TESLA Report 2005-21

Master Thesis

Numerical Computation of Space-Charge Fields of Electron Bunches in a Beam Pipe of Elliptical Shape

Universität Rostock
Fakultät für Informatik und Elektrotechnik

submitted by

ALEKSANDAR MARKOVIĆ

September 28, 2005

Supervisor:

PROF. DR. U. VAN RIENEN
DR. G. PÖPLAU

Declaration

I have written this thesis independently, solely based on the literature and tools mentioned in the chapters and the appendix. This document in the present or a similar form has not and will not be submitted to any other institution apart from the University of Rostock to receive an academical grade.

Rostock, 28th of September, 2005

(Aleksandar Markoviḱ)

Contents

1 Introduction

[3] The significance of high energy accelerators of subatomic particles is ever increasing, in medical or industrial applications as well as in basic scientific research which often results in products or tools that become valuable to the world outside the high-energy physics.

There are two general types of accelerator designs, linear (linac) - the particles are accelerated in a straight line, and circular (synchrotron). Today both types use electromagnetic resonant cavities (Figure 1) made by conducting or superconducting materials to accelerate charged particles by means of radiofrequency electric fields, the wavelength of those microwave fields is ranging from 0.1 to 1m [15]. The resonant oscillations are time harmonic. One resonant mode is suitable for acceleration if it has a strong axial (longitudinal) electric field. The passage of the *bunches* (groups of close to each other particles are usually referred to as a bunch) is synchronized with the phase of the accelerating field, it means the size of the cavities in the accelerator is matched to the wavelength of the microwaves so that the electric and magnetic field patterns repeat every three cavities along the accelerator. In such a way the particles are accelerated almost to the speed of light. According to the relativity theory they do not exceed the speed of light, since $E = mc^2$ they get heavier gaining additional energy [14].

However behind the relative simple functioning principle of particle acceleration there are plenty of problems which are being addressed from the scientists in the accelerator beam physics. One kind of problem is the **interaction** of the bunches of particles of same and different kind on each other. Particles of equal charge repel each other due to space-charge forces and it is difficult to pack a high charge in a small volume. Another well known problem is the electron cloud phenomenon which causes a single-bunch instability arising from the interaction on successive turns of a single bunch with the cloud generated by the previous bunches. This instabilities may eventually lead to a beam breakup. Another problem are the beam-induced cavity fields which are known as wake fields.

In each case there is a collective (Lorentz) force experienced by a particle in the collective fields [3]. In order to study the effects of these forces on the trajectories of the beam particles numerical simulations are used. The importance of the numerical simulations is enormous in investigating the behaviour of the beam under different circumstances at present beam lines as well as in the development phase of new beam lines. Calculating the space-charge forces in order to simulate their influence on the beam dynamics, demands calculation of the space-charge fields in the beam line domain. Many simulation codes calculate 3D space-charge fields of bunches in a *cuboidal* domain [1] where the solution methods are based on Fast Fourier Transformations (FFT) [6] or Multigrid method [13].

This work deals in particularly with 3D numerical simulations of space-charge fields from electron bunches in a *beam pipe with elliptical cross-section*. To obtain the space-charge fields it is necessary to calculate the *Poisson equation* (2) with given boundary condition and space charge distribution. The discretization of the Poisson equation by the method of finite differences on a Cartesian grid, as well as setting up the coefficient matrix $\boldsymbol{A}$ for the elliptical domain are explained in the section 2. In the section 3 the properties of the coefficient matrix and possible numerical algorithms suitable for solving *non-symmetrical* linear systems of

equations are introduced. In the following section 4, the applied solver algorithms will be investigated by numerical tests with right hand side function for which the analytical solution is known.

The program codes for this purpose are written in the programming language ANSI C.

The algorithms described in the sections 2 and 3 have been implemented in the *software package "MOEVE"* [10]. This package solves 3D Poisson's equation by means of multigrid. Because of the fast computation of space-charge fields "MOEVE" is integrated as a function in the particle tracking code "GPT" [13]. The implementation of the routines for the case of structures with elliptic cross-section in the package "MOEVE" made it possible to investigate the effects of the elliptical boundaries on the potential generated by the charged particles. Hence in section 5 the results of the 3D space-charge field calculation in a beam pipe of elliptical shape are compared with those of the 3D space-charge field calculation based on a multigrid Poisson solver on a rectangular domain. Further it will be possible to integrate the new routines as a functions in the existing tracking code such as "ASTRA" and "GPT".

2 Problem and Discretization

The primary interest is to simulate the overall behavior of bunches of charged particles influenced from different fields along their way through the accelerator.

The aim of this work is a numerical simulation of space-charge fields of electron bunches taking into account the influence of the elliptic cross-section of the beampipe. It is clear that for the numerical simulation the particles in the bunch have to be abstracted with so called *macro-particles*. The macro-particles are representing many elementary particles which are comprising the bunch and the macro-particles are the ones which are considered in the numerical simulations.

Figure 1: Superconducting 1.3 GHz 9-cell cavity for the TESLA Test Facility [2].

In order to describe the dynamics of the particles like as in Newton's fundamental law of dynamics $F = ma$ the force and the mass are needed in order to calculate the acceleration. Certainly here it has to be dealt with relativistic mass and acceleration. It is being used the Lorenz factor

$$\gamma_i := (1 - \boldsymbol{v}_i^2/c^2)^{-1/2},$$

where $\boldsymbol{v}_i$ denotes the absolute velocity of the i-th macro-particle and c the speed of light. The force that effects the particle is the Lorenz force

$$F = q(\boldsymbol{E}_i + \boldsymbol{v}_i \times \boldsymbol{B}_i),$$

where *q* is the total charge of the macro-particle. The electric field $\boldsymbol{E}_i$ and the magnetic flux density $\boldsymbol{B}_i$ are the superposition of external and self-induced fields (the so-called space-charge forces) at the position of the i-th macro-particle. The relativistic equations of motion describing the particle dynamic read as follows [14]:

$$\begin{aligned} \frac{\mathrm{d}\gamma_i \boldsymbol{v}_i}{\mathrm{d}t} &= \frac{q}{m}(\boldsymbol{E}_i + \boldsymbol{v}_i \times \boldsymbol{B}_i), \\ \frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} &= \boldsymbol{v}_i = \frac{\gamma_i \boldsymbol{v}_i}{\sqrt{\gamma_i \boldsymbol{v}_i^2/c^2 + 1}}, \qquad i = 1, \ldots, M. \end{aligned} \tag{1}$$

Here $\mathrm{d}\boldsymbol{x}_i$ denotes the change of the position of the i-th particle over the discrete time step $\mathrm{d}t$ and M represents the total number of macro-particles. The equations allow to calculate the motion of each macro-particle over a discrete period of time in which the position of the particles changes. This is nummerically realized by means of a Runge-Kutta scheme in tracking codes such as ASTRA [4], GPT [13]. As the disposition changes, changes as well the space-charge field which effects each macro-particle. Thus there is a need to calculate the electric field in each discrete time step in order to track the movement of the particles. The electric field

is calculated in the frame that is moving along with the same velocity as the bunch itself, it is said that the problem is considered in a *rest frame*.

The calculation of the fields from spatially distributed charges requires a solution of the Poisson equation [15, Page 23]

$$-\Delta\varphi = \frac{\varrho}{\varepsilon_0}, \tag{2}$$

where ε_0 is the dielectric constant and ϱ the charge density. In the following two domains are considered in which the Poisson equation has to be solved. One is the rectangular box domain Γ with $Dirichlet$ boundary conditions on $\partial\Gamma_1$ and $open$ boundary conditions on $\partial\Gamma_2$

$$\begin{aligned} -\Delta\varphi &= \frac{\varrho}{\varepsilon_0} && \text{in } \Gamma \subset \mathbb{R}^3, \\ \varphi &= g && \text{on } \partial\Gamma_1, \\ \frac{\partial\varphi}{\partial n} + \frac{1}{r}\varphi &= 0 && \text{on } \partial\Gamma_2, \end{aligned} \tag{3}$$

where

$\Gamma = [a_x, b_x] \times [a_y, b_y] \times [a_z, b_z]$, and the boundary $\partial\Gamma$ is splited into $\partial\Gamma_1$ and $\partial\Gamma_2$ with $\partial\Gamma = \partial\Gamma_1 \bigcup \partial\Gamma_2$.

The second domain Ω is a cylindrical structure with an elliptic cross-section, where (2) is considered with the following boundary conditions:

$$\begin{aligned} -\Delta\varphi &= \frac{\varrho}{\varepsilon_0} && \text{in } \Omega \subset \mathbb{R}^3, \\ \varphi &= 0 && \text{on } \partial\Omega_1, \\ \frac{\partial\varphi}{\partial n} + \frac{1}{r}\varphi &= 0 && \text{on } \partial\Omega_2, \end{aligned} \tag{4}$$

where $\partial\Omega_1$ is the coating of the cylinder with

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \text{ and } z_a < z < z_b,$$

$\partial\Omega_2$ are the two elliptical bases of the cylinder satisfying

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1$$

and being perpendicular to the z-axis at $z = a_z$ and $z = b_z$. The boundary condition $\varphi = 0$ on $\partial\Omega_1$ means that the surface of the cavities and the beampipe in general act as an ideal electrical conductor (the cavities from Figure 1 are made of superconducting material [2]). The open boundary condition in z-direction approximates the indefinitely long cylinder on a finite computational domain.

The discretization volume in which the cylindrical computational domain Ω is embedded (see Figure 2) is the same rectangular box Γ as in (3). Following the notations in [11] the box Γ is discretized along the x-, y- and z-axis in N_x, N_y and N_z subintervals, respectively. The x-coordinate is discretized by N_x subintervals $h_{x,0}$, $h_{x,1}$, ..., h_{x,N_x-1} with $b_x - a_x = \sum_{i=0}^{N_x-1} h_{x,i}$. Analogously, the y- and z-coordinate are discretized by N_y and N_z subintervals.

Further we introduce

$$\tilde{h}_{x,i} = \begin{cases} \dfrac{h_{x,i-1} + h_{x,i}}{2}, & i = 1, \ldots, N_x - 1 \\ \dfrac{h_{x,i}}{2}, & i = 0,\ N_x \end{cases} \tag{5}$$

($\tilde{h}_{y,i}$, $i = 0, 1, \ldots, N_y$ and $\tilde{h}_{z,i}$, $i = 0, 1, \ldots, N_z$ in the same way) which in Finite Integration Technique is known as mesh spacing (edges) on the dual grid.

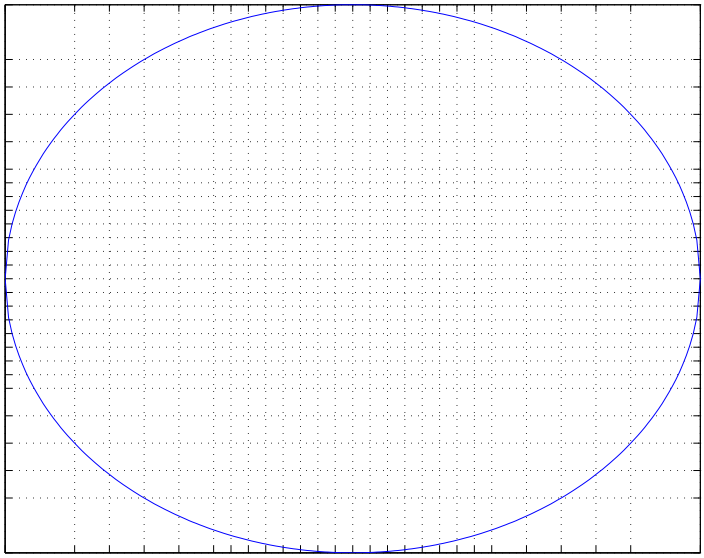

Figure 2: Elliptic cross-section of the domain Ω embedded in Γ.

The discretization of the second order derivative with second order finite differences in the general case gives

$$\frac{\partial^2 \varphi(x_i, y_j, z_k)}{\partial x^2} \approx \frac{\varphi(x_{i-1}, y_j, z_k)}{\tilde{h}_{x,i} h_{x,i-1}} - \frac{2\varphi(x_i, y_j, z_k)}{h_{x,i} h_{x,i-1}} + \frac{\varphi(x_{i+1}, y_j, z_k)}{h_{x,i} \tilde{h}_{x,i}}. \tag{6}$$

Let $\varphi_{i,j,k} = \varphi(x_i, y_j, z_k)$. Than the discretization of the Poisson equation with second order finite differences on the above described non-equidistant mesh leads to the following system of equations:

$$\begin{aligned} & \tilde{h}_{y,j}\tilde{h}_{z,k}\left(-\tfrac{1}{h_{x,i-1}}\varphi_{i-1,j,k} + \left(\tfrac{1}{h_{x,i-1}} + \tfrac{1}{h_{x,i}}\right)\varphi_{i,j,k} - \tfrac{1}{h_{x,i}}\varphi_{i+1,j,k}\right) \\ + \ & \tilde{h}_{x,i}\tilde{h}_{z,k}\left(-\tfrac{1}{h_{y,j-1}}\varphi_{i,j-1,k} + \left(\tfrac{1}{h_{y,j-1}} + \tfrac{1}{h_{y,j}}\right)\varphi_{i,j,k} - \tfrac{1}{h_{y,i}}\varphi_{i,j+1,k}\right) \\ + \ & \tilde{h}_{x,i}\tilde{h}_{y,j}\left(-\tfrac{1}{h_{z,k-1}}\varphi_{i,j,k-1} + \left(\tfrac{1}{h_{z,k-1}} + \tfrac{1}{h_{z,k}}\right)\varphi_{i,j,k} - \tfrac{1}{h_{z,k}}\varphi_{i,j,k+1}\right) \\ = \ & \tilde{h}_{x,i}\tilde{h}_{y,j}\tilde{h}_{z,k} f_{i,j,k} \end{aligned} \tag{7}$$

for $i = 1, \ldots, N_x - 1$, $j = 1, \ldots, N_y - 1$, $k = 1, \ldots, N_z - 1$. The same system of equations is obtained with the application of the Finite Integration Technique (FIT) which has been introduced by Weiland [16].

For a compact notation the Kronecker product '$\otimes$' for matrices is used which is defined as (see e. g. [8])

$$\boldsymbol{A} \otimes \boldsymbol{B} \quad := \quad \begin{pmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \ddots & \ddots \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & a_{23}\boldsymbol{B} & \ddots \\ & & \ddots & \\ & \ddots & a_{nn-1}\boldsymbol{B} & a_{nn}\boldsymbol{B} \end{pmatrix}.$$

The above equation (7) on Γ reads in matrix vector notation as

$$\boldsymbol{A}\boldsymbol{\varphi} = \tilde{\boldsymbol{H}}_z \otimes \tilde{\boldsymbol{H}}_y \otimes \tilde{\boldsymbol{H}}_x \boldsymbol{f},$$

with

$$\boldsymbol{A} = \tilde{\boldsymbol{H}}_z \otimes \tilde{\boldsymbol{H}}_y \otimes \boldsymbol{A}_x + \tilde{\boldsymbol{H}}_z \otimes \boldsymbol{A}_y \otimes \tilde{\boldsymbol{H}}_x + \boldsymbol{A}_z \otimes \tilde{\boldsymbol{H}}_y \otimes \tilde{\boldsymbol{H}}_x. \tag{8}$$

Considering Dirichlet bounadry conditions for $\partial\Gamma = \Gamma_1$ we get $\tilde{\boldsymbol{H}}_x$=$\tilde{\boldsymbol{H}}_{x,D}$ with

$$\tilde{\boldsymbol{H}}_{x,D} \quad := \quad \mathrm{diag}(\tilde{h}_{x,1}, \tilde{h}_{x,2} \ldots, \tilde{h}_{x,N_x-1}),$$

and $\boldsymbol{A}_x$=$\boldsymbol{A}_{x,D}$ with

$$\boldsymbol{A}_{x,D} \quad := \quad \begin{pmatrix} \left(\frac{1}{h_{x,0}} + \frac{1}{h_{x,1}}\right) & -\frac{1}{h_{x,1}} & & \\ -\frac{1}{h_{x,1}} & \left(\frac{1}{h_{x,1}} + \frac{1}{h_{x,2}}\right) & -\frac{1}{h_{x,2}} & \\ & & \ddots & \\ & & -\frac{1}{h_{x,N_x-2}} & \left(\frac{1}{h_{x,N_x-2}} + \frac{1}{h_{x,N_x-1}}\right) \end{pmatrix}.$$

The diagonal matrices $\tilde{\boldsymbol{H}}_y$ and $\tilde{\boldsymbol{H}}_z$ are defined analogously to $\tilde{\boldsymbol{H}}_x$ and the finite difference matrices $\boldsymbol{A}_y$ and $\boldsymbol{A}_z$ analogously to $\boldsymbol{A}_x$. Note the different dimensions of the matrices corresponding to the number of mesh lines in every coordinate direction. The vectors $\boldsymbol{f} = (f_{i,j,k})_{i=1,j=1,k=1}^{N_x-1,N_y-1,N_z-1}$ and $\boldsymbol{\varphi} = (\varphi_{i,j,k})_{i=1,j=1,k=1}^{N_x-1,N_y-1,N_z-1}$ contain the values of the right hand side and the potential at the mesh points, respectively.

The above system of equations (7) has $N_p = (N_x + 1) \times (N_y + 1) \times (N_z + 1)$ unknowns in the case of open boundary conditions on the domain Γ. The matrix $\boldsymbol{A}_x$ and the vector $\tilde{\boldsymbol{H}}_x$ in the case of open boundary conditions on $\partial\Gamma = \Gamma_2$ would have the form $\tilde{\boldsymbol{H}}_x$=$\tilde{\boldsymbol{H}}_{x,O}$ with

$$\tilde{\boldsymbol{H}}_{x,O} \quad := \quad \mathrm{diag}(\tilde{h}_{x,0}, \tilde{h}_{x,1} \ldots, \tilde{h}_{x,N_x}),$$

and $\boldsymbol{A}_x$=$\boldsymbol{A}_{x,O}$ with

$$\boldsymbol{A}_{x,O} \quad := \quad \begin{pmatrix} \left(\frac{1}{h_{x,0}} + \frac{1}{r}\right) & -\frac{1}{h_{x,0}} & & \\ -\frac{1}{h_{x,0}} & \left(\frac{1}{h_{x,0}} + \frac{1}{h_{x,1}}\right) & -\frac{1}{h_{x,1}} & \\ & & \ddots & \\ & & -\frac{1}{h_{x,N_x-1}} & \left(\frac{1}{h_{x,N_x-1}} + \frac{1}{r}\right) \end{pmatrix}.$$

Accordingly the vectors of the right hand side and the potential at the mesh points contain as well the points on the boundaries, i.e. $\boldsymbol{f} = (f_{i,j,k})_{i=0,j=0,k=0}^{N_x,N_y,N_z}$ and $\boldsymbol{\varphi} = (\varphi_{i,j,k})_{i=0,j=0,k=0}^{N_x,N_y,N_z}$. That means that the linear system of equations in case of open boundary conditions is larger (for the number of the boundary points) than the system with Dirichlet boundary conditions.

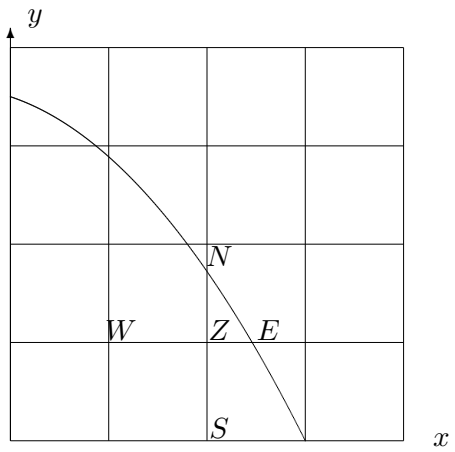

Figure 3: Two-dimensional representation of the elliptic shape of the boundary.

The same discretization of Poisson's equation (as in equation (7)) on the domain Ω is considering only the discrete points which are inside Ω. In that case the number of the unknowns is considerably smaller because in each (x, y)-plane all grid points which are outside the ellipse are skipped (Figure 3). The matrix $\boldsymbol{A}$ remains block structured but the blocks will have different dimensions (see Figure 6). For each point near the boundary $\partial\Omega$ the discretization star (equation (7)) is not entirely determinated from the mesh. The discretization star of each point next to the boundary has to be build in such a manner that it takes care of the distances to the points which lay on the elliptic shape of Ω (i.e. h_n, h_e in Figure 4).

If equation (7) is written for the point Z (two-dimensional representation in Figure 4) the distances $h_{x,i} \equiv h_e$, $h_{y,j} \equiv h_n$ and with it also $\tilde{h}_{x,i}$, $\tilde{h}_{y,j}$ will have to be extra calculated. They depend on the intersection points of the ellipse with the grid. From Figure 4 it is evident that the discretization star for the point Z will be non-symmetric because the lengths h_n, h_s, h_w, h_e are not equal. Especially it differs from the neighbouring point which is situated inside Ω. This implies that the discretization matrix $\boldsymbol{A}$ for the elliptical domian Ω will be non-symmetric. On the other hand dispite of the non-equidistant discretization the matrix $\boldsymbol{A}$ will be symmetric, if the domain is the box Γ.

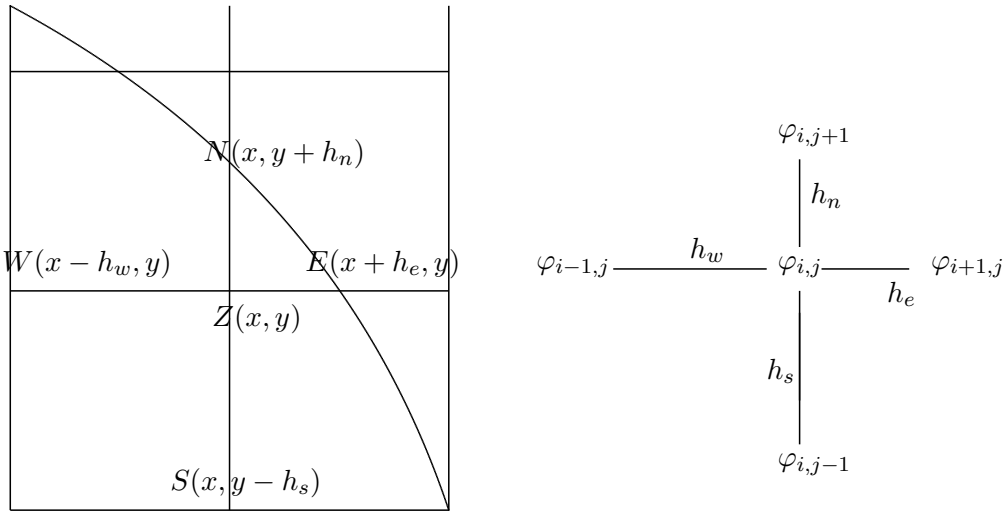

Figure 4: Non-symmetric 2-D Shortley-Weller Star.

In order to illustrate this we consider the first row in (7). For the two consecutive points m and $m+1$ from Figure 5 we obtain for $\tilde{h}_{y,j}$:

$$\tilde{h}^m_{y,j} = \frac{h_{n,m} + h_{s,m}}{2}, \quad \tilde{h}^{m+1}_{y,j} = \frac{h_{n,m+1} + h_{s,m+1}}{2},$$

where m and $m+1$ in the superscript means $\tilde{h}_{y,j}$ for the $m-th$ and $(m+1)-th$ point, respectively. The first row of equation (7) will be for the point m

$$\tilde{h}^m_{y,j}\tilde{h}^m_{z,k}\left(-\tfrac{1}{h_{x,i-1}}\varphi_{i-1,j,k} + \left(\tfrac{1}{h_{x,i-1}} + \tfrac{1}{h_{x,i}}\right)\varphi_{i,j,k} - \tfrac{1}{h_{x,i}}\varphi_{i+1,j,k}\right) \qquad (9)$$

and for $m+1$

$$\tilde{h}^{m+1}_{y,j}\tilde{h}^m_{z,k}\left(-\tfrac{1}{h_{x,i}}\varphi_{i,j,k} + \left(\tfrac{1}{h_{x,i}} + \tfrac{1}{h_{x,i+1}}\right)\varphi_{i+1,j,k} - \tfrac{1}{h_{x,i+1}}\varphi_{i+2,j,k}\right) \qquad (10)$$

The discret-star equation (7) of the points m and $m+1$ (Figure 5) determines two neighbouring rows in the matrix $\boldsymbol{A}$. The parts in (9) and (10) are giving the entries of $\boldsymbol{A}$ which are next to the main diagonal and they also contribute to the diagonal entries. From Figure 5 it is obvious that in the case of the rectangular domain $h_{s,m}$ and $h_{s,m+1}$ are equal as well as $h_{n,m}$ and $h_{n,m+1}$. Therefore the coefficients multiplying the entries in the matrix $\boldsymbol{A}$ for two consecutive rows $\tilde{h}^m_{y,j}$ and $\tilde{h}^{m+1}_{y,j}$ are equal. For symmetry of $\boldsymbol{A}$ the coefficient of $\varphi_{i+1,j,k}$ in (9) has to coincide with the coefficient of $\varphi_{i,j,k}$ in (10).

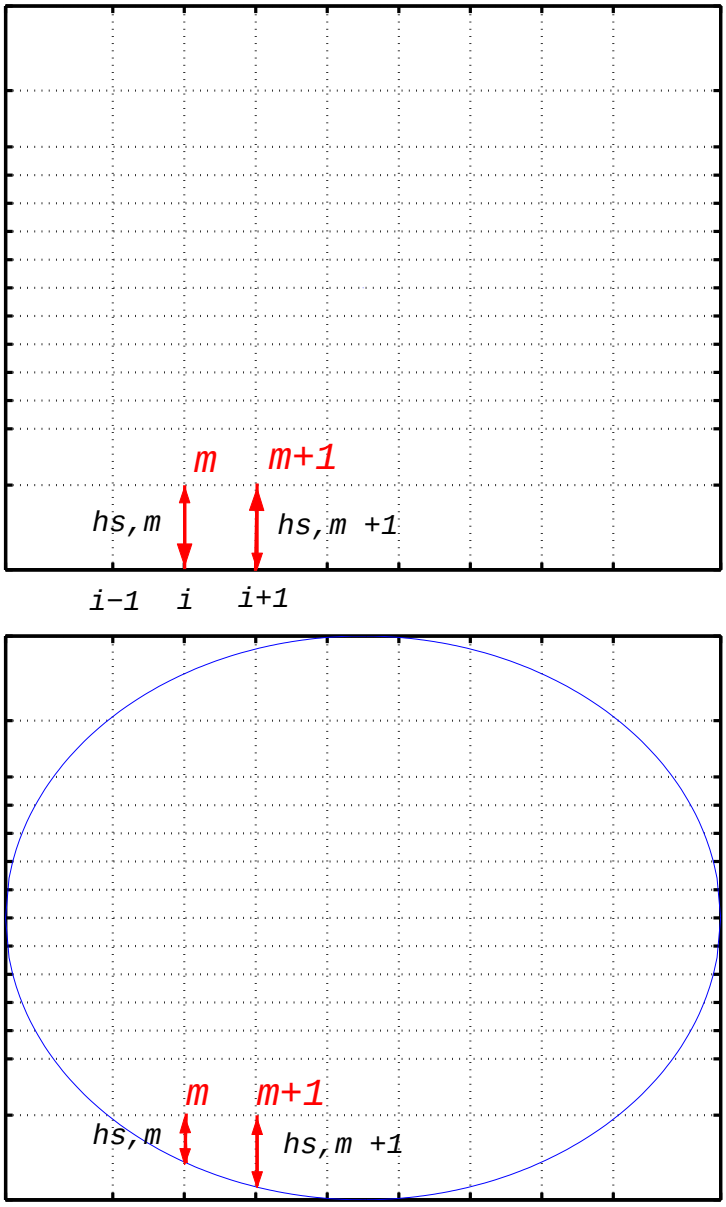

Figure 5: Two points determining two neighbouring rows in the matrix $\boldsymbol{A}$.

In case of a rectangular domain it holds that $\tilde{h}^m_{y,j}$=$\tilde{h}^{m+1}_{y,j}$ and the respective matrix entries will have the same value. In the case of the cylindrical domain Ω the same entries in the matrix $\boldsymbol{A}$ will not be equal since $h_{s,m} \neq h_{s,m+1}$ and so the coefficients $\tilde{h}^m_{y,j} \neq \tilde{h}^{m+1}_{y,j}$ for two consecutive rows of $\boldsymbol{A}$. This in general case results in a non-symmetrical matrix $\boldsymbol{A}$.

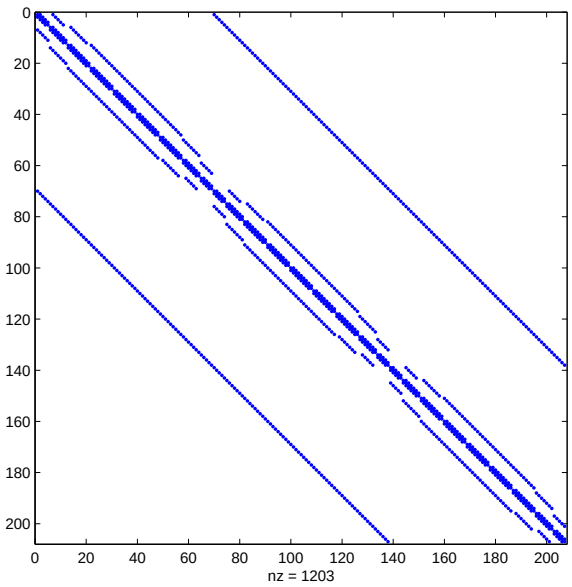

Figure 6: Structure of the matrix A: nz = non-zero elements.

However the band structure of the matrix $\boldsymbol{A}$ will remain symmetric with respect to the main diagonal as shown in Figure 6. Furthermore the matrix will remain weak diagonally dominant which eventually makes it also non-singular as it will be shown in the section 3.

3 Numerical Algorithms

As the volume Ω was discretized and the matrix $\boldsymbol{A}$ and the right hand side vector b were assigned, we got the system of linear equations

$$\boldsymbol{A}x = b, \tag{11}$$

where x is the vector of unknowns. This is a relatively large system of equations which depends on the number of discret nodes in the volume. Storage requirements as well as computation times often set limits to direct methods for the solution of such systems of linear equations. If the coefficient matrix is sparse, which is the case here (see Figure 6) then iterative methods offer an alternative. It is worth to mention that it is typical to have sparse coefficient matrices $\boldsymbol{A}$ for systems resulting from discretization of partial differential equations by finite differences, finite integration technique or finite elements.

Before proceeding to solve the emerged linear system of equations (11) and finding a suitable algorithm, we have to consider the properties of the matrix $\boldsymbol{A}$. As mentioned in section 2 the matrix $\boldsymbol{A}$ resulting from the discretization of Poisson's equation on the domain Ω is non-symmetric (unlike the symmetric matrix that we get if the discretization domain is Γ). A matrix $\boldsymbol{M}$ is called *diagonal dominant* if

$$|a_{kk}| \geq |\sum_{\substack{n=1 \\ n\neq k}}^{N} a_{kn}|, \qquad k = 1, \ldots, N. \tag{12}$$

A matrix $\boldsymbol{M}$ is *weak diagonal dominant*, if there is at least one k (one row) for which the inequality (12) is strict [9]. This is actually the case in the rows of $\boldsymbol{A}$ corresponding to the discretization points next to the boundaries of Ω, if $\boldsymbol{A}$ arises from (4).

Although the matrix $\boldsymbol{A}$ is non-symmetric it is a positive definite. For real non-symmetric matrices the necessary and sufficient condition to be positive definite is that the symmetric part $\boldsymbol{A}_s = \frac{(\boldsymbol{A}+\boldsymbol{A}^T)}{2}$, where $\boldsymbol{A}^T$ is the transpose, is positive definite [7]. In our case $\boldsymbol{A}_s$ beside symmetric is also weak diagonal dominant because both $\boldsymbol{A}$ and $\boldsymbol{A}^T$ are weak diagonal dominant. According to [9] a symmetric, weak diagonal dominant matrix with positive diagonal elements is positive definite, this qualifies $\boldsymbol{A}_s$ and with it $\boldsymbol{A}$ to be positive definite. Finally if $\boldsymbol{A}$ is positive definite than it has a positive determinant which means that $\boldsymbol{A}$ is also a non-singular.

In order to solve the system (11) in a reasonable amount of time on a normal personal computer we consider only iterative methods. For the fast iterative solution of the system there are many alternative algorithms depending on the properties of the matrix $\boldsymbol{A}$. The most straightforward iterative methods are the relaxation type methods. Typical examples are the Jacobi, Gauss-Seidel, and SOR (Successive Over Relaxation) algorithm. These classical iteration methods, also known as stationary methods, are not fast enough in comparison with instationary methods as the Krylov-subspace methods like CG (Conjugate Gradient), BiCG (Bi-Conjugate Gradient) and its derivates, the minimal residual algorithms like the GMRES (Generalized Minimal Residual method) or hybrid methods like the BiCGSTAB (Bi-Conjugate Gradient Stabilized) [15].

Since in the application of particle tracing (calculating space charge forces in discrete time steps) the same system of equation (11) has to be solved over and over

again in each discrete time step, the solving time has a crucial meaning. Therefore this work concentrates only on the fast iterative algorithms.

The discretization of Poisson's equation necessary for the calculation of space charge forces requires for different kinds of bunches either equidistant meshes with a huge number of unknowns or non-equidistant meshes which have a high aspect ratio. The aspect ratio of the mesh is defined as $A_{\text{mesh}} = h_{max}/h_{min}$, where h_{max} and h_{min} denote the global maximal and minimal step size, respectively.

The fact that geometric multigrid technique allows higher aspect ratios is used in [12] to build an adaptive mesh, to realize an appropriate resolution of the charged particle bunch with a relatively small number of mesh lines. However the geometrical multigrid technique as a solver is applicable if the matrix $\boldsymbol{A}$ carries in itself the whole lexicographic order of points in Γ. Unfortunately on the domain Ω the use of geometrical multigrid technique to solve the system would not be that straitforward because of the additional efforts in retaining the geometrical order of the points while coarsening and prolongating the grid.

It is well known [9] that the CG algorithm requires a symmetric and positive definite matrix, since the residual vector can no longer be made othogonal via a short recurrence. The BiConjugate Gradient method takes another approach, replacing the orthogonal sequence of residuals by two mutually orthogonal sequences [15], at the price of no longer providing a minimization.

The algorithms which are implemented and tested in this work to solve the nonsymmetric system of linear equations (11), are given and discussed in the following subsections. These are the BiConjugate Gradient method, the Preconditioned BiConjugate Gradient method and the BiConjugate Gradient Stabilized method.

3.1 BiConjugate Gradient method (BiCG)

The BiConjugate Gradient method uses similar relations as the Conjugate Gradient method to update the residuals, yet it is based on both the matrix $\boldsymbol{A}$ and its transposed $\boldsymbol{A}^T$. Thus we update two sequences of residuals

$$r^{(i)} = r^{(i-1)} - \alpha_i \boldsymbol{A} p^{(i)}, \qquad \tilde{r}^{(i)} = \tilde{r}^{(i-1)} - \alpha_i \boldsymbol{A}^T \tilde{p}^{(i)}$$

and two sequences of search directions

$$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)}, \qquad \tilde{p}^{(i)} = \tilde{r}^{(i-1)} + \beta_{i-1} \tilde{p}^{(i-1)}.$$

The BiCG algorithm for solving the linear system $\boldsymbol{A}x = b$ is implemented as a pair of coupled two-term recurrences as follows [5].

Algorithm 3.1 *BiConjugate Gradient method*

Given: initial guess $x^{(0)}$.
Compute $r^{(0)} = b - \boldsymbol{A}x^{(0)}$.
Choose $\tilde{r}^{(0)}$ *(for example,* $\tilde{r}^{(0)} = r^{(0)}$*).*
for i*=1,2,...*

$\rho_{i-1} = {r^{(i-1)}}^T \tilde{r}^{(i-1)}$
if $\rho_{i-1} = 0$, method fails
if $i = 1$

$p^{(i)} = r^{i-1}$
$\tilde{p}^{(i)} = \tilde{r}^{i-1}$

else

$\beta_{i-1} = \dfrac{\rho_{i-1}}{\rho_{i-2}}$
$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)}$
$\tilde{p}^{(i)} = \tilde{r}^{(i-1)} + \beta_{i-1} \tilde{p}^{(i-1)}$

endif

$q^{(i)} = \boldsymbol{A} p^{(i)}$
$\tilde{q}^{(i)} = \boldsymbol{A}^T \tilde{p}^{(i)}$
$\alpha_i = \dfrac{\rho_{i-1}}{\tilde{p}^{(i)^T} q^{(i)}}$
$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
$\tilde{r}^{(i)} = \tilde{r}^{(i-1)} - \alpha_i \tilde{q}^{(i)}$
check convergence; continue if necessery

end

The choices

$$\alpha_i = \rho_{i-1} / \tilde{p}^{(i)^T} \boldsymbol{A} p^{(i)} \qquad \text{and} \qquad \beta_{i-1} = \rho_{i-1} / \rho_{i-2}$$

ensure that the sequence generated by the algorithm satisfies the following biorthogonality and biconjugacy conditions [17]:

$$\tilde{r_n}^T r_m = 0, \;\; \tilde{p_n}^T \boldsymbol{A} p_m = 0, \;\; for \;\; m \neq n.$$

A general problem of all Lanczos-type algorithm (as BiCG and CG) is that the biorthogonality (or orthogonality in the symmetric case) of the vectors is usually lost in finite precision arithmetic. The recurrence coefficients generated in the finite precision arithmetic may be quite different from those that would be generated in exact arithmetic and thus the iterates computed in a finite precision arithmetic may differ significantly from the corresponding exact quantities [5].

Apart from this numerical constraint for symmetric, positive definite systems the BiCG method delivers the same results as CG, but at twice the cost per iteration. In practice it is also observed that the convergence behavior may be quite irregular, and the method may even break down. BiCG requires computing of two matrix-vector products, $\boldsymbol{A} p^{(i)}$ and $\boldsymbol{A}^T \tilde{p}^{(i)}$. In some applications the latter product may be impossible to perform, for instance if the matrix is not formed and the matrix-vector product is only given as an operation.

In our case in order to implement the BiCG method it was necessary to get the transposed matrix $\boldsymbol{A}^T$ from the matrix $\boldsymbol{A}$. As already mentioned the matrix $\boldsymbol{A}$ is a large, sparse matrix (Figure 6). In order to have efficient computing and good storage utilization the matrix is being saved as a set of floating point arrays which point out only on non-zero elements and their position in the matrix. This kind of so called "compact memory technique" [9] is being used in the program "MOEVE" [10]. Since this new algorithms should be a part of this software package, its

matrix representation is adopted. The actual non-zero elements in the matrix are saved in one array which has the length of the total number of non-zero elements. In this array, denoted as A in the following text, the values are saved row-wise whereby the first value written from each row is the main diagonal element. The second array JA which has the same length as A gives the column number in the matrix $\boldsymbol{A}$ for the corresponding non-zero elements from the array A. Another third array IA is needed to point on the indices which the main diagonal elements from the matrix $\boldsymbol{A}$ have in the array A. This array has the length n, as n is the dimension of the squared matrix $\boldsymbol{A}$ (the number of unknowns). These three arrays provide enough information to reconstruct the matrix $\boldsymbol{A}$. This will be illustrated with the following example matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{pmatrix} 4 & 1 & 2 & 0 \\ 3 & 7 & 0 & 5 \\ 1 & 4 & 3 & 0 \\ 2 & 0 & 2 & 9 \end{pmatrix}.$$

The corresponding array A will be

$$A = [4\ 1\ 2\ 7\ 3\ 5\ 3\ 1\ 4\ 9\ 2\ 2].$$

The array JA for the matrix $\boldsymbol{A}$ will be

$$JA = [1\ 2\ 3\ 2\ 1\ 4\ 3\ 1\ 2\ 4\ 1\ 3]$$

and the IA has the entries

$$IA = [1\ 4\ 7\ 10].$$

In order to be compatible with "MOEVE" and to take advantage of the already implemented matrix-vector products the transposed matrix $\boldsymbol{A^T}$ has to be represented in the same way, thus the vectors At, JAt and IAt have to be generated.
We search the entries belonging to a column of $\boldsymbol{A}$ one by one from the array A and place them in the array At, for each and every column of $\boldsymbol{A}$. Along with it the vectors IAt and JAt are being filled as well.

The pseudocode for this routine which is performed only once in the beginning of the BiCG method is given in the following.

Algorithm 3.2 *Transposed matrix*

Given:
Matrix $\boldsymbol{A}$ represented with A, JA, IA
N - number of unknowns (the number of the columns of $\boldsymbol{A}$)
$Neint$ length of the array A (number of non-zero elements in the matrix $\boldsymbol{A}$)

k=0
for $index = 0, 1...N$

    t=0
    for $i = 0, 1, 2...Neint$

$q = 0$
if$(JA[i] == index)$
(Comment: If the element is in the corresponding
column $index$ then find in which row q it is.)
while$(i > IA[q])$
$q++$
if$(q > (N-1)),\ break$
end (while)
if$(index == q-1)$
(Comment: The element is a main diagonal element)
$At[k] = A[i]$
$IAt[index] = k$
$JAt[k] = index$
$k = k + t + 1$
else
if $(index > (q-1))$
(Comment: The element is over the main diagonal)
$t++$
$At[k+t] = A[i]$
$JAt[k+t] = q-1$
else
if$(index < (q-1))$
(Comment: The element is under the main diagonal)
$At[k] = A[i]$
$JAt[k] = q-1$
$k++$
endif
endif
endif
end (for)
end

3.2 Preconditioned BiConjugate Gradient Method (PBiCG)

The pseudocode for the Preconditioned BiConjugate Gradient Method with (PBiCG) preconditioner matrix $\boldsymbol{M}$ is as follows [5]:

Algorithm 3.3 *PBiCG*

Given: initial approximation $x^{(0)}$.
Compute $r^{(0)} = b - \boldsymbol{A}x^{(0)}$.
Choose $\tilde{r}^{(0)}$ *(for example,* $\tilde{r}^{(0)} = r^{(0)}$*).*
for i*=1,2,...*

solve $\boldsymbol{M}z^{(i-1)} = r^{(i-1)}$
solve $\boldsymbol{M}^T\tilde{z}^{(i-1)} = \tilde{r}^{(i-1)}$
$\rho_{i-1} = {z^{(i-1)}}^T\tilde{r}^{(i-1)}$
if $\rho_{i-1} = 0$, method fails
if $i = 1$

$p^{(i)} = z^{i-1}$
$\tilde{p}^{(i)} = \tilde{z}^{i-1}$

else

$\beta_{i-1} = \dfrac{\rho_{i-1}}{\rho_{i-2}}$
$p^{(i)} = z^{(i-1)} + \beta_{i-1}p^{(i-1)}$
$\tilde{p}^{(i)} = \tilde{z}^{(i-1)} + \beta_{i-1}\tilde{p}^{(i-1)}$

endif
$q^{(i)} = \boldsymbol{A}p^{(i)}$
$\tilde{q}^{(i)} = \boldsymbol{A}^T\tilde{p}^{(i)}$
$\alpha_i = \dfrac{\rho_{i-1}}{{\tilde{p}^{(i)}}^T q^{(i)}}$
$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
$\tilde{r}^{(i)} = \tilde{r}^{(i-1)} - \alpha_i\tilde{q}^{(i)}$
check convergence; continue if necessery

end

As a preconditioning matrix $\boldsymbol{M}$ we used the diagonal of the matrix $\boldsymbol{A}$. This is known as the (point) Jacobi preconditioner. It is not difficult to realize since each entry of the residual vector has to be divided by the corresponding diagonal element form the matrix $\boldsymbol{A}$: $z[i] = r[i]/\boldsymbol{A}[\boldsymbol{I}\boldsymbol{A}[i]]$.

3.3 BiConjugate Gradient Stabilized Algorithm (BiCGSTAB)

The biconjugate gradient stabilized algorithm (BiCGSTAB) is being used to eliminate the need of the transposed matrix $\boldsymbol{A}^{\boldsymbol{T}}$ which is necessary in the BiCG algorithm. BiCGSTAB was developed to solve non-symmetric linear systems. It also smooths the convergence of the Conjugate Gradient Squared method (CGS) [15] upon which it is built. CGS is the first of this class of techniques referred to as transpose-free variants of the bi-conjugate gradient method [15].

Algorithm 3.4 *BiCGSTAB*

Given: initial approximation x_0.
Compute $r^{(0)} = b - \boldsymbol{A}x^{(0)}$ *and set* $p_0 = r_0$.
Choose $\tilde{r}^{(0)}$ *such that* ${r_0}^T\tilde{r_0} \neq 0$ *(for example,* $\tilde{r}^{(0)} = r^{(0)}$*).*
for i*=1,2,...*

$q_p^{(i-1)} = \boldsymbol{A}p^{(i-1)}$

$$a_{i-1} = \frac{r^{(i-1)^T}\tilde{r}^0}{q_p^{(i-1)^T}\tilde{r}^0}$$

Set

$$x^{(i-1/2)} = x^{(i-1)} + a_{i-1}p^{(i-1)}$$

$$r^{(i-1/2)} = r^{(i-1)} - a_{i-1}q_p^{(i-1)}$$

$$q_r^{(i-1)} = \boldsymbol{A}r_{i-1/2}$$

$$\omega_i = \frac{r_{i-1/2}^T q_r^{(i-1)}}{q_r^{(i-1)^T} q_r^{(i-1)}}.$$

Set

$$x^{(i)} = x^{(i-1/2)} + \omega_i r^{(i-1/2)}$$

$$r^i = r^{i-1/2} - \omega_i q_r^{(i-1)}$$

$$b_i = \frac{a_{i-1}}{\omega_i}\frac{r^{i^T}\tilde{r}^0}{r^{(i-1)^T}\tilde{r}^0}$$

$$p^i = r^i + b_i(p^{i-1} - \omega_i q_p^{(i-1)}).$$

end

The numerical test examples of the next two sections (4 and 5) show that beside the stabilizing effect the BiCGSTAB method is much faster than BiCG and PBiCG.

4 Comparison of the Solvers with a Test Case Function

In order to perform numerical tests we need to have the particle charge distribution ϱ for which we know the analytical solution of the Poisson equation, namely the potential distribution φ as a function of the space coordinates (x, y, z). For the purpose of these numerical test simulations we choose the elliptical domain Ω to be a cylinder. Hence the Poisson equation is given as:

$$\begin{aligned} -\Delta\varphi(x,y,z) &= f(x,y,z) && \text{in } \Omega \\ \text{with } \Omega: &= \{(x,y,z) \in \mathbb{R}^{\mathbf{3}}: && \sqrt{x^2+y^2} < \tfrac{1}{2},\ |\ z\ | < \tfrac{1}{2}\}, \\ \varphi &= 0 && \text{on } \partial\Omega. \end{aligned} \tag{13}$$

A particulary suitable potential distribution function for this case will be a rotation symmetrical distribution with respect to the z-axis (the axis along the way of the bunch). Furthermore we take a function that is also symmetrical with respect to the (x, y)-plane at z=0. The following function given in cylindric coordinates fits those requirements:

$$\varphi(r,\phi,z) = (1-2r)^3(6r+1)\sin(\pi(z-0.5)) \tag{14}$$

with

$$r = \sqrt{x^2+y^2} < \frac{1}{2}, \quad -\frac{1}{2} < z < \frac{1}{2}.$$

For this analytical expression of the potential φ we calculate the right hand side of the Poisson equation (13) by applying the Laplace operator Δ on $\varphi(r, \phi, z)$. The Laplace operator for cylindric coordinates has the following form:

$$\Delta\varphi(r,\phi,z) = \frac{1}{r}\frac{\partial}{\partial r}(\rho\frac{\partial\varphi}{\partial r}) + \frac{1}{r^2}\frac{\partial^2\varphi}{\partial\phi^2} + \frac{\partial^2\varphi}{\partial z^2}.$$

Because of the rotation symmetrical distribution the second term $\frac{\partial^2\varphi}{\partial\phi^2} = 0$, thus we get only:

$$\Delta\varphi(r,\phi,z) = \frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial\varphi}{\partial r}) + \frac{\partial^2\varphi}{\partial z^2}. \tag{15}$$

For the first part of the equation above we obtain

$$\frac{\partial}{\partial r}(r\frac{\partial\varphi}{\partial r}) = -96r(1-2r)(1-4r)\sin(\pi(z-0.5)) \tag{16}$$

and for the second

$$\frac{\partial^2\varphi}{\partial z^2} = -(1-2r)^3(6r+1)\pi^2\sin(\pi(z-0.5)). \tag{17}$$

Finally from (16) and (17) we get the right hand side f of the Poisson equation (13):

$$f = (96(1-4r) + (1-2r)^2(6r+1)\pi^2)(1-2r)\sin(\pi(z-0.5)). \tag{18}$$

We discretize (18) in the elliptical domain Ω (13) which is embedded in Γ as defined in (3) with unit side length in each direction. The end coordinates of Γ are $a_x = -0.5$, $b_x = 0.5$, $a_y = -0.5$, $b_y = 0.5$ $a_z = -0.5$, $b_z = 0.5$.

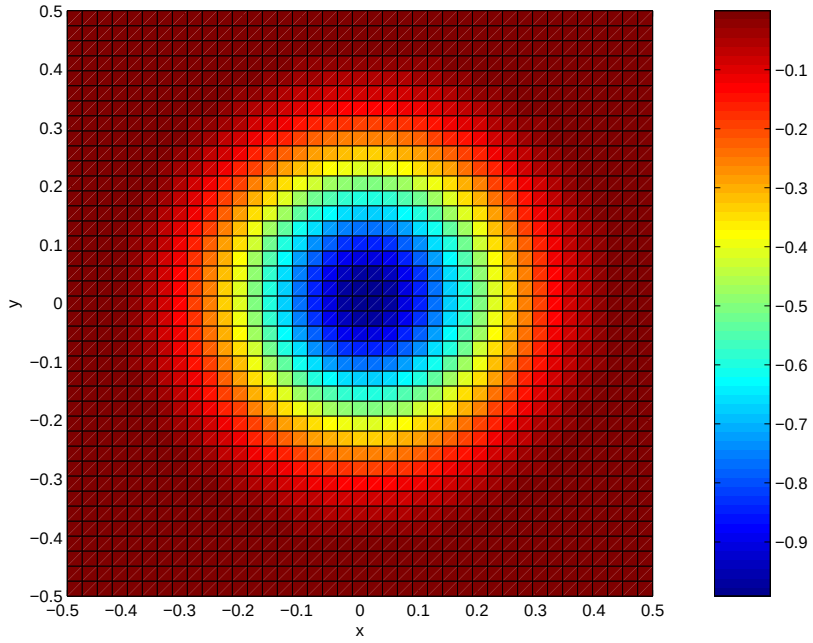

Figure 7: Simulated potential distribution, (x, y)-plane cross-section at $z = 0$.

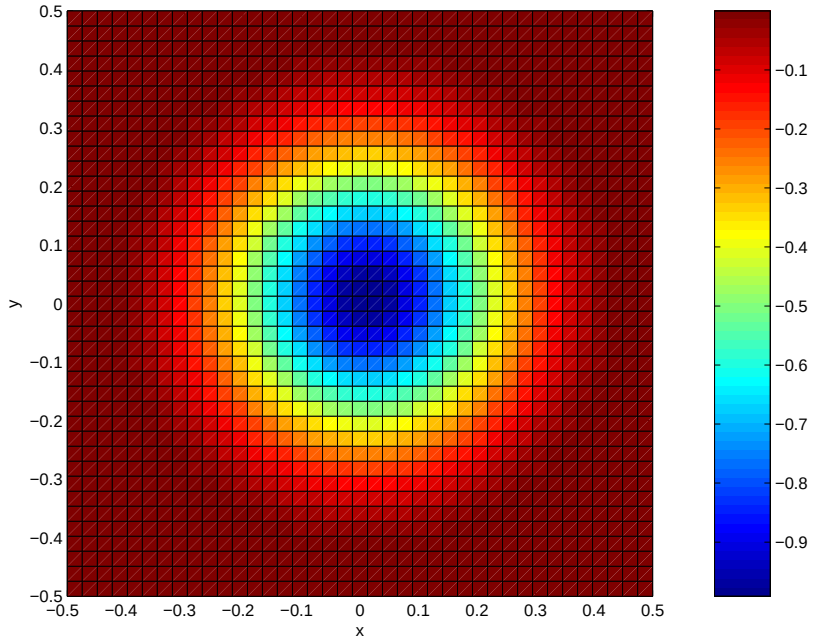

Figure 8: Analytical potential distribution, (x, y)-plane cross-section at $z = 0$.

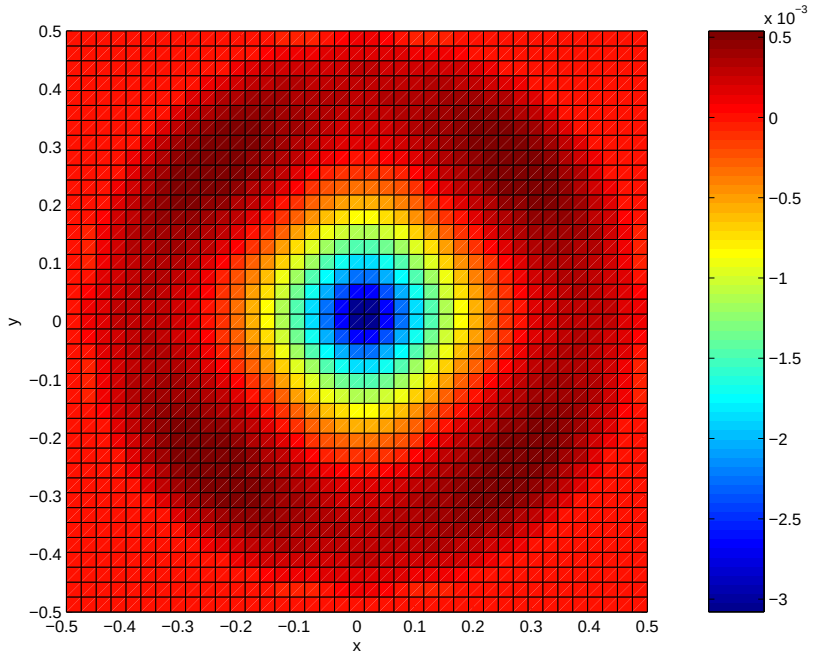

Figure 9: Numerical error of the potential distribution, (x, y)-plane cross-section at $z = 0$.

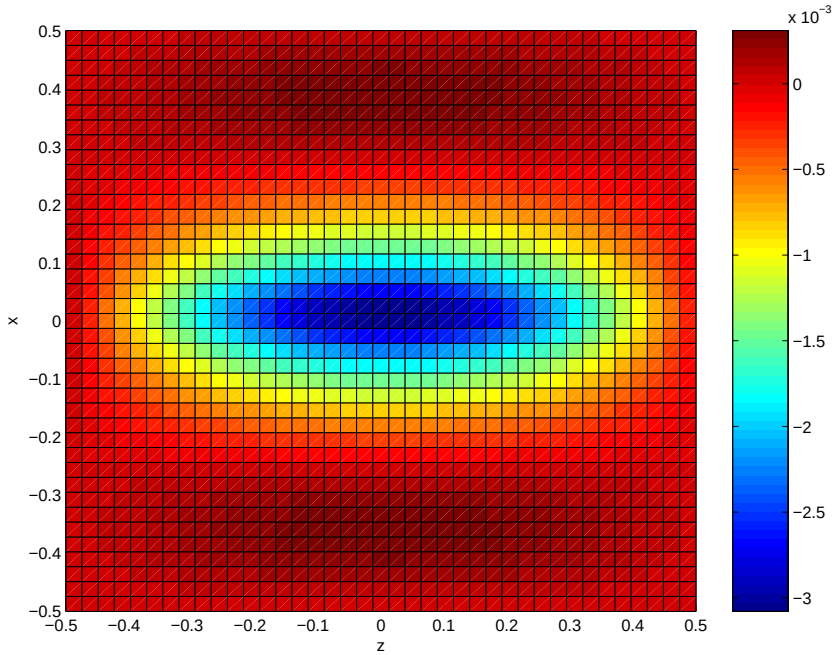

Figure 10: Numerical error of the potential distribution, (z, x)-plane cross-section at $y = 0$.

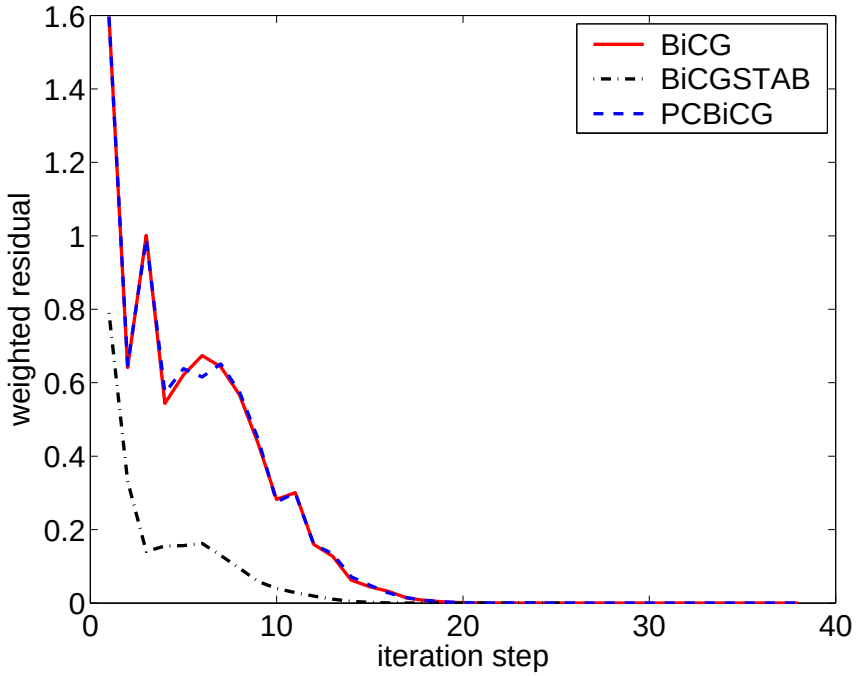

Figure 11: Weighted residual vs. iteration step for an equidistant grid with $N_x = N_y = N_z = 30$.

The simulated (Figure 7) as well as the analytical (Figure 8) potential distribution are displayed at the cross-section of the beam pipe at $z = 0$. Although the Figures (7) and (8) coincide there is a difference between the exact analytical values of the potential and the simulated ones. The relative error between the analytical and the simulated potential is shown in Figure 9 and Figure 10. Nevertheless a general conclusion from the Figures 7 to 10 will be a validation of the 3D numerical computation of space-charge fields in a beam pipe with elliptical cross-section.

The different algorithms given in the previous section have been performed until the relative residual was smaller than 10^{-6} in the maximum norm, where the relative residual is given by $\| r^{(k)} \| / \| r^{(0)} \|$, with $\| r^{(k)} \| = \| f - \boldsymbol{A}\varphi^{(k)} \|$.

In Figures 11 to 16 the development of the relative residual is shown for each method, for different number of grid points ($N_x = N_y = N_z = 30$ and $N_x = N_y = N_z = 40$) and for equidistant as well as for non-equidistant distribution of the mesh points. From Figures 12, 14 and 16 it is evident that the BiCGSTAB method converges in much less iteration steps than the BiCG or the pre-conditioned BiCG algorithm. Comparing Figure 12 and 14 we see that with rising number of grid points the PBiCG starts to perform better than BiCG. In Figures 11, 13 and 15 can be spotted a certain stabilization effect while preconditioning, but still the performance of BiCGSTAB justifies its name. However a general conclusion is that at bigger number of unknowns all methods are becoming less stabile and certainly need more iteration steps for the solution. For the non-equidistant discretization (Figure 16) all algorithms perform slower but the convergence is more stabile than in the case of an equidistant grid (Figure 14).

Although the BiCG method can even become instable for some problems and

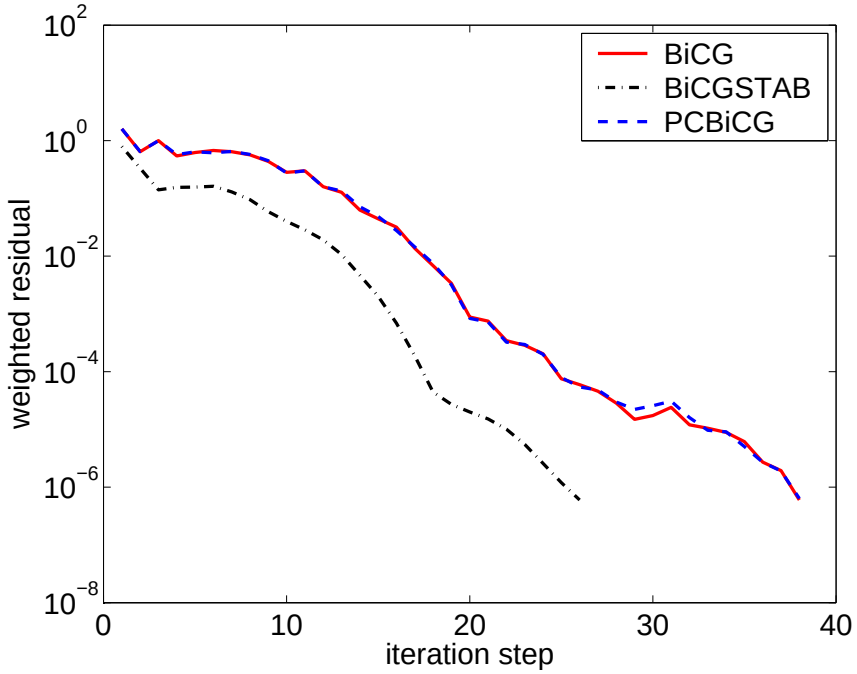

Figure 12: Logarithmic scaling of the weighted residual vs. iteration step for an equidistant grid with $N_x = N_y = N_z = 30$.

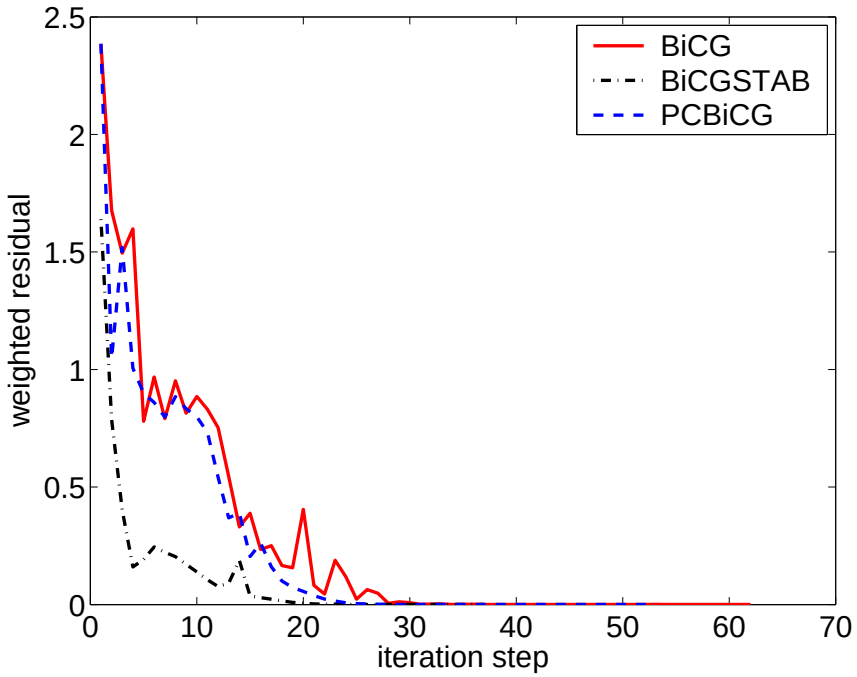

Figure 13: Weighted residual vs. iteration step for an equidistant grid with $N_x = N_y = N_z = 40$.

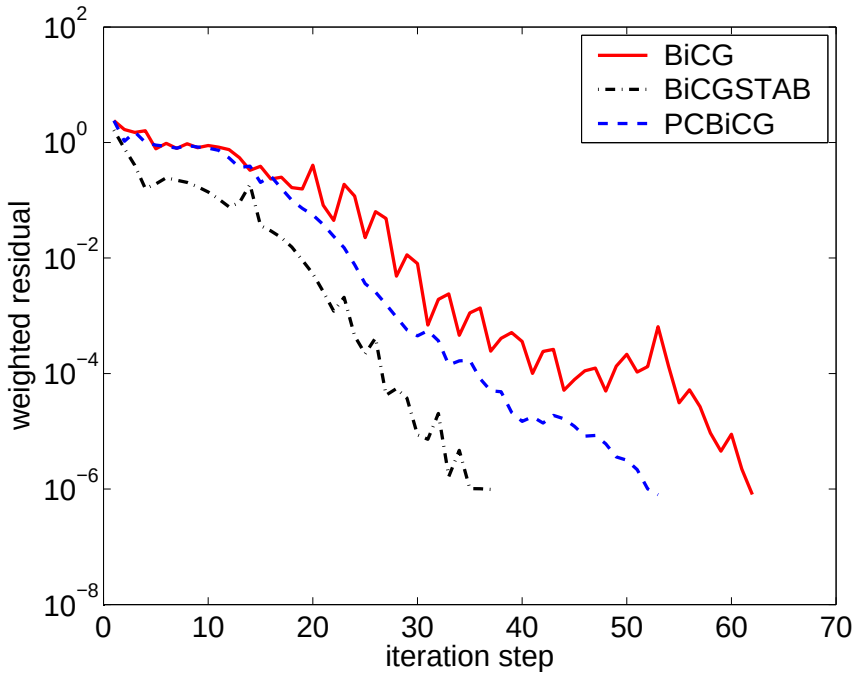

Figure 14: Logarithmic scaling of the weighted residual vs. iteration step for an equidistant grid with $N_x = N_y = N_z = 40$.

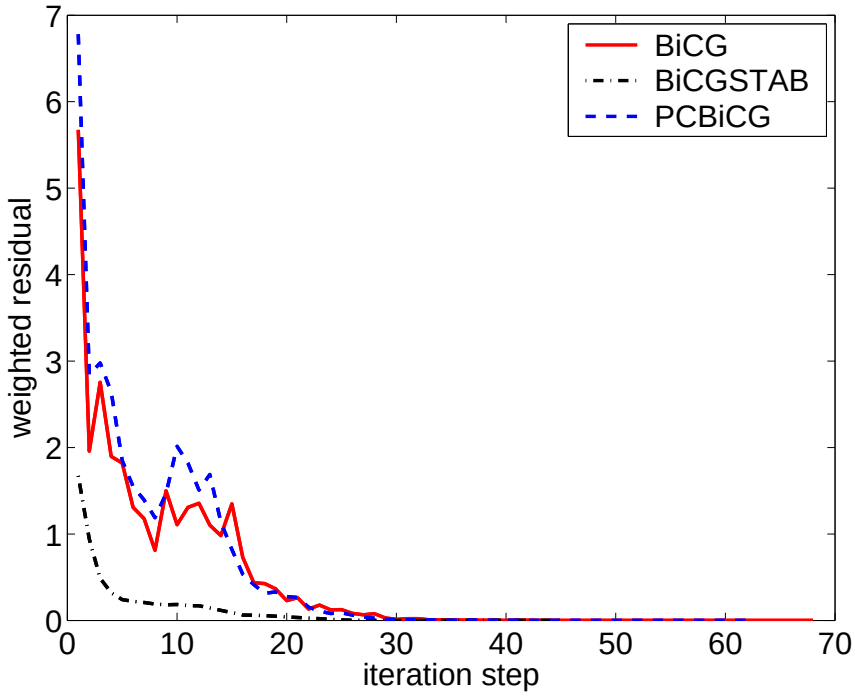

Figure 15: Weighted residual vs. iteration step for an non-equidistant grid with $N_x = N_y = N_z = 40$.

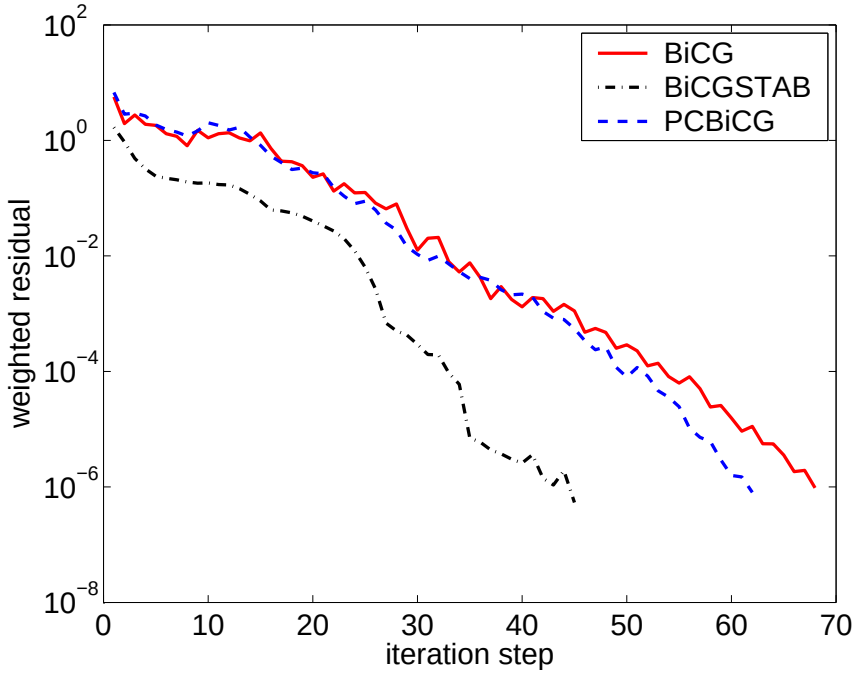

Figure 16: Logarithmic scaling of the weighted residual vs. iteration step for an non-equidistant grid with $N_x = N_y = N_z = 40$.

eventually diverge [5], here we have not experienced such a behavior.

Nevertheless the main drawback for the use of BiCG (and the same applies for PBiCG) here is that we have to find the transpose of the coefficient matrix $\boldsymbol{A^T}$. The proposed algorithm (3.2) performs it at the cost of $N \cdot Neint$ operation, where N is the number of the unknowns and the $Neint$ is number of non-zero elements in the matrix $\boldsymbol{A}$ which is certainly less than $7 \cdot N$. Due to the assigning of $\boldsymbol{A^T}$ the BiCG and PBiCG algorithm have a much longer performance than BiCGSTAB algorithm, as it can be observed from the Figures (17) and (18). For instance if $N_x = N_y = N_z = 40$, BiCG and PBiCG need more than 120 second to solve the system whereas with BiCGSTAB takes less than 2 second.

It is obvious that for the application in particle tracking BiCG and PCBiCG are not suited because we need the transpose of the coefficient matrix $\boldsymbol{A^T}$.

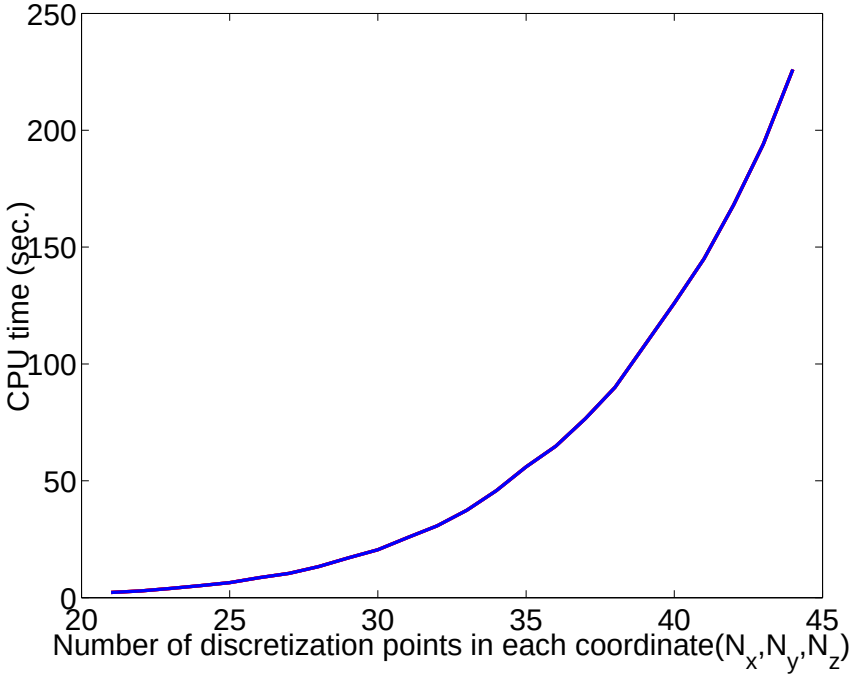

Figure 17: Computation time of BiCG for gradually increasing number of discretization points in each coordinate (N_x,N_y,N_z).

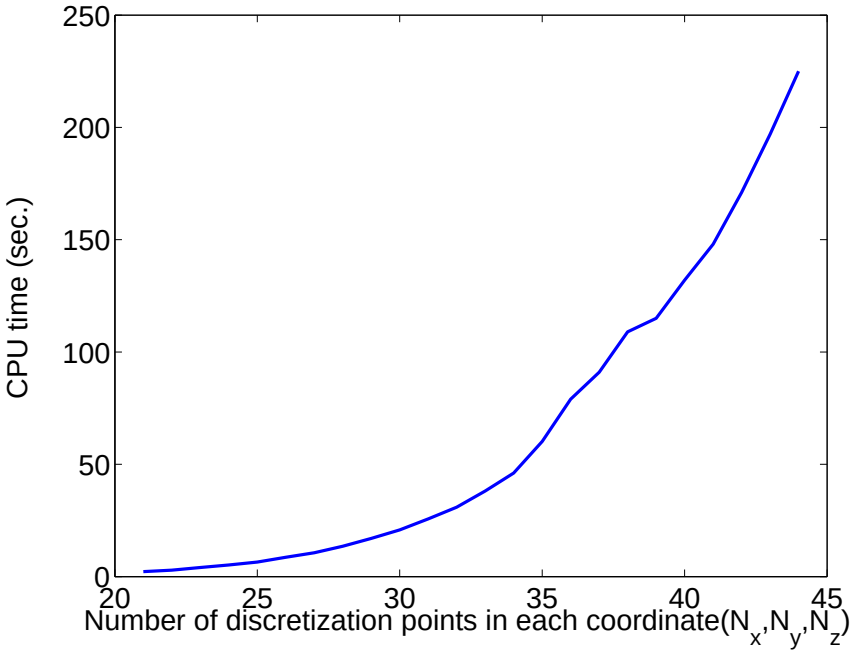

Figure 18: Computation time of PCBiCG for gradually increasing number of discretization points in each coordinate(N_x,N_y,N_z).

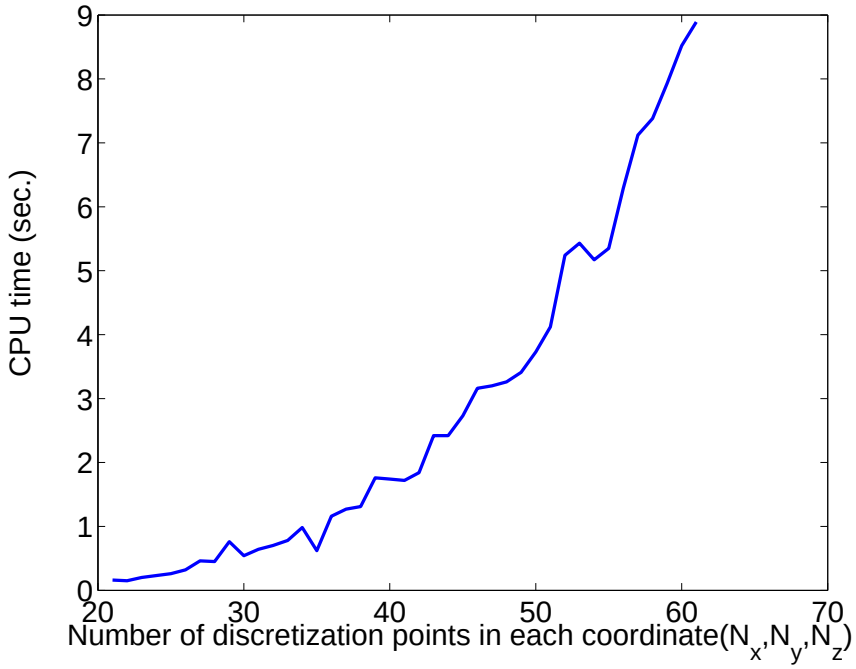

Figure 19: Computation time of BiCGSTAB for gradually increasing number of discretization points in each coordinate(N_x,N_y,N_z).

5 Space-Charge Fields of Spherical Electron Bunches Computed in a Beam Pipe

Numerical simulations with the program "MOEVE" [10] employ solvers based on the geometrical multigrid method for the fast calculation of space-charge fields in the cuboidal domain Γ (defined in (3)). Furthermore it is presumed that the real elliptical boundary of the beam pipe lies far enough from the bunch so that its influence on the field distribution can be neglected. This presumption allows the rectangular cross-section in the (x, y)-plane of the discretization domain Γ to be smaller than the real cross-section of the beam pipe, as shown in Figure 20. Consequently on the boundaries of such a reduced domain which incorporates the bunch, we define open boundary conditions. However solving equation (3) by applying open boundary conditions on the rectangular surfaces of Γ does not match the actual geometry and the boundary conditions of the beam pipe. On the other hand we define the domain Ω as given in (4) to match the real dimensions of the beam pipe. The elliptical boundary $\partial\Omega$ of the domain Ω with the potential $\varphi = 0$ gives a good approximation of the conducting or superconducting surface of the beam pipe. Hence it is interesting to calculate the space-charge fields in the domain Ω and investigate the effects of the elliptical boundaries on those fields.

One particular question that arises is in how far the simulations in the reduced domain Γ (with open boundary conditions) differs from the simulations in Ω taking into account the elliptical conductive boundaries.

In order to answer this question we compare the potential distribution calculated in the discretization domain Ω as defined in (4), with the potential distribution

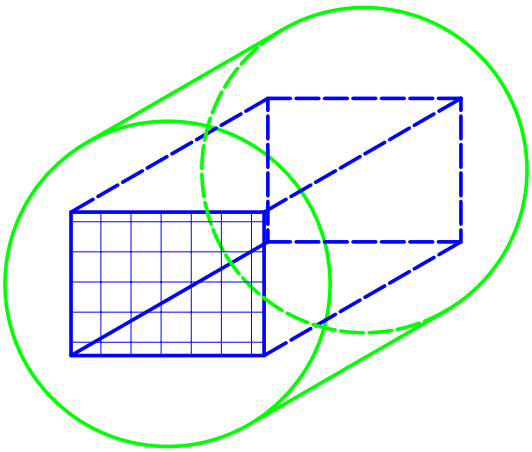

Figure 20: The rectangular domain Γ inside the beam pipe.

calculated in a smaller rectangular domain Γ as defined in (3) where $\partial\Gamma_2$ includes all rectangular boundary surfaces of Γ, meaning that open boundary condition are defined on each of them. We calculate the difference between the potentials we obtained in the simulations in Ω and Γ at each mesh point inside the discretization domain Γ. Thus we have to provide that the grid in the space of Γ is the same for both discretization domains. Obviously the resulting linear system of equations from (4) would be larger for the number of points in Ω which are outside of the domain Γ.

After providing that the grid in the space of Γ is the same for both discretization domains Γ and Ω, we can calculate the space-charge fields. As a model for an electron bunch we assume a sphere with homogeneous charge distribution. For the radius of the sphere we take $R_{sph} = 0.2$. As a typical values of the total charge that one bunch may carry we take $Q = 10^{-9}$C, dividing it with the volume of the sphere and the vacuum permittivity ε_0 we obtain the right hand side of the Poisson equation (2) :

$$f(x,y,z) = \frac{\varrho}{\varepsilon_0} = \begin{cases} \dfrac{3Q}{4\pi\varepsilon_0 R_{sph}^3} & \sqrt{(x^2+y^2+z^2)} \le R_{sph}, \\ 0 & \sqrt{(x^2+y^2+z^2)} > R_{sph}. \end{cases} \tag{19}$$

In the numerical simulations that follow the linear system of equations from the discretization of (4) is being solved with the BiCGSTAB (Algorithm 3.4), while the linear system of equations resulting from (3) is being solved with the multigrid algorithm, which is part of the software package "MOEVE".

Two elliptical domains have been considered. The first one is a cylinder with the circular cross-section:

$$\Omega_1 = \{(x,y,z) \in \boldsymbol{R}^3 : \sqrt{x^2+y^2} < 1.9, \ \ \mid z \mid < 1\}.$$

The number of the mesh points in each direction are chosen as $N_x = 51$, $N_y = 51$, $N_z = 41$. The second one is a cylinder with the elliptical cross-section with half-

axes $a = 1.6$ and $b = 2.3$:

$$\Omega_2 = \{(x, y, z) \in \boldsymbol{R}^3 : \sqrt{\frac{x^2}{a^2} + \frac{y^2}{b^2}} < 1, \ \mid z \mid < 1\}.$$

Here the number of the mesh points in each direction, are $N_x = 43$, $N_y = 59$, $N_z = 41$.

Based inside of these elliptical domains Ω_1 and Ω_2 we define the cuboids Γ with different quadratic cross-sections in the (x, y)-plane. They have a common center of axis with the domain Ω_1 or Ω_2. Furthermore we perform simulations with both equidistant and non-equidistant mesh distribution. The number of mesh points in each direction in the domains Ω_1 and Ω_2 remains the same in both cases only the step sizes are different. Since the same meshing have to be adopted in the cuboids Γ we chose their sides such that they coincide with the mesh lines of the larger Ω domain and also taking care that they have to be completely inside the elliptical domain. Thus the domians Γ have the form

$$\Gamma = [-a, a] \times [-a, a] \times [-1, 1].$$

The sides of the domains Γ depend only on the disceretization. Therefore they are different for non-equidistant and equidistant grids. In the following tables the side coordinates $\mid x \mid = \mid y \mid = \mid a \mid$ are given as well as the number of the discretization points in each direction, note that for the z-direction there is always the same number of discretization points since that side remains unchanged.

a	$N_x = N_y$	N_z	MAXIMUM RELATIVE DIFFERENCE
0.72	19	41	0.0737
0.8	21	41	0.0760
0.96	25	41	0.0787
1.2	31	41	0.0789
1.36	35	41	0.0790

Table 1: Maximum relative difference of the potential calculated in different domains Γ and in the domain Ω_1 for equidistant grid.

a	$N_x = N_y$	N_z	MAXIMUM RELATIVE DIFFERENCE
0.56	19	41	0.0996
0.65	21	41	0.1062
0.83	25	41	0.1134
1.1	31	41	0.1172
1.28	35	41	0.1177

Table 2: Maximum relative difference of the potential calculated in different domains Γ and in the domain Ω_1 for non-equidistant grid.

The maximum difference between the value of the potential simulated in one Ω domain (Ω_1 or Ω_2) and the potential simulated in the corresponding Γ domain

a	$N_x = N_y$	N_z	MAXIMUM RELATIVE DIFFERENCE
0.56	19	41	0.1314
0.65	21	41	0.1383
0.83	25	41	0.1459
1.1	31	41	0.1498
1.28	35	41	0.1503

Table 3: Maximum relative difference of the potential calculated in different domains Γ and in the domain Ω_2 for non-equidistant grid.

at the same point, divided by the maximal value of the potential in Ω we get the maximum relative difference given in the tables. As it can be realized from the tables and the Figures 23 to 37 the difference between the potential simulated in one Ω domain and the potential simulated in the corresponding Γ domain rises as the domain Γ gets larger.

Applying open boundary conditions on Γ was conditional since we assumed that the real elliptical boundary of the beam pipe lies far enough from the bunch so that its influence on the field distribution can be neglected. However as the domain Γ gets larger it is also nearer to the real elliptic boundary of the beam pipe so the open boundary conditions on the domain Γ are no more justified. From Table 2 and Table 3 can be observed that the maximum differences between the potential for the same domain Γ in Ω_1 (circular cross-section) and Ω_2 (elliptical cross-section) are not equal. The differences are larger in the case of Ω_2, since for the same domain Γ its rectangular cross-section will be nearer to the boundary of the elliptical cross-section of Ω_2 than to the boundary of the circular shaped cross-section Ω_1.

In addition the figures of the potential distribution calculated in the described domains Ω_1 and Ω_2 are shown for non-equidistant grid. It is noticeable that the maximum potential value for the simulation in the circular domain Ω_1 is slightly larger than the one simulated in the domain Ω_2, there is also certain difference in the distribution to be recognize on Figures 21 and 22. The plots of the relative difference between the potential simulated in one Ω domain (Ω_1 or Ω_2) and the potential simulated in the corresponding Γ domain are shown in Figures 23 to 37.

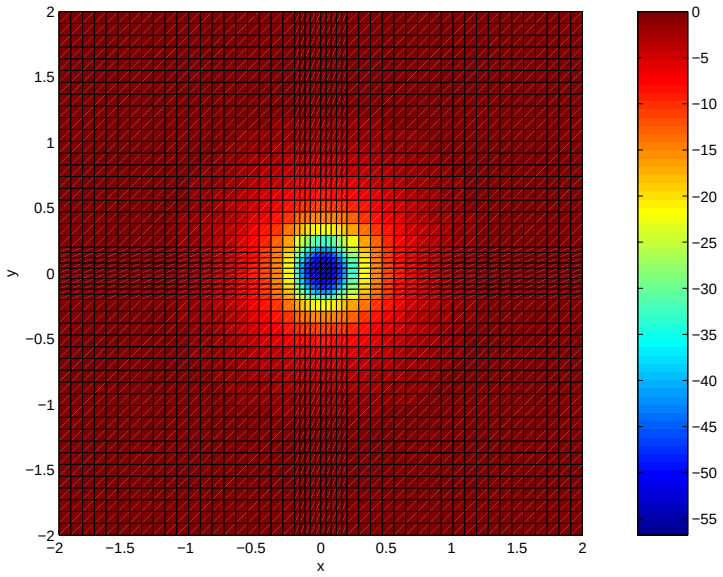

Figure 21: Potential distribution in Ω_1, (x, y)-plane cross-section at $z = 0$.

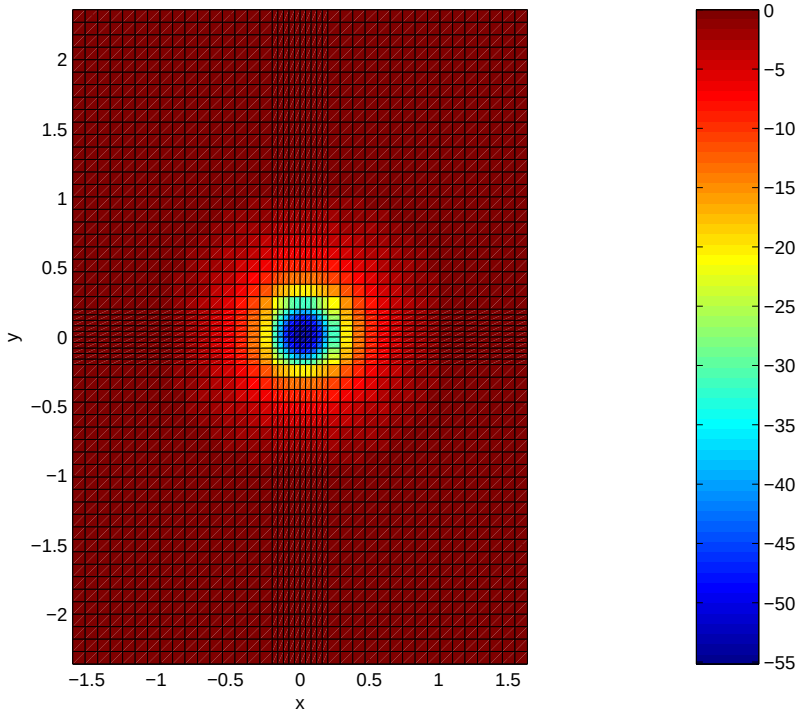

Figure 22: Potential distribution in Ω_2, (x, y)-plane cross-section at $z = 0$.

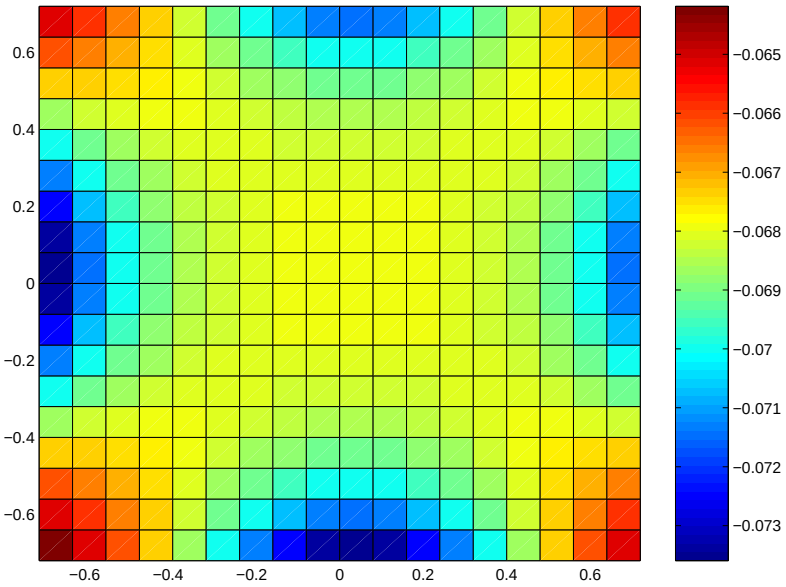

Figure 23: Relative difference $a = 0.72$ $N_x = N_y = 19$, $N_z = 41$, equidistant grid, (x, y)-plane cross-section at $z = 0$.

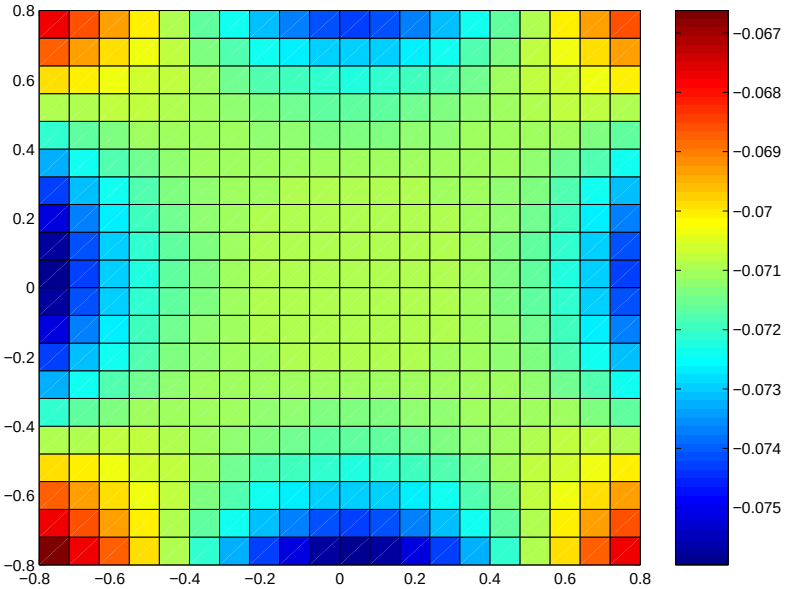

Figure 24: Relative difference $a = 0.8$ $N_x = N_y = 21$, $N_z = 41$, equidistant grid, (x, y)-plane cross-section at $z = 0$.

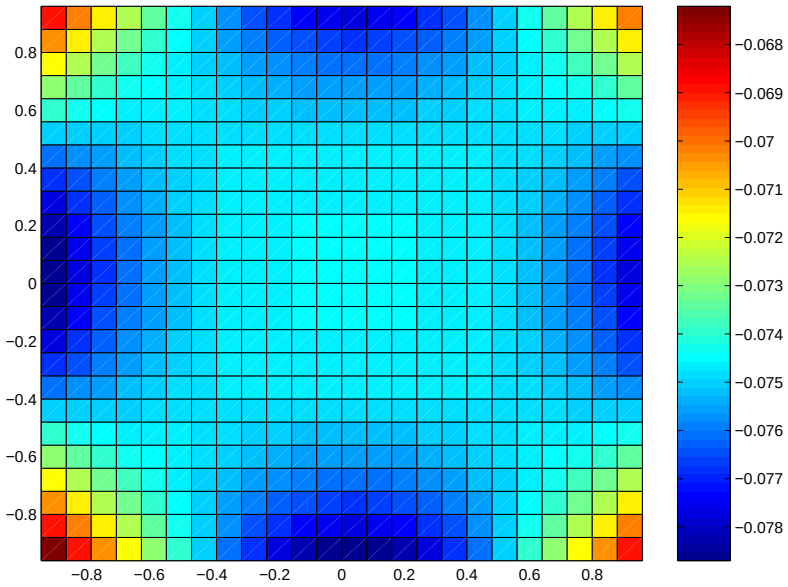

Figure 25: Relative difference $a = 0.96$ $N_x = N_y = 25$, $N_z = 41$, equidistant grid, (x, y)-plane cross-section at $z = 0$.

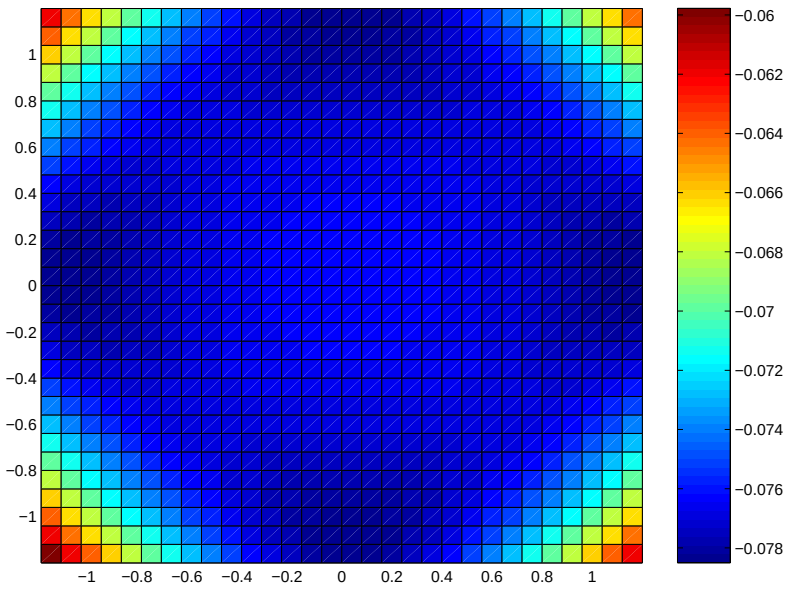

Figure 26: Relative difference $a = 1.2$ $N_x = N_y = 31$, $N_z = 41$, equidistant grid, (x, y)-plane cross-section at $z = 0$.

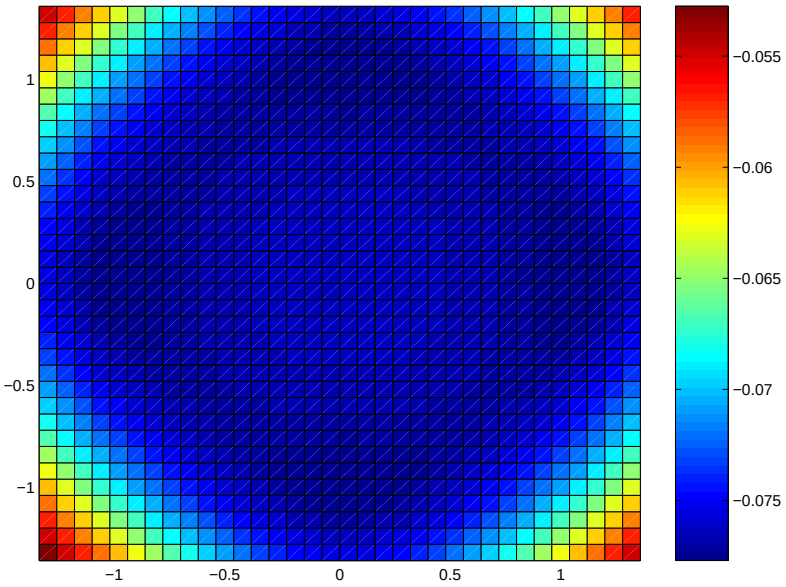

Figure 27: Relative difference $a = 1.36$ $N_x = N_y = 35$, $N_z = 41$, equidistant grid, (x, y)-plane cross-section at $z = 0$.

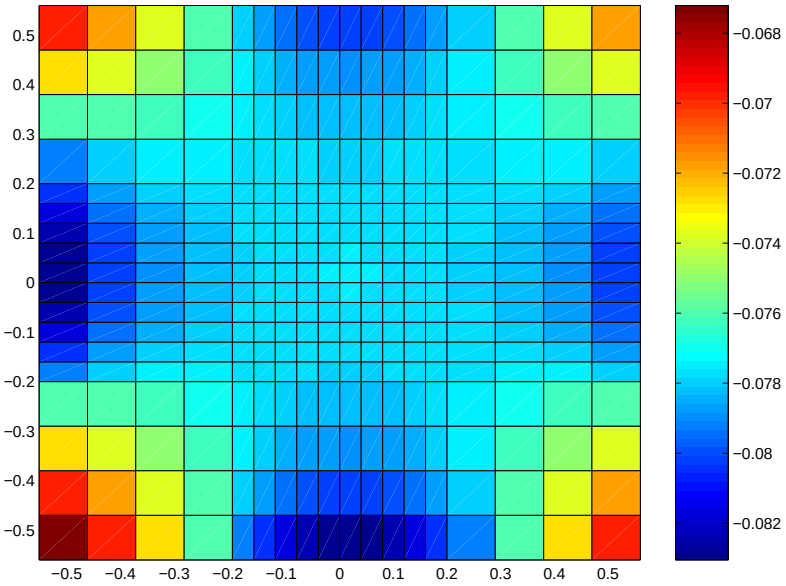

Figure 28: Relative difference $a = 0.56$ $N_x = N_y = 19$, $N_z = 41$, non-equidistant grid, (x, y)-plane cross-section at $z = 0$.

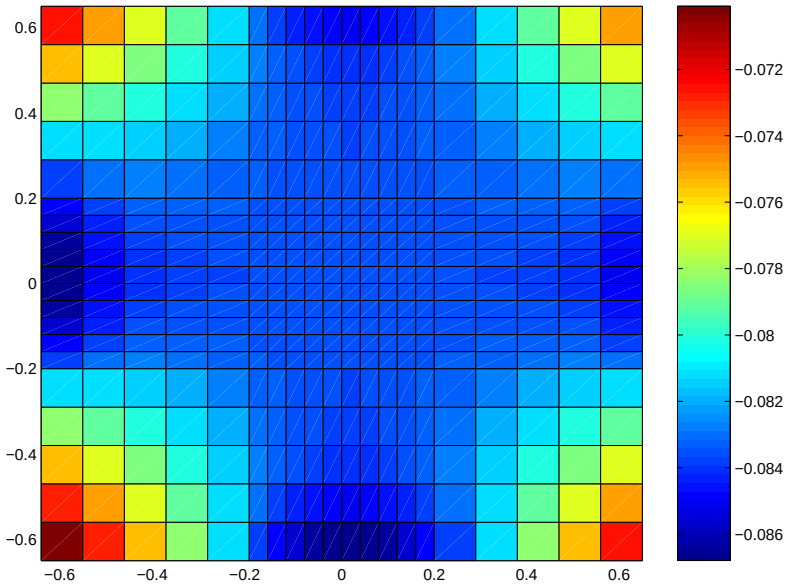

Figure 29: Relative difference $a = 0.65$ $N_x = N_y = 21$, $N_z = 41$, non-equidistant grid, (x, y)-plane cross-section at $z = 0$.

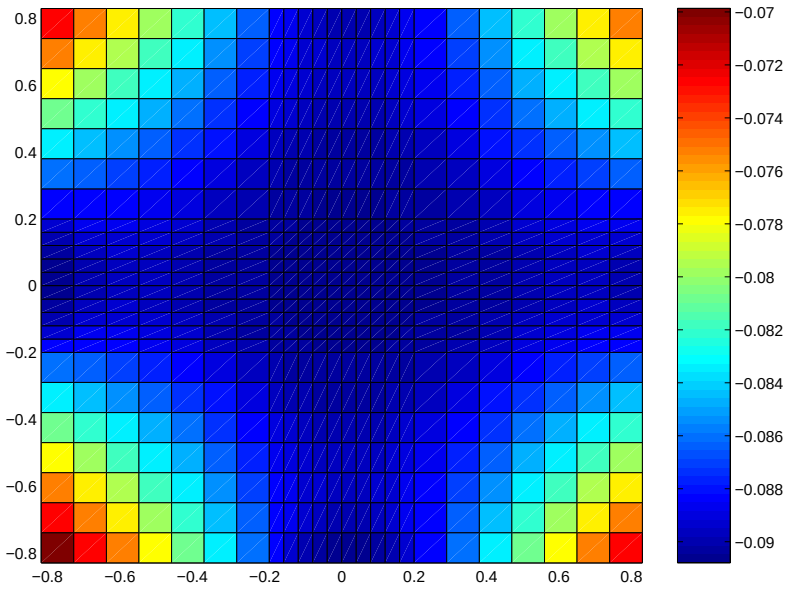

Figure 30: Relative difference $a = 0.83$ $N_x = N_y = 25$, $N_z = 41$, non-equidistant grid, (x, y)-plane cross-section at $z = 0$.

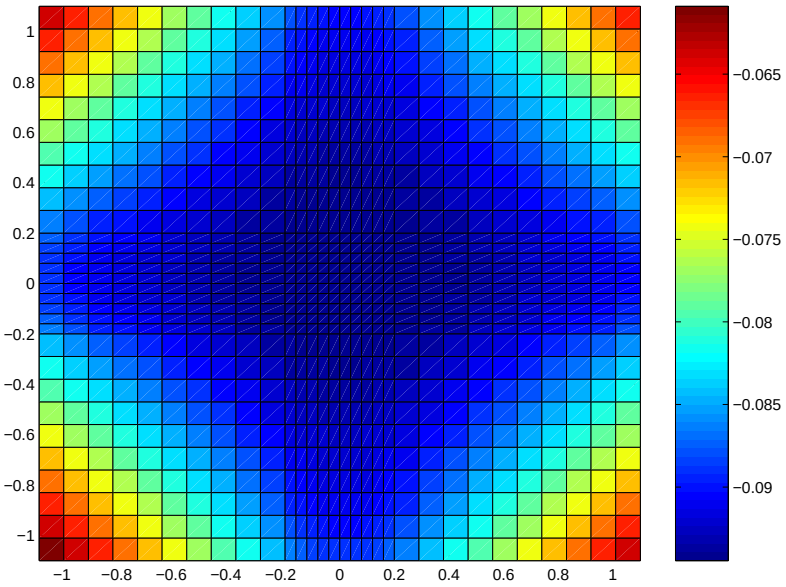

Figure 31: Relative difference $a = 1.1$ $N_x = N_y = 31$, $N_z = 41$, non-equidistant grid, (x, y)-plane cross-section at $z = 0$.

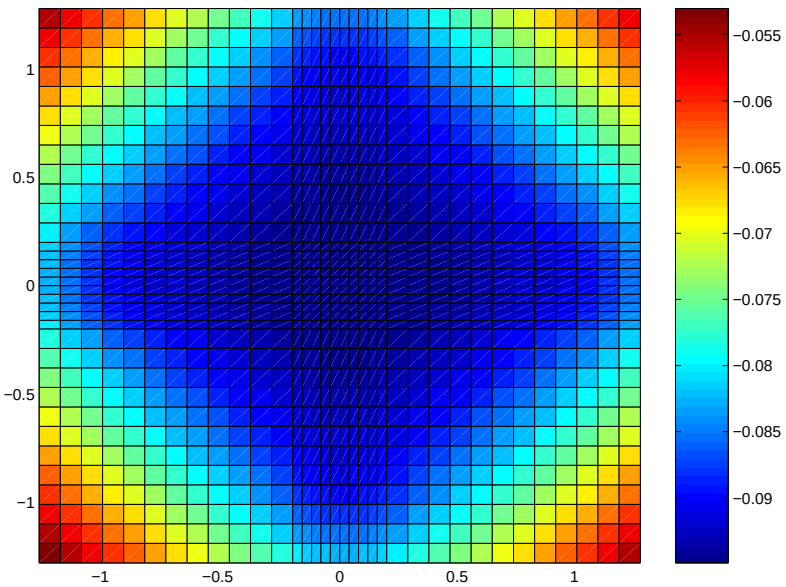

Figure 32: Relative difference $a = 1.28$ $N_x = N_y = 35$, $N_z = 41$, non-equidistant grid, (x, y)-plane cross-section at $z = 0$.

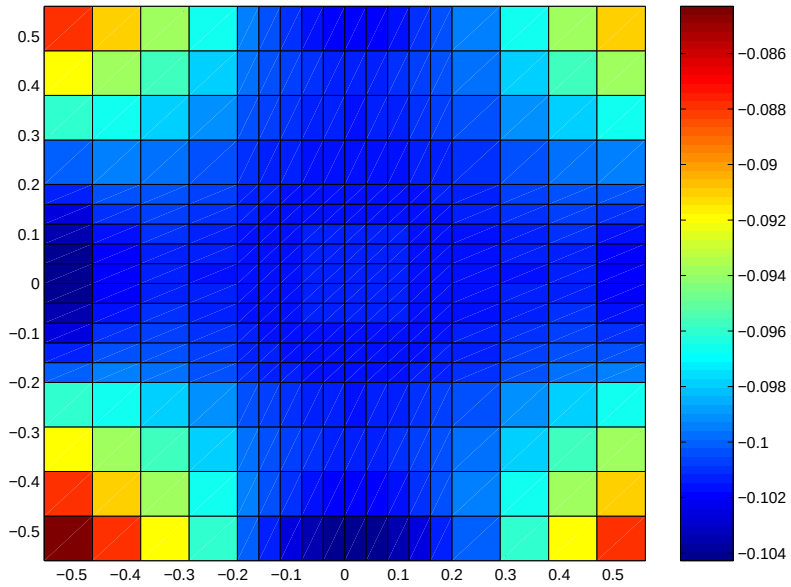

Figure 33: Relative difference $a = 0.56$ $N_x = N_y = 19$, $N_z = 41$, non-equidistant grid, (x, y)-plane cross-section at $z = 0$, elliptical domain Ω_2.

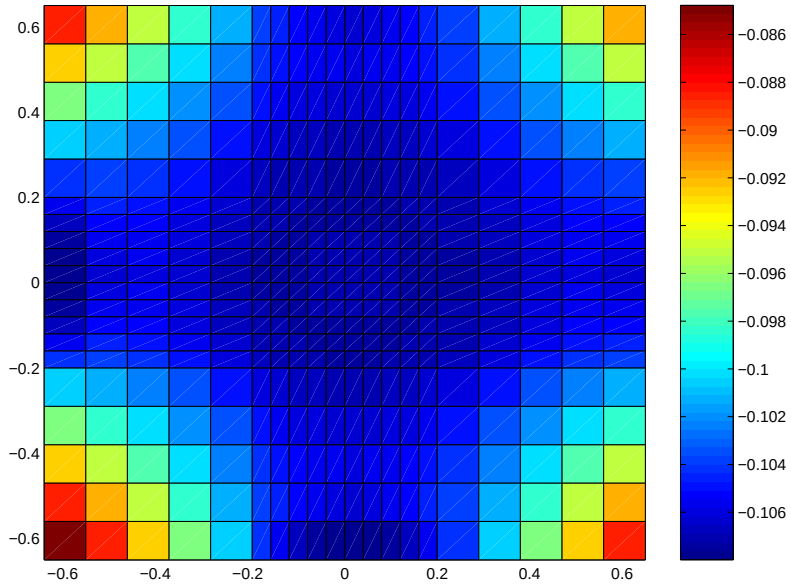

Figure 34: Relative difference $a = 0.65$ $N_x = N_y = 19$, $N_z = 41$, non-equidistant grid, (x, y)-plane cross-section at $z = 0$, elliptical domain Ω_2.

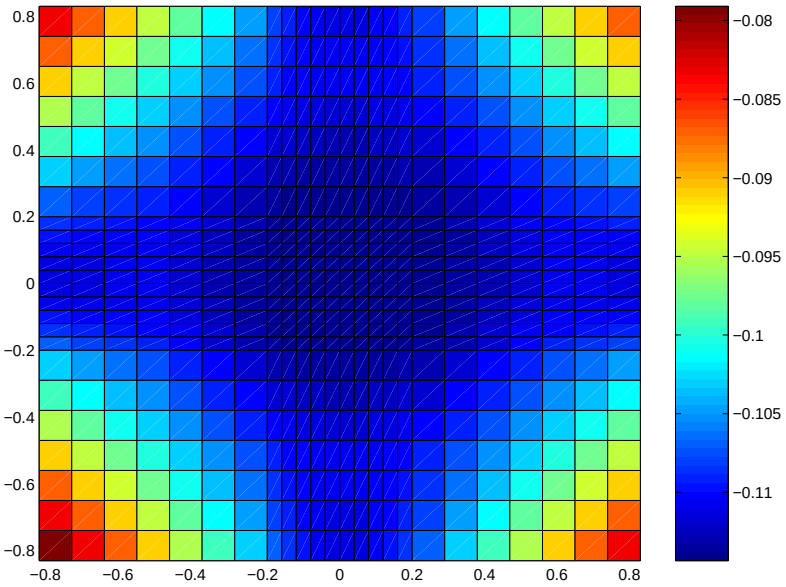

Figure 35: Relative difference $a = 0.83$ $N_x = N_y = 25$, $N_z = 41$, non-equidistant grid, (x, y)-plane cross-section at $z = 0$, elliptical domain Ω_2.

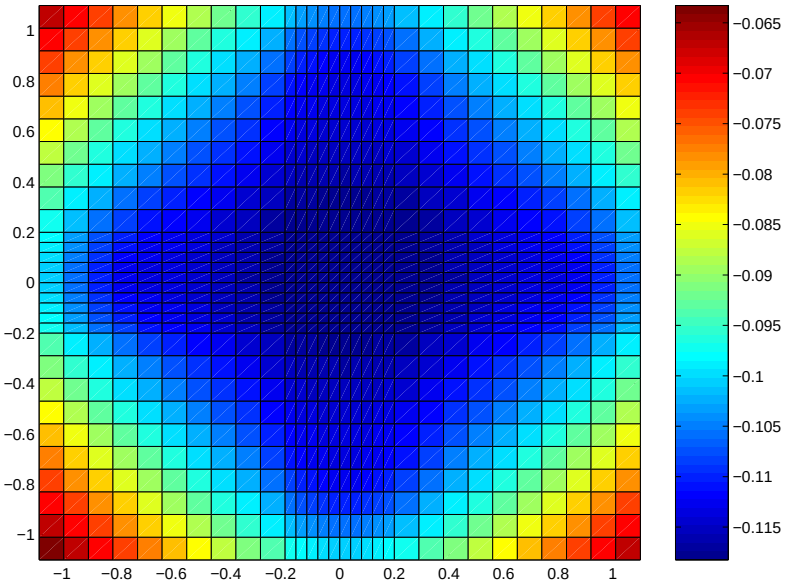

Figure 36: Relative difference $a = 1.1$ $N_x = N_y = 31$, $N_z = 41$, non-equidistant grid, (x, y)-plane cross-section at $z = 0$, elliptical domain Ω_2.

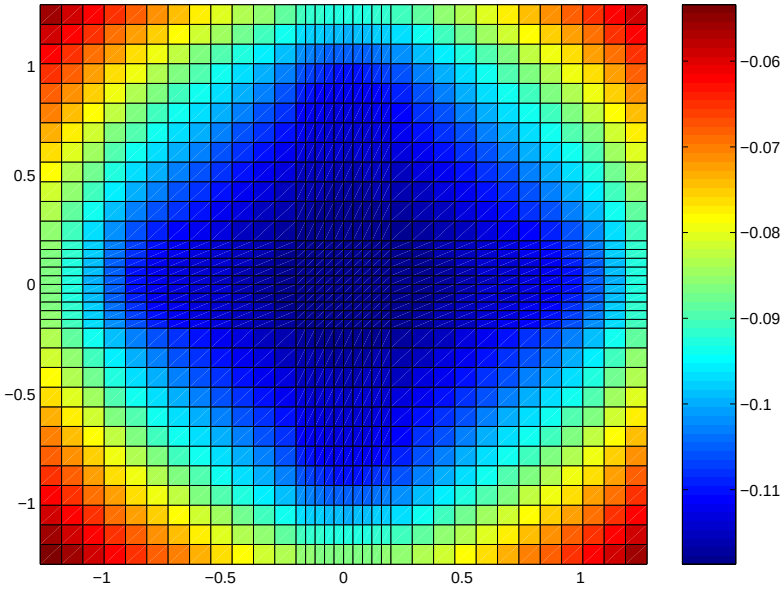

Figure 37: Relative difference $a = 1.28$ $N_x = N_y = 35$, $N_z = 41$, non-equidistant grid, (x, y)-plane cross-section at $z = 0$, elliptical domain Ω_2.

6 Conclusions

The computation of space-charge forces of bunches of charged particles is an important part of the simulation of the particle dynamics. Most of the particle tracking software packages calculates the 3D space-charge fields of bunches in a rectangular domain. In this work the space-charge fields are calculated in different 3D elliptical cross-section domains that are corresponding to the real boundaries of the beam pipe. The results (from section 5) of different simulations made in the elliptical domains Ω_1 and Ω_2 compared with the results from the simulations of the same bunch model in a rectangular domain with open boundary condition (Γ) show that neglecting the influence of the elliptic boundaries and solving the Poisson equation in a rectangular domain with open boundaries is error prone. The difference between the potential simulated in an elliptical cross-section domain Ω and the potential simulated in the corresponding Γ domain rises as the domain Γ gets larger.

The implemented BiCGSTAB algorithm for solving the non-symmetrical linear system of equations resulting from the discretization in the elliptical cross-section domain performs in relative short time. That makes these routines suitable for implementation in tracking programs where it is necessary to perform consecutive space-charge fields calculations.

Prospective work in the direction simulation of space-charge fields in elliptical cross-section domains would be to broaden the same algorithm for elliptical domains with variable cross-section along the z-axis. This would allow us to simulate the space-charge fields in arbitrary cavity shapes. Applying the multigrid method to solve the resulting linear system of equations from the discretization of

the elliptical cross-section domain is another further intention, which would allow higher aspect ratio discretizations and it would generalize the software package ”MOEVE” further as a multigrid solver of the Poisson equation in arbitrary elliptical cross-section domains.

References

[1] W. Chou and J.M. Jowett. Appendix 1. *ICFA Beam Dynamics Newsletter No.31*, August 2003.

[2] Tesla Colaboration Community. Superconducting TESLA cavities. *PHYSICAL REVIEW SPECIAL TOPICS - ACCELERATORS AND BEAMS, 092001 (2000)*, VOLUME 3:25, 2000.

[3] G. Dugan. *USPAS Jan 2002 Accelerator School, Phys 450B: Introduction to Accelerator Physics*. `http://www.lns.cornell.edu/~dugan/USPAS`, New York, 2002.

[4] K. Flöttmann. *ASTRA*. DESY, Hamburg, `www.desy.de/~mpyflo`, 2000.

[5] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA 19104 USA, 1997.

[6] R.W. Hockney and J.W. Eastwood. *Computer Simulation Using Particles*. Institut of Physics Publishing, Bristol, 1992.

[7] C. R. Johnson. Positive definite matrices. *Amer. Math. Monthly*, 77(259-264), 1970.

[8] Ch. Van Loan. *Computational Frameworks for the Fast Fourier Transform*, volume 10 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, 1992.

[9] G. Maeß. *Vorlesungen ber numerische Mathematik I*. Akademie-Verlag Berlin, 1984.

[10] G. Pöplau. *MOEVE: Multigrid Poisson Solver for Non-Equidistant Tensor Product Meshes*. Universität Rostock, 2003.

[11] G. Pöplau, U. van Rienen, S.B. van der Geer, and M.J. de Loos. Multigrid algorithms for the fast calculation of space-charge effects in accelerator design. *TESLA Report 2003-31*, 2003.

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

[13] Pulsar Physics, De Bongerd 23, 3762 XA Soest, The Netherlands, `www.pulsar.nl/gpt`. *General Particle Tracer (GPT)*, release 2.70 edition.

[14] M. Reiser. *Theory and Design of Charged Particle Beams*. Wiley, New York, 1994.

[15] U. van Rienen. *Numerical Methods in Computational Electrodynamics. Linear Systems in Practical Application*, volume 12 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin Heidelberg, 2001.

[16] T. Weiland. Eine Methode zur Lösung der Maxwellschen Gleichungen für sechskomponentige Felder auf disketer Basis. *AEÜ*, 31:116–120, 1977.

[17] E.W. Weisstein. *BiconjugateGradientMethod*. MathWorld-A Wolfram Web Resource `http://mathworld.wolfram.com/BiconjugateGradientMethod.html`.