THE FAST MULTIPOLE METHOD IN THE DIFFERENTIAL ALGEBRA FRAMEWORK FOR THE CALCULATION OF 3D SPACE CHARGE FIELDS

By

He Zhang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Physics

2013

ABSTRACT

THE FAST MULTIPOLE METHOD IN THE DIFFERENTIAL ALGEBRA FRAMEWORK FOR THE CALCULATION OF 3D SPACE CHARGE FIELDS

By

He Zhang

The space charge effect is one of the most important collective effects in beam dynamics studies. In many cases, numerical simulations are inevitable in order to get a clear understanding of this effect. The particle-particle interaction algorithms and the particle-in-cell algorithms are widely used in space charge effect simulations. But they both have difficulties in dealing with highly correlated beams with abnormal distributions or complicated geometries. We developed a new algorithm to calculate the three dimensional self-field between charged particles by combining the differential algebra (DA) techniques with the fast multipole method (FMM). The FMM hierarchically decomposes the whole charged domain into many small regions. For each region it uses multipole expansions to represent the potential/field contributions from the particles far away from the region and then converts the multipole expansions into a local expansion inside the region. The potential/field due to the far away particles is calculated from the expansions and the potential/field due to the nearby particles is calculated from the Coulomb force law. The DA techniques are used in the calculation, translation and converting of the expansions. The new algorithm scales linearly with the total number of particles and it is suitable for any arbitrary charge distribution. Using the DA techniques, we can calculate both the potential/field and its high order derivatives, which will be useful for the purpose of including the space charge effect into transfer maps in the future.

We first present the single level FMM, which decomposes the whole domain into boxes of the same size. It works best for charge distributions that are not overly non-uniform. Then we present the multilevel fast multipole algorithm (MLFMA), which decomposes the whole domain into different sized boxes according to the charge density. Finer boxes are generated where the higher charge density exists; thus the algorithm works for any arbitrary charge distribution. A Message Passing Interface (MPI) based parallel version of the MLFMA is developed, so that we can take advantage of cluster machines and enhance our simulation ability. The algorithms are described in details and the numerical experimental results about the efficiency and accuracy of the algorithm are presented and discussed. In the end, we give an example of using this algorithm in the photo emission process simulation. Some simulation related topics are discussed, such as: how to choose the proper units for the variables in the beam dynamics equations, how to transform the space charge fields from the bunch frame to the laboratory frame, and how to avoid artificial collisions between the charged particles.

To my parents and my grandpa

ACKNOWLEDGMENTS

I have spent the past six years happily and gratefully on studying and doing research in the Physics and Astronomy Department at Michigan State University. Looking back, first I want to thank my adviser Professor Martin Berz. Without his continuous support and guidance, I would not be able to finish my research and my thesis. I also want to thank Professor Kyoko Makino and Professor Chong-Yu Ruan. I have benefited a lot from their kind help in the work. I truly appreciate Professor C.-P. Yuan, Professor Gary Westfall, and Professor Brad Sherrill serving in my committee.

I am grateful to my friends, Dr. He Huang and Dr. Jianghao Yu. When I started to work on the fast multipole method, I discussed a lot with Dr. Huang. Dr. Yu is always able to give suggestions when I have difficulties in physics. I also thank Ravi Jagasia, Alexander Wittig, Zhensheng Tao and Jenni Portman, for the valuable discussions and smooth cooperation we had in the work.

My appreciation also goes to the former and current beam physics group members, Johannes Grote, Pavel Snopok, Shashikant Manikonda, Youn-Kyung Kim, Alexey Poklonskiy, Pierluigi Di Lizia, Roberto Armellin, Robert Hipple, and Ben Loseth, and my friends, Hui Wang, Kaijie Xu, Kaiyu He, Feng Wu, and Chao Cheng, for their friendship.

I also want to give my acknowledgment to Debbie Barratt and Brenda Wenzlick, secretaries of the Physics and Astromony Department at Michigan State University, for their help in all these years.

Finally I want to thank my parents, who are always supporting me at any time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

A Brief Review on the Algorithms for Self-Field Calculation between Charged Particles

1.1 Introduction

The Coulomb interaction between charged particles inside a bunch is one of the most important collective effects in the study of beam dynamics. As scientists try to approach higher beam intensity, this effect becomes more significant in modern high brilliance particle accelerators or free electron laser devices. This effect may also be dominant in contemplated time resolved electric microscopes where the beam energy is relatively low [45, 48]. Although many analytical models and discussions have been made, numerical simulation is inevitable to have a good understanding of this effect in many cases.

Usually we assume all the charged particles have very small velocities in the beam frame, so that their interaction can be treated as an electrostatic field, which can be calculated from the Coulomb force law. However, the computation cost of the pairwise formula of the Coulomb force law scales with the square of the particle number N; this makes it not applicable in many cases because we often have a huge number of charged particles inside a

bunch. To increase the efficiency many fast calculation algorithms have been developed. In this chapter, we will first give a brief historic review of the algorithms developed for beam dynamics simulations, then concentrate on our algorithm, the fast multipole method (FMM) based on the differential algebra (DA) , whose computation cost scales linearly with N.

1.2 A Glimpse into the Space Charge Calculation Routines for Beam Dynamics Simulations

Beam physicists have been using numerical simulations to study the space charge effect for decades. Many fast algorithms and programs have been developed; most of them can be classified into two categories: the particle-particle-interaction (PPI) method and the particle-in-cell (PIC) method [51].

1.2.1 The Particle-Particle-Interaction Method

The basic idea of the PPI method is to use macroparticles, each of which represents a group of real particles, calculate the field on the macroparticles and move them in the simulation [60, 51, 35, 52, 50]. In order to calculate the field, one makes some assumptions concerning the bunch shape and charge distribution, such as an elliptical bunch with uniform charge distribution inside [17, 18], or fit the real charge distribution by a group of analytical functions such as a Gaussian [52, 51, 50]. Table 1.1 lists some typical PPI programs with the assumptions of their algorithm. The programs work well as long as the assumptions are valid.

MAPRO2 uses macroparticles, each representing 10^5 to 10^7 real particles, in simulation.

Table 1.1: Some PPI programs

Programs	Year	Assumptions
MAPRO2	1971	Ellipsoidal shape and Gaussian charge distribution
SC3DELP	1991	Ellipsoidal shape
SCHERM	1996	Ellipsoidal symmetry in transverse directions
Improved SCHERM	1996	NONE

The charge distribution inside the bunch is assumed to be a three dimensional Gaussian distribution such as[60]

$$n(x,y,z) = n_0 \exp\left\{-\frac{1}{2}\left(\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\right)\right\}, \tag{1.1}$$

where a, b, and c are the root mean square (rms) bunch sizes and $n_0 = 4N/((2\pi)^{3/2} a\cdot b\cdot c)$. Then the field on each macroparticle can be calculated by numerical integration in Eq. (1.2).

$$\begin{aligned}
E_x &= \frac{q\cdot a\cdot b\cdot c\cdot x}{2\varepsilon_0}\cdot\int_0^\infty \frac{n_0 \exp\left[-\frac{1}{2}\left(\frac{x^2}{a^2+s}+\frac{y^2}{b^2+s}+\frac{z^2}{c^2+s}\right)\right]ds}{(a^2+s)\sqrt{(a^2+s)(b^2+s)(c^2+s)}},\\
E_y &= \frac{q\cdot a\cdot b\cdot c\cdot y}{2\varepsilon_0}\cdot\int_0^\infty \frac{n_0 \exp\left[-\frac{1}{2}\left(\frac{x^2}{a^2+s}+\frac{y^2}{b^2+s}+\frac{z^2}{c^2+s}\right)\right]ds}{(b^2+s)\sqrt{(a^2+s)(b^2+s)(c^2+s)}},\\
E_z &= \frac{q\cdot a\cdot b\cdot c\cdot z}{2\varepsilon_0}\cdot\int_0^\infty \frac{n_0 \exp\left[-\frac{1}{2}\left(\frac{x^2}{a^2+s}+\frac{y^2}{b^2+s}+\frac{z^2}{c^2+s}\right)\right]ds}{(c^2+s)\sqrt{(a^2+s)(b^2+s)(c^2+s)}},
\end{aligned} \tag{1.2}$$

where q is the total charge of the macroparticle, ε_0 is the permittivity. MARPO2 assumes that the bunch has an elliptical shape and a Gaussian distribution of charges, which is not necessarily true in practice, especially for high density beams, and may lead to inaccurate results.

SC3DELP (1991) expands the charge distribution from the Gaussian distribution to any

arbitrary distribution, although it still assumes an elliptical shape. If the density is described as[35]

$$n(t_0) = n\left(\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2}\right), \tag{1.3}$$

where a, b, and c are the rms bunch sizes and t_0 defines the isodensity contours of the distribution, the electric field can be expressed as[35]

$$\begin{aligned} E_x &= \frac{q \cdot a \cdot b \cdot c \cdot x}{2\varepsilon_0} \int_0^\infty \frac{n(t)ds}{(a^2+s)^{3/2}(b^2+s)^{1/2}(c^2+s)^{1/2}}, \\ E_y &= \frac{q \cdot a \cdot b \cdot c \cdot y}{2\varepsilon_0} \int_0^\infty \frac{n(t)ds}{(a^2+s)^{1/2}(b^2+s)^{3/2}(c^2+s)^{1/2}}, \\ E_z &= \frac{q \cdot a \cdot b \cdot c \cdot z}{2\varepsilon_0} \int_0^\infty \frac{n(t)ds}{(a^2+s)^{3/2}(b^2+s)^{1/2}(c^2+s)^{3/2}}, \end{aligned} \tag{1.4}$$

where q is the total charge of the macroparticle, ε_0 is the permittivity, and $t = t(x, y, z, s) = x^2/(a^2+s) + y^2/(b^2+s) + z^2/(c^2+s)$. The charge density can be described as an expansion of the Fourier series, in which the coefficients can be approximated as a summation over all macroparticles. For each macroparticle, plug in the coordinates x, y, and z in Eq. (1.4) and the field can be calculated by performing the integration over s.

SCHERM (1996) only assumes the ellipsoidal symmetry in the transverse directions. The longitudinal shape is described by the overlap of two or three Gaussian functions [51]. The charge distribution can be expressed by a Hermite series expansion or a Cesaro-Fejer series expansion [50]. Since the bunch is decomposed into two or three ellipsoidal bunches, the field is the summation of the contribution from each bunch, which can be numerically calculated by Eq. (1.4). The improved SCHERM removes all the assumptions on symmetry, based on the knowledge that the three dimensional charge distribution of a bunch can be expressed

by a Hermite series expansion as[52]

$$\begin{aligned}\rho(\frac{x}{a},\frac{y}{b},\frac{z}{c}) &= \sum_{i,j,k} A_{i,j,k} H_i\left(\frac{x}{a}\right) H_j\left(\frac{y}{b}\right) H_k\left(\frac{z}{c}\right) \exp\left(-\frac{x^2}{2a^2}-\frac{y^2}{2b^2}-\frac{z^2}{2c^2}\right), \quad (1.5)\\ \text{with}\\ A_{i,j,k} &= \frac{q}{(2\pi)^{3/2} i!j!k!} \sum_{p=1}^{P} H_i\left(\frac{x_p}{a}\right) H_j\left(\frac{y_p}{b}\right) H_k\left(\frac{z_p}{c}\right).\end{aligned}$$

The charge distribution in a bunch can be considered as the Gaussian distribution

$$\rho_0(x,y,z) = A_{000} \exp\left(-\frac{x^2}{2a^2}-\frac{y^2}{2b^2}-\frac{z^2}{2c^2}\right)$$

with some modifications. The field due to the Gaussian distribution can be numerically calculated by Eq. (1.4) as before and the field due to the other terms with $A_{i,j,k} \neq A_{000}$ can be approximately calculated by Eq. (1.6) as follows [52].

$$\begin{aligned}E_x &= -\alpha A_{i,j,k} H_{i-1}\left(\frac{x}{a}\right) H_j\left(\frac{y}{b}\right) H_k\left(\frac{z}{c}\right) \exp\left(-\frac{x^2}{2a^2}-\frac{y^2}{2b^2}-\frac{z^2}{2c^2}\right),\\ E_y &= -\beta A_{i,j,k} H_i\left(\frac{x}{a}\right) H_{j-1}\left(\frac{y}{b}\right) H_k\left(\frac{z}{c}\right) \exp\left(-\frac{x^2}{2a^2}-\frac{y^2}{2b^2}-\frac{z^2}{2c^2}\right), \quad (1.6)\\ E_z &= -\gamma A_{i,j,k} H_i\left(\frac{x}{a}\right) H_j\left(\frac{y}{b}\right) H_{k-1}\left(\frac{z}{c}\right) \exp\left(-\frac{x^2}{2a^2}-\frac{y^2}{2b^2}-\frac{z^2}{2c^2}\right),\end{aligned}$$

where α, β, and γ are parameters that need to be optimized for each term and that satisfies $\alpha+\beta+\gamma=1$.

The PPI codes have been developed for several decades. In the beginning one can only treat some simple cases with the ellipsoidal symmetry in shape and the uniform distribution or the Gaussian distribution of charges. Now one can treat bunches with any arbitrary shape and any arbitrary charge distribution. The most time consuming part is to fit the charge

distribution and numerically calculate the electric field. Except for a few special cases, there is no explicit formula for the self-field inside a charged particle bunch. In most cases the field calculation depends on numerical methods. Although new algorithms are being developed [66, 73], it is always challenging to do it in a fast and accurate way. In some cases, it may be very challenging to fit the charge distribution. From Eq. (1.5) one can see it will be easier to fit the charge distribution if it is close to the Gaussian distribution. However this is not necessarily true in practice. The charge distribution of a high intensity bunch under external fields may be far away from a Gaussian distribution, which makes the fitting difficult.

1.2.2 The Particle-In-Cell Method

The PIC method is the most popular method to calculate the space charge effect in beam dynamics simulations, which has been developed for several decades. In the earlier days, assumptions on symmetry were used to help simplify the calculation and solve the problem. For example, SCHEFF, a two dimensional PIC code, assumes the bunch shape has cylindrical symmetry [69]. Thanks to the development of algorithms and computer hardware, the contemporary PIC codes can deal with the bunches containing billions of macroparticles with any arbitrary shape and charge distribution. In the latest two decades, many PIC codes, for example, WARP [43], GPT [71], VORPAL [63], and IMPACT [74], have been developed and successfully applied in beam dynamics simulation in different areas such as laser-plasma accelerators, electron guns, high intensity ion or electron storage rings, etc [42, 63]. Some PIC codes can simulate millions of particles on a PC. Parallel version of many contemporary PIC codes has also been developed [63, 74, 41], which allows them to run on the modern cluster machines. Taking advantage of supercomputers, some PIC codes can simulate billions of particles [77, 62].

The strategy of the PIC method can be described in three steps. First, one needs to set up a grid on the space with charges and distribute the charge density onto the mesh points. Then, solve the Poisson equation to obtain the electric field on the mesh points, with which the electric field on the charged particles can be calculated by interpolation. Finally solve the dynamics equations of the charged particles or solve the Vlasov equation of the flow.

So the first challenge is how to set up the grid, which has effects on accuracy and efficiency. Usually, in practice one needs to find a balance between the accuracy and the efficiency. In earlier codes, an equidistant grid is often used. But by now to efficiently deal with more complicated charge density distributions, adaptive griding techniques with nonequidistant grids have been developed, which generate a finer grid where higher charge density exists and a coarser grid where lower charge density exists [72, 87]. The second challenge is to solve the Poisson equation. This problem has been addressed by physicists and mathematicians for many years, and there are many well developed methods that can be selected such as the fast Fourier transformation based solver used in WARP [43], the Green's function based solver used in IMPACT [74], the finite difference time-domain solver used in VORPAL[63] , etc. Finally, one has to solve the dynamics equations, which is also a classical problem. Leap-frog integrators and Runge-Kutta integrators are often employed [43, 63, 71, 74]. In one sentence the PIC method is a mature and promising method to solve the self-field between charged particles.

1.2.3 The Fast Multipole Method

Since so many simulation codes for the space charge effect calculation have been developed and successfully applied, a question naturally raised is why we need a new algorithm? Why do we not simply use the existing codes? To answer this question, we have to make it

clear what we want to calculate. From the beginning, we plan to include the space charge effect into the transfer map, which is widely used in the beam dynamics simulation for single particle dynamics studies. The space charge may have significant effects on the beam dynamics, which can be reflected by the transfer map [16, 65]. In the long run, we want to develop a method that can generate the transfer map for any arbitrary field. Our group has developed a method to generate the map for any given external field [57, 58, 59]. To generate the transfer map for the self-field inside a charged particle bunch is the question in front of us. For this purpose we need to calculate not only the field but also the derivatives of the field on the reference particle. Using the PIC method, the field on the macroparticle is calculated by interpolation, as a result it is very difficult to calculate the derivatives of the field. Using the PPI method, it is possible to calculate the derivatives of the field if we use a DA based numerical integrator to calculate the field, but there always exists the danger that the fitting algorithm for the charge density distribution may fail if the charge density is too complicated. So we need an algorithm that can treat any arbitrary charge distribution and that can calculate both the field and the high order derivatives of the field.

The year 1986 brought the publication of the tree code algorithm, also known as Barnes-Hut algorithm [8], in which the potentials of the particles far away from the observer are represented by the multipole expansions in powers of $1/r$, covering larger and larger boxes further and further away from the point of interest. In this way the original point to point interaction is replaced by the point to box (that contains the multipole expansion) interaction. It could be shown that the cost of the method scales with $O(N \log N)$[8, 9]. In 1987, Greengard and Roklin published the fast multipole method (FMM), in which multipole expansions are converted into the local expansions in the near region of the observer. Thus the point to box interaction is converted into the box (that contains the multipole expan-

sion) to box (that contains the local expansion) interaction, and the efficiency is further increased to $O(N)$[39, 19].The original FMM focused on the Coulomb $(1/r)$ potential, and the three dimensional potential of a point charge is expanded in terms of the spherical harmonic functions in spherical coordinates[10]. Later on, other algorithms in the FMM family were developed for different problems, such as the screened Coulomb interaction[40, 55], electromagnetic scattering (Helmholtz equation)[32, 25], $1/r^{\nu}$ potential for any real ν [80], and potentials that have no explicit expression and can only be expressed numerically[5, 56, 34]. Some of the algorithms express their multipole expansions and local expansions in Cartesian coordinates. Both FMM and tree code have been widely applied in may areas and have also gained some attentions in beam dynamics simulations. The two dimensional FMM algorithm has been employed in Accsin to calculate the transverse space charge field [46, 47]. The tree code has been used in some electron beam emittance studies [26]. We build a new algorithm that fits our needs by combining the FMM with our differential algebra (DA) method. The FMM in the DA framework can calculate not only the three dimensional self-field, but also its high order derivatives between charged particles in any distribution. In the following chapters, the details of the DA based FMM will be presented and discussed.

Chapter 2

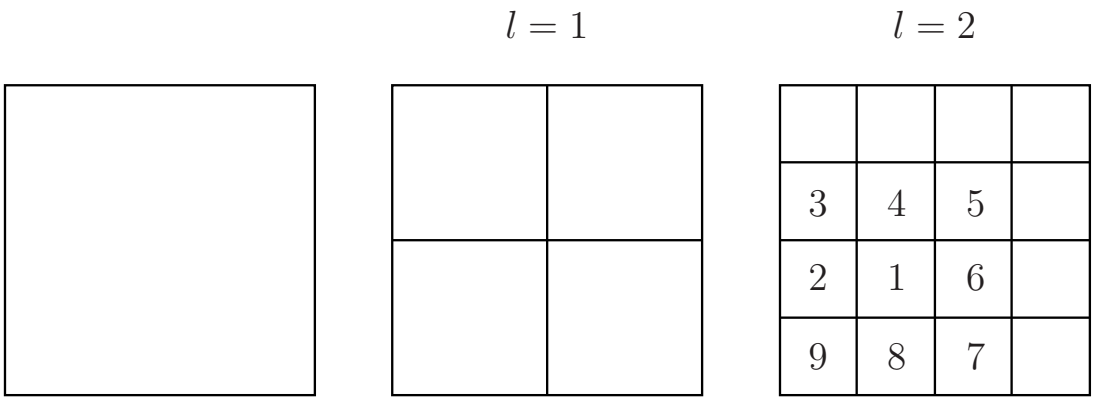
Single Level Fast Multipole Method in the Differential Algebra Framework

2.1 The Algorithm of the single level FMM

The key idea of the FMM is to represent the potential of suitable groups of source particles that are far away from the observer in terms of expansions involving powers of $1/r$, which we refer to as a far multipole expansion, and use the fact that far enough away, higher powers of $1/r$ are less and less significant. This makes it possible to compute the action of source particles on observer particles in a more efficient manner. Furthermore, far multipole expansions corresponding to certain groups can be translated and combined. And finally, far multipoles can also be locally expanded involving powers of r, which we call a local expansion. This local expansion thus allows the treatment of groups of nearby observer particles in a combined manner.

In practice, one first encloses all the particles in a group of cube boxes, the zero level boxes. Then each of these cube boxes is cut into eight equal small cube boxes, which we call the first level boxes. Then each first level box is cut in the same way into eight smaller boxes, leading to the second level boxes. This process is continued until a pre-specified level is reached. In practice, the number of levels is determined such that the average number of particles in the finest boxes is near a pre-specified value, the size of which will affect the

Figure 2.1: Sequential cutting of an original box



efficiency of the method.

We call boxes of the same level neighbors if they touch. For a given box A, we denote the region made of all same level neighbors and A itself as the near region of A, and everything else as the far region A. For a box A, a next higher level box B containing A is called a parent box of A, and A is called one of the child boxes of B. Apparently each child box has only one parent box, and each parent box has eight child boxes.

A two dimensional example about how to cut the boxes is shown in Figure 2.1. In the first level, the square box is cut into four square boxes, and in the second level, 16 boxes are cut out, and so on. Furthermore, boxes 2 through 9 are all neighbors of box 1, and together they describe the near region of 1. All the unnumbered boxes are in the far region of the box 1. There is no difference between the two dimensional case and the three dimensional case in principle, except that each box is cut into four child boxes in the two dimensional case rather than eight in the three dimensional case.

The FMM algorithm consists of two parts, the first of which we now describe.

1. Cut the box of interest into the desired levels.

2. In each box of the finest level, calculate the far multipole expansion of the particles

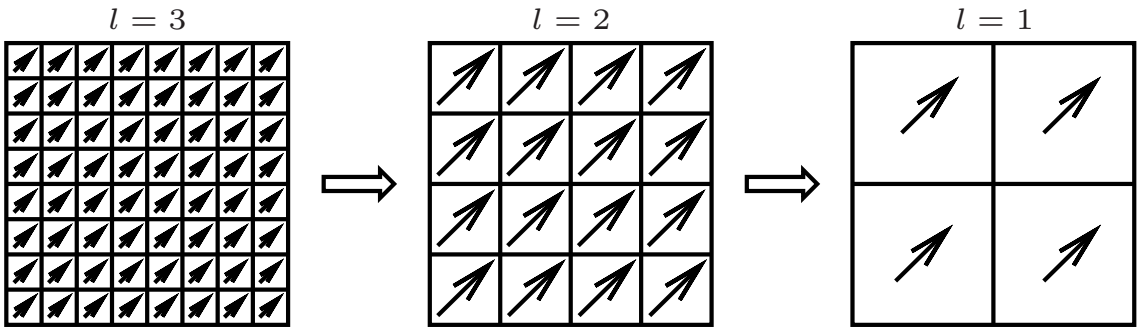
inside that box around the center of the box

3. From the finest level to the second finest level, translate the multipole expansions from the center of the children boxes into the center of their parent box, and then add them up to obtain the far multipole expansion of the parent box, as Figure 2.2 shows.

4. Repeat this at all levels.

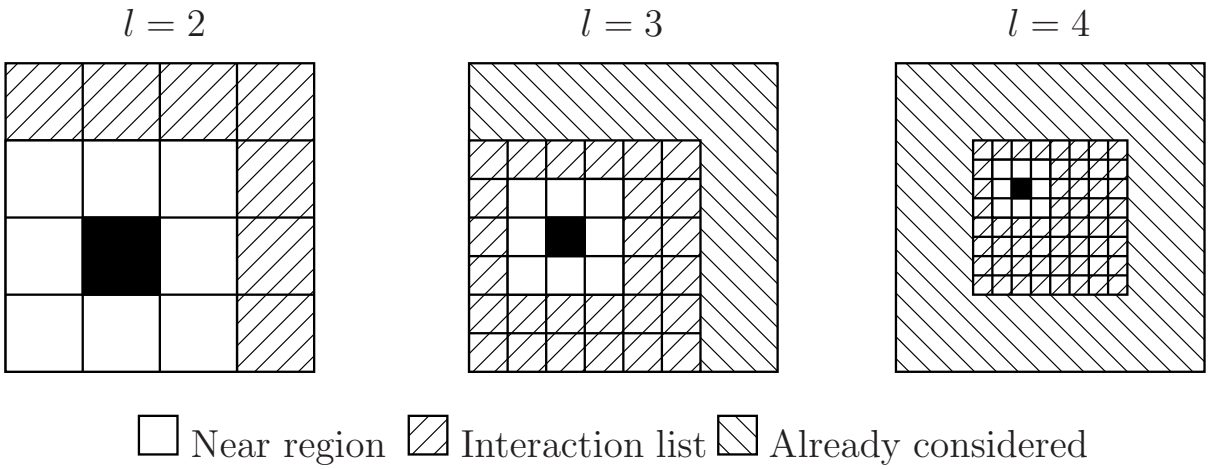
After this part is completed, we now have a far multipole expansion for each box in each level.

Figure 2.2: The propagation of the far multipole expansions



We now proceed to the second part of the FMM. First, for any box A having a parent B, we define the interaction list to be the collection of those boxes of the same level that belong to the far region of A, but to the near region of its parent B. These boxes of "medium distance" to A will be important in the subsequent algorithm because they require special treatment. To illustrate the concept of the interaction list, consider Figure 2.3, which shows a box of interest in black, its near region in white, and its interaction list by hatching of diagonal lines from lower left to upper right, for three different levels.

Figure 2.3: The behavior of the interaction list of boxes at three levels



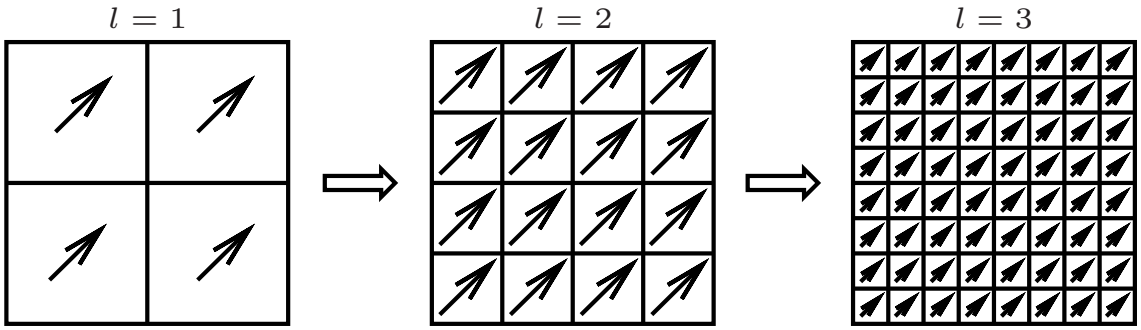
The second part of the FMM algorithm now begins at the first level at which boxes have interaction lists.

1. For each box with an interaction list, compute a local expansion around its center point from the far expansions of the boxes making up the interaction list

2. At the next finer level, again compute a local expansion around the center of each box from the far expansion of the boxes making up the interaction list. To this, add the local expansion of the parent's interaction list by re-expanding the parent's local expansion to its own center, as shown in Figure 2.4

3. Proceed until the lowest level is reached.

Figure 2.4: The translation of the local expansions



After the completion of the second step, each box at the finest level now possesses a local expansion of all contributions to the potential from outside its near region. The gradient of this local expansion consequently represents the field contributions from everything in the far region.

The third step of the method now consists of determining the field on each particle of interest.

1. For each particle, determine the finest level box to which the particle belongs, and evaluate the local expansion of the field of that box at the particle's coordinates. This results in an approximation of the field at the particle due to all other particles outside the near region of the box in which the particle lies.

2. To the far field, explicitly add the Coulomb field of all other particles in the near region.

We note that the accuracy of this approximation depends on the orders used for the far expansions and the local expansions that occur, and in principle can be made as high as desired.

In the remainder of the paper we will discuss how to perform all the relevant far and local expansions using the differential algebraic method.

2.2 A Brief Review of Differential Algebra and COSY

Before we continue to present our work on the FMM in the DA framework, we want to give a brief introduction of the DA method. The basic concepts and the deductions will be presented directly without proof, please refer to [14] for details.

Consider the vector space of the infinitely differentiable functions $C^\infty(R^v)$, in which we can define an equivalence relation "$=_n$" between two functions $a, b \in C^\infty(R^v)$ via $a =_n b$ if $a(0) = b(0)$ and if all the partial derivatives of a and b at 0 agree up to the order n. Note that the point 0 is selected for convenience, and any other point could be chosen as well. The set of all b that satisfies $b =_n a$ is called the equivalence class of a, which is denoted by $[a]_n$. We denote all the equivalence classes with respect to $=_n$ on $C^\infty(R^v)$ as ${}_nD_v$. The addition, scalar multiplication and multiplication on ${}_nD_v$ can be defined as eq. (2.1)

$$\begin{aligned} [a]_n + [b]_n &:= [a+b]_n, \\ c \cdot [a]_n &:= [c \cdot a]_n, \\ [a]_n \cdot [b]_n &:= [a \cdot b]_n, \end{aligned} \tag{2.1}$$

where $a, b \in {}_nD_v$ and c is a scalar, so that ${}_nD_v$ is an algebra. We can also define the derivation operator ∂_v as eq. (2.2)

$$\partial_v [a]_n := \left[\frac{\partial}{\partial x_v} a \right]_{n-1}, \tag{2.2}$$

where x_v is the v^{th} variable of the function a. The operator ∂_v satisfies

$$\partial_v([a]\cdot[b]) = [a]\cdot(\partial_v[b]) + (\partial_v[a])\cdot[b] \tag{2.3}$$

An algebra with a derivation is called a differential algebra. There are v special classes $d_v = [x_v]$, whose elements are all infinitely small. According to the fixed point theorem[14], the inverse and the roots of any element that is not infinitely small in ${}_nD_v$ exist and can be calculated easily. Furthermore, all real power series can be extend to the DA within their radius of convergence. If a function a in ${}_nD_v$ has all the derivatives $c_{J_1,...J_v} = \partial^{J_1+...+J_v}a/\partial x_1^{J_1}\cdot...\cdot\partial x_v^{J_v}$, then $[a]$ can be written as

$$[a] = \sum c_{J_1,...J_v}\cdot d_1^{J_1}\cdot...\cdot d_v^{J_v}. \tag{2.4}$$

Thus $d_1^{J_1}\cdot...\cdot d_v^{J_v}$ is a basis of the vector space of ${}_nD_v$. The eq. (2.4) reminds us of the Taylor expansion of a function. Actually if we have a function f in $C^\infty(R^v)$ and f_T is its Taylor expansion up to order n, obviously we have $f =_n f_T$ in ${}_nD_v$. In practice this means we can express f by its Taylor expansion up to an arbitrary order n as an element in ${}_nD_v$, and we can calculate the derivative classes of f and any other function that can be derived by applying the elemental operations, divisions, roots, and power series on f. [11]

A beam optical system can be described in terms of the transfer map method, which relates the final positions and velocities of the particles to the initial conditions, so that it makes the tracking more efficient than solving the dynamics equations element by element. The power of the DA make it possible to calculate the map up to any arbitrary order[12].

COSY Infinity is a program developed for high performance modern scientific computing,

which is used in beam optical system design. COSY supports various advanced data types, such as DA, TM and VE[15]. In our following work on the FMM, we will use the DA data type. By evaluating a function f in DA data type in COSY, one can obtain its Taylor expansion f_T up to an arbitrary predetermined order n , because a DA vector carries all its coefficients $c_{J_1,\ldots,J_v}$, which are exactly the coefficients of f_T. If we consider two functions $f, g \in {}_nD_v$ which can also be viewed as two maps M_f and M_g, the function $g(f)$ or the composition of the two maps $M_g \circ M_f$ can be calculated by the command POLVAL in COSY. If f is a linear function or a linear map, a more efficient command DATRN can be used.[15] Furthermore, the TM data type can be used for the rigorous calculation. In the future we will use the TM in our algorithm to obtain rigorous error bounds.[13] COSY also supports parallel calculation. Our current code can be easily revised for the parallel calculation and run in a cluster machine.

2.3 FMM in the DA Framework

2.3.1 The Far Multipole Expansion

Given a unit point charge located at position $\vec{r_i}$, its potential at the location $\vec{r}$ of an observer is given by

$$\phi = \frac{q}{|\vec{r} - \vec{r_i}|}, \tag{2.5}$$

The key idea of the FFM method lies in grouping together particles that are sufficiently near and treat them with a combined expansion. As it turns out, it is not immediately obvious what the best choice of variables is to achieve this end. It is known that using spherical

coordinates, ϕ can be expressed as series of the spherical harmonic functions, which are composed of products of $\cos\theta$, $e^{i\psi}$ and $1/r$ [10]. The desire to operate directly in Cartesian coordinates ultimately leads to the inspiration of expanding the function with respect to $1/r$, x/r^2, y/r^2, and z/r^2. As it turns out this is one variable more than necessary and contains redundancy because (x, y, z) already determines r. But the use of four variables leads to a particularly transparent algorithm.

Back to the FMM, we consider a cube box centered at the origin, whose side length is a, and there are n particles inside the box whose coordinates are (x_i, y_i, z_i). At any point (x, y, z), whose distance to the origin $r \gg a$, the potential can be expressed as

$$\begin{aligned}
\phi &= \sum_i^n \frac{q_i}{\sqrt{(x_i-x)^2+(y_i-y)^2+(z_i-z)^2}} \\
&= \sum_i^n \frac{q_i/\sqrt{x^2+y^2+z^2}}{\sqrt{1+\frac{x_i^2+y_i^2+z_i^2}{x^2+y^2+z^2}-\frac{2x_ix}{x^2+y^2+z^2}-\frac{2y_iy}{x^2+y^2+z^2}-\frac{2z_iz}{x^2+y^2+z^2}}} \\
&= \sum_i^n \frac{d_1}{\sqrt{1+(x_i^2+y_i^2+z_i^2)d_1^2-2x_id_2-2y_id_3-2z_id_4}}
\end{aligned} \tag{2.6}$$

with

$$\begin{aligned}
d_1 &= \frac{1}{\sqrt{x^2+y^2+z^2}} = \frac{1}{r}, \quad d_2 = \frac{x}{x^2+y^2+z^2} = \frac{x}{r^2}, \\
d_3 &= \frac{y}{x^2+y^2+z^2} = \frac{y}{r^2}, \quad d_4 = \frac{z}{x^2+y^2+z^2} = \frac{z}{r^2}.
\end{aligned}$$

If we consider points (x, y, z) whose distance to the origin $r \gg a$, the expression above can apparently expanded in powers of the quantities d_1, d_2, d_3 and d_4, since these quantities go to zero as r becomes larger and larger. The resulting expansion mathematically represent a

Taylor series in d_1, d_2, d_3 and d_4, and in common physical terminology a multipole expansion. To avoid confusion with the common definition of multipole expansions in particle optics, we call this expansion the far multipole expansion if it is not clear from the context.

Using the variables d_1, d_2, d_3 and d_4 as the expansion variables in the DA algorithm, COSY can now readily compute the far multipole expansions to any order. Furthermore, quite importantly, the same method can be applied to other types of the potential, for example the two dimensional Coulomb potential, the common van der Waals potentials, or the potentials of certain macroparticle charge distributions, and in the DA formalism, without much additional difficulty far multipole expansions can be calculated by evaluation in DA arithmetic.

We now use the formula to calculate the multipole expansions in the finest level boxes. To assess the performance of the method, we performed some test calculations of the far multipole expansion. One group of results are presented in Table 2.1. We put 50 electrons into each of the two cube boxes, whose centers are $(0, 0, 0)$ and $(4, 0, 0)$ and whose side length are both 2. In terms of the terminology introduced above, this means the second box lies in the far region of the first and is hence subject to the far multipole expansion; but it represents the closes such box, where thus the convergence of the method is particularly difficult to achieve.

The potential on the electrons in the second box are calculated and repeated for 1000 times with different random numbers, so that we have 50000 data points together. The relative error is calculated by $\sqrt{\sum_{i=1}^{N}(\Delta\phi/\phi)^2}$ with $N = 50000$. We can see as the DA order increases, the relative error decreases, which shows the convergence of this multipole expansion.

Table 2.1: The relative errors of the potential calculated from the multipole expansions

DA order	Error
1	2.088×10^{-2}
2	3.816×10^{-3}
3	1.524×10^{-3}
4	9.681×10^{-4}
5	1.381×10^{-4}
6	5.758×10^{-5}

2.3.2 Translation of a Multipole Expansion

If we have a multipole expansion in the cube box centered at the origin $(0, 0, 0)$, we can translate it into another frame whose origin is at (x'_o, y'_o, z'_o). The distance from the observer to the origin of the new frame is assumed to be much larger than the side length of the box centered at (x'_o, y'_o, z'_o), which assures that the new DA variables $d'_1 ... d'_4$ in eq. (2.7) are small enough to still have convergence. Specifically, we have

$$\begin{aligned} d'_1 &= \frac{1}{\sqrt{(x-x'_o)^2+(y-y'_o)^2+(z-z'_o)^2}} = \frac{1}{\sqrt{x'^2+y'^2+z'^2}} = \frac{1}{r'} \\ d'_2 &= \frac{x-x'_o}{r'^2} = \frac{x'}{r'^2} \\ d'_3 &= \frac{y-y'_o}{r'^2} = \frac{y'}{r'^2} \\ d'_4 &= \frac{z-z'_o}{r'^2} = \frac{z'}{r'^2} \end{aligned} \tag{2.7}$$

In eq. (2.7), (x', y', z') is the position of the observer in the new frame of $(x'_o, y'_o z'_o)$, and r' is the distance from the observer to the new origin (x'_o, y'_o, z'_o). The new DA variables and

the old DA variables in eq. (2.6) are related via

$$\begin{aligned}
d_1 &= \frac{1}{r} = \frac{1}{\sqrt{(x-x'_o+x'_o)^2+(y-y'_o+y'_o)^2+(z-z'_o+z'_o)^2}} \\
&= \frac{1}{\sqrt{\begin{array}{c}(x-x'_o)^2+(y-y'_o)^2+(z-z_o)^2+x'^2_o+y'^2_o+z'^2_o \\ +2(x-x'_o)x'_o+2(y-y'_o)y'_o+2(z-z'_o)z'_o\end{array}}} \\
&= \frac{1}{\sqrt{1/d'^2_1+x'^2_o+y'^2_o+z'^2_o+2x'_o d'_2/d'^2_1+2y'_o d'_3/d'^2_1+2z'_o d'_4/d'^2_1}} \\
&= \frac{d'_1}{\sqrt{1+(x'^2_o+y'^2_o+z'^2_o)d'^2_1+2x'_o d'_2+2y'_o d'_3+2z'_o d'_4}} \\
&= d'_1\cdot\sqrt{R}, \qquad (2.8)\\
d_2 &= xd_1^2 = (x-x'_o+x'_o)d_1^2 = (d'_2+x'_o d'^2_1)\cdot R, \\
d_3 &= yd_1^2 = (y-y'_o+y'_o)d_1^2 = (d'_3+y'_o d'^2_1)\cdot R, \\
d_4 &= zd_1^2 = (z-z'_o+z'_o)d_1^2 = (d'_4+z'_o d'^2_1)\cdot R,
\end{aligned}$$

with

$$R = \frac{1}{1+(x'^2_o+y'^2_o+z'^2_o)d'^2_1+2x'_o d'_2+2y'_o d'_3+2z'_o d'_4}.$$

If we substitute eq. (2.8) into eq. (2.6), we obtain the multipole expansion in the new frame. If one considers eq. (2.6) as a map $M_{\mathrm{c2m}}(d_1,d_2,d_3,d_4)$ and eq. (2.8) as a map $M_1(d'_1,d'_2,d'_3,d'_4)$, the substitution is actually merely the composition of the two maps

$$M_{\mathrm{m2m}} = M_{\mathrm{c2m}}\circ M_1, \qquad (2.9)$$

which is a common operation in DA methods, and which here yields the expression of ϕ in the new frame with the new DA variables. With eq. (2.8) and eq. (2.9), to calculate

the multipole expansion in a box not in the finest level, we can translate all the multipole expansions of all its children boxes to its center, then take the summation.

2.3.3 Conversion of a Far Multipole Expansion into a Local Expansion

The potential of a far multipole expansion at the frame at $(0, 0, 0)$ on an observer (x, y, z) can be converted into a local expansion in the near region of the observer. Since the new frame has its origin (x'_o, y'_o, z'_o) close to the observer, it is natural to choose the coordinates of the observer (x', y', z') in the new frame as the new DA variables, as eq. (2.12) shows.

$$\begin{aligned} d'_1 &= x', \\ d'_2 &= y', \\ d'_3 &= z'. \end{aligned} \tag{2.10}$$

So, in contrast to the far multipole expansion, which is a Taylor expansion in the variable d_1, d_2, d_3 and d_4, which vanish at infinity but diverge in close proximity, we now generate a Taylor expansion in the variables x', y' and z', which vanish at the origin. The old DA variables in eq. (2.6) and the new DA variables in eq. (2.12) have the relation as shown in

eq. (2.11).

$$
\begin{aligned}
d_1 &= \frac{1}{\sqrt{x^2+y^2+z^2}} = \frac{1}{\sqrt{(x'_o+d'_1)^2+(y'_o+d'_2)^2+(z'_o+d'_3)^2}} = \sqrt{R}, \\
d_2 &= \frac{x}{x^2+y^2+z^2} = \frac{x'_o+d'_1}{(x'_o+d'_1)^2+(y'_o+d'_2)^2+(z'_o+d'_3)^2} = (x'_o+d'_1)\cdot R, \\
d_3 &= \frac{y}{x^2+y^2+z^2} = \frac{y'_o+d'_2}{(x'_o+d'_1)^2+(y'_o+d'_2)^2+(z'_o+d'_3)^2} = (y'_o+d'_2)\cdot R, \\
d_4 &= \frac{z}{x^2+y^2+z^2} = \frac{z'_o+d'_3}{(x'_o+d'_1)^2+(y'_o+d'_2)^2+(z'_o+d'_3)^2} = (z'_o+d'_3)\cdot R.
\end{aligned}
$$

with (2.11)

$$
R = \frac{1}{(x'_o+d'_1)^2+(y'_o+d'_2)^2+(z'_o+d'_3)^2}.
$$

If we call eq. (2.11) the map $M_2(d'_1, d'_2, d'_3)$, the local expansion can again be written as a composition of M_{c2m} and M_2 as

$$
M_{\mathrm{m2l}} = M_{\mathrm{c2m}} \circ M_2. \tag{2.12}
$$

Having eq. (2.11) and eq. (2.12), we can convert the multipoles in the interaction list of each box into the local expansion inside the box itself. The result of a test calculation to show the convergence of the local expansion is presented in Table 2.2. The set up of the boxes and electrons, the number of the data points, and the definition the relative error are the same with those of Table 2.1. We can see that the relative error decreases as we increase the DA order.

Table 2.2: The relative errors of the potential calculated from the local expansions

DA order	Error
1	3.136×10^{-2}
2	8.348×10^{-3}
3	3.024×10^{-3}
4	1.165×10^{-3}
5	2.086×10^{-4}
6	8.036×10^{-5}

2.3.4 Translation of a Local Expansion

Consider two local frames whose origins are both close to the observer. If we know the local expansion of the frame at $(0, 0, 0)$ is as eq. (2.12) shows, we can translate it into the other local frame at (x'_o, y'_o, z'_o). Since the new frame's origin is also close to the observer, we can still choose the coordinates of the observer in the new frame as the new DA variables d'_1, d'_2, and d'_3. Obviously the old DA variables d_1, d_2, and d_3 , which are the coordinates of the observer in the local frame at $(0, 0, 0)$, have the simple relation shown in eq. (2.13) with the new DA variables.

$$\begin{aligned} d_1 &= x'_o + d'_1, \\ d_2 &= y'_o + d'_2, \\ d_3 &= z'_o + d'_3. \end{aligned} \tag{2.13}$$

We call eq. (2.13) the map M_3, then the new local expansion can be calculated by composing M_{m2l} with M_3 as

$$M_{\mathrm{l2l}} = M_{\mathrm{m2l}} \circ M_3. \tag{2.14}$$

We can translate the local expansion of a box into its children boxes using eq. (2.13) and eq. (2.14).

2.3.5 Representation of Potential and the Field as Polynomials

In order to calculate the high order transfer map of a beam optical system, one needs not only the field on the reference orbit but also the derivatives of the field[12]. If we have the field of the Coulomb interaction and its derivatives, we can include the Coulomb interaction into the map. Eq. (2.14) provides the expansion of the potential ϕ with respect to the position of the observer up to the order p. This expansion converges inside the corresponding finest box. The constant part of the expansion is the value of the potential at the center of the box, and the coefficient of each monomial is the value of the corresponding partial derivative of the potential.

Table 2.3 shows an example of the local expansion. In a large cube box, we put 10000 electrons. The cube box is cut into the third level. And the third level box size is 2, which means if we put a frame at the center of one finest box, we have $x, y, z \in [-1, 1]$. The local expansion of each 3rd level box is calculated up to the 10^{th} order, and the coefficients till the third order of one 3rd level box are presented in Table 2.3. The forth column and the eighth column show the exponents of each DA variables in each monomial. For example, the index triple 2 1 0 denotes the monomial $x^2y^1z^0 = x^2y$. The second and sixth columns show the coefficient of each monomial. From the table, we observe the decreasing trend of the coefficients according to the increase of the order. To obtain sufficiently meaningful statistics, we repeat the above process for 40 times with different groups of electrons. Each time we find the maximum absolute value of the coefficients for each order and calculate the average absolute value for the coefficients for each order. Then we take the average over the

Table 2.3: A local expansion

I	COEFFICIENT	p	EXP.	I	COEFFICIENT	p	EXP.
1	-10173.27230228843	0	0 0 0	11	-.3770785081465575	3	3 0 0
2	77.96202591617063	1	1 0 0	12	0.3065719640561415	3	2 1 0
3	84.83337126367421	1	0 1 0	13	1.430687924061862	3	1 2 0
4	94.59868392563654	1	0 0 1	14	0.1167018056511110	3	0 3 0
5	-3.199969762547595	2	2 0 0	15	-1.761289530718459	3	2 0 1
6	4.505077944915953	2	1 1 0	16	0.7359490919857412	3	1 1 1
7	2.824067476091086	2	0 2 0	17	-1.758649785663027	3	0 2 1
8	1.929372636609234	2	1 0 1	18	-.2994523996221934	3	1 0 2
9	-3.396989424915588	2	0 1 1	19	-.6566773810094656	3	0 1 2
10	0.3759022864564996	2	0 0 2	20	1.173313105460495	3	0 0 3

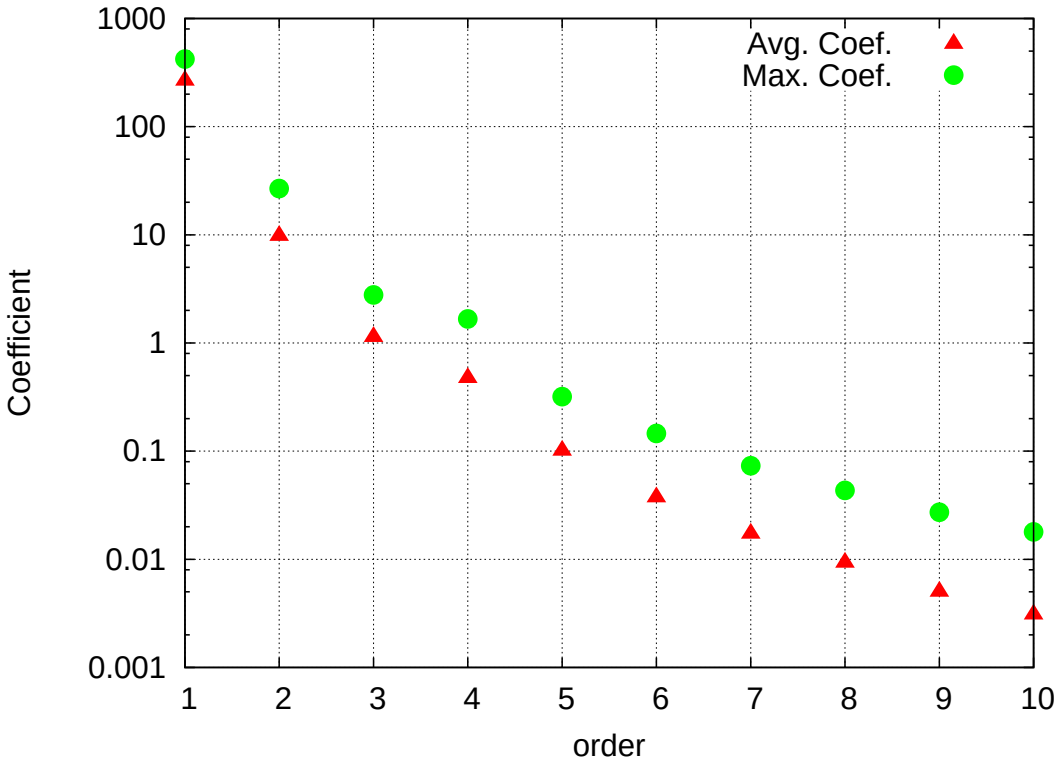
40 processes and present the result in Figure 2.5. The green dots show the maximum value of the coefficients, and the red dots show the average value of the coefficients. It is clear that the coefficients decrease as the order increase, which suggests the convergence of the local expansion.

If we want to calculate the expansion of the Coulomb interaction field on a reference point, we only need to determine which finest box the point belongs to, then by Eq. (2.13) and Eq. (2.14) we can translate the local expansion of the potential from the center of the finest box to the reference point. Taking the derivative of the position (x, y, z), we obtain the expansion of the field (E_x, E_y, E_z) up to the order $p - 1$. The expansion of the field only includes the contribution of the electrons in the far region. As to the electrons in the near region, we obtain the extension of their field by representing each of them by Gaussian distribution, which will be discussed in the following.

2.3.6 Gaussian Macroparticles Instead of Point Charges

Sometimes it is important to use a finite number of particles to represent a continuous distribution. There are two important cases where this arises; the first is when the number of

Figure 2.5: The absolute value of the different order coefficients of the local expansion. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this thesis



actual particles is substantially larger than the number than what can practically be handled in a simulation. The second case is the desire to compute a transfer map and aberrations of the system at hand, which requires the knowledge of high-order field expansions near the reference point, which cannot be done reliably if point charges are near. To avoid this, one can use three dimensional spherical Gaussian distributions instead of point charges as

$$f(x,y,z) = \frac{q}{(\sqrt{2\pi}\sigma)^3} \cdot \exp(-\frac{x^2}{\sigma^2} - \frac{y^2}{\sigma^2} - \frac{z^2}{\sigma^2}) = \frac{1}{(\sqrt{2\pi}\sigma)^3} \cdot \exp(-\frac{r^2}{\sigma^2}). \tag{2.15}$$

It is easy to see from Gauss's law that the field along the radius can be expressed as

$$E_r = \frac{q}{r^2\sqrt{2\pi}\sigma^3}\left[\sqrt{2\pi}\mathrm{erf}(\frac{r}{\sqrt{2}\sigma})\sigma^3 - 2r\sigma^2\exp(-\frac{r^2}{2\sigma^2})\right], \tag{2.16}$$

where $\mathrm{erf}(r/\sqrt{2}\sigma)$ is the error function. By setting a proper value for σ, we can use the Gaussian distribution in eq. (2.15) to represent a point charge. When $r \gg \sigma$, E_r is equal to the field of a point charge. When $r \to 0$, $E_r \to 0$. To calculate the expansion of the error function, we need to find all the derivatives of it. Considering the definition of the error function

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt, \tag{2.17}$$

it is easily to determine its derivative

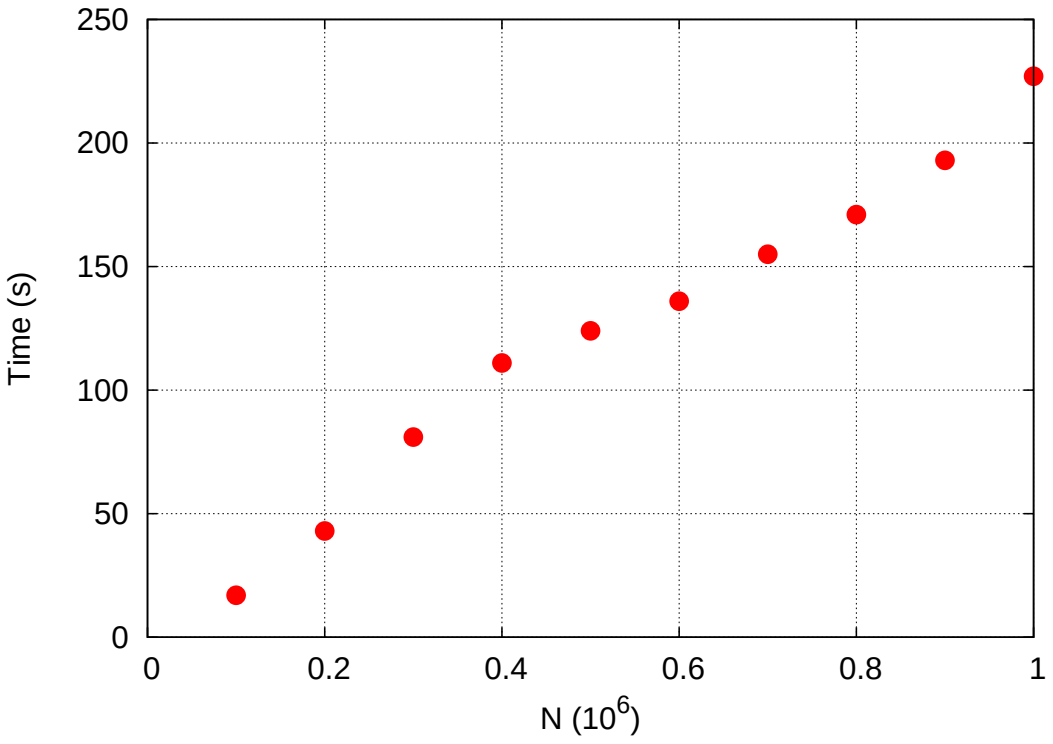
$$\frac{d\mathrm{erf}(x)}{dx} = \frac{2}{\sqrt{\pi}}e^{-x^2}. \tag{2.18}$$

For any given point x, we calculate $\mathrm{erf}(x)$ by the rational Chebyshev approximation[24] and $d\mathrm{erf}(x)/dx$ by eq. (2.18), which is infinitely differentiable and allows the computation of all higher derivatives.

2.3.7 Example Calculations Exhibiting the Linear Scaling Property

To show the performance of the method, for reasons of space we limit ourselves here to the most important characteristic, namely the scaling of the computation time to the number of particles. We perform a series of simulations with expansion order 5 and with varying numbers of electrons N on our computer with 4 core Intel Xeon Processor X5677 running

Figure 2.6: Computation time for different number of electrons



at 3.5 GHz. The single process program uses only one core in the computation. Figure 2.6 shows the required computation time for a simulation with varying particle numbers N. We observe the nearly linear behavior, with a slight deviation around $N = 0.4 \times 10^6$, at which point the number of particles crosses a threshold for transitions from a three-level scheme to a four level scheme, which leads to slightly improved performance after the transition. Overall, the expected linearity is well achieved.

2.4 Use of the FMM in tracking simulations

The FMM described above always starts from a cubical box, then cuts it into eight child boxes and keeps all of them. The method works well when the bunch sizes in all three dimensions are close to each other. However, in practice this is not always the case. In fact,

usually, the bunch sizes evolve with time, and the bunch may have an oblate shape where one of the dimensions is much smaller than the others, or a prolate shape where one of the dimensions is significantly larger than the other two. In these cases, the FMM above will lose some of its efficiency because a large number of the resulting boxes are empty.

However, what plays the critical role in the FMM algorithm is the hierarchical relation between the boxes, while the details of the cutting can be modified as necessary. We will now discuss a more efficient way of cutting boxes which still maintains the necessary hierarchical relation.

2.4.1 Strategy of cutting boxes

Assume we know the length of the bunch in all the three dimensions (L_x, L_y, L_z). Without losing generality, we assume $L_x \geq L_y \geq L_z$. We also know the total number of the particles N, and the average number of particles inside each box of the finest level n, from which we can obtain the least number of boxes needed N_b. The new idea of box cutting can be described as follows.

1) Enclose the bunch by a cubic box whose size is slightly greater or equal to the longest side of the bunch L_x. We say this box is of level zero. Without losing generality we assume this box is centered at $(0, 0, 0)$.

2) Try to cut this cube box from the center into eight equal size cube boxes. Compare the coordinates (c_x, c_y, c_z) of the center of each child box with $(L_x/2, L_y/2, L_z/2)$. If a child box whose center satisfies $|c_i| < L_i/2$ for all the three directions, it is accepted. Otherwise, if in any direction, $|c_i| < L_i/2$ is not satisfied, c_i is set to zero to obtain a new center point with the coordinates of the other directions unchanged, and the child box is replaced by a new box that is centered at the newly obtained point. In this way the number of child boxes

is decreased. For example, assuming two child boxes with centers (x_1, y_1, z_1) and (x_1, y_1, z_2), if in x and y directions, $|c_i| < L_i/2$ is satisfied, but the absolute values of both z_1 and z_2 are greater than $L_z/2$, they will be replaced by one box whose center is $(x_1, y_1, 0)$. The side length of the child boxes is set to be half of the parent box.

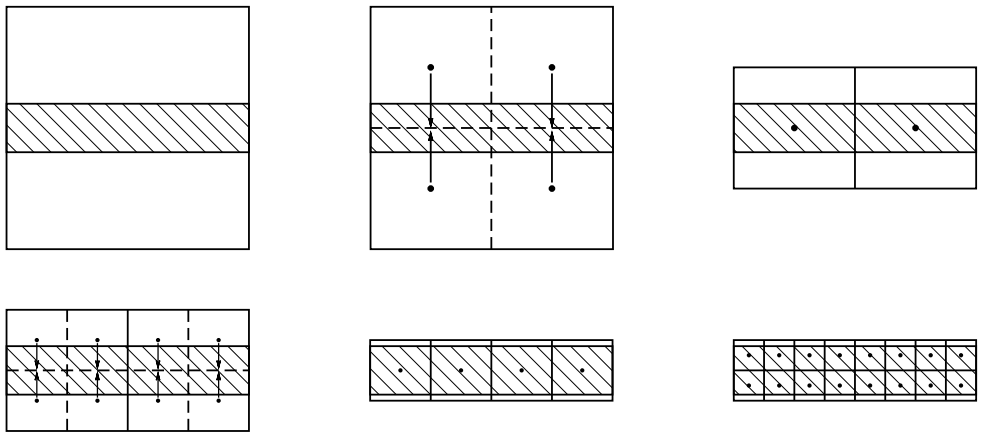
3) Cut each box of the current finest level in the same way as 3) to obtain the boxes of a finer level.

4) Repeat 3) until the box number at the finest level is greater than or equal to N_b.

To make the idea clearer, let us consider a two dimensional example as shown in Figure 2.7. The hatching part shows the dimensions of the bunch. First, we enclose the bunch by a square box. Then we try to cut the square box into four child boxes, but we notice that the y coordinates of all the centers of the child boxes do not satisfy $|c_y| < L_y/2$. So we set the y coordinates of them zero, and the four center points fall into two new points as Figure 2.7 shows. We use these two points as the centers of the new child boxes whose side length is half of their parent box, so that we obtain two child boxes of the first level instead of four. Then we repeat the same cutting process for each box of the first level, we obtain two child boxes for each of them. So we have four boxes of the second level. When we cut the boxes of the second level, we find all the centers of the child boxes satisfy $|c_i| < L_i/2$, so we accept all of them and obtain 16 boxes of the third level. As shown in Figure 2.7, when we use one square box to enclose the bunch in the beginning, there is a lot of blank place in the box. But after we find the proper cutting, the dimension of the boxes fits the bunch well. In this way, we avoid the problem of a large number of empty boxes, which negatively affects the performance of the FMM method.

Although the strategy is described as above, in practice we do not need to check the centers of all child boxes for each cutting. What we really need to know is how many times

Figure 2.7: The box cutting strategy

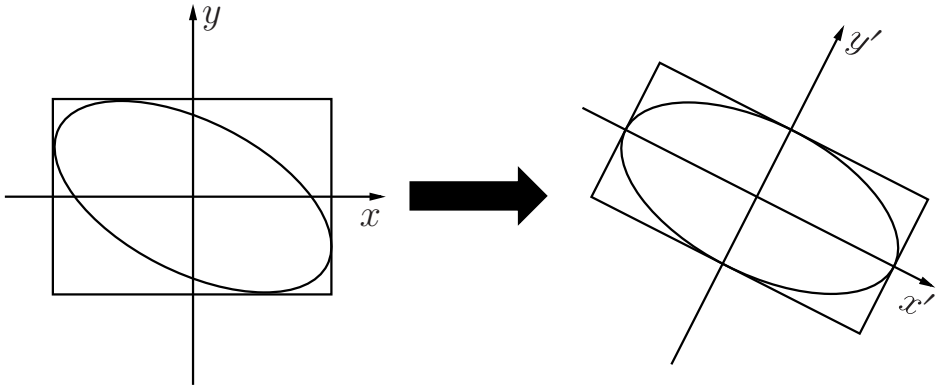


we need to cut each dimension for each level. For this purpose, we can simply compare the size of each dimension to a scale. For the first level the scale is set to be the size of the largest dimension, and for each finer level the scale is divided by two. Each dimension that is larger than half of the scale will be cut. If a dimension is cut in some level, it will definitely be cut in all the finer levels. We label each box by four numbers (l, n_x, n_y, n_z). l is the level, and n_i is the index of box in the i^{th} direction. $n_i = 0$ if the i^{th} direction is not cut in the l^{th} level, or $n_i \in [1, 2^{l_i}]$ if the i^{th} direction is cut l_i times in the l^{th} level. Assuming the zero level box is centered at the origin $(0, 0, 0)$, the center position of each box can be simply calculated from its label as

$$c_{ln_i} = s_l \cdot n_i + 0.5 \cdot s_l \cdot (1 - 2^{l_i}) \tag{2.19}$$

where c_{ln_i} is the center position in the i^{th} dimension of the $\vec{n}^{\text{th}}$ box of the l^{th} level, and s_l is the box size of the l^{th} level boxes. The center positions are needed when we translate or convert the far multipole expansions and the local expansions.

Figure 2.8: Frame rotation



2.4.2 Frame rotation

Another method to avoid empty boxes is to set the frame consistent with the principal axes of the bunch as shown in Figure 2.8 for a two dimensional case. We start from a frame whose origin is the center of mass of the bunch. The position of the i^{th} particle is $\mathbf{X_i} = (x_{i1}, x_{i2}, x_{i3})$. If the coordinate axes are denoted by x_j, $j = 1, 2, 3$, then the moment of inertia coefficient matrix element I_{jk} can be written as

$$I_{jk} = \sum_i m_i (r_i^2 \delta_{jk} - x_{ij} x_{ik}), \tag{2.20}$$

where i is the index of the particle and $r_i^2 = x_{i1}^2 + x_{i2}^2 + x_{i3}^2$. In the principal axes frame, only the diagonal elements of the matrix $\mathbf{I}$ are nonzero. Assuming a matrix $\mathbf{P}$ that diagonalizes $\mathbf{I}$ as $\mathbf{P}^{-1}\mathbf{I}\mathbf{P} = \mathbf{I}' = \text{diag}\{I'_{11}, I'_{22}, I'_{33}\}$, the position of the particle in the principal axes frame is $\mathbf{X}' = \mathbf{P}^{-1}\mathbf{X}$. If the electric field in the principal axes frame is $\mathbf{E}'$, the field in the original frame is $\mathbf{E} = \mathbf{P}\mathbf{E}'$.

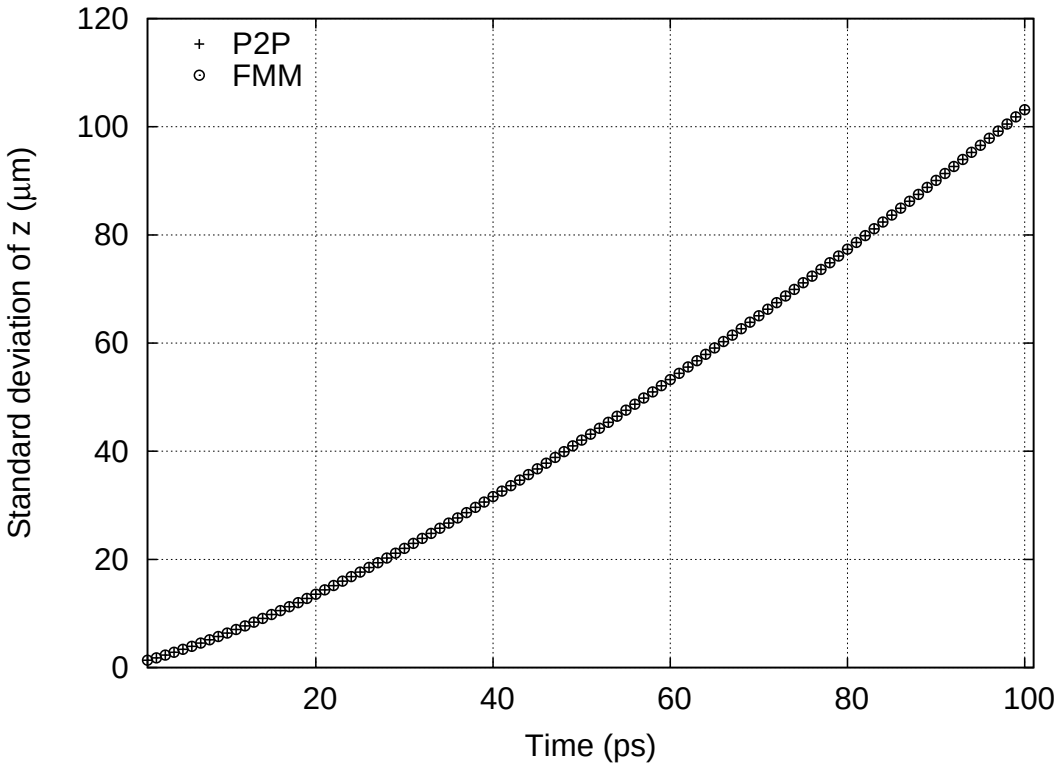
So when we have a bunch in the lab frame, we first shift the origin of the frame to the center of mass of the bunch, and rotate the frame so that it is consistent with the principal

axes of the bunch. We then cut the boxes as stated in the previous subsection (2.4.1) and calculate the potential or/and the field. Finally we translate the field back to the lab frame.

This approach automatically determines an initial enclosing box of smallest possible volume.

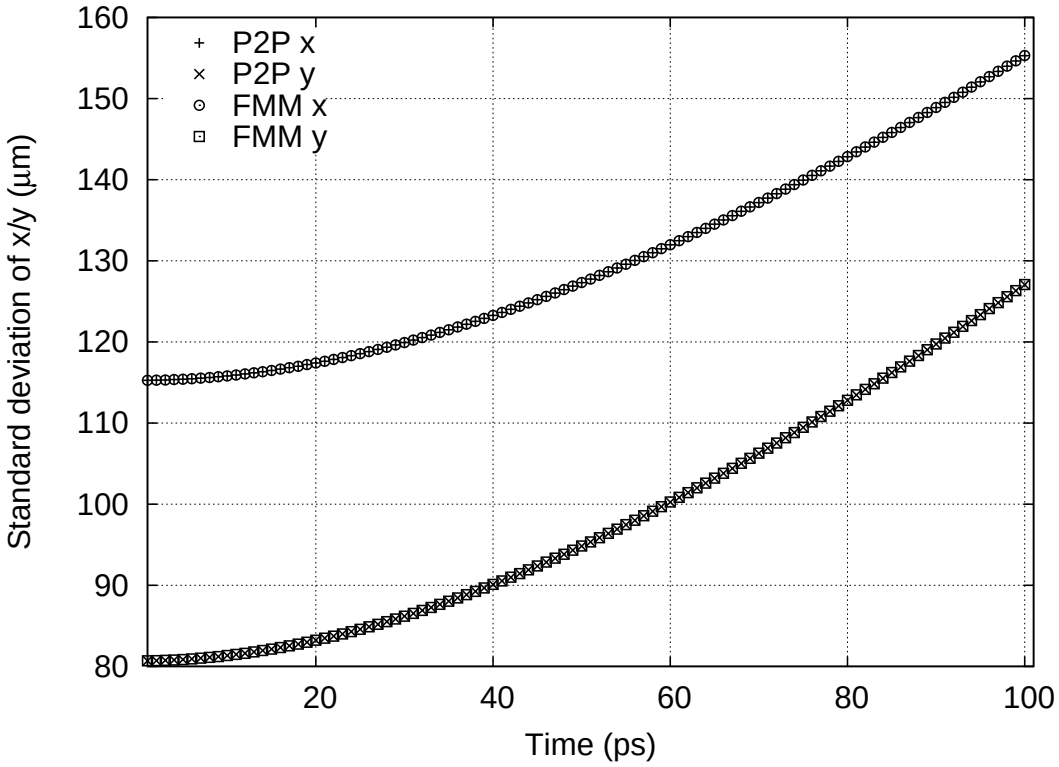
2.5 Examples of Tracking Simulations

Figure 2.9: Longitudinal bunch size during the free expansion of an electron bunch



We used the FMM in tracking simulations, and compared the results with those obtained by the much more expensive use of a pairwise Coulomb formula. One example is shown in Figure 2.9 and Figure 2.10. We have a bunch of 2,082,000 electrons with a three dimensional Gaussian distribution. The initial size of the bunch is $(115\mu\text{m}, 81\mu\text{m}, 1\mu\text{m})$. We use 100,000 macroparticles, each of which represents 20.82 electrons, and a fourth order Runge-Kutta

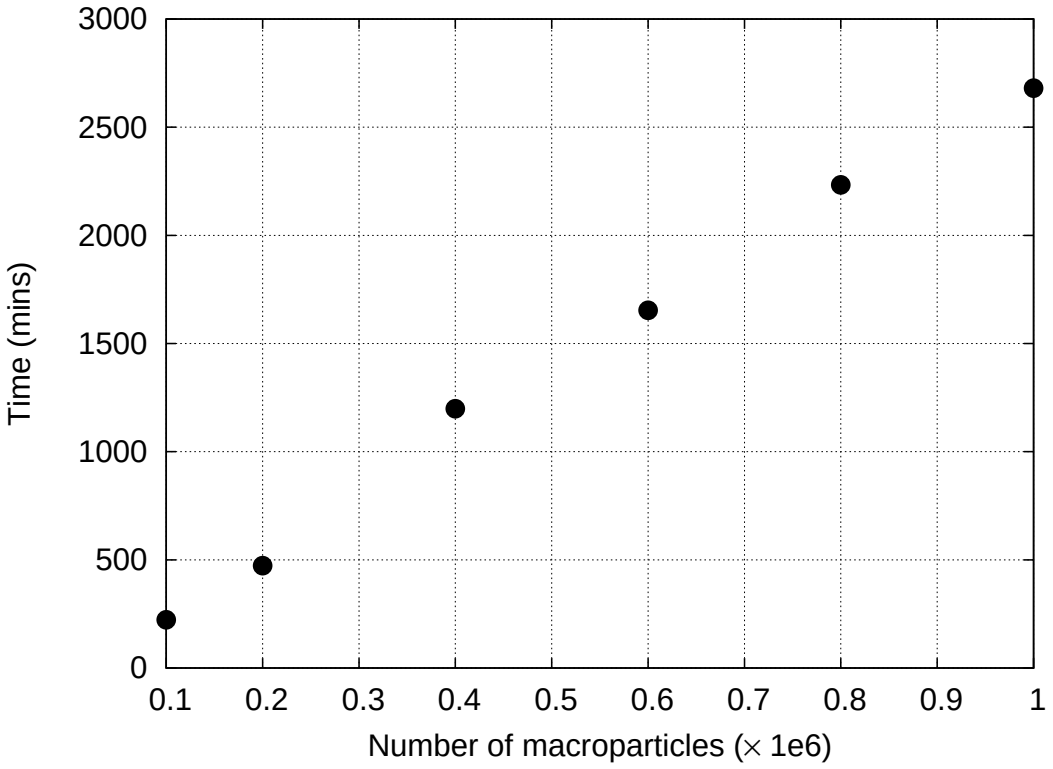
Figure 2.10: Transverse bunch size during the free expansion of an electron bunch



integrator with fixed step size to simulate the free expansion of the bunch without any external field. The step size is 1ps, and the simulation runs for 100 steps to 100ps. The results of the FMM with a fifth order DA expansion are presented in dots, and the results of the pairwise Coulomb formula are presented in lines. The evolution of the bunch size with respect to time in z direction is shown on the left, and those in x and y directions are shown on the right. The final bunch sizes are $(155.21\mu\text{m}, 127.08\mu\text{m}, 103.15\mu\text{m})$ and $(155.29\mu\text{m}, 127.06\mu\text{m}, 103.13\mu\text{m})$ respectively. The relative difference is less than 0.0515%.

Figure 2.11 shows the computational expense for the simulations of the free expansion of a proton bunch. The bunch has 2,082,000 protons with a uniform distribution in all dimensions, and its initial size is $(66.45\mu\text{m}, 46.76\mu\text{m}, 0.58\mu\text{m})$. We use a fourth order Runge-Kutta integrator with fixed step size 1ps and simulate to 100ps on our computer with 4 Core (8

Figure 2.11: Computational expense for the simulations for different numbers of macroparticles



Hyperthreaded) Intel Xeon Processor X5677 running at approximately 3.5GHz. Our simulation program is a single process program, although the FMM can be parallelized in principle. The final bunch size is $(66.52\mu\text{m}, 46.84\mu\text{m}, 29.57\mu\text{m})$. The macroparticle numbers range from 100,000 to 1,000,000, and the computational expenses for the simulations with different numbers of macroparticles are presented in Figure 2.11. Apparently the computational expense does indeed increase linearly with the number of macroparticles, as expected from the above theoretical arguments.

Table 2.4: Computational expenses for the Gaussian/uniform distribution bunches

Distr.	dx (μm)	dy(μm)	dz(μm)	lx	ly	lz	time (min)
GS	99.91	99.95	100.03	5	5	5	16.10
U	99.91	99.96	100.00	5	5	5	5.93
GS	1.00	99.95	100.03	0	6	6	10.64
U	1.00	99.96	100.00	0	6	6	3.00
GS	1.00	1.00	100.03	3	3	9	10.44
U	1.00	1.00	100.00	3	3	9	6.53

2.6 Discussion Concerning the Efficiency

We have shown that the FMM can be used in the tracking simulation, and its computational expense scales with the macroparticle number. However, we also notice that the efficiency of treating a bunch with the Gaussian distribution is worse than that of a bunch with a uniform distribution when they have the same number of macroparticles and similar shapes. Some examples are presented in Table 2.4. We calculated the electric field for a bunch of 1,000,000 electrons with either the Gaussian distribution or the uniform distribution of varying bunch shapes. (dx, dy, dz) is the r.m.s. size of the bunch, and (lx, ly, lz) is how many times the bunch is divided in (x, y, z) direction. The single process program runs on our computer with 4 Core (8 Hyperthreaded) Intel Xeon Processor X5677 running at 3.5 GHz and uses only one core in the computation. We can see the computational expense of a bunch with a uniform distribution is much less than that of a bunch with the Gaussian distribution of similar shape. This is because the FMM has the best efficiency when all of the boxes of the finest level have the same number of particles inside. On the other hand, when a bunch has a Gaussian distribution, its center has a much higher charge density than its edge, which results in the boxes at the edge being either empty or enclosing few particles and the boxes at the center enclosing many more particles than expected. To solve this problem, we need to use the adaptive FMM, also called multiple level fast multipole algorithm (MLFMA),

by which we cut the boxes into finer levels where the charge density is higher, so that we can make sure all the boxes of the finest levels (different lowest levels at different positions) enclose similar numbers of particles, which guarantees the efficiency [19, 20].

Chapter 3

Multiple Level Fast Multipole Algorithm in the Differential Algebra Framework

The single level FMM cuts the whole domain of charges into the same level boxes everywhere. It works well when the charged particle distribution is uniform or close to it. However in many cases the charged particles have a more complicated distribution. The loss of ability to deal with heavily non-uniform distribution limits the use of the single level FMM. For example, a Gaussian distribution has much higher charge density at the center than at the edge. In this case, if we cut the whole domain to a very deep level so that the boxes at the center enclose a small number of particles, we will end up with many empty boxes or boxes with few particles at the edge, which reduces the calculation speed. A deeper level also leads to a larger number of boxes and thus a larger amount of memory is needed, because of which the program may fail due to lack of memory. If we do not cut to such a deep level, the boxes at the center will enclose many more particles than they should, and this will also significantly lower the efficiency. In some cases the charge distribution may be more complicated. For example, an FFAG accelerator has several bunches distributed along a helix. The single level FMM has difficulties to deal with such kind of complicated distributions. However,

the multiple level fast multipole algorithm (MLFMA) can treat these complicated charge distributions of N particles with an efficiency of $O(N)$, by cutting the whole domain into different levels at different places according to the charge density there, which means higher level boxes are created where higher density is and lower level boxes are created where lower density is. So one can make sure the number of particles inside the boxes in the lowest levels at different places are close to each other, which improves the efficiency.[19, 20] To expand the usage of the FMM in the beam dynamics simulation, we developed the MLFMA in the DA framework, which we are going to present in the following.

3.1 Analytical Tools

3.1.1 The Far Multipole Expansion from the Charges

Suppose that n particles with charge q_i located at $\vec{r}_i(x_i, y_i, z_i)$ with $r_i < r_0$, the electrostatic potential at a point $\vec{r}(x, y, z)$ with $r > r_0$ can be expressed as

$$\begin{aligned}
\phi &= \sum_{i=1}^{n} \frac{q_i}{\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}} \\
&= \sum_{i=1}^{n} \frac{q_i/\sqrt{x^2+y^2+z^2}}{\sqrt{1+\frac{x_i^2+y_i^2+z_i^2}{x^2+y^2+z^2}-\frac{2x_ix}{x^2+y^2+z^2}-\frac{2y_iy}{x^2+y^2+z^2}-\frac{2z_iz}{x^2+y^2+z^2}}} \\
&= \sum_{i=1}^{n} \frac{d_r \cdot q_i}{\sqrt{1+(x_i^2+y_i^2+z_i^2)d_r^2-2x_id_x-2y_id_y-2z_id_z}} \\
&= d_r \cdot \bar{\phi}_{\mathrm{M}}
\end{aligned} \tag{3.1}$$

with

$$\begin{aligned}
&d_x = \frac{x}{x^2+y^2+z^2}, \quad d_y = \frac{y}{x^2+y^2+z^2}, \\
&d_z = \frac{z}{x^2+y^2+z^2}, \quad d_r = \sqrt{d_x^2+d_y^2+d_z^2}, \\
&\bar{\phi}_{\mathrm{M}} = \sum_{i=1}^{n} \left\{ q_i/\sqrt{1+(x_i^2+y_i^2+z_i^2)d_r^2-2x_id_x-2y_id_y-2z_id_z} \right\}.
\end{aligned}$$

If we choose d_x, d_y, and d_z as DA variables, $\bar{\phi}_{\mathrm{M}}$ can be expressed as a DA vector, which can be considered as the Taylor expansion of $\bar{\phi}_{\mathrm{M}}$ with d_x, d_y, and d_z at infinity. d_r does not exist in the DA framework, because d_x, d_y, and d_z are all infinitesimal, so is the sum of their squares, and the square root operation on an infinitesimal is not well defined[14]. So we can not express ϕ in the DA framework. But it is enough to have the expansion of $\bar{\phi}_{\mathrm{M}}$ in the DA framework and keep d_r in mind. We will explain how it works in the following.

3.1.2 Translation of The Far Multipole Expansion

The potential ϕ at (x, y, z) of a far multipole expansion at the origin $(0, 0, 0)$ can be translated into another far multipole expansion at the point (x'_o, y'_o, z'_o). The new DA variables can be chosen as

$$\begin{aligned} d'_x &= \frac{x - x'_o}{r'^2} = \frac{x'}{r'^2}, \\ d'_y &= \frac{y - y'_o}{r'^2} = \frac{y'}{r'^2}, \\ d'_z &= \frac{z - z'_o}{r'^2} = \frac{z'}{r'^2}, \end{aligned}$$

with $r' = \sqrt{x'^2 + y'^2 + z'^2}$. With some direct algebra, one can find the relation between the new DA variables and the old DA variables as shown in Eq. (3.2).

$$\begin{aligned} d_x &= (d'_x + x'_o \cdot (d'^2_x + d'^2_y + d'^2_z)) \cdot R, \\ d_y &= (d'_y + y'_o \cdot (d'^2_x + d'^2_y + d'^2_z)) \cdot R, \\ d_z &= (d'_z + z'_o \cdot (d'^2_x + d'^2_y + d'^2_z)) \cdot R, \end{aligned} \tag{3.2}$$

with

$$R = \frac{1}{1 + (x'^2_o + y'^2_o + z'^2_o)(d'^2_x + d'^2_y + d'^2_z) + 2x'_o d'_x + 2y'_o d'_y + 2z'_o d'_z}.$$

Eq. (3.2) can be considered as the map between the new DA variables and the old DA variables, which we refer to M_1. If we translate $\bar{\phi}_{\mathrm{M}}$ in the old frame into $\tilde{\phi}_{\mathrm{M}}$ in the new frame by

$$\tilde{\phi}_{\mathrm{M}} = \bar{\phi}_{\mathrm{M}} \circ M_1,$$

where $\circ$ means the composition of two maps, the potential in the new frame can be written as

$$\phi' = \tilde{\phi}_{\mathrm{M}} \cdot d'_r \cdot \sqrt{R} = d'_r \cdot \bar{\phi}'_{\mathrm{M}}, \tag{3.3}$$

knowing that d_r in the old frame can be translated into the new frame as

$$\begin{aligned}
d_r &= \frac{1}{\sqrt{(x - x'_o + x'_o)^2 + (y - y'_o + y'_o)^2 + (z - z'_o + z'_o)^2}} \\
&= \frac{1}{\sqrt{\begin{array}{c}(x - x'_o)^2 + (y - y'_o)^2 + (z - z_o)^2 + x'^2_o + y'^2_o + z'^2_o \\ +2(x - x'_o)x'_o + 2(y - y'_o)y'_o + 2(z - z'_o)z'_o\end{array}}} \\
&= \frac{1}{\sqrt{1/d'^2_r + x'^2_o + y'^2_o + z'^2_o + 2x'_o d'_x/d'^2_r + 2y'_o d'_y/d'^2_r + 2z'_o d'_z/d'^2_r}} \\
&= \frac{d'_r}{\sqrt{1 + (x'^2_o + y'^2_o + z'^2_o)d'^2_r + 2x'_o d'_x + 2y'_o d'_y + 2z'_o d'_z}} \\
&= d'_r \cdot \sqrt{R},
\end{aligned}$$

with

$$d'_r = \sqrt{d'^2_x + d'^2_y + d'^2_z}.$$

Same as d_r, d'_r does not exist in the DA framework, but for now we only need to calculate $\bar{\phi}'_{\mathrm{M}}$ and keep the existence of d'_r in mind.

3.1.3 Convert The Far Multipole Expansion into a Local Expansion

Given a far multipole expansion at the origin $(0, 0, 0)$, a local expansion at (x'_o, y'_o, z'_o), which creates the same potential on the observers, can be found. It is called "local" because

(x'_o, y'_o, z'_o) is close to the observer (x, y, z). So it is natural to choose the new DA variables as

$$\begin{aligned} d'_x &= x - x'_o = x', \\ d'_y &= y - y'_o = y', \\ d'_z &= z - z'_o = z'. \end{aligned} \tag{3.4}$$

(x', y', z') are the new coordinates of the observer (x, y, z) if we shift the origin to (x'_o, y'_o, z'_o). The old and the new DA variables have the relation as Eq. (3.5) shows

$$\begin{aligned} d_x &= \frac{x}{x^2+y^2+z^2} = \frac{x'_o + d'_x}{(x'_o + d'_x)^2 + (y'_o + d'_y)^2 + (z'_o + d'_z)^2} = (x'_o + d'_x) \cdot R, \\ d_y &= \frac{y}{x^2+y^2+z^2} = \frac{y'_o + d'_y}{(x'_o + d'_x)^2 + (y'_o + d'_y)^2 + (z'_o + d'_z)^2} = (y'_o + d'_y) \cdot R, \\ d_z &= \frac{z}{x^2+y^2+z^2} = \frac{z'_o + d'_z}{(x'_o + d'_x)^2 + (y'_o + d'_y)^2 + (z'_o + d'_z)^2} = (z'_o + d'_z) \cdot R. \end{aligned}$$

with

$$R = \frac{1}{(x'_o + d'_x)^2 + (y'_o + d'_y)^2 + (z'_o + d'_z)^2}. \tag{3.5}$$

We note Eq. (3.5) as M_2. To convert the far multipole expansion in Eq. (3.3) into a local expansion, we need to work on the two parts separately. $\bar{\phi}'_{\mathrm{M}}$ can be converted into $\tilde{\phi}_{\mathrm{L}}$ in the new frame as

$$\tilde{\phi}_{\mathrm{L}} = \bar{\phi}'_{\mathrm{M}} \circ M_2,$$

and d'_r can be converted into $\sqrt{R}$ in the new frame. Therefore we have the local expansion as

$$\phi_{\mathrm{L}} = \tilde{\phi}_{\mathrm{M}} \cdot \sqrt{R}. \tag{3.6}$$

Note that by now we have gotten the expansion of the potential, not just a component of it, as a DA vector.

3.1.4 The Local Expansion from the Charges

Consider an observer point $\vec{r}(x, y, z)$ within a spherical region of the radius r_0 centered at (x'_o, y'_o, z'_o), the electrostatic potential on the observer from n particles outside the spherical region can be expressed as a local expansion at (x'_o, y'_o, z'_o). We choose the DA variables as

$$\begin{aligned} d'_x &= x - x'_o = x', \\ d'_y &= y - y'_o = y', \\ d'_z &= z - z'_o = z'. \end{aligned} \tag{3.7}$$

Assuming the i^{th} source particle has charge q_i and located at $\vec{r_i}(x_i, y_i, z_i)$, the local expansion of the potential is

$$\begin{aligned} \phi_{\text{L}} &= \sum_{i=1}^{n} \frac{q_i}{\sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}} \\ &= \sum_{i=1}^{n} \frac{q_i}{\sqrt{(x'_o - x_i + d'_x)^2 + (y'_o - y_i + d'_y)^2 + (z'_o - z_i + d'_z)^2}} \end{aligned} \tag{3.8}$$

3.1.5 Translate The Local Expansion

A local expansion at the origin $(0, 0, 0)$ can be translated to (x'_o, y'_o, z'_o), assuming both points are close to the observer (x, y, z). Choosing the new DA variable as the same as Eq. (3.7),

the relation between the old and the new DA variables is just a linear shift.

$$\begin{aligned} d_x &= x'_o + d'_x, \\ d_y &= y'_o + d'_y, \\ d_z &= z'_o + d'_z. \end{aligned} \tag{3.9}$$

We call eq. (3.9) the map M_3, then the new local expansion can be calculated by

$$\phi'_{\mathrm{L}} = \phi_{\mathrm{L}} \circ M_3. \tag{3.10}$$

3.1.6 Calculate the Potential and the Field from the Expansions

It is straightforward to calculate the potential from a local expansion or a multipole expansion. Since we have the potential expressed as an p^{th} order polynomial, we only need to plug in the value of (d_x, d_y, d_z) for each particle to obtain the potential on it.

The local expansion of the potential is a polynomial of the observer's coordinates. Taking the derivative of the potential with respect to a coordinate, one can obtain the $(p-1)^{\mathrm{th}}$ order polynomial of the field on the respective direction.

To calculate the field by the multipole expansion is more complicated, because the DA variables are not the coordinates, but some analytical functions of the coordinates. We need to take the derivative with respect to the coordinates by the chain rule. After some direct

algebra, we obtain the expression of the field as

$$\begin{aligned} E_x &= \{-\frac{\partial\bar{\phi}}{\partial d_x}\cdot(d_r^2-2d_x^2)+2\frac{\partial\bar{\phi}}{\partial d_y}\cdot d_x d_y+2\frac{\partial\bar{\phi}}{\partial d_z}\cdot d_x d_z+\bar{\phi}\cdot d_x\}\cdot d_r \\ E_y &= \{2\frac{\partial\bar{\phi}}{\partial d_x}\cdot d_y d_x-\frac{\partial\bar{\phi}}{\partial d_y}(d_r^2-2d_y^2)+2\frac{\partial\bar{\phi}}{\partial d_z}\cdot d_y d_z+\bar{\phi}\cdot d_y\}\cdot d_r \\ E_z &= \{2\frac{\partial\bar{\phi}}{\partial d_x}\cdot d_z d_x+2\frac{\partial\bar{\phi}}{\partial d_y}\cdot d_z d_y-\frac{\partial\bar{\phi}}{\partial d_z}\cdot(d_r^2-d_z^2)+\bar{\phi}\cdot d_z\}\cdot d_r \end{aligned}$$

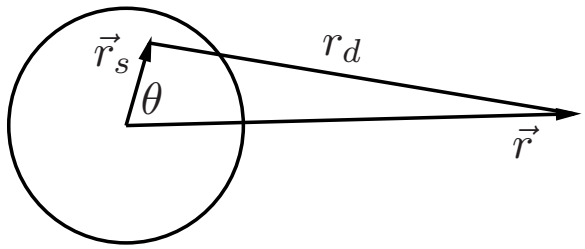
with (3.11)

$$d_r = \sqrt{d_x^2+d_y^2+d_z^2}.$$

When we calculate the field using Eq. (3.11), the part inside the brackets and d_r needs to be treated separately, because d_r does not exist in the DA framework. Note that d_r^2 can be expressed as a DA vector, so the part inside the brackets can be calculated in the DA framework. We first calculate the value of it. Then we calculate d_r and the multiplication for each particle.

3.2 Error Analysis

Figure 3.1: Far multipole expansion of the potential of a point charge



In order to give an error estimation of the far multipole expansion of the charges, let us first consider a source charge inside a spherical region of radius a as shown in Figure 3.1.

Without loss of the generality, we assume the center of the sphere is at the origin. The source charge is at the position $\vec{r}_s(x_s, y_s, z_s)$, the observer point is $\vec{r}(x, y, z)$, the distance between them is $r_d = |\vec{r}_s - \vec{r}|$ and the angle between $\vec{r}$ and $\vec{r}_s$ is θ. The potential on the observer can be expressed as a far multipole expansion as shown in Eq. (3.1).

$1/r_d$ can be expressed as an Taylor expansion shown in Eq. (3.12),

$$\frac{1}{r_d} = \frac{1}{|\vec{r} - \vec{r}_s|} = \sum_{n=0}^{\infty} \frac{1}{n!} \cdot (-1)^n \cdot \vec{r}_s^{\,n} \cdot n \cdot \nabla^n \frac{1}{r}. \tag{3.12}$$

In the above $\cdot n \cdot$ is the n-fold tensor contraction operation, which is defined as [7]

$$\vec{A}^n \cdot n \cdot \vec{B}^n = A^n_{\alpha 1 \ldots \alpha n} B^n_{\alpha n \ldots \alpha 1}$$

where the convention of implied summation over repeated Greek subscripts is followed. The gradients of $1/r$ can be calculated as Eq. (3.13) shows according to the Maxwell Cartesian tensors[7, 80],

$$\begin{aligned} \partial_\alpha^{n1} \partial_\beta^{n2} \partial_\gamma^{n3} \left(\frac{1}{r}\right) &= (-1)^n r^{-2n} \sum_{m_1=0}^{\mathrm{F}(n_1)} \sum_{m_2=0}^{\mathrm{F}(n_2)} \sum_{m_3=0}^{\mathrm{F}(n_3)} (-1)^m \begin{bmatrix} n_1 \\ m_1 \end{bmatrix} \begin{bmatrix} n_2 \\ m_2 \end{bmatrix} \begin{bmatrix} n_3 \\ m_3 \end{bmatrix} \\ &\quad \times r^{2m} f(n-m-1) x^{n_1 - 2m_1} y^{n_2 - 2m_2} z^{n_3 - 2m_3} \end{aligned} \tag{3.13}$$

with $n = n_1 + n_2 + n_3$, $m = m_1 + m_2 + m_3$, $f(v) = 1 \times 3 \times 5 \times \ldots \times (2v+1)$, $\mathrm{F}(x) = \lfloor \frac{x}{2} \rfloor$ and

$$\begin{bmatrix} n \\ m \end{bmatrix} = \frac{n!}{2^m m! (n-2m)!}.$$

To compare with Eq. (3.12), we can also expand $1/r_d$ in powers of r using the Legendre Polynomials[94, 6].

$$\frac{1}{r_d} = \begin{cases} \sum_{n=0}^{\infty} \frac{r_s^n}{r^{n+1}} P_n(\cos\theta) & \text{if } \left|\frac{r_s}{r}\right| < 1 \\ \sum_{n=0}^{\infty} \frac{r^n}{r_s^{n+1}} P_n(\cos\theta) & \text{if } \left|\frac{r_s}{r}\right| > 1 \end{cases} \tag{3.14}$$

where $P_n(\cos\theta)$ are Legendre Polynomials. Comparing Eq. (3.12) with Eq. (3.14), one can see they have a term-by-term correspondence, which indicates

$$P_n(\cos\theta) = \frac{(-1)^n}{n!} \cdot r^{n+1} \cdot \left(\frac{\vec{r_s}}{r_s}\right)^n \cdot n \cdot \nabla^n \frac{1}{r}. \tag{3.15}$$

Notice that the Legendre Polynomials $P_n(\cos\theta)$ only depend on the angle θ.

The Taylor expansion of $\bar{\phi}_{\mathrm{M}}$ in Eq. (3.1) can be expressed explicitly by tensors, such as

$$\begin{aligned} \bar{\phi}_{\mathrm{M}} &= \sum_{i=1}^{N} \frac{q_i}{\sqrt{1 + (x_i^2 + y_i^2 + z_i^2)d_r^2 - 2x_i d_x - 2y_i d_y - 2z_i d_z}} \\ &= \sum_{i=1}^{N} \sum_{n=0}^{\infty} \frac{q_i}{n!} \partial_\alpha^{n1} \partial_\beta^{n2} \partial_\gamma^{n3} \bar{\phi}_{\mathrm{M}}|_{(0,0,0)} \cdot d_x^\alpha d_y^\beta d_z^\gamma \\ &= \sum_{i=1}^{N} \sum_{n=0}^{\infty} \vec{A}_i^n \cdot n \cdot \vec{d}_r^n. \end{aligned} \tag{3.16}$$

where ∂_α,∂_β, and ∂_γ refer to the partial derivatives of d_x, d_y, and d_z, $\vec{d}_r = \vec{r}/r^2$, and

$$\begin{aligned} A^n_{i,n_1 n_2 n_3} &= \frac{q_i}{n!} \partial_\alpha^{n1} \partial_\beta^{n2} \partial_\gamma^{n3} \bar{\phi}_{\mathrm{M}}|_{(0,0,0)} \\ &= \frac{q_i}{n!} \sum_{m_1=0}^{\mathrm{F}(n_1)} \sum_{m_2=0}^{\mathrm{F}(n_2)} \sum_{m_3=0}^{\mathrm{F}(n_3)} (-1)^m \begin{bmatrix} n_1 \\ m_1 \end{bmatrix} \begin{bmatrix} n_2 \\ m_2 \end{bmatrix} \begin{bmatrix} n_3 \\ m_3 \end{bmatrix} \\ &\quad \times r_i^{2m} f(n-m-1) \cdot x_i^{n_1-2m_1} y_i^{n_2-2m_2} z_i^{n_3-2m_3}. \end{aligned} \tag{3.17}$$

Comparing Eq. (3.17) with Eq. (3.13) we obtain

$$\begin{aligned}\bar{\phi}_{\mathrm{M}} &= \sum_{i=1}^{N}\sum_{n=0}^{\infty} q_i \frac{r_i^n}{r^n}\cdot\frac{(-1)^n}{n!}\cdot r_i^{n+1}\cdot\left(\frac{\vec{r}}{r}\right)^n\cdot n\cdot\nabla^n\frac{1}{r_i} \\ &= \sum_{i=1}^{N}\sum_{n=0}^{\infty} q_i\left(\frac{r_i}{r}\right)^n\cdot P_n(\cos\theta_i),\end{aligned} \tag{3.18}$$

where Eq. (3.15) is used to get the result. Hence the far multipole expansion of the potential is

$$\phi = \sum_{i=1}^{N}\sum_{n=0}^{\infty} q_i\cdot\frac{r_i^n}{r^{n+1}}\cdot P_n(\cos\theta_i). \tag{3.19}$$

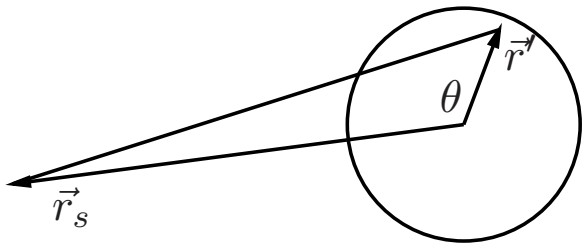
If we keep the first p terms in the expansion, The error can be estimated as follows.

$$\begin{aligned}|\epsilon| &= \left|\sum_{i=1}^{N}\sum_{n=p+1}^{\infty} q_i\cdot\frac{r_i^n}{r^{n+1}}\cdot P_n(\cos\theta_i)\right| \\ &\le \sum_{i=1}^{N}|q_i|\cdot\left|\sum_{n=p+1}^{\infty}\frac{a^n}{r^{n+1}}\right| \\ &= C\cdot\frac{a^{p+1}}{r^{p+2}}\cdot\left|\sum_{n=0}^{\infty}\left(\frac{a}{r}\right)^n\right| \\ &= C\cdot\frac{a^{p+1}}{r^{p+2}}\cdot\frac{1}{1-\frac{a}{r}} \\ &= C\cdot\left(\frac{a}{r}\right)^{p+1}\cdot\frac{1}{r-a},\end{aligned} \tag{3.20}$$

where $C=\sum_{i=1}^{N}|q_i|$ and $r_i\le a$ for any i.

When we translate a far multipole expansion from a child box's center to its parent box's center, because the transfer map between the DA variables in different frame does not contain a constant part, as shown in Eq. (3.2), the polynomial we obtain in the parent box frame should be the same with the one calculated directly from the charges in the parent

Figure 3.2: Local expansion of the potential of a point charge



box frame by Eq. (3.1), so that the error has the same expression with Eq. (3.20) with r and a defined in the parent box frame.

The local expansion from the charges is calculated as Eq. (3.8), which can be written as Eq. (3.21) using Eq. (3.14).

$$\begin{aligned}
\phi_{\mathrm{L}} &= \sum_{i=1}^{N} \frac{q_i}{\sqrt{(x'_o - x_i + d_x)^2 + (y'_o - y_i + d_y)^2 + (z'_o - z_i + d_z)^2}} \\
&= \sum_{i=1}^{N} \frac{q_i}{\left|\vec{r}_{s,i} - \vec{r}'\right|} \\
&= \sum_{i=1}^{N} \sum_{n=0}^{\infty} q_i \cdot \frac{r'^n}{r_{s,i}^{n+1}} \cdot P_n(\cos\theta_i),
\end{aligned} \tag{3.21}$$

where $\vec{r}_s$ is the position of a source charge in the observer's local frame with $\vec{r}_{s,i} = \vec{r}_i - \vec{r}'_o$, $\vec{r}'$ is the position of the observer in its local frame, and θ is the angle between $\vec{r}_s$ and $\vec{r}'$ as

shown in Figure 3.2. If we keep the first p terms, then

$$\begin{aligned}
|\epsilon| &= \left| \sum_{i=1}^{N} \sum_{n=p+1}^{\infty} q_i \cdot \frac{r'^n}{r_{s,i}^{n+1}} \cdot P_n(\cos\theta_i) \right| \\
&\leq \sum_{i=1}^{N} |q_i| \cdot \left| \sum_{n=p+1}^{\infty} \frac{r'^n}{b^{n+1}} \right| \\
&= C \cdot \frac{r'^{p+1}}{b^{p+2}} \cdot \left| \sum_{n=0}^{\infty} \left(\frac{r'}{b}\right)^n \right| \\
&= C \cdot \frac{r'^{p+1}}{b^{p+2}} \cdot \frac{1}{1-\frac{r'}{b}} \\
&= C \cdot \left(\frac{r'}{b}\right)^{p+1} \cdot \frac{1}{b-r'},
\end{aligned} \tag{3.22}$$

where $C = \sum_{i=1}^{N} |q_i|$ and $r_{s,i} \geq b$ for any i.

When we convert the far multipole expansion into a local expansion, the polynomial we obtain for the local expansion is different with the one that is calculated directly from the charges by Eq. (3.8). That's because the transfer map between the DA variables contain a constant term, so that the truncated terms of $p+1$ or higher order contribute to the constant term of the local expansion, which is missing. From Eq. (3.5), we can see the constant term of the transfer map is $\vec{r}'_o/r'^2_o$ where $\vec{r}'_o$ is the position of the local frame's origin in the far multipole frame, hence the missing terms are

$$\text{Const}_{\text{missing}} = \sum_{i=1}^{N} \sum_{n=p+1}^{\infty} \vec{A}_i^n \cdot n \cdot \left(\frac{\vec{r}'_o}{r'^2_o}\right)^n, \tag{3.23}$$

where $\vec{A}_i^n$ is defined in Eq. (3.17). Similar to Eq. (3.20), the error of the missing constant part is

$$\left|\epsilon'\right| \leq C \cdot \left(\frac{a}{r'_o}\right)^{p+1} \cdot \frac{1}{r'_o - a}, \tag{3.24}$$

where $C = \sum_{i=1}^{N} |q_i|$ and $r_i \leq a$ for any i. Then the error of the local expansion converted from a far multipole expansion is

$$|\epsilon| \leq C \cdot \left(\frac{a}{r'_o}\right)^{p+1} \cdot \frac{1}{r'_o - a} + C \cdot \left(\frac{r'}{b}\right)^{p+1} \cdot \frac{1}{b - r'}. \tag{3.25}$$

In the translation of the local expansion of a parent box into its child boxes, since the transfer map is linear, there is no truncation error created in this process.

3.3 The Multiple Level Fast Multipole Algorithm

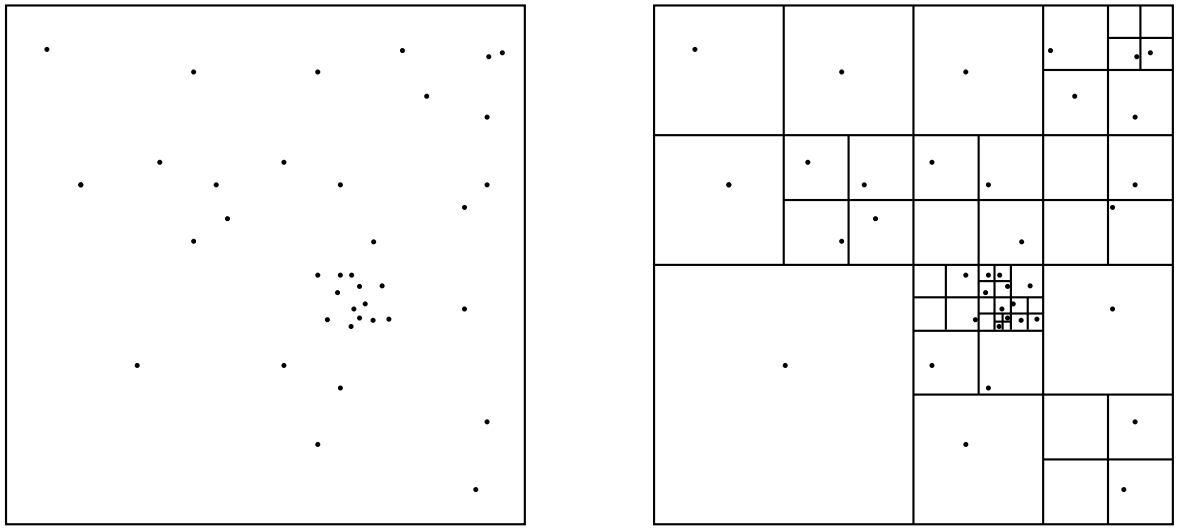
3.3.1 Strategy of the MLFMA

In 1988 Carrier, Greengard and Rokhlin introduced the adaptive FMM [19] , which is often referred as the MLFMA. The general strategy of the MLFMA is similar to the uniform level FMM. In order to evaluate the electrostatic potential and/or field due to an arbitrary distribution of charges, we divide the charges into groups and evaluate the interactions between the groups far away enough by multipole expansions and calculate the interactions between nearby particles directly.

To be more specific, consider a two dimensional example as shown in Figure 3.3. We want to calculate the interaction of a bunch of charges with non-uniform distribution. Without losing generality, we can enclose the charges with a square box, which we call the level zero box. We select a number s. If the number of charges inside the box is larger than s, the box is a parent box. It will be cut from the center into four equal square boxes, which are its child boxes. All the child boxes form a new level. If a child box contains more than s charges, it will be cut again and will have its child boxes. If the parent box is in level l, the

child boxes are in level $l+1$. This process results in the hierarchical structure of the boxes that refine the level zero box into smaller and smaller regions. Different with the uniform level FMM in [93], we do not use the same number of levels for the whole domain. For each level of refinement, we only subdivide the boxes that contains more than s charges. Generally where the higher charge density is, it is refined with the higher level boxes, as shown in Figure 3.3, where $s = 1$. At each level of refinement, we only keep the nonempty boxes. The empty boxes are discarded and completely ignored by the subsequent processes. The three dimensional case can be treated in the same way; the only difference is the cube box has eight child boxes.

Figure 3.3: The hierarchical structure of the boxes



3.3.2 Notation

We choose our notation following reference [19] as shown below. Note that all the boxes here are nonempty boxes.

A *child box* is a nonempty box resulting from the division of a larger box.

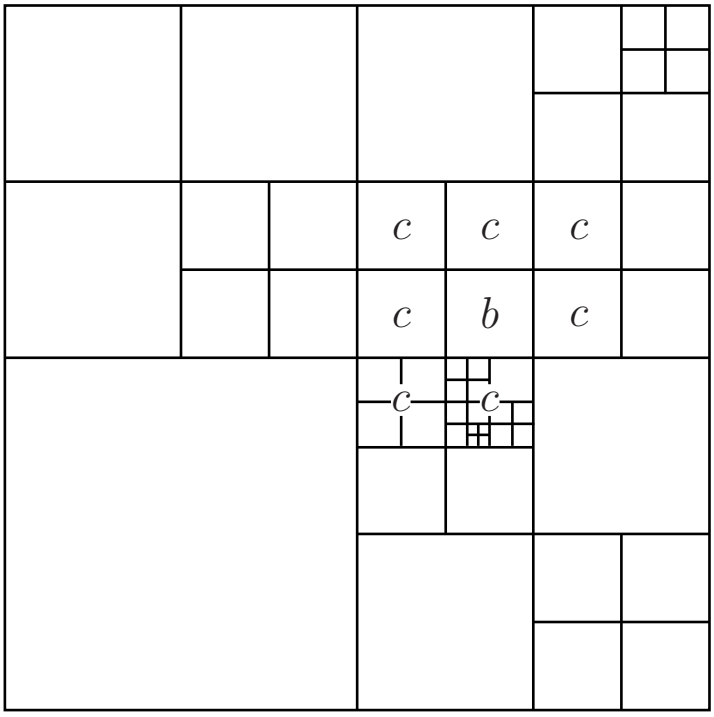
If a box has one or more child boxes, it is called a *parent box*. Otherwise it is called a *childless box*.

The *ancestor boxes* of box b include b's parent box, the parent box of the parent box, and so on till the root box.

If two boxes share at least one vertex, they are called adjacent. Otherwise they are well separated.

Colleagues are adjacent boxes of the same size (at the same level). A given box has at most 27 colleagues including itself. A two dimensional example is given in Figure 3.4.

Figure 3.4: Box (b) and its colleagues (c)



Because of the hierarchical structure, any box b only directly interacts with the boxes who are colleagues of b's parent box and their descents, and b inherits the contribution from other boxes from its parent box. There are four kinds of relations between b and the boxes who directly interacts with it. Based on their sizes and positions, which determine the

Figure 3.5: Case 1. Adjacent boxes

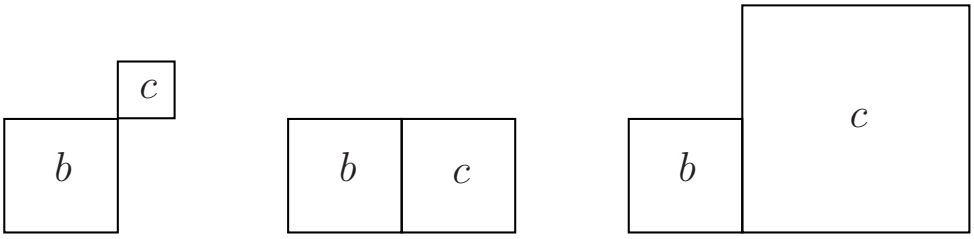
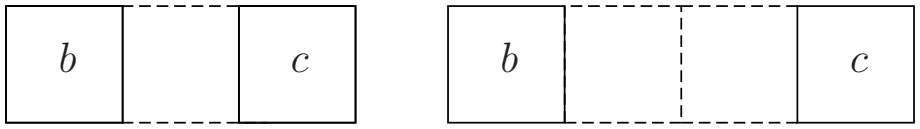


Figure 3.6: Case 2. Separated boxes of the same size



interactions, we can define four lists for box b as follows:

The first kind of relation is shown in Figure 3.5. Box b and box c touch each other. If both of them are childless, the field between the charged particles in b and c should be calculated directly, regardless of the box sizes of them. If any of them is a parent box, no actions can be taken on this stage. Hence we can define list 1 of the box b, denoted by U_b. If b is a parent box, U_b is empty. If b is childless, any box that is childless and adjacent to b is contained in U_b.

The second kind of relation is shown in Figure 3.6. Box b and box c have the same size. They are separated, and the distance between them is equal to or larger than their side length. In this case, regardless whether they are childless boxes or parent boxes, we can always translate the far multipole expansions of one box into the local expansions in the other box, according to Eq. (3.25) with Eq. (3.20). Box b and the boxes in list 2 of b, denoted by V_b, have this kind of relation. Any box whose parent is a colleague of b's parent and who itself is well separated from b is contained in V_b.

The third kind of relation is shown in Figure 3.7 on the left. Box b and box c are

Figure 3.7: Case 3 and case 4. Separated boxes of different sizes

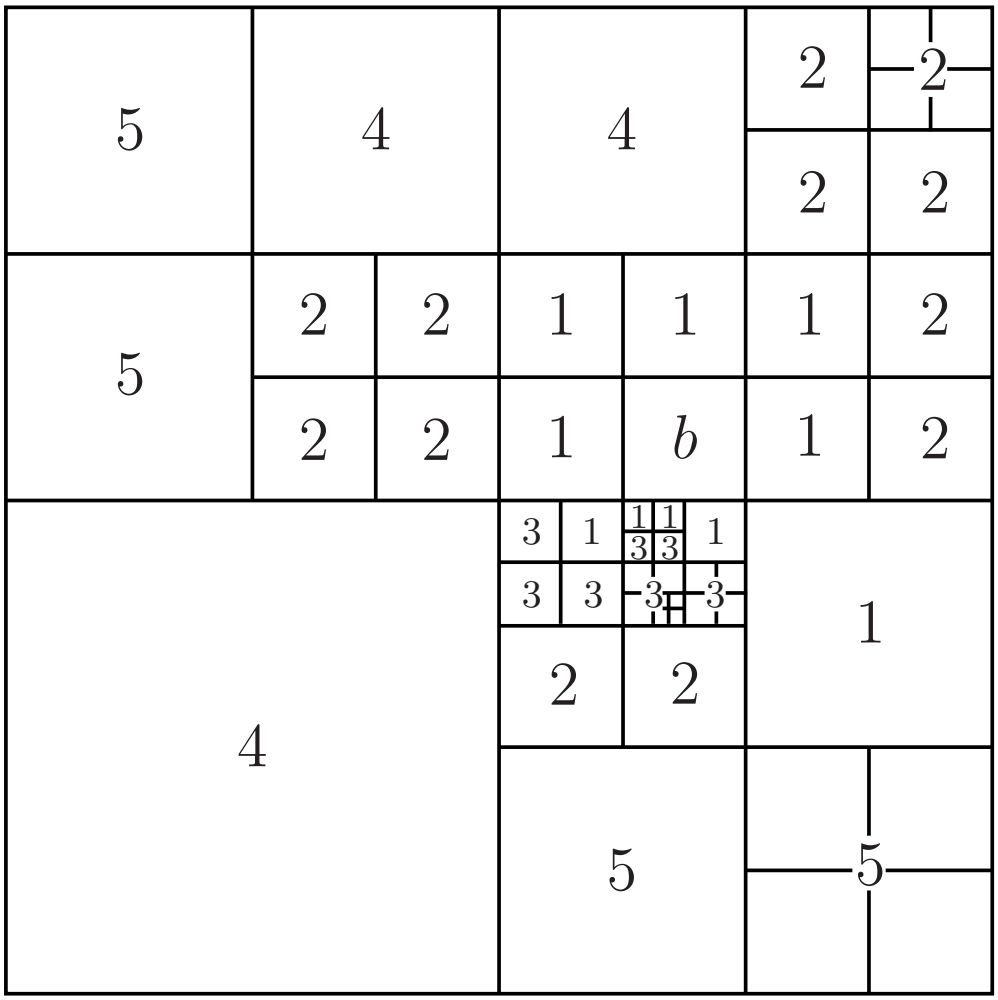
separated, box b is larger than box c, and the distance in between is equal to or greater than the side length of c, but less than the side length of b. If b is childless, because the distance between any particle inside b and the center of c is larger than the side length of c, according to Eq. (3.20) the field on the particles in b due to the particles in c could be calculated by the far multipole expansion of c. For the same reason, according to Eq. (3.22) the field inside c due to the particles in b could represented by a local expansion in c calculated directly from the particles in b using Eq. (3.8). If b is a parent box, no action can be taken on this stage. Hence we can define list 3 of the box b , denoted by W_b. If b is a parent box, W_b is empty. If b is a childless box, any box whose parent is adjacent to b and who itself is not adjacent to b is contained in W_b.

The forth kind of relation is shown in Figure 3.7 on the right. This is the same with the third kind of relation, except that box b and box c are replaced by each other. Hence we can define the List 4 of the box b, denoted by X_b. Any box whose list 3 includes b is contained in X_b.

We can add an extra list, List 5, for the box b, denoted by Y_b. Any box that is not adjacent to b's parent is contained in Y_b.

A two dimensional example of the above lists is given in Figure 3.8.

Figure 3.8: Box b and the associated lists 1-5



3.3.3 Brief Description of the Algorithm

Same with the single level FMM, MLFMA can be divided into a upward process and a downward process. The algorithm can be described as follows.

(1) Build up the hierarchical tree structure of boxes by firstly selecting a number s, then cutting the boxes until the number of particles in each childless box is less than or equal to s.

(2) Upward process. From the finest level to the coarsest level, calculate the multiple

expansion for each box. If the box is childless, calculate its multipole expansion from the particles inside using Eq. (3.1). If the box is a parent box, calculate its multipole expansion by shifting and adding up the multipole expansions of its child boxes using Eq. (3.2) and Eq. (3.3).

(3) Downward process. From the coarsest level to the finest level. For each box check the boxes in its colleague list and in its parent box's colleague list and the descents of these boxes if needed, and take actions according to their relations. Considering box b, if c is in U_b, calculate the field inside b due to particles in c by the Coulomb force law if they are both childless boxes. If c is in V_b, convert the multiple expansion of c into the local expansion of b. If c is in W_b, calculate the field inside b from the multipole expansion of c using Eq. (3.11). If c is in X_b, calculate the local expansion of b due to the particles in c using Eq. (3.8). Also translate the local expansion from the parent box into its child boxes. After the downward process, all the boxes have a local expansion, which may come from three sources, the particles inside other boxes, the multipole expansions of other boxes, and the local expansions from the parent box. For all childless boxes, the field inside has been partially calculated from the particles inside the near boxes and/or the multipole expansions from some other boxes.

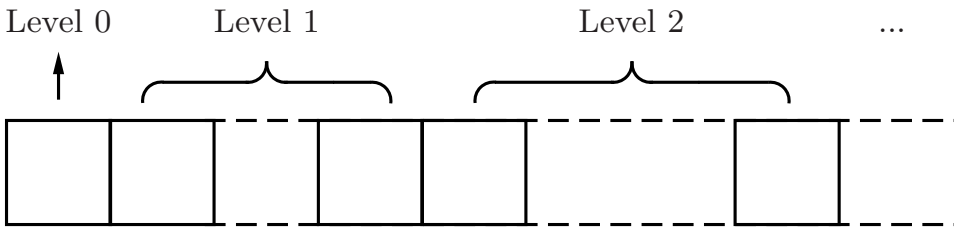
(4) Field calculation. For each childless box, calculate the field inside from the local expansions and add the results to what have been calculated. By now we have calculated the field for all the particles.

3.3.4 Data Structure

We use several arrays to save the necessary information for the boxes. The data is saved in such a way that (1) only the non-empty boxes are saved; (2) Boxes are saved consecutively;

(3) the coarser level boxes are saved before the finer level boxes, as Figure 3.9 shows; and (4) different kinds of information are saved into different arrays, using the index of the element in the arrays as the identification of the boxes, which means the elements with the same index in different arrays store different information for the same box and the elements in the same array with different indexes store the same information for different boxes. The data structure is described with details in the following.

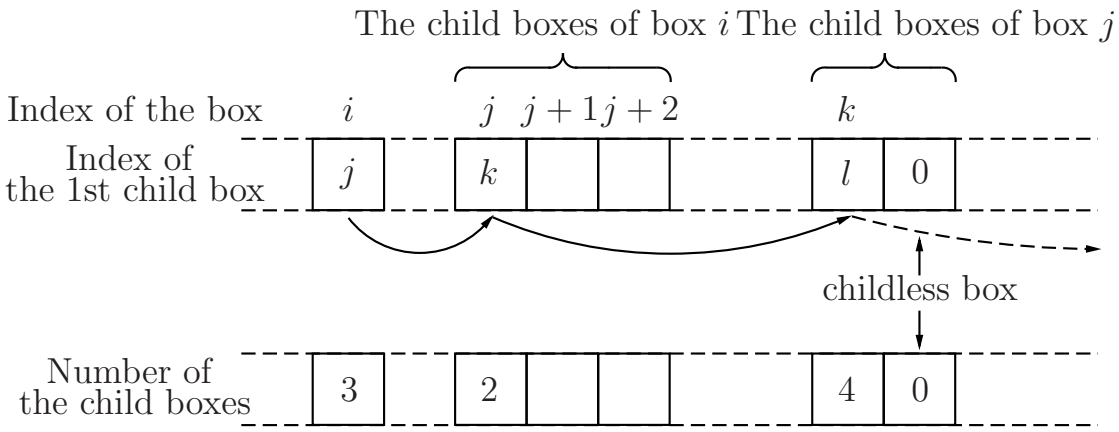
Figure 3.9: Sequenced boxes



We use the linked list structure to save the hierarchical structure of the boxes. Two one dimensional arrays, *box* and *nchld*, are used, both having the same length as the number of boxes. The i^{th} element of *box* saves the index of the first child box of the box i, and the i^{th} element of *nchld* saves the number of child boxes of the box i. The i^{th} element of both *box* and *nchld* are zero if the box i is childless. So if the i^{th} element of *box* has the value j, and the corresponding element of *nchld* has the value k, we know that the box from j to $j+k-1$ are all child boxes of the box i. The value of each element from j to $j+k-1$ in the array *box* tells us where its first child box is, and the value of each box from j to $j+k-1$ in the array *nchld* tells us how many child boxes it has. Using *box* and *nchld* we can search the hierarchical structure downwards, as shown in Figure 3.10. We use another one dimensional array *prnt* with the same length as the number of boxes to save the index of the parent box for each box. If the i^{th} element of *prnt* has the value k, it means the parent box of the box i

is the box k. The first element of *prnt* is zero, because the level zero box has no parent box. Together with the array *prnt*, we can search the hierarchical structure upwards.

Figure 3.10: Storage of the hierarchical structure of the boxes



The one dimensional array *lvl* saves the level for each box. The two dimensional array *cntr* saves the coordinates of the center for each box. The one dimensional array *mltp* and *lcl* save the far multipole expansion and the local expansion for each box respectively.

The coordinates of all the particles are saved in three vectors, x, y and z, and the fields on them are saved in three other vectors E_x, E_y, and E_z. Some operations on the vector data type can improve the efficiency [15]. The linked list structure is used again to save the information about the particles inside each childless box. If the i^{th} box is childless, the index of the first particle inside it is saved in the i^{th} element of the one dimensional array *idx*, and the total number of particles inside is saved in the i^{th} element of the one dimensional array *nptcl*. For a parent box, the corresponding elements in *idx* and *nptcl* are both zero. There is another one dimensional array *link*, whose length equals the number of particles. Each element of *link* saves the index of the next particle, or zero if there are no more particles, in the same childless box. In order to find all the particles inside any childless box i, we first find the first particle from $idx[i]$, then find the following from $link[i]$, $link[link[i]]$, ... until

we reach the zero. $nptcl[i]$ gives the number of particles inside the box i, which is useful when we need to create a vector to save the particles' coordinates.

Each box may have at most 27 colleagues including itself. We use one dimensional array clg to save the colleagues of all boxes. Each element of clg contains a vector, whose first element is the total number of the saved colleagues for the corresponding box and the following elements are the indexes of those colleagues. The vector length is at most 28. Note that for a box b, only the colleagues whose indexes are behind b are saved.

Table 3.1: Variables and their sizes

Array	*box*	*nchld*	*lvl*	*prnt*	*cntr*	*clg*
Size	$1 \times N_b$	$1 \times N_b$	$1 \times N_b$	$1 \times N_b$	$3 \times N_b$	$28 \times N_b$
Array	*mltp*	*lcl*	*idx*	*nptcl*	*link*	
Size	$N_{da} \times N_b$	$N_{da} \times N_b$	$1 \times N_b$	$1 \times N_b$	$1 \times N_p$	
Vector	x	y	z	E_x	E_y	E_z
Size	$1 \times N_p$	$1 \times N_p$	$1 \times N_p$	$1 \times N_p$	$1 \times N_p$	$1 \times N_p$

Table 3.1 shows the variables and their sizes. N_b means the number of boxes, N_p means the number of particles, and N_{da} means the size of a DA vector, which is determined by the order of the DA vector and the number of DA variables. If a DA vector is an n^{th} order polynomial of v variables, $N_{da} = C(n+v)/(C(n) \cdot C(v))$, where $C(n)$ means factorial n [14].

3.4 Numerical Results

In the following we present some numerical results from experimental calculations. All the calculations are performed on a computer with a Four Core (Eight Hyperthreaded) Intel Xeon Processor X5677 running at approximately 3.5 GHz, but our program is a single process program. In Table 3.2 we list some calculation results of an electron bunch with the uniform distribution. N is the total number of the electrons, s is the largest number of

electrons that each childless box can hold, t_{M} is the computation time using MLFMA, t_{d} is the computation time using direct pair-to-pair formula, and Err_1, Err_2, and Err_3 are relative errors of E_x, E_y, and E_z, which is defined as

$$Err = \sqrt{\frac{\sum_i (E_{\mathrm{M},i} - E_{\mathrm{d},i})^2}{\sum_i E_{\mathrm{d},i}^2}}, \tag{3.26}$$

where E_{M} is the electric field calculated by MLFMA and E_{d} is the electric field by direct pair-to-pair formula. t_{d} for 400,000 or more electron cases is estimated according to the values for less electron cases. Same results for an electron bunch with the Gaussian distribution and a group of ten electron bunches, each one holding one tenth of the total electrons with the Gaussian distribution, are listed in Table 3.3 and Table 3.4. From these results, we can see the MLFMA has much better efficiency than the direct pair-to-pair calculation. If we plot the computation time with the particle numbers, as shown in Figure 3.11, we can see the computation time shows a roughly linear increase with the number of particles.

Figure 3.11: Relation between computation time and particle number

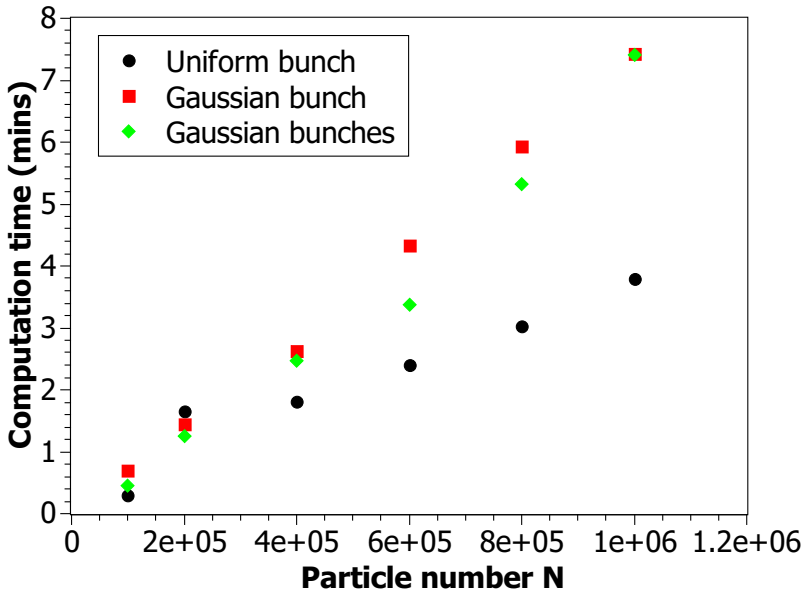


Table 3.2: Comparison of the computation time for the uniform distribution bunches of n particles by direct calculation (t_{d}) and MLFMA calculation (t_{M}) with DA order p and each childless box holding at most s particles, given the relative errors (Err_1,Err_2, and Err_3) of the field in three different direction

N	s	p	t_{M} (min)	t_{d} (min)	Err_1 ($\times 10^{-4}$)	Err_2 ($\times 10^{-4}$)	Err_3 ($\times 10^{-4}$)
1e5	400	5	0.292	75.315	1.241	1.388	1.271
2e5	400	5	1.655	300.030	1.498	1.504	1.498
4e5	400	5	1.801	1.205×10^3	1.482	1.519	1.554
6e5	400	5	2.395	2.711×10^3	1.538	1.550	1.524
8e5	400	5	3.026	4.820×10^3	1.522	1.497	1.537
1e6	400	5	3.785	7.532×10^3	1.520	1.402	1.526

Table 3.3: Comparison of the computation time for the Gaussian distribution bunches of n particles by direct calculation (t_{d}) and MLFMA calculation (t_{M}) with DA order p and each childless box holding at most s particles, given the relative errors (Err_1,Err_2, and Err_3) of the field in three different direction

N	s	p	t_{M} (min)	t_{d} (min)	Err_1 ($\times 10^{-4}$)	Err_2 ($\times 10^{-4}$)	Err_3 ($\times 10^{-4}$)
1e5	400	5	0.693	75.315	4.074	3.181	3.509
2e5	400	5	1.448	300.030	4.174	3.260	3.370
4e5	400	5	2.624	1.205×10^3	4.051	3.281	3.509
6e5	400	5	4.325	2.711×10^3	4.189	3.268	3.501
8e5	400	5	5.932	4.820×10^3	4.122	3.307	3.668
1e6	400	5	7.420	7.532×10^3	3.995	3.200	3.464

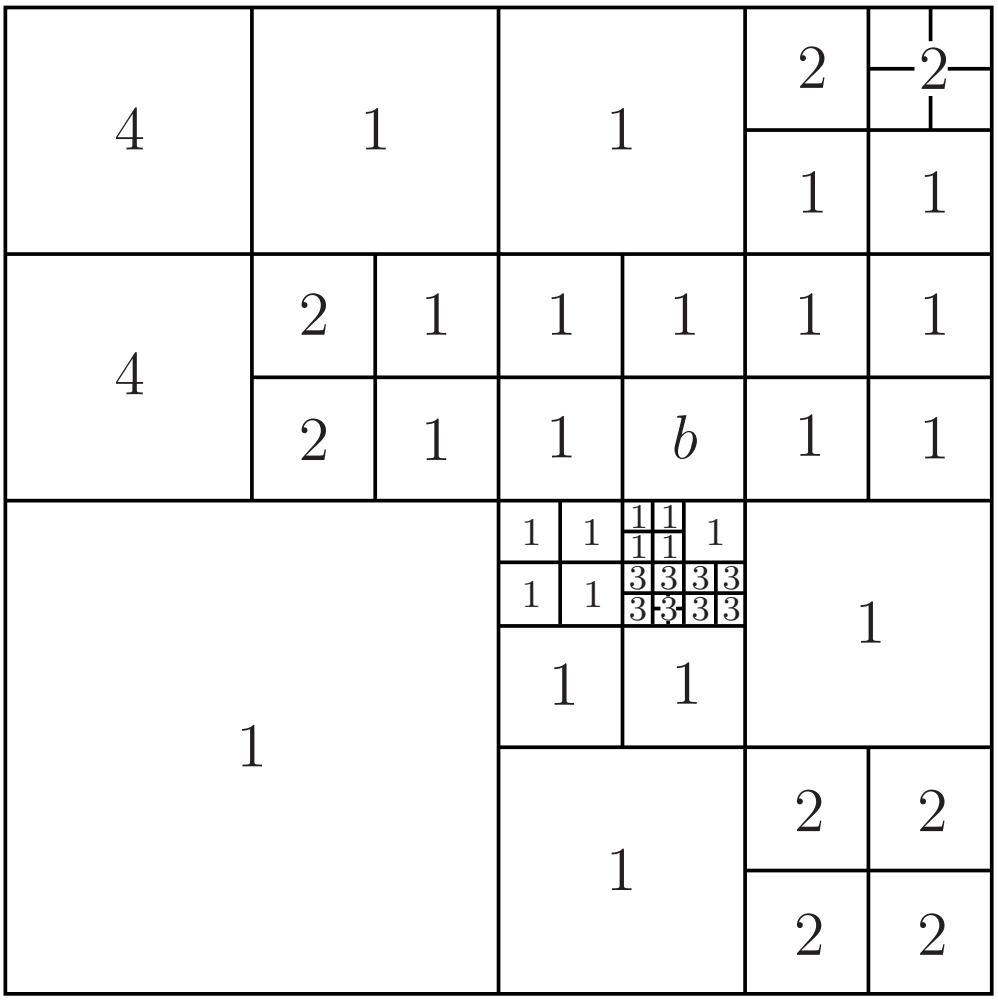
Table 3.4: Comparison of the computation time for the Gaussian distribution bunch groups of n particles by direct calculation (t_{d}) and MLFMA calculation (t_{M}) with DA order p and each childless box holding at most s particles, given the relative errors (Err_1,Err_2, and Err_3) of the field in three different direction

N	s	p	t_{M} (min)	t_{d} (min)	Err_1 ($\times 10^{-4}$)	Err_2 ($\times 10^{-4}$)	Err_3 ($\times 10^{-4}$)
1e5	400	5	0.449	75.315	2.864	4.257	3.262
2e5	400	5	1.256	300.030	5.276	4.954	4.040
4e5	400	5	2.475	1.205×10^3	1.505	3.852	3.124
6e5	400	5	3.374	2.711×10^3	7.115	3.332	3.765
8e5	400	5	5.321	4.820×10^3	7.693	4.574	4.163
1e6	400	5	7.412	7.532×10^3	2.972	6.436	3.652

3.5 Increasing of The Accuracy

From the discussion about the error in section 3.2, one can see the error is determined by the truncation order p and the distance between the observers and the sources. To increase the accuracy, one natural way is to keep more terms in the multipole expansions and the local expansions. Besides that, one can also increase the distance between the source box, in which the charges are approximately represented by the multipole expansion, to the observer box. One only needs to change the definition of the "adjacent boxes" as follows. For a box b

Figure 3.12: Box b and the associated lists 1-5 with $n_a = 2$



with the side length s_b, a box c with the side length s_c is adjacent to b if the distance between the center of b and the center of c is less than $0.5 \cdot s_b + n_a \cdot s_c$, where n_a is a predefined positive integer. When $n_a = 1$, the definition is the same with what in section 3.3.2. One can choose a larger n_a for better accuracy. One example of $n_a = 2$ in two dimensional case is shown in Figure (3.12). Comparing with Figure (3.8), one can see in Figure (3.8) the distance between observer box b and any source box c in V_b, W_b or X_b is at least one times the side length of c, referred to s_c; however in Figure (3.12) this distance is at least $2 \cdot s_c$. It is easy to see for a given n_a, if a box c is well separated with box b, then there are at least n_a boxes with the same size of c in between. So more boxes are included in the near region of b, if we increase n_a. In Table 3.5, we present some numerical results for a bunch of 1,000,000 electrons with the Gaussian distribution. s is set to be 400. The electric fields are calculated with different p and different n_a. Err is the average of Err_1, Err_2, and Err_3. Figure 3.13 shows the error goes down faster with the DA order p for a larger n_a, which is as expected. From Eq. (3.20), Eq. (3.22) and Eq. (3.25) one can see the error is in proportion to K^p. Considering two well separated boxes, K is determined by the rate of the box size to their distance, and k is always less than one. When we increase n_a, K decreases, so that the error converges faster. This explains Figure 3.13, in which when the error is plotted in logarithm scale, it shows a linear relation with p, and the slope is larger when n_a is larger. In practice, we care more about how much time is needed for a given accuracy. We plot the error with the computation time for different n_a in Figure 3.14, in which the error and the computation time shows a roughly linear relation in the logarithm scale and the larger n_a leads to the larger slope. This fact suggests a larger n_a may be a wise choice if we want to achieve very high accuracy, for example 1/100,000,000 or even better. For clarity, in the following chapters we assume $n_a = 1$, if n_a is not specified.

Table 3.5: Computation time t_{M} and the relative error Err for a Gaussian bunch with 1000,000 electrons for different DA order p and different n_a, i.e. different definition of the "adjacent boxes"

p	n_a	t_{M} (min)	Err	p	n_a	t_{M} (min)	Err
3	1	3.275	5.805×10^{-3}	5	2	36.631	2.267×10^{-5}
4	1	4.916	1.314×10^{-3}	6	2	69.087	3.592×10^{-6}
5	1	8.528	3.553×10^{-4}	7	2	150.910	6.099×10^{-7}
6	1	16.866	1.008×10^{-4}	8	2	282.467	1.160×10^{-7}
7	1	32.620	3.124×10^{-5}	3	3	33.953	3.823×10^{-4}
8	1	61.569	8.566×10^{-6}	4	3	53.489	3.453×10^{-5}
9	1	115.170	3.243×10^{-6}	5	3	93.360	3.914×10^{-6}
10	1	212.149	1.032×10^{-6}	6	3	176.112	4.369×10^{-7}
3	2	13.270	1.113×10^{-3}	7	3	377.196	5.250×10^{-8}
4	2	20.947	1.434×10^{-4}	8	3	697.151	8.368×10^{-9}

Figure 3.13: Relation between the error and the DA order for different n_a, i.e. the different definitions of the "adjacent boxes"

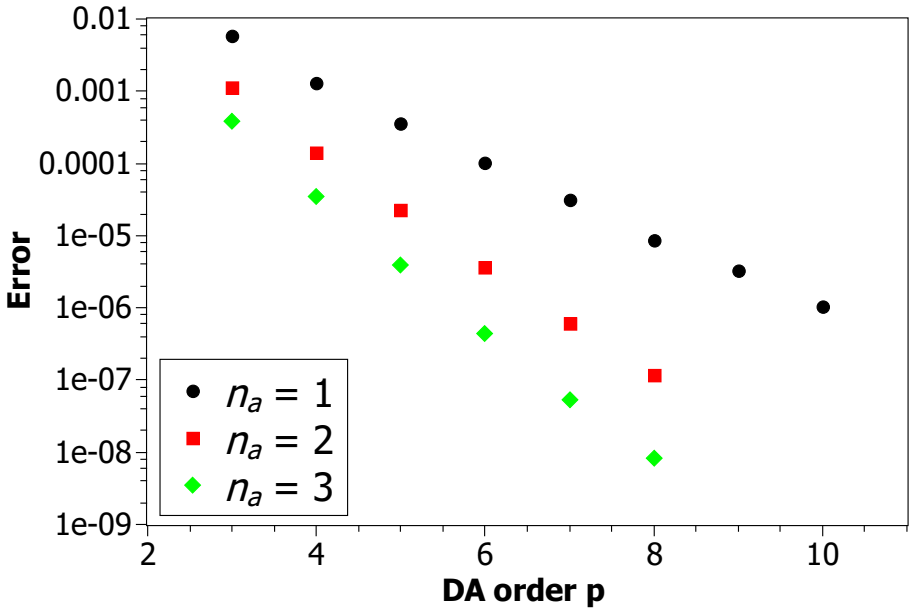
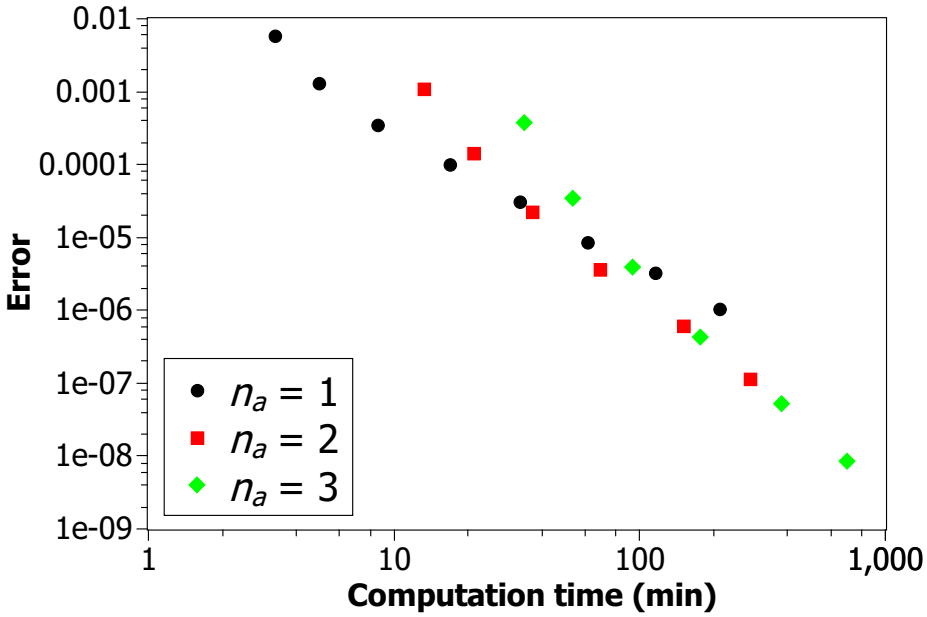


Figure 3.14: Computation time of different accuracies for different n_a, i.e. the different definitions of the "adjacent boxes"



Chapter 4

Parallel Version of the Multiple Level Fast Multipole Algorithm

4.1 Introduction

To simulate modern accelerators with high intensity beams, one may have to calculate the interaction between billions of particles, which makes it necessary to take the advantage of the parallel calculation and run the code in cluster machines with thousands of processors. The single level fast multiple algorithm for uniformly distributed charged particles is easier to parallelize. The MLFMA is relatively difficult to parallelize, because its hierarchical box structure depends on the distribution of charged particles, which keeps changing during the simulation and makes it difficult to predict how many boxes will be created and how many boxes will have an interaction with a given box.

Our parallel approach is based on COSY Infinity 9.1 with MPI support. The official release of COSY Infinity 9.1 does not have MPI support built in, which is likely being rolled out in the official version 10. However, users can easily revise the source files and compile COSY with MPI support following the instruction in COSY 9.1 programmer manual[15]. COSY supports the communication between the processes in a distributed cluster machine by PLOOP, which has the following structure:

PLOOP I 1 NP ;

...

STATEMENTS ;

...

ENDPLOOP PMAP ;

where NP is the number of processes and PMAP is an array whose last dimension is equal to or greater than NP. PLOOP distributes the STATEMENTS over all processes. Each process takes actions and writes its result into the respective element of PMAP to share with the other processes. After PLOOP, each process receives the results from all the others. As one can see, PLOOP is an all-to-all communication, which is the only MPI communication that is currently supported by COSY. In our parallel MLFMA code, all the communications between processes use PLOOP.

There are some good discussions on how to parallelize MLFMA in both the shared memory machine and the distributed memory machine. The topics cover the creation, storage, and sort of the partial octree that represents the hierarchical box structure, the calculation of the multipole expansions, the calculation of the local expansions and the fields in parallel. In 1987, Zhao developed a parallel algorithm for the single level FMM in the distributed memory machine [94], soon after which Greengard and Gropp developed the parallel algorithm for the MLFMA in the shared memory machine [37]. The most challenging is the parallel algorithm for the MLFMA in the distributed memory machine, which is highly desired since most modern super computers with thousands of processors have the distributed memory structure and support MPI. The hierarchical box structure of the MLFMA is highly unbalanced. It is difficult to predict the total number of boxes and the number of interaction boxes for each single box, so that it is difficult to predict the total calculation cost and also difficult to decompose and distribute the work load to each process. The MLFMA is

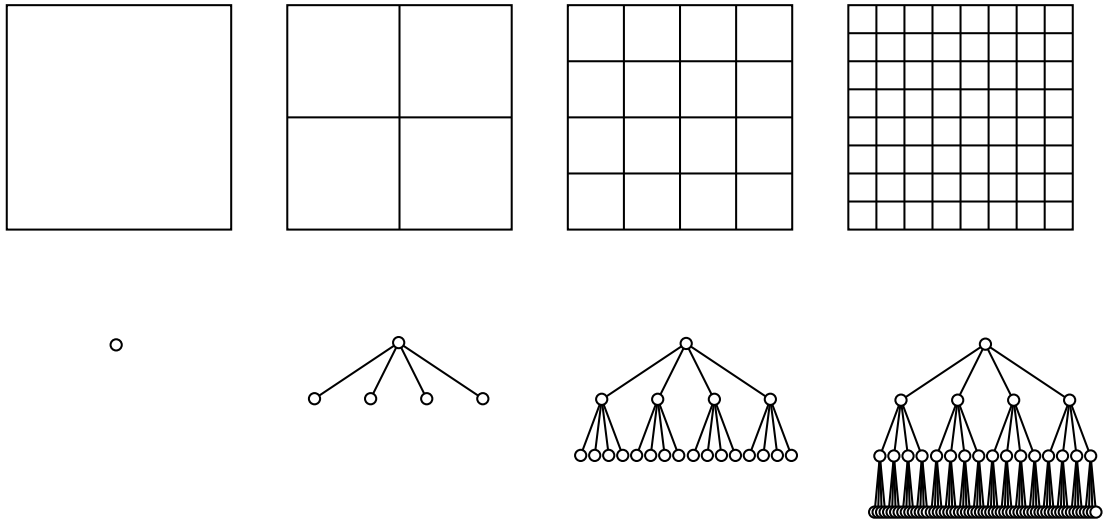
actively applied in solving Helmholtz equation related problems, such as the reflection of electromagnetic waves, and the electrostatic potential related problems. Different parallel algorithms have been presented in these two areas [79, 78, 4, 44, 89, 88, 29, 33] and many discussions have been made [91, 81, 82], while people in other areas also made their contribution [21, 92]. Teng [86] and Aluru [79] independently developed different parallel algorithms with provably good partitioning and load balancing. We borrowed some ideas, for example the compressed tree from Aluru [3, 2], from the previous work. Now COSY only supports the all-to-all communication, which is the most efficient way to communicate with all processes. We develop our own algorithm based on the all-to-all communication, and this has an effect on our algorithm, which will be explained in the following sections.

4.2 Hierarchical Tree Structure

4.2.1 Octree and Compressed Octree

It is natural to use a tree to present the hierarchical structure of the boxes. The largest box that includes all the particles is the root of the tree. The child boxes of the largest box construct the next level nodes connected to the root. A child box may also have child boxes, which construct the following level nodes. In the three dimensional case, the hierarchical structure of the boxes composes an octree. The data structure of a single level FMM is a full tree. A two dimensional example is shown in Figure 4.1. As to a full tree, the number of boxes (nodes) can be calculated as $\sum_{n=0}^{L} 4^L$ for the two dimensional case and $\sum_{n=0}^{L} 8^L$ for the three dimensional case, where L is the depth of the tree. If the charged particles have a uniform distribution, so that all the childless boxes hold the same or very similar number of particles, L should satisfy $4^L > N/s > 4^{L-1}$ or $8^L > N/s > 8^{L-1}$ for two

Figure 4.1: Full tree structure



dimensional and three dimensional, where N is the total number of particles and s is the largest number of particles each childless box can hold. Thus $L = \lceil \log(N/s) - \log 4 \rceil$ or $L = \lceil \log(N/s) - \log 8 \rceil$. The number of boxes inside the interaction list is also fixed, 27 for the two dimensional and 189 for the three dimensional. With all these information, it is easier to parallelize the single level FMM. However, the hierarchical structure of the boxes for the MLFMA should be represented as a partial octree. A two dimensional example of the partial tree and the corresponding hierarchical box structure is shown in Figure 4.2. Comparing with the full tree for single level FMM in Figure 4.1, some notes are missing in the partial tree, which is decided by the distribution of the charged particles. And it is impossible to predict the number of boxes only from N and s, if the charge distribution is arbitrary and unpredictable.

In a normal tree, a box could have only one child box, if the child box is not childless. In this case, the box and its only child box contains the same group of particles. A path as shown in Figure 4.3 could happen. All the boxes of $(b, d, ..., e, f)$ contain the same group

Figure 4.2: Partial tree structure

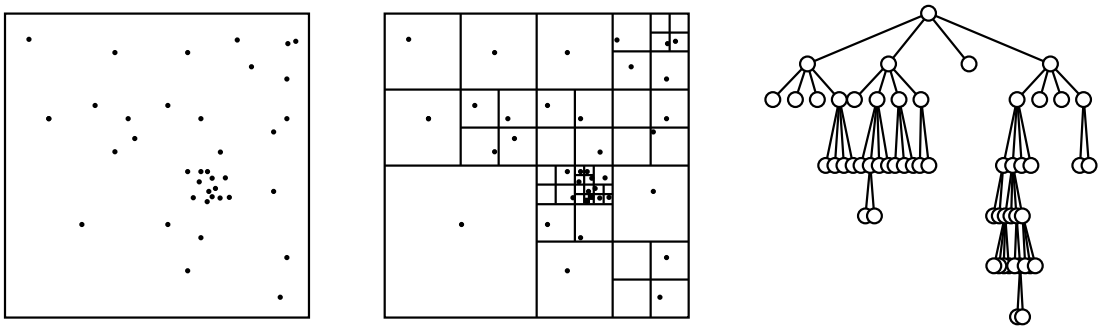
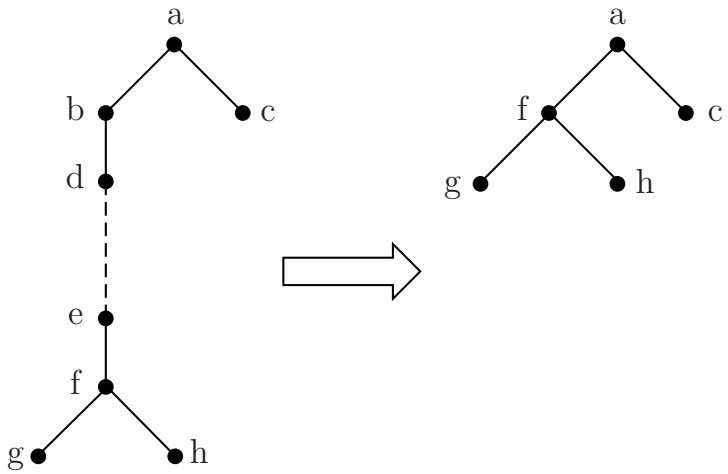


Figure 4.3: From a normal tree to a compressed tree



of particles, although they have different sizes. If we use the normal tree structure, all the boxes of $(b, d, ..., e, f)$ will be stored in the memory, and their multipole expansions and local expansions will be calculated, which is actually not necessary. To avoid this, a compressed tree structure can be employed. The difference between a compressed tree and a normal tree is any box in the compressed tree has at least two child boxes. As Figure 4.3 shows, to change a normal tree into a compressed tree, one simply needs to remove all the boxes who has only one child box, and links the remaining part of the branch to the parent box of the removed boxes. We used the compressed octree in our algorithm.

Even if the compressed octree structure is used, it is still difficult to predict the number of

boxes. In the worst case, the number of boxes could be very close to the number of particles. Consider an two dimensional example in Figure 4.4. Assuming we have a bunch of charged particles, which we enclosed by a square box, when we cut the box into the next level, we find one particle in each box, except that the box in the lower right corner has more than s particles inside. In the next step we only need to cut the box at the lower right corner. After we cut it, the same thing appears and only the box in the lower right corner is left to be cut again. We repeat this process until the box in the lower right corner encloses $s-2$ particles. In this way the number of boxes we generate is $N-s+1$, which is very close to the number of particles N, since in general N is a very large number, from hundreds of thousands to millions, while s is much smaller number, a few hundreds or less. In our code we do not want to use the array with the length up to N to save the hierarchical box structure, because in most cases we do not have so many boxes and it will be a waste of memory if we do so. Instead we will estimate the length of the array according to our experience. If it turns out the array is not long enough, the algorithm will quit and restart automatically with a larger array.

4.2.2 Storage of the Compressed Octree

We use several arrays to save the octree and some related information. The data is saved in such a way that (1) boxes are saved consecutively; (2) the higher level boxes are saved before the lower level boxes, as Figure 4.5 shows; and (3) different kinds of information are saved into different arrays, using the index of the element in the arrays as the identification of the boxes, which means the elements with the same index in different arrays store different information for the same box and the elements with different indexes in the same array store the same kind of information for different boxes. Please note in the compressed tree

Figure 4.4: The maximum number of boxes generated in the worst case

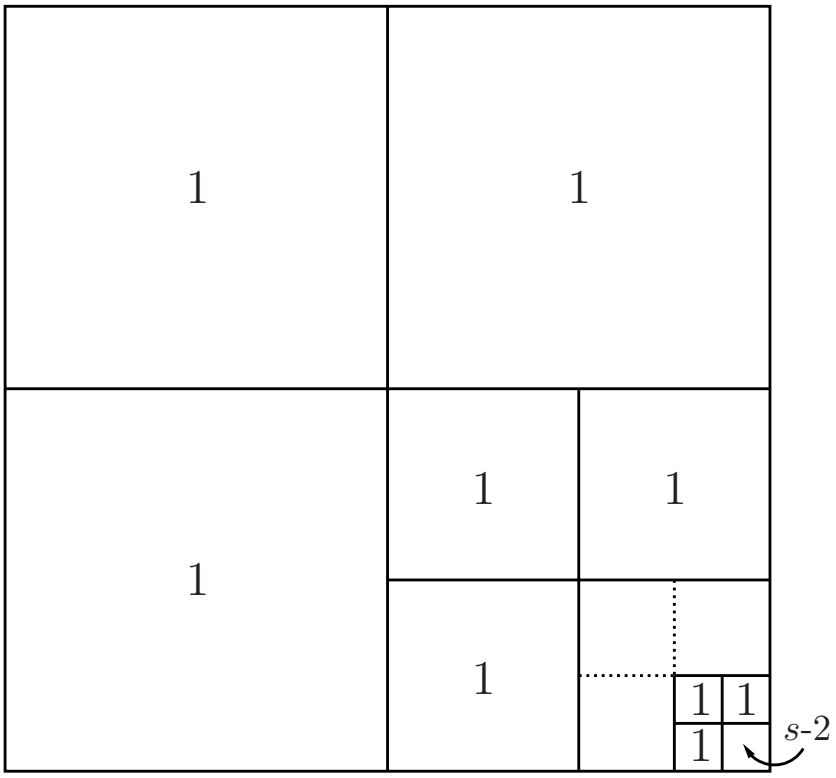
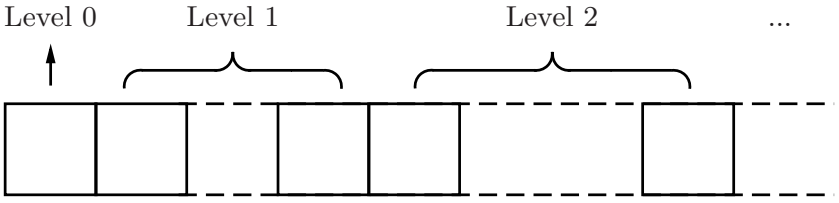


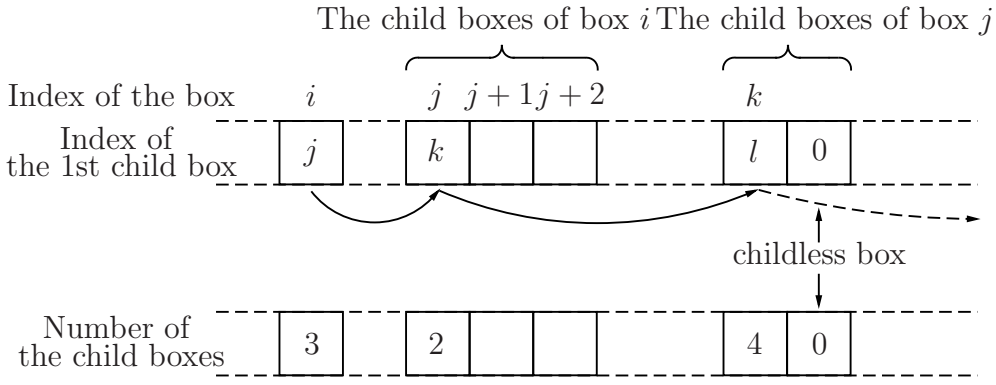
Figure 4.5: Sequenced boxes



structure, there are two kinds of "level". One is related to the position of a box inside the tree, and the other is related to the size of a box. We use "level", as in the above (2), referring to the tree position level , and use "cutting level" referring to the box size level. In a normal tree, all the boxes in the same level of the tree have the same box size, so there is no difference between the two kinds of "level". However in the compressed tree, since we removed those boxes that have only one child each, we can not relate the box size directly to its position in the tree. We have to use an extra number "cutting level" to represent the box size. Take the compressed tree in Figure 4.3 as an example, box f and c are in the same level, but f has a smaller size, hence a finer cutting level. The data structure is described with details in the following.

We use the linked list structure to save the hierarchical structure of the boxes. Two one dimensional arrays, *box* and *nchld*, are used, both having the same length as the number of boxes. The i^{th} element of *box* saves the index of the first child box of the box i, and the i^{th} element of *nchld* saves the number of child boxes of the box i. The i^{th} element of both *box* and *nchld* are zero if the box i is childless. So if the i^{th} element of *box* has the value j, and the corresponding element of *nchld* has the value k, we know that the box from j to $j + k - 1$ are all child boxes of the box i. The value of each element from j to $j + k - 1$ in the array *box* tells us where its first child box is, and the value of each box from j to $j + k - 1$ in the array *nchld* tells us how many child boxes it has. Using *box* and *nchld*

Figure 4.6: Storage of the compressed tree



we can search the compressed tree structure downwards, as shown in Figure 4.6. We use another one dimensional array *prnt* with the same length as the number of boxes to save the index of the parent box for each box. If the i^{th} element of *prnt* has the value k, it means the parent box of the box i is the box k. The first element of *prnt* is zero, because the level zero box has no parent box. Together with the array *prnt*, we can search the compressed tree structure upwards.

The one dimensional array *lvl* saves the level for each box. The two dimensional array *cntr* saves the coordinates of the center for each box. Each processor save the coordinates of n_{pp} particles in three vectors x, y, and z. The total particle number $N = n_{pp} \cdot n_p$, if our program runs on n_p nodes. Another three vectors, xx, yy and zz are used to save the coordinates of all the particles, in order to decrease the communication time.

The linked list structure is used again to save the information about the particles inside each childless box. If the i^{th} box is childless, the index of the first particle inside it is saved in the i^{th} element of the one dimensional array *idx*, and the total number of particles inside is saved in the i^{th} element of the one dimensional array *nptcl*. For a parent box, the corresponding elements in *idx* and *nptcl* are both zero. There is another one dimensional array *link*, whose length equals the number of particles. Each element of *link* saves the index

of the next particle, or zero if no more particles, in the same childless box. In order to find all the particles inside any childless box i, we first find the first particle from $idx[i]$, then find the following from $link[i]$, $link[link[i]]$, ... until we reach the zero. $nptcl[i]$ gives the number of particles inside the box i, which is useful when we need to create a vector to save the particles' coordinates.

4.2.3 Tree Construction

In the parallel algorithm, each process saves the coordinates of n_{pp} particles. The basic idea of the tree construction in parallel is to let each process only deal with the n_{pp} particles saved in itself. To determine whether a box should be split, we only need to know the number of particles inside the box, which can be calculated by gathering the information from all processes and take the summation. During the tree construction procedure, each process uses a temporary array $plink$ with the length n_{pp} to save which box each particle belongs to. In the end we will gather $plink$ from all processes, and by linking the paths saved in different $plink$ for each childless box, we can construct the array $link$. We will also gather x, y, and z from all the processes and construct xx, yy, and zz for future computation. The procedure of the parallel tree construction is shown in Figure 4.7.

The algorithm of the tree construction in parallel can be described as follows.

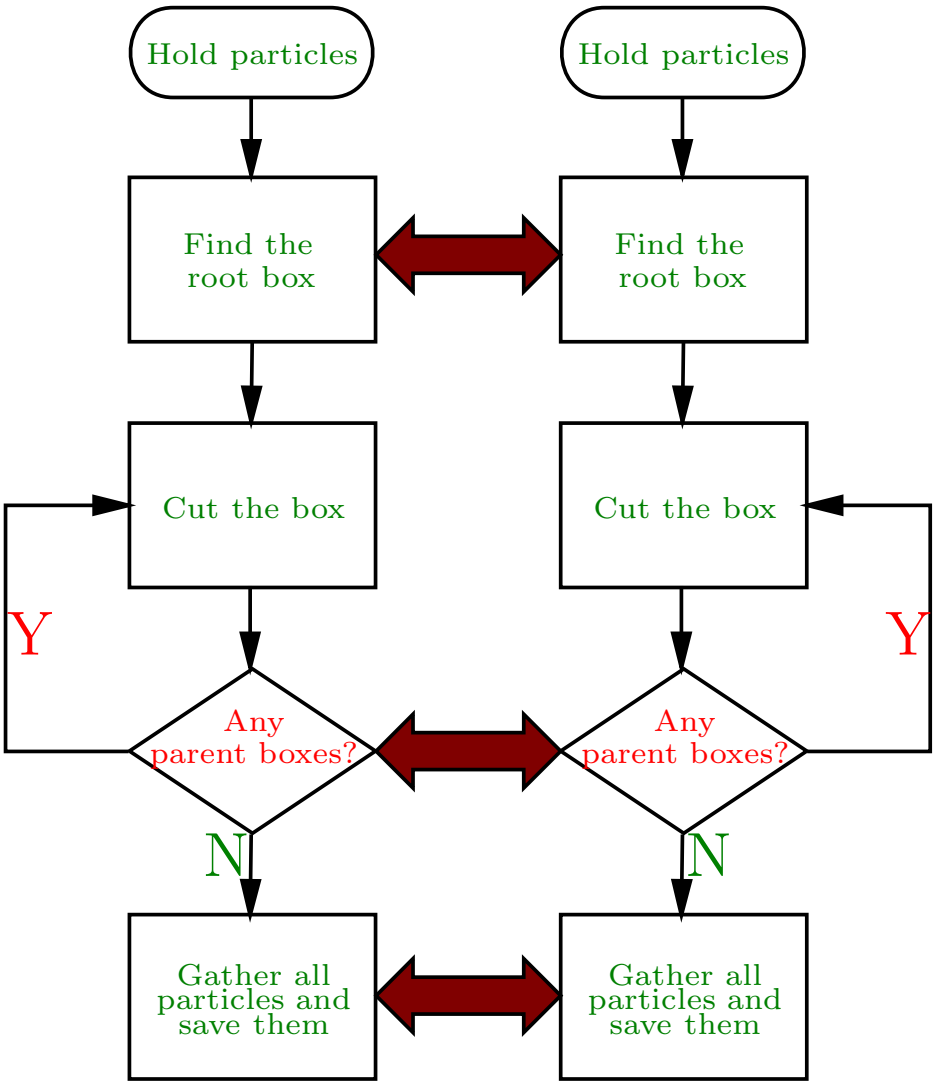
(1) Each process saves the coordinates of n_{pp} particles.

(2) Calculate the range of the coordinates in each process, gather the results from all processes to determine the proper size of the root box that contains all the particles.

(3) Set the value of the first element in $cntr$, lvl and $nptcl$, which represents the root box.

(4) Use two pointers to record the range of the boxes to treat. Initialize them as $pnt1 = 1$

Figure 4.7: Parallel tree construction



and $pnt2 = 1$.

(5) For all the boxes between $pnt1$ and $pnt2$, if it contains more than s particles, split it into eight child boxes, distribute the particles into the child boxes and for each child box save the number of particles inside it and save the particle list in idx and $plink$.

(6) For all the boxes between $pnt1$ and $pnt2$, gather the number of particles in their child boxes. Check each box between $pnt1$ and $pnt2$ in sequence as follows. For each child box take the summation of all processes to obtain the total particle number. If a box has only one child box, replace the information of the parent box by its child box and split the child box. Repeat until two or more child boxes are found. If a box has more than one child boxes, set the respective values in box, $nchld$, and $nptcl$ for the parent box, and in $cntr$, lvl, and $prnt$ for the child boxes.

(7) Set $pnt1 = pnt2 + 1$, and then make $pnt2$ point to the last element in the box array.

(8) Repeat (5) to (7) until all the childless boxes contain less than s particles.

(9) Gather the idx value for each childless box and $plink$ from all processes, and set proper values for idx and $link$.

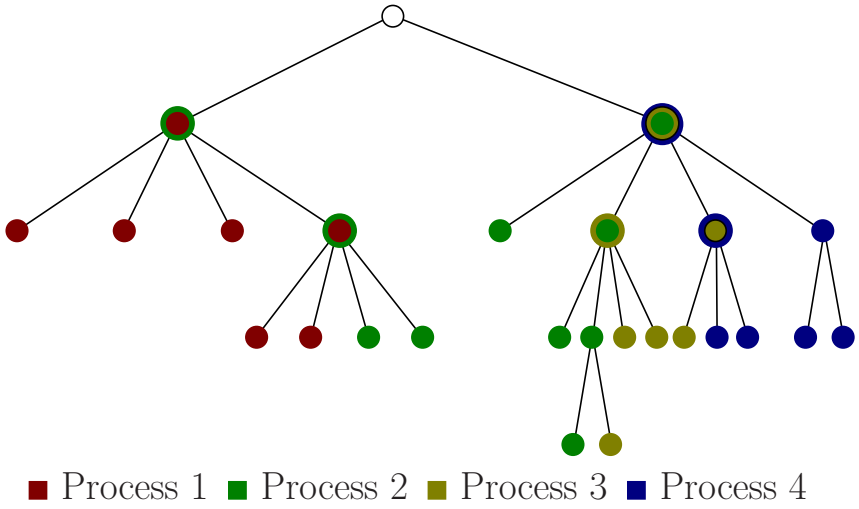
(10) Scale x, y, and z so that the side length of the smallest childless box is 2. Gather the scaled x, y, and z from all the processes and construct xx, yy, and zz, which save the coordinates of all particles.

4.3 Calculation of the Far Multipole Expansions

4.3.1 Tree Partitioning

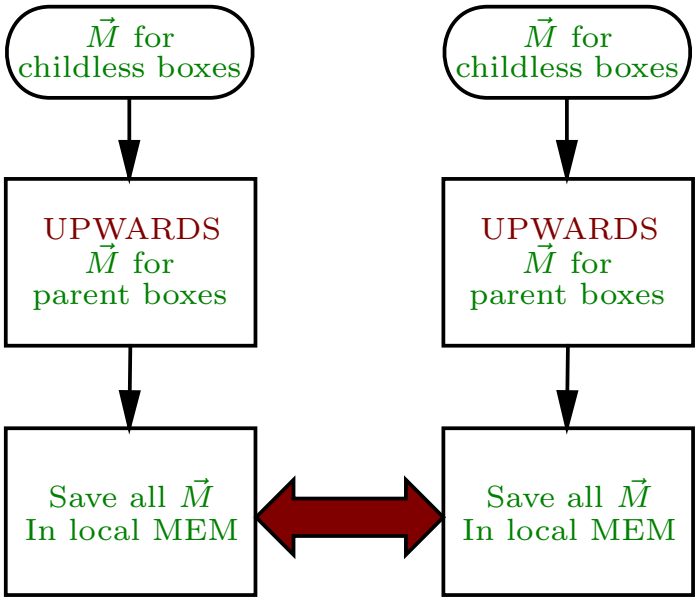
In order to increase the efficiency we want each process to deal with only a part of the tree. We partition the whole tree in such a way that each process is assigned the same number

Figure 4.8: Tree partitioning



of childless boxes and all the ancestor boxes of them. Take the tree in Figure 4.8 as an example. We have totally 18 childless boxes in the tree. Assume we want to distribute this tree into four processes. First, we sequence all the childless boxes from left to the right, no matter which level the box belongs. The we assign the first five childless boxes and their ancestors to the first process, the following five to the second process, the following four to the third process, and final four to the forth process. In this way we try to put as many childless boxes who share the same ancestors as possible to the same process. In Figure 4.8 the belongings to the different processes are presented by different colors. One can see that some ancestor boxes belong to multiple processes, which is presented by multiple colors. It is clear that two consecutive processes can share at most $L-1$ ancestor boxes, with L being the depth of the tree. Since each parent box has at least two child boxes in the compressed tree, if there are n_c childless boxes, the length of the local tree is limited by $2^l - 1$, where $l = 1 + \lceil (\log n_c - \log n_p)/\log 2 \rceil$.

Figure 4.9: Parallel multipole expansion calculation



4.3.2 Parallel Calculation of the Far Multipole Expansions

The local tree is saved in two arrays similar to *box* and *prnt*, which let us search upwards and downwards the tree, in a sequence that a child box is always before its parent box. For an arbitrary tree, the boxes can be sequenced by labeling them in the following way.

(1) Check all the boxes from the root of the tree.

(2) If the box is childless or all the child boxes of it have been labeled, label this box and go back to its parent box. Otherwise check the first unlabeled child box of it.

(3) Continue until all the boxes are labeled.

The procedure of calculating the far multipole expansions in parallel is shown in Figure 4.9, and the algorithm can be described as follows.

(1) Each process calculates the far multipole expansions for the boxes in its own local tree from the beginning to the end of the array. If the box is childless, the far multipole expansion is calculated from the particles inside the box. If the box is a parent box, the far

multipole expansion is calculated by shifting the far multipole expansions of its child boxes to the center of the parent box and taking the summation. Since any box is sequenced before its parent box, when we need to calculate the far multipole expansion of a parent box, the far multipole expansions of all its child boxes have been calculated.

(2) Gather the far multipole expansions from all processes and save them in the array *mltp* for future calculation. If a box belongs to multiple processes, which means its far multipole expansion has been calculated by multiple processes, take the summation.

When using PLOOP to gather all the far multipole expansions in (2), the length of the element in PMAP is determined by the longest local tree. The total memory needed is the production of the memory needed to save the far multipole expansions of the longest local tree and the number of processes, which is often more than the memory needed to save all the far multipole expansions of the whole octree. In some case, when the local trees are high unbalanced, the overestimation may drain too much memory and cause the failure of the algorithm. An alternative that avoids this is to calculate the far multipole expansions according to the different level in the whole octree, which can be described as follows.

(1) Find the boxes in the finest level of the whole octree and distribute them evenly to all processes.

(2) Each process calculates the far multipole expansion for the boxes it gets.

(3) Communicate between all processes to share the far multipole expansions that are calculated in (2).

(4) Find the boxes in the coarser level and repeat (2) and (3), until all the boxes are treated.

In this way we will need L times of communication, where L is the depth of the whole octree, but in each communication the memory needed is much less than the total memory

needed to store the far multipole expansions of the whole octree. The overestimation in PMAP is less than the memory needed to save n_p far multipole expansions. Note that the sequenced local tree is still useful in the following calculation, although not used in the calculation of far multipole expansions.

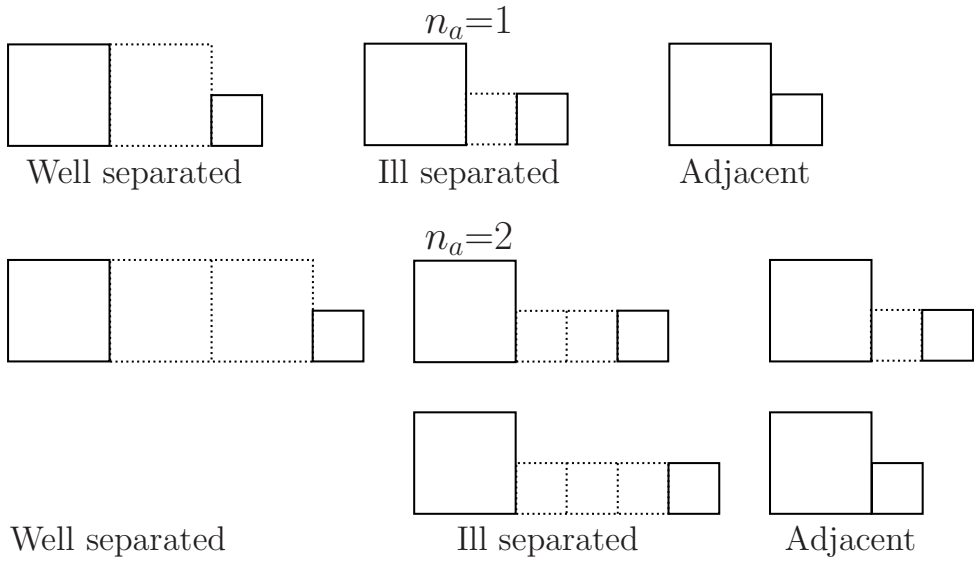
4.4 Calculation of the Local Expansions and the Fields

4.4.1 Relation between Boxes

To calculate the local expansions, we have to consider the relation between the boxes, which is a little more complicated in a compressed tree than in a normal tree. For any arbitrary two boxes, there are three kinds of relations between them according to their positions and their sizes. Consider two boxes b and c. Without loss of the generality, we assume b is ether larger than or have the same size with c, referred to as $S_b \geq S_c$, where S_b and S_c are the side length of b and c. d is the largest value of the projections of the distance between the centers of b and c in x, y, and z direction. If $d > (0.5 + n_a) \cdot S_b$, where n_a is a predetermined positive integer, b and c are well separated; if $(0.5+n_a) \cdot S_b > d > 0.5 \cdot S_b + n_a \cdot S_c$, they are ill separated; otherwise they are adjacent. Roughly say, if we can put n_a larger boxes between the two boxes, they are well separated; if we can't do that but we can put n_a smaller boxes in between, they are ill separated; otherwise they are adjacent.Note that the value of d is discrete. if $d > (0.5 + n_a) \cdot S_b$, the smallest value of d is $(0.5 + n_a) \cdot S_b + 0.5 \cdot S_c$. Figure 4.10 shows examples for the three kinds of relations between two boxes for $n_a = 1$ and $n_a = 2$. Note that when $n_a > 1$, the two boxes that are adjacent to each other are not necessarily touching each other.

According to the strategy of the FMM, for any box b, the field or potential of those

Figure 4.10: Relations between two boxes



particles close to it is calculated directly, the contribution of those particles further away is calculated by the multipole expansions or the local expansions, and as to those particles more further away, their contributions are represented by the local expansions in the ancestor boxes of b and will be transferred to b. So every box only interacts with those boxes who do not have interactions with its ancestor boxes. For each parent box b, we set up an action list. The list contains the following four kinds of boxes: (1) the boxes in the same level of b , who are smaller than b and adjacent to or ill separated with b, (2) the boxes in the same level of b, who are larger than b and adjacent to b, (3) the boxes in the lower level of b, who are childless and adjacent to b, and (4) the box b itself. These boxes in the action list of b and/or their descendants enclose the particles whose contributions are not included in the local expansions of b or its ancestor boxes. And, these boxes will interact with the child boxes of b. A childless box has no action list. The length of the action list is limited by $(2 \cdot n_a + 1)^D$, where D is the dimension of the problem. $D = 3$ for a three dimensional problem.

The action lists for all the parent boxes can be generated in parallel as follows.

(1) Each process initializes the action list for the root, which only contains the root box itself.

(2) Each process checks the local tree from the last element to the first element. If the element b is a parent box, do step (3) to (5) to generate its action list. Note b is stored before its parent box, so the action list of b's parent box is always generated before b gets checked, if we check the local tree backwards.

(3) Find the parent box of b. Check each box in the action list of b's parent box. If the box is childless and adjacent to b, add it into b's action list. If the box is a parent box, check all of its child boxes as step (4) states.

(4) If the box is larger than b and adjacent to b, add it to b's action list. If the box is smaller than or in the same size with b and adjacent to or ill separated with b, add it to b's action list.

(5) Add b itself into b's action list.

4.4.2 Parallel Calculation of the Local Expansions and the fields

Assume we want to calculate the local expansion or the field insight the box b, and w is a box in the action list of b's parent box or its descendant. The different actions we can take depend on the different conditions of b and w, which are shown in the Table 4.1. The $\bigcirc$ in the "Box size" column means both $S_b < S_w$ and $S_b \geq S_w$, and the $\bigcirc$ in the "box b" and "Box w" columns means both "childless" and "parent". $C_w \rightarrow L_b$ means to calculate the local expansion in b by the charges in w; $M_w \rightarrow C_b$ means to calculate the field on the charges inside b from the multipole expansion in w; $C_w \rightarrow C_b$ means to calculate the field on the charges inside b from the charges inside w by the Coulomb's theory; $M_w \rightarrow L_b$ means

Table 4.1: Relations between two boxes and the respective actions

Relation	Box size	Box b	Box w	Action
ill separated	$S_b < S_w$	◯	childless	$C_w \to L_b$
		◯	parent	Check w's descendants
	$S_b \geq S_w$	childless	◯	$M_w \to C_b$
		parent	◯	Add w into b's action list
Adjacent	◯	childless	childless	$C_w \to C_b$
			parent	Check w's descendants
		parent	◯	Add w into b's action list
well separated	◯	◯	◯	$M_w \to L_b$

to translate the multipole expansion in w into the local expansion in b.

The procedure to calculate the local expansions and the fields inside each box in parallel is shown in Figure 4.11 and the algorithm can be described as follows.

(1) Each process creates the action list for each box in its local tree.

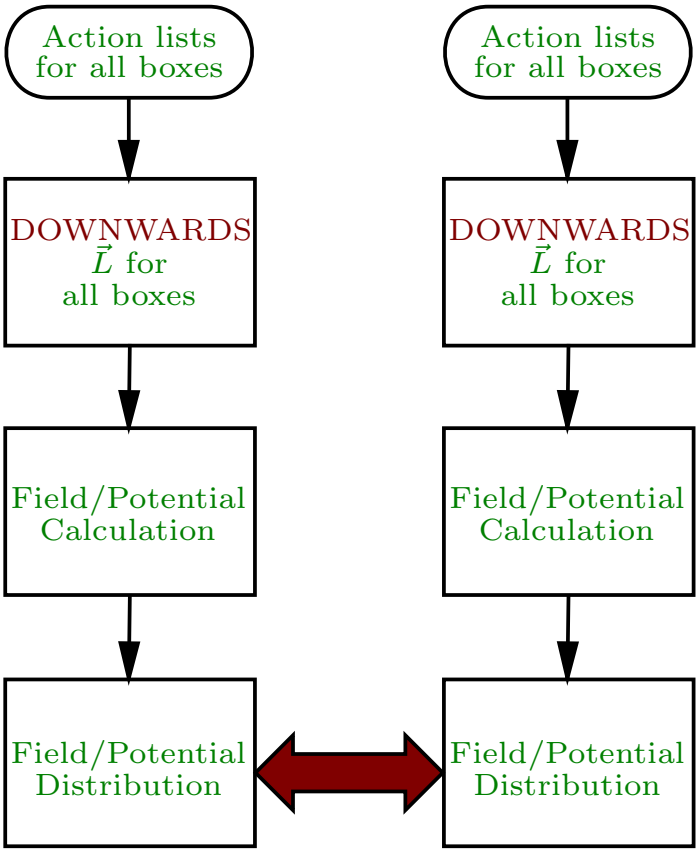
(2) Each process creates three arrays pex, pey, and pez to save the field in x, y, and z direction of the particles inside the childless boxes in its local tree.

(3) Each process checks the local tree from the last element to the first element, and calculates the local expansion and the field as steps (4) and (5) show.

(4) If a box b in the local tree is a parent box, we only need to calculate the local expansion in it. The local expansion comes from three sources, inherited from b's parent box, $C_w \to L_b$, and $M_w \to L_b$, where w is a box in the action list of b's parent box or its descendant. So to calculate the local expansion of b, firstly check each box in the action list of b's parent box. If the box is a childless box, take actions according to Table 4.1. If the box is a parent box, check all of its child boxes one by one and take actions according to Table 4.1. Then translate the local expansion of b's parent box into b and add it to the local expansion that has been calculated from $C_w \to L_b$, and $M_w \to L_b$.

(5) If a box b in the local tree is a childless box, we need to calculate the field on the

Figure 4.11: Parallel local expansion and field calculation



charges inside it. The field comes from three sources, local expansion of b to the charges inside it, $C_w \to C_b$ and $M_w \to C_b$. Check each box in the action list of b's parent box. If the box is a childless box, take actions according to Table 4.1. If the box is a parent box, check all of its child boxes one by one and take actions according to Table 4.1. Then translate the local expansion of b's parent box into b and add it to the local expansion that has been calculated from $C_w \to L_b$, and $M_w \to L_b$. Finally calculate the field from the local expansion of b and add it to the field that has been calculated from $C_w \to C_b$ and $M_w \to C_b$.

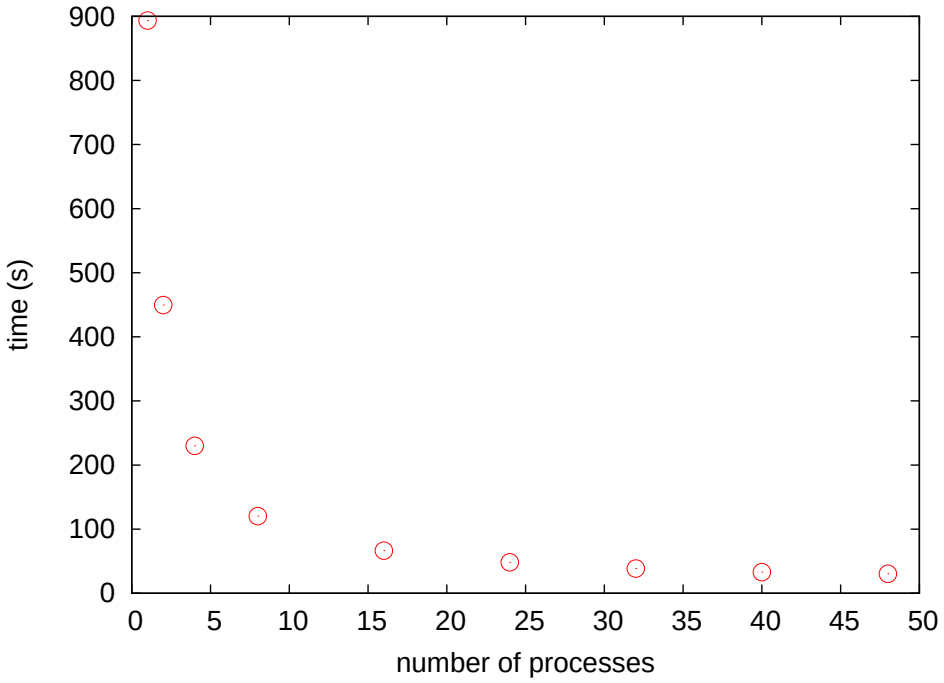
(6) Gather the field data from all the processes. Each process searches for the respective field for its own particles. Note that the particles whose fields are calculated by a process depends on the childless boxes in which they are enclosed, so generally they are not the same particles stored in the process.

4.5 Numerical Results and Discussion

To test the performance of the parallel algorithm, we performed some calculations on our small cluster machine with 48 nodes, each running at 2.2 GHz. Figure 4.12 shows the computational cost to calculate the three dimensional electric field between 1,000,000 electrons of the Gaussian distribution with different numbers of processes. It cost 894 seconds if using only one node. As the number of processes increases, the cost of time decreases. And it cost only 30 seconds when using all the 48 nodes. In these calculations, the DA order is kept up to five and the relative error is less than 0.001. The parallel efficiency can be defined as

$$Eff = \frac{T_{\text{seq}}(N)}{q \cdot T_p(N, n_p)}, \tag{4.1}$$

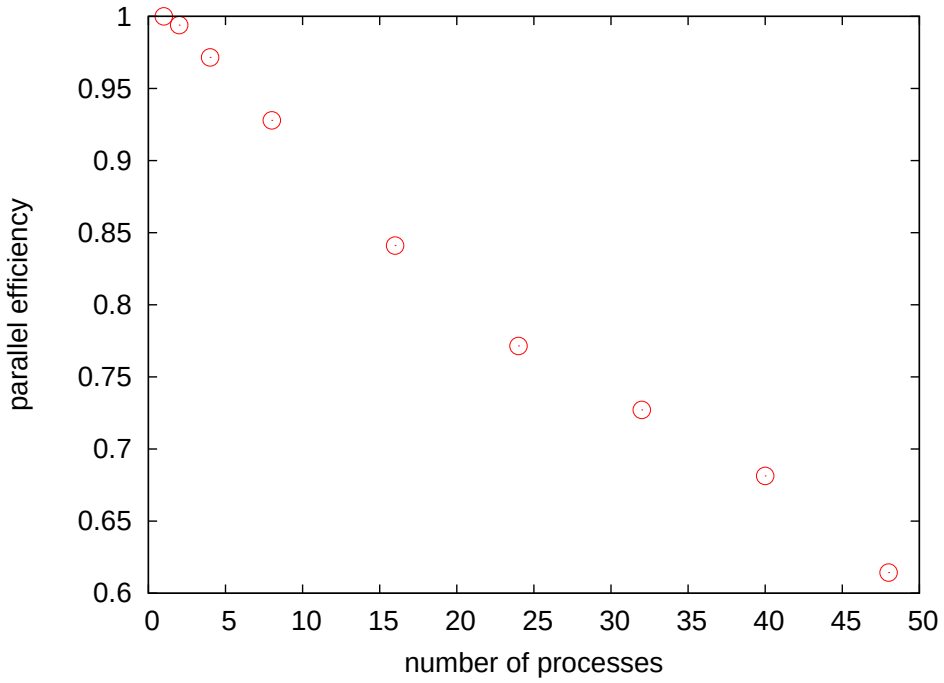
Figure 4.12: Computation time for 1,000,000 electrons with different number of processes



where n_p is the number of processes, N is the number of unknowns, which is 1,000,000, the number of particles, in this case, T_{seq} is the time cost of the sequential algorithm, and T_p is the time cost of the parallel algorithm. As Figure 4.13 shows the parallel efficiency decreases with the number of processes. When we use all the 48 nodes, the parallel efficiency is still above 60%.

Currently we can calculate the three dimensional field for at most about 10,000,000 charged particles with the parallel algorithm. Using 96 nodes from the cluster machine of MSU HPCC, it takes 167 seconds with the DA order up to five and relative error less than 0.001. The parallel efficiency is also about 60%. The bottleneck of the current parallel algorithm is the memory cost. Without a proper compiler, for now we can only compile COSY Infinity 9.1 as a 32 bit Fortran program; so each process can only assess no more than 2 GB memory. We save the coordinates of all particles and all the multipole expansions in the local memory of each process. The memory cost increases with the number of particles,

Figure 4.13: Parallel efficiency for 1,000,000 electrons with different number of processes



no matter how many processes we use. Since the memory that each process can assess is limited by 2 GB, the number of particles that we can treat is also limited. If we do not save these data in the local memory of each process, one process may need to obtain some data stored in another process in the calculation. Because we only have all-to-all communication, all the processes have to stop and wait, even if the communication is only needed between two processes. Since this kind of communication will happen frequently, the efficiency will be heavily affected. In one sentence, we save this data in local memory to avoid too many unnecessary all-to-all communications in order to keep a better efficiency. The price we paid is the memory.

Chapter 5

Using the Fast Multipole Method in Beam Dynamics Simulations

The essential purpose of developing the DA based MLFMA is to simulate the space charge effects in the high density system. In this chapter it will be presented how the algorithm can be used in beam dynamics simulations. As an example, we simulated the photoemission process of applying a laser beam on a metal surface, in which the space charge effect plays an important role. The charge distributions are illustrated in different phase space plots in the early stage soon after the electrons are created. Some related topics are discussed, such as how to choose the proper units for the variables in the beam dynamics equations, how to transform the space charge fields from the bunch frame to the laboratory frame, and how to avoid artificial collisions between the charged particles.

5.1 Beam Dynamics Equations

The first step of tracking the motion of a charged particles in electromagnetic field is to find the proper dynamics equations of $(\vec{p}, \vec{x})$, where $\vec{p}$ represents the momentum and $\vec{x}$ represents the position of the particle. The differential equation for $\vec{x}$ can be simply written down as

$$\frac{d\vec{x}}{dt} = \frac{\vec{p}}{\gamma m}, \tag{5.1}$$

where γ is the Lorentz factor and m is the static mass of the particle. The differential equation of $\vec{v}$ can be derived from the Lorentz force formula and Newton's second law.

$$\frac{d\vec{p}}{dt} = \vec{F} = q\vec{E} + q\vec{v} \times \vec{B} = q\vec{E} + q\frac{\vec{p}}{\gamma m} \times \vec{B}, \tag{5.2}$$

where t is the time, and $\vec{E}$ and $\vec{B}$ are the electric and magnetic fields experienced by the particle. γ in Eq.(5.1) and Eq.(5.2) can be expressed as a function of p, such as

$$\gamma = \sqrt{1 + \frac{p^2}{m^2c^2}}, \tag{5.3}$$

where c is the speed of light. Using Eq.(5.3), catastrophic cancellation of significant digits can be avoided in the computation. Then, the dynamics equations are

$$\begin{aligned} \frac{d\vec{x}}{dt} &= \frac{\vec{p}}{m\sqrt{1 + \frac{p^2}{m^2c^2}}}, \\ \frac{d\vec{p}}{dt} &= q\vec{E} + \frac{q\vec{p}}{m\sqrt{1 + \frac{p^2}{m^2c^2}}} \times \vec{B}. \end{aligned} \tag{5.4}$$

In order to avoid too small or too large numbers in calculation, an appropriate choice of the units is necessary. For example, in most cases time t is a small number because of the fast speed of the particles, but we can multiple with a constant number, the speed of light c, to the time t and use meter as the unit, so that the scale of time is in a proper range. Divided by c in both sides, Eq.(5.4) becomes

$$\begin{aligned}\frac{d\vec{x}}{d\tau} &= \vec{\beta} = \frac{\vec{p}}{mc\sqrt{1+\frac{p^2}{m^2c^2}}},\\ \frac{d\vec{p}}{d\tau} &= \frac{q}{c}\vec{E} + \frac{q\vec{p}}{mc\sqrt{1+\frac{p^2}{m^2c^2}}} \times \vec{B},\end{aligned} \tag{5.5}$$

where $\tau = tc$ and $\vec{\beta} = \vec{v}/c$ is the dimensionless Lorentz factor. If one chooses the following units for the other variables,

$$\begin{aligned}&\vec{x} = \vec{\mathbf{x}}[\mathrm{m}] \quad \vec{p} = \vec{\mathbf{p}}[\lambda\mathrm{eV/c}], \quad q = \mathbf{q}[\mathrm{e}], \quad m = \mathbf{m}[\lambda\mathrm{eV/c^2}],\\ &\vec{E} = \vec{\mathbf{E}}[\lambda\mathrm{V/m}], \quad \vec{B} = \vec{\mathbf{B}}[\mathrm{T}] = \vec{\mathbf{B}}[\mathrm{Vs/m^2}], \quad c = \mathbf{c}[\lambda\mathrm{m/s}].\end{aligned} \tag{5.6}$$

where λ is one of the SI prefixes such as k,M, G, etc., substituting them into Eq.(5.5), all the units are canceled and one gets the dimensionless dynamics equations as Eq.(5.7) shows.

$$\begin{aligned}\frac{d\vec{\mathbf{x}}}{d\boldsymbol{\tau}} &= \vec{\beta} = \frac{\vec{\mathbf{p}}}{\mathbf{m}\sqrt{1+\frac{\mathbf{p}^2}{\mathbf{m}^2}}} = \frac{\vec{\mathbf{p}}}{\gamma\mathbf{m}},\\ \frac{d\vec{\mathbf{p}}}{d\boldsymbol{\tau}} &= \mathbf{q}\vec{\mathbf{E}} + \mathbf{qc}\frac{\vec{\mathbf{p}}}{\mathbf{m}\sqrt{1+\frac{\mathbf{p}^2}{\mathbf{m}^2}}} \times \vec{\mathbf{B}} = \mathbf{q}\left(\vec{\mathbf{E}} + \mathbf{c}\vec{\beta} \times \vec{\mathbf{B}}\right).\end{aligned} \tag{5.7}$$

In the above, one coefficient λ is used to control the scale. In our simulation, we choose $\lambda = 10^6$. Then the units are

$$\begin{aligned}&\vec{x} = \vec{\mathbf{x}}[\mathrm{m}] \quad \vec{p} = \vec{\mathbf{p}}[\mathrm{MeV/c}], \quad q = \mathbf{q}[\mathrm{e}], \quad m = \mathbf{m}[\mathrm{MeV/c^2}],\\ &\vec{E} = \vec{\mathbf{E}}[\mathrm{MV/m}], \quad \vec{B} = \vec{\mathbf{B}}[\mathrm{T}] = \vec{\mathbf{B}}[\mathrm{Vs/m^2}], \quad c = \mathbf{c}[\mathrm{Mm/s}].\end{aligned} \tag{5.8}$$

5.2 The Lorentz Transformation of the Space Charge Field

In Eq.(5.7), all the physical quantities, including $\vec{E}$ and $\vec{B}$, are measured in the laboratory frame. $\vec{E}$ and $\vec{B}$ usually have two components, the external field and the space charge field. The external field is often calculated in the laboratory frame. But the space charge field is calculated as the static Coulomb field in the bunch frame, which moves together with the bunch, under the assumption that all the particles have similar momenta. If the bunch moves with a speed close to the speed of light, we have to consider the relativistic effects and convert the space charge field from the bunch frame into the laboratory frame by the Lorentz transform.

We refer to the laboratory frame as Σ and the bunch frame Σ'. All the physical quantities measured in Σ are represented by an alphabet without prime and all that measured in Σ' with a prime. For example, $x,\ y,$ and z are position coordinates in Σ and $x',\ y'$, and z' are position coordinates in Σ'. Assume in Σ the bunch moves with velocity $\vec{v}_z$ in z direction, which in practice can be chosen as the average velocity of all particles in z direction, then we have

$$\begin{aligned} x' &= x, \\ y' &= y, \\ z' &= \gamma \cdot (z - v_z t), \end{aligned} \tag{5.9}$$

where $\gamma = \sqrt{1 - v_z^2/c^2}$. (x', y', z') is used to calculate the Coulomb field between the particles using the DA based MLFMA in Σ'. It is apparent that $\vec{B}' = 0$ in Σ'. Then we convert $\vec{E}'$

in Σ' into $\vec{E}$ and $\vec{B}$ in Σ by Eq. (5.10).

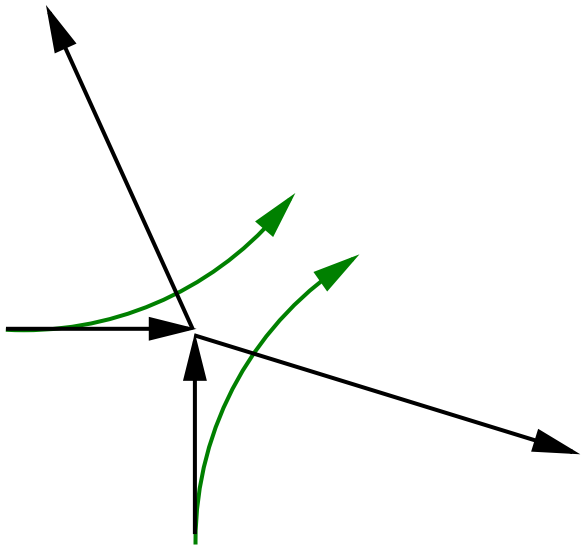
$$\begin{aligned} E_x &= \gamma E'_x, \qquad B_x = -\gamma \frac{v_z}{c^2} \cdot E'_y, \\ E_y &= \gamma E'_y, \qquad B_y = \gamma \frac{v_z}{c^2} \cdot E'_x, \\ E_z &= E'_z, \qquad B_z = 0, \end{aligned} \tag{5.10}$$

where c is the speed of light. Knowing $\vec{E}$ and $\vec{B}$ we can integrate the beam dynamics equations in the laboratory frame.

5.3 Collisions Between Charged Particles in the Near Region

Since the dynamics equations are numerically solved on discrete time steps, an artificial collision between charged particles, as shown in Figure 5.1, could happen in the near region, where the field is calculated directly by the the Coulomb force law. In the real world, the charged particles feel a stronger repulsive force between them when they move closer and thus they follow the traces like the green curves in the picture. But when we integrate the dynamics equations in a discrete time step, the force between the pair of charged particles is calculated at the very beginning and is kept as a constant during the whole time step. The force is underestimated when the pair of particles move closer to each other. It could happen that in some extreme case the two particles get very close to each other, as the black lines show in the picture, and they will suddenly feel a very strong force in the next step, which leads to a sudden increasing of their momentum. And later they will leave the bunch. Actually when we simulate millions of particles, this artificial collision will not affect the

Figure 5.1: The artificial collision



collective behavior if it only happens on a few particles. But it may affect some statistical quantities, which are used to describe the collective behavior. For example, the r.m.s. value of the positions, which is often used to describe the bunch size, may be increased simply because the pair of particles in collision have left the bunch and contribute a huge distance to the center of the bunch. So we need to prevent this collision from happening in the simulation, or detect it when it happens and fix its effect. Another reason that we want to prevent this artificial collision is due to the use of macroparticles. In many cases, we still use one macroparticle to represent a group of particles to increase the efficiency, although the MLFMA is applied. The macroparticles are treated as point charges in the near region field calculation. However, as a group of particles, they in fact should have volume. When two macroparticles go extremely close to each other, the two groups of real particles overlap each other and the field between them will not go as high as the Coulomb force law calculates for two point charges.

One easy way to detect the collision is to check the momentum dispersion of the bunch.

The sudden increasing of the momentum of the particles experiencing collision often leads to the discontinuity in the momentum dispersion curve. Figure 5.2 shows an example, in which we simulate the free expansion of an electron bunch of 10,000 electrons for 10 ps in 200 steps. No external field is applied and only the space charge field makes effects in the expansion process. The dynamics equations are solved by the fourth order Runge-Kutta integrator with fixed step size. The r.m.s. value of the momentum and the minimum distance between all pairs of electrons are plotted in Figure 5.2. There are two jumps of the momentum dispersion curve and for each one a local minimum value of the distance between electrons can be found. We plot all the electrons in $z - p_z$ phase space, as shown in Figure 5.3, then we found one pair of electrons that have much larger momenta than all the others. These two electrons reached a very close distance in the last time step. Then in this time step, the field between them goes extremely high, thus they feel a much large impulse than the other electrons, which gives them much larger momenta than all the other electrons. While from the charge density plot in Figure 5.4 we can see most electrons have very small momenta and stay in the center of the space. The behavior of the two electrons do not affect the collective behavior of the whole bunch and they will leave the bunch soon if they do not collide with other electrons. But the values of their momenta are large enough to affect the r.m.s of the momenta of the whole bunch.

We have three methods to avoid this. First, when we calculate the statistical quantities of the whole bunch, we do not count the contribution of these two electrons. Second, we can set a cut-off rate γ. When we calculate the field by the DA based MLFMA, we know exactly how many particles each childless box holds. For each childless box, we calculate the average distance r_a. If the distance between two particles r is less than $\gamma \cdot r_a$, we use $\gamma \cdot r_a$ instead of r to calculate the field between these two particles. In this way we actually set a maximum

Figure 5.2: Discontinuity of the momentum dispersion due to the near region collision

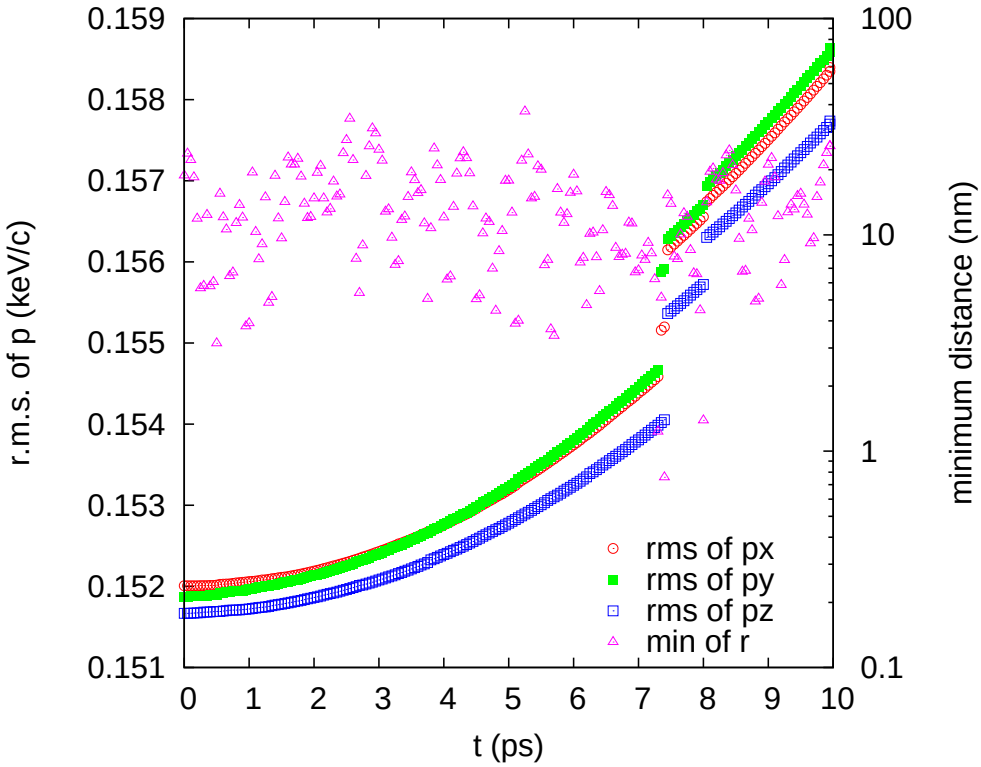


Figure 5.3: Electrons in $z - p_z$ phase space

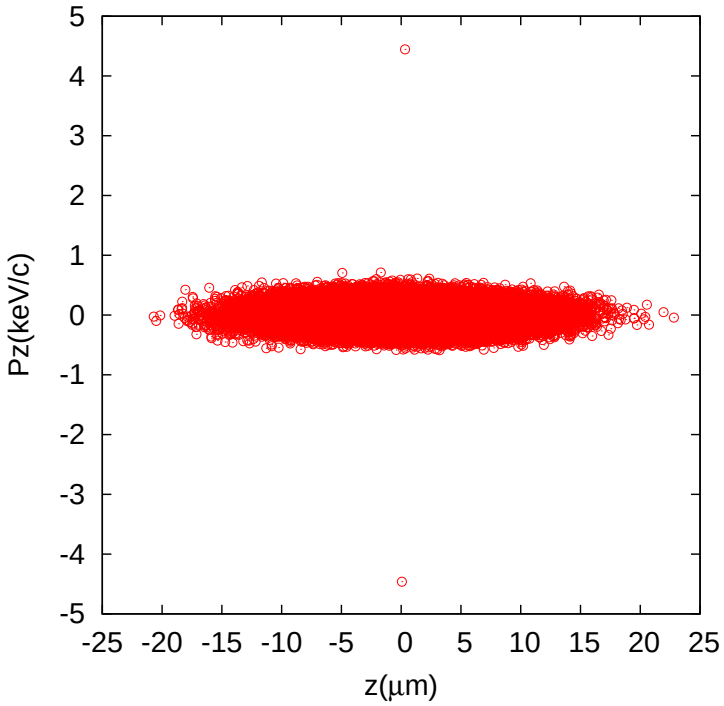


Figure 5.4: Electron density in $z - p_z$ phase space

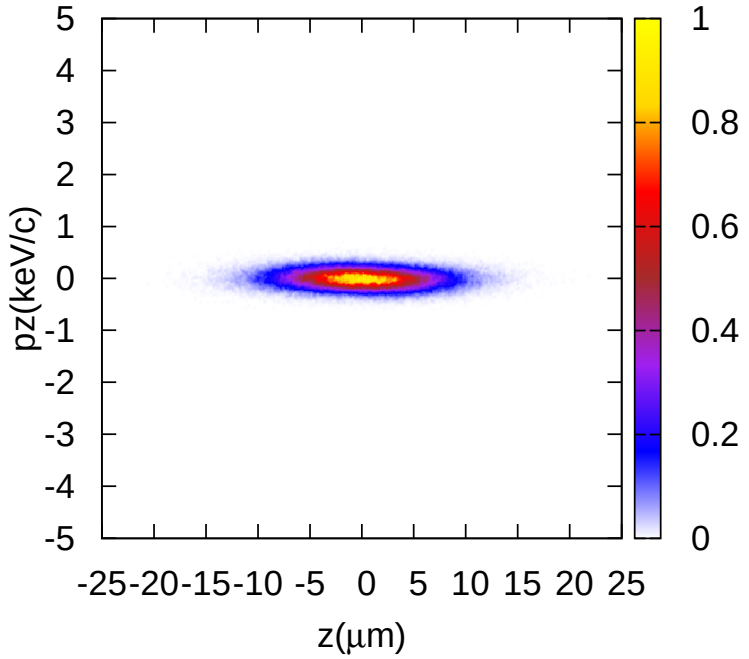


Figure 5.5: Momentum dispersion after applying the first method

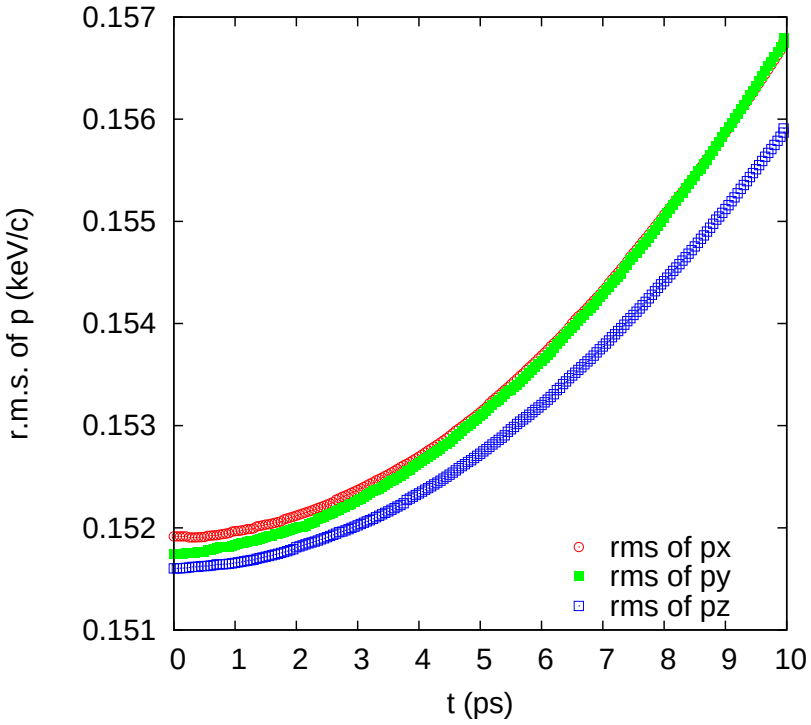


Figure 5.6: Momentum dispersion after applying the second method

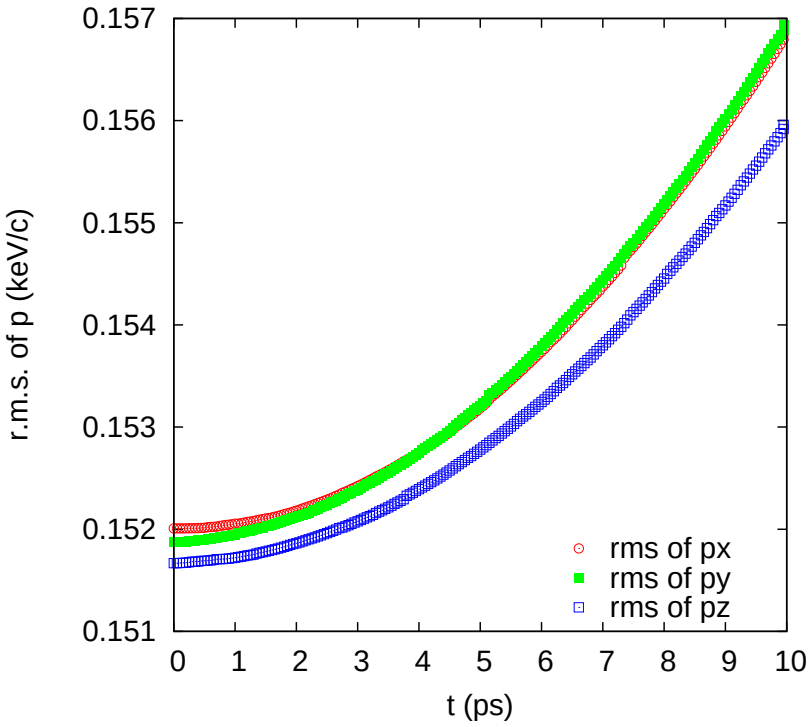
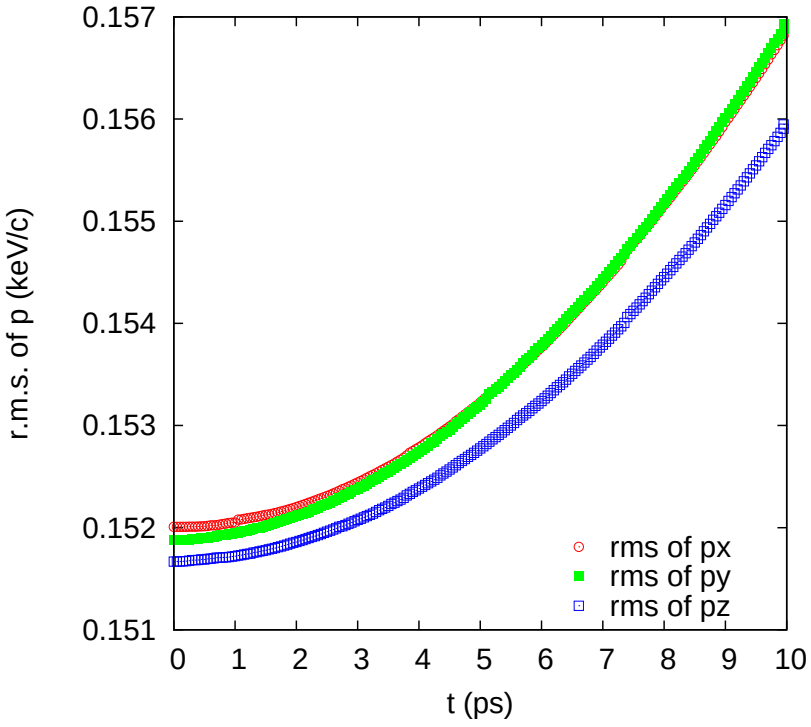


Figure 5.7: Momentum dispersion after applying the third method



force between each pair of particles for each childless box. Once the distance between a pair of particles is less than the cut-off distance $\gamma \cdot r_a$, we use the maximum force to replace the real force. Third, similarly to the second, when the distance between two particles r is less than $\gamma \cdot r_a$, we use a three dimensional Gaussian distribution model, instead of the point charge model, to represent the source particle. When the observer particle gets closer to the center of the Gaussian distribution, the force it feels goes to zero. Figure 5.5, Figure 5.6 and Figure 5.7 respectively show the momentum dispersion curve after applying the above method. In all the three cases the discontinuity disappears. In the last two simulations, γ is chosen to be 0.001. The time step size need to be considered for the proper choice of γ, because the change of the momentum depends on the impulse. Besides the above methods, an integrator with automatic step size control can be employed to avoid this issue. One needs to carefully choose the criteria of the step size control.

5.4 An Example of Photoemission Process Simulation

5.4.1 Model of the Photoemission Process

When an ultrafast laser pulse is applied on a gold film, electrons are generated. The the motion of the electrons are dominated by the space charge field between them. We simulated this process and the DA based MLFMA was used to calculate the space charge field. The following model is used in the simulation. We let the gold film lie in the $x - y$ plane, and z direction is perpendicular to the film. The incident laser pulse is applied on the surface of the gold film and the electrons come out of the surface after colliding with the photons and absorbing their energy. Assume we know the Fermi energy of the gold film E_f, the work function of the gold film W and the photon energy $E_{ph} = h\nu$ with ν the laser frequency.

The highest possible energy of an electron inside the gold film is the Fermi energy E_f. We assume the electron absorb all the energy of the photon after their collision. If the initial energy of an electron is E_i, after the collision its energy is $E_i + E_{ph}$. If this electron gets enough energy to overcome the work function and escape from the gold film, we should have $E_i + E_{ph} - W > E_f$. So the lower bound of the energy for an electron to escape is $E_f - E_{ph} + W$. The higher bound is of course E_f. We assume the electrons have a uniform distribution within the energy band of $[E_f - E_{ph} + W, E_f]$. After the collision, the energy of the electron E_e is within the range of $[E_f + W, E_f + E_{ph}]$. And the velocity of the electron satisfies

$$v = \sqrt{\frac{2E_e}{m}}, \tag{5.11}$$

where m is the electron mass. To overcome the work function, the electron's velocity in z direction should be at least $mv^2/2 = E_f + W$, so that

$$v_{z,min} = \sqrt{\frac{2(E_f + W)}{m}}. \tag{5.12}$$

This implies that for a given energy $E_e \in [E_f + W, E_f + E_{ph}]$, only the electrons whose velocities are inside an incident cone of the angle θ_{max} with respect to the z direction can escape, where

$$\cos\theta_{max} = \frac{v_{z,min}}{v} = \sqrt{\frac{E_f + W}{E_e}}. \tag{5.13}$$

From Eq. (5.13) we can see a smaller E_e leads to a smaller θ, and therefore a smaller cone. If we assume the electron moves in a random direction after its collision with the photon, Eq. (5.13) indicates the electrons with lower energies have lower possibilities to escape. The possibility is in proportion to the solid angle of the cone, which is $2\pi(1 - \cos\theta_{max})$. The

solid angle of a sphere is 4π, so we have the possibility to escape as

$$P_{esc.} = \frac{2\pi(1-\cos\theta_{max})}{4\pi} = 0.5 \cdot (1-\cos\theta_{max}),$$

where θ_{max} is always a sharp angle. With a given energy E_e and a given outgoing angle θ, the speed of the electron can be calculated by Eq. (5.11), and then the velocity in z direction is

$$v_z = v \cdot \cos\theta. \tag{5.14}$$

Given another angle $\phi \in [0, 2\pi]$, in the transverse directions we have

$$v_x = v \cdot \sin\theta \cdot \cos\phi, \text{ and } v_y = v \cdot \sin\theta \cdot \sin\phi. \tag{5.15}$$

Eq.(5.14) and Eq. (5.15) are the velocities on the metal side. If we assume the work function is abrupt at the surface, the initial velocities of the electrons in the air side are

$$v_{xi} = v_x,\ v_{yi} = v_y, \text{ and } v_{zi} = \sqrt{v_z^2 - 2(E_f + W)/m}. \tag{5.16}$$

The algorithm of how we sample the electrons and simulate their motion is described as follows. Note that if we do not specify the distribution of a random number, it has the uniform distribution.

1. Assume the laser pulse has a Gaussian distribution with standard deviation σ in the time domain. We cut the range of $[-3\sigma, +3\sigma]$ into N even parts. The i^{th} part covers from $t_1 = -3\sigma + (i-1) \cdot dt$ to $t_2 = -3\sigma + i \cdot dt$, with $dt = 6\sigma/N$. If the total number of electrons created during the photoemission process is N_e, the number of electrons created in the i^{th}

time interval is $(\mathrm{erf}(t_2/\sqrt{2}) - \mathrm{erf}(t_1/\sqrt{2}))/(2 \times 0.997)$.

2. For each outgoing electron, we generate its initial velocities and positions in the following way. (a) Generate a random number $E_e \in [E_f + W, E_f + E_{ph}]$, calculate $\cos\theta_{max}$ by Eq. (5.13), and then generate a random number $c \in [0, 1]$. (b) If $c > 1 - \cos\theta$, discard E_e and repeat step (a). Otherwise accept E_e and continue. (c) Generate a random number $a \in [0, \cos\theta_{max}]$ and another random number $\phi \in [0, 2\pi]$, let $\theta = \arccos a$, and calculate the initial velocities by Eq. (5.14), Eq. (5.15) and Eq. (5.16). (d) Generate two random numbers x_i and y_i of the Gaussian distributions with the standard deviations σ_x and σ_y respectively for the initial coordinates in x and y direction. (e) The initial coordinate in z direction is set to be zero.

3. For each outgoing electron, there is a positive residual left behind, assuming that other electrons do not have enough time to compensate it. For each electron coming out of the surface, there are three kinds of interactions: space charge field between the electrons, field from the positive holes and the external field that is applied to help extract the electrons out. The space charge field will be calculated by the DA based MLFMA. The external extracting field is constant. The model of the positive residual field will be described in Section 5.4.2.

4. Solve the dynamics equations for the outgoing electrons. Once the z coordinate of an electron is less than zero, remove it from our calculation.

5.4.2 Model of the Positive Residual Field

If we assume the incident laser pulse has a Gaussian distribution with standard deviation σ_{xy} in the transverse plane, the outgoing electrons have the same transverse distribution and so do the positive holes on the surface. Assume all positive residuals lie in a circular area whose radius is $6\sigma_{xy}$. We cut this area into N_s circular stripes whose width are all $6\sigma_{xy}/N_s$.

The radius of the inner side and the outer side of the i^{th} stripe is $r_{1i} = (i-1) \cdot 6\sigma_{xy}/N_s$ and $r_{2i} = i \cdot 6\sigma_{xy}/N_s$ respectively. If the positive residuals have a distribution on the surface of the form

$$f(r) = \frac{1}{2\pi\sigma_{xy}^2} \exp\left(-\frac{r^2}{2\sigma_{xy}^2}\right) \tag{5.17}$$

and the total charges are Q, then the charges inside a circular area with radius r_1 are

$$\int_0^{2\pi} d\theta \int_0^{r_1} f(r) \cdot r dr = 2\pi \int_0^{r_1} \frac{1}{2\pi\sigma_{xy}^2} \exp\left(-\frac{r^2}{2\sigma_{xy}^2}\right) \cdot r dr = 1 - \exp\left(-\frac{r_1^2}{2\sigma_{xy}^2}\right). \tag{5.18}$$

So that the charges inside the i^{th} stripe are

$$Q_i = Q\left[\exp\left(-\frac{r_{1i}^2}{2\sigma_{xy}^2}\right) - \exp\left(-\frac{r_{2i}^2}{2\sigma_{xy}^2}\right)\right]. \tag{5.19}$$

Then the whole area can be approximately represented by N_s rings with uniform charge distribution. The radius of the i^{th} ring is $r_i = 0.5 \cdot (r_{1i} + r_{2i})$ and the charge density of it is $\rho_i = Q_i/2\pi r_i$. The explicit formula for the three dimensional field of a uniformly charged ring can be derived as follows.

Assume we have a uniformly charged ring, whose center is (0,0,z_0) and whose radius is r_0, lying in the $z = z_0$ plate. Its charge density is a constant number ρ. Using cylindrical

coordinate, the electric field on an arbitrary point $(r,\, \theta,\, z)$ can be written as

$$\begin{aligned}
\phi &= \int_0^{2\pi} \frac{\rho r_0 d\theta_0}{\sqrt{(z-z_0)^2 + r^2 + r_0^2 - 2rr_0\cos(\theta_0 - \theta)}} \\
&= \int_0^{2\pi} \frac{\rho r_0 d\psi}{\sqrt{(z-z_0)^2 + r^2 + r_0^2 - 2rr_0\cos\psi}} \\
&= 2\int_0^{\pi} \frac{\rho r_0 d\psi}{\sqrt{(z-z_0)^2 + r^2 + r_0^2 - 2rr_0\cos\psi}},
\end{aligned} \tag{5.20}$$

where $\psi = (\theta_0 - \theta)$ and the symmetry is used in the last step. Let

$$R^2 = (r + r_0)^2 + (z - z_0)^2, \tag{5.21}$$

$$k^2 = 4rr_0/R^2. \tag{5.22}$$

We have

$$\begin{aligned}
& (z-z_0)^2 + r^2 + r_0^2 - 2rr_0\cos\psi \\
= \;& R^2 - 2rr_0 - 2rr_0\cos\psi \\
= \;& R^2 - 2rr_0(1 + \cos\psi) \\
= \;& R^2 - 4rr_0\sin^2\psi' \\
= \;& R^2(1 - k^2\sin^2\psi'),
\end{aligned} \tag{5.23}$$

where $\psi = \pi - 2\psi'$ and $1 + \cos\psi = 2\sin^2\psi'$. Then the potential in Eq. (5.20) can be

expressed as follows.

$$\begin{aligned}
\phi &= 2\int_0^{\pi} \frac{\rho r_0 d\psi}{R\sqrt{1-k^2\sin^2\psi'}} \\
&= 2\int_{\pi/2}^{0} \frac{\rho r_0(-2) d\psi'}{R\sqrt{1-k^2\sin^2\psi'}} \\
&= 4\int_0^{\pi/2} \frac{\rho r_0 d\psi'}{R\sqrt{1-k^2\sin^2\psi'}} \\
&= \frac{4\rho r_0}{R} K(k),
\end{aligned} \tag{5.24}$$

where $K(k)$ is the complete elliptic integral of the first kind, which is defined as

$$K(k) = \int_0^{\pi/2} \frac{d\alpha}{\sqrt{1-k^2\sin^2\alpha}}. \tag{5.25}$$

The complete elliptic integral of second kind $E(k)$ can be defined as

$$E(k) = \int_0^{\pi/2} \sqrt{1-k^2\sin^2\alpha} d\alpha, \tag{5.26}$$

and $K(k)$ and $E(k)$ have the following relations in Eq. (5.27) and Eq. (5.28), which is going to be used in our derivation.

$$\frac{dK}{dk} = \frac{E}{k(1-k^2)}, \tag{5.27}$$

$$\frac{dE}{dk} = \frac{E}{k} - \frac{K}{k}. \tag{5.28}$$

In cylindrical coordinates, we have

$$\vec{\nabla}\phi = \frac{\partial\phi}{\partial\rho}\hat{\rho} + \frac{1}{\rho}\frac{\partial\phi}{\partial\theta}\hat{\theta} + \frac{\partial\phi}{\partial z}\hat{z}. \tag{5.29}$$

So that

$$\begin{aligned} E_z &= -\frac{\partial\phi}{\partial z} = -\frac{\partial}{\partial z}\left(\frac{4\rho r_0}{R}K(k)\right) \\ &= -4\rho r_0\left[-\frac{1}{R^2}K\frac{\partial R}{\partial z} + \frac{1}{R}\frac{\partial K}{\partial k}\frac{\partial}{\partial z}\left(\frac{\sqrt{4rr_0}}{R}\right)\right] \\ &= \frac{4\rho r_0}{R^2}\left[K + \frac{1}{R}\frac{\partial K}{\partial k}\sqrt{4rr_0}\right]\frac{\partial R}{\partial z} \\ &= \frac{4\rho r_0}{R^2}\left[K + \left(\frac{E}{1-k^2} - K\right)\right]\frac{z-z_0}{R} \\ &= \frac{4\rho r_0}{R^3}\frac{E}{1-k^2}(z-z_0), \end{aligned} \tag{5.30}$$

and

$$\begin{aligned} E_r &= -\frac{\partial\phi}{\partial r} = -\frac{\partial}{\partial r}\left(\frac{4\rho r_0}{R}K(k)\right) \\ &= -4\rho r_0\frac{\partial}{\partial r}\left(\frac{K(k)}{R}\right) \\ &= -4\rho r_0\left[-\frac{1}{R^2}K\frac{\partial R}{\partial r} + \frac{1}{R}\frac{\partial K}{\partial k}\frac{\partial}{\partial k}\left(\frac{\sqrt{4rr_0}}{R}\right)\right] \\ &= \frac{4\rho r_0}{R^2}\left[K\frac{\partial R}{\partial r} + \frac{1}{R}\frac{\partial K}{\partial k}\left(\sqrt{4rr_0}\frac{\partial R}{\partial r}\right) - \frac{\partial K}{\partial k}\sqrt{\frac{r_0}{r}}\right] \\ &= \frac{4\rho r_0}{R^2}\left[\left(K + \frac{1}{R}\frac{\partial K}{\partial k}\sqrt{4rr_0}\right)\frac{\partial R}{\partial r} - \frac{\partial K}{\partial k}\sqrt{\frac{r_0}{r}}\right] \\ &= \frac{4\rho r_0}{R^2}\left[\frac{E}{1-k^2}\frac{r+r_0}{R} - \sqrt{\frac{r_0}{r}}\left(\frac{E}{(1-k^2)k} - \frac{K}{k}\right)\right] \\ &= \frac{4\rho r_0}{R^2}\left[\frac{E}{1-k^2}\frac{r+r_0}{R} - \frac{R}{2r}\left(\frac{E}{1-k^2} - K\right)\right] \\ &= \frac{4\rho r_0}{rR^2}\left[\frac{E}{1-k^2}\frac{r^2+rr_0}{R} - \frac{R}{2}\left(\frac{E}{1-k^2} - K\right)\right]. \end{aligned} \tag{5.31}$$

In Eq. (5.30) and Eq. (5.31), relations as shown in Eq. (5.27), Eq. (5.28) and Eq. (5.32-5.35)

are used.

$$\frac{\partial R}{\partial z} = \frac{z - z_0}{R} \tag{5.32}$$

$$RK = \sqrt{4rr_0} \tag{5.33}$$

$$\frac{\partial R}{\partial r} = \frac{r + r_0}{R} \tag{5.34}$$

$$\frac{1}{k}\sqrt{\frac{r_0}{r}} = \frac{R}{2r} \tag{5.35}$$

Now we can calculate the three dimensional field of a uniformly charged ring by Eq. (5.30) and Eq. (5.31), in which the elliptic integral $K(k)$ and $E(k)$ can be calculated numerically by Chebyshev polynomial expansions[23]. To check the validity of Eq. (5.30) and Eq. (5.31), we let $r = 0$ and derive the formula for E_z and E_r on the axis, which is well known. When $r = 0$, $k = 0$, $R = \sqrt{r_0^2 + (z - z_0)^2}$, and $E(k) = E(0) = \pi/2$. From Eq. (5.30), we get

$$\begin{aligned} E_z|_{r=0} &= \frac{4\rho r_0}{(r_0^2 + (z - z_0))^{3/2}} \cdot \frac{\pi}{2} \cdot (z - z_0) \\ &= \frac{2\pi\rho r_0 (z - z_0)}{(r_0^2 + (z - z_0))^{3/2}}. \end{aligned} \tag{5.36}$$

From Eq. (5.31), we obtain

$$\begin{aligned} E_r &= \frac{4\rho r_0}{R^2}\left[\frac{E}{1 - k^2}\frac{r + r_0}{R} + \frac{R}{2}\left(\frac{K}{r} - \frac{E}{r(1 - k^2)}\right)\right] \\ &= \frac{4\rho r_0}{R^2}\left[\mathrm{I} + \mathrm{II}\right], \end{aligned} \tag{5.37}$$

where

$$\begin{aligned} \mathrm{I} &= \frac{E}{1-k^2}\frac{r+r_0}{R}, \qquad (5.38)\\ \mathrm{II} &= \frac{R}{2}\left(\frac{K}{r}-\frac{E}{r(1-k^2)}\right). \qquad (5.39) \end{aligned}$$

When $r=0$,

$$\mathrm{I}|_{r=0} = \frac{\pi r_0}{2\sqrt{r_0^2+(z-z_0)^2}}. \qquad (5.40)$$

The other part is[49]

$$\begin{aligned} \mathrm{II} &= \int_0^{\pi/2}\frac{d\alpha}{r\sqrt{1-k^2\sin^2\alpha}} - \int_0^{\pi/2}\frac{\sqrt{1-k^2\sin^2\alpha}}{r(1-k^2)}d\alpha \\ &= \frac{1}{r}\int_0^{\pi/2}\frac{k^2\sin^2\alpha-k^2}{(1-k^2)\sqrt{1-k^2\sin^2\alpha}}d\alpha \qquad (5.41)\\ &= \frac{4r_0}{(r+r_0)^2+(z-z_0)^2}\int_0^{\pi/2}\frac{\sin^2\alpha-1}{(1-k^2)\sqrt{1-k^2\sin^2\alpha}}d\alpha. \end{aligned}$$

When $r=0$, We have

$$\begin{aligned} \mathrm{II}|_{r=0} &= \frac{4r_0}{r_0^2+(z-z_0)^2}\int_0^{\pi/2}(\sin^2\alpha-1)d\alpha \\ &= -\frac{\pi r_0}{r_0^2+(z-z_0)^2}. \qquad (5.42) \end{aligned}$$

Then

$$E_r|_{r=0} = \mathrm{I}|_{r=0} + \mathrm{II}|_{r=0} = 0. \qquad (5.43)$$

The expressions of E_r and E_z on the axis in Eq.(5.43) and Eq.(5.36) are as expected.

In our simulation, we use 1000 stripes ($N_s = 1000$) to approximately calculate the field of the positive residuals on a 101×101 grid, which covers the area of $[0, 579\mu\mathrm{m}]\times[0, 579\mu\mathrm{m}]$.

Figure 5.8: Longitudinal field

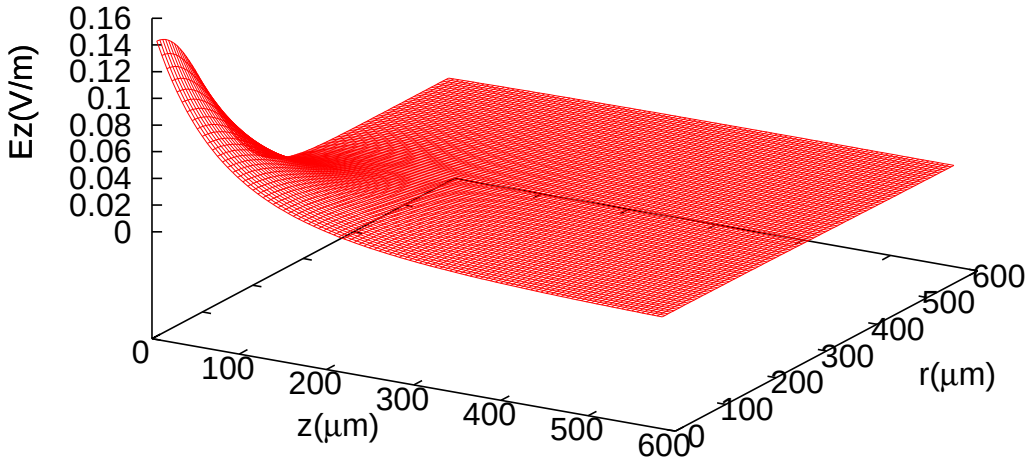
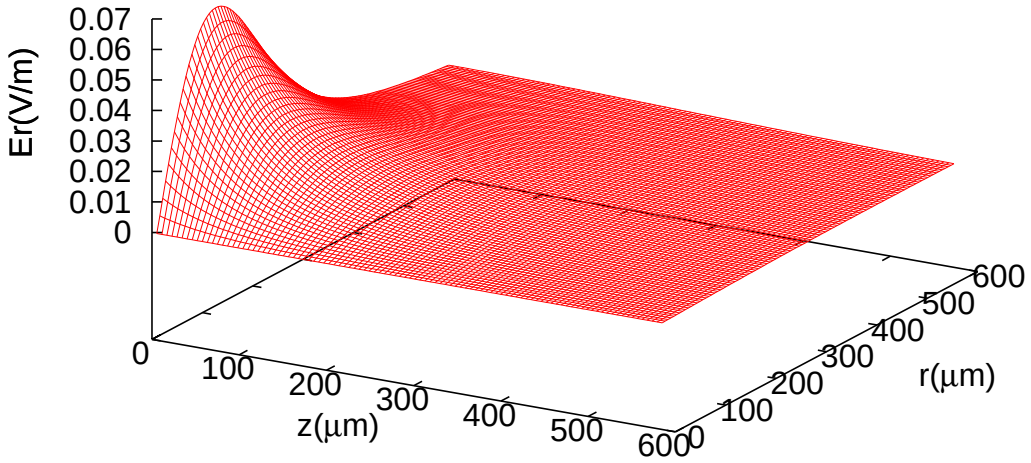


Figure 5.9: Radial field



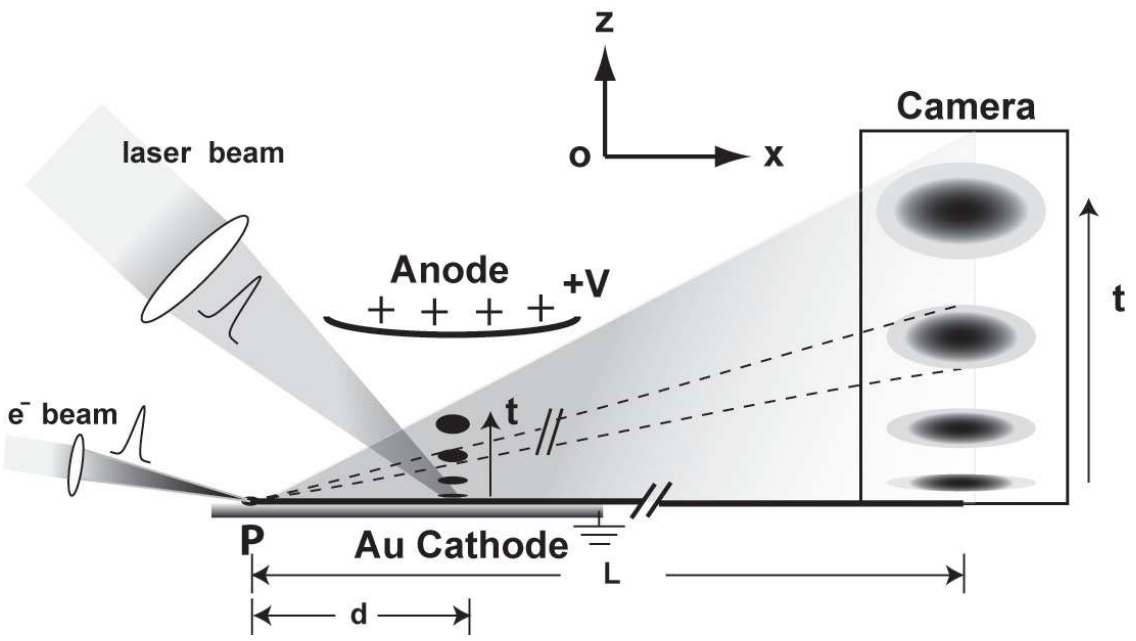
The longitudinal field and the radial field on the grid of one unit charge with Gaussian distribution on the surface are shown respectively in Figure 5.8 and Figure 5.9. We assume the charge distribution of the positive residuals does not charge, only the number of charges changes and it is always equal to the number of electrons outside the surface. So at any time,

fields always have the same shape as Figure 5.8 and Figure 5.9 show. But their strength changes with time. If we have N particles outside the surface, which leaves N positive residuals on the surface, the field on the grid of N residuals is just N times the field of a unit charge. The fields on the particles are calculated by linear interpolation.

5.4.3 Comparison of the Simulation Results with the Experiment Data

Our colleagues performed experiments to measure the ultrafast electron pulse dynamics immediately following photoemission[84, 85] using a point projection imaging techniques[75]. The setup of the measurement system is shown in Figure 5.10. A 50 fs laser pulse is applied on the gold photo cathode surface with a incidence of 45° to trigger the photoemission. The laser has a Gaussian profile with an elliptical cross-section having $\sigma_x = 115\mu$m and $\sigma_y = 81\mu$m. The wavelength of the laser is 266 nm, and the photon energy is 4.66 eV, which is slightly higher than the work function of gold film that ranges from 4.0-4.6 eV[83], allowing photoemission with a small energy spread. The exciting fluence of the laser pulse can be adjusted from 1-10 mJ/cm^2. A positive electrode (anode) is separated 5 mm from the cathode, providing an applied field (F_a) to facilitate the photoemission. The potential on the anode can be adjusted from 0-2400 V, which leads to the range of F_a 0-0.4 MV/m calculated by Field Precision, an electromagnetic field calculation program. The laser pulse together with the the applied extracting field can produce a wide range from 10^4 to 10^8 electrons/pulse. A point laser source is synchronized to the exciting laser beam with a well defined delay with respect to photoemission. Once the electron pulse is emitted, the point electron source costs a shadow of the electron pulse onto a metalized phosphor screen connected to an intensified

Figure 5.10: Experiment setup, from Zhensheng Tao

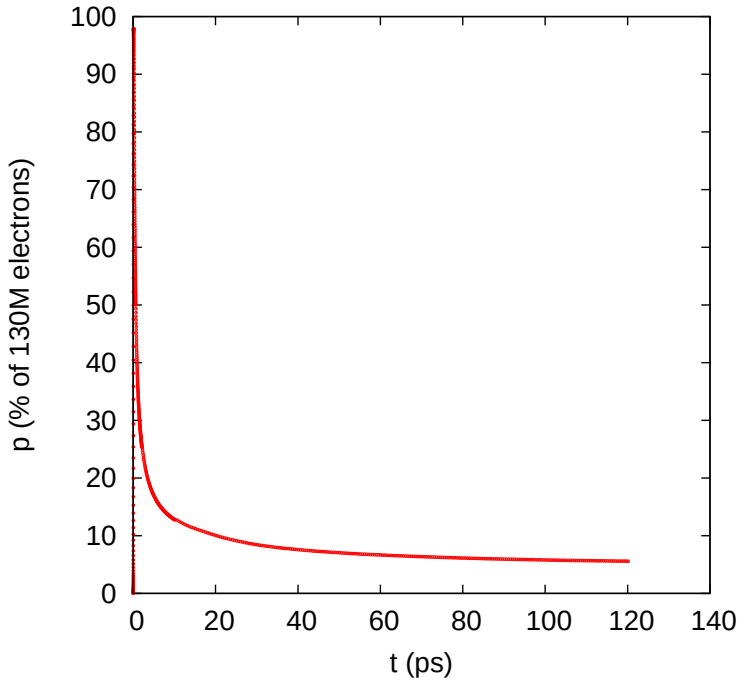


CCD camera. The position and the size of the bunch can be obtained by measuring the shadow patterns and fitting the experimental data with an analytical expression describing the projection geometry[75].

Here we compare a group of experimental data with simulation results. In experiment we set the extracting field F_a to be 0.32 MV/m and the fluence of the exciting laser 5mJ/cm^2, the electrons generated by photoemission is predicted to be in the order of 10^2[31] and the electrons that survived after 100 ps are measured to be in the order of 10^6. There is a two order's drop of the electron population. In the simulation, we set the Fermi energy to be 5 eV, photon energy 4.65 eV, work function 4.45 eV. The FWHM of the laser pulse is 50 fs, which means the standard deviation σ is 21.23 fs. To model the creation of the electrons, we divide the time region of 6σ into 100 each pieces. In total 130 million electrons are created, which are presented by 1.3 million macro-particles. Each macro-particle represents 100 electrons. Then we simulate the following 120 ps in 620 steps with various step sizes.

The dynamics equations are solved by COSY's intrinsic eighth order Runge-Kutta integrator. The space charge field between the macro-particles are calculated by the DA based MLFMA. A constant external field of 0.28 MV/m is applied to help extract the electrons. We save the number of particles, the bunch size, and the momentum dispersion in each time step, and save the positions and the momenta of all macro-particles in some selected steps. Figure 5.11 shows how the number of particles changes with respect to time. All the particles are created in the first 120 fs. The peak value of the percentage is slightly less than one, which means some particles go back to the surface almost immediately after they jump out. The number of particles keeps decreasing. In the first 10 ps, the percentage decreases to 10%. Then the curve becomes flatter. After 80 ps, the number of particles is almost a constant. After 100 ps there are 7.26 million electrons, which agrees with the experiment in order.

Figure 5.11: Number of particles as a function of time



The shadow pattern on the CCD screen can give us the information about the bunch size and the position of the bunch. The raw data of the the shadow pattern at 70 ps is shown in Figure 5.12. The horizontal axis represents the longitudinal position on the CCD screen and the vertical axis the depth of the shadow. The peak at about 0.008 m tells us where the bunch is in longitudinal direction and the shape of the peak give us information about the longitudinal bunch size. In the simulation, we divide the 600 μm distance in the longitudinal direction into 600 slots, each of which has the width of 1 μm. Then we count the particle number in each slot. To compare with the simulation result, the position data of the experiment is scaled as $x^{'} = (x - x(1))/33$, where x is the data before scaling, $x^{'}$ is the data after scaling and $x(1)$ is the value of the first point before scaling. The shadow depth data is scaled as $y^{'} = 5000 \cdot y$, where y is the data before scaling and $y^{'}$is the data after scaling. The simulation result and the experiment data at 70 ps, 80 ps, 90 ps and 100 ps are compared in Figure 5.13, Figure 5.14, Figure 5.15 and Figure 5.16. Ignoring the data points before 0.14 mm, the peak position, amplitude, and shape agree well in all cases. Figure 5.17 shows how the longitudinal bunch size evolves as time goes, in which the simulation and the experiment agree too. The quadratic curve, instead of a linear line, is as expected due to the space charge effect.

Figure 5.12: Raw data of the shadow patterns at 70 ps

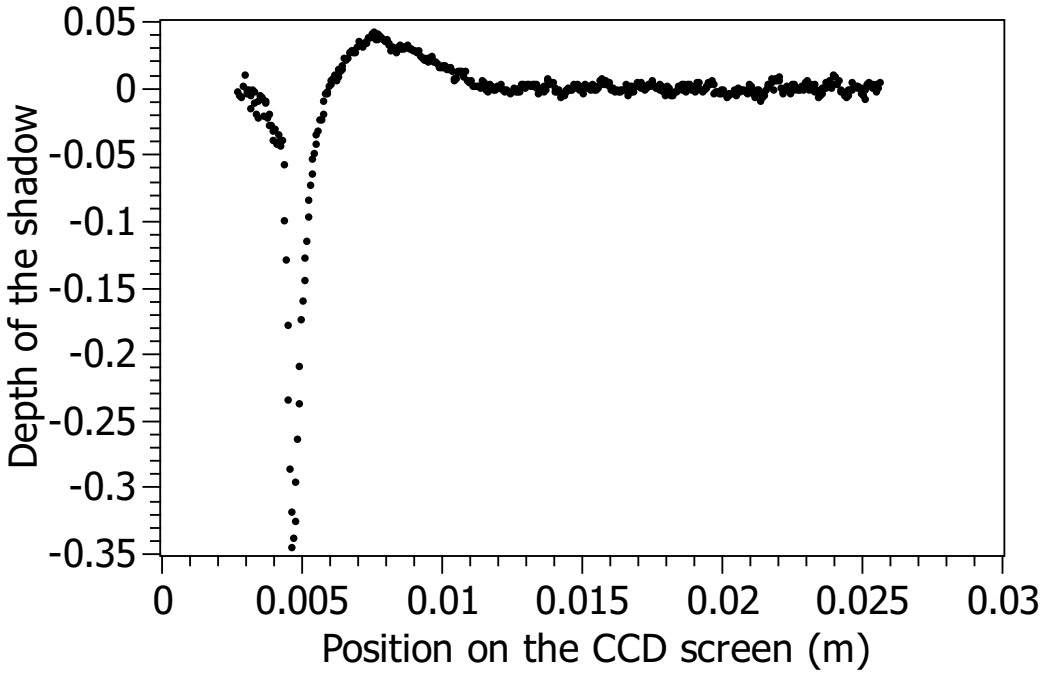


Figure 5.13: Comparison of the experiment with the simulation at 70 ps

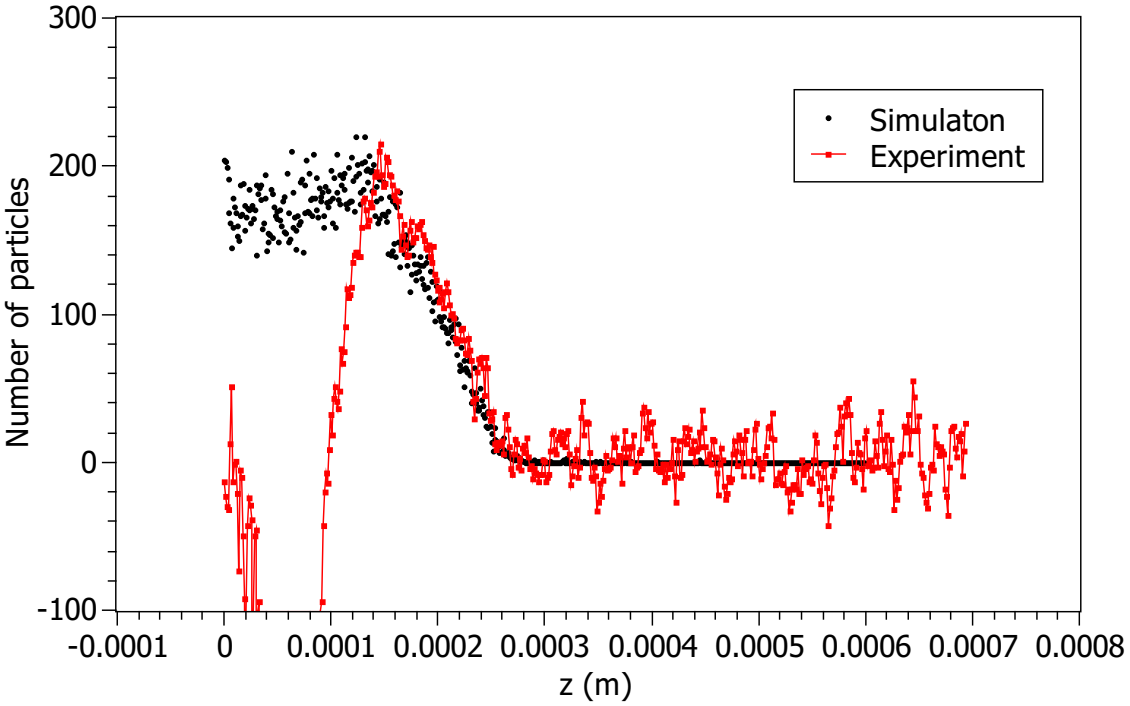


Figure 5.14: Comparison of the experiment with the simulation at 80 ps

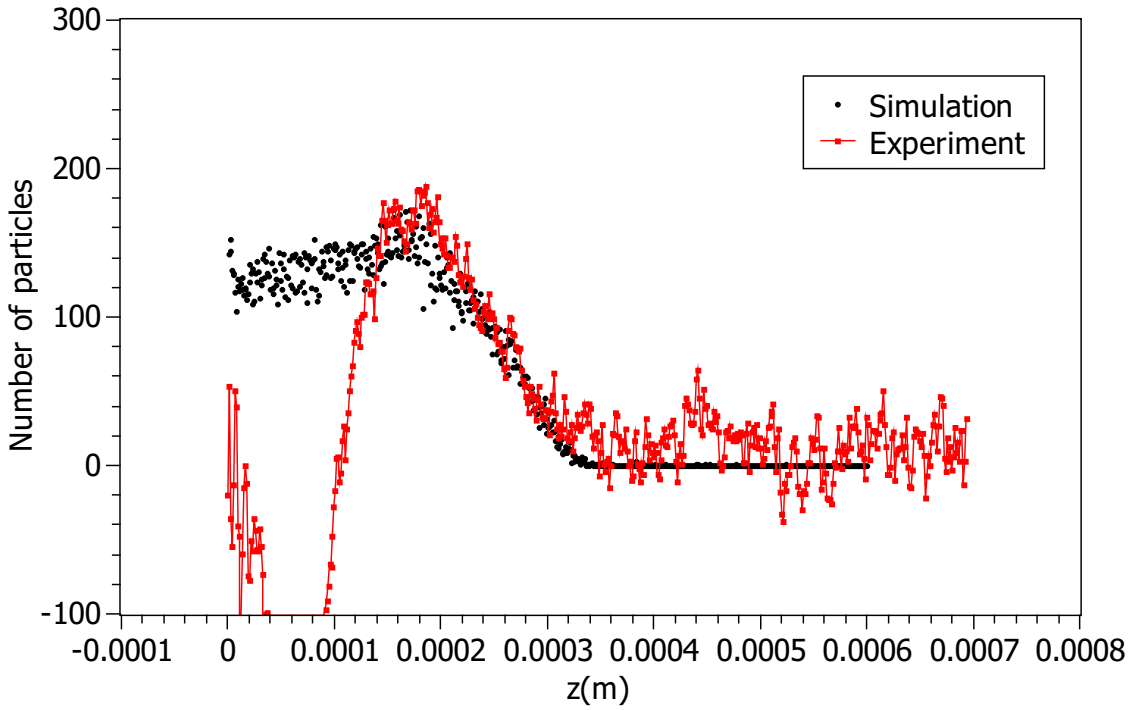


Figure 5.15: Comparison of the experiment with the simulation at 90 ps

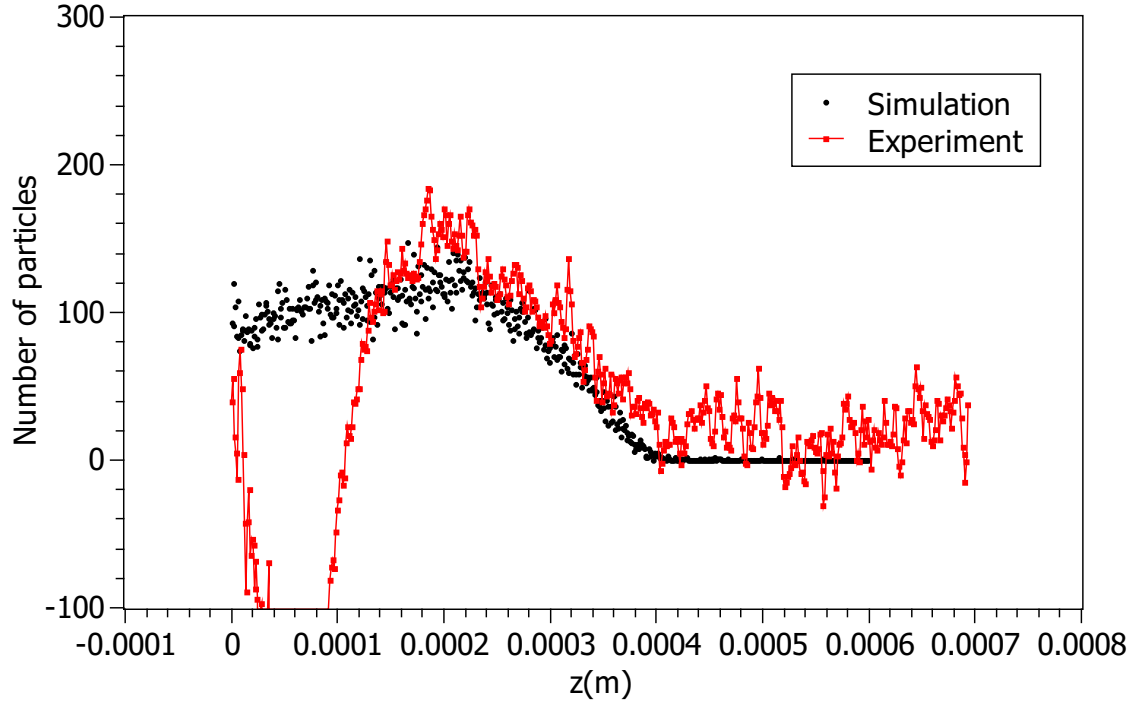


Figure 5.16: Comparison of the experiment with the simulation at 100 ps

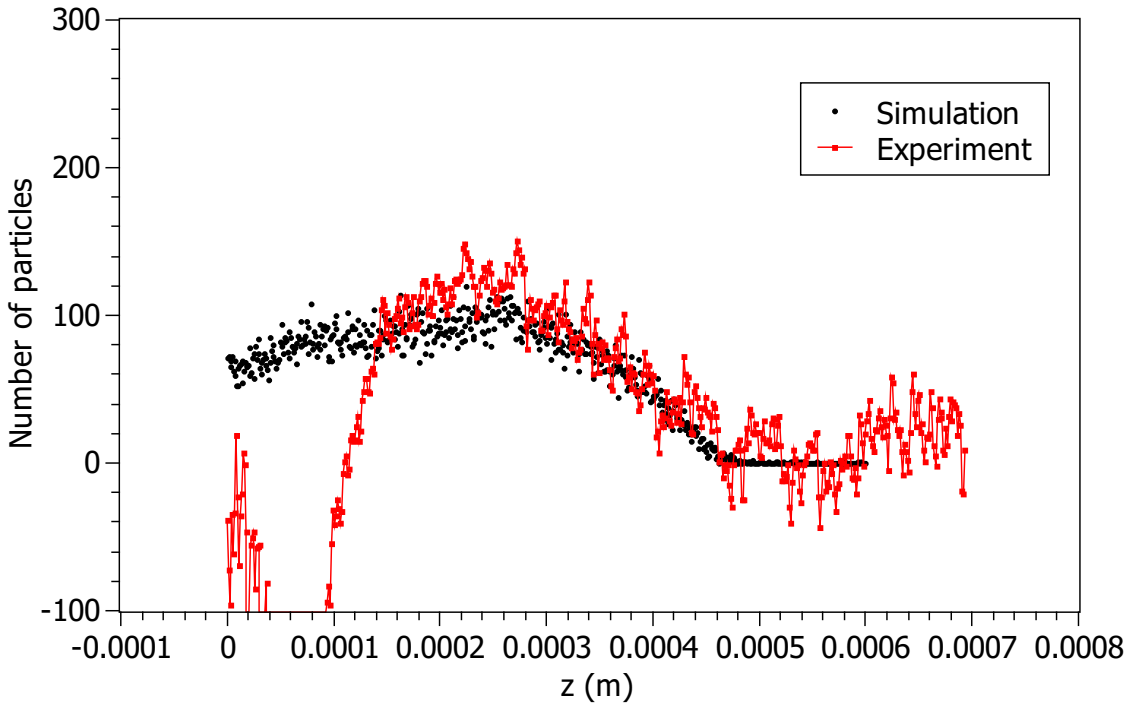


Figure 5.17: Comparison of the bunch sizes in the experiment and the simulation

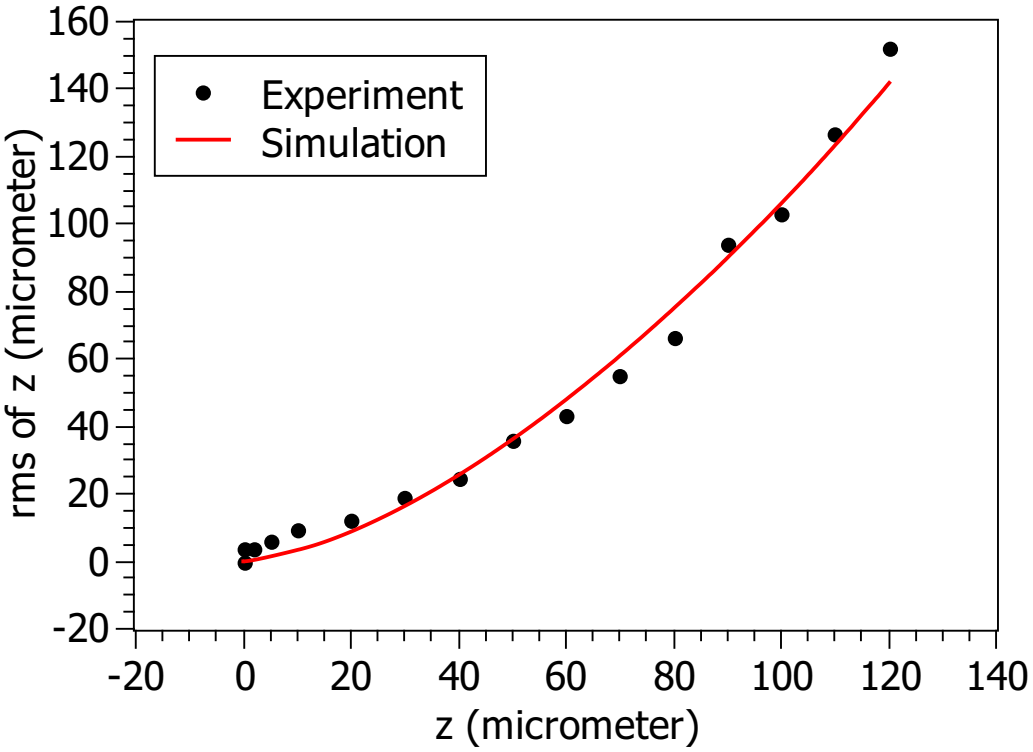
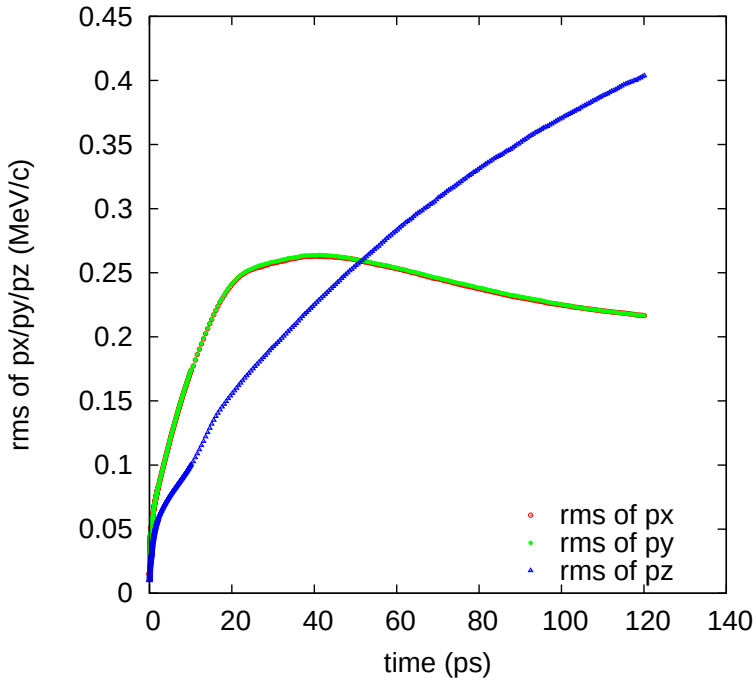


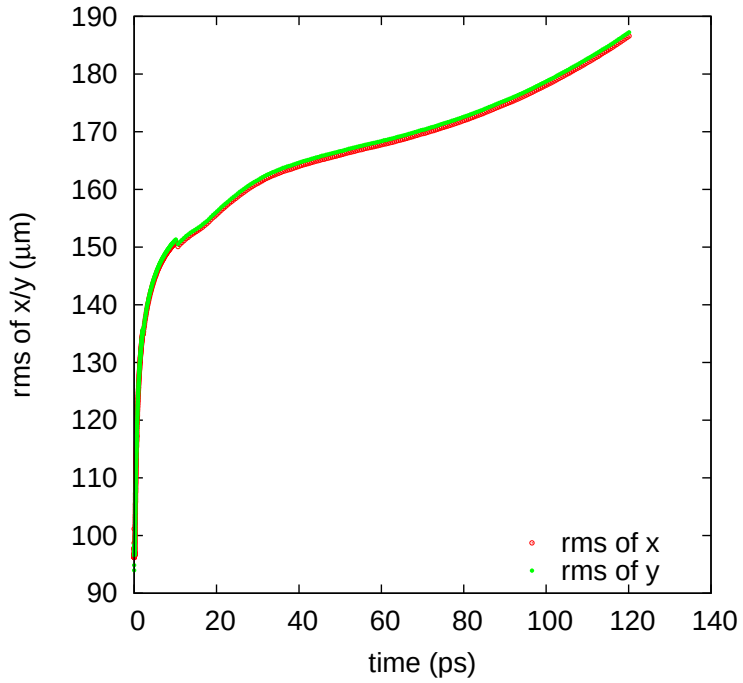
Figure 5.18: The momentum dispersion as a function of time



Simulations can give us some information that is difficult to measure directly in the experiments, for example the momentum dispersion. Figure 5.18 shows the momentum dispersion at different time in all the three directions. We are glad not to see any discontinuities. The curves for the two transverse directions are almost identical because the whole system has rotational symmetry in the simulation. For the same reason we also obtain two identical curves of the transverse bunch sizes in Figure. 5.19.

More interesting is the charge distribution in different normal space and phase space. Figure 5.20 shows the charge distribution in $x - z$ phase space in the first 130 fs, during which the laser pulse is applied on the surface and the electrons leave the surface and form the bunch. Different color means different densities, and the brighter the color is, the higher the density is. We can see how the electrons go out little by little, how they are accumulated and how the bunch is generated. The center of the bunch has a higher density than the edge has, which is reasonable since the Gaussian laser pulse has higher photon density at the

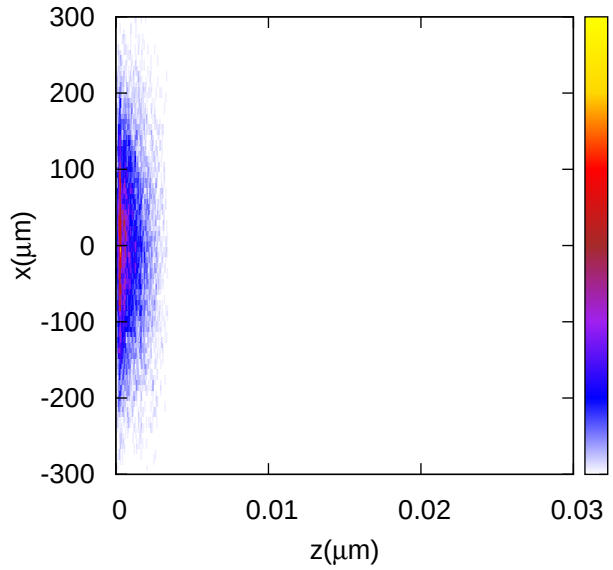
Figure 5.19: The transverse bunch sizes evolve with time



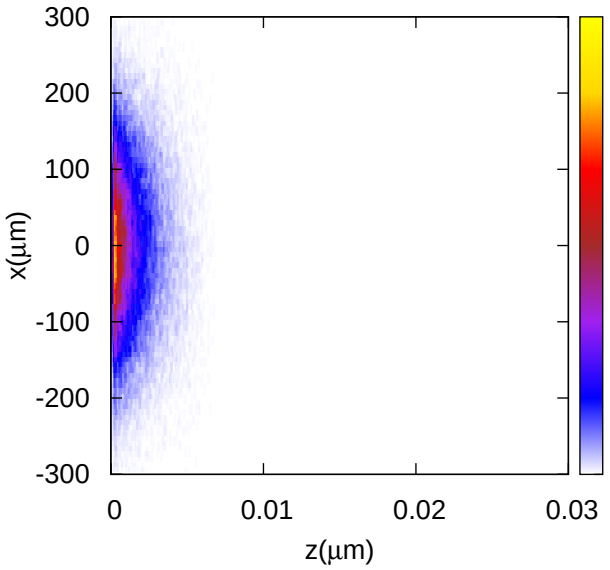
center too. Figure 5.21 shows the charge distribution in $z - p_z$ phase space in the first 130 fs. After the bunch is created, the charged particles form a triangle which covers the upper left area in the $z - p_z$ phase space. The discontinuity in the first a few figures is due to the discreteness in time of the integration.

Figure 5.20: Charge distribution in $x - z$ phase space in the first 130 fs

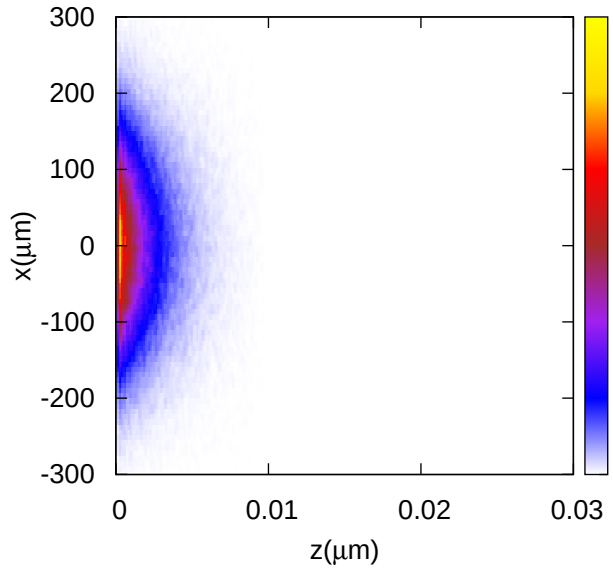
(a) t=13fs



(b) t=25fs



(c) t=38fs



(d) t=51fs

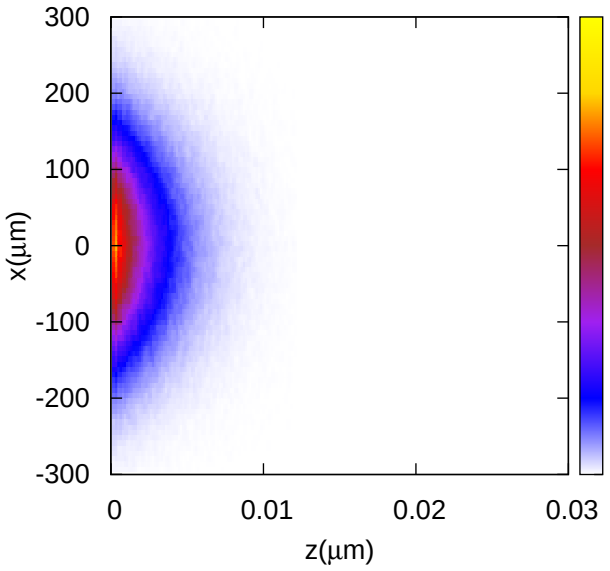
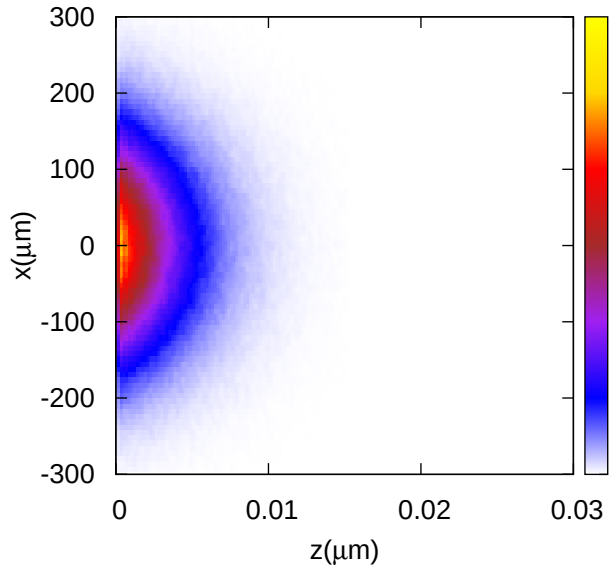
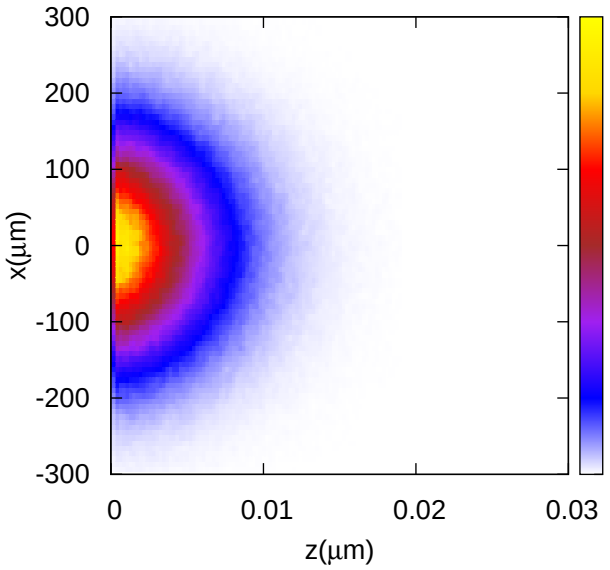


Figure 5.20: (cont'd)

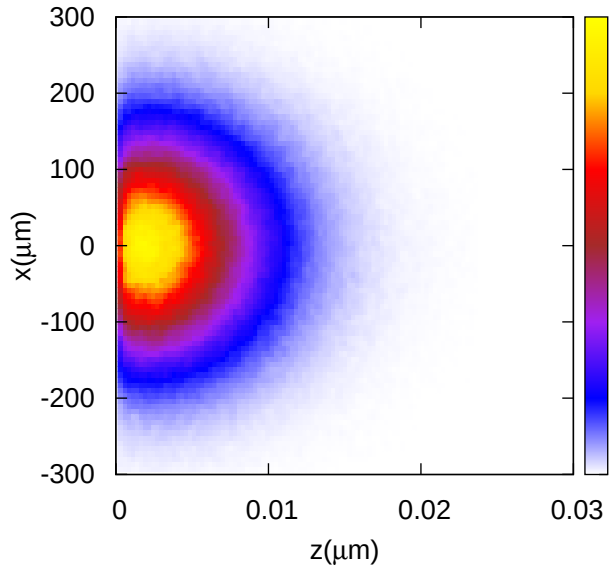
(e) t=63fs



(f) t=76fs



(g) t=89fs



(h) t=102fs

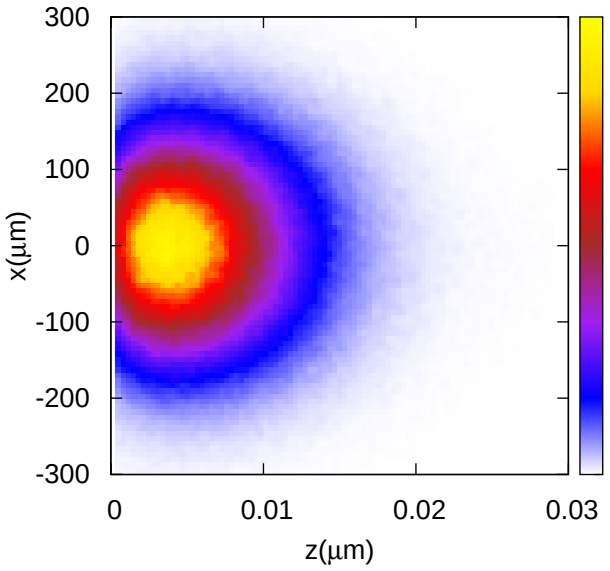
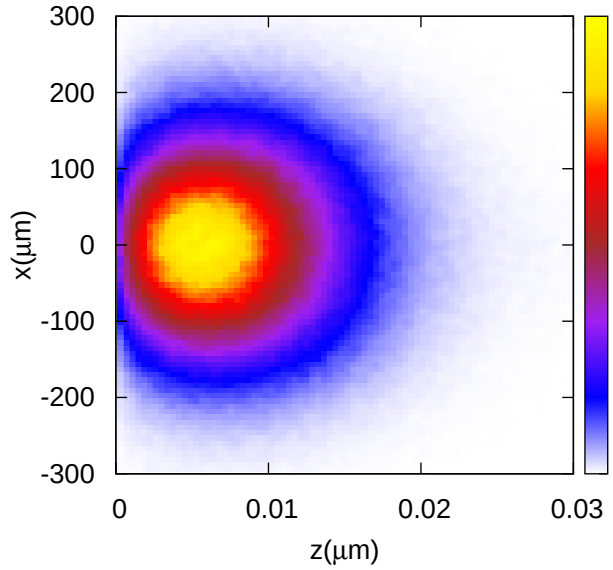


Figure 5.20: (cont'd)

(i) t=115fs



(j) t=127fs

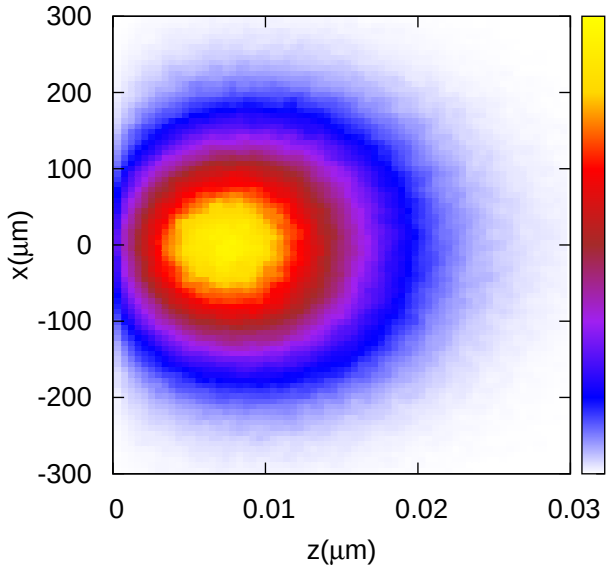
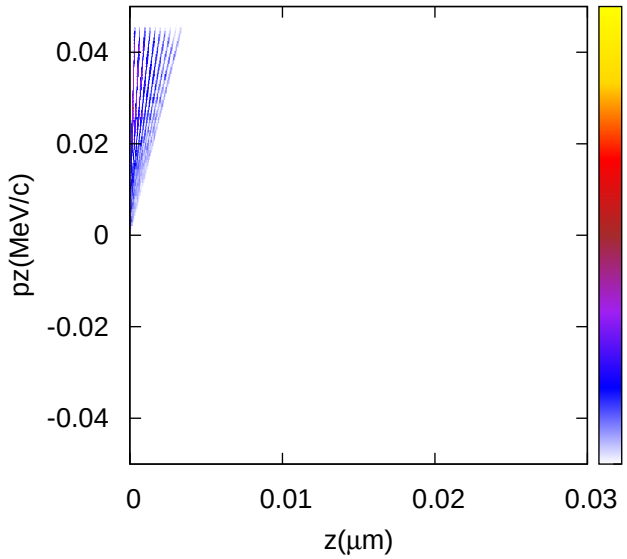


Figure 5.21: Charge distribution in $x - p_z$ phase space in the first 130 fs

(a) t=13fs



(b) t=25fs

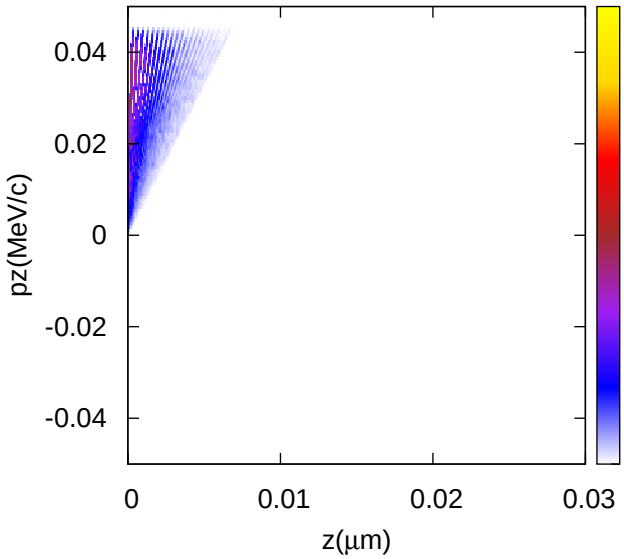
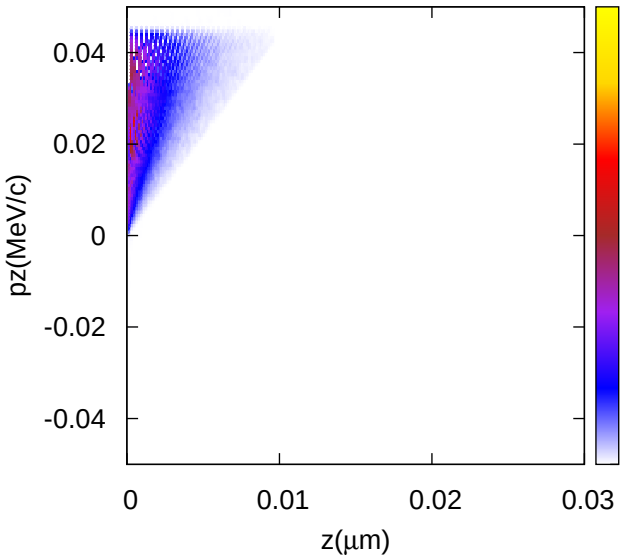
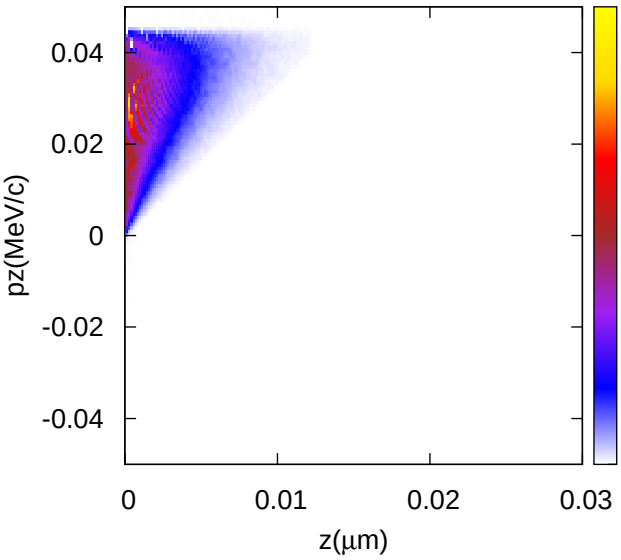


Figure 5.21: (cont'd)

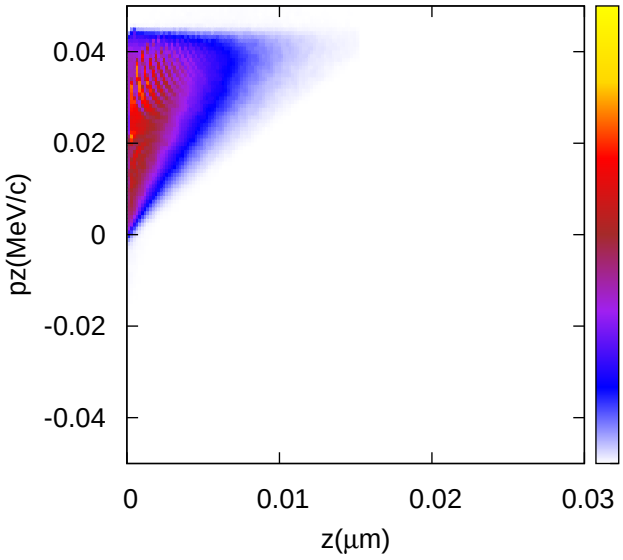
(c) t=38fs



(d) t=51fs



(e) t=63fs



(f) t=76fs

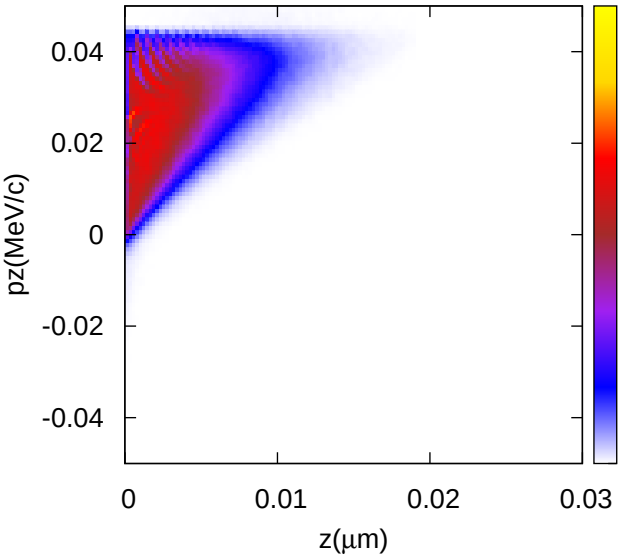
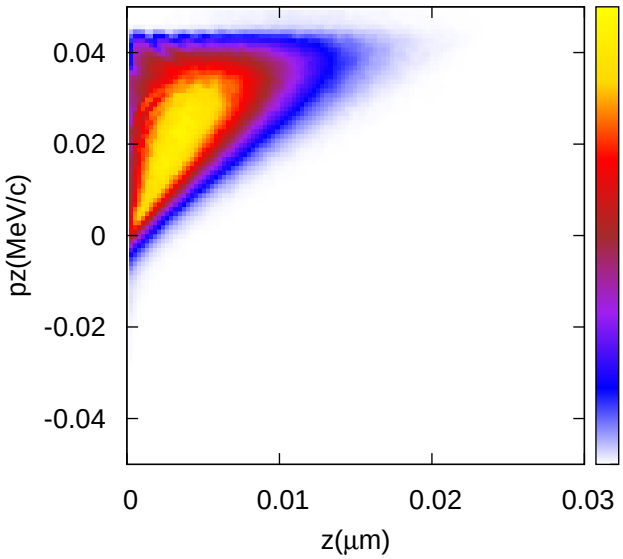
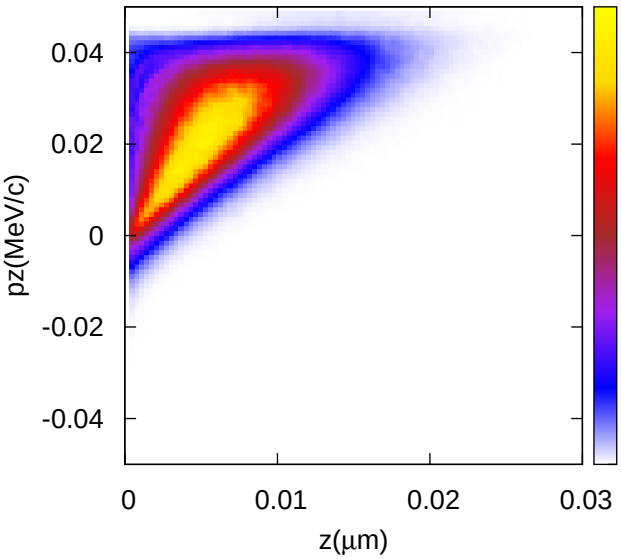


Figure 5.21: (cont'd)

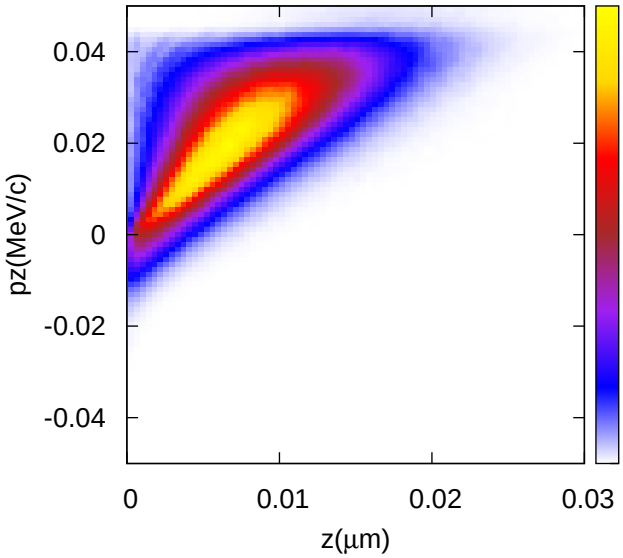
(g) t=89fs



(h) t=102fs



(i) t=115fs



(j) t=127fs

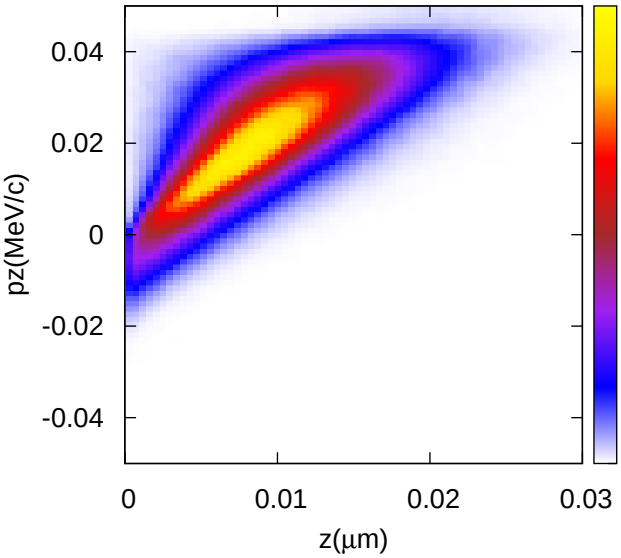
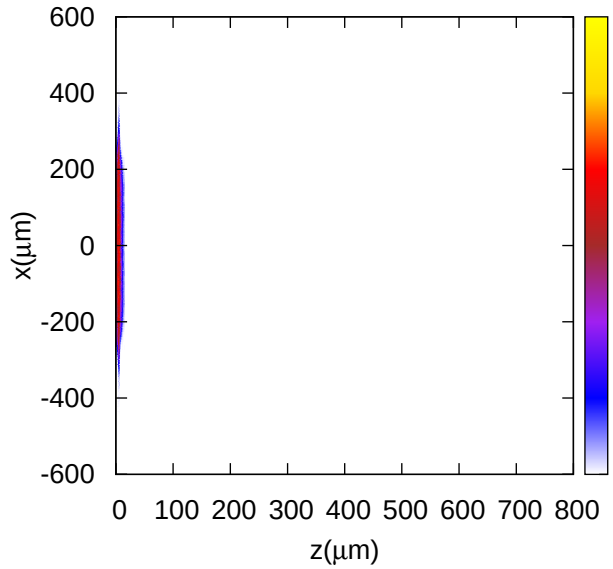


Figure 5.22 shows the charge distribution in $x - z$ phase space at different times in the following 120 ps, and Figure 5.23 shows the charge distribution in $x - y$ phase space. In the first 20 ps, the bunch is attached to the surface. During this period of time, the number of particles decreases very fast as shown in Figure 5.11. From the $x-y$ phase space distribution we can see that the bunch has the highest density in the center at 1 ps and then the charge density at the center is decreasing till 20 ps. This is because the charges at the center feel a stronger space charge field and a stronger field from the positive residuals. After 60 ps, the bunch starts to leave the surface and the number of particles is almost constant. In the $x - y$ phase space, most particles gather at the center again. Figure 5.24 shows the charge distribution in $z - p_z$ phase space. At 10 ps, a linear chirp has been formed. As time goes, the momentum difference between the two ends of the chirp increases due to the space charge field, which is as expected.

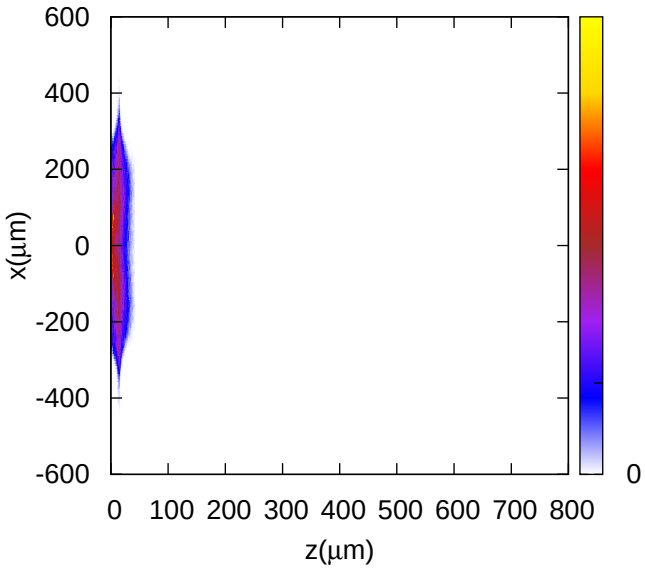
From this example, we see the DA based MLFMA works well although the charge density keeps changing. The simulation helps us understand the experimental data. Similar simulations can be used to study how the laser shape affects the photo electron distributions and how the electron bunch evolves under different initial conditions and different external fields.

Figure 5.22: Charge distribution in $x - z$ phase space in the following 120 ps

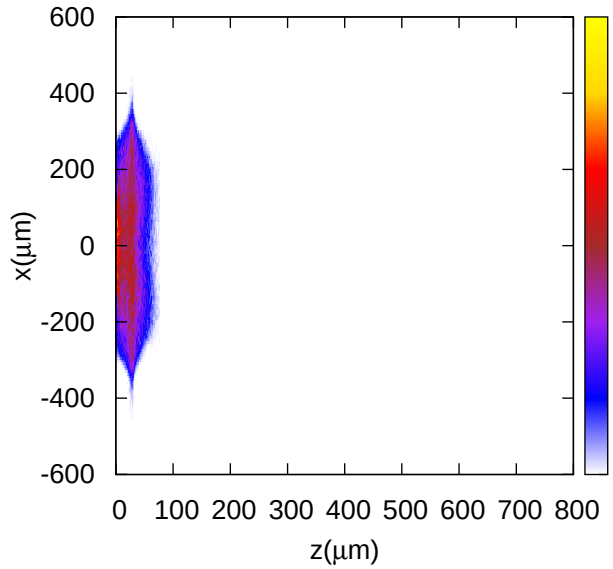
(a) t=10ps



(b) t=20ps



(c) t=30ps



(d) t=40ps

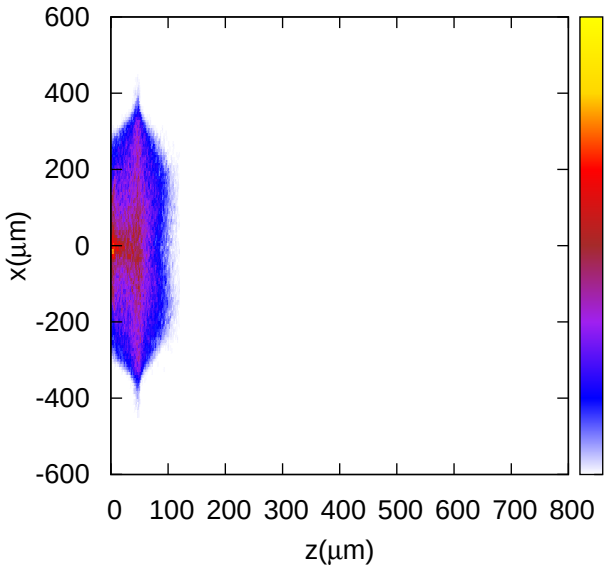
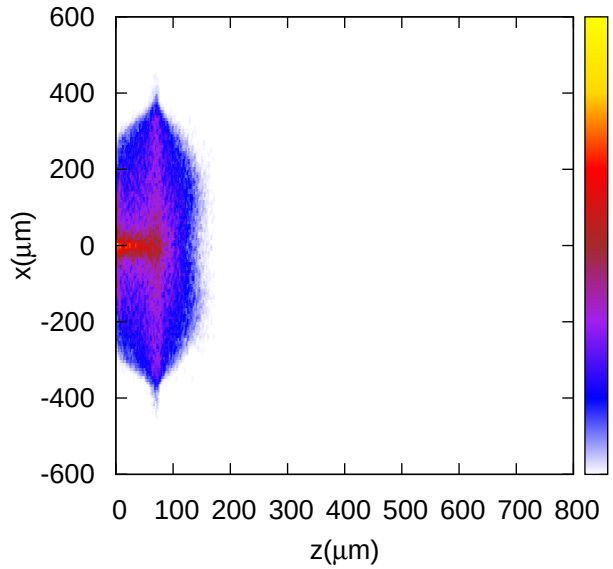
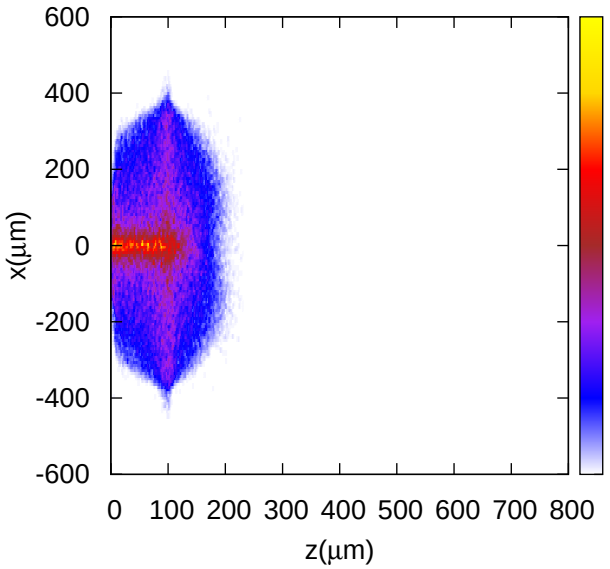


Figure 5.22: (cont'd)

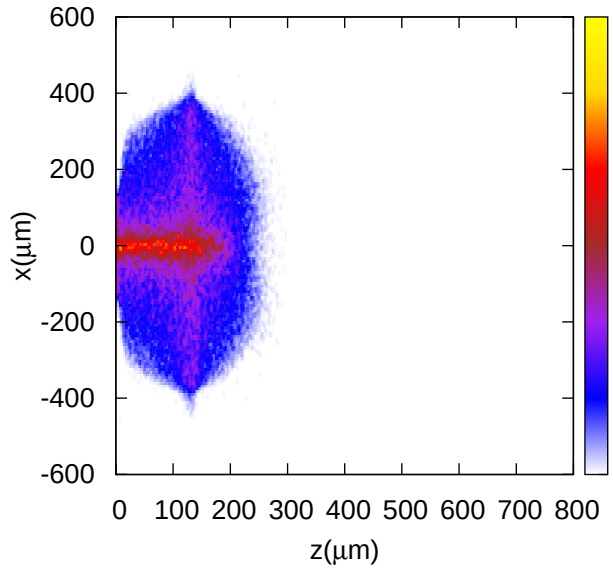
(e) t=50ps



(f) t=60ps



(g) t=70ps



(h) t=80ps

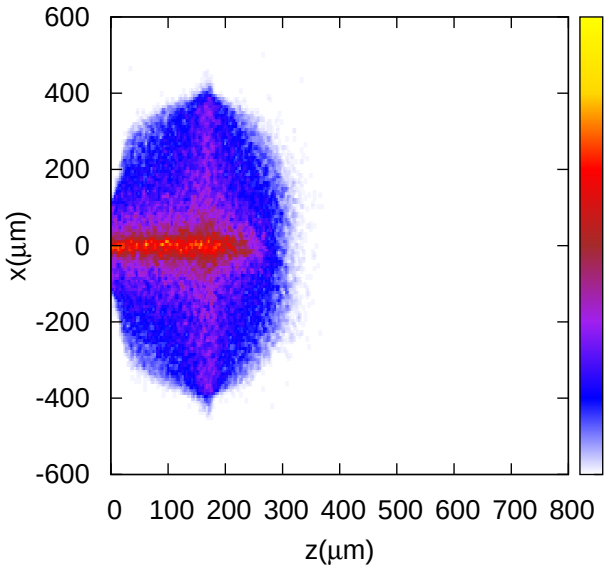
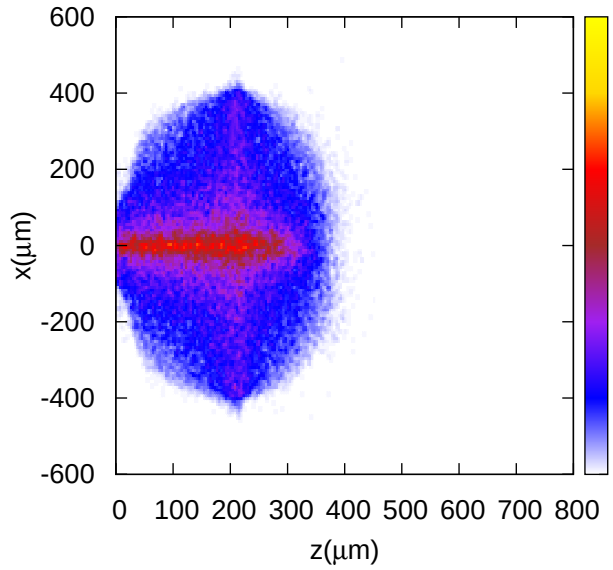
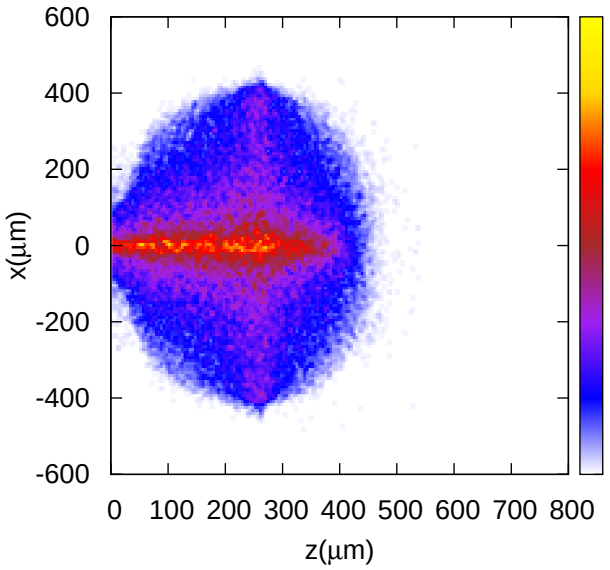


Figure 5.22: (cont'd)

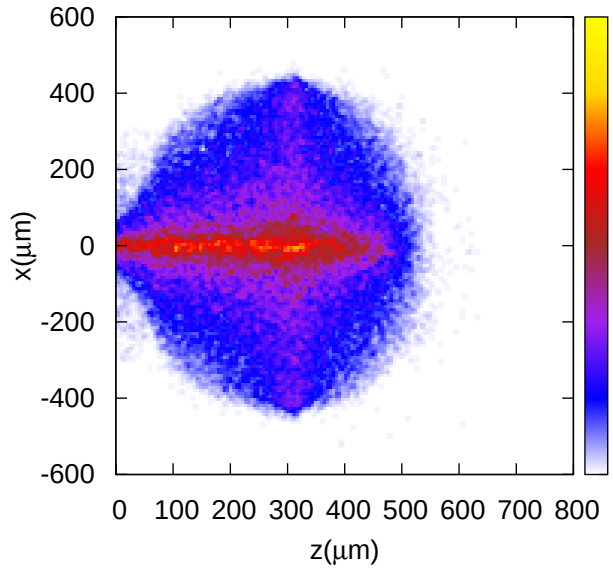
(i) t=90ps



(j) t=100ps



(k) t=110ps



(l) t=120ps

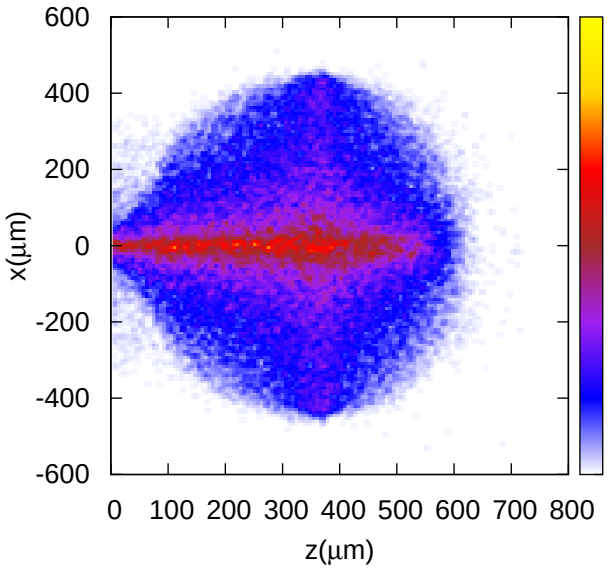
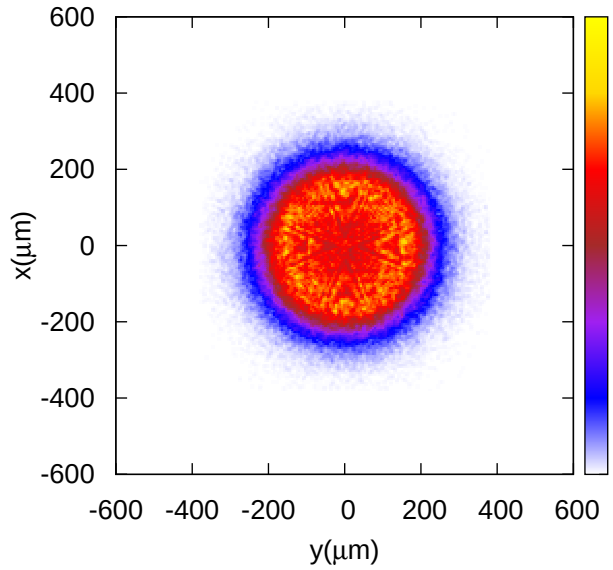
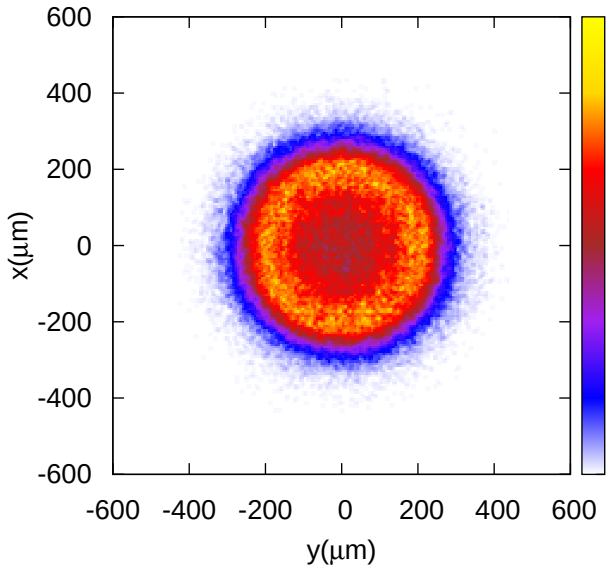


Figure 5.23: Charge distribution in $x-y$ phase space in the following 100 ps

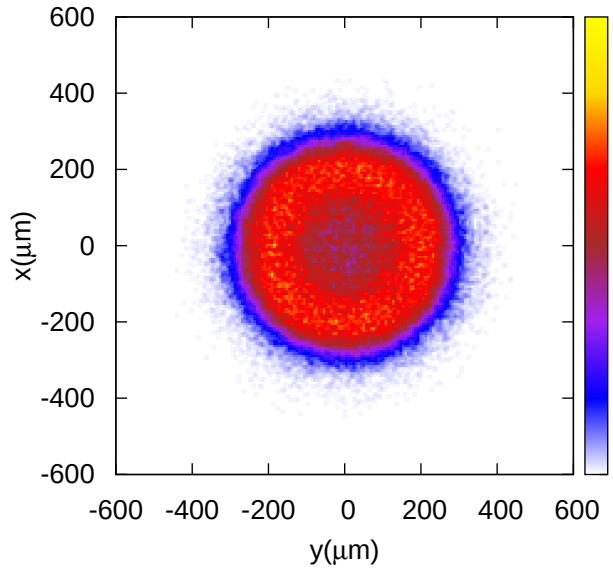
(a) t=1ps



(b) t=5ps



(c) t=10ps



(d) t=20ps

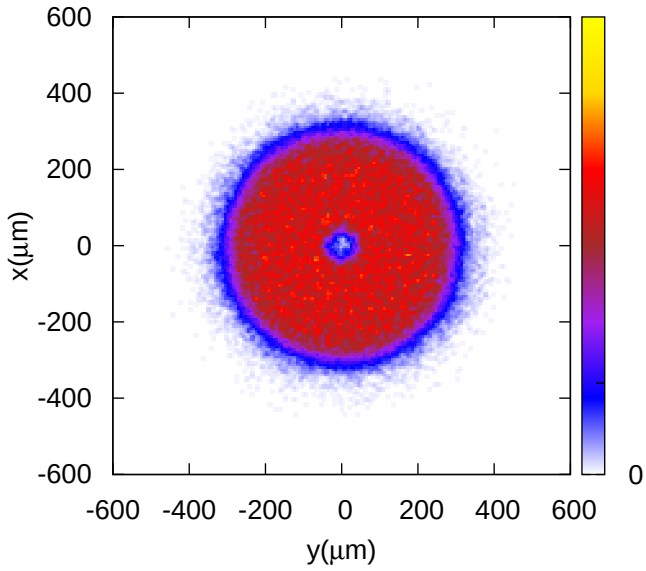
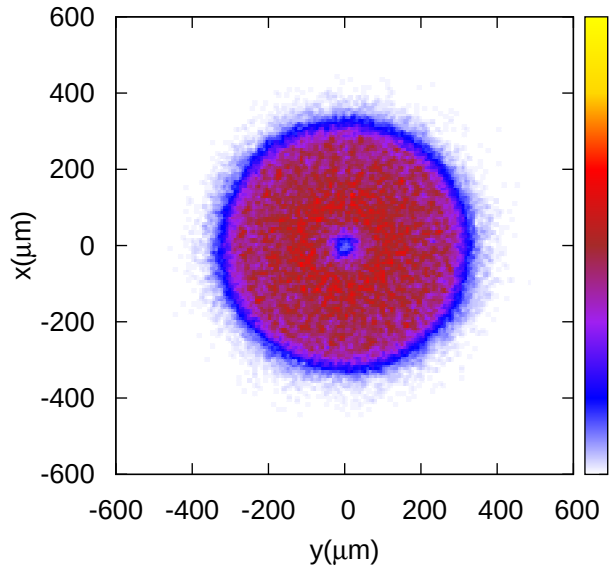
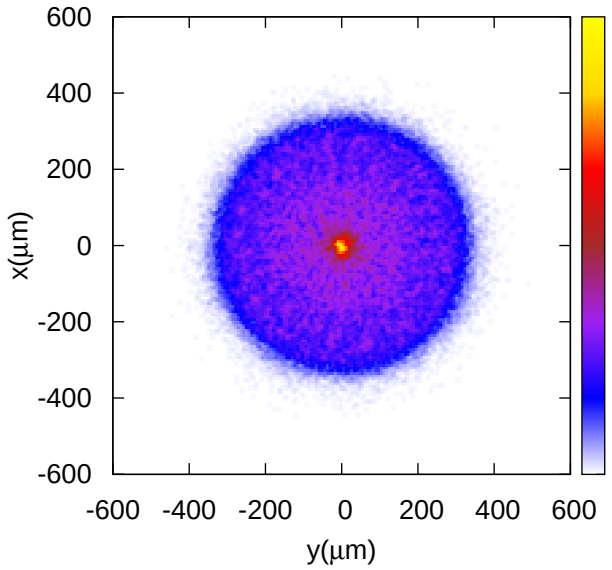


Figure 5.23: (cont'd)

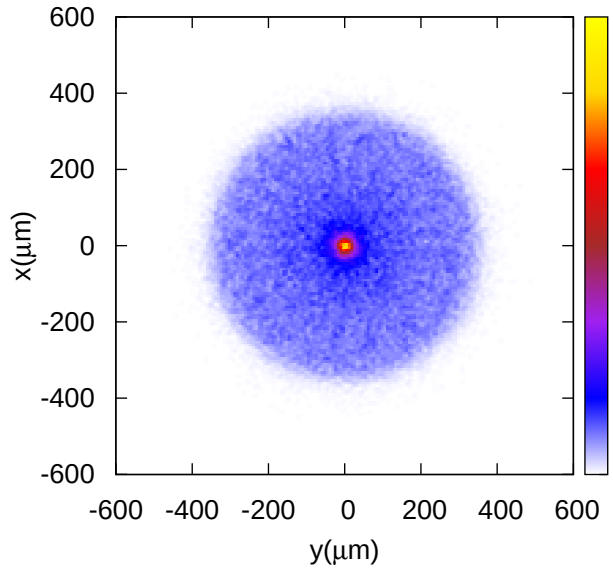
(e) t=30ps



(f) t=40ps



(g) t=50ps



(h) t=60ps

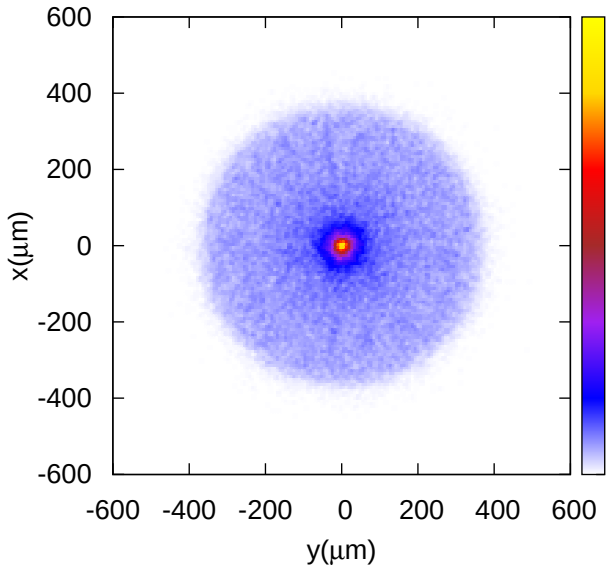
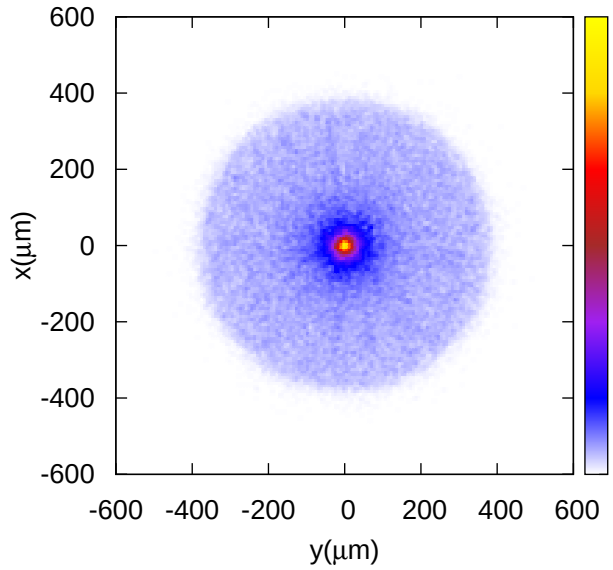
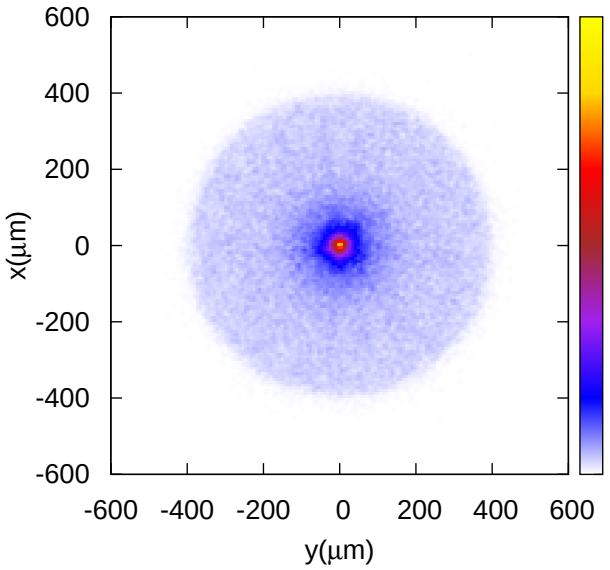


Figure 5.23: (cont'd)

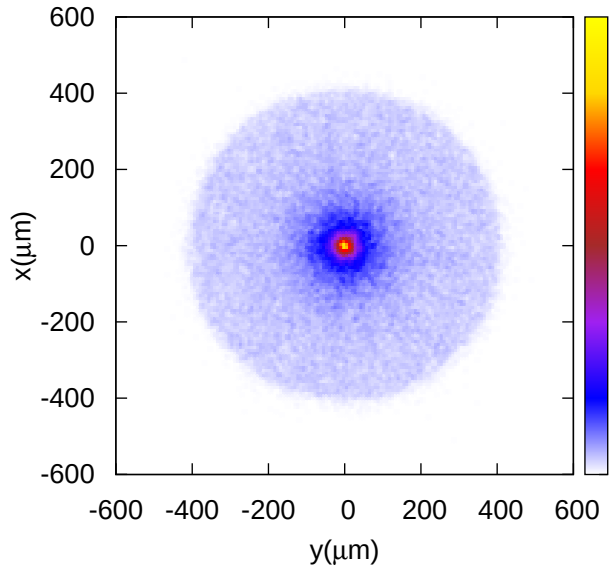
(i) t=70ps



(j) t=80ps



(k) t=90ps



(l) t=100ps

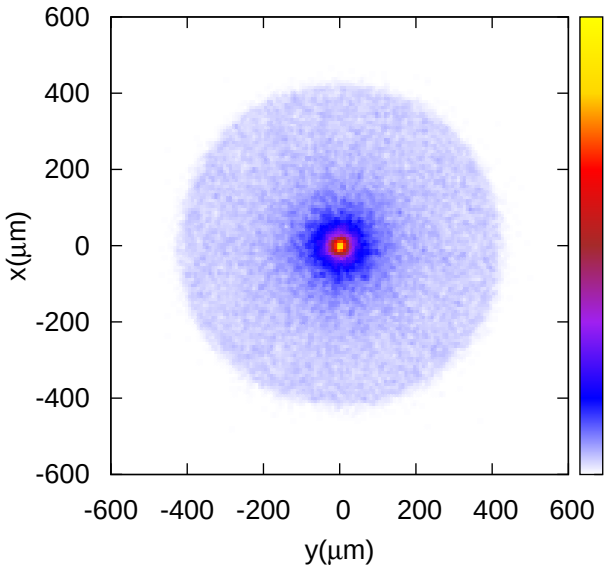
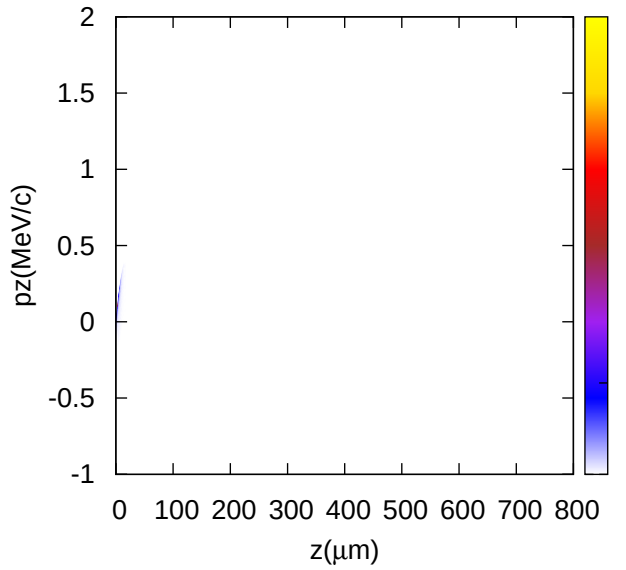
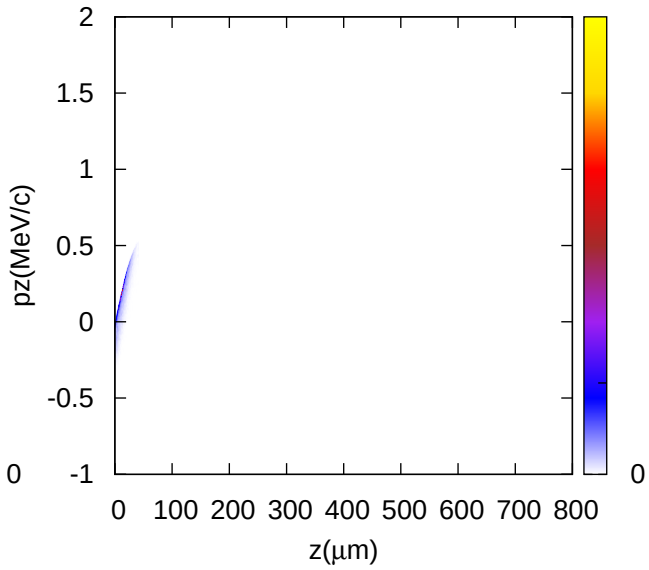


Figure 5.24: Charge distribution in $x - z$ phase space in the following 120 ps

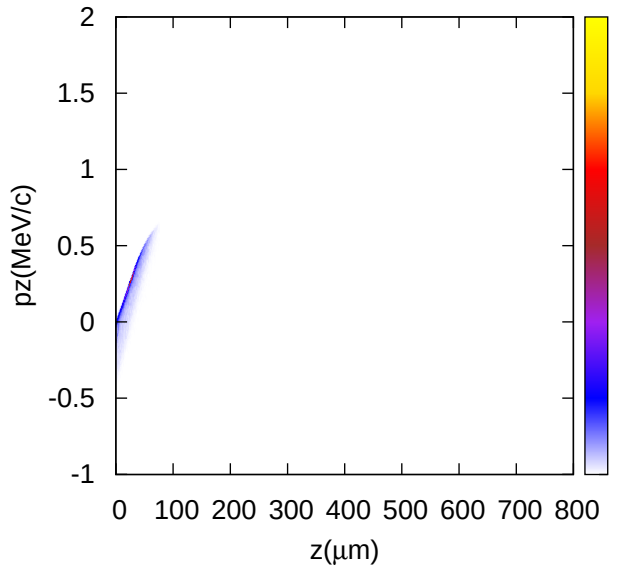
(a) t=10ps



(b) t=20ps



(c) t=30ps



(d) t=40ps

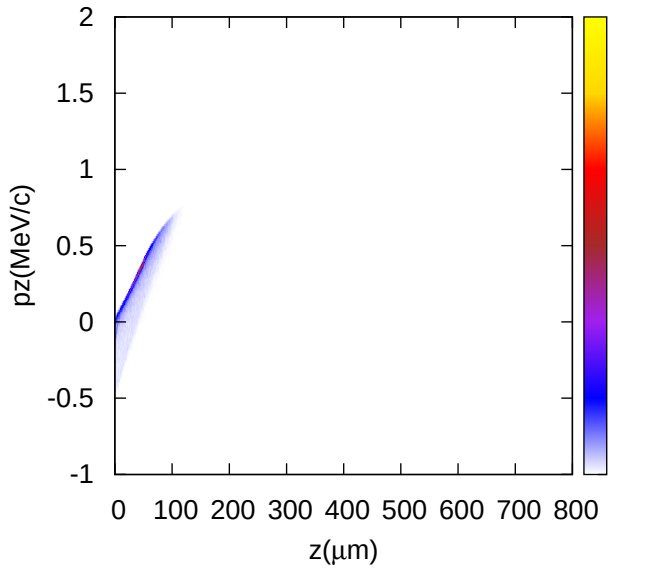
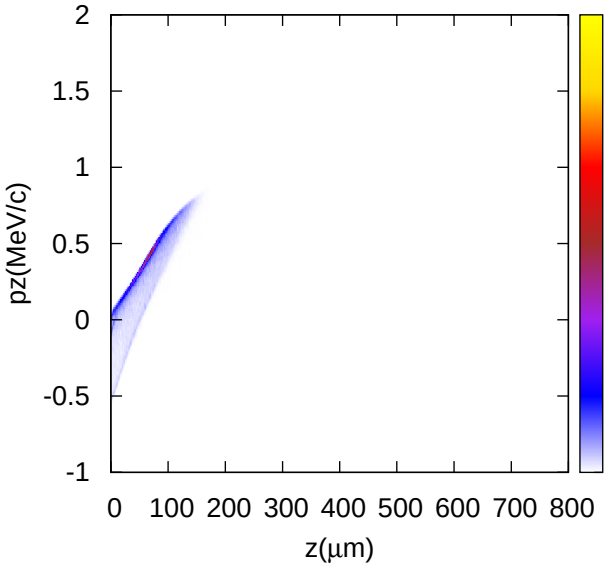
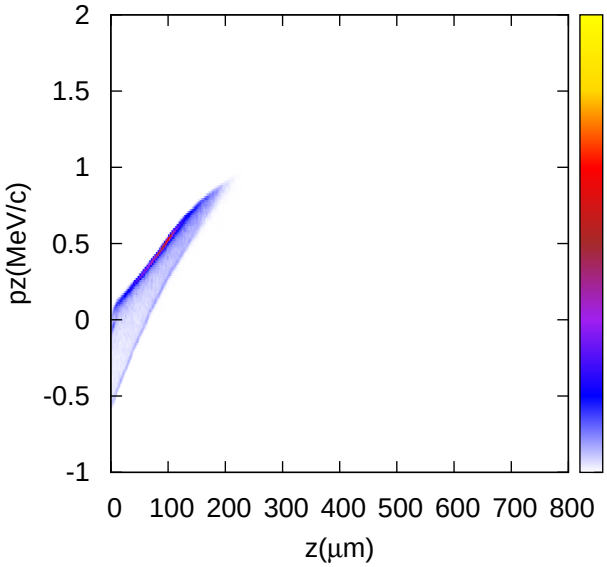


Figure 5.24: (cont'd)

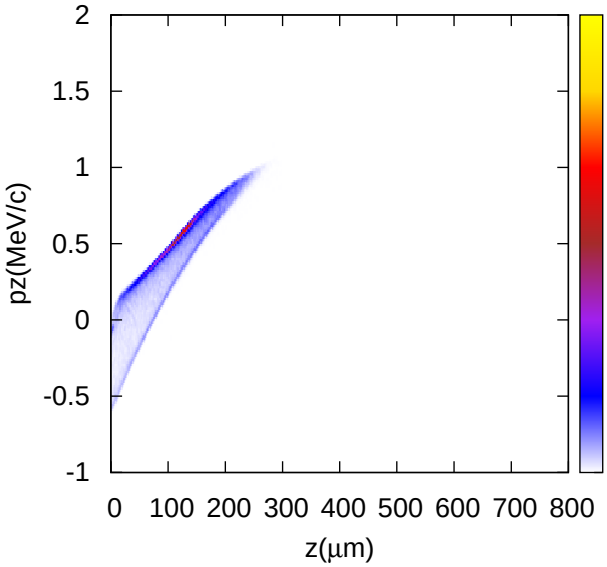
(e) t=50ps



(f) t=60ps



(g) t=70ps



(h) t=80ps

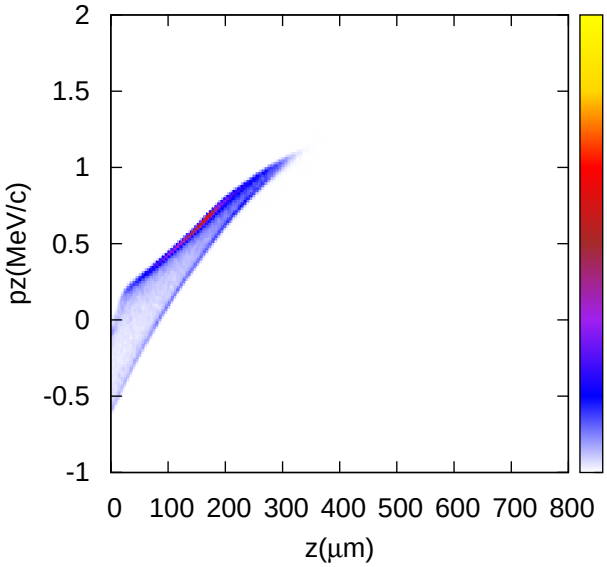
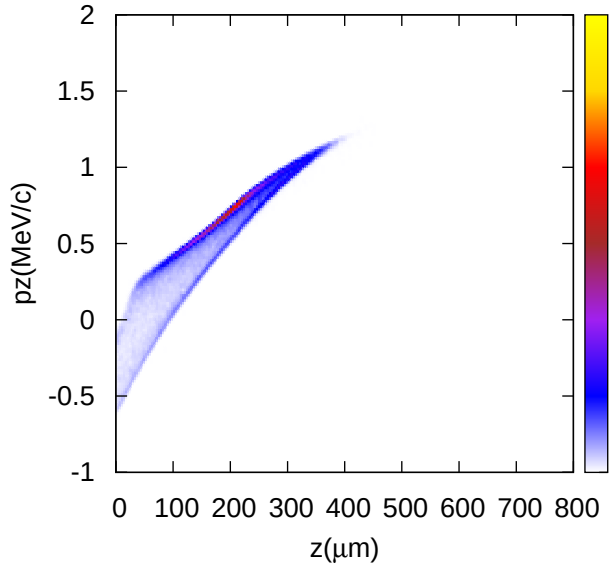
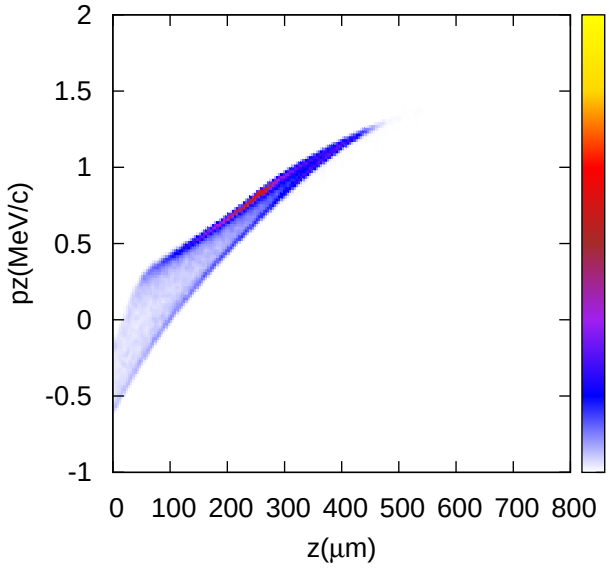


Figure 5.24: (cont'd)

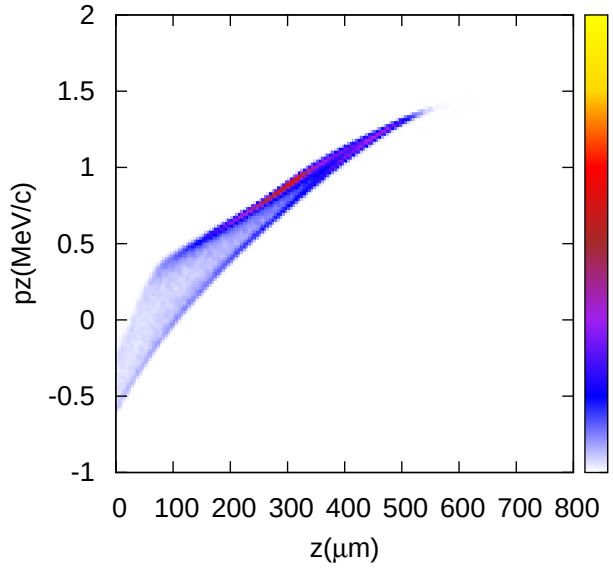
(i) t=90ps



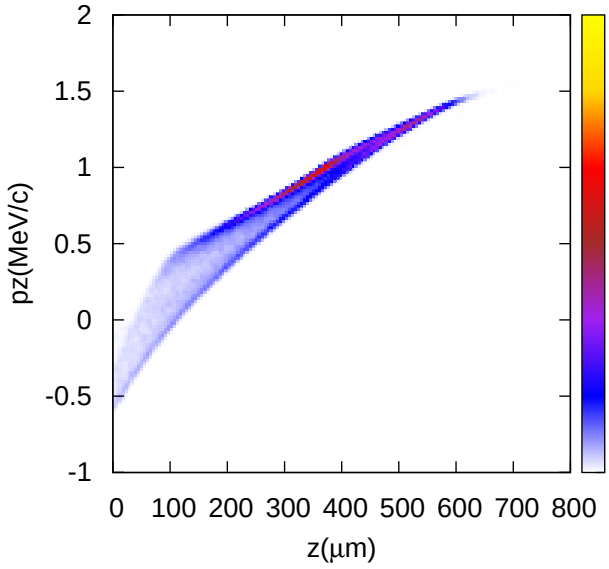
(j) t=100ps



(k) t=110ps



(l) t=120ps



5.5 Benefit of Using the Grid-free Fast Multipole Method

The particle-in-cell (PIC) method is currently the most popular algorithm for the space charge effect calculation in the beam community. There are two important differences that distinguish the grid-free fast multipole method from the PIC method. First, the PIC method depends on a grid, which makes it difficult to deal with systems with strongly nonuniform charge distribution or/and complex geometry[68, 22]. Advanced techniques are often required to treat curved geometries by the PIC method. The fast multipole method is grid-free and it treats any charge distribution and any geometry in a natural way. Some PIC programs can solve problems with regular boundaries, such as rectangular, circular, or elliptical boundary. But they have difficulties to solve problems with irregular boundaries. The fast multipole method can be combined with the boundary integral equation method (BIEM) [76, 38, 64] to solve problems with complicated geometries. Second, the PIC method artificially smooths the field. In a highly correlated system, it is difficult to accurately calculate the local interaction to simulate the fine structures by the PIC method without using a very fine grid, which greatly increases the computation cost[68, 90, 22]. In the research on advanced accelerators, sometimes a high density beam with a certain profile is desired[70]. For example, a flat beam may help to easily generate the accelerating wake field[61] and a three dimensional uniformly-filled ellipsoidal electron bunch is good for high-power free electron lasers[70]. The grid-free fast multipole method is an ideal simulation tool for these problems since it accurately calculates the nearby interactions, which affect the fine structure of the charge distribution. Besides the above-mentioned benefits, one can easily increase the accuracy using the grid-free fast multipole method, whenever the high accuracy is needed. As is described in the previous chapters, the accuracy can be increased by simply increasing

the order of the multipole expansions or increasing the distance between a box and its near neighbors. It does not depend on how we decompose the whole region into boxes. While using the PIC algorithm, the accuracy is limited by the grid size. To increase the accuracy, which requires finer grid size, is hard, especially for three dimensional problems.

Nowadays, accelerator physicists are trying to provide higher and higher intensity beams, so that it becomes more and more important to understand the interaction between charged particles. For example, muon cooling is an essential part in Fermilab's Neutrino Factory and Muon Collider program[36, 95], and some experiments have been launched in the international MICE project[28]. Electron cooling for the ion beam is also a challenge in Jefferson Lab's MEIC project[1]. The electron cloud effect is being studied in the main injector of Fermilab's Project X[53, 54] and Cornell University's CesrTA program[27, 67]. The space charge effect, which is dominant in the photoemission process, has to be understood and controlled in Cathode R&D for free electron laser (FEL) or electron microscope design[30]. Numerical simulation is a very useful tool for studies on collective effects, and it is inevitable when the collective effects are strong and highly nonlinear. Because of its grid-free property and good efficiency, the fast multipole method will be either a good cross-check to the classical PIC method or a challenger in high density highly correlated systems with complicated geometries.

APPENDIX

Detailed Description of the DA-Based MLFMA Algorithm

Choose DA order and s, the maximum number of particles inside a childless box.

Stage 1. Hierarchical structure of boxes.

Create the zero level box and put it into the array *box*. Update *lvl*, *prnt*, *cntr*, *idx*, *nptcl*, and *link* if applicable.

if *box*[1] contains more than s particles

$i_1 \leftarrow$the index of the last element in *box*. $a \leftarrow b$ means giving the value of b to a.

subdivide *box*[1] into eight boxes, add the nonempty boxes into *box*, update *nchld*, *lvl*, *prnt*, *cntr*, *idx*, *nptcl*, and *link* if applicable.

$i_2 \leftarrow$the index of the last element in *box*.

while $i_2 > i_1$

loop $i = i_1 + 1, i_2, 1$

if *box*[i] contains more than s particles

subdivide *box*[i] into eight boxes, add the nonempty boxes into *box*, update *nchld*, *lvl*, *prnt*, *cntr*, *idx*, *nptcl*, and *link* if applicable.

endif

endloop

$i_1 \leftarrow i_2$

$i_2 \leftarrow$the index of the last element in box.

endwhile

endif

Comment We denote by $nbox$ the total number of boxes formed in Stage 1. And we put the in line comments inside the "{}".

Stage 2. Multipole expansion.

loop $i = nbox, 2, -1$

if $box[i]$ is childless

Calculate the multipole expansion around the center of $box[i]$ according to the particles inside by Eq. (3.1).

else {$box[i]$ is a parent box}

Calculate the multipole expansion of $box[i]$ by shifting the multipole expansions of its child boxes to its center and take the summation using Eq. (3.2) and Eq. (3.3). {Note again that in Eq. (3.3) we do not calculate d'_r. }

endif

endloop

Comment Since we sort backwards from the last box, the multipole expansions of the child boxes are always calculated before that of their parent box.

Stage 3. Fields.

loop $i = 2, nbox, 1$

$b \leftarrow box[i]$.

$clg[i] \leftarrow 1\&i$. {Initialize the colleague list for b.}

if $lvl[i] > 1$

Inherit the local expansion from b's parent box using Eq. (3.9) and Eq. (3.10).

endif

if b is childless

Calculate the interaction between the particles inside b directly, and save the resulting field in $E(b)$.

endif

loop $j = 1, clg[prnt[i]]|1, 1$

$pclg \leftarrow clg[prnt[i]]|(j + 1)$.{Sort the colleague list of b's parent box.}

loop $k = box[pclg], box[pclg] + nchld[pclg] - 1, 1$

$w \leftarrow box[k]$. {w is a child box of a colleague of b's parent box.}

if $k > i$

if w is adjacent to b

Add w into b's colleague list.

For the operations here, see description (1) in the following .

else

w is in the list V_b. Convert the multipole expansion of w into the local expansion of b using Eq. (3.5) and Eq. (3.6), and add it to $lcl[i]$. In the

same way convert the multipole expansion of b into the local expansion of w and add it to $lcl[k]$.

endif

endif

endloop

endloop

if b is childless

Calculate the field on the particles inside b according to the local expansion of b, and add the result to $E(b)$.

Add $E(b)$ to the corresponding elements in E_x, E_y, and E_z.

endif

endloop

Comment We only calculate w that is behind b in the array box, because the interactions between b and those boxes before b have already been calculated. The diagram of stage 3 is shown in Figure A.1.

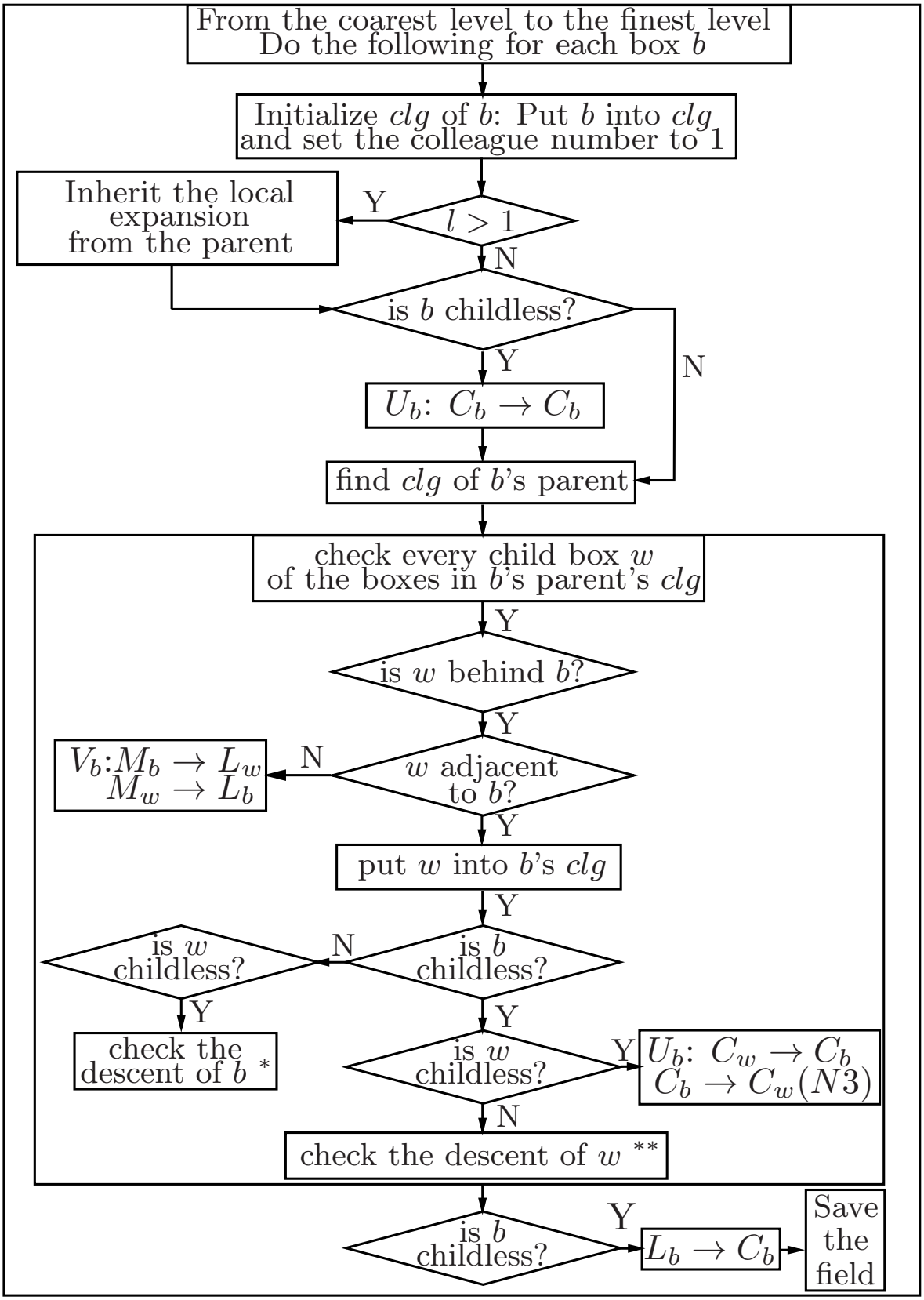
Description (1)

if b is childless

if w is childless

w is in the list U_b. Calculate the contribution of the particles in w directly and add the results to $E(b)$. Use Newton's third law to calculate the field on the

Figure A.1: Diagram of stage 3



particles inside w, and add the results to the corresponding elements in E_x, E_y, and E_z.

else {w is a parent box.}

Check the descent of w. See description (2).

endif

elseif w is childless

Check the descent of b. See description (3).

endif

Description (2)

Create a *stack*.

Put all child boxes of w into the *stack*.

$pnt \leftarrow$length of the *stack*.

while $pnt > 0$

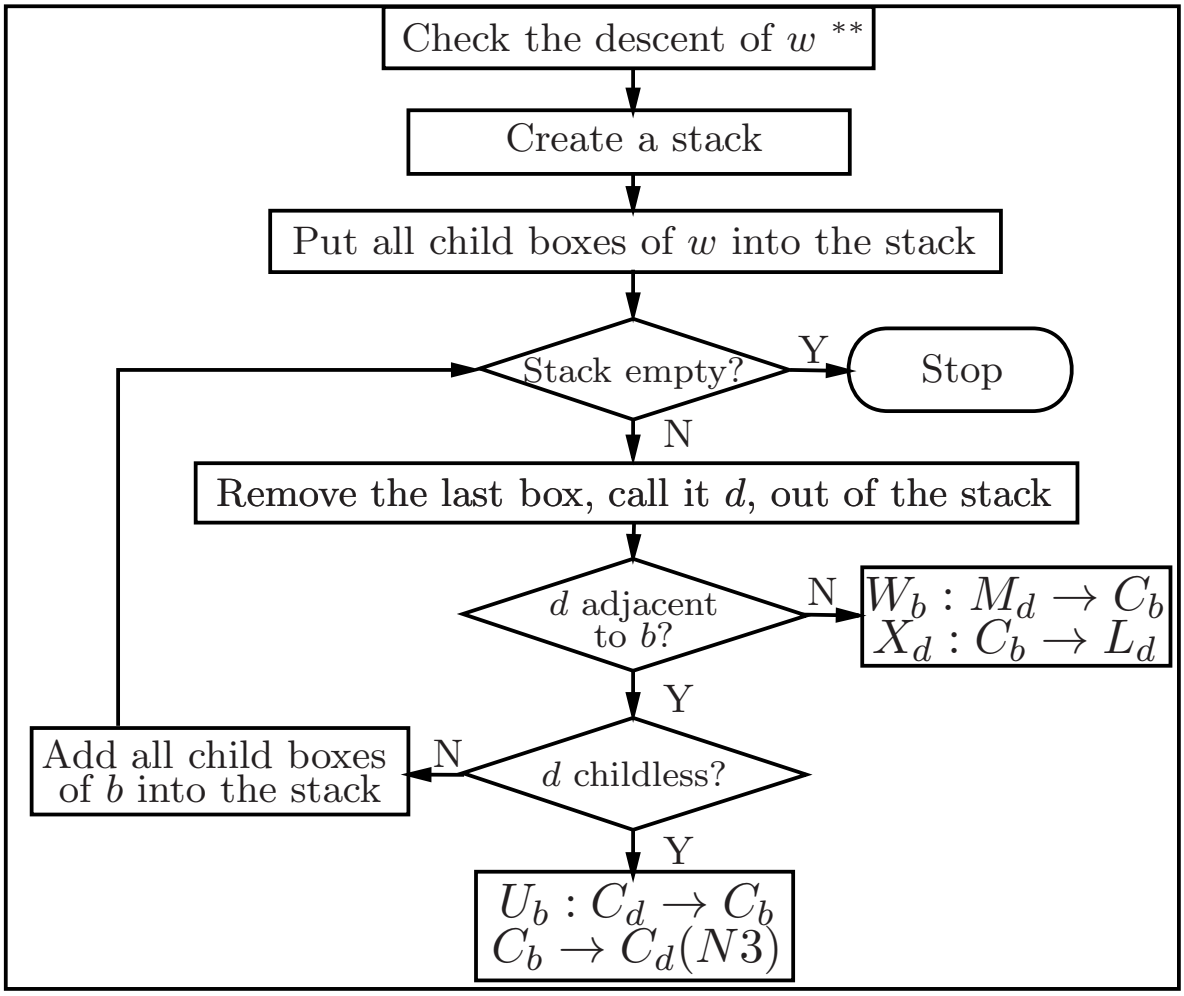
$d \leftarrow stack[pnt]$.

$pnt \leftarrow pnt - 1$. {Remove the last element from the *stack*.}

if d is not adjacent to b

d is in the list W_b. Calculate the fields on the particles inside b according to the multipole expansion of d using Eq. (3.11), and add the results to $E(b)$. Calculate the local expansion around the center of d according to the charges inside b using Eq. (3.7) and Eq. (3.8), and add the result to the corresponding element of *lcl*.

Figure A.2: Diagram of description (2)



else {d is adjacent to b.}

if d is childless

d is in the list U_b. Calculate the contribution of the particles in d directly and add the results to $E(b)$. Use Newton's third law to calculate the field on the particles inside d, and add the results to the corresponding elements in E_x, E_y, and E_z.

else {d is a parent box.}

Add all the child boxes of d into the *stack*.

pnt ←the length of the *stack*.

endif

endif

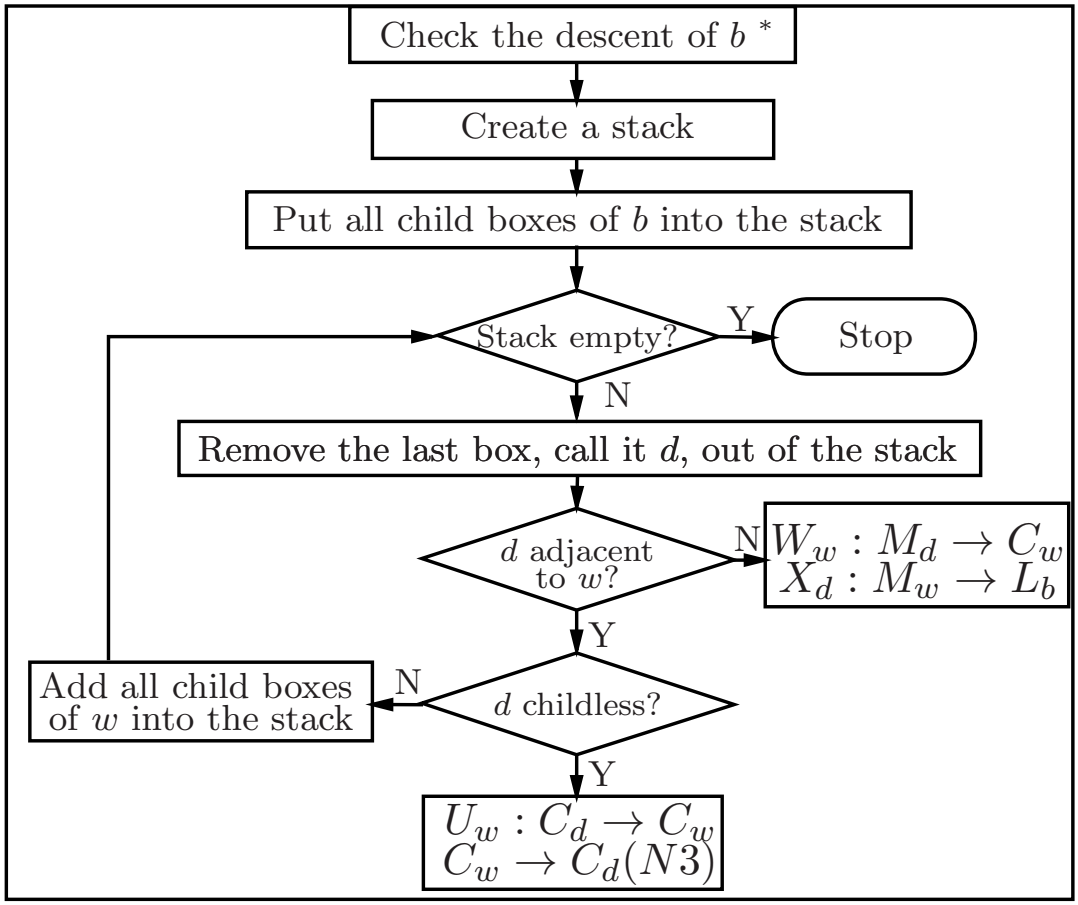
endwhile

Comment The diagram of description (2) is shown in Figure A.2.

Description (3)

The algorithm is the same with description (2). One only needs to replace all w by b and all b by w, and finally add one more step to add $E(w)$ to the corresponding elements in E_x, E_y, and E_z.

Figure A.3: Diagram of description (3)



BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S. Ahmed, A. Bogacz, Y. Derbenev, A. Hutton, G. Krafft, R. Li, V. Morozov, and F. Pilat. Conceptual design of a polarized medium energy electron-ion collider at jlab. *Polarization*, 70:80.

[2] S. Aluru, J. Gustafson, G.M. Prabhu, and F.E. Sevilgen. Distribution-independent hierarchical algorithms for the n-body problem. *The Journal of Supercomputing*, 12:303–323, 1998. 10.1023/A:1008047806690.

[3] S. Aluru, G.M. Prabhu, and J. Gustafson. Truly distribution-independent algorithms for the n-body problem. In *Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, pages 420–428. ACM, 1994.

[4] S. Aluru and F. Sevilgen. Dynamic compressed hyperoctrees with application to the n-body problem. In *Foundations of Software Technology and Theoretical Computer Science*, pages 21–33. Springer, 1999.

[5] C.R. Anderson. An implementation of the fast multipole method without multipoles. *SIAM Journal on Scientific and Statistical Computing*, 13:923, 1992.

[6] L.C. Andrews. *Special functions for engineers and applied mathematicians*. Macmillan, 1985.

[7] J. Applequist. Traceless cartesian tensor forms for spherical harmonic functions: new theorems and applications to electrostatics of dielectric media. *Journal of Physics A: Mathematical and General*, 22:4303, 1989.

[8] J. Barnes and P. Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *nature*, 324:4, 1986.

[9] J. E Barnes. A modified tree code: don't laugh; it runs. *Journal of Computational Physics*, 87(1):161–170, 1990.

[10] R. K. Beatson and L. Greengard. A short course on fast multipole methods. *Wavelets, Multilevel Methods and Elliptic PDEs*, pages 1–37, 1997.

[11] M. Berz. Arbitrary order description of arbitrary particle optical systems. *Nuclear Instruments and Methods*, A298:426, 1990.

[12] M. Berz. Modern map methods for charged particle optics. *Nuclear Instruments and Methods*, 363:100, 1995.

[13] M. Berz. From Taylor series to Taylor models. *AIP CP*, 405:1–20, 1997.

[14] M. Berz. *Modern Map Methods in Particle Beam Physics*. Academic Press, San Diego, 1999. Also available at http://bt.pa.msu.edu/pub.

[15] M. Berz and K. Makino. COSY INFINITY Version 9.1 programmer's manual. Technical report, Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, 2011. See also http://cosyinfinity.org.

[16] T. Bountis and C. Skokos. Space charges can significantly affect the dynamics of accelerator maps. *Physics Letters A*, 358(2):126–133, 2006.

[17] D.L. Bruhwiler and M.F. Reusch. High-order optics with space charge: The topkark code. In *AIP Conference Proceedings*, volume 297, page 524, 1993.

[18] D.L. Bruhwiler and M.F. Reusch. The effect of various particle distribution functions on the modeling of space charge effects in a high-order optics code. In *Proceeding of Fourth European Particle Accelerator Conference 1994*, page 1168, 1994.

[19] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM Journal on Scientific and Statistical Computing*, 9:669, 1988.

[20] H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of Computational Physics*, 155(2):468 – 498, 1999.

[21] C.H. Choi, K. Ruedenberg, and M.S. Gordon. New parallel optimal-parameter fast multipole method (opfmm). *Journal of Computational Chemistry*, 22(13):1484–1501, 2001.

[22] A.J. Christlieb, R. Krasny, J.P. Verboncoeur, J.W. Emhoff, and I.D. Boyd. Grid-free plasma simulation techniques. *Plasma Science, IEEE Transactions on*, 34(2):149–165, 2006.

[23] W.J. Cody. Chebyshev polynomial expansions of complete elliptic integrals. *Math. Comp*, 19:249–259, 1965.

[24] W.J. Cody. Rational chebyshev approximations for the error function. *Mathematics of Computation*, 23(107):631–637, 1969.

[25] R. Coifman, V. Rokhlin, and S. Wandzura. The fast multipole method for the wave equation: A pedestrian prescription. *Antennas and Propagation Magazine, IEEE*, 35(3):7–12, 1993.

[26] C. Coleman-Smith, H.A. Padmore, and W. Wan. Limits to electron beam emittance from stochastic coulomb interactions. Technical report, Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA (US), 2008.

[27] J.A. Crittenden, J.R. Calvey, G.F. Dugan, D.L. Kreinick, Z. Leong, J.A. Livezey, M.A. Palmer, D.L. Rubin, D.C. Sagan, R.L. Holtzapple, et al. Progress in studies of electron-cloud-induced optics distortions at CesrTA. Technical report, Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA (US), 2010.

[28] A. Dobbs, D. Adams, K. Long, J. Pasternak, and M. Apollonio. The MICE muon beam: Status and progress. In *Proceedings of 1st International Particle Accelerator Conference*, 2010.

[29] J. Dong, S.L. Chai, and J.J. Mao. An automatic load-balanced parallel multilevel fast multipole method for large scale electromagnetic scattering problem. In *Microwave Conference Proceedings, 2005. APMC 2005. Asia-Pacific Conference Proceedings*, volume 3, pages 4, IEEE, 2005.

[30] D.H. Dowell, I. Bazarov, B. Dunham, K. Harkay, C. Hernandez-Garcia, R. Legg, H. Padmore, T. Rao, J. Smedley, and W. Wan. Cathode R&D for future light sources. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 622(3):685–697, 2010.

[31] D.H. Dowell and J.F. Schmerge. Quantum efficiency and thermal emittance of metal photocathodes. *Physical Review Special Topics-Accelerators and Beams*, 12(7):074201, 2009.

[32] N. Engheta, W.D. Murphy, V. Rokhlin, and M.S. Vassiliou. The fast multipole method (fmm) for electromagnetic scattering problems. *Antennas and Propagation, IEEE Transactions on*, 40(6):634–641, 1992.

[33] O. Ergul and L. Gurel. Efficient parallelization of the multilevel fast multipole algorithm for the solution of large-scale scattering problems. *Antennas and Propagation, IEEE Transactions on*, 56(8):2335–2345, 2008.

[34] W. Fong and E. Darve. The black-box fast multipole method. *Journal of Computational Physics*, 228(23):8712–8725, 2009.

[35] R.W. Garnett and T.P. Wangler. Space-charge calculation for bunched beams with 3-D ellipsoidal symmetry. In *Proceedings of the 1991 IEEE Particle Accelerator Conference (APS Beam Physics)*, page 330, 1991.

[36] S. Geer. Muon cooling R&D. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 503(1):64–69, 2003.

[37] L. Greengard and W.D. Gropp. A parallel version of the fast multipole method. *Computers & Mathematics with Applications*, 20(7):63–71, 1990.

[38] L. Greengard and J.Y. Lee. A direct adaptive poisson solver of arbitrary order accuracy. *Journal of Computational Physics*, 125(2):415–424, 1996.

[39] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[40] L.F. Greengard and J. Huang. A new version of the fast multipole method for screened coulomb interactions in three dimensions. *Journal of Computational Physics*, 180(2):642–658, 2002.

[41] D.P. Grote, A. Friedman, and I. Haber. Methods used in warp3d, a three-dimensional pic/accelerator code. In *AIP Conference Proceedings*, volume 391, page 51, 1997.

[42] D.P. Grote, A. Friedman, I. Haber, W. Fawley, and J.L. Vay. New developments in warp: Progress toward end-to-end simulation. *Nuclear Instruments and Methods in Physics Research-Section A Only*, 415(1):428–432, 1998.

[43] D.P. Grote, A. Friedman, J.L. Vay, and I. Haber. The warp code: Modeling high intensity ion beams. In *16th international Workshop on ECR Ion Sources, edited by M. Leitner, AIP, Berkeley, CA*, 2004.

[44] B. Hariharan and S. Aluru. Efficient parallel algorithms and software for compressed octrees with applications to hierarchical methods. *Parallel computing*, 31(3-4):311–331, 2005.

[45] G.H. Jansen. *Coulomb Interactions in Particle Beams (Advances in Electronics and Electron Physics Supplement 21)*. Academic Press, 1990.

[46] F.W. Jones. A hybrid fast-muitipole technique for space-charge tracking with halos. In *AIP Conference Proceedings*, pages 359–370. IOP INSTITUTE OF PHYSICS PUBLISHING LTD, 1998.

[47] F.W. Jones and H.O. Schonauer. New space-charge methods in accsim and their application to injection in the cern ps booster. In *Particle Accelerator Conference, 1999. Proceedings of the 1999*, volume 4, pages 2933–2935. IEEE, 1999.

[48] P. Kruit and G.H. Jansen. Space charge and statistical Coulomb effects, second edition. *Handbook of Charged Particle Optics*, pages 341–391, 2009.

[49] K. Kuns. Calculation of magnetic field inside plasma chamber. *UCLA report*, 2(3):1–11, 2007.

[50] P. Lapostolle, J.M. Lagniel, S. Nath, N. Pichoff, E. Tanke, and S. Valero. The scherm space charge routine, limitations and solutions. In *Linac 1998 conference*.

[51] P. Lapostolle, A.M. Lombardi, E. Tanke, S. Valero, R.W. Garnett, and T.P. Wangler. A modified space charge routine for high intensity bunched beams. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 379(1):21–40, 1996.

[52] P.M. Lapostolle, A.M. Lombardi, S. Nath, E. Tanke, S. Valero, and T.P. Wangler. A new approach to space charge for linac beam dynamics codes. In *Proceedings of the 18th International Linear Accelerator Conference*, page 375, 1996.

[53] P. Lebrun, P. Spentzouris, J.R. Cary, P. Stoltz, and S.A. Veitzer. Accurate simulation of the electron cloud in the fermilab main injector with VORPAL. In *Proceedings of 1st International Particle Accelerator Conference: IPAC10*, 2010.

[54] P.L.G. Lebrun, P. Spentzouris, J.R. Cary, P. Stoltz, and S.A. Veitzer. Accurate simulation of the electron cloud in the fermilab main injector with VORPAL. Technical report, Fermi National Accelerator Laboratory (FNAL), Batavia, IL, 2011.

[55] P. Li, H. Johnston, and R. Krasny. A cartesian treecode for screened coulomb interactions. *Journal of Computational Physics*, 228(10):3858–3868, 2009.

[56] J. Makino. Yet Another Fast Multipole Method without Multipoles–Pseudoparticle Multipole Method. *Journal of Computational Physics*, 151(2):910–920, 1999.

[57] S.L. Manikonda and M. Berz. An accurate high-order method to solve the Helmholtz boundary value problem for the 3D Laplace equation. *International Journal of Pure and Applied Mathematics*, 23,3:365–378, 2005.

[58] S.L. Manikonda and M. Berz. Multipole expansion solution of the Laplace equation using surface data. *Nuclear Instruments and Methods*, 558,1:175–183, 2006.

[59] S.L. Manikonda, M. Berz, and K. Makino. High-order verified solution of the 3D Laplace equation. *Transactions on Computers*, 4-11:1604–1610, 2005.

[60] M. Martini and M. Prome. Computer studies of beam dynamics in a proton linear accelerator with space charge. *Particle Accelerators*, 2:289–299, 1971.

[61] D. Mihalcea, P. Piot, P. Stoltz, and B. Cowan. Wakefield generation in compact rectangular dielectric-loaded structures using flat beams. In *Proceedings of 2011 Particle Accelerator Conference*, 2011.

[62] S. Nath, J. Qiang, R. Ryne, J. Stovall, H. Takeda, L. Young, K.R. Crandall, N. Pichoff, and D. Uriot. Comparison of linac simulation codes. In *Particle Accelerator Conference, 2001. PAC 2001. Proceedings of the 2001*, volume 1, pages 264–266. IEEE, 2001.

[63] C. Nieter and J.R. Cary. VORPAL: a versatile plasma simulation code. *Journal of Computational Physics*, 196(2):448–473, 2004.

[64] N. Nishimura. Fast multipole accelerated boundary integral equation methods. *Applied Mechanics Reviews*, 55:299, 2002.

[65] E. Nissen and B. Erdelyi. A new paradigm for modeling, simulation, and analysis of intense beams. In *Proccedings of the 46th ICFA Advanced Beam Dynamics Workshop on High-Intensity and High Brightness Hadron Beams*, 2010.

[66] A. Orzhekhovskaya and G. Franchetti. A space charge algorithm for the bunches of elliptical crossection with arbitrary beam size and particle distribution. *Proceedings of International Computational Accelerator Physics, ICAP*, pages 106–109, 2006.

[67] M.A. Palmer, J.P. Alexander, M.G. Billing, J.R. Calvey, C. Conolly, J.A. Crittenden, J.A. Dobbins, G.F. Dugan, N. Eggert, E. Fontes, et al. Electron cloud at low emittance in CesrTA. In *Proceedings of 1st International Particle Accelerator Conference: IPAC10*, 2010.

[68] S. Pfalzner and P. Gibbon. *Many-body tree methods in physics*. Cambridge University Press, 2005.

[69] N. Pichoff, J.M. Lagniel, and S. Nath. Simulation results with an alternate 3D space charge routine, PICNIC. In *Proceedings of the XIX International Linac Conference*, volume 141, pages 23–28, 1998.

[70] P. Piot, C. Behrens, C. Gerth, F. Lemery, D. Mihalcea, and M. Vogt. Generation and characterization of electron bunches with ramped current profile at the flash facility.

[71] G. Pöplau, U. Van Rienen, J. Staats, and T. Weiland. Fast algorithms for the tracking of electron beams. *energy*, 1(5):0–001, 2000.

[72] G. Poplau, U. van Rienen, B. van der Geer, and M. de Loos. Multigrid algorithms for the fast calculation of space-charge effects in accelerator design. *IEEE Transactions on Magnetics*, 40(2 Part 2):714–717, 2004.

[73] D. Potts, G. Pöplau, and U. van Rienen. Calculation of 3d space-charge fields of bunches of charged particles by fast summation. *Scientific Computing in Electrical Engineering*, page 30, 2004.

[74] J. Qiang, R.D. Ryne, S. Habib, and V. Decyk. An object-oriented parallel particle-in-cell code for beam dynamics simulation in linear accelerators. *Journal of Computational Physics*, 163(2):434–451, 2000.

[75] R.K. Raman, Z. Tao, T.R. Han, and C.Y. Ruan. Ultrafast imaging of photoelectron packets generated from graphite surface. *Applied physics letters*, 95(18):181108–181108, 2009.

[76] V. Rokhlin. Rapid solution of integral equations of classical potential theory. *Journal of Computational Physics*, 60(2):187–207, 1985.

[77] R.D. Ryne. Advanced computing tools and models for accelerator physics. Technical report, Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA (US), 2008.

[78] F. Sevilgen and S. Aluru. A unifying data structure for hierarchical methods. In *Supercomputing, ACM/IEEE 1999 Conference*, pages 24–24. IEEE, 1999.

[79] F.E. Sevilgen, S. Aluru, and N. Futamura. A provably optimal, distribution-independent parallel fast multipole method. In *Proceedings of the 14th International Parallel and Distributed Processing Symposium*, pages 77–84. IEEE, 2000.

[80] B. Shanker and H. Huang. Accelerated cartesian expansions-A fast method for computing of potentials of the form $r^{-\nu}$ for all real ν. *Journal of Computational Physics*, 226(1):732–753, 2007.

[81] J.P. Singh, C. Holt, J.L. Hennessy, and A. Gupta. A parallel adaptive fast multipole method. In *Proceedings of the 1993 ACM/IEEE conference on Supercomputing*, pages 54–65. ACM, 1993.

[82] J.P. Singh, C. Holt, T. Totsuka, A. Gupta, and J.L. Hennessy. Load balancing and data locality in adaptive hierarchical N-body methods: Barnes-Hut, fast multipole, and radiosity. *Journal of Parallel and Distributed Computing*, 27(2):118–141, 1995.

[83] T. Srinivasan-Rao, J. Fischer, and T. Tsang. Photoemission studies on metals using picosecond ultraviolet laser pulses. *Journal of applied physics*, 69(5):3291–3296, 1991.

[84] Z. Tao, H. Zhang, P. Duxbury, M. Berz, and C. Ruan. Quantitative imaging of ultrashort photoelectron pulse dynamics. *Bulletin of the American Physical Society*, 56, 2011.

[85] Z. Tao, H. Zhang, P.M. Duxbury, M. Berz, and C.Y. Ruan. Space charge effects in ultrafast electron diffraction and imaging. *Journal of Applied Physics*, 111:044316, 2012.

[86] S.H. Teng. Provably good partitioning and load balancing algorithms for parallel adaptive n-body simulation. *SIAM Journal on Scientific Computing*, 19(2):635–656, 1998.

[87] S.B. van der Geer, M.J. de Loos, G. Pöplau, and U. van Rienen. Adaptive space-charge meshing in the general particle tracer code. In *Proceedings of PAC*, 2011.

[88] S. Velamparambil and W.C. Chew. Analysis and performance of a distributed memory multilevel fast multipole algorithm. *Antennas and Propagation, IEEE Transactions on*, 53(8):2719–2727, 2005.

[89] S. Velamparambil, W.C. Chew, and J. Song. 10 million unknowns: is it that big? *Antennas and Propagation Magazine, IEEE*, 45(2):43–58, 2003.

[90] J.P. Verboncoeur. Particle simulation of plasmas: review and advances. *Plasma physics and controlled fusion*, 47:A231, 2005.

[91] M.S. Warren and J.K. Salmon. A parallel hashed oct-tree n-body algorithm. In *Proceedings of the 1993 ACM/IEEE conference on Supercomputing*, pages 12–21. ACM, 1993.

[92] L. Ying, G. Biros, D. Zorin, and H. Langston. A new parallel kernel-independent fast multipole method. In *Supercomputing, 2003 ACM/IEEE Conference*, pages 14–14. IEEE, 2003.

[93] H. Zhang and M. Berz. The fast multipole method in the differential algebra framework. *Nuclear Instruments and Methods A 645*, pages 338–344,, 2011.

[94] F. Zhao. An O (N) Algorithm for Three-Dimensional N-Body Simulations. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 1987.

[95] M.S. Zisman. The muon collider. Technical report, Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA (US), 2011.