solved, conserving the total energy and momentum, while producing four different final particles.

Equations (2.132) in combination with (2.128), and (2.133) in combination with (2.129), can be solved using the algorithm of the previous chapter. In order to do this, the values of $\vec{P}'_1$ and $\vec{P}'_2$ must be defined. A simple way of guaranteeing (2.131) is given by choosing

$$\vec{P}'_1 = \vec{P}_1 \quad \text{and} \quad \vec{P}'_2 = \vec{P}_2. \tag{2.134}$$

The total energy gets redistributed between the two locations, making four particles with the same energy per weight. This makes it possible to generate different final particles from the initial particles, since the two pairs may not share the same point in space, may have different weights, and, according to the analytics surrounding Fig. 2.11 the randomly chosen k will result in different momenta per pair as well.

A more general approach offers an additional degree of freedom, which can be used to increase the variety in the momentum distribution of the final particles. This brings with it the caveats discussed at the beginning of chapter 2.4.2. This approach is similar to the approach taken in chapter 2.3.2, in that the total momentum of the two groups will be distributed over the two groups. Instead of using (2.134),

$$\vec{P}'_1 = m\vec{P} \quad \text{and} \quad \vec{P}'_2 = (1-m)\vec{P} \tag{2.135}$$

is used. Eq. (2.131) holds and another degree of freedom, in the form of m, is gained. Another advantage is the fact that both problems (2.132) and (2.133) now use the same back-rotation, as they both use the same vector on the right-hand side, which saves on computational resources.

The momentum and the energy are conserved for this process:

$$\begin{aligned} \vec{P}' &= \sum_i w'_i \vec{p}'_i = \frac{w_1}{2}(\vec{p}'_1 + \vec{p}'_2) + \frac{w_2}{2}(\vec{p}'_3 + \vec{p}'_4) \\ &= \vec{P}'_1 + \vec{P}'_2 = m\vec{P} + (1-m)\vec{P} = \vec{P} \end{aligned} \tag{2.136}$$

$$\epsilon' = \frac{w_1}{2}(\epsilon'_1 + \epsilon'_2) + \frac{w_2}{2}(\epsilon'_3 + \epsilon'_4) = w_1\frac{\epsilon}{W} + w_2\frac{\epsilon}{W} = \frac{w_1 + w_2}{W}\epsilon = \epsilon \tag{2.137}$$

Taking advantage of this additional degree of freedom, m can be chosen randomly, which will increase the diversity of the momenta in the system, carrying with it the above mentioned advantages and disadvantages. It is unlikely to randomly choose a value of m that leads to completely equal particles in one pair.

If a randomly determined m is desired, it is important to adhere to certain restrictions for the value of m. As was shown before, there is a prerequisite that ensures total energy and momentum conservation per final pair is possible. The availability of so-called "available energy" (2.122) is a necessity for splitting particles. Only when $E_{\text{avail}} > 0$, the two initial particles are not totally equal. This prerequisite imposes extra constraints on the new variable m.

The distribution of the "available energy" must happen in a way that ensures that the two final pairs each have some of it available. Equation (2.121) gives a criteria for the required amount of "available energy". Using the right-hand side of (2.132), with (2.135), for $\widetilde{a}$ (which is the absolute value of the vector on the right-hand side):

$$4d - \widetilde{a}^2 > 0 \tag{2.138}$$

$$\Leftrightarrow \qquad 4d - \left(\frac{2|m\vec{P}|}{w_1}\right)^2 > 0 \tag{2.139}$$

This is a downward pointing parabola in m, with zero points at

$$m = \pm\frac{w_1\sqrt{d}}{|\vec{P}|}. \tag{2.140}$$

Therefore, all m for which

$$-\frac{w_1\sqrt{d}}{|\vec{P}|} < m < \frac{w_1\sqrt{d}}{|\vec{P}|} \tag{2.141}$$

holds, are solutions to equation (2.139). Equation (2.121) must also hold for the right hand side of (2.133), again with (2.135). This gives another downward pointing parabola, comparable to (2.139). Its solutions m are given by

$$1 - \frac{w_2\sqrt{d}}{|\vec{P}|} < m < 1 + \frac{w_2\sqrt{d}}{|\vec{P}|}. \tag{2.142}$$

Finally, the complete boundaries for m are found to be

$$\max\left(-\frac{w_1\sqrt{d}}{|\vec{P}|}, 1 - \frac{w_2\sqrt{d}}{|\vec{P}|}\right) < m < \min\left(\frac{w_1\sqrt{d}}{|\vec{P}|}, 1 + \frac{w_2\sqrt{d}}{|\vec{P}|}\right). \tag{2.143}$$

These are the boundaries for the value of m in (2.132) and (2.133), with (2.135). The actual value for m is chosen randomly inside these limits.

It will be shown in the following, that if the two initial particles were not completely equal, the set of possible values for m is never empty. Since the mean value of the set (2.141), given by

$$\frac{1}{2}\left(-\frac{w_1\sqrt{d}}{|\vec{P}|} + \frac{w_1\sqrt{d}}{|\vec{P}|}\right) = 0, \tag{2.144}$$

is smaller than the mean value of the set (2.142), given by

$$\frac{1}{2}\left(1 - \frac{w_2\sqrt{d}}{|\vec{P}|} + 1 + \frac{w_2\sqrt{d}}{|\vec{P}|}\right) = 1, \tag{2.145}$$

it suffices to show that the upper boundary of (2.141) is bigger than the lower boundary of (2.142):

$$w_1 \frac{\sqrt{d}}{|\vec{P}|} > 1 - \frac{w_2\sqrt{d}}{|\vec{P}|} \tag{2.146}$$

$$\Leftrightarrow \qquad w_1 + w_2 > \frac{|\vec{P}|}{\sqrt{d}} \tag{2.147}$$

$$\Leftrightarrow \qquad \sqrt{\frac{\epsilon^2 - W^2}{\vec{P}^2}} > 1 \tag{2.148}$$

Since equation (2.123) must hold, as it corresponds to (2.121),

$$\sqrt{\frac{\epsilon^2 - \vec{P}^2}{W^2} - 1} > 0 \quad \Leftrightarrow \quad \frac{\epsilon^2 - \vec{P}^2 - W^2}{W^2} > 0 \quad \Leftrightarrow \quad \epsilon^2 - W^2 - \vec{P}^2 > 0, \tag{2.149}$$

it follows that

$$\sqrt{\frac{\epsilon^2 - W^2}{\vec{P}^2}} = \sqrt{\frac{\epsilon^2 - W^2 - \vec{P}^2}{\vec{P}^2} + 1} > 1. \tag{2.150}$$

Therefore a suitable m can always be found.

This concludes the algorithm to determine the momenta of the final particles after a splitting operation. Compared to the merging operation, the presented algorithm is very cheap in terms of its computational cost.

2.5 Modification of final locations for first-order particles

After merging or splitting is finished, pairs of final particles are located on a common location. If first-order particle shapes are used, it is possible to shift these particles from their common location, without disturbing the charge or current density distribution. Starting with (2.16) and (2.30) and using $N = 2$ it follows that if two first-order particles have $w = w_1 = w_2$, i.e. equal weights, and start at the same location $\vec{x} = \vec{x}_1 = \vec{x}_2$, the change in charge distribution $\Delta\rho_{jkl}$, by moving them δx in opposite x-directions, is given by

$$\begin{aligned}\Delta\rho_{jkl} = w\zeta_{2k}^{(1)}(y)\,\zeta_{3l}^{(1)}(z)\Big[&\zeta_{1j}^{(1)}(x+\delta x) - \zeta_{1j}^{(1)}(x) + \\ &\zeta_{1j}^{(1)}(x-\delta x) - \zeta_{1j}^{(1)}(x)\Big].\end{aligned} \tag{2.151}$$

Assuming that the particles start in one cell and that they do not leave the cell due to the movement by δx, this equation can be written as

$$\begin{aligned}\Delta\rho_{jkl} = w\zeta_{2k}^{(1)}(y)\,\zeta_{3l}^{(1)}(z)\cdot \\ \cdot\frac{-|x+\delta x-x_j|-|x-\delta x-x_j|+2|x-x_j|}{\Delta x}.\end{aligned} \tag{2.152}$$

Evaluating this expression for the two affected slices of grid-points gives

$$\begin{aligned}\Delta\rho_{0kl} &= w\zeta_{2k}^{(1)}(y)\,\zeta_{3l}^{(1)}(z)\cdot \\ &\quad\cdot\frac{-(x+\delta x-x_j)-(x-\delta x-x_j)+2(x-x_j)}{\Delta x} \\ &= 0, \quad \text{and}\end{aligned} \tag{2.153}$$

$$\begin{aligned}\Delta\rho_{1kl} &= w\zeta_{2k}^{(1)}(y)\,\zeta_{3l}^{(1)}(z)\cdot \\ &\quad\cdot\frac{(x+\delta x-x_j)+(x-\delta x-x_j)-2(x-x_j)}{\Delta x} \\ &= 0.\end{aligned} \tag{2.154}$$

In conclusion, it is possible to shift two first-order particles at the same location and with equal weight in opposite directions, along an arbitrary dimension axis, without introducing unphysical divergence into the system, as long as both particles do not cross cell boundaries. Unfortunately, this simple shift is not possible for second-order particles. This can be easily seen by performing the above analysis and replacing $\zeta^{(1)}$ with $\zeta^{(2)}$ (2.53). The non-linear nature of this particle shape makes it necessary that the corresponding shifts are different for the two particles.

Shifting the final particles in this way, will increase diversity in the new particles, as they quit sharing their spatial location. The specific dimension for this shift will be chosen at random. All subsequent test cases employ this shift for the final particles after both, merging and splitting, operations.

2.6 Minimization of the number of different weight species

The amount of different weight species, present in each cell, is bound to vary when employing a particle merge/split module. More quasi-particles in each cell are generally desirable as they increase the statistical significance of the simulation result. A multitude of weight species counteracts this advantage as can be easily seen when imagining a very heavy particle in a cell containing many very light particles. The density variation will be governed by the heavy particle, while the light particles only add computational load. The exact analytical correlation of this was shown in chapter 1.3, especially equation (1.42) with equation (1.44).

In order to improve the statistical significance of particles a second adjustment step has proven to be beneficial. It consists of the above detailed merging and splitting algorithm with the restriction that the particle number in each cell can not change. By identifying the number of quasi-particles with a weight above and below the average weight, a maximum number of possible merges and splits can be found, that, when performed, leave the number of particles in a cell constant. Merging and splitting particles in these specific weight groups, in the correct ratio, will decrease the number of weight species, but will keep the total number of particles in a cell constant. This increases the statistical significance of each quasi-particle, while decreasing the computational demand of the simulation. Unfortunately, these savings in computational demand need to be counterbalanced by the extra amount of computations the adjustment step takes. It still seems reasonable to perform the extra step in many cases, since particle processing should not be necessary for every time step. It only needs to be performed during an amount of time steps orders of magnitude fewer than that. Performing particle processing too frequently has also been observed to unnecessarily inflate the number of weight species in the system.

In order to adjudicate the efficiency of this extra step, two different setups were considered. The first setup consists of a hot ($T = 1\,\mathrm{keV}$), one dimensional plasma slab with a length of $0.5\,\mu$m, which expands unhindered, except by electromagnetic forces, into a vacuum box of size $2.5\,\mu$m. In this setup all particles are initially generated with the same weight.

The second setup consists of a hot ($T = 1\,\mathrm{keV}$), one dimensional gaussian plasma distribution with $\sigma = 0.25\,\mu$m, which expands unhindered, except by electromagnetic forces, into a vacuum box of size $2.5\,\mu$m. In this case, the initial particle setup could not solely contain particles of the same weight, as the density distribution was not constant. A special setup was written that ensured that only particle weights of a power of two were employed.

All boundaries are periodic. The grid size is 128 points. The simulation only performs calculations for the electrons, the ions are stationary. The number of particles per cell is adjusted at each time step, in order to keep it at a high level of 200 ± 20 particles per cell.

In this way, the number of particles inside the initial plasma, as well as in the regions of vacuum that get populated by plasma, need to be continuously increased via splitting. Merging and splitting was performed with $Q = 2$. It was triggered every 50th time step, starting with the first.

Results of the first test case are shown in Fig. 2.12. The first test case shows the proper functionality of the algorithm. As the plasma expands into the vacuum, the algorithm increases the number of particles from 5200 to ≈ 25000, a factor of 5, which is the fraction of space that initially contained plasma. The plasma therefore occupies the whole box using 200 quasi-particles in each cell.

In order to do this, two main weight species are needed: $1/8$ and $1/4$. In this case, the additional adjustment does not improve the results. It employs the same amount of weight species with a very similar number of quasi-particles. The reason for this is that the standard algorithm is carefully avoiding the creation of too many species as well. By always merging the lightest and splitting the heaviest particles, which is guaranteed by the weight sorting step, detailed in

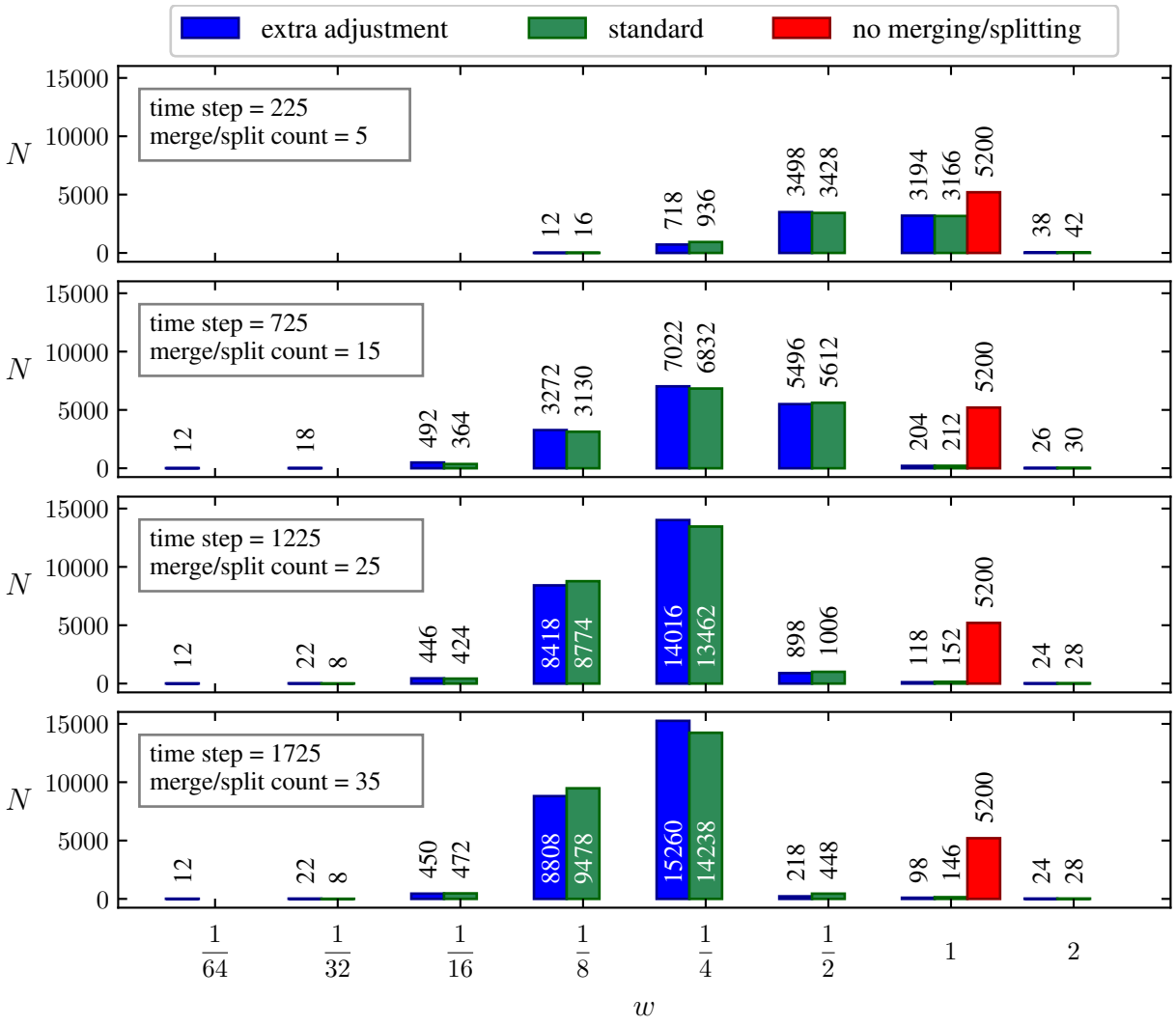

Fig. 2.12 – Results of a simple PIC simulation to illustrate the effects of an extra adjustment step. All graphs show the number of particles N for the different weight species w in the simulation. Three cases are shown: i) splitting and merging followed by an additional adjustment step (blue), ii) only splitting and merging (green), iii) no splitting or merging (red). The specific number of particles for each bar is plotted above or below the respective bar. Each graph contains the respective time step and the total number of performed particle processing operations. The total particle number for the blue case and per plotted timestep is given by 7460, 16542, 23954 and 24892 quasi-particles, respectively. For the green case it is 7588, 16180, 23854 and 24818, respectively. For the red case this number is constant for all time steps: 5200.

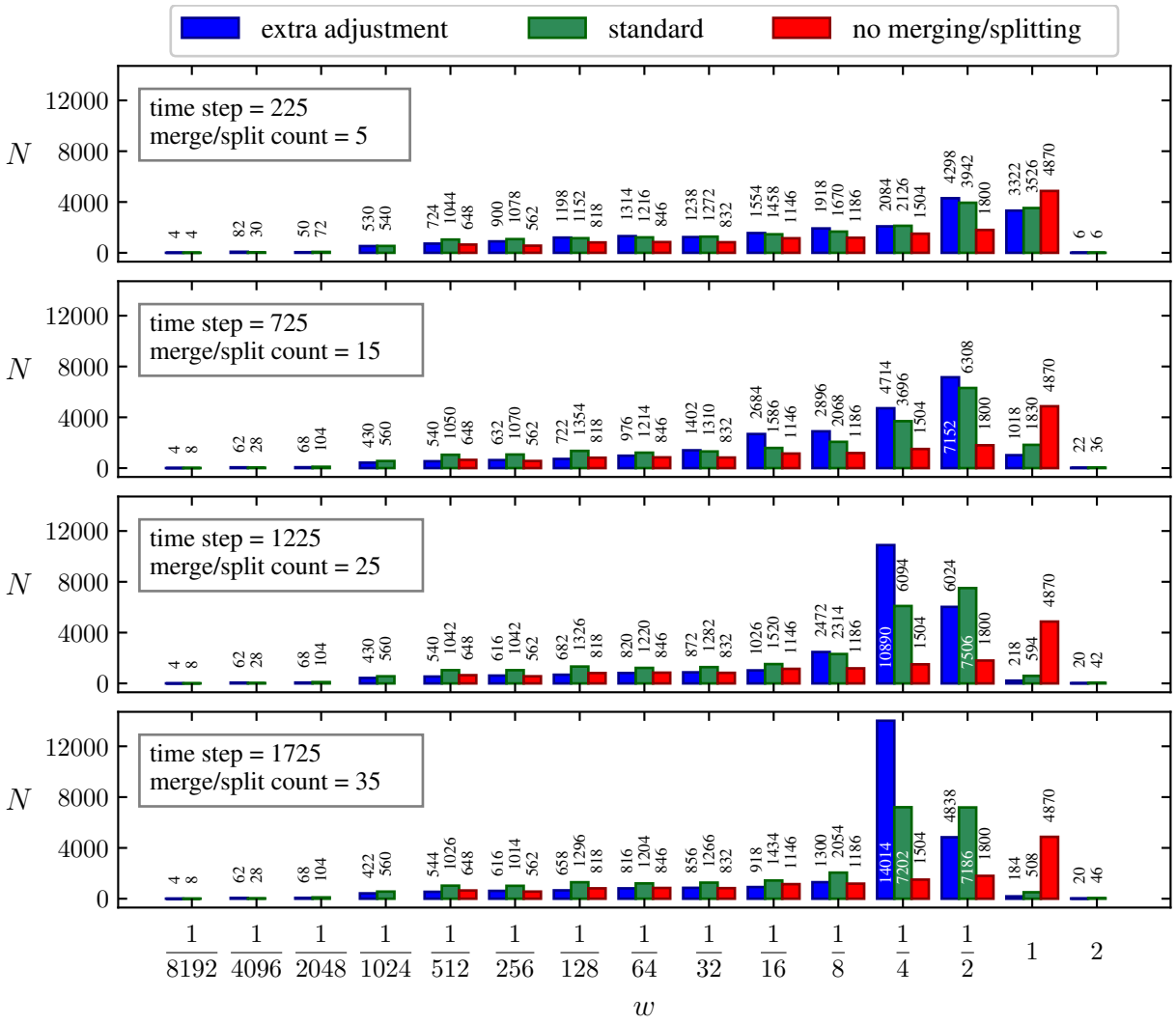

Fig. 2.13 – Results of a simple PIC simulation to illustrate the effects of an extra adjustment step in order to reduce the number of different weight species. All graphs show the number of particles N for the different weight species w in the simulation. Three cases are shown: i) splitting and merging followed by an additional adjustment step (blue), ii) only splitting and merging (green), iii) no splitting or merging (red). The specific number of particles for each bar is plotted above or below the respective bar. Each graph contains the respective time step and the total number of performed particle processing operations. The total particle number for the blue case and per plotted timestep is given by 19222, 23322, 24744 and 25320 quasi-particles, respectively. For the green case the respective particle numbers are given by 19136, 22222, 24682 and 24936. For the red case this number is constant for all time steps: 14212.

chapter 2.2.2, the amount of weight species is kept in check. This example shows the ability of the algorithm to minimize the number of weight species. In order to show the advantages of the additional post-processing step, a more complex case has to be constructed.

Results of the second test case are shown in Fig. 2.13. It has a much more complex initial weight distribution (shown in red), caused by the more complex plasma density and the specific particle setup algorithm. Initially, this setup already consists of 10 different weight species, given by powers of two from $w = 1/512$ to $w = 1$ (the red bars in Fig. 2.13). The outpouring plasma then requires a lot of splitting to keep the number of particles per cell at 200.

In this case the additional adjustment processing is beneficial. The last graph, the one for time step 1725, shows one very dominant weight species for the blue data, while the green data has a total of two important weight species. As given in the caption to Fig. 2.13, both, the blue and the green simulation, have approximately the same amount of quasi-particles, in accordance with the definition of the additional processing step.

When looking closely at the data in Fig. 2.13, it can be seen that the additional particles blue is exhibiting in the dominant $w = 1/4$ species, come from the lighter weight species with $w < 1/4$. This is reasonable, as the additional adjustment step enables the algorithm to more effectively get rid of these species when they are not needed anymore.

This shows, that the utility of this additional post-processing step is heavily dependent on the characteristics of the physical problem under consideration. An interesting observation is the amount of failed searches for initial momenta during the **Random selection of initial momenta values** algorithm detailed in chapter 2.3.2. In these simple test cases only $\approx 3.64\,\%$ of all the searches (over all groups, cells, and processing steps) failed to find an initial momentum. This number did not depend heavily on the specific setup and configuration.

The spatial location search, detailed in chapter 2.3.1, was successful in all (over all groups, cells, and processing steps) cases. This is probably due to the test cases being one-dimensional problems.

Another algorithmic detail is a possible restriction of splitting to only the heaviest particles in each cell. The heaviest particles are the particles with the highest weight. This restriction does not need an additional sorting step, as sorting by weight is already done. The splitting algorithm always starts with the heaviest particles. After any amount of particles in the bin with the heaviest particles have been split (and the target number of particles has not been reached), the algorithm will only proceed to the next (lighter) weight bin, if it was able to split all particles in that heaviest bin. If it was not possible to split all particles with the largest weight, (probably due to the sorting algorithm in momentum space failing to find suitable candidates), splitting is stopped for this cell and this processing iteration.

This may decrease the amount of created particles, but it especially helps in the case of $Q = 2$. In this case, the relevancy of each quasi-particle (in terms of its effect on the signal-to-noise ratio) decreases exponentially if the heavier particles remain in the cell. This increases the computational cost with little justification or payback. Thus, it is important to restrict the number of

weight species in a cell to two, which can be achieved by only splitting the heaviest particles. This modification is enabled for all simulations with $Q = 2$, except when noted otherwise.

2.7 Adapting the algorithm for photons

It does not require fundamental modifications to extend the algorithm to be able to handle photons. The main differences are:

1. Photons do not have different kinds, which reduces the amount of necessary sorting.
2. Photons do not have a charge, which makes moving them in a cell much simpler, as they do not interact with the electromagnetic fields. Therefore, the algorithm, detailed in chapter 2.3.1, is unnecessary for photons. Photons can be shifted at will, without introducing errors in the electromagnetic field values.
3. Photons have a different normalized energy, since they have zero mass.

The last point leads to the following modifications to the previously established equations. Equations (2.60) and (2.61) become

$$\epsilon_g^{\text{bc}} = w_g \left| \vec{p}_g^{\,\text{bc}} \right| \quad \text{and} \tag{2.155}$$

$$\epsilon^{\text{spare}} = \Sigma_i w_i \left| \vec{p}_{\,i} \right| - \Sigma_g \epsilon_g^{\text{bc}}, \tag{2.156}$$

respectively, equation (2.69) becomes

$$\begin{aligned} \epsilon^{\text{spend}} =& \Sigma_g^{\vec{\Pi}_g \text{defined}} \left(w'_{g1} \left| \vec{p}_g^{\,\text{bc}} + \vec{\Pi}_g \right| \right. \\ & \left. + w'_{g2} \left| \vec{p}_g^{\,\text{bc}} - \vec{\Pi}_g \right| \right), \end{aligned} \tag{2.157}$$

equation (2.81) becomes

$$\epsilon_1^{\text{oc}} = \left(\vec{p}_g^{\,\text{bc}} \cdot \hat{e}_l \right)^2 + \left(\vec{p}_g^{\,\text{bc}} \cdot \hat{e}_m \right)^2, \tag{2.158}$$

equations (2.85) and (2.86) become

$$\epsilon_1^{\text{oc}} = \left[\left(\vec{p}_g^{\,\text{bc}} + \vec{\Pi}_g \right) \cdot \hat{e}_k \right]^2 + \left(\vec{p}_g^{\,\text{bc}} \cdot \hat{e}_m \right)^2 \quad \text{and} \tag{2.159}$$

$$\epsilon_2^{\text{oc}} = \left[\left(\vec{p}_g^{\,\text{bc}} - \vec{\Pi}_g \right) \cdot \hat{e}_k \right]^2 + \left(\vec{p}_g^{\,\text{bc}} \cdot \hat{e}_m \right)^2, \tag{2.160}$$

respectively, equations (2.89) and (2.90) become

$$\epsilon_1^{\text{oc}} = \left[\left(\vec{p}_g^{\,\text{bc}} + \vec{\Pi}_g \right) \cdot \hat{e}_k \right]^2 + \left[\left(\vec{p}_g^{\,\text{bc}} + \vec{\Pi}_g \right) \cdot \hat{e}_l \right]^2 \quad \text{and} \tag{2.161}$$

$$\epsilon_2^{\text{oc}} = \left[\left(\vec{p}_g^{\,\text{bc}} - \vec{\Pi}_g \right) \cdot \hat{e}_k \right]^2 + \left[\left(\vec{p}_g^{\,\text{bc}} - \vec{\Pi}_g \right) \cdot \hat{e}_l \right]^2, \tag{2.162}$$

respectively, equation (2.94) becomes

$$\epsilon_g = \frac{2}{w_g}\left(\kappa\left(\epsilon^{\text{spare}} - \epsilon^{\text{spend}}\right) + w'_{g1}\left|\vec{p}_g^{\,\text{bc}} + \vec{\Pi}_g\right| + w'_{g2}\left|\vec{p}_g^{\,\text{bc}} - \vec{\Pi}_g\right|\right), \tag{2.163}$$

equation (2.95) becomes

$$\epsilon_g = \left|\vec{p}_g^{\,\text{bc}} + \vec{\Pi}_g + \xi_g\vec{\sigma}_{\text{mod}}\right| + \left|\vec{p}_g^{\,\text{bc}} - \vec{\Pi}_g - \xi_g\vec{\sigma}_{\text{mod}}\right|, \tag{2.164}$$

equation (2.97) becomes

$$\begin{aligned} S = &\left(\epsilon_g^2(\vec{\Pi}_g\cdot\vec{\sigma}_{\text{mod}}) - 4(\vec{\Pi}_g\cdot\vec{p}_g^{\,\text{bc}})(\vec{p}_g^{\,\text{bc}}\cdot\vec{\sigma}_{\text{mod}})\right)^2 \\ &- D\left[\frac{1}{4}\epsilon_g^4 - \epsilon_g^2\left(\left|\vec{\Pi}_g\right|^2 + \left|\vec{p}_g^{\,\text{bc}}\right|^2\right)\right. \\ &\left. + 4(\vec{\Pi}_g\cdot\vec{p}_g^{\,\text{bc}})^2\right], \end{aligned} \tag{2.165}$$

and equation (2.99) becomes

$$\xi_g = \pm\frac{\epsilon_g}{2}\sqrt{\frac{4\left(\left|\vec{p}_g^{\,\text{bc}}\right|^2\right) - \epsilon_g^2}{D}}. \tag{2.166}$$

In the splitting part, the following modifications are necessary. Equation (2.103) becomes

$$E = w|\vec{p}^{\,2}| = \frac{w}{2}|\vec{p}| + \frac{w}{2}|\vec{p}|, \tag{2.167}$$

while equation (2.106) becomes

$$\epsilon = \sum_{i=0}^{N} w_i|p_i|. \tag{2.168}$$

The variable d in (2.114), (2.115), (2.128) and (2.129) becomes

$$d = \frac{\epsilon^2}{W^2}. \tag{2.169}$$

Equation (2.118) becomes

$$4d - \widetilde{a}^2 = 4\frac{\epsilon^2}{W^2} - \left(\frac{2|\vec{P}|}{W}\right)^2 = 4\frac{\epsilon^2 - \vec{P}^2}{W^2}, \tag{2.170}$$

while equation (2.122) becomes

$$E_{\text{avail}} = \sqrt{\frac{\epsilon^2 - \vec{P}^2}{W^2}}, \tag{2.171}$$

and equation (2.123) becomes

$$\sqrt{\frac{\epsilon^2 - \vec{P}^2}{W^2}} > 0. \tag{2.172}$$

Equation (2.148) becomes

$$\frac{\epsilon}{|\vec{P}|} > 1, \tag{2.173}$$

which can be shown to hold true analogously to the equation for the particles. Since equation (2.149) becomes

$$\sqrt{\frac{\epsilon^2 - \vec{P}^2}{W^2}} > 0 \quad \Leftrightarrow \quad \epsilon^2 - \vec{P}^2 > 0, \tag{2.174}$$

which is now using (2.172), and therefore, analogously to the original version in (2.150), it follows that

$$\frac{\epsilon}{|\vec{P}|} = \sqrt{\frac{\epsilon^2}{\vec{P}^2}} = \sqrt{\frac{\epsilon^2 - \vec{P}^2}{\vec{P}^2} + 1} > 1, \tag{2.175}$$

which means that, in the case of photons as well, a suitable m will always be found.

3 Merging/Splitting Results

The complete algorithm, detailed in chapter 2, has been implemented for the PIC code PSC in 1D, 2D and 3D. The implementation comprises about 6000 lines of C code. Special care was taken to produce a maintainable, readable, efficient and well-engineered code.

Extensive test cases have been written as well. A total of 90 different test configurations have been designed to ensure that the algorithm fulfills its stated targets. The tested goals are momentum conservation, energy conservation and freedom of any spurious divergence. These targets are tested for one-, two- and three-dimensional simulations and for a variety of other parameters, including a fixed or variable Q, employment of the additional adjustment step, electromagnetic interaction, different particle sorting methods and check-pointing. The code performs perfectly for all test cases. The stated goals are achieved up to machine precision for all configurations. The test suite has been run after all major changes to the code in order to ensure a predictable performance, and has been integrated into the standard test suite of the PSC.

In this chapter, the performance of the algorithm will be evaluated. For this purpose multiple test cases have been established and analyzed. First, the well known two-stream instability is investigated for a multitude of different parameters. Second, the presented algorithm will be compared to a published algorithm, using the same metrics as in the publication.

The test cases will exclusively employ first-order particles. Therefore, in all cases, it suffices to only show the momentum space and its development, as the algorithm will not affect the configuration space in any way. The splitting/merging process will modify the quasi-particle positions, but it will leave the charge density undisturbed.

3.1 Two-stream instability

In order to evaluate the quality of the conservation properties of the scheme, the well-known two-stream problem in one spatial and one momentum dimension (i.e. in a two-dimensional phase space) is investigated. Two counter-propagating plasma streams, each with a certain temperature, move through each other, while exhibiting heavy particle merging or splitting. The quality of conservation is measured by comparing phase space plots of these two streams for different cases. A two-dimensional phase space was chosen, as it provides the possibility of giving the whole momentum space picture with just one figure.

It is important to note that in the case of splitting, the particles do start in a two-dimensional phase space, but due to the splitting algorithm, which tries to increase the variation in momentum space, the particles gain momenta in the off-directions (c.f. to Fig. 2.11, the resulting momenta $\vec{p}'_1$ and $\vec{p}'_2$ are not in the same plane as the total initial momentum $\vec{P}$). Still, since the particles do no start with any momentum in the off-directions, the algorithm will not be able to improve the situation in the monitored momentum direction meaningfully, as there is no additional "available energy" (2.122). Interestingly, initializing the simulations with a three-dimensional momentum space distribution, with a temperature in all three momentum directions and without requiring strong

fidelity in those extra dimensions, improves the results for both merging (through additional ϵ^{spare} (2.61)) and splitting (through additional "available energy" (2.122)). Since these results are not really meaningful for actual simulations, they are not shown here.

The generation of off-direction momenta during splitting could be remedied by switching the additional variation generating part of the algorithm off. It could also prove worthwhile to specialize the presented splitting algorithm to fewer momentum dimensions. Since the goal of this chapter is to test the given algorithm, this is not done for the test cases in this section.

The merging algorithm does not exhibit this momentum space expanding behavior. The resulting momenta of the merging algorithm are initial momenta or barycenter momenta, which are not in the off-directions by definition, that are modified to distribute the remaining energy. This distribution is done by going in the direction of the standard deviation of the initial momentum distribution. This standard deviation, also by definition, lies in the initial one-dimensional momentum space as well. This way, if the initial particles start in a two-dimensional phase space the merged particles will inhabit the same two-dimensional phase-space. This conservation has also been verified for all merging results.

The complete physical and numerical parameters can be found in Tab. 3.1. The plasma density and the number of quasi-particles is spatially constant. All boundaries of the simulation box are periodic for particles and fields. The simulations in this section are not meant to reproduce actual physical processes. Due to the very limited amount of only 10 timesteps, the evolution of the plasma is not meaningfully explored. The point of these simulations is to benchmark the fidelity in momentum space the merging/splitting algorithm is able to guarantee for different sets of parameters independent of any physical evolution. In order to investigate this, the particles are merged or split every timestep, which will not commonly happen in real scientific simulations. It is verified that the 10 simulated timesteps suffice to reach a converged particle number in all considered test cases. The results will therefore present the amount of momentum space preservation the algorithm is able to guarantee in the most extreme case. Additionally, the parameter set that guarantees a very competitive amount of preservation is identified as well.

The additional step, introduced in chapter 2.6, is not employed in order to make the results easier to analyze. All simulations restrict the weight space to powers of two and therefore $Q = 2$ (2.5) for every merging or splitting operation. This also requires a weight sorting, introduced in chapter 2.2.2, that exclusively sorts into bins of powers of two.

Figures in this section always show two different cases. The first case is a phase space diagram after merging or splitting up or down from a specific particle number. This is shown in red. The second case is a fresh setup, using a random number generator to fill the temperature defined momentum space, employing the resulting number of particles from the first case for the specific respective timestep (rounded up to a multiple of the number of plasma streams times the number of grid-points, which is equal to 2000). Therefore the distribution for this second case is strictly better than the distribution that is attainable by the merge/split procedure, and can therefore serve as a fidelity optimum. It is shown in blue.

Parameter & notation	Value
Plasma density, n_0	$1 \times 10^{16}\,\mathrm{cm}^{-3}$
Plasma initial x-direction temperature	$0.1\,\mathrm{eV}$
Plasma normed momentum separation	± 0.002
Initial number of quasi-particles for merging	100000
Initial number of quasi-particles per cell per stream for merging, N_{merge}	50
Initial number of quasi-particles for splitting	20000
Initial number of quasi-particles per cell per stream for splitting, N_{split}	10
Courant-Friedrichs-Lewy number [4]	0.75
Number of timesteps	10
Timestep size	$\mathit{6.25} \times 10^{-18}\,\mathrm{s}$
Number of momentum bins, N_{bins}	256
Evaluated normed momentum range, $[r_{\text{s}}, r_{\text{e}}]$	$[-0.004, 0.004]$
Normed momentum bin size, $\Delta p = \frac{0.008}{N_{\text{bins}}}$	3.125×10^{-5}
Box size in x-direction	$2.5\,\mu\mathrm{m}$
Number of spatial bins, N_{grid}	1000
Spatial grid size, Δx	$2.5 \times 10^{-3}\,\mu\mathrm{m}$

Table 3.1 – Parameters of the two-stream simulations. Numbers typed in italic are approximate values.

The data for these two cases is the result of performing each setup $N_{\text{samples}} = 1024$ times and calculating the mean value and the standard deviation of the density in phase space for the last timestep, timestep number 10. Particle numbers are always given for the whole simulation box. If a particle number is followed by a $\pm$-value, the number is the average value over all samples and the number after the $\pm$ symbol is its standard deviation. The dashed lines and the colored regions in each figure give the standard deviation of the respective mean quantities. The different samples use different initializations of the random number generator, leading to different particle setups as well as different results in the merging/splitting algorithm. The standard deviation is calculated separately for values above and below the mean, as the distribution is non-symmetric for values close to zero.

Figures in this chapter depict the mean of the binned momentum density ρ_p, which is a discretization of the quasi-particle distribution. The quasi-particles have a 0^{th}-order form factor in momentum space (a δ-distribution). The binned momentum density $\forall\, p_i \in \{r_{\text{s}} + (i - 1/2)\Delta p \,|\, i \in [1, N_{\text{bins}}]\}$ is therefore given by

$$\rho_p\left(p_i\right) = \frac{1}{N_{\text{t}} N_{\text{grid}} \Delta p} \sum_{\{j \,|\, p_i - \frac{1}{2}\Delta p \le p_j < p_i + \frac{1}{2}\Delta p\}} w_j p_j, \tag{3.1}$$

where N_{t} is either N_{merge} or N_{split}, p_j are the quasi-particle momenta in the x-direction and w_j are their weights. The factor N_{t} is necessary, as, for a constant n_0, the PSC will need to scale

the particle weights with the number of quasi-particles per cell. The mean $\mu(\rho_p)$ and standard deviation $\sigma(\rho_p)$ of this quantity is calculated discreetly for each bin p_i over all samples N_{samples}.

Different values for $N_{\text{max}}^{\text{sort}}$ (equation (2.4) defined in chapter 2.2.3) and l_{min} (2.6) for the underlying momentum sort parameters were tested, representing different requirements for the faithfulness in momentum space. Parameter $N_{\text{max}}^{\text{sort}}$ fixes the amount of quasi-particles per momentum space volume, while l_{min} divides the momentum space up into buckets of a certain size, allowing particle processing only for particles inside the same bucket. It is important to note that these buckets are not symmetrically aligned over the momentum space. Due to the modifications to the minimum and maximum values for each momentum dimension, explained in list item one in chapter 2.2.3, the specific bisection point inside the momentum space is chosen randomly. For $l_{\text{min}} = 1$ there are two buckets in the box, except for two singular cases, where the bisection point is exactly at the original minimum or maximum value, in which case there is only one bucket. The amount of buckets increases exponentially with the value of l_{min}. For $l_{\text{min}} = 2$ there is a maximum of four buckets in the box, randomly placed over it, and so on. The number and size of the buckets defines a certain requested resolution in momentum space.

The following two sections will show that $N_{\text{max}}^{\text{sort}}$ is a more important parameter for splitting, while l_{min} is more important for merging.

3.1.1 Merging performance

First, the merging behavior of the algorithm is explored. Switching off splitting and forcing aggressive merging can be done by setting $N_{\text{min}}^{\text{cell}} = 0$ and $N_{\text{max}}^{\text{cell}} = 1$ in equation (2.1), which means that the algorithm tries to reduce the total number of particles to $N_{\text{grid}} = 1000$. This number is never reached and therefore ensures that, if it is able to find appropriate target groups, the algorithm never stops merging. This way, these test cases show the most extreme results, and also provide information about the minimum particle number that is needed to represent the distribution. The different tested values for $N_{\text{max}}^{\text{sort}}$ and l_{min} are given by $N_{\text{max}}^{\text{sort}} = [12, 26, 8000]$ and $l_{\text{min}} = [0, 2, 4]$.

In the case of $Q = 2$, 1D merging, each merging event needs a minimum of four initial particles, three particles because of the analysis shown in (2.39), plus one to get $Q = 2$. Therefore, setting $N_{\text{max}}^{\text{sort}} = 12$ results in approximately two merge ranges in each bin (one range needs four particles that are merged to two final particles at a common location). This way, the range capability and the redistribution of energy between ranges is tested. The different values for $N_{\text{max}}^{\text{sort}}$ also lead to different rates of merging per timestep, as can be seen in the number of particles in the box.

The results for $N_{\text{max}}^{\text{sort}} = 8000$ and $l_{\text{min}} = 0$ are shown in Fig. 3.1. It is obvious that these parameters do not work well for this particular problem. The particle momenta after merging are still representing a two-stream situation, but the two streams are much less distinct. In this figure, as well as in all the following figures, the sampling standard deviation (the vertical variation) of the merged distribution is very similar to the sampling standard deviation of the fresh setup. It seems

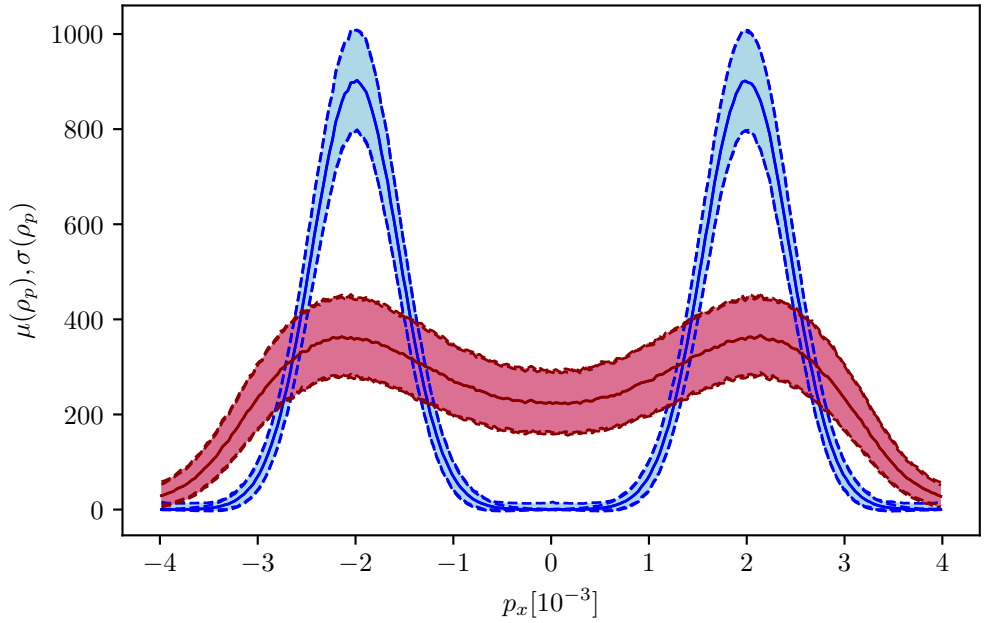

Fig. 3.1 – The red line shows the mean momentum space density after 10 timesteps of heavy merging, with the parameters $N_{\text{max}}^{\text{sort}} = 8000$ and $l_{\text{min}} = 0$. The total number of particles was reduced from 100000 particles at timestep zero to an average of 8102±18 particles at timestep 10. The blue line shows a fresh setup with 10000 particles.

that the merging algorithm is very deterministic, even though it contains a fair amount of random numbers. In the following sections, the impact of modifying $N_{\text{max}}^{\text{sort}}$ and l_{min} will be investigated.

Scan over different values for $N_{\text{max}}^{\text{sort}}$

In this section the impact of decreasing the value of $N_{\text{max}}^{\text{sort}}$ from 8000 to 12 will be investigated. The results for $N_{\text{max}}^{\text{sort}} = 26$ are shown in Fig. 3.2. They have improved a bit from the values shown in Fig. 3.1. Due to the stricter requirements, the algorithm was not able to reduce the number of particles as much as before. The average final particle number is given by 11657±43, while it previously was only 8102±18.

The remnants of the two-stream setup are a bit more pronounced. But overall, the momentum phase structure is still meaningfully different and it is not expected that the physical processes will be adequately reproduced.

Further reducing $N_{\text{max}}^{\text{sort}}$ to 12 yields the values shown in Fig. 3.3. The final number of particles has increased again. Unfortunately, compared to the previous case shown in Fig. 3.2, the momentum space structure has not improved a lot. It still is too different to expect a reasonable fidelity to the physical characteristics of the un-merged setup.

In conclusion, parameter $N_{\text{max}}^{\text{sort}}$ is not very well positioned to guarantee faithfulness in momentum

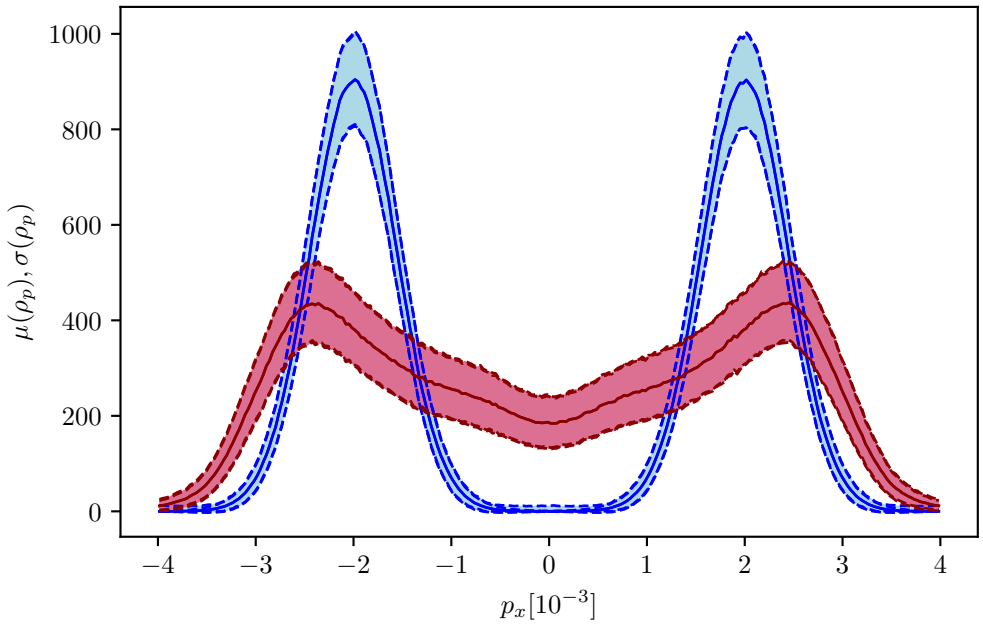

Fig. 3.2 – The red line shows the mean momentum space density after 10 timesteps of heavy merging, with the parameters $N_{\text{max}}^{\text{sort}} = 26$ and $l_{\text{min}} = 0$. The total number of particles was reduced from 100000 particles at timestep zero to an average of 11657±43 particles at timestep 10. The blue line shows a fresh setup with 12000 particles.

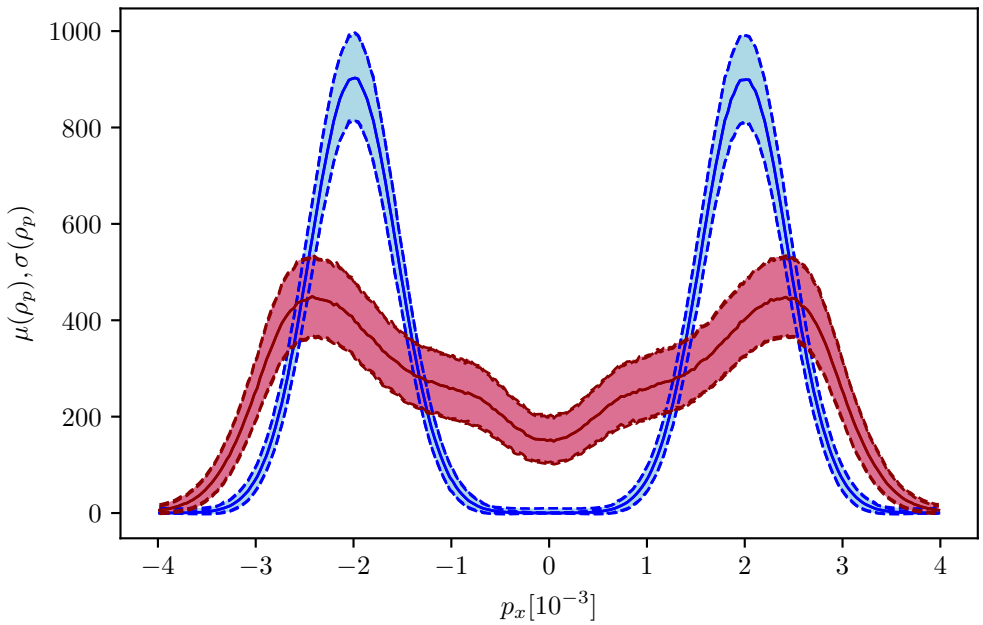

Fig. 3.3 – The red line shows the mean momentum space density after 10 timesteps of heavy merging, with the parameters $N_{\text{max}}^{\text{sort}} = 12$ and $l_{\text{min}} = 0$. The total number of particles was reduced from 100000 particles at timestep zero to an average of 13550±79 particles at timestep 10. The blue line shows a fresh setup with 14000 particles.

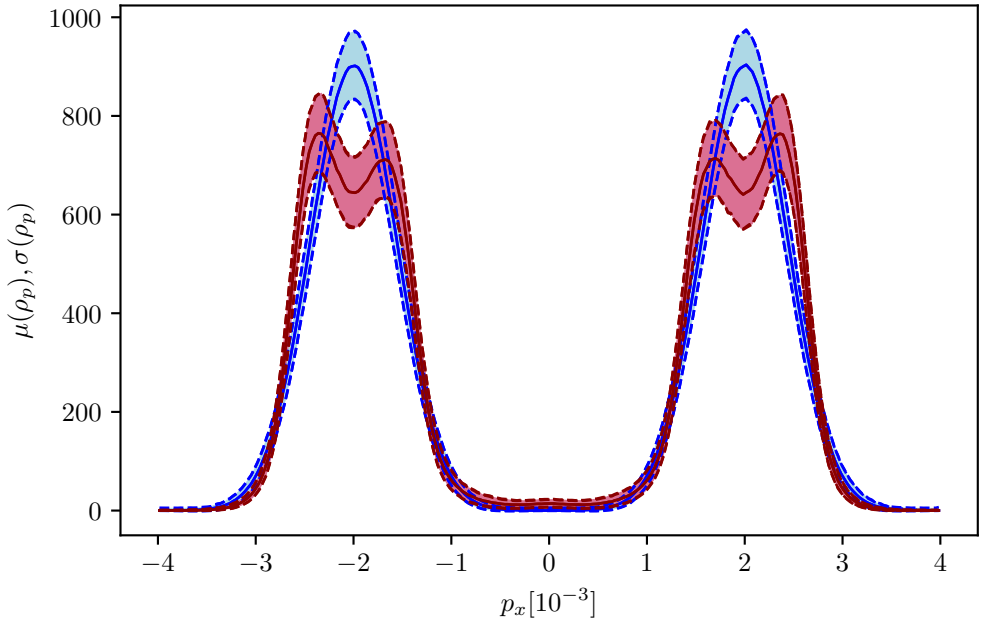

Fig. 3.4 – The red line shows the mean momentum space density after 10 timesteps of heavy merging, with the parameters $N_{\text{max}}^{\text{sort}} = 8000$ and $l_{\text{min}} = 2$. The total number of particles was reduced from 100000 particles at timestep zero to an average of 23261±82 particles at timestep 10. The blue line shows a fresh setup with 24000 particles.

space for the merging operation. Even for the very restrictive $N_{\text{max}}^{\text{sort}} = 12$, the results are not very convincing. The fact that the initial number of quasi-particles is not constant over the momentum space, coupled with this parameter's increased rate of merging in places with more particles, deteriorates the momentum space distribution meaningfully. The second parameter l_{min} will prove to be much better suited for this benchmark.

Scan over different values for l_{min}

The impact of different values for l_{min} on the merging operation is investigated in this section. It will be seen that this parameter is substantially more important. Fig. 3.4 uses $l_{\text{min}} = 2$. Compared to all previous tests shown in Fig. 3.2 and Fig. 3.3, the momentum space fidelity is already much better. Unfortunately, the two streams are now made up of four peaks instead of only two. This case uses an average of four buckets in momentum space, which is too coarse for the investigated distribution. The two distinct peaks therefore get washed out into four peaks. Still, guaranteeing a certain minimum momentum resolution generates a much better result than what was possible by modifying $N_{\text{max}}^{\text{sort}}$.

This assessment is confirmed when considering the results for $l_{\text{min}} = 4$, shown in Fig. 3.5. Again, these results are much better than the ones shown in Fig. 3.3. Increasing the minimum sort level to four increases the fidelity in momentum space to a very respective level. The difference to

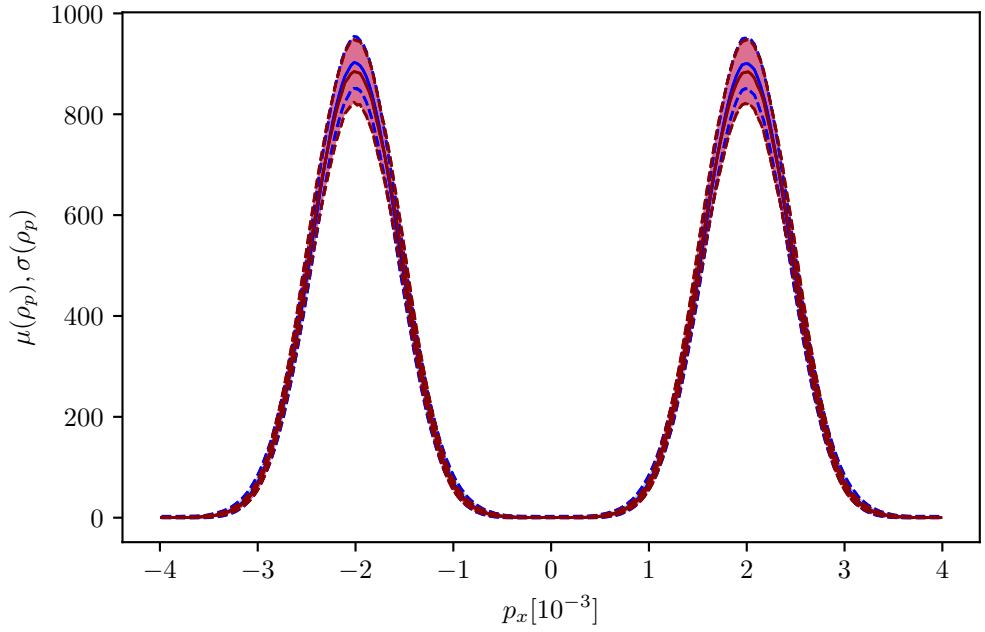

Fig. 3.5 – The red line shows the mean momentum space density after 10 timesteps of heavy merging, with the parameters $N_{\text{max}}^{\text{sort}} = 8000$ and $l_{\text{min}} = 4$. The total number of particles was reduced from 100000 particles at timestep zero to an average of 42572±125 particles at timestep 10. The blue line shows a fresh setup with 44000 particles.

the optimal setup is minimal. The final number of particles is considerably higher (42572±125) in comparison to the most restrictive $N_{\text{max}}^{\text{sort}} = 12$ case (13550±79), but in turn, the fidelity in momentum space is unmatched.

Merging with very restrictive parameters

Finally, Fig. 3.6 shows results for a run with $N_{\text{max}}^{\text{sort}} = 12$ and $l_{\text{min}} = 4$. The difference to the data shown in Fig. 3.5 is minimal, but it still constitutes a small improvement. The momentum space fidelity is great, even though the merging algorithm was not specialized for this problem. Even the standard deviation (the vertical colored region) of the merged case (shown by the dashed red lines) is approximately the same as the optimal one (shown by the dashed blue lines). Expectedly, the final amount of particles 44494±129 is the largest amount of all tests. Still, the number of quasi-particles has been reduced by more than half of the initial amount. It seems that at least 44 quasi-particles per cell are needed to adequately resolve the two-stream setup.

In conclusion, it is important to be aware of the approximate shape of the distribution function in momentum space when setting the merging parameters. The amount of distinct features present in the momentum space prescribes the value of the minimum sorting level l_{min}. For $l_{\text{min}} = 4$ the two-stream setup can be adequately resolved. Therefore, $l_{\text{min}} = 4$ is chosen for the benchmarks shown later in chapter 3.2. Parameter l_{min} is more important and restrictive than $N_{\text{max}}^{\text{sort}}$, and should

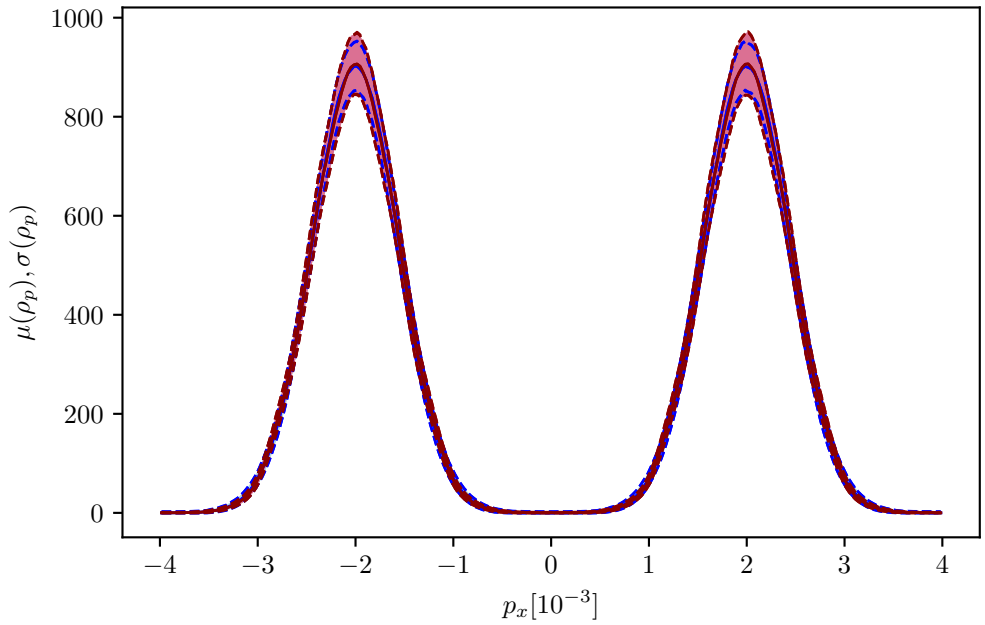

Fig. 3.6 – The red line shows the mean momentum space density after 10 timesteps of heavy merging, with the parameters $N_{\text{max}}^{\text{sort}} = 12$ and $l_{\text{min}} = 4$. The total number of particles was reduced from 100000 particles at timestep zero to an average of 44494±129 particles at timestep 10. The blue line shows a fresh setup with 46000 particles.

always be at least > 0, if a temperate plasma needs to be resolved. It is also of importance to tailor the parameters to the amount of merging needed for the problem, as more restrictive parameter values may not be able to provide for a sufficiently momentous decrease in quasi-particles.

3.1.2 Splitting performance

After the investigation into the merging behavior in the previous section, the splitting behavior of the algorithm, with respect to the two-stream problem, is explored. Switching off merging and forcing aggressive splitting can be done by setting $N_{\text{min}}^{\text{cell}} = 100$ and $N_{\text{max}}^{\text{cell}} = 10000$ in equation (2.1), which means that the algorithm tries to increase the total number of particles to 100000, 100 particles per cell. Each cell starts with 20 quasi-particles.

In comparison with the results shown in the previous section, the standard deviation in this section will be much smaller for many figures, at least for the optimal case. The reason for this is the increased number of quasi-particles in the last timestep, which leads to a decrease in the standard deviation, as shown in chapter 1. It is important to disable the restriction of only splitting the heaviest particles, detailed in the last paragraph of chapter 2.6, for the simulations shown in this chapter. This restriction does not work well for repeated calls to the algorithm. The benchmarks execute the splitting algorithm ten times in a row, which prohibits most quasi-particles from switching the cell. Splitting only the heaviest particles therefore leads to the situation,

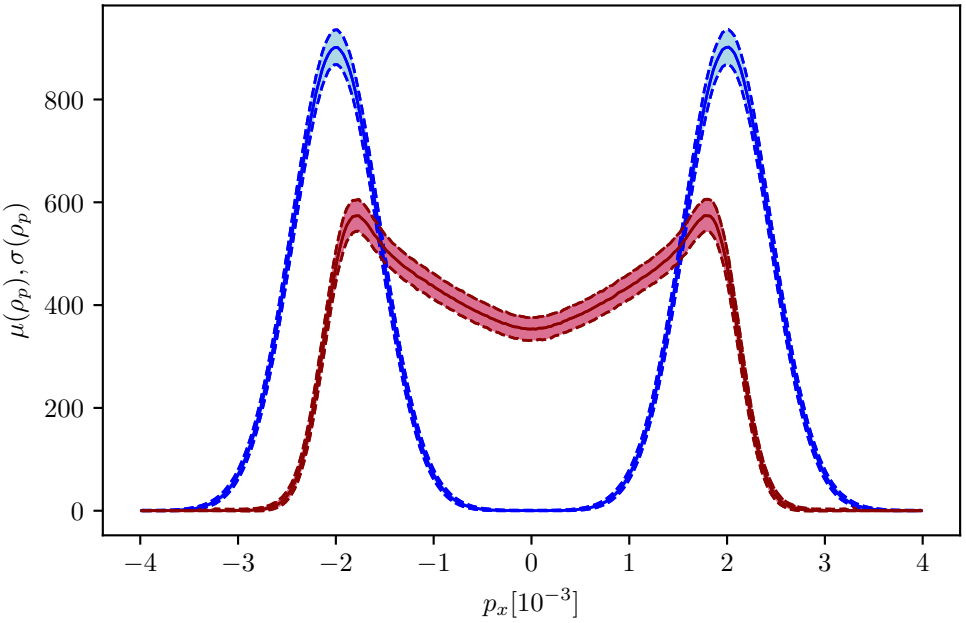

Fig. 3.7 – The red line shows the mean momentum space density after 10 timesteps of heavy splitting, with the parameters $N_{\text{max}}^{\text{sort}} = 4000$ and $l_{\text{min}} = 0$. The total number of particles was increased from 20000 particles at timestep zero to an average of 99690±18 particles at timestep 10. The blue line shows a fresh setup with 100000 particles.

where a very small number of residual heavy quasi-particles in a cell completely stop any further splitting, leading to untypical results.

Again, different values for $N_{\text{max}}^{\text{sort}} = [6, 13, 4000]$ (equation (2.4) defined in chapter 2.2.3) and $l_{\text{min}} = [0, 2, 4]$ (2.6) for the underlying momentum sort parameters were tested.

In contrast to the previous chapter, we use a different range of values for $N_{\text{max}}^{\text{sort}}$. Each test in the case of splitting will be investigated using an $N_{\text{max}}^{\text{sort}}$ value that is half of the value that was used for the merging tests. The reason for this difference can be found in the amount of particles per merging/splitting event. In the case of $Q = 2$, 1D merging, each merging event needs a minimum of four initial particles, three particles because of the analysis shown in (2.39), plus one to get $Q = 2$. In contrast to that value, splitting only needs two initial particles per $Q = 2$ event. This was shown in (2.123) and the subsequent reasonings. Therefore, it holds that if $N_{\text{max,merge}}^{\text{sort}} = 2 \cdot N_{\text{max,split}}^{\text{sort}}$ the amount of possible merge/split events is the same for both types of operations. Thus, choosing the parameters in this way makes the two cases comparable, when investigating the amount of distortions introduced into the momentum space distribution. This relation must be kept in mind when comparing the different figures in chapter 3.1.1 and chapter 3.1.2.

The results for $N_{\text{max}}^{\text{sort}} = 4000$ and $l_{\text{min}} = 0$ are shown in Fig. 3.7. As it was found for the merging case, these weak parameters do not lead to an appreciable fidelity in momentum space. Nevertheless, these results are better than the results for the merge case, shown in Fig. 3.1. The

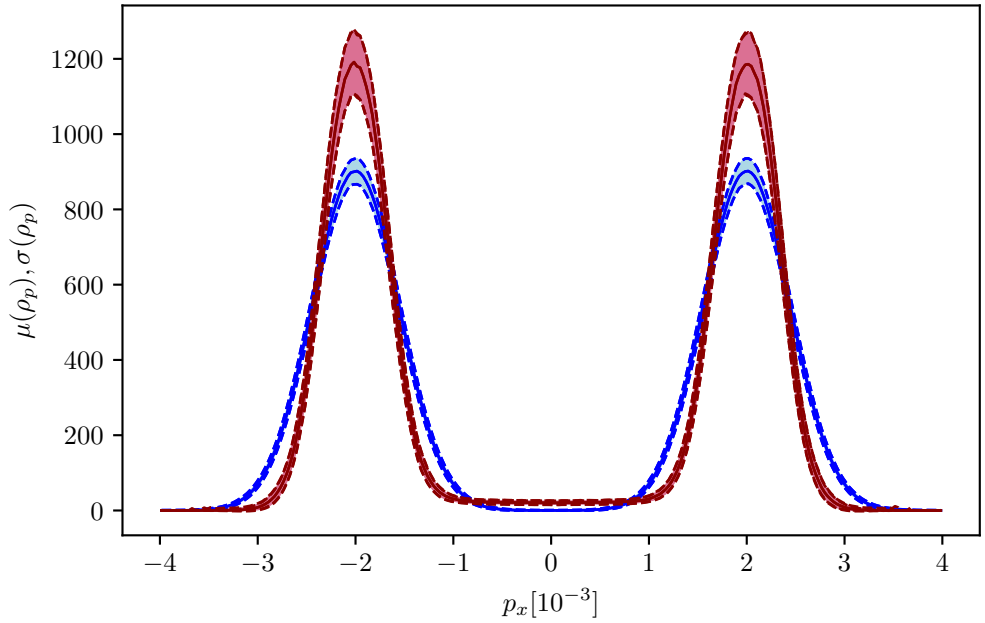

Fig. 3.8 – The red line shows the mean momentum space density after 10 timesteps of heavy splitting, with the parameters $N_{\text{max}}^{\text{sort}} = 13$ and $l_{\text{min}} = 0$. The total number of particles was increased from 20000 particles at timestep zero to an average of 99606±20 particles at timestep 10. The blue line shows a fresh setup with 100000 particles.

two counter-propagating streams are more visible and have not been deteriorated as much by the algorithm. Again, the sampling standard deviation (the vertical variation) of the distribution after splitting is very similar to the sampling standard deviation of the fresh setup. It seems that the splitting algorithm is very deterministic as well. In the following sections the impact of varying $N_{\text{max}}^{\text{sort}}$ and l_{min} will be investigated.

Scan over different values for $N_{\text{max}}^{\text{sort}}$

In this section, the impact of decreasing the value of $N_{\text{max}}^{\text{sort}}$ from 4000 to 6 will be investigated. The result for $N_{\text{max}}^{\text{sort}} = 13$ is shown in Fig. 3.8. It has markedly improved from the values shown previously in Fig. 3.7, without affecting the amount of final particles. The average final particle number is given by 99606±20, while it was 99690±18 before. The two streams are very clearly visible, the main difference to the optimal distribution, shown in blue, is given by a different shape of the two streams. The two streams do not follow the correct Gaussian distribution, as they exhibit a smaller temperature, signified by the smaller width of the two peaks.

Further reducing $N_{\text{max}}^{\text{sort}}$ to 6 yields the values shown in Fig. 3.9. This step fixes the previous problem of smaller temperatures in the two streams. The two Gaussian distributions now very closely resemble the optimal distributions, given in blue. The final number of particles is still exactly the requested amount of approximately 100000 quasi-particles. The main difference can

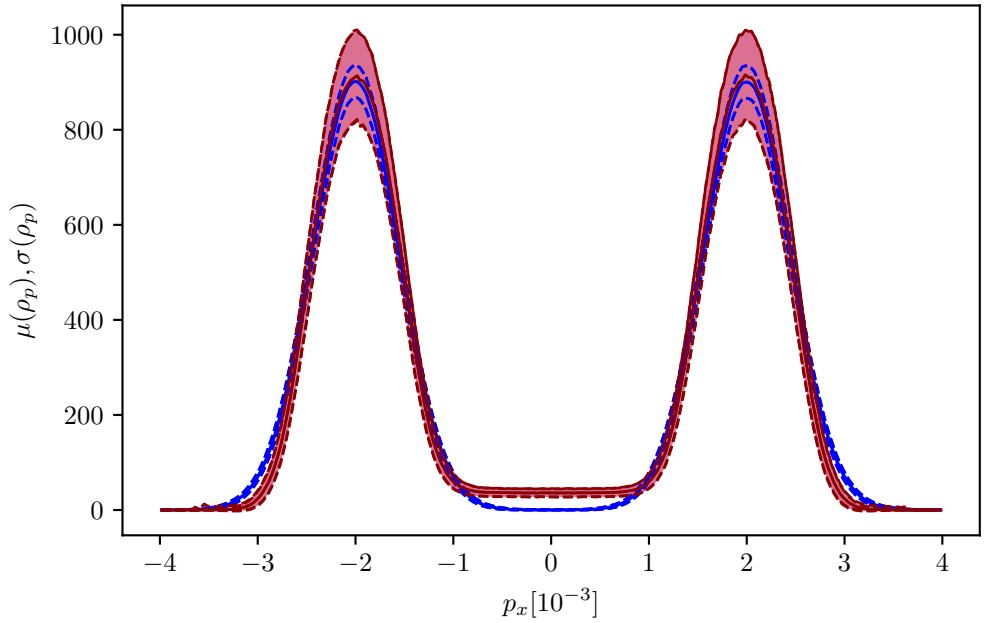

Fig. 3.9 – The red line shows the mean momentum space density after 10 timesteps of heavy splitting, with the parameters $N_{\text{max}}^{\text{sort}} = 6$ and $l_{\text{min}} = 0$. The total number of particles was increased from 20000 particles at timestep zero to an average of 99621±20 particles at timestep 10. The blue line shows a fresh setup with 100000 particles.

be found in the standard deviation (the vertical colored region), which is about double the size for the split results. This means, that particle splitting is introducing some added variance into the momentum distribution. Comparing these results with Fig. 3.3, it is immediately apparent that splitting benefits much more from stricter $N_{\text{max}}^{\text{sort}}$ values than merging does. The next section will show that the inverse is true for l_{min}.

Scan over different values for l_{min}

The impact of different values for l_{min} is investigated in this section. In contrast to the merging case, it will be shown that this parameters has less impact for splitting. Setting $N_{\text{max}}^{\text{sort}}$ to 6 already provided a very good result and it will be seen that parameter l_{min} can not challenge it. Fig. 3.10 uses $l_{\text{min}} = 2$. It is quite a bit worse than Fig. 3.8, which is the opposite behavior as was seen in the case of merging (Fig. 3.4 and Fig. 3.2). The problem of the decreased temperature in the two counter-propagating streams appears with an even higher magnitude than before. These parameters really push the momentum density into the two beams and therefore do not provide appreciable results.

The results for $l_{\text{min}} = 4$ are shown in Fig. 3.11. These results are slightly worse than the ones shown in Fig. 3.9. Still, increasing l_{min} does fix the problem of the decreased temperature like decreasing $N_{\text{max}}^{\text{sort}}$ does, but the peak value is still a bit too high, compared to the optimal setup. As

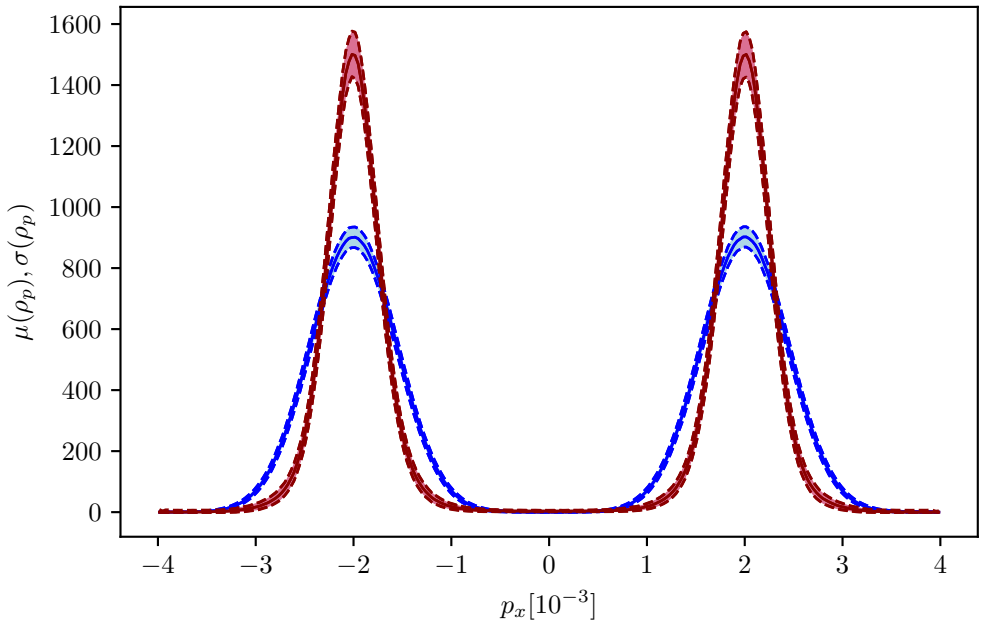

Fig. 3.10 – The red line shows the mean momentum space density after 10 timesteps of heavy splitting, with the parameters $N_{\text{max}}^{\text{sort}} = 4000$ and $l_{\text{min}} = 2$. The total number of particles was increased from 20000 particles at timestep zero to an average of 99612±21 particles at timestep 10. The blue line shows a fresh setup with 100000 particles.

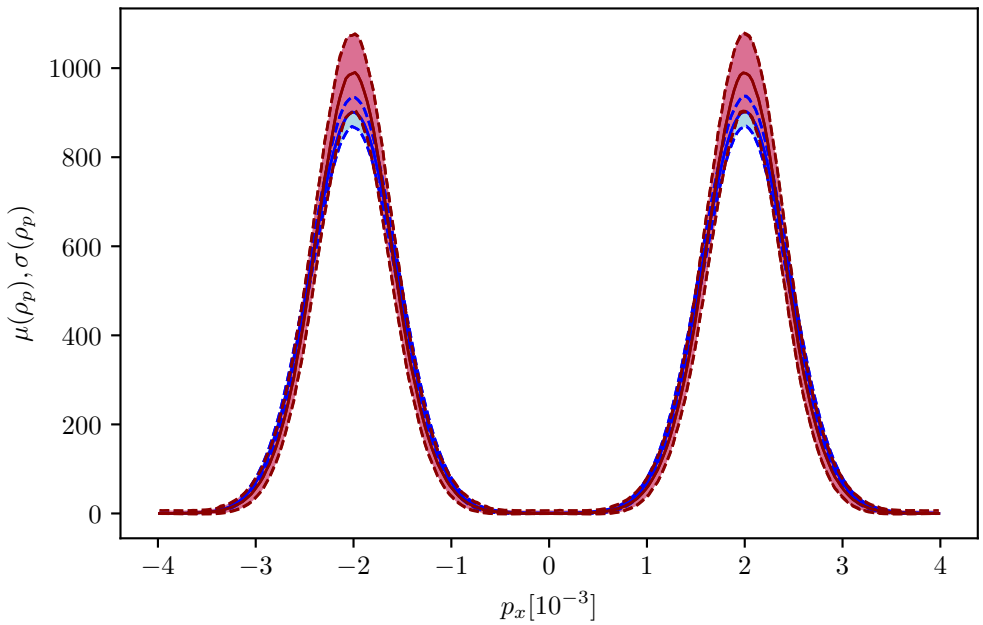

Fig. 3.11 – The red line shows the mean momentum space density after 10 timesteps of heavy splitting, with the parameters $N_{\text{max}}^{\text{sort}} = 4000$ and $l_{\text{min}} = 4$. The total number of particles was increased from 20000 particles at timestep zero to an average of 99638±22 particles at timestep 10. The blue line shows a fresh setup with 100000 particles.

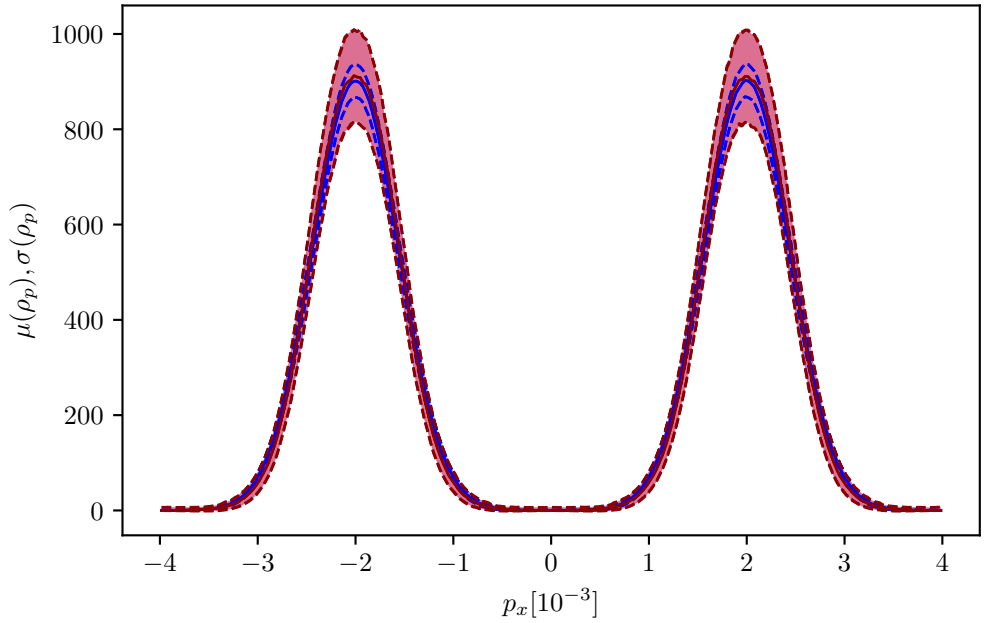

Fig. 3.12 – The red line shows the mean momentum space density after 10 timesteps of heavy splitting, with the parameters $N_{\text{max}}^{\text{sort}} = 6$ and $l_{\text{min}} = 4$. The total number of particles was increased from 20000 particles at timestep zero to an average of 99633±21 particles at timestep 10. The blue line shows a fresh setup with 100000 particles.

in the previous case, Fig. 3.9, the standard deviation is again too large, compared to the optimal case. The final number of particles is still the requested amount (99638±22), and about the same value as in the most restrictive $N_{\text{max}}^{\text{sort}} = 6$ case (99621±20). Overall, these results are good enough, the momentum fidelity should be sufficient for most cases.

Splitting with very restrictive parameters

Finally, Fig. 3.12 shows the results for a run with $N_{\text{max}}^{\text{sort}} = 6$ and $l_{\text{min}} = 4$. The momentum space fidelity is great, even though the splitting algorithm does not take any additional properties of the specific test case into account. The difference to Fig. 3.9 is minimal. The final amount of particles (99633±21) is again exactly as it was requested from the algorithm.

In summary, for splitting, increasing l_{min} seems to not be very important, even though it still improves the results. It also seems to be less important to know about the approximate shape of the distribution function in momentum space when setting the parameters in general. This is reasonable, as the splitting algorithm is much more localized, as it does not combine parameters of that many different particles. Contrary to the case of merging, parameter $N_{\text{max}}^{\text{sort}}$ is more powerful than l_{min}. It does not seem very important to tailor the parameters to the amount of splitting needed for the problem, as more restrictive parameter values do not hinder a sufficiently momentous increase in quasi-particles.

Parameter & notation	Value
Plasma density, $\bar{n}_0$	$3.14 \times 10^8\,\mathrm{cm}^{-3}$
Plasma thermal x-direction velocity, $\bar{v}_{\mathrm{th}}$	$1 \times 10^7\,\frac{\mathrm{m}}{\mathrm{s}}$
Plasma initial temperature, $T = m_{\mathrm{e}}\bar{v}_{\mathrm{th}}^2$	*0.57* KeV
Plasma drift velocity, $\bar{v}_{\mathrm{d}}$	$\pm 4\bar{v}_{\mathrm{th}}$
Plasma frequency, ω_{p}	$1 \times 10^9\,\frac{1}{\mathrm{s}}$
Initial number of quasi-particles for merging	2000000
Final number of quasi-particles after merging	160000
Initial number of quasi-particles per cell per stream for merging, N_{merge}	1000
Final number of quasi-particles per cell per stream after merging	80
Courant-Friedrichs-Lewy number [4]	0.98
Number of timesteps	30592
Timestep size	$\mathit{3.27} \times 10^{-12}\,\mathrm{s}$
Number of momentum bins, N_{bins}	200
Evaluated velocity range, $\pm 2 \cdot \bar{v}_{\mathrm{d}}$	$[-8\bar{v}_{\mathrm{th}}, 8\bar{v}_{\mathrm{th}}]$
Velocity bin size, $\Delta\bar{v} = \frac{2 \cdot 8\bar{v}_{\mathrm{th}}}{N_{\mathrm{bins}}}$	$0.08\bar{v}_{\mathrm{th}}$
Evaluated normed momentum range, $[r_{\mathrm{s}}, r_{\mathrm{e}}]$, $\pm 2 \cdot \gamma\bar{v}_{\mathrm{d}}/c$	$[\mathit{-0.267}, \mathit{0.267}]$
Normed momentum bin size, $\Delta p = \frac{2 \cdot \mathit{0.267}}{N_{\mathrm{bins}}}$	$\mathit{2.67} \times 10^{-3}$
Box size in x-direction, L	1 m
Number of spatial bins, N_{grid}	1000
Spatial grid size, Δx	1 mm
Maximum number of particles per bin after sorting (2.4), $N^{\mathrm{sort}}_{\mathrm{max,merge}}$	12
Maximum number of particles per bin after sorting (2.4), $N^{\mathrm{sort}}_{\mathrm{max,split}}$	6
Minimum sort binary tree splitting level (2.6), l_{min}	4

Table 3.2 – Parameters of the two-stream simulations replicating [29] chapter 4.4. Numbers typed in italic are approximate values. Parameter m_{e} is the electron mass.

3.2 Comparison with a different published algorithm

In this chapter, results of the newly established algorithm are compared to results published in [29]. To this end, tests as well as their respective analysis from [29] chapter 4.4 are reproduced and run using the presented algorithm and the PSC. It was possible to generate very comparable results with decisive advantages and improvements.

The parameters for these simulations can be found in Tab. 3.2. The previous chapter 3.1, especially section 3.1.1, was essential to finding the optimal values for the parameters of the merging algorithm, and led to choosing $N^{\mathrm{sort}}_{\mathrm{max,merge}} = 12$ and $l_{\mathrm{min}} = 4$. In contrast with the previous chapter, the additional step, introduced in chapter 2.6, is employed, with the hope that it increases the SNR value of the simulations. All simulations restrict the weight space to powers of two and therefore $Q = 2$ (2.5) for every merging or splitting operation. This also requires weight space

sorting, introduced in chapter 2.2.2, exclusively sorting into bins of powers of two.

The paper describes the setup in plasma units and uses velocity instead of momentum throughout. The reason for this is that the paper does not present a relativistic algorithm. The PSC setup was done using SI units. In order to be a faithful representation of the simulations that were presented in [29] chapter 4.4, the parameters in SI units need to fulfill the following equations,

$$\bar{v}_\mathrm{d} = \pm 4 \bar{v}_\mathrm{th}, \tag{3.2}$$

$$\lambda_\mathrm{D} = \sqrt{\frac{\epsilon_0 T}{n_0 e^2}} = \frac{L}{100}, \tag{3.3}$$

where λ_D is the Debye length, ϵ_0 is the permittivity of the vacuum and e is the elementary charge. All other values are defined in Tab. 3.2. Equation (3.3) can be written as

$$\omega_\mathrm{p} = \sqrt{\frac{n_0 e^2}{m_\mathrm{e} \epsilon_0}} = 100 \frac{\bar{v}_\mathrm{th}}{L}. \tag{3.4}$$

The SI values, given in Tab. 3.2, satisfy these conditions. This guarantees that the present setup and the one that was reported about in the paper by Teunissen and Ebert, are equal. In the following it will be seen that the results are comparable as well.

The first result is given in Fig. 3.13. Like the test cases shown in [29], merging is performed in five consecutive timesteps, going from a total number of about 2×10^6 quasi-particles (10^6 quasi-particles per stream) to about 16×10^4 (8×10^4 per stream) quasi-particles. The paper [29] identifies an important additional parameter for the quality of the merging process. This parameter is the point in time at which the particle number is reduced. It turns out that earlier merging will be more detrimental to the fidelity of the simulation. The non-linear evolution of the two-stream mixing process is perturbed and evolves very differently if the merging happens before meaningful changes in the phase space occur. In order to study this effect, two different points in time for the merging to start are identified and compared. The merging therefore either happens at $t = 5/\omega_\mathrm{p}$ or at $t = 20/\omega_\mathrm{p}$. This can be seen in the second and third row of the phase space plots in Fig. 3.13. The last row, for which the merging started at $t = 20/\omega_\mathrm{p}$, generates a very comparable phase space diagram in comparison to the reference case shown in the first row. The second row behaves much worse, with the phase space evolution being very different from the reference case. This is the same conclusion that can be derived from Fig. 7 and Fig. 8 in [29]. Since these plots only show a qualitative measure it is very hard to glean any more information from them.

In order to do that, more quantitative measures need to be established. The first step in this direction is shown in Fig. 3.14. It reproduces the analysis of Fig. 9 in [29]. The results of the new algorithm are very similar to the results of the $\mathbf{v}_r$ algorithm of Teunissen and Ebert, while they decisively outperform the results of all other shown algorithms. For these parameters and for this merge ratio (about $100000 \Rightarrow 80000$ quasi-particles per stream) the new algorithm is able to almost exactly preserve the momentum distribution. This is also related to the fact, that the final

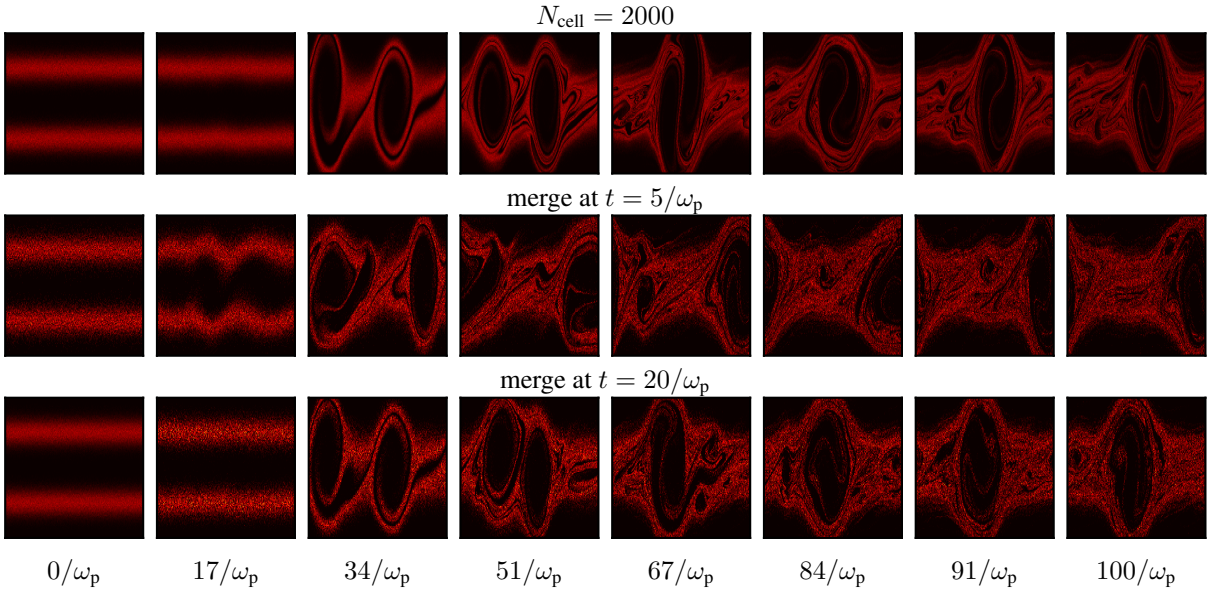

Fig. 3.13 – Phase space plots of the two-stream reference case. The horizontal axis is the x-axis and the vertical axis is the p_x-axis, conforming to the corresponding Fig. 7 and Fig. 8 in [29]. The columns correspond to different times in the evolution of the simulation, while the rows depict different starting times for the merging algorithm. The first row does not utilize any merging and instead uses 2000 quasi-particles per cell (1000 quasi-particles per stream and cell) for the whole simulation. It serves as a baseline for the phase space fidelity.

number of 160 quasi-particles per cell can very comfortably represent a two-stream scenario. In the last chapter, it was shown that much more restrictive scenarios are still sufficient to produce a very desirable fidelity in momentum space. As shown in Fig. 3.6, only about 44 quasi-particles per cell are needed to resolve the two-stream scenario.

The only difference between the red (with merging) and the blue (without merging) line in Fig. 3.14, is the size of the sampling standard deviation, given by the dashed lines and the colored regions. The standard deviation of the merge case is bigger. This is expected, as the case with merging has fewer particles available to represent the momentum distribution. As was shown in chapter 1, a smaller number of quasi-particles always leads to more noise in the simulation.

As a last step, a very rigorous quantitative measure is established. Equation 4 from [29] uses the relative difference of the velocity distribution function, given by

$$\frac{|\mathbf{f}_v(t) - \mathbf{f}_{v,0}(t)|}{|\mathbf{f}_{v,0}(t)|}, \tag{3.5}$$

where $\mathbf{f}_v(t)$ denotes the velocity distribution with merging, $\mathbf{f}_{v,0}(t)$ denotes the velocity distribution without merging and $|\cdot|$ is the L^2 norm, as a way to meaningfully adjudicate the quality of the merging algorithm. The quantities in this equation are vector quantities, as they hold a value

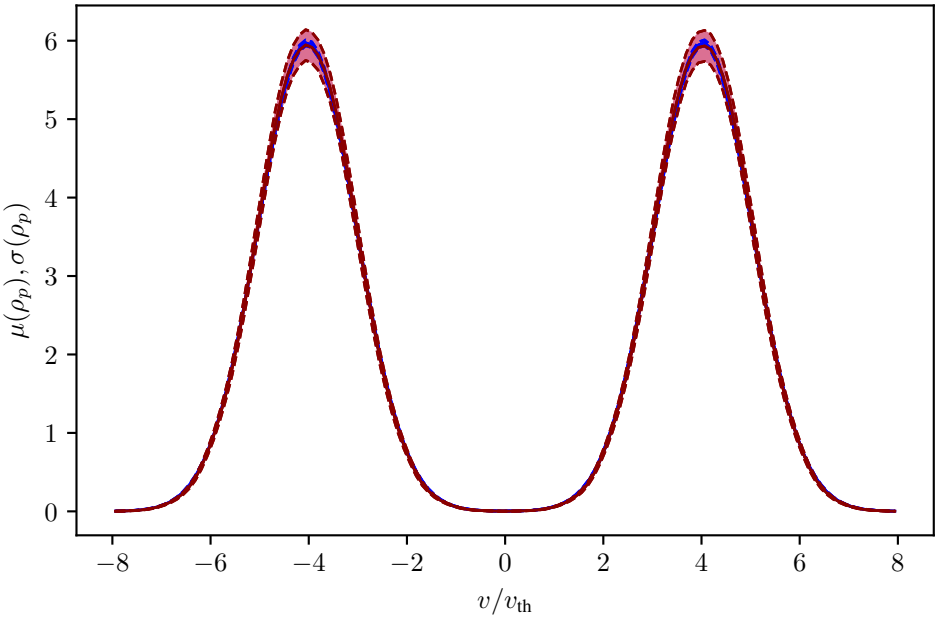

Fig. 3.14 – Results of the two-stream instability configuration as defined in [29], chapter 4.4, comparable to Fig. 9. The red line shows the mean momentum space density shortly after merging took place at $t = 5/\omega_{\text{p}}$, computed analogously to (3.1). The total number of particles decreased from 2000000 particles at timestep zero to an average of 160058±67 particles at timestep 1541. The blue line shows the mean momentum space density from the simulation without merging at the same point in time. The dashed lines give the standard deviation of their respective mean quantities, with the colored regions marking the distance to the mean value. In contrast to this figure, Fig. 9 in [29] does not show results without merging. A total number of $N_{\text{samples}} = 1056$ simulations were performed and averaged for both the red and the blue line. The particle numbers given here are a factor of two bigger in comparison to the numbers given in the caption of [29], Fig. 9. The reason for this is that the particle numbers given by Teunissen and Ebert are always per stream, while the numbers given here are for the whole simulation box.

for each of the 200 momentum bins in the simulation. The difference is calculated for each pair of momentum bins in $\mathbf{f}_v\,(t)$ and $\mathbf{f}_{v,0}\,(t)$. Using this equation, it is possible to attach a relative error to every timestep of the simulation. This is done in Fig. 10 of [29]. Fig. 3.15 reproduces this measurement, using the presented algorithm, and can therefore be directly compared to that figure. The only difference between the two results is the fact that the present result uses the momentum distribution instead of the velocity distribution. Since the relativistic factor is only $\gamma = 1/\sqrt{1-(4\bar{v}_{\text{th}})^2/c^2} \approx 1.009$, the considered quantity is a relative quantity and switching to the momentum only changes the employed bins and not the actual value of the distribution function, this will not meaningfully affect the results.

Compared to the number of samples in the paper $N_{\text{samples}} = 100$, a much higher number of samples $N_{\text{samples}} = 1056$ was used for the PSC case. This brings the measured empirical probability closer to the real probability. In order to achieve these results a complex array of run and

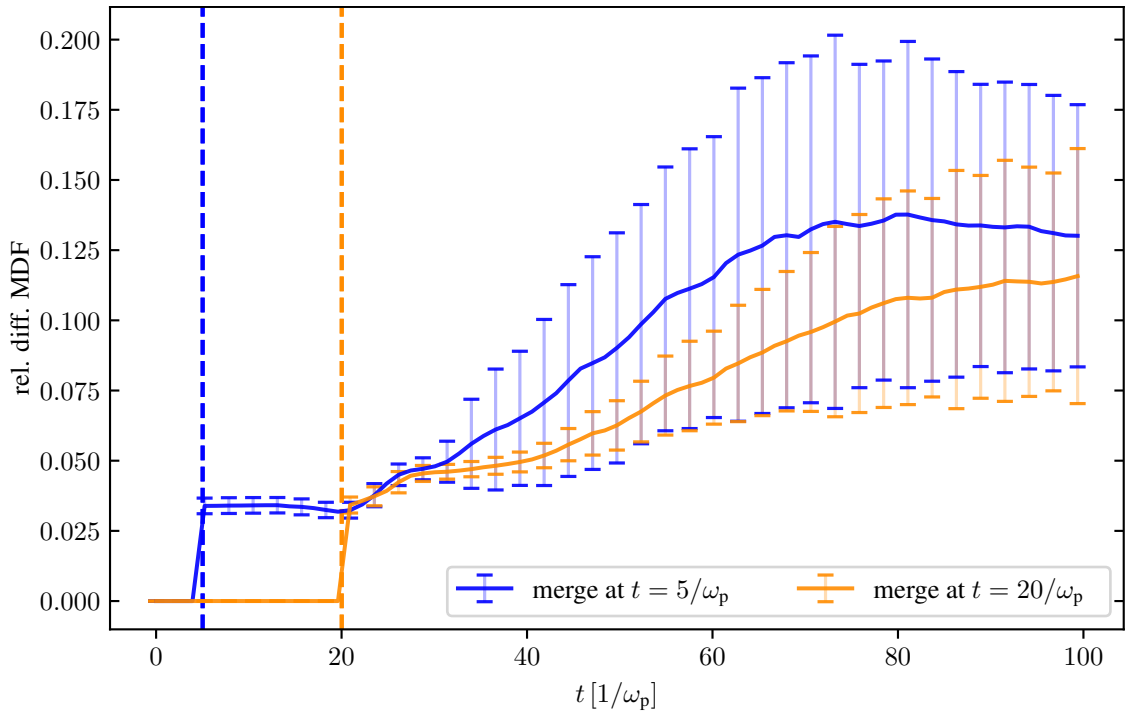

Fig. 3.15 – Comparison data for Fig. 10 in [29]. The blue line is the resulting mean relative difference in the momentum distribution function for merging at time $t = 5/\omega_\mathrm{p}$, the orange line corresponds to merging at time $t = 20/\omega_\mathrm{p}$. The capped vertical solid light blue and light orange lines represent the standard deviation of each mean value over all samples. The vertical dashed blue and orange lines give the starting point of the merging process. A total of $N_\mathrm{samples} = 1056$ simulations were run to generate these results.

post-processing scripts was necessary. Each sample, out of the total amount of 1056 samples, is processed in the following sequence:

1. Start a total of three runs, one for the case of no merging, one for the case of merging at $t = 5/\omega_\mathrm{p}$ and one for the case of merging at $t = 20/\omega_\mathrm{p}$, using the same random number seed for each of them. Employing the same random number seed guarantees an equivalent setup of the quasi-particles for each case.

2. For every 400^{th} timestep, the current quasi-particles are binned in 200 momentum bins. This gives two (for the different merging times) vector (for the different momentum bins) values for $\mathbf{f}_p\,(t)$ and one vector value for $\mathbf{f}_{p,0}\,(t)$ for each timestep.

3. After all three runs have finished, equation (3.5) can be used to compute the relative difference in the momentum distribution function for each timestep and for each of the two merging times.

The final relative differences of the momentum distribution function for each timestep and for all samples are then put together and the mean and standard deviation of these quantities are calcu-

lated over all samples. The shown results required simulations totaling ≈ 5.3 Kch to compute. The result of this whole process is shown in Fig. 3.15.

The mean error values are consistently below the best case values of [29]. The presented algorithm is a combination of all presented schemes in [29] but has the additional feature of being free of spurious divergence. Through combining the different schemes from [29], the algorithm presented in this dissertation achieves improved results. Additionally, it is not necessary to choose a specific scheme at the start of the simulation, which was necessary for the algorithm presented in [29]. The different schemes are summarized in the caption of Table 1 in [29]. It can be hard to adjudicate if the investigated physical process favors a scheme that conserves the energy, or that conserves the momentum, or that uses randomly chosen previous velocities (which are optionally scaled to conserve energy). The new algorithm simultaneously achieves the properties of all the different schemes and surpasses their performance when regarding the difference of the momentum distribution function measure.

4 Generation of controllable plasma wakefield noise in particle-in-cell simulations

After giving a theoretical model of quasi-particle noise in chapter 1 and following that up with the description of a novel algorithm to adapt and modify this noise in chapter 2 and chapter 3, this chapter and chapter 5 will outline an experimental application of the previous work. This application will not include the actual merging/splitting algorithm. Nonetheless, this algorithm may be useful for future work, as, for example, the particle injection may be heavily influenced by the amount of quasi-particles in the injected beam. The experimental application was done within the AWAKE ("Advanced Proton Driven Plasma Wakefield Acceleration Experiment") collaboration. In order to become a part of this collaboration it was necessary to prove the capabilities of the PSC. Furthermore, the specific goal of this participation needed to be established and well justified. This chapter will describe the process of earning the trust of the collaboration and of nailing down the targets of the cooperation. A peer-reviewed publication, a result of this cooperation, will be presented in the last section. This chapter aims at giving a thorough understanding of the respective share of the work for this publication that was completed by Nils Moschüring. To this end, it is written in the first person.

4.1 Becoming a member of AWAKE

I followed the progress of the AWAKE collaboration very closely during my time at the LMU. Through regular participation at the annual collaboration meetings, where I gave frequent talks on my progress, the simulation coordinator, Konstantin Lotov, was able to make sure my targets aligned well with the targets of the collaboration. Leading up to the first (found in this chapter) and second (found in chapter 5) peer-reviewed publication, I participated in five AWAKE collaboration meetings: The meeting in Lisbon in June 2012, the meeting in Düsseldorf in June 2013, the meeting at CERN in March 2015, the meeting in Lisbon in March 2016 and the meeting at CERN in September 2018. Each time, I gave a detailed report of my progress. I also presented the results and status updates on two international conferences, the Conference on Computational Physics in July 2016 in Johannesburg, South Africa, and the Laser-Plasma Accelerator Workshop 2019 in May 2019 in Split, Croatia.

My first presentation, which can be viewed as an application to participate in the collaboration, was in June 2012, at the collaboration meeting in Lisbon. This meeting was hosted by the IST, which is home to the OSIRIS PIC code, and my talk was a basic introduction to the PSC and its capabilities. I also presented some very early simulation results of a 2D simulation, with parameters that were close to some AWAKE models, of which you can find a sample in Fig. 4.1. Resulting from this first meeting, the determination to work together was formed. In order to qualify as a contributor, our simulation capabilities needed to be up to par. To test this, Konstantin Lotov supplied multiple fixed criteria the PSC had to fulfill in order to be accepted as a source of trusted results. These criteria are i) E_z wakefield strength stability with driver distance, ii) the reproduction and stability of the plasma wavelength in the resulting wakefield, iii) an acceptable amount of lateral wakefield deformation with propagation distance and iv) an acceptable amount of numerical heating. After the collaboration meeting, Tobias Kleine,

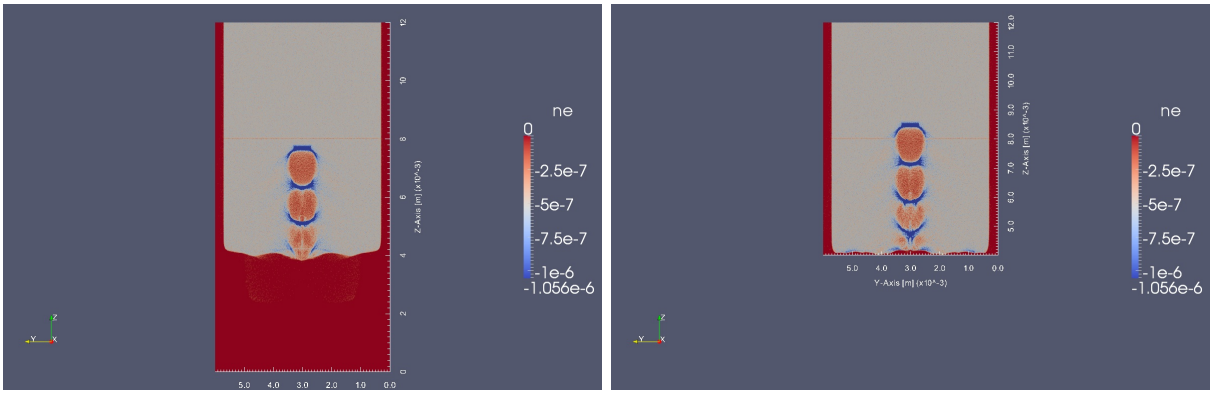

Fig. 4.1 – Two figures from my presentation at the IST in Lisbon in 2012. Two consecutive points in time of a 2D simulation of proton-driven wakefield generation are shown. The application of dynamic patches can be seen in the switch-off of the lowest patch in the figure on the right.

a diploma student, investigated these issues in collaboration with me. From this work Tobias Kleine achieved his diploma in 2013 [11].

Chapter 6 in [11] discusses the wakefield stability. In this chapter, Fig. 6.1 and Fig. 6.2 as well as Table 6.1 are of special interest. They depict the E_z stability, in terms of the field strength and the wavelength, as well as the lateral field deformation with the propagation distance. Additionally, Table 1 shows the generated plasma wavelength. The work of Tobias Kleine [11] discusses the numerical heating properties of the code as well. Fig. 5.9 in chapter 5 as well as Fig. 6.5 in chapter 6 and the contents of chapter 7 contain an in-depth investigation into this problem. The different benchmark results demonstrate the good performance of the PSC. They are also very important in order to gauge the required resolutions to achieve the desired faithfulness.

I presented the results of these benchmarks at the next collaboration meeting in June 2013 at the Heinrich-Heine University in Düsseldorf. The results were welcomed and deemed sufficient to show that the PSC is a good candidate for providing results for the AWAKE effort.

4.2 Enabling a large-scale simulation of the AWAKE baseline case

With this confirmation and validation I started investing more time into preparing large-scale simulations. The full resolution requires multiple Mch of computational resources and was therefore not suitable as a test system for the simulations. Therefore, I used smaller resolutions in order to regularly verify my efforts. It took some time to prepare a simulation test case that offered all the necessary parameters and capabilities. This case should offer multiple charged particle beams

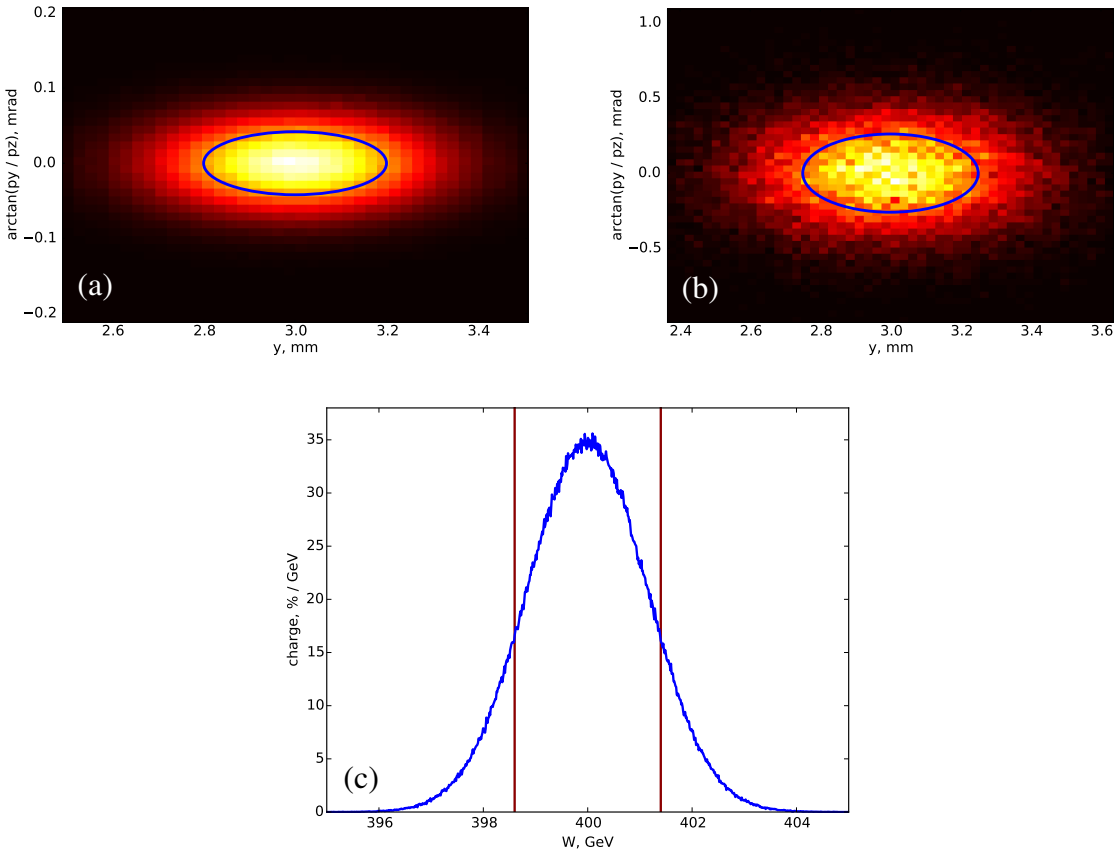

Fig. 4.2 – Verification of the setup of (a) the proton transversal emittance, where the area A enclosed by the blue ellipsis satisfies $\epsilon_{nb} = \gamma_b\beta_b\frac{A}{\pi} = \gamma_b\beta_b\sigma_\theta\sigma_{rb} = \gamma_b\beta_b\,0.2\,\text{mm}\,0.042\,\text{mrad} \approx 3.6\,\text{mm mrad}$, (b) the electron transversal emittance, where the area A enclosed by the blue ellipsis satisfies $\epsilon_{ne} = \gamma_e\beta_e\frac{A}{\pi} = \gamma_e\beta_e\sigma_\theta\sigma_{re} = \gamma_e\beta_e\,0.25\,\text{mm}\,0.26\,\text{mrad} \approx 2\,\text{mm mrad}$ and (c) the proton longitudinal emittance, where the red lines signify the energies $E = 400\,\text{GeV} \pm \frac{0.35}{100}400\,\text{GeV}$.

with a full configuration (population N, length σ_z, radius σ_r, energy W, energy spread δW and angular spread $\delta\alpha$) and a plasma column with a complex density profile.

Results for this new simulation case, using an updated baseline configuration, were presented by me at the AWAKE collaboration meeting in March 2015 at CERN. The updated baseline configuration uses on-axis injection of the witness electron beam, which makes it much simpler than the final configuration. The simulation contains the full 10 m propagation, but uses a reduced resolution of $N_{\text{per}\lambda_\text{p}} = 51$, which makes its runtime much shorter than the final configuration will require. Since the runtime is directly proportional to the resolution to the fourth power, the runtime with the reduced resolution is only $(51/130)^4 \approx 2.4\,\%$ of the runtime of the final configuration. In this case, the plasma takes the shape of a truncated cone. The remaining parameters can be found in Table 1 in [17]. In order to carefully verify the validity of this

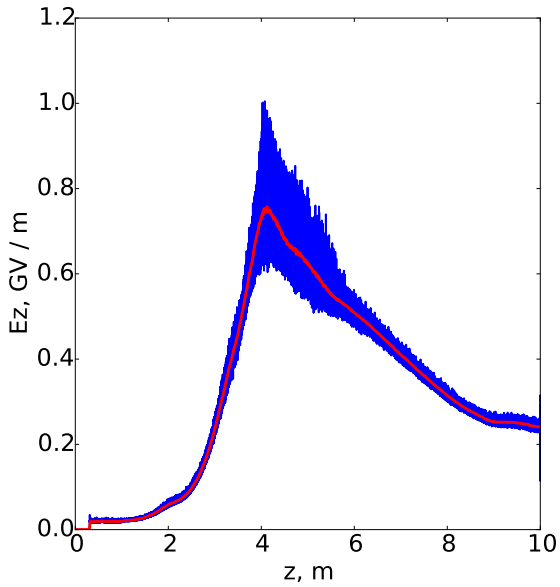

Fig. 4.3 – Verification of the E_z field evolution. The plot shows the results of my test case, where $E_z(z) = \max_{x,y,t} E_z(x,y,z,t)$ and other parameters as given in Table 1 in [17]. The red line in this plot is a moving window average of the data plotted in blue. It can be compared directly with Fig. 1 (b) in [16], which shows results of LCODE simulations using different values for the proton bunch population N_b. The red line, with $N_b = 3 \times 10^{11}$, in Fig. 1 (b) of [16] corresponds to the data in this plot.

improved test case I prepared a couple of diagnostics. Fig. 4.2 shows the results of applying these diagnostics. They focus on the correctness of the emittance as this is an important parameter, which is not straightforward to implement. All diagnostics verified that the improved test case represented the physics correctly.

I was also able to produce some meaningful results from this simulation. Fig. 4.3 shows the evolution of the maximum in-plasma E_z field value over all time and lateral space in the simulation. It can be directly compared with Fig. 1 (b) in [16]. This is very important data as a correctly-simulated maximum E_z field value depends on a multitude of non-linear processes.

It includes the correct interaction of the proton beam with the plasma, which needs to exhibit the self-modulation instability, followed by the de-phasing of the proton-beam particles, which leads to the subsequent dissipation of the generated strong wakefield. In summary, in order to reproduce the correct E_z field values the whole highly non-linear interaction of the plasma and the ion-beam has to evolve correctly.

Another important comparison is made possible by the data plotted in Fig. 4.4. This figure shows the final energy of the witness electrons of the simulation. It can be directly compared with Fig. 5 (b) in [17], which shows results of LCODE and Osiris. The final witness beam energy depends on the non-linear processes as well and is therefore a very good measure for the overall

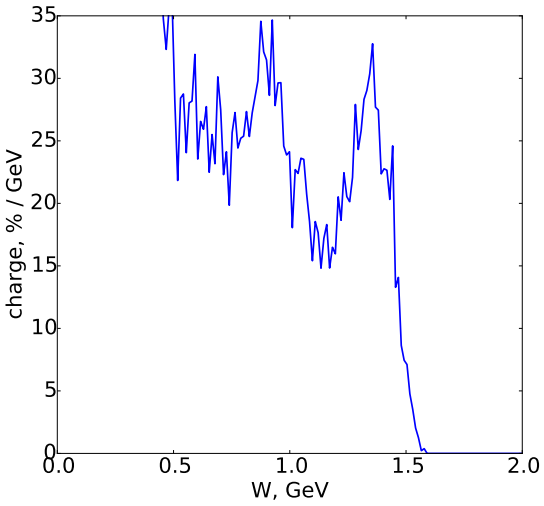

Fig. 4.4 – Verification of the final electron energy of the simulation. The plot shows the final witness electron energy of the PSC simulation. These results can be directly compared with Fig. 5 (b) in [17], which shows comparable results of LCODE and Osiris. Since the PSC simulation uses a periodic boundary in the transversal directions, the re-entering of witness particles was possible. This leads to the non-zero values in the left part of the plot. Particles can only exit the simulation through deceleration, which rarely happens.

correctness of the setup. In summary, Fig. 4.3 and Fig. 4.4 lead to the conclusion that the PSC, equipped with the new test case, is able to reproduce complex processes and to generate results that are equivalent to trusted community codes.

Equipped with these good results, a proposal for a new and bigger simulation was made at that same meeting at CERN in 2015. After a lengthy application for computing time through the GCS (Gauss Centre for Supercomputing) Call for Large-Scale Projects, a total budget of 35 Mch was acquired. A major part of this budget was reserved for AWAKE collaboration simulations.

With these resources, the following general simulation outline is conceivable: i) full 3D, ii) fully kinetic, iii) non-quasistatic, iv) 130 grid-points per plasma wavelength and v) three quasi-particles per cell per species. A simple cost estimate was presented at the conference, omitting all output and field solving as well as the proton and witness beam simulation, since they represent four orders of magnitude fewer quasi-particles in comparison to the plasma.

Using the parameters given in Table 1 in [17], the plasma filled truncated cone volume is calculated to be $V = L_{\text{max}}\pi(r_0^2 + r_0 r_1 + r_1^2)/3 \approx 5 \cdot 10^{-5}\text{m}^3$. For $\Delta x = \lambda_\text{p}/N_{\text{per}\lambda_\text{p}}$, $\Delta t = k_\text{cfl}\sqrt{\Delta x^2/(3c^2)}$, where k_cfl is the Courant–Friedrichs–Lewy factor [4], an estimated number of particles pushed per second per core of $f_\text{pps} = 1.2 \cdot 10^6$, the computational time per core can

be estimated by

$$t_{\text{pc}} = \frac{V N_{\text{ppc}}}{\Delta x^3 f_{\text{pps}}} \frac{L_{\text{max}}}{c\Delta t} \frac{L_{\text{window}}}{L_{\text{max}}} = \frac{\sqrt{3} V L_{\text{window}}}{\lambda_{\text{p}}^4 f_{\text{pps}} k_{\text{cfl}}} N_{\text{per}\lambda_{\text{p}}}^4 N_{\text{ppc}} \tag{4.1}$$

$$\Rightarrow t_{\text{pc}} = 3.15 \cdot 10^{-3} \cdot N_{\text{per}\lambda_{\text{p}}}^4 N_{\text{ppc}}\,\text{ch} = 2.7\,\text{Mch}. \tag{4.2}$$

The plan was to run on four islands of SuperMUC phase 1, which total 32768 cores, giving an estimated walltime of the simulation of $t_{\text{W}} = 2.7 \cdot 10^6/32768\,\text{cpuh} \approx 82\,\text{cpuh}$. At a cost of $\approx$ 0.01 - 0.013 € per 1 ch [20] one run of the simulation was estimated to cost about 27000 - 35000 €. The simulation would need about 2 TB out of the available 50 TB of memory. It was anticipated that these simulations would be finished in 2015.

The justification to perform these simulations is given in the following. The simulations would help to again justify the remaining assumptions in baseline simulations, which are the sufficiency of the quasistatic and cylindrically symmetrical two-dimensional approach. Furthermore, it would be possible to study the transversal beam filamentation for the first time. They would also enable a direct code comparison to other simulation codes and the direct comparison of PSC results with potential measurements. Finally, large-scale PIC simulations are relatively rare and it would be worthwhile to gather some experience with them. There were also other possible options, including the study of the plasma density step behavior [15], but in the end we settled on the large-scale simulation.

4.3 Quasi-particle noise and lateral beam filamentation

In May 2015, Konstantin Lotov proposed a road-map to properly investigate the shot-noise to lateral beam filamentation [10] relation in simulations and extrapolate possible findings to the experiment.

The lateral beam instability leads to an effect called hosing or beam filamentation. The ion beam breaks up laterally and dissolves into filaments. If this effect would happen in an experiment it would be very detrimental. In order to minimize the risk of this happening the baseline case incorporates a relatively low plasma density and ion beam radius, which are decisive parameters for this effect. The side-effect of this generous safety margin is a greatly reduced accelerating potential. An improved understanding of the lateral beam filamentation would enable an increase in plasma density or ion beam radius resulting in a much higher acceleration of the witness beam.

Unfortunately, the source of this non-linear effect is the shot-noise, which is not an easily controllable parameter. This is unlike the source of the self-modulation instability, which is triggered by a seed perturbation, in AWAKEs case the half-cut of the beam. Since noise is always massively over-estimated in PIC simulations, the predictions derived from the simulations only offer an upper bound for the onset of the filamentation instability. If specific parameters do not lead to beam filamentation in the simulation it is almost guaranteed that it will work in an experiment

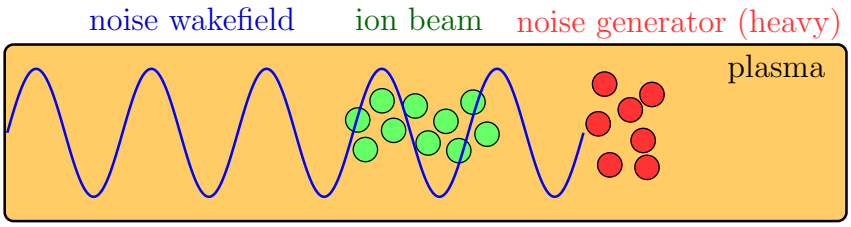

Fig. 4.5 – Sketch of the setup of controlled noise simulations.

as well. But finding the beam filamentation threshold in simulations will not provide any information about the threshold in the experiments. The simulations will produce beam filamentation much earlier than the experiments, due to their increased shot-noise. Therefore, the experiments may very well be fine for much larger plasma densities, since their shot-noise is much lower. Finding the safe plasma density or ion beam radius threshold would enable increased acceleration gradients, as shown in Fig. 1 (a) in [16]. Increased acceleration gradients would, in turn, lead to higher final witness energies, which is very important for AWAKE.

Since the lateral beam instability competes with the desired self-modulation instability, both must be simulated at the same time. This means that this problem requires a full 3D, fully kinetic simulation. Furthermore, solving this issue needs many medium-scale runs and some large-scale runs, in other words a large amount of computing resources, which makes this problem a good fit for the GCS budget.

The first step on Konstantin Lotov's proposed road-map is finding a way to introduce controlled noise into the simulation. The lateral beam filamentation is triggered by the noise of the randomly distributed beam particles, the so-called shot-noise. If we can understand, generate and control this shot-noise we achieve controlled noise. The straightforward way to generate this controlled noise is by inserting randomly distributed infinitely heavy particles in front of our setup (c.f. to Fig. 4.5).

After implementing a proper way to achieve controlled noise, the road-map to higher accelerating gradients in AWAKE would follow the sketches in Fig. 4.6. The well-understood controlled noise level may therefore serve to find the scaling of the filamentation threshold with the different parameters of the simulation. Using experimental beam noise thresholds, these scalings can then be corrected and improved. Finally, we can extrapolate the corrected scalings of the controlled noise simulations to predict filamentation thresholds for AWAKE, which has a much lower noise level than the simulations. These predictions should then be verified using large-scale AWAKE baseline simulations with under- and over-threshold plasma densities. It should then be possible to demonstrate energy gains that exceed the baseline values.

The above road-map needs an additional simulation capability. For the task of understanding controlled noise and scanning the different parameter spaces for their filamentation threshold, a simpler test case, in comparison to the AWAKE baseline case, is utilized. This simpler test case was quick and easy to implement. It uses a constant current beam, which means that the beam

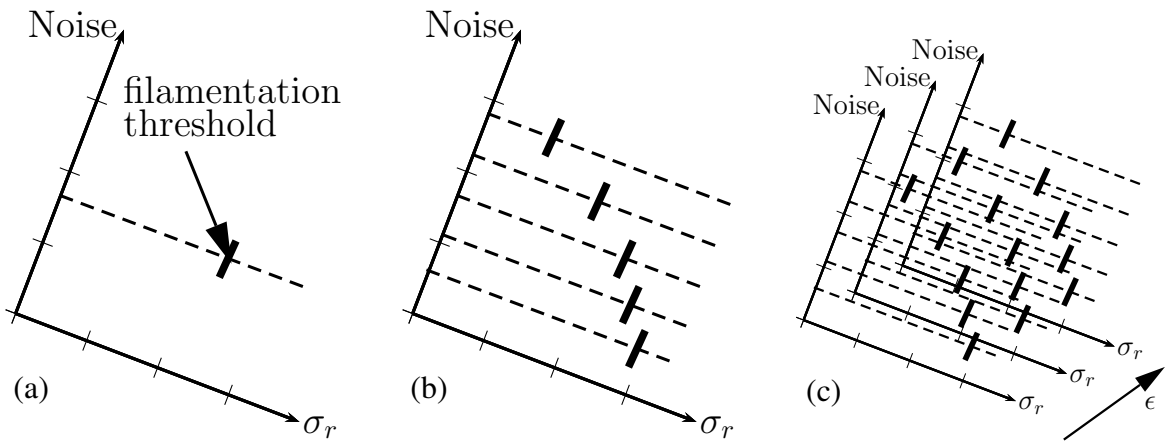

Fig. 4.6 – Three sketches illustrating the road-map to exploit controlled noise for predicting the lateral beam filamentation instability emergence. In the plots σ_r is the beam radius and ϵ is the beam emittance. Sketch (a) describes scanning the beam radius in order to find the filamentation threshold. This is then repeated for different noise levels in sketch (b), enabling intra- and extrapolation of the filamentation threshold in relation to the noise level. Finally, sketch (c) shows a scan over the emittance, resulting in predictive capabilities for the filamentation threshold in relation to the beam radius and the emittance.

has a longitudinally uniform density. The radial dependency is still Gaussian. The maximum beam density n_{b0} is again very small in comparison to the plasma density. This results in a completely linear plasma response to the beam. Furthermore, only the first 15 periods of the plasma wavelength are simulated, and a very fast $\gamma = 400$ positron beam, with a very small emittance, is used. This makes the simulations faster as the beam keeps pace with the light.

With this extra test case, we were ready to tackle the first item on the road-map. The achievement of properly understanding and generating controlled noise is validated by a publication [21], which is also included in this dissertation in chapter 4.4. All simulation results for this paper have been created and post-processed by me, using the previously explained simple test case. I also participated heavily in optimizing and enhancing the PSC itself. Chapter 1, the introduction of the paper, was written by me and Konstantin Lotov. In chapter 2 the main idea of the paper is explained. While this chapter was written by Konstantin Lotov, and the idea of cosine-shaped rods, which serve to get rid of spurious Cherenkov radiation, was brought up by him as well, my simulations were key in recognizing the problem and subsequently defining and solving it. Chapter 3 was written mainly by Konstantin Lotov, with the mathematical expressions verified analytically and numerically by me. Chapter 4 was written mainly by me, except for the last part, starting with formula (15), which was written by Roman Spitsyn. The summary was a joint effort of all authors. Fig. 3, Fig. 5, Fig. 6, and their respective captions have been created and written by me. Fig. 7 uses the results from my simulation, including its post-processing. I contributed heavily in the editing of the whole paper, restructuring sentences and correcting mistakes. In the end, I was very happy to achieve a peer-reviewed publication participating in this international collaboration.

4.4 Full-text of the publication

Reproduced from Physics of Plasmas 24, 103129 (2017), https://doi.org/10.1063/1.4986399, with the permission of AIP Publishing.

Physics of Plasmas

Generation of controllable plasma wakefield noise in particle-in-cell simulations

Cite as: Phys. Plasmas **24**, 103129 (2017); https://doi.org/10.1063/1.4986399
Submitted: 05 June 2017 . Accepted: 05 October 2017 . Published Online: 20 October 2017

N. Moschuering, H. Ruhl, R. I. Spitsyn, and K. V. Lotov

PHYSICS OF PLASMAS **24**, 103129 (2017)

Generation of controllable plasma wakefield noise in particle-in-cell simulations

N. Moschuering,[1] H. Ruhl,[1] R. I. Spitsyn,[2,3] and K. V. Lotov[2,3]
[1]*Ludwig-Maximilians-Universität, 80539 Munich, Germany*
[2]*Budker Institute of Nuclear Physics SB RAS, 630090 Novosibirsk, Russia*
[3]*Novosibirsk State University, 630090 Novosibirsk, Russia*

(Received 5 June 2017; accepted 5 October 2017; published online 20 October 2017)

Numerical simulations of beam-plasma instabilities may produce quantitatively incorrect results because of unrealistically high initial noise from which the instabilities develop. Of particular importance is the wakefield noise, the potential perturbations that have a phase velocity which is equal to the beam velocity. Controlling the noise level in simulations may offer the possibility of extrapolating simulation results to the more realistic low-noise case. We propose a novel method for generating wakefield noise with a controllable amplitude by randomly located charged rods propagating ahead of the beam. We also illustrate the method with particle-in-cell simulations. The generation of this noise is not accompanied by parasitic Cherenkov radiation waves. *Published by AIP Publishing.* https://doi.org/10.1063/1.4986399

I. INTRODUCTION

Understanding a collective interaction of relativistic charged particle beams with plasmas is important for a wide variety of physical problems. Among them are space plasmas,[1] fast ignition schemes for inertial fusion,[2,3] turbulent plasma heating for magnetic fusion,[4–6] positron bunch instabilities driven by an electron cloud in colliders,[7] plasma wakefield acceleration,[8–10] and many others. In most cases, the interaction has the form of an instability that develops starting from some low-amplitude shot noise.

A realistic noise level[11] is difficult to reproduce in numerical simulations, since the number of simulated quasi-particles (or macro-particles) is usually much smaller than the number of real particles in the system. Fewer quasi-particles, each carrying a larger charge, produce random fields with amplitudes which are orders of magnitude too high in comparison with the experiment. This, in turn, may result in faster growth of unstable perturbations and quantitatively wrong simulation results.

Since one-to-one simulations of typical beam-plasma systems of interest fall far beyond state-of-the-art computational capabilities, the only available choices are ordered initial distributions of particles[11] and extrapolating high-noise simulation results to low-noise physical systems. The second approach is schematically illustrated in Fig. 1. Obviously, it would benefit from having several simulation points with different noise levels. This makes it desirable to develop a simple method with the capability to produce a noise field with a controllable amplitude. For reliable extrapolation, the noise level must be controlled independently of key simulation parameters, like the grid resolution or the number of quasi-particles.

For many instabilities of relativistic beams, the noise harmonics of interest are potential (Langmuir) plasma waves with a phase velocity which is close to the speed of light c and the wavevector $\vec{k}$ directed along the axis of beam propagation. These waves are excited by individual beam particles just as regular plasma wakefields are excited by the drive beam as a whole. We will call these waves wakefield noise. Properties of the wakefield noise were studied in Ref. 11. At first glance, a controllable wakefield noise might be easily excited by an ensemble of randomly located point-like quasi-particles propagating ahead of the beam (Fig. 2) or inside the beam. By changing the charge of the quasi-particles and their number, it is possible to control the noise level. However, relativistic quasi-particles usually emit numerical Cherenkov radiation, if simulated using particle-in-cell (PIC) codes.[12–15] This radiation critically affects a clean study of the beam instability and must therefore be avoided.

In this paper, we propose a novel method for generating wakefield noise with a controllable amplitude by randomly located charged rods (Sec. II). The noise is free from parasitic Cherenkov radiation. We also give expressions for the relations between the amplitude of the noise field and the rod parameters (Sec. III) and illustrate the method using particle-in-cell (PIC) simulations (Sec. IV). The main findings are summarized in Sec. V. Studies of particular beam instabilities fall beyond the scope of this paper.

For all simulations detailed in this paper we use the PIC code PSC.[16] The coordinates are either Cartesian (x, y, z) or cylindrical (r, ϕ, z) with the z-axis being the direction of beam propagation. The co-moving coordinate $\xi = z - ct$ is used wherever convenient.

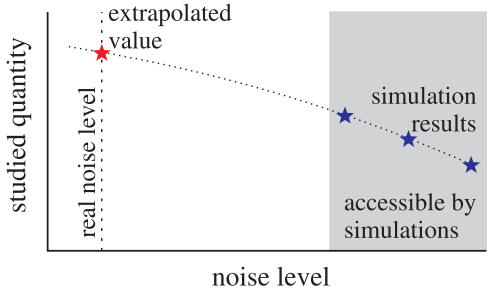

FIG. 1. Illustration of the extrapolation approach.

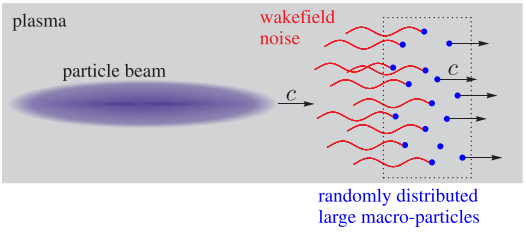

FIG. 2. Illustration of the controllable noise generation.

II. THE IDEA OF CHARGED RODS

The problem of numerical Cherenkov radiation depends on the details of the numerical solver. In the present paper, we use the finite-difference time-domain (FDTD) scheme. The discretization of the FDTD scheme leads to a phase velocity of the propagated radiation, which is strictly smaller than the velocity of light. A quasi-particle with sufficient energy can therefore travel faster than the radiation on the grid. This leads to numerical Cherenkov radiation, as illustrated in Fig. 3(a). A point-like charge Q moving with the speed of light emits short-wavelength electromagnetic radiation, the amplitude of which is much higher than the amplitude of the useful plasma wave. The radiation wavelength is of the order of the grid size and is much shorter than the plasma wavelength $\lambda_p = 2\pi k_p^{-1}$. The plasma wavelength is determined by the plasma density n_0 through the plasma wavenumber $k_p = \sqrt{4\pi n_0 e^2/(mc^2)}$, where e is the elementary charge and m is the electron mass. The amplitude of the emitted Cherenkov electromagnetic waves is proportional to the amplitude of short-wavelength harmonics in the Fourier spectrum of the emitting charge. More specifically, the z-component of the wavevector is of importance. This already hints at the idea of how the numerical Cherenkov radiation can be reduced in comparison to the longer-wavelength plasma waves: The radiating source must be long and smooth. In other words, it should be shaped like a charged rod.

We consider two variants of rods: (i) rectangular rods with the linear charge density

$$\lambda(\xi) = k_p Q/\pi, \quad -\lambda_p/2 < \xi - \xi_0 < 0, \tag{1}$$

and (ii) cosine-shaped rods with

$$\lambda(\xi) = \frac{k_p Q}{\pi}(1 - \cos(2k_p(\xi - \xi_0))), \quad -\lambda_p/2 < \xi - \xi_0 < 0. \tag{2}$$

For efficient excitation of Langmuir waves, the rod length must be $\lessapprox \lambda_p$. We set it to $\lambda_p/2$. This value does not maximize the wakefield of the rod and is chosen only for the simplification of the subsequent analytical calculations it provides.

The Fourier spectra

$$F(k_z) = \left| \int_{-\infty}^{\infty} \lambda(\xi) e^{ik_z \xi} d\xi \right| \tag{3}$$

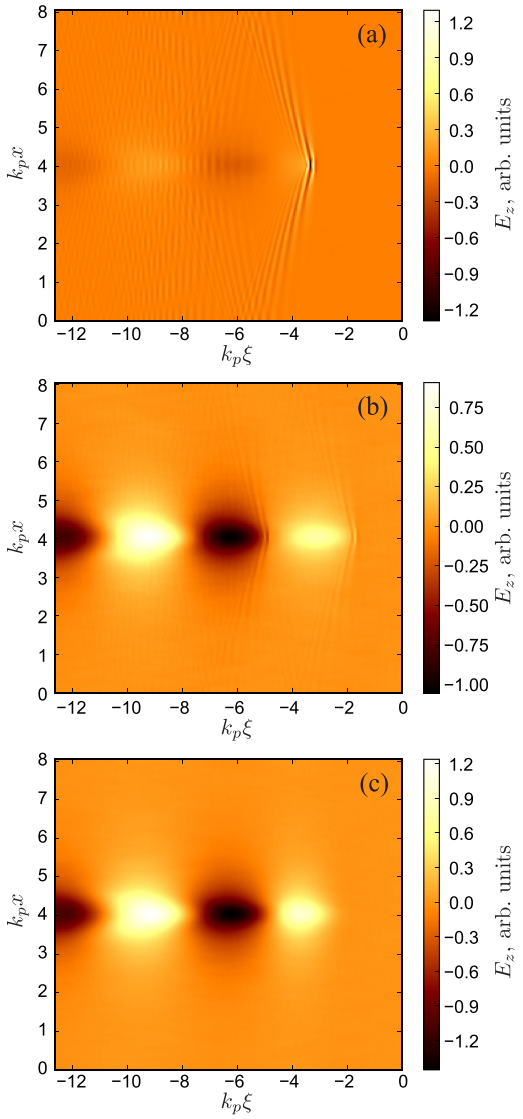

FIG. 3. Simulated wake patterns for moving objects of different shapes: point-like charge (a), rectangular-shaped rod (b), and cosine-shaped rod (c).

of these rods and for the point charge Q are shown in Fig. 4. For the wave number of plasma waves holds that $k_z \approx k_p$ (vertical dotted line in Fig. 4) and they are excited equally well by all charge distributions. Numerical Cherenkov waves for typical resolutions of PIC code simulations have wave numbers for which $k_z/k_p \sim 10^2$–10^3 (shaded area in Fig. 4) holds, and their excitation is suppressed by some orders of magnitude in the case of cosine-like rods. The smoothness degree of the charge distribution $\lambda(\xi)$ determines the decrease rate of the spectrum $F(k_z)$ for high k_z. In the case of the cosine-like distribution, this decrease rate is given by $F(k_z) \propto k_z^{-3}$.

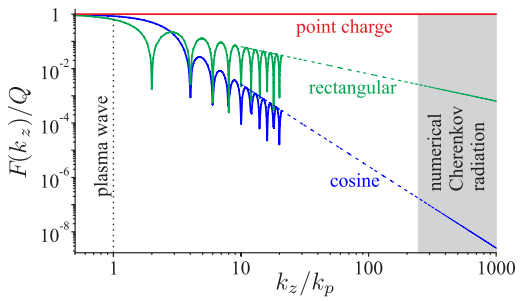

FIG. 4. Fourier spectra of various charge distributions. For large k_z, only the envelopes of the oscillating spectra are shown.

A "beam" of equally charged rods would produce not only the noise field, but also a regular wakefield which is excited by the total rod charge. This is undesirable. To get rid of the regular field, the sign of the charge of the different rods must be chosen randomly.

Another necessary condition for achieving a correct noise is the equivalence of all wakefield phases. The contributions of different rods must, on average, uniformly cover the whole period of the plasma wave. Using alternating rod charges the minimum length of the rod area is $\lambda_p/2$, as rods of opposite charge contribute to opposite half-periods of the wave.

Note that an exact nulling of the total charge of the rod ensemble, for example, by choosing an equal number of positively and negatively charged rods, is erroneous. This additional constraint on the rod ensemble would break the equivalence of wakefield phases. This can be easily seen in the extreme case of two oppositely charged rods.

III. AMPLITUDE OF THE WAKEFIELD NOISE

In this section, we will analytically calculate the root mean square (rms) of the longitudinal electric field excited by an ensemble of cosine-shaped rods (2). Assume that the rod heads (characterized by random coordinates $\vec{r}_{\perp 0}$ and ξ_0) are uniformly distributed in the z-direction over an infinitely wide layer of thickness πk_p^{-1}. The average density in this region is given by $2n$, with n being the average number density of rods of each charge sign (Q or $-Q$).

We will first find an expression for the wakefield of a single rod. Given the charge density of the rod

$$\rho(\vec{r}_\perp, \xi) = \delta(\vec{r}_\perp)\lambda(\xi), \quad \vec{r}_\perp = (x, y) \tag{4}$$

the longitudinal component of the wakefield is[17]

$$\begin{aligned} E_z(\vec{r}_\perp, \xi) &= 2k_p^2 \int_\xi^\infty d\xi' \int d\vec{r}'_\perp \, \rho(\vec{r}'_\perp, \xi') \\ &\quad \times K_0\left(k_p|\vec{r}_\perp - \vec{r}'_\perp|\right)\cos\left(k_p(\xi - \xi')\right) \\ &= \frac{2k_p^2 Q}{\pi} K_0(k_p r) G(\xi - \xi_0), \end{aligned} \tag{5}$$

where K_0 is the modified Bessel function of the second kind, and

$$\begin{aligned} G(\xi) &= k_p \int_{\max(-\pi k_p^{-1}, \xi)}^0 d\xi' (1 - \cos(2k_p\xi')) \cos(k_p(\xi - \xi')) \\ &= \begin{cases} -\frac{8}{3}\sin(\xi), & k_p\xi < -\pi, \\ \frac{2}{3}(\sin(2\xi) - 2\sin(\xi)), & -\pi < k_p\xi < 0, \\ 0, & \xi > 0. \end{cases} \end{aligned} \tag{6}$$

In Ref. 11, the rms noise field of N charges is calculated as

$$E_{\text{rms}}^2 = N\langle E_z^2 \rangle, \tag{7}$$

where the angle brackets denote the averaging of the field of a single charge over possible locations of this charge with respect to the observation point. We have to modify our approach in this case, as we take the number of rods to be infinite, which makes the average field vanish. Assume the rods occupy a cylindrical area with a large radius R. Then

$$N = 2\pi^2 k_p^{-1} R^2 n \tag{8}$$

and the average field

$$\begin{aligned} E_{\text{rms}}^2(\xi) &= N\left\langle \left(\frac{2k_p^2 Q}{\pi} K_0(k_p r) G(\xi - \xi_0)\right)^2 \right\rangle \\ &= 2n \frac{4k_p^4 Q^2}{\pi^2} \int_0^R 2\pi r K_0^2(k_p r) dr \int_\xi^0 G^2(\xi - \xi_0) d\xi_0 \\ &= \frac{8k_p^2 Q^2 n}{\pi} (1 + k_p^2 R^2 (K_0^2(k_p R) - K_1^2(k_p R))) \\ &\quad \times \int_\xi^0 G^2(\xi - \xi_0) d\xi_0. \end{aligned} \tag{9}$$

In the limit $R \to \infty$ it holds that

$$E_{\text{rms}}^2(\xi) = \frac{8k_p^2 Q^2 n}{\pi} \int_\xi^0 G^2(\xi - \xi_0)\, d\xi_0, \tag{10}$$

and for $\xi < -\pi k_p^{-1}$ (trailing the rods)

$$E_{\text{rms}}^2(-\infty) = \frac{256 k_p Q^2 n}{9}. \tag{11}$$

IV. WAKEFIELD NOISE IN PIC SIMULATIONS

We now describe our approach to the production of wakefield noise in three-dimensional PIC simulations, how the noise looks like, and which additional actions are required to observe a close quantitative agreement between simulation results and the developed theory.

We configure the code to use widespread algorithms: a standard Boris pusher[18] with second order particles to perform particle pushing and a standard FDTD-scheme[19] to perform field pushing. The simulation window moves with the speed of light in a patch-based manner, that is, by appending a simulation grid and quasi-particles on one end of the box

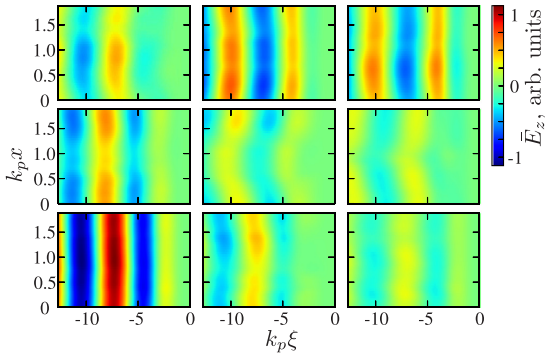

FIG. 5. The longitudinal wakefield $\bar{E}_z(x, y, \xi)$ at $y = 0.15\lambda_p$ for several rod distributions.

and detaching the same amount of volume on the opposite side. This process is iterated continuously. The window size is given by $X \times Y \times Z$ with $X = Y = 0.3\lambda_p$, $Z = 2.02\lambda_p$. The boundary conditions are periodic in transverse dimensions (x and y) and reflecting in the z dimension. The simulated propagation length, moving in the z direction, is $10\lambda_p$. Initially, the plasma is cold and uniform. It is composed using 3 quasi-particles per cell for the electrons, while the ions are not simulated and treated as immobile charges. The spatial resolutions are $\Delta x = \Delta y = \Delta z = \lambda_p/130 \approx 0.05k_p^{-1}$ and the time step is $\Delta t = 0.99\Delta x/\left(c\sqrt{3}\right) \approx 0.28\omega_p^{-1}$. The simulated time for the whole simulation is given by $T = 10\lambda_p/c$, which results in a number of time steps of about 2260.

At the start of the simulation, 100 random positions are computed in the cuboid given by $0 \le x < X$, $0 \le y < Y$ and $1.5\lambda_p \le z \le 2\lambda_p$. These positions (x_i, y_i, z_i) are used as head positions for 100 rods of length $\lambda_p/2$. The rods are constructed as strings of quasi-particles with a distance of Δz between each adjacent pair. The strings start at $(x_i, y_i, z_i - 0.5\Delta z)$ and continue with decreasing z-coordinates. These quasi-particles have the same shape as the plasma quasi-particles, a relativistic factor of $\gamma = 10^{10}$, and a charge that varies according to the cosine-like distribution (2). The total charge of a single rod (the sum of all charges in the constituting string of quasi-particles) is $|Q| \approx 1.8 \times 10^{-3} e n_0 k_p^{-3}$, where the sign of the charge is chosen randomly. This charge is sufficiently high for a clear observation of the wakefield and sufficiently low to stay within the regime of a linear plasma response.

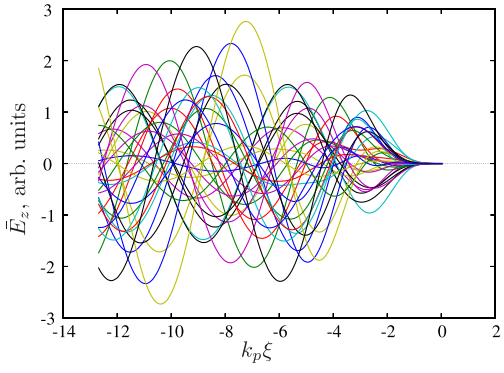

FIG. 6. The longitudinal wakefield $\bar{E}_z(\xi)$ at $x = y = 0.15\lambda_p$ generated by 100 cosine rods for 32 different random seeds.

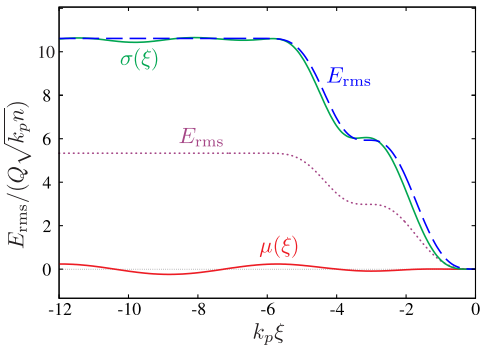

FIG. 7. The mean $\mu(\xi)$ and the standard deviation $\sigma(\xi)$ of the simulated wakefield (using 11 264 rod samples with 100 rods each) and the calculated average $E_{\rm rms}(\xi)$ with (dashed line) and without (dotted line) taking into account correlation effects due to the periodical boundary conditions.

To enhance the wakefield noise against other noise harmonics (which are always present in PIC simulations), we average the fields over many time steps according to the formula

$$\bar{E}_z(x, y, \xi) = \frac{1}{1000} \sum_{t=1201\Delta t}^{2200\Delta t} E_z(x, y, \xi, t). \tag{12}$$

This averaging strongly suppresses all perturbations except those propagating with the speed of light and being stationary in the co-moving frame.

The simulation is then run for a large number of samples n_s with different random number generator seeds, which leads to different rod positions. Examples of the produced field distributions are shown in Fig. 5. Since the frequency ω_p and the phase velocity c of the perturbation are fixed, the wavelength λ_p is fixed as well, which makes the wakefield noise periodic in ξ. It therefore does not look like usual noise. The transverse distance for field correlations is about k_p^{-1}, which makes random field changes in transverse directions only visible in the case of wider simulation areas. The amplitude and the phase of the generated fields are quite random even for narrow simulation areas (Fig. 6), and their properties can be characterized by the mean value $\mu(\xi)$ and the standard deviation $\sigma(\xi)$

$$\mu(\xi) = \frac{1}{n_s} \sum_{i=1}^{n_s} \bar{E}_z^i(x_m, y_m, \xi), \tag{13}$$

$$\sigma(\xi) = \sqrt{\frac{1}{n_s - 1} \sum_{i=1}^{n_s} \left(\bar{E}_z^i(x_m, y_m, \xi) - \mu(\xi)\right)^2}, \tag{14}$$

where the superscript i denotes the sample number, and x_m and y_m are the coordinates of the observation line. For the following, we put $x_m = y_m = 0.15\lambda_p$.

The resulting values for $\mu(\xi)$ and $\sigma(\xi)$ are shown in Fig. 7. The dotted line in Fig. 7 gives the value of $E_{\rm rms}$

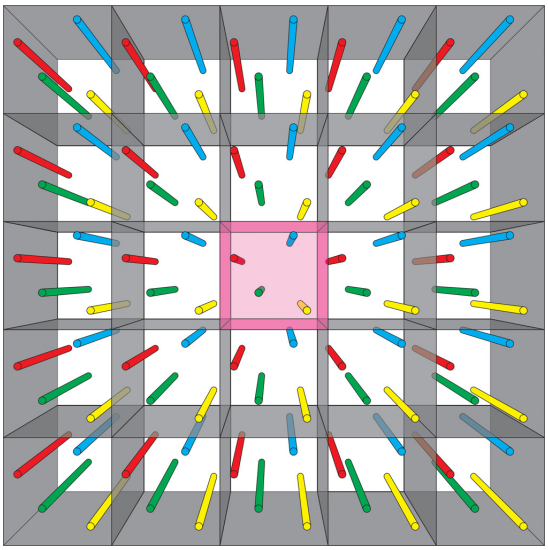

FIG. 8. Illustration of the effects of periodic boundaries. Physical processes in the rectangular area (shown in pink) simulated with periodic boundaries are equivalent to processes with a domain where the entire, infinite space is composed of identical replications of the pink area (replicas shown by the boxes with grey walls). Rods of the same color are identical in charge and longitudinal position and are located strictly periodically in transverse directions.

calculated for these rods according to the formula (10). It is substantially lower than the established average field in simulations. The difference results from the periodical boundary conditions. Each rod in the simulation domain has an infinite number of replicas, the positions of which are correlated with the rod position (Fig. 8). Consequently, for correct comparison of the theory and the simulation results, the radial averaging in (9) has to be modified

$$\int_0^R 2\pi r K_0^2(k_p r)dr \;\rightarrow\; \int_{-X/2}^{X/2}\int_{-Y/2}^{Y/2} dx\,dy \sum_{i,j} K_0^2(k_p r_{ij}), \quad (15)$$

where

$$r_{ij} = |\vec{r}_\perp + iX\vec{e}_x + jY\vec{e}_y|, \quad (16)$$

$\vec{e}_x$ and $\vec{e}_y$ are unit vectors, and indices $i,j \in \mathbb{Z}$, where $i = j = 0$ corresponds to the position of one specific rod (situated in the pink area in Fig. 8) and all other values correspond to the positions of its replicas (situated in gray boxes in Fig. 8). The dashed line in Fig. 7 shows the modified values for the average field.

V. SUMMARY

We have shown a reliable method of producing controllable noise levels in PIC simulations. This controllable wakefield noise does not suffer from numerical Cherenkov radiation. Analytical expressions for the rms amplitude of this noise have been given. These expressions include formulas for two different external boundaries, namely, for periodic boundaries as well as for open or absorbing boundaries. These expressions and the method in general were illustrated using very common PIC simulation techniques, making them very comparable and applicable for the research community. A very good agreement has been found. This method can be used to perform noise level scans for a multitude of noise seeded physical processes. This paper also provides an understanding of the to-be-expected noise structure. Using these noise level scans, the correct behavior can be predicted by extrapolation of the generated data points. Further research should focus on the application of this novel method on specific beam-plasma nonlinearities.

ACKNOWLEDGMENTS

Contribution of R.S. and K.L. to this work is supported by The Russian Science Foundation, Grant No. 14-50-00080. N.M. and H.R. acknowledge the hospitality of the Arnold Sommerfeld Center for Theoretical Physics at the Ludwig Maximilians University. N.M. acknowledges the very useful advice of K. U. Bamberg during the development of the computational simulation. This work was supported by the Cluster-of-Excellence Munich Centre for Advanced Photonics (MAP) and the Gauss Centre for Supercomputing (GCS), project PLASMA SIMULATION CODE, LRZ-ID pr84me.

[1]A. Bret, L. Gremillet, and M. E. Dieckmann, Phys. Plasmas **17**, 120501 (2010).
[2]M. H. Key, Phys. Plasmas **14**, 055502 (2007).
[3]M. Tabak, D. S. Clark, S. P. Hatchett, M. H. Key, B. F. Lasinski, R. A. Snavely, S. C. Wilks, R. P. J. Town, R. Stephens, E. M. Campbell, R. Kodama, K. Mima, K. A. Tanaka, S. Atzeni, and R. Freeman, Phys. Plasmas **12**, 057305 (2005).
[4]A. Burdakov, A. Arzhannikov, V. Astrelin, V. Batkin, V. Burmasov, G. Derevyankin, V. Ivanenko, I. Ivanov, M. Ivantsivskiy, I. Kandaurov, V. Konyukhov, K. Kuklin, S. Kuznetsov, A. Makarov, M. Makarov, K. Mekler, S. Polosatkin, S. Popov, V. Postupaev, A. Rovenskikh, A. Shoshin, S. Sinitsky, V. Stepanov, Y. Sulyaev, Y. Trunev, L. Vyacheslavov, and Ed. Zubairov, Fusion Sci. Technol. **55**(2T), 63 (2009).
[5]I. V. Timofeev and K. V. Lotov, Phys. Plasmas **13**, 062312 (2006).
[6]I. V. Timofeev and A. V. Terekhov, Phys. Plasmas **17**, 083111 (2010).
[7]F. Zimmermann, Phys. Rev. Spec. Top. Accel. Beams **7**, 124801 (2004).
[8]N. Kumar, A. Pukhov, and K. Lotov, Phys. Rev. Lett. **104**, 255003 (2010).
[9]K. V. Lotov, Phys. Plasmas **22**, 103110 (2015).
[10]K. V. Lotov, Phys. Plasmas **22**, 123107 (2015).
[11]K. V. Lotov, G. Z. Lotova, V. I. Lotov, A. Upadhyay, T. Tuckmantel, A. Pukhov, and A. Caldwell, Phys. Rev. Spec. Top. Accel. Beams **16**, 041301 (2013).
[12]B. N. Godfrey, J. Comput. Phys. **15**, 504 (1974).
[13]R. Lehe, A. Lifschitz, C. Thaury, V. Malka, and X. Davoine, Phys. Rev. Spec. Top. Accel. Beams **16**, 021301 (2013).
[14]R. Nuter, M. Grech, P. Gonzalez de Alaiza Martinez, G. Bonnaud, and E. d'Humieres, Eur. Phys. J. D **68**, 177 (2014).
[15]J.-L. Vay, R. Lehe, H. Vincenti, B. B. Godfrey, I. Haber, and P. Lee, Nucl. Instrum. Methods A **829**, 353 (2016).
[16]K. Germaschewski, W. Fox, S. Abbott, N. Ahmadi, K. Maynard, L. Wang, H. Ruhl, and A. Bhattacharjee, J. Comput. Phys. **318**, 305 (2016).
[17]T. Katsouleas, S. Wilks, P. Chen, J. M. Dawson, and J. J. Su, Part. Accel. **22**, 81 (1987).
[18]J. P. Boris, in *Proceedings of the 4th Conference on Numerical Simulation of Plasmas* (Naval Research Laboratory, Washington, D.C., 1970).
[19]K. Yee, IEEE Trans. Antennas Propag. **14**(3), 302 (1966).

5 First fully kinetic three-dimensional simulation of the AWAKE baseline scenario

Chapter 4.3 introduced a road-map for the study of the relation between the quasi-particle noise and the lateral beam filamentation. Continuing this previously established road-map, this chapter presents the results of a large-scale simulation of the AWAKE baseline case. This simulation yielded a peer-reviewed publication as well. This chapter and chapter 4 aim at establishing clearly, which share of the overall work for these publications has been achieved by Nils Moschüring. Therefore, this chapter is written in the first person as well.

5.1 Improved and updated baseline test case

While investigating the controlled noise, I continuously improved the capabilities of the baseline test case as well. This test case is needed for the final verification of the predicted filamentation threshold. The capabilities were improved to accommodate for the new baseline case ([20], Table 1), which includes oblique witness beam injection at a shallow angle and a plasma density ramp.

In-line data analysis is data post-processing that is performed during the simulation, instead of after the simulation has finished. Therefore, it is able to use the full computational power and parallelism of the used supercomputer. It may also make writing intermediary data to disk unnecessary, saving a lot of run-time.

By implementing a host of new in-line data analysis, the performance was improved and a lot of hard disk space and I/O bandwidth was saved. The following values were calculated in-line and $\forall x = j\Delta x, \forall y = k\Delta y, \forall z = l\Delta z, \forall t = n \cdot 1000\Delta t, \forall s \in \{\text{beam}, \text{plasma}, \text{witness}\}$:

$$E_{z,\max}(z) = \max_t \max_{r<r_\mathrm{p}} E_z(r,z,t) \quad (5.1) \qquad n_s(p_z,t) \quad (5.2)$$

$$E_{z,\text{beam}}(z,t) = E_z(x_\mathrm{b},y_\mathrm{b},z,t) \quad (5.3) \qquad n_s(p_\perp,t) \quad (5.4)$$

$$E_{z,\text{avg}}(x,z,t) = \sum_{\hat{t}=t-1000}^{t} E_z\left(x,y_\mathrm{b},z,\hat{t}\right) \quad (5.5) \qquad n_s(\gamma,t) \quad (5.6)$$

$$\epsilon_{s,x}(z,t) = \sqrt{\langle x_s^2\rangle\langle p_{s,x}^2\rangle - \langle x_s p_{s,x}\rangle^2} \quad (5.7) \qquad X_{s,0}(z,t) = \langle x_s\rangle \quad (5.8)$$

$$\epsilon_{s,y}(z,t) = \sqrt{\langle y_s^2\rangle\langle p_{s,y}^2\rangle - \langle y_s p_{s,y}\rangle^2} \quad (5.9) \qquad Y_{s,0}(z,t) = \langle y_s\rangle \quad (5.10)$$

$$n_s(z,t) \quad (5.11) \qquad X_s(z,t) = \sqrt{\langle x_s^2\rangle - X_{s,0}^2} \quad (5.12)$$

$$n_{s,\text{wf}}(z,t) = \sum_{r<\frac{c}{\omega_\mathrm{p}}} n_s(r,z,t) \quad (5.13) \qquad Y_s(z,t) = \sqrt{\langle y_s^2\rangle - Y_{s,0}^2} \quad (5.14)$$

Here, E_z is the z-component of the electric field, x_b and y_b are the initial location of the maximum density of the ion beam in the x- and y-direction, r_p is the plasma radius, x_s is the x-coordinate

of a quasi-particle of species s, y_s is the y-coordinate of a quasi-particle of species s, $p_{x,s}$ is the x-coordinate of the momentum of a quasi-particle of species s, $p_{y,s}$ is the y-coordinate of the momentum of a quasi-particle of species s, $\langle\cdot\rangle$ averages its respective argument over all quasi-particles in all cells with a specific value range $[z - \Delta z/2, z + \Delta z/2]$ in the z-direction, $r = \sqrt{(x - x_\mathrm{b})^2 + (y - y_\mathrm{b})^2}$, n_s is the density of quasi-particles of species s (summed over all dimensions not present in its arguments) and is the result of binning all quasi-particles into certain slices of configuration space, for spatial dimensions $n_s = \sum_{i=1}^{N} w_{i,s}\zeta_{jkl}(\vec{x}_{i,s})$ with ζ from (2.53) and N the total number of quasi-particles, while for momentum dimensions 0-th order shape functions (δ functions) and 16384 bins are used, $n_{s,\mathrm{wf}}$ is the same quantity restricted to the core wakefield, $\gamma = 1/\sqrt{1 - v^2/c^2}$, c is the speed of light and ω_p is the plasma frequency.

These values were saved to disk immediately, which enabled a very close observation of the simulation development as well as the possibility to immediately correct any emerging problems. This is especially important for the large-scale simulation, as it can not be repeated due to its high cost.

The code was also configured to provide full dumps of the complete simulation data at 11 different points in time. These points in time were chosen in order to provide diagnostic information of several important milestones in the developing simulation (t_sim is the total run-time of the simulation):

$2\,\% \cdot t_\mathrm{sim}$	Initial configuration
$4\,\% \cdot t_\mathrm{sim}$	Beam entry into plasma
$7\,\% \cdot t_\mathrm{sim}$	Seed wakefield settled
$15\,\% \cdot t_\mathrm{sim}$	Merging of the witness electrons
$19\,\% \cdot t_\mathrm{sim}$	Trapping finished
$31\,\% \cdot t_\mathrm{sim}$	Wave growth
$41\,\% \cdot t_\mathrm{sim}$	Wakefield maximum reached
$50\,\% \cdot t_\mathrm{sim}$	Established wave phase
$93\,\% \cdot t_\mathrm{sim}$	Just before any exit effects
$96\,\% \cdot t_\mathrm{sim}$	Beam exit from the plasma
$100\,\% \cdot t_\mathrm{sim}$	Final configuration

I added individual particle tracking to the code as well. It was configured to track 50 individual quasi-particles of each species during the whole simulation and to write their complete configuration to disk every 20 timesteps. This output was buffered in order to get rid of the disc I/O latency. The resulting trajectories were very helpful in analyzing the processes happening during the simulation. The data is also the source of a major part of the final figures in the paper as well as the videos.

Very critical performance benchmarks are the scaling properties of the code. The proposed simulation would need to scale up to at least 4 islands (32768 cores), but it would benefit from scaling to even more cores, up to the maximum of 18 islands on SuperMUC phase 1.

In this regard, many modules of the PSC were optimized. The load balancer was improved and specialized for the AWAKE test case. Furthermore, the I/O system of the PSC was specialized, in cooperation with Karl-Ulrich Bamberg, for the SuperMUC GPRS file system. Very importantly, a rigorous scalability analysis was performed. Any data structure that did not scale adequately needed to be identified and possibly removed or improved. Several critical code paths were corrected during this effort.

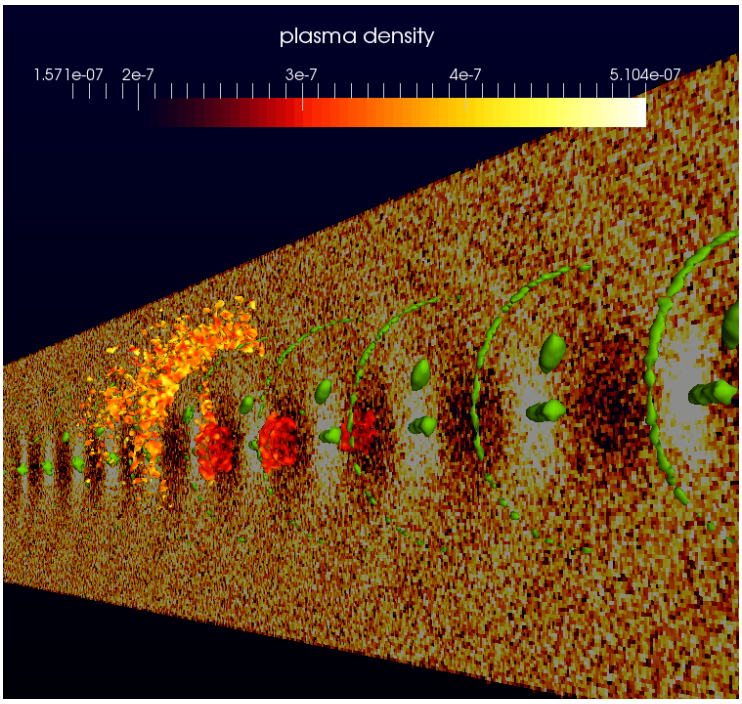

Fig. 5.1 – Simulation state after 5m of beam propagation inside the plasma. The three-dimensional green structures are the ion beam micro-bunches with density $1.9 \cdot 10^{12}\text{cm}^{-3}$. The three-dimensional red and orange structures are parts of the trapped electron witness beam at density $1.9 \cdot 10^{10}\text{cm}^{-3}$. The color legend gives the density in units of $1.9 \cdot 10^{21}\text{cm}^{-3}$ of the 2D plane.

5.2 Preliminary runs and final large-scale simulation

New results, using these new capabilities but with reduced resolutions, were presented by me at two different conferences, the AWAKE collaboration meeting in March 2016 in Lisbon, Portugal and the Conference on Computational Physics in July 2016 in Johannesburg, South Africa. Fig. 5.1 and Fig. 5.2 show some of these results, configured to use 65 points per λ_p, which is half of the necessary final spatial resolution. These reduced resolution runs were needed in order to gain experience with the simulation and to find potential pitfalls and problems. They also served as a last check before committing to the full run, which would use up a lot of the granted budget, therefore essentially being a point of no return.

After finalizing these new simulation capabilities, making them in tune with the experimental work, a run using the full resolution was performed. Making this large-scale run took a lot of extra work. The reduced stability of cutting-edge hardware, like SuperMUC phase 1, for runs using a large part of their total computational capabilities, made adaptations as well as several restarts necessary. The simulation also ran into trouble with the implementation of MPI used on the system, which only materialized when scaling to the total amount of resources. This required additional modifications to different algorithms. Luckily, Karl-Ulrich Bamberg had a lot

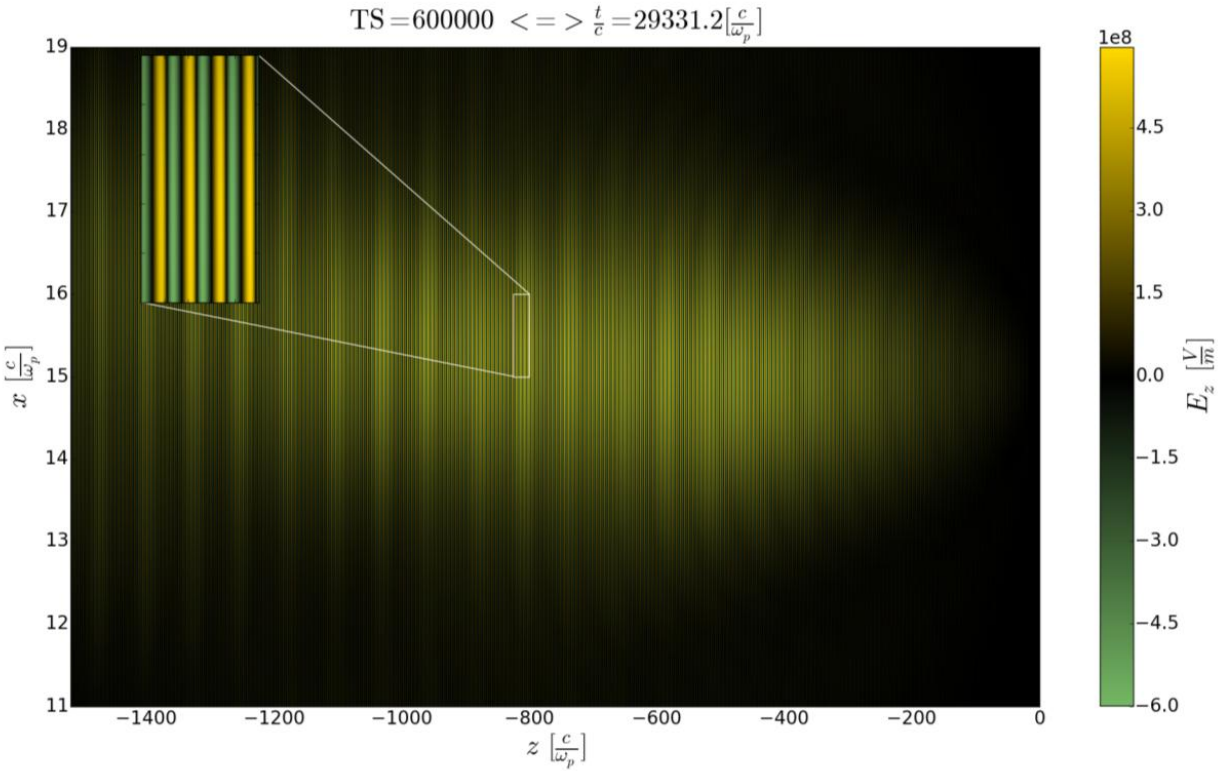

Fig. 5.2 – Maximum E_z field amplitude averaged over 1000 time steps at each x-z-position of simulation space after 5m of beam propagation inside the plasma.

of valuable experience with the SuperMUC system through his work on a KONWIHR (Kompetenznetzwerk für Technisch-Wissenschaftliches Hoch- und Höchstleistungsrechnen in Bayern) project. The run was finally finished after a total run-time of about 28 days. The results of this run were analyzed and showed some very interesting properties and novel effects, especially during the electron injection process. This process is vital for the success of AWAKE and needs to be well understood in order to achieve the proposed goals.

I presented these findings at the AWAKE Collaboration Meeting at CERN in September 2018 and at the Laser-Plasma Accelerator Workshop 2019 in May 2019 in Split, Croatia. Because of the very relevant and interesting results it was decided to publish the results of this run. The tremendous preparatory work and final execution was rewarded with another peer-reviewed paper [20]. This paper is included in this dissertation in chapter 5.3.

While the bulk of the work has been done by Konstantin Lotov and me, the other authors are included for their vital effort in helping to achieve the large-scale simulation. These contributions mainly include code optimization, fruitful discussions and organizing the large-scale runs and computational resources. Karl-Ulrich Bamberg has been especially helpful in running the code and managing the access to SuperMUC, as well as in the data management and data backup. All simulation results have been produced using my test case, which has about 4000 lines of C code, and have been post-processed by me. The abstract has been written by me.

Chapter 1 and chapter 2 have been written by Konstantin Lotov. Chapter 3 has been written by me. Chapter 4 and chapter 5 have been written by Konstantin Lotov. The summary is a joint effort of all authors. Fig. 2, Fig. 3 and Fig. 4 and their respective captions have been created and written by me. Fig. 5 and Fig. 6 have been created by Konstantin Lotov, using my simulation results and post-processing. I have written about 3000 lines of python code to perform the post-processing and plotting. Additionally, multimedia files, available on the IOP website of the paper under the heading *Supplementary data*, called `trapping_and_acceleration.mp4` and `dynamic_domain.mp4`, were created by me and show graphs that are very similar to Fig. 5 and Fig. 6. These videos very nicely illustrate the physical processes and enable a better understanding of the newfound phenomena. Furthermore, I heavily contributed in the editing of the whole paper. I was also the corresponding author and processed most of the questions of the referees. In total, I submitted 15 pages, containing the questions and answers as well as a list of changes to the paper to the referees. These 15 pages were mostly written and compiled by me. Konstantin Lotov provided a small amount of answers totaling about one page.

5.3 Full-text of the publication

Reproduced from N Moschuering et al 2019 Plasma Phys. Control. Fusion 61 104004, https://doi.org/10.1088/1361-6587/ab411e, with the permission of IOP Publishing Limited.

Plasma Physics and Controlled Fusion

PAPER

First fully kinetic three-dimensional simulation of the AWAKE baseline scenario

To cite this article: N Moschuering *et al* 2019 *Plasma Phys. Control. Fusion* **61** 104004

IOP Publishing Plasma Physics and Controlled Fusion
Plasma Phys. Control. Fusion **61** (2019) 104004 (9pp) https://doi.org/10.1088/1361-6587/ab411e

First fully kinetic three-dimensional simulation of the AWAKE baseline scenario

N Moschuering[1,4], K V Lotov[2,3], K Bamberg[1], F Deutschmann[1] and H Ruhl[1]

[1] Ludwig-Maximilians-Universität, D-80539 Munich, Germany
[2] Budker Institute of Nuclear Physics, Novosibirsk, 630090, Russia
[3] Novosibirsk State University, Novosibirsk, 630090, Russia

E-mail: Nils.Moschuering@physik.uni-muenchen.de, K.V.Lotov@inp.nsk.su, Karl-Ulrich.Bamberg@physik.uni-muenchen.de, Fabian.Deutschmann@physik.uni-muenchen.de and Hartmut.Ruhl@physik.uni-muenchen.de

Received 17 May 2019, revised 7 August 2019
Accepted for publication 3 September 2019
Published 18 September 2019

Abstract

The 'Advanced Proton Driven Plasma Wakefield Acceleration Experiment' (AWAKE) aims to accelerate leptons via proton-beam-driven wakefield acceleration. It comprises extensive numerical studies as well as experiments at the CERN laboratory. The baseline scenario incorporates a plasma volume of approximately 62 cm^3. The plasma wavelength is about 1.25 mm and needs to be adequately resolved, using a minimum of 130 points per plasma wavelength, in order to accurately reproduce the physics. The baseline scenario incorporates the proton beam micro-bunching, the concurrent nonlinear wakefield growth as well as the off-axis electron beam injection, trapping and acceleration. We present results for the first three-dimensional simulation of this baseline scenario with a full model, using a sufficient resolution. The simulation consumed about 22 Mch of computer resources and scaled up to 32 768 cores, thanks to a multitude of adaptions, improvements and optimization of the simulation code PSC. Through this large-scale simulation effort we were able to verify the results of reduced-model simulations as well as identify important novel effects during the electron injection process.

Supplementary material for this article is available online

Keywords: plasma wakefield acceleration, proton driver, numerical simulations, particle-in-cell

(Some figures may appear in colour only in the online journal)

1. Introduction

Proton-beam-driven plasma wakefield acceleration promises to increase the lepton energy which is available to scientific laboratories [1–3]. The first step towards future colliders based on this principle is the AWAKE experiment at CERN [4–6], which has already demonstrated controlled self-modulation of the proton beam [7, 8] (the key element of the concept) as well as electron acceleration in the proton-driven wakefield [9].

In order to better understand the physics of proton and electron beam interaction with plasma, a massive simulation campaign accompanied the experiment from its very early stages [1, 4, 10–25]. However, because of the large beam and plasma sizes, large when compared to the scale which needs to be resolved [26], simulations were limited to reduced models. These models include the quasi-static approximation [27] or the axially symmetric two-dimensional (2D) geometry. There is an obvious need of three-dimensional (3D) simulations based on first principles to check the validity of the reduced models and to search for new effects which might have been missed in these reduced models. In this paper we report the first fully kinetic 3D simulation of the baseline AWAKE variant. We show that 2D quasi-static models adequately describe the proton beam dynamics, while the electron injection needs a 3D treatment and is rich in novel effects.

[4] Author to whom any correspondence should be addressed.

Plasma Phys. Control. Fusion **61** (2019) 104004 N Moschuering *et al*

Table 1. Parameters of the simulated variant. Numbers typed in italic are approximate values.

Parameter and notation	Value	Dimensionless value (unit)
Reference plasma density, n_0	7×10^{14} cm^{-3}	1 (n_0)
Plasma skin depth, c/ω_p	0.2 mm	1 (c/ω_p)
Plasma length, L	10 m	50 000 (c/ω_p)
Transition area half-length, $L_{\rm tr}$	40 cm	2000 (c/ω_p)
Plasma radius, r_p	1.4 mm	7 (c/ω_p)
Orifice diameter, D	10 mm	50 (c/ω_p)
Plasma ion-to-electron mass ratio, M_i	157 000	157 000
Wavebreaking field, $E_0 = mc\omega_p/e$	*2.54* GV m^{-1}	1 (E_0)
Plasma initial temperature, T_e	0.1 eV	2×10^{-7} (mc^2)
Uncut driver population, N_b	3×10^{11}	3×10^{11}
Driver length, σ_{zb}	12 cm	600 (c/ω_p)
Driver radius, σ_{rb}	0.2 mm	1 (c/ω_p)
Driver energy, W_b	400 GeV	*783 000* (mc^2)
Driver energy spread, δW_b	0.35%	*2740* (mc^2)
Driver angular spread, $\delta\alpha_b$	4.5×10^{-5}	4.5×10^{-5}
Maximum driver density, n_{b0}	4×10^{12} cm^{-3}	*0.0056* (n_0)
Electron beam population, N_e	1.25×10^{9}	1.25×10^{9}
Electron beam length, σ_{ze}	1.2 mm	6 (c/ω_p)
Electron beam radius, σ_{re}	0.25 mm	1.25 (c/ω_p)
Electron beam energy, W_e	16 MeV	32 (mc^2)
Electron beam energy spread, δW_e	0	0
Electron beam angular spread, $\delta\alpha_e$	0	0
Maximum electron beam density, n_{e0}	1×10^{12} cm^{-3}	*0.0015* (n_0)
Injection angle for electron beam, α_0	0.0028	0.0028
Injection delay, ξ_e	11.5 cm	575 (c/ω_p)
Intersection of beam trajectories, z_0	*142* cm	*7100* (c/ω_p)

We describe the simulated AWAKE scenario in section 2 and the simulation details in section 3. Then we discuss the self-modulation of the proton beam in section 4 and the trapping of externally injected witness electrons in section 5. In section 6, we summarize the main findings.

2. AWAKE baseline scenario

The simulated scenario is close to the scenario used in [4, 5, 24], except the electron beam has no initial angular spread. It includes a sharp plasma boundary in the radial direction [28], realistic longitudinal density transitions at the plasma cell orifices [29], a positive density gradient along the plasma cell [21], and oblique electron injection at a shallow angle [24]. The initial angular spread (or emittance) of the electron beam can be neglected and taken as zero, if the spread gained during injection to the wave is much larger than the initial value. The complete set of physical parameters is listed in table 1. Here m is the electron mass, $e > 0$ is the elementary charge, c is the speed of light, and ω_p is the electron plasma frequency.

We either use cylindrical coordinates (r, φ, z) or Cartesian coordinates (x, y, z) with the z-axis being the direction of the beam propagation. We additionally consider the co-moving coordinate $\xi = z - ct$. The longitudinal plasma density profile $n(z)$ is

$$\begin{aligned}
|z| \leqslant L_{\rm tr}:\ & \frac{n_0}{2}\left(1+\frac{z/D}{\sqrt{(z/D)^2+0.25}}\right),\\
L_{\rm tr} < z < L - L_{\rm tr}:\ & n_0\left(1+0.01\frac{z-L_{\rm tr}}{L-2L_{\rm tr}}\right),\\
|z-L| \leqslant L_{\rm tr}:\ & \frac{1.01n_0}{2}\left(1-\frac{(z-L)/D}{\sqrt{\left(\frac{z-L}{D}\right)^2+0.25}}\right),
\end{aligned} \tag{1}$$

and zero otherwise. The orifice diameter D determines the density profile near the entrance to and exit from the plasma cell (located at $z = 0$ and $z = L$). The plasma is radially uniform within the radius r_p and composed of single ionized rubidium ions.

Before entering the plasma, the beam density is

$$\begin{aligned}
n_b(r, z, t) = 0.5\, n_{b0}\, e^{-r^2/2\sigma_{rb}^2}\left[1+\cos\left(\sqrt{\frac{\pi}{2}}\frac{\xi}{\sigma_{zb}}\right)\right],\\
-\sigma_{zb}\sqrt{2\pi} < \xi < 0,
\end{aligned} \tag{2}$$

and zero otherwise. The tail of the driving beam at the simulation start is positioned at $z = -L_{\rm tr}$ (figure 1). The cut density profile (2) mimics plasma creation through rapid ionization of a neutral gas by a co-propagating short and

Table 2. Simulation parameters. Numbers typed in italic are approximate values.

Parameter and notation	Value	Dimensionless value (unit)
Minimum number of grid points per $\lambda_p = 2\pi c/\omega_p$		130
Timestep size	1.6×10^{-14} s	0.025 $(1/\omega_p)$
Spatial grid size, Δx	*9.40* μm	0.047 (c/ω_p)
Spatial grid size, Δy	*9.38* μm	0.047 (c/ω_p)
Spatial grid size, Δz	*9.59* μm	0.048 (c/ω_p)
Initial box size in x-direction	0.8124 cm	40.62 (c/ω_p)
Reduced box size in x-direction	0.42 cm	21 (c/ω_p)
Box size in y-direction	0.42 cm	21 (c/ω_p)
Full box size in z-direction	11.1008 m	55 504 (c/ω_p)
Galilean window size in z-direction, L_w	30.38 cm	1519 (c/ω_p)
Initial grid size		$864 \times 448 \times 1157120 = 5 \times 10^{11}$
Reduced grid size (Galilean, beams merged)		$448 \times 448 \times 32400 = 6 \times 10^{9}$
Initial number of patches		$54 \times 28 \times 72\,320$
Reduced number of patches (Galilean, beams merged)		$28 \times 28 \times 2025$
Number of timesteps		2198 134
Number of quasi-particles per cell for each species		3
Courant–Friedrichs–Lewy number [33]		0.9
Real particles per quasi-particle with weight 1, N_{real}		*7.35* $\times 10^7$
Plasma peak real particles per quasi-particle, $N_{\text{real}} n_0/n_0$		*7.35* $\times 10^7$
Driver peak real particles per quasi-particle, $N_{\text{real}} n_{b0}/n_0$		*4.13* $\times 10^7$
Electron peak real particles per quasi-particle, $N_{\text{real}} n_{e0}/n_0$		*1.10* $\times 10^7$
Average number of quasi-particles in box		*7.4* $\times 10^7$

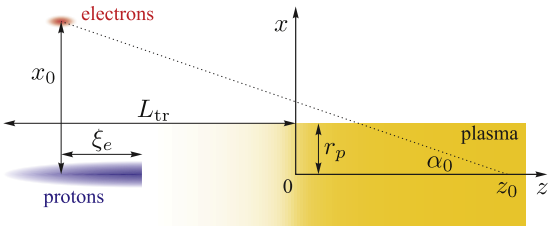

Figure 1. Scheme of beam injection into the plasma.

intense laser pulse. The electron beam has the density profile

$$n_e(x, y, z, t) = 0.5\, n_{e0} \exp\left(-\frac{(x - x_0)^2 + y^2}{2\sigma_{re}^2}\right) \times \left[1 + \cos\left(\sqrt{\frac{\pi}{2}}\frac{(\xi + \xi_e)}{\sigma_{ze}}\right)\right], \quad |\xi + \xi_e| < \sigma_{ze}\sqrt{2\pi}, \quad (3)$$

and zero otherwise. Here

$$x_0 = (z_0 - z)\tan\alpha_0 \quad (4)$$

is the transverse location of the electron beam center. All electrons have the same initial momentum $(-\alpha_0 p_0,\ 0,\ p_0)$ with $p_0 = W_e/c$.

3. Simulations

The simulation is realized using the fully relativistic 3D particle-in-cell (PIC) code PSC [30], utilizing a standard Boris particle pusher [31] with second-order quasi-particles, and a finite-difference-time-domain field solver [32]. A collection of numerical parameters of the simulation is shown in table 2. This full 3D fully kinetic simulation of the AWAKE baseline requires substantial computational resources. We acquired the considerable amount of 25 Mch at the SuperMUC Phase 1 supercomputer in Munich through the Gauss Center for Supercomputing.

The PSC offers a powerful mechanism, which provides several key capabilities for this simulation. This mechanism subdivides the simulation box into several smaller groups of grid points, called patches. These patches can then be freely distributed over the available resources and can also be activated or deactivated. Taking full advantage of this dynamic patch system enables three very important ways to save on computational resources. First, it makes the usage of a moving Galilean window possible (figure 2). This window moves at the same speed as the proton beam. Through this, we reduce the active simulation domain to $1.01\sqrt{2\pi}\,\sigma_{zb} \approx 31$ cm in the z-direction. Second, we shrink the simulation domain by ≈48% after ≈17% of the simulation by switching off the space occupied by the witness beam after its entry into the plasma cylinder (figure 2). Third, we meaningfully reduce load imbalance by continuously rearranging the patches on the participating resources. Patches with less quasi-particles are grouped and assigned to cores in a way which aims at achieving an equal distribution of quasi-particles to cores. This is done while maintaining a consecutive spatial order of the patches along a space-filling curve, which is necessary to keep the communication costs low.

The boundary conditions in the lower and upper z-direction are fairly straightforward as they do not affect the physics at all. The reason for this is that no field or quasi-particle can reach the upper boundary as the boundary travels with the same velocity as the particle beam. The lower boundary is continuously moving as well, which gets rid of

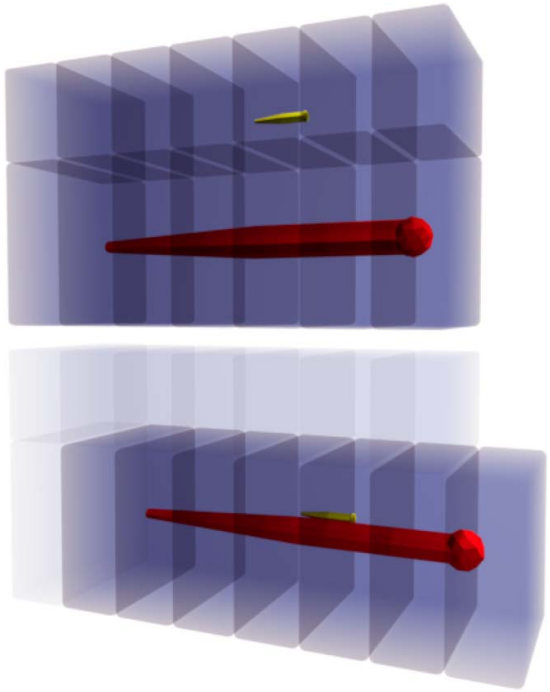

Figure 2. The upper figure shows the initial simulation setup containing the ion beam in red and the witness beam in yellow, offset in the x-direction. The blue-gray boxes represent the patches. The real simulation uses many more patches in all three directions as given in table 2. The lower figure shows the configuration of the simulation after the witness and ion beam have merged. The upper patches have been disabled. A patch in front of the beam has been added, while the patches in the back of the beam have been disabled. The PSC continuously generates and removes patches in this manner in order to realize the Galilean window. Disabled patches are represented by having a higher transparency.

any spurious fields and particles before they can return and interact with the simulation. This makes it reasonable to choose a very simple and predictable reflecting boundary in the positive and negative z-direction. Unfortunately, the transversal direction proves to be very tricky. After trying out different implementations of universal perfectly-matched layers [34, 35] and conducting boundaries we come to the conclusion that it is too risky to run the full simulation with these boundaries. The boundary cells exhibit a certain degree of nonlinear numerical field growth in all tested cases which make the results unreliable. This may be remedied in future simulations by improvements to the patch mechanism in conjunction with the boundary algorithms. Spurious currents and the continuously charging wall make it very questionable to just delete exiting particles. Because of this, we implement periodic boundaries in the transversal direction for particles and fields. Periodic boundaries are very easy to predict and are very unlikely to behave in an unexpected manner. Since the plasma provides very strong electromagnetic shielding and the Galilean window guarantees a maximum number of box traversals for electromagnetic waves, there is no effect to the particle dynamics by recurring waves. Unfortunately, recurring particles are a problem in the case of the electron beam. Because of the periodic boundary conditions in the transversal direction, witness beam particles can only leave the simulation box via deceleration. This enables the trapping of witness particles which had already left the plasma after interacting with the beam, particles which, in the experiment, would be lost. Because of this effect, we can not make proper observations of the captured charge. An easy solution to this problem would have been to flag each electron quasi-particle which traverses the periodic boundaries. This would have made it possible to identify and filter these particles during the post-processing step of the simulation. Unfortunately, this was not implemented for the present simulation, and the simulation could not be repeated due to its high cost.

The simulation uses three different particle species: plasma electrons, ion beam particles and electron beam particles. Since ion motion was shown to be negligible for these beam parameters [4, 19], the rubidium ions are assumed to be stationary and are not simulated. The charge-conserving particle pushing algorithm [36] of the PSC assumes a balanced charge distribution at the start of the simulation. Not generating any quasi-particles for the rubidium ions therefore acts in the same way as having fixed positive charges at the initial positions of the plasma electrons. These fixed charges consume no computing resources. Each kind uses three quasi-particles per cell. The weight of a quasi-particle is given by n/n_0, n_b/n_0 and n_e/n_0, respectively, where the density is taken at the center of the cell where the quasi-particle is created. The number of real particles per quasi-particle with a weight of 1 is given by $N_{\text{real}} = n_0/3 \times \Delta x \times \Delta y \times \Delta z$. This number is valid for quasi-particles of all species. It is also the peak number of real particles per plasma quasi-particle, since in this case, $n/n_0 = 1$, which makes plasma quasi-particles at locations with a peak plasma density have a weight of 1. Each cell with a nonzero density value for a specific kind contains three particles of that kind. We thereby achieve a good resolution of the particle dynamics even in low density areas of the simulation. These formulas generate quasi-particles with weights which differ by orders of magnitude. Since we do not use any Monte-Carlo collisions, this does not affect the quality of our results. On average $\approx$16.3% of the quasi-particles are driver particles, $\approx$0.3% are electron beam particles and $\approx$83.4% are plasma particles. This makes the number of quasi-particles per cell, which are used to represent the plasma, a dominating factor in determining the cost of the simulation. We choose to use 3 quasi-particles per cell for the plasma in order to keep the simulation cost reasonable. Since the plasma exhibits only linear dynamics, this number suffices to accurately represent the physics. By using 3 quasi-particles for the beam and electron species as well, our simulation has a two orders of magnitude smaller amount of real particles per quasi-particle for these species, and therefore supplies a sufficient resolution for their nonlinear dynamics. The non-peak value of the number of particles per quasi-particle is even lower, which gives the slopes of the Gaussian-distributed beam and electron densities, and

therefore the majority of the volume, an even higher resolution. The plasma quasi-particles have an additional restriction: only cells with a plasma density $>0.01n_0$ receive three quasi-particles, other cells receive zero quasi-particles. This sets a sensible cutoff point for the quasi-particle generation without meaningfully altering the physics. There is no overlap between the driver and the plasma quasi-particles at the start as the plasma density is below this threshold in the regions where driver particles are setup. The only purpose of the initial plasma temperature T_e is to generate distinguishable quasi-particles in each cell. Since all three quasi-particles are initialized at the same location inside their respective cell, they would not behave differently if they had equal momentum values. The value of T_e leads to an approximate difference in location of $L_w/c \times \sqrt{2T_e/m_e}/\Delta x \approx 20$ cells at the end of the lifetime of a plasma quasi-particle. Therefore, the quasi-particles are sufficiently different a short time after their initialization.

Using a simple model we can calculate a lower bound for the computational cost. For this model we assume that (a) a single core can push one million particles per second and that there is no (b) communication, (c) field solving, (d) data output cost and (e) load imbalance. With this model the computational cost of the simulation can be approximated to be $7.4 \times 10^9/(10^6/\mathrm{s}) \times 219\,813 \times 1\,\mathrm{c} = 4.5$ Mch. The total computational cost of the final simulation turns out to be about 22 Mch. From the cost of about 85 M € for building SuperMUC Phase 1, the total number of cores, which is $18\,432 \times 8$ (Intel Xeon E5-2680) $+ 820 \times 10$ (Intel Xeon E7-4870) $= 155\,656$, and a total runtime of approximately 6 years we can calculate a very conservative price per core-hour of about $85 \times 10^6/(155\,656 \times 6 \times 356 \times 24) \approx 0.01$ €. This lower bound fits well to scientific evaluations [37]. From this follows that the minimal total cost of this simulation amounts to ≈220 k€. This cost makes multiple simulations at the required resolution prohibitive.

It is extremely valuable to incorporate most of the data analysis into the simulation code itself. This enables the exploitation of the supercomputer resources for post-processing and reduces the amount of written data and successive disk input/output (I/O). Every 1000th timestep, the code provides outputs of a variety of smoothed field values with already performed additional processing like averaging and maximum finding. Furthermore it generates processed particle data like emittance, average locations and respective standard deviations as well as several different kinds of densities for different projections of the configuration and momentum space. This reduces the output data size immensely and makes it possible to continuously monitor the simulation. Additionally, ten specific timesteps are saved to disk in their entirety. These timesteps are chosen to portray important milestones in the evolving simulation, like the orifice entry, the merging of the witness beam, reaching the wakefield maximum, and exiting the plasma through the second orifice. Another very important data output is the full configuration of 50 particles of each kind every 20 timesteps.

We identify several key I/O parameters of the respective supercomputer and use these parameters for our optimization. These parameters may differ for different kinds of file systems and may therefore not easily transfer to other supercomputers. Two very important parameters of this kind are the number of active concurrent file-streams and the number of files per directory for which the I/O performance remains acceptable. For SuperMUC Phase 1's file system it is necessary to keep both of these numbers below 1000. It is also of paramount importance to only access each individual directory from a single node at a time. Using these findings, we carefully designed a parallel output algorithm which serializes the access to directories in an optimal way. We also tested output libraries, like MPI-IO, but found that due to the PIC data configuration, which consists of large chunks of unscattered data, these libraries did not perform well and did not improve performance meaningfully. Finally, we are able to achieve a data rate of 105 GByte s^{-1}. The theoretical machine maximum is given at 125 GByte s^{-1}. A full checkpoint, which is necessary to be written regularly because of the maximum job time length, had an initial size of 120 TByte and took 2.5 h to write. We reduced the data size to 12 TByte and are able to write it to disk in three minutes. This is achieved by exploiting additional features like in-memory compression and optimized data structures.

The second area of improvement is concerned with the scalability of the code. Running on 32 768 cores simultaneously, will uncover any non-scaling code structure. We got rid of a slew of unnecessary data duplication and output files. Due to the need for large amounts of memory as well as an optimized scaling behavior of the code, we find that 32 768 cores is the optimal amount of cores for our case.

4. Proton beam self-modulation

The plasma transforms the long proton beam into a train of short micro-bunches, which resonantly drive the plasma wave. A good quantitative measure of this process is the generated longitudinal electric field (figure 3). From comparing runs with different resolutions, we find that a fine grid is necessary in order to adequately represent the physical processes in the system. For a sufficiently small grid step size of $\lambda_p/130 \approx 0.05c/\omega_p$, the results of the 3D PSC simulation closely agree with the high resolution reference simulation of the 2D quasi-static code LCODE [27, 38], with a grid step size of $0.01c/\omega_p$.

Both high-resolution runs show the typical stages of wakefield evolution as the beam propagates through the plasma [39]. During the first meter in the plasma the wakefield stays at approximately the seed level. This is then followed by exponential growth, which changes into non-exponential growth after 2 m. The beam is fully micro-bunched at about 4 m (figure 4(a)) into the plasma and at this point it excites the strongest wakefield. After this, the wakefield decays because the micro-bunches do not fully reside in the focusing phase of the wave, and their defocused parts gradually erode [40] (figure 4(b)). The field spike near the exit appears, because the plasma wavelength slightly increases as the plasma density decreases. The micro-bunches then

Plasma Phys. Control. Fusion **61** (2019) 104004 N Moschuering *et al*

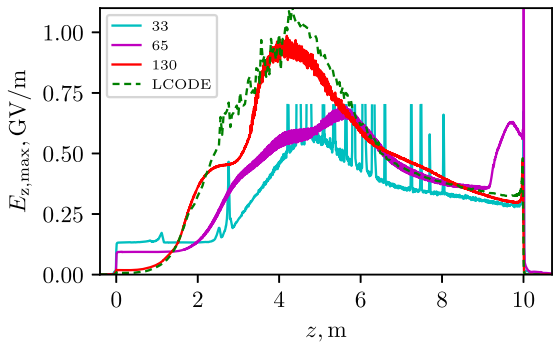

Figure 3. Comparison of the maximum accelerating gradient $E_{z,\max}(z) = \max_{x,y,t} E_z(x, y, z, t)$ (PSC) and $E_{z,\max}(z) = \max_{r,t} E_z(r, z, t)$ (LCODE) for different resolutions using the PSC and the quasi-static code LCODE. The numbers in the legend for the first three lines refer to the number of points per λ_p when using the PSC. The dashed line refers to the reference simulation with LCODE using ≈628 points per λ_p. In order to reduce clutter, resulting from a lot of noise, the 33 points per λ_p results have been capped at 0.7 GV m^{-1} maximum field value.

gradually shift into the stronger decelerating phase of the wave. Particles in this phase drive the wake more efficiently, which increases the field values. However, this occurs at the expense of a weaker to non-existing radial focusing, which is why the reduced plasma density can only boost the wakefield for a short distance.

We particularly emphasize that we did not find any sign of beam hosing (figure 4). Even the tail bunches, which are not perfectly symmetric, follow the head bunches perfectly. Theory [41, 42] and simulations of shorter beams [13, 42–44] predicted that the seeded self-modulation suppresses the hosing, but this is the first observation of no hosing occurring in full scale simulations of AWAKE. The seeding of the hosing instability in the simulation is even stronger than in reality, because the number of beam quasi-particles in the simulation is smaller than the number of protons in the real beam (table 2) [13]. Therefore, when considering a particle-noise-induced hosing instability, it has an even lower probability to develop in experiments.

5. Electron injection

As the electron bunch approaches and crosses the plasma boundary, it experiences a strong force which is caused by the redistribution of charges in the plasma (figure 5). The force is both focusing the bunch and attracting the bunch to the plasma boundary. The force action starts in the vacuum and increases as the bunch approaches the boundary (figure 5(a)). In the plasma, the force is strong enough to keep most of the bunch tightly focused. The transverse momentum, delivered by this force to individual electrons, depends on the electron position in the bunch and is very large, which makes the beam emittance blow up during the boundary crossing (figure 5(b)).

Most of the electrons in the beam head are simply reflected from the plasma (figure 6). They enter the plasma, experience a positive radial force, turn around and travel away from the plasma. An example for this behavior is particle 1 in figure 6. During their time in the plasma, they create a wakefield that can focus the rest of the beam. The wakefield force is the gradient of the wakefield potential Φ shown in figure 6(a). Here, red regions are potential wells for the electrons, and blue regions are potential humps. As we see from the potential map, the head particles are reflected from the proton beam wakefield, which is mostly repelling for electrons during the first two meters of propagation.

Particles further downstream (like particles 2–4) are confined by the potential well created by upstream particles and can even make several small-amplitude transverse oscillations in the well. This potential well traps most of the beam electrons and makes them move in a very similar manner. The transverse positions of particles 2–4 at $z = 1.25$ m almost coincide (figure 6(b)) in spite of their different initial positions and their acquired transverse momenta. The potential well follows the transverse position of the electron beam head and eventually comes out of the plasma, pulling the beam body along with it. Most of the electrons trapped in the well have sufficient transverse momenta and are able to overcome the plasma attraction and fly away (particle 2). A few, however, remain close to the plasma and oscillate near the boundary (particles 3 and 4).

As the proton beam self-modulates, the potential wells produced by the proton driver become deeper and wider, eventually reaching the plasma boundary and attracting electrons to inner plasma areas (particle 4 at $z \approx 3.5$ m). Electrons in opposite wakefield phases (potential humps in the inner area) get locked between the hump and the boundary and oscillate there until they drift to the well. This eventual drifting is possible because of the difference between the wakefield phase velocity and the electron longitudinal velocity (particle 3, which enters the central area only at $z \approx 5.5$ m).

Shortly after the self-modulation, the driver wakefield has the following structure: the wave phase in the region with $r \gtrsim c/\omega_p$, which is driven by defocused protons, is opposite to the wave phase in the region around $r = 0$, which is driven by the remaining protons. Because of this structure, electrons can oscillate for a time around a non-zero radial position inside the plasma (particle 4 at $z \approx 4$ m), before they are finally able to reach the near-axis region (particle 4), return back to the boundary (particle 3), or get lost (not shown). Close to the axis, the wakefield is strong, and the electron energy quickly grows as high as 2 GeV (particle 4). There are, however, only a few monitored electrons exhibiting a behavior like this.

Since all wakefields, including the wakefield the electron bunch generates in its rear part, are oscillating with the same plasma wavelength, there is a chance for electrons to squeeze through the boundary wakefields and to directly arrive at the axis (particle 5). These particles are very small in number, as well.

6. Summary

Even at the minimum acceptable resolution, full 3D simulations of proton-driven plasma wakefield acceleration require huge

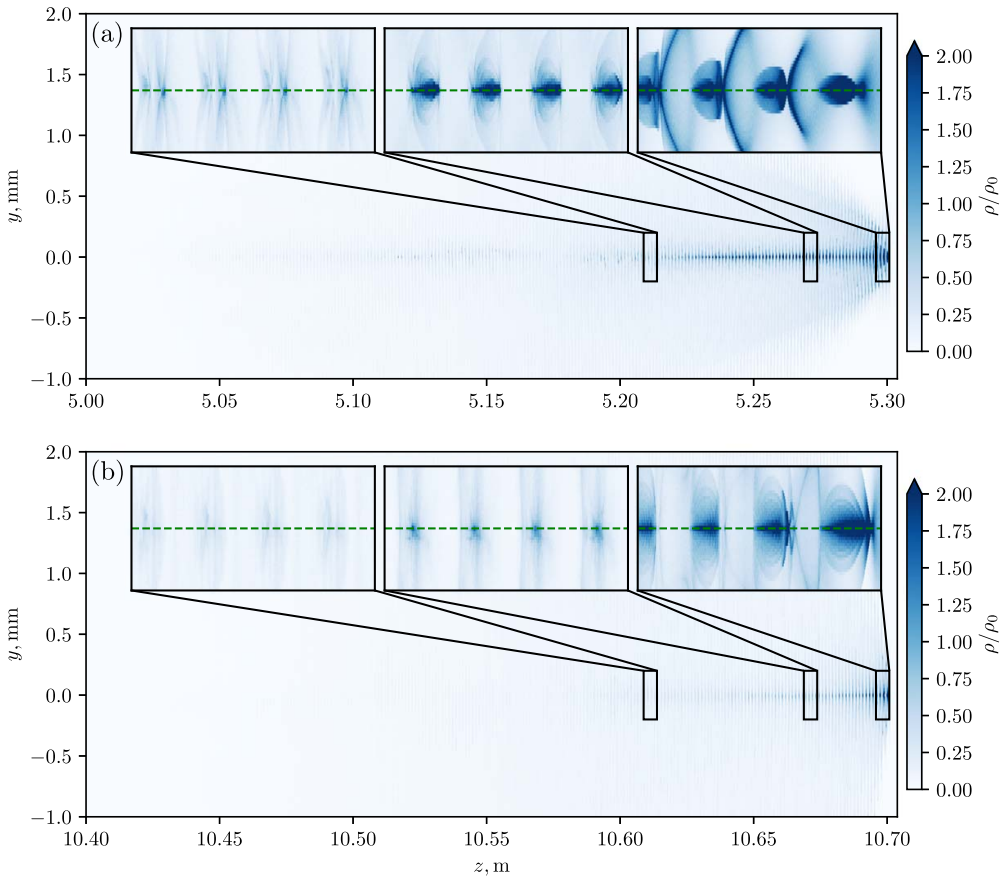

Figure 4. Proton density $\rho(y, z)$ integrated in x-direction (a) in the middle of and (b) near the exit from the plasma section. The shown figure corresponds to the perspective of the electron beam, which is looking down onto the driver. The colorbar unit ρ_0 corresponds to the density at the center of the unperturbed beam. The green dashed lines in the insets show the propagation axis.

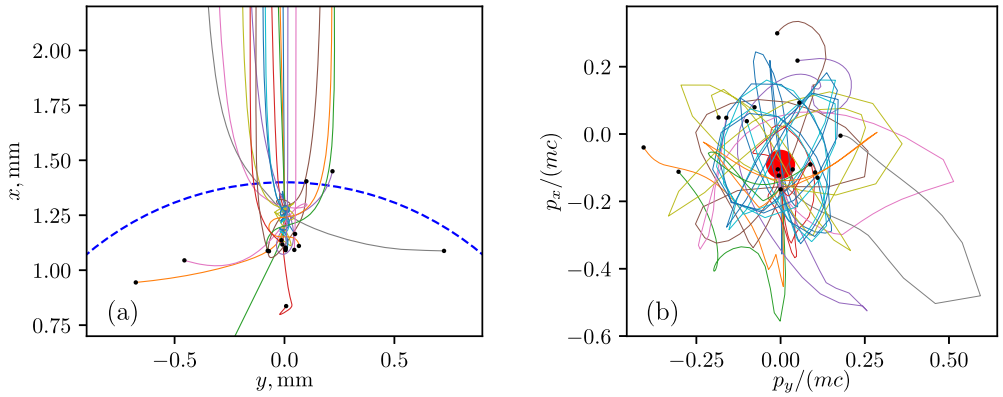

Figure 5. Parts of typical electron trajectories in (a) real and (b) momentum space. The small red circle in (b) corresponds to a beam with radius σ_{re}, energy W_e, and normalized emittance 10 mm mrad.

computational resources and should give way to reduced models wherever possible. The seeded self-modulation of proton beams is one such effect, which can reliably be simulated with axially symmetric quasi-static codes. In contrast, however, electron injection into the wakefield needs 3D simulations even if the electron bunch charge is small compared to the proton beam

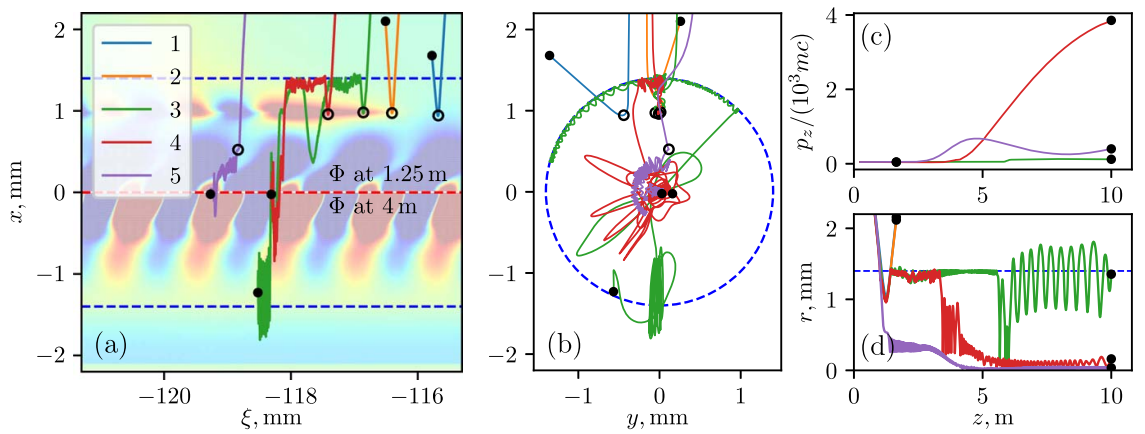

Figure 6. Typical electron trajectories in (a) and (b) real space, (c) longitudinal momentum p_z versus the propagation distance and (d) radial position r versus the propagation distance. Black dots show the endpoints of the drawn trajectories (not of the actual particle trajectories), open circles show the particle positions at $z = 1.25$ m. Dashed blue lines show the plasma boundary. Background colors in (a) show the wakefield potential Φ (x, ξ) at the midplane $y = 0$ for $z = 1.25$ m (top) and $z = 4$ m (bottom). The potential $\Phi > 0$ in regions with a red background color (wells for electrons) and $\Phi < 0$ in regions with a blue background color (humps for electrons). Optimal electron acceleration happens in regions with zero-crossings of Φ from hump to well.

charge. When a low-energy electron bunch crosses the plasma boundary, it induces its own wakefield in the plasma, which, in combination with the peripheral proton wakefield, results in the reflection of most of the beam from the plasma and blowing up the angular spread of the remaining part. This effect may be responsible for the lower amount of accelerated charge [9], compared to the predictions made with reduced models [4], in experiments.

Acknowledgments

This work was supported by the Cluster-of-Excellence Munich Centre for Advanced Photonics (MAP) and the Gauss Centre for Supercomputing (GCS), project PLASMA SIMULATION CODE, LRZ-ID pr84me, and by the Russian Fund for Basic Research, project 19-02-00243. NM, KB, FD and HR acknowledge the hospitality of the Arnold Sommerfeld Center for Theoretical Physics at the Ludwig Maximilians University and thank Patrick Boehl for the fruitful discussions and support.

ORCID iDs

N Moschuering https://orcid.org/0000-0001-5842-6467
K V Lotov https://orcid.org/0000-0001-9366-7848

References

[1] Caldwell A and Lotov K V 2011 Plasma wakefield acceleration with a modulated proton bunch *Phys. Plasmas* **18** 103101
[2] Adli E and Muggli P 2016 Proton-Beam-Driven plasma acceleration *Rev. Accel. Sci. Technol.* **9** 85
[3] Caldwell A and Wing M 2016 VHEeP: a very high energy electron-proton collider *Eur. Phys. J.* C **76** 463
[4] Caldwell A *et al* 2016 Path to AWAKE: evolution of the concept *Nucl. Instrm. Methods* A **829** 3
[5] Gschwendtner E *et al* 2016 AWAKE, the advanced proton driven plasma wakefield acceleration experiment at CERN *Nucl. Instrm. Methods* A **829** 76
[6] Muggli P *et al* 2018 AWAKE readiness for the study of the seeded self-modulation of a 400 GeV proton bunch *Plasma Phys. Control. Fusion* **60** 014046
[7] Turner M *et al* 2019 Experimental observation of plasma wakefield growth driven by the seeded self-modulation of a proton bunch *Phys. Rev. Lett.* **122** 054801
[8] Adli E *et al* 2019 Experimental observation of proton bunch modulation in a plasma at varying plasma densities *Phys. Rev. Lett.* **122** 054802
[9] Adli E *et al* 2018 Acceleration of electrons in the plasma wakefield of a proton bunch *Nature* **561** 363
[10] Xia G, Assmann R, Fonseca R A, Huang C, Mori W, Silva L O, Vieira J, Zimmermann F and Muggli P 2012 A proposed demonstration of an experiment of proton-driven plasma wakefield acceleration based on CERN SPS *J. Plasma Phys.* **78** 347
[11] Vieira J, Fonseca R A, Mori W B and Silva L O 2012 Ion motion in self-modulated plasma wakefield accelerators *Phys. Rev. Lett.* **109** 145005
[12] Lotov K V, Pukhov A and Caldwell A 2013 Effect of plasma inhomogeneity on plasma wakefield acceleration driven by long bunches *Phys. Plasmas* **20** 013102
[13] Lotov K V, Lotova G Z, Lotov V I, Upadhyay A, Tuckmantel T, Pukhov A and Caldwell A 2013 Natural noise and external wakefield seeding in a proton-driven plasma accelerator *Phys. Rev. Spec. Top. Accel. Beams* **16** 041301
[14] Lotov K V 2013 Excitation of two-dimensional plasma wakefields by trains of equidistant particle bunches *Phys. Plasmas* **20** 083119
[15] Siemon C, Khudik V, Yi S A, Pukhov A and Shvets G 2013 Laser-seeded modulation instability in a proton driver plasma wakefield accelerator *Phys. Plasmas* **20** 103111

[16] Vieira J, Fonseca R A, Mori W B and Silva L O 2014 Ion motion in the wake driven by long particle bunches in plasmas *Phys. Plasmas* **21** 056705
[17] Lotov K V, Sosedkin A P and Petrenko A V 2014 Long-term evolution of broken wakefields in finite-radius plasmas *Phys. Rev. Lett.* **112** 194801
[18] Assmann R *et al* 2014 Proton-driven plasma wakefield acceleration: a path to the future of high-energy particle physics *Plasma Phys. Control. Fusion* **56** 084013
[19] Lotov K V, Minakov V A and Sosedkin A P 2014 Parameter sensitivity of plasma wakefields driven by self-modulating proton beams *Phys. Plasmas* **21** 083107
[20] Lotov K V, Sosedkin A P, Petrenko A V, Amorim L D, Vieira J, Fonseca R A, Silva L O, Gschwendtner E and Muggli P 2014 Electron trapping and acceleration by the plasma wakefield of a self-modulating proton beam *Phys. Plasmas* **21** 123116
[21] Petrenko A, Lotov K and Sosedkin A 2016 Numerical studies of electron acceleration behind self-modulating proton beam in plasma with a density gradient *Nucl. Instrm. Methods* A **829** 63
[22] Turner M, Petrenko A, Biskup B, Burger S, Gschwendtner E, Lotov K V, Mazzoni S and Vincke H 2016 Indirect self-modulation instability measurement concept for the AWAKE proton beam *Nucl. Instrm. Methods* A **829** 314
[23] Berglyd Olsen V K, Adli E and Muggli P 2018 Emittance preservation of an electron beam in a loaded quasilinear plasma wakefield *Phys. Rev. Accel. Beams* **21** 011301
[24] Gorn A A, Tuev P V, Petrenko A V, Sosedkin A P and Lotov K V 2018 Response of narrow cylindrical plasmas to dense charged particle beams *Phys. Plasmas* **25** 063108
[25] Minakov V A, Tacu M, Sosedkin A P and Lotov K V 2018 Witness emittance growth caused by driver density fluctuations in plasma wakefield accelerators *Phys. Plasmas* **25** 093112
[26] Lotov K V 2018 AWAKE-related benchmarking tests for simulation codes *Nucl. Instrm. Methods* A **909** 446
[27] Lotov K V 2003 Fine wakefield structure in the blowout regime of plasma wakefield accelerators *Phys. Rev. Spec. Top.—Accel. Beams* **6** 061301
[28] Oz E and Muggli P 2014 A novel Rb vapor plasma source for plasma wakefield accelerators *Nucl. Instrm. Methods* A **740** 197
[29] Plyushchev G, Kersevan R, Petrenko A and Muggli P 2018 A rubidium vapor source for a plasma source for AWAKE *J. Phys. D: Appl. Phys.* **51** 025203
[30] Germaschewski K, Fox W, Abbott S, Ahmadi N, Maynard K, Wang L, Ruhl H and Bhattacharjee A 2016 The plasma simulation code: a modern particle-in-cell code with patch-based load-balancing *J. Comput. Phys.* **318** 305–26
[31] Boris J P 1970 Relativistic plasma simulation-optimization of a hybrid code *Proc. 4th Conf. on Numerical Simulation of Plasmas (Washington, DC)* (Naval Res. Lab) p 367
[32] Yee K 1966 Numerical solution of initial boundary value problems involving maxwells equations in isotropic media *IEEE Trans. Antennas Propag.* **14** 302–7
[33] Courant R, Friedrichs K and Lewy H 1928 Ueber die partiellen Differenzengleichungen der mathematischen *Phys. Math. Ann.* **100** 32–74
[34] Gedney S D 1996 An anisotropic perfectly matched layer-absorbing medium for the truncation of FDTD lattices *IEEE Trans. Antennas Propag.* **44** 1630–9
[35] Taflove A and Hagness S C 2000 *Computational Electrodynamics—The Finite Difference Time Domain Method* 2nd edn (Boston, MA: Artech House Publishers)
[36] Esirkepov T Z 2001 Exact charge conservation scheme for Particle-in-Cell simulation with an arbitrary form-factor *Comput. Phys. Commun.* **135** 144–53
[37] Walker E 2009 The real cost of a CPU hour *IEEE Comput.* **42** 35–41
[38] Sosedkin A P and Lotov K V 2016 LCODE: a parallel quasistatic code for computationally heavy problems of plasma wakefield acceleration *Nucl. Instrm. Methods* A **829** 350
[39] Lotov K V 2015 Physics of beam self-modulation in plasma wakefield accelerators *Phys. Plasmas* **22** 103110
[40] Lotov K V 2011 Controlled self-modulation of high energy beams in a plasma *Phys. Plasmas* **18** 024501
[41] Schroeder C B, Benedetti C, Esarey E, Gruner F J and Leemans W P 2013 Coherent seeding of self-modulated plasma wakefield accelerators *Phys. Plasmas* **20** 056704
[42] Vieira J, Mori W B and Muggli P 2014 Hosing instability suppression in self-modulated plasma wakefields *Phys. Rev. Lett.* **112** 205001
[43] Kumar N, Pukhov A and Lotov K 2010 Self-modulation instability of a long proton bunch in plasmas *Phys. Rev. Lett.* **104** 255003
[44] Vieira J, Fang Y, Mori W B, Silva L O and Muggli P 2012 Transverse self-modulation of ultra-relativistic lepton beams in the plasma wakefield accelerator *Phys. Plasmas* **19** 063105

6 Conclusions

This dissertation contains two main topics. The first topic, discussed in chapter 1 to 3, concerns the dynamic adaptation of the number of quasi-particles in a PIC simulation. The chapters first present a suitable theory describing the effects of adapting the quasi-particle number, followed by an algorithm that is able to perform these adaptations during a simulation, and finally a selection of simulations with interesting results and benchmarks. A direct comparison to previous work on this topic suggests that the newly developed algorithm is uniquely able to perform very efficient and correct Adaptive-Particle-Refinement (APR) during a simulation.

For future work, it seems viable to try to link the quasi-particle number in the simulation with the theoretical SNR implications of it. An algorithm can be envisioned that does not focus on keeping the actual number of quasi-particles in a certain range, but instead guarantees a certain SNR value for each cell. In doing this, a scientist performing the simulation would need to solely fix a desired SNR value for regions in the simulation, which is much more meaningful in comparison to demanding a certain number of quasi-particles.

Furthermore, it would be very interesting to make use of the APR scheme for other problems. Setups that experience a large temporal and spatial variance in the particle distribution are prime candidates to potentially benefit from employing an APR scheme. A good example for this behavior is the bubble regime ([24], [18], [6], [7]) of laser-driven plasma wakefield acceleration (LWFA) [28]. The self-injection of accelerated charges happens in a regime with very low plasma densities at the bubble border. If the quasi-particle size, and therefore the minimal plasma density, is chosen at the beginning of the simulation and subsequently stays fixed, it is necessary to either use a large number of particles at the start of the simulation, when it is not yet necessary, or have a very low SNR value during the injection process. This problem can be solved elegantly using an APR scheme. Future work could employ the algorithm presented in this work, and investigate its impact on the reliability of the obtained results.

Another interesting possibility derives from the fact that the momentum space adaptation part of the algorithm does not rely on the employed electromagnetic field solver. Therefore, this part can also be used for grid-less simulation codes.

The second main topic of this dissertation is the meaningful contribution to the AWAKE experiment, shown in chapter 4 and chapter 5. Of special additional importance are the many contributions that were made to the PSC code. Using these contributions an adaptive simulation box was realized, which saved a lot of computational resources.

Since the experiment is ongoing, there is a host of possible directions for future work. One obvious objective is to finish the campaign that is outlined in chapter 4.3. Fig. 4.6 gives the general overview of this proposed campaign. Finishing these investigations, using the developed test case, is certainly viable and could result in important benefits for the AWAKE effort.

Other research possibilities are the investigation of the side-injection efficiency, multi-stage setups, or on-axis injection schemes. It might also be viable to include the APR scheme for several of these problems, as the particle density undergoes heavy variations. The side-injection scheme harbors a lot of unknowns and thereby offers a tremendous amount of interesting investigative

directions. Again, the developed test case should make this viable and straightforward. It might prove important to develop a setup, where the simulation may be started shortly after the micro-bunching has finished and shortly before the external beam enters the plasma. This would save a lot of computational resources, but it necessitates the development of some additional code infrastructure.

The viability of multi-stage setups, setups with multiple plasma columns that are exited and entered by the proton beam, is also of interest. This would enable different injection schemes as well as modifications to the accelerating plasma length and density. The beam stability during the exiting and entering of multiple plasma columns is the main topic of interest in this case.

Finally, the on-axis injection scheme may be a more reliable alternative to the side-injection scheme. The results, published in [20] and printed here in chapter 5, suggest that side-injection may pose meaningful difficulties. These may be overcome by switching to on-axis injection.

Bibliography

[1] F. Assous, T. Pougeard Dulimbert, and J. Segré. A new method for coalescing particles in pic codes. *Journal of Computational Physics*, 187:550–571, 2003.

[2] C.K. Birdsall and D. Fuss. Clouds-in-clouds, clouds-in-cells physics for many-body plasma simulation. *Journal of Computational Physics*, 135:141–148, 1968 (Published 1997).

[3] G.-H. Cottet and P. A. Raviart. Particle methods for the one-dimensional vlasov-poisson equations. *SIAM Journal on Numerical Analysis*, 21(1):52–76, 1984.

[4] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100(1):32–74, 1928.

[5] T.Zh. Esirkepov. Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor. *Computer Physics Communications*, 135(2):144–153, 2001.

[6] J. Faure, Y. Glinec, A. Pukhov, S. Kiselev, S. Gordienko, E. Lefebvre, J.-P. Rousseau, F. Burgy, and V. Malka. A laser–plasma accelerator producing monoenergetic electron beams. *Nature*, 431(7008):541–544, Sep 2004.

[7] C. G. R. Geddes, Cs. Toth, J. van Tilborg, E. Esarey, C. B. Schroeder, D. Bruhwiler, C. Nieter, J. Cary, and W. P. Leemans. High-quality electron beams from a laser wakefield accelerator using plasma-channel guiding. *Nature*, 431(7008):538–541, Sep 2004.

[8] Kai Germaschewski, William Fox, Stephen Abbott, Narges Ahmadi, Kristofor Maynard, Liang Wang, Hartmut Ruhl, and Amitava Bhattacharjee. The plasma simulation code: A modern particle-in-cell code with patch-based load-balancing. *Journal of Computational Physics*, 318:305–326, 8 2016.

[9] Giacomo Grasso, Michele Frignani, Federico Rocchi, and Marco Sumini. Hierarchical agglomerative sub-clustering technique for particles management in pic simulations. *Nuclear Instruments and Methods in Physics Research A*, 620:56–62, 2010.

[10] Rhon Keinigs and Michael E Jones. Two-dimensional dynamics of the plasma wakefield accelerator. *Physics of Fluids*, 30(1):252, 1987.

[11] Tobias Kleine. Large scale wakefield simulations. Master's thesis, 2013.

[12] A. Bruce Langdon. Kinetic theory for fluctuations and noise in computer simulation of plasma. *Phys. Fluids*, 22:163, 1979.

[13] Giovanni Lapenta. Particle rezoning for multidimensional kinetic particle-in-cell simulations. *Journal of Computational Physics*, 181:317–337, 2002.

[14] Melvin Lax. Fluctuations from the nonequlibrium steady state. *Review of Modern Physics*, 32(1), 1960.

[15] K. V. Lotov. Physics of beam self-modulation in plasma wakefield accelerators. *Physics of Plasmas*, 22(10):103110, 2015.

[16] K. V. Lotov, V. A. Minakov, and A. P. Sosedkin. Parameter sensitivity of plasma wakefields driven by self-modulating proton beams. *Physics of Plasmas*, 21(8):083107, 2014.

[17] K.V. Lotov, A.P. Sosedkin, A.V. Petrenko, L.D. Amorim, J. Vieira, R.A. Fonseca, L.O. Silva, E. Gschwendtner, and P. Muggli. Electron trapping and acceleration by the plasma wakefield of a self-modulating proton beam. *Phys. Plasmas*, 21(arXiv:1408.4448):123116. 9 p, Aug 2014. Comments: 9 pages, 9 figures, 1 table, 44 references.

[18] S. P. D. Mangles, C. D. Murphy, Z. Najmudin, A. G. R. Thomas, J. L. Collier, A. E. Dangor, E. J. Divall, P. S. Foster, J. G. Gallacher, C. J. Hooker, D. A. Jaroszynski, A. J. Langley, W. B. Mori, P. A. Norreys, F. S. Tsung, R. Viskup, B. R. Walton, and K. Krushelnick. Monoenergetic beams of relativistic electrons from intense laser–plasma interactions. *Nature*, 431(7008):535–538, Sep 2004.

[19] Peter Meyer, Günter Wunner, Wolfgang Schmitt, and Hanns Ruder. Unified particle simulation technique for the plasma bulk and the cathode sheath of a dc glow discharge. *Journal of Applied Physics*, 77(3):992, 1995.

[20] N Moschuering, K V Lotov, K Bamberg, F Deutschmann, and H Ruhl. First fully kinetic three-dimensional simulation of the AWAKE baseline scenario. *Plasma Physics and Controlled Fusion*, 61(10):104004, sep 2019.

[21] N. Moschuering, H. Ruhl, R. I. Spitsyn, and K. V. Lotov. Generation of controllable plasma wakefield noise in particle-in-cell simulations. *Physics of Plasmas*, 24(10):103129, 2017.

[22] Nils Moschüring. Simulation of radiation generation in boosted frames. Master's thesis, Ludwig-Maximilians-Universität München, 2011.

[23] Hideo Okuda. Nonphysical noises and instabilities in plasma simulation due to a spatial grid. *Journal of Computational Physics*, 10:475–486, 1972.

[24] A. Pukhov and J. Meyer ter Vehn. Laser wake field acceleration: the highly non-linear broken-wave regime. *Applied Physics B: Lasers and Optics*, 74(4-5):355–361, January 2002.

[25] Norman Rostoker. Fluctuations of a plasma (1). *Nucl. Fusion*, 1:101–120, 1961.

[26] Hartmut Ruhl. Classical particle simulations. In M. Bonitz and D. Semkat, editors, *Introduction to Computational Methods in Many Body Physics*. Rinton Press, 2006.

[27] C.H. Shon, H. J. Lee, and J.K. Lee. Method to increase the simulation speed of particle-in-cell (pic) code. *Computer Physics Communications*, 141:322–329, 2001.

[28] Toshiki Tajima and JM Dawson. Laser electron accelerator. *Physical Review Letters*, 43(4):267, 1979.

[29] Jannis Teunissen and Ute Ebert. Controlling the weights of simulation particles: adaptive particle management using k-d trees. *Journal of Computational Physics*, 259:318–330, 2014.

[30] D.R. Welch, T.C. Genoni, R.E. Clark, and D.V. Rose. Adaptive particle management in a particle-in-cell code. *Journal of Computational Physics*, 227:143–155, 2007.

[31] D. H. D. West. Updating mean and variance estimates: an improved method. *Communications of the ACM*, 22(9):532–535, 1979.

Acknowledgements

I would like to thank all the people who supported me in writing my dissertation.

- First of all I want to thank Prof. Hartmut Ruhl, for giving me the opportunity to work in his group. He was a great supervisor, engaging me in many fruitful discussions and always being a huge support, not only scientifically. Due to his open-mindedness and his visionary approach to science he is a valuable role-model I aspire to emulate.

- I want to thank all the people working at the chair of Prof. Ruhl at the Ludwig-Maximilians-Universität München.

- Additionally, I want to mention Fabian Deutschmann, Karl-Ulrich Bamberg, Tobias Kleine, Patrick Böhl and Constantin Klier who were always available for discussions and very productive consultation. They introduced me to many software tools and always gave me important hints and suggestions. Karl-Ulrich Bamberg has been indispensable in organizing and managing the runs and the resulting data and data backups.

- The members of the AWAKE collaboration provided a lot of great help and support. I really enjoyed the collaboration meetings and the friendly and productive atmosphere. The introduction to a large professional and international research collaboration was very fascinating to me. Additionally, I want to extend my thanks and gratitude to Konstantin Lotov. His knowledge, support, advice and kindness was invaluable to me and this work.

- Prof. Dr. Kai Germaschewski, for the support in understanding the inner workings of the PSC. His insight into the C programming language and various tools were a very big help.

- The developers of TEX, LATEX and Inkscape which I used excessively to easily create and manipulate documents and graphics.

- Linus Torvalds and all other developers and contributors to Git. It is one of the greatest version management systems around and was a tremendous help in writing this thesis.

- All the developers of the Python programming language and the various packages I used to my great advantage. They spend their time, free of charge, to enable progress and advancement towards a truly great and powerful programming environment.

- Finally I want to thank my parents, my brother and my sisters. They always stood behind me as a solid support. I could always rely on their help and endorsement.