UNIVETZITET U SARAJEVU

PRIRODNO – MATEMATIČKI FAKULTET

ODSJEK ZA FIZIKU

II CIKLUS STUDIJA – FIZIKA – MEDICINSKA RADIJACIONA FIZIKA

SIMULATION OF LOW ENERGY BEAM TRANSPORT

ZAVRŠNI RAD II CIKLUSA STUDIJA

Mentori:

Dr. Mariusz Sapinski

Prof. dr. Adnan Beganović

Kandidat:

Benjamin Dedić

Sarajevo, oktobar 2021.

I present gratitude to my mentors Dr. Mariusz Sapinski and prof. dr. Adnan Beganović for support and assistance in choosing the topic and writing this master thesis.

I also thank my family and friends who were supportive during the process of writing the paper.

Table of contents

1 Introduction

1.1 A brief introduction

High-energy physics research has always been the driving force behind the development of particle accelerators. They started life in physics research laboratories in glass envelopes sealed with varnish and putty with shining electrodes and frequent discharges, but they have long since outgrown this environment to become large-scale facilities offering services to large communities. Although the particle physics community is the main group a lot more groups have joined in as the interest for high energy as well as the low energy particles is one of the fastest growing branches of physics. Furthermore, there is a branch of physics dedicated to the accelerators called accelerator physics.(1)

1.2 History of accelerators

The History of accelerators can be divided into three different roots that are based on the idea for different acceleration mechanisms.(2)

1.2.1 The main timeline

The first root to be described is generally taken as the principal "history line", since it was the logical consequence of the vigorous physics research program in progress at the turn of the century. Particle physics research was the forefront behind the accelerator development and high-energy physics can be presented as the birthplace. In the table (Table 1) are presented the main events from the end of 19^{th} century to the early 20^{th} century which show the progression through atomic physics to nuclear physics.(3)

Table 1: list of main events from the 19th century to the early 20th century representing the progression of atomic physics to nuclear physics (2)

YEAR	EVENT
1895	Lenard. Electron scattering on gases (Nobel Prize) – discovered electrons of energies lower than 100 keV and the first electrostatic generator – Wimschurst machine
1913	Franck and Hertz excited electron shells by electron bombardment.
1906	Rutherford bombards mica sheet with natural alphas and develops the theory of atomic scattering – reached the energies of several MeV with alpha particles
1911	Rutherford publishes theory of atomic structure.
1919	Rutherford induces a nuclear reaction with natural alphas. Rutherford believes he needs a source of many MeV to continue research on the nucleus. This is far beyond the electrostatic machines then existing.
1928	Gamov predicts tunnelling and perhaps 500 keV would suffice.
1928	Cockcroft & Walton start designing an 800 kV generator encouraged by Rutherford.
1932	Generator reaches 700 kV and Cockcroft & Walton split lithium atom with only 400 keV protons. They received the Nobel Prize in 1951.

At about the same time as Cockroft and Walton, Van de Graaff, an American who was in Oxford as a Rhodes scholar, invented an electrostatic generator for nuclear physics research and later in Princeton, he built his first machine, which reached a potential of 1.5 MV. It took some time to develop the acceleration tube and this type of machine was not used for physics research until well after the atom had been split in 1932. Two new features appeared in later versions of the Van de Graaff generator. Firstly, the sparking threshold was raised by putting the electrode system and accelerating tube in a high-pressure tank containing dry nitrogen, or Freon, at 9-10 atmospheres, which enables operation typically up to 10 MV. The second was a later development, which has the special name of the Tandem accelerator.(2)

1.2.2 Second timeline

The direct-voltage accelerators were the first to be exploited for nuclear physics research, but they were limited to the maximum voltage that could be generated in the system (except for the astute double use of the applied voltage in the Tandem). This limitation was too restrictive for the requirements of high-energy physics. An alternative had been proposed in 1924 in Sweden by Ising (4). He planned to repeatedly apply the same voltage to the particle using alternating fields and his invention was to become the underlying principle of all of today's ultra-high energy accelerators This is known as resonant acceleration.

In the table (Table 2) the second timeline is presented.

Table 2: chronology events in the second timeline of the history of accelerators(2)

YEAR	EVENT
1924	Ising proposes time-varying fields across drift tubes. This is "resonant acceleration", which can achieve energies above that given by the highest voltage in the system.
1928	Wideröe demonstrates Ising's principle with a 1 MHz, 25 kV oscillator to make 50 keV potassium ions.
1929	Lawrence, inspired by Wideröe and Ising, conceives the cyclotron.
1931	Livingston demonstrates the cyclotron by accelerating hydrogen ions to 80 keV.
1932	Lawrence's cyclotron produces 1.25 MeV protons and he also splits the atom just a few weeks after Cockcroft and Walton (Lawrence received the Nobel Prize in 1939).

The difference between the acceleration mechanisms of Cockcroft and Walton and Ising depend upon whether the fields are static (conservative) or time-varying (nonconservative). The electric field can be expressed in a very general form as the sum of two terms, the first being derived from a scalar potential and the second from a vector potential,

$$\vec{E} = -\nabla\phi - \frac{\delta}{\delta t}\vec{A} \quad (1)$$

where $\vec{B} = \nabla \times \vec{A}$. (2)(5)

1.2.3 Third timeline

In the contrast to the first two timelines the third one is fainter and is based on the discoveries by the first two. The third timeline starts with Wideröe (6,7), a Norvergian student, who suggested a different acceleration mechanism now known as “betatron acceleration”. This device, the betatron, is insensitive to relativistic effects and was therefore ideal for accelerating electrons. The betatron has also the great advantages of being robust and simple. The one active element is the power converter that drives the large inductive load of the main magnet. The focusing and synchronization of the beam energy with the field level are both determined by the geometry of the main magnet.

The third timeline can be tracked from the 1923. up until the middle of 20th century and the events are described in the table (Table 3).

Table 3: events which are considered to be significant during accelerator development (2)

YEAR	EVENT
1923	Wideröe, a young Norwegian student, draws in his laboratory notebook the design of the betatron with the well-known 2-to-1 rule. Two years later he adds the condition for radial stability but does not publish.
1927	Later in Aachen Wideröe makes a model betatron, but it does not work. Discouraged he changes course and builds the linear accelerator mentioned in (previous table second timeline cross refference).
1940	Kerst re-invents the betatron and builds the first working machine for 2.2 MeV electrons.
1950	Kerst builds the world's largest betatron of 300 MeV.

The development of betatrons for high-energy physics was short, ending in 1950 when Kerst built the world's largest betatron (300 MeV), but they continued to be built commercially for hospitals and small laboratories where they were considered as reliable and cheap. In fact, the betatron acceleration mechanism is still of prime importance. In the present-day synchrotron, there is a small contribution to the beam's acceleration which arises from the increasing field in the main dipoles. If an accurate description of the longitudinal motion is required, then the betatron effect has to be included.

1.2.4 Development after the middle of the 20th century

By the 1940's three acceleration mechanisms had been demonstrated: DC acceleration, resonant acceleration and the betatron mechanism. In fact, there were to be no new ideas for acceleration mechanisms until the mid-1960's, when collective acceleration was proposed in which heavy ions are accelerated in the potential well of an electron ring and the 1980's when there were several Workshops devoted entirely to finding new acceleration techniques. (8)

Multiple accelerators were designed during this brief period of human history and few of them are listed in the continuation.

The microtron, sometimes known as the electron cyclotron, was an ingenious idea due to Veksler. The electrons follow circular orbits of increasing radius, but with a common tangent. An RF[1] cavity positioned at the point of the common tangent supplies a constant energy increment on each passage. The relativistic mass increase slows the revolution frequency of the electrons, but by a constant increment on each passage. If this increment is a multiple of the RF oscillator frequency, the electrons stay in phase, but on a different orbit. Microtrons operate at microwave frequencies and are limited to tens of MeV. They are available commercially and are sometimes used as an injector to a larger machine.

[1] RF – radio frequent

The radio-frequency quadrupole (RFQ) suggested in 1970 by I. Kapchinski and V. Telyakov is useful at low energies and is increasingly replacing the Cockcroft-Walton as injector. The RFQ combines focusing and acceleration in the same RF field. (2)

The linear accelerator was eclipsed during the thirties by circular machines. However, the advances in ultra-high frequency technology during World War II (radar) opened up new possibilities and renewed interest in linac structures. Berkeley was first, with a proton linear accelerator of 32 MeV built by Alvarez in 1946. The Alvarez accelerator has become very popular as an injector for large proton and heavy-ion synchrotrons all over the world with energies in the range of 50–200 MeV, that is essentially non-relativistic particles. The largest proton linear accelerator to date is the 800 MeV 'pion factory' (LAMPF) at Los Alamos. The first electron linear accelerators were studied at Stanford and at the Massachusetts Institute for Technology (MIT) in 1946. This type of accelerator has also had a spectacular development, up to the largest now in operation, the 50 GeV linear accelerator at the Stanford Linear Accelerator Centre (SLAC). Like betatrons they have become very popular in fields outside nuclear physics, particularly for medicine. (9)

1.3 Theoretical introduction

The purpose of this chapter is to provide the reader with the relevant concepts and equations needed to understand the results in this thesis work. In the following chapter starts with brief electrodynamics introduction and continuous with low-energy transport, beam characteristics and beam formation.

1.3.1 Introduction into electrodynamics

In principle, the task of beam extraction and the following low-energy beam transport (LEBT) system are quite simple. The ion source extraction consists of the plasma electrode (the front plate of the ion source), and at least one other electrode (the puller or extractor electrode), which provides the electric field for accelerating the charged particles from the ion source to form an ion beam. Whether or not the extraction contains any other electrodes, the beam leaves the extraction at energy

$$E = q(V_{\text{source}} - V_{\text{beamline}}) \tag{2}$$

defined by the charge q of the particles and the potential difference between the ion source, V_{source}, and the following beamline, V_{beamline}. The intensity of the particle beam depends, as a first approximation, on the flux of charged particles hitting the plasma electrode aperture. The extracted ion beam current is calculated as:

$$I = \frac{1}{4} A q n \bar{v} \tag{3}$$

where A is the plasma electrode aperture, q is the charge of the particles, n is the ion density in the plasma and $\underline{v}$ is the mean velocity of extracted particles in the ion source plasma. Assuming a Maxwell– Boltzmann distribution for the extracted plasma particles, the mean velocity

$$\bar{v} = \sqrt{\frac{8kT}{\pi m}}. \tag{4}$$

From the point of view of the extraction, the plasma electrode aperture can be adjusted to tune the beam intensity. The practical solutions are unfortunately much more complicated in most cases. The applications following the LEBT, which typically are accelerators to bring the beam to higher energies, often pose strict requirements for the ion beam parameters. (10)

Without careful design of the focusing elements, the space-charge force of the beam blows up the beam to the walls of the vacuum chamber, and only a part of the generated beam gets transported to the following accelerator. The extraction focusing systems must also provide some adjustability because, in most cases, the plasma conditions might not be constant in day-to-day operations. (11)

1.3.2 Low energy beam transport

The ion beam travels in the beam transport line from one ion optical element to another along a curved path, which is usually defined as the longitudinal direction z. The transverse directions x and y are defined relative to the center of the transport line, the optical axis, where $x = 0$ and $y = 0$. The transport line is usually designed in such a way that a so-called reference particle travels along the optical axis with nominal design parameters. The ion beam is an ensemble of charged particles around the reference particle, with each individual particle at any given time described by spatial coordinates (x, y, z) and momentum coordinates (p_x, p_y, p_z). This six-dimensional[2] space is known as the particle phase space. In addition to these coordinates, often inclination angles α and β or the corresponding tangents x_0 and y_0 are used. These are defined by $x' = \tan\ \alpha\ = \frac{p_x}{p_z}$ and $y' = \tan\ \beta\ = \frac{p_y}{p_z}$.

The motion of a charged particle in electromagnetic fields E and B is described by the Lorentz force F and Newton's second law, giving

$$\frac{\mathrm{d}\vec{p}}{\mathrm{d}t} = F = q(\vec{E} + \vec{v} \times \vec{B}) \tag{5}$$

where p is the momentum and v is the velocity of the particle with charge q. In general, the trajectory of a charged particle can be calculated by integrating the equation of motion if the fields are known. In the case of beam transport, the fields have two origins:

i) External field – generated by ion optical elements;
ii) Beam generated fields (11)

1.3.2.1 Beamline elements

The ion optical elements of the beam transport line come in two varieties: magnetic and electric. In the case of high-energy beams, where $v \approx c$, magnetic elements are used because the force, which is created with an easily produced magnetic field of 1 T, equals the force from an electric field of 300 $\frac{\mathrm{MV}}{\mathrm{m}}$, which is impossible to produce in a practical device. (11,12)

[2] The six dimensional space is the general case while in practice 2D and 4D phase spaces can be used depending on the problem setup.

Since there is no electric field in the LEBT and the achievable forces are comparable, the other factors such as size, cost, power consumption and the effects of beam space-charge compensation are the ones that play a bigger role. An important factor in the selection of the type of beamline elements is also the fact that electrostatic fields do not separate ion species. In electrostatic systems the particles follow trajectories defined only by the system voltages. The magnetic elements, on the other hand, have a dependence on mass-to-charge ratio m/q. This allows separation of different particle species from each other. The common beamline elements that are used to build LEBT systems include immersion lens, einzel lens, solenoid, dipole and quadrupole lenses. (13,14)

1.3.2.1.1 Immersion lens

The immersion lens (or gap lens) is simply a system of two electrodes with a potential difference of $\Delta V = V2 - V1$. The lens can be either accelerating or decelerating and in addition to changing the particle energy by $q\Delta V$. This element can be used for focusing as well. The focal length of the immersion lens is given by (13)

$$\frac{f}{L} = \frac{4\left(\sqrt{\frac{V_1}{V_2}} + 1\right)}{\frac{V_1}{V_2} + \frac{V_2}{V_1} - 2} \tag{6}$$

where L is the distance between the electrodes. The electrostatic extraction systems always have gap lenses, which accelerate the beam to the required energy. The first acceleration gap (length between plasma and puller electrode) is a special case of the immersion lens because of the effect of the plasma on the electric field. (11)

1.3.2.1.2 Einzel lens

The einzel lens is made by combining two gap lenses into a system of three electrodes with first and last electrodes at the beamline potential V_0 and the center electrode at a different potential V_{einzel}. The einzel lens, which is in most cases cylindrically symmetric for round, is the main tool for beam focusing in many electrostatic extraction systems. The einzel focusing power is dependent on the geometry and the voltage ratio $R = \frac{V_{einzel} - V_0}{V_0}$, assuming that at the zero potential the kinetic energy is zero as well. The einzel lens may have the first gap accelerating and the second gap decelerating (known as accelerating einzel lens, $R > 0$) or vice versa (known as decelerating einzel lens, $R < 0$). Accelerating einzel lenses should be preferred if the required higher voltage (and electric fields) can be handled, because they have lower spherical aberrations than decelerating einzel lenses, especially when the required refractive power is high. A special case of the einzel lens, where the first electrode and the third electrode are at different potentials, is also possible. This type of setup is known as a three-aperture immersion lens or zoom lens. (11,13)

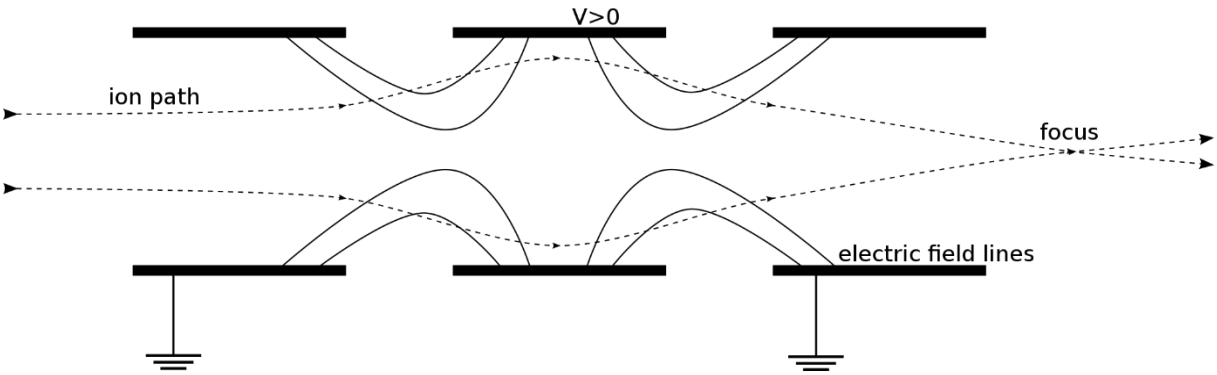

Figure 1: Implementation of an einzel lens showing the ion path. Six plates are parallel to the ion flight path with the middle plate at a particular potential.

1.3.2.1.3 Solenoid lens

A solenoid lens is the magnetic equivalent of the electrostatic einzel lens. It consists of rotationally symmetric coils wound around the beam tube, creating a longitudinal magnetic field with the maximum value at the center of the solenoid (15). The radial magnetic field at the entrance of the solenoid gives the particle entering the field with $v_r = 0$ at radius r_0 an azimuthal thrust

$$v_\theta = \frac{qBr_0}{2m} \tag{7}$$

which makes the trajectories helical inside the solenoid. At the exit of the solenoid, the particle receives a thrust cancelling the azimuthal velocity, but leaving the particle with a radial velocity

$$v_r = -\frac{r_0 q^2}{4m^2 v_z}\int B^2 \, \mathrm{d}z \tag{8}$$

This radial velocity causes the beam to converge towards the optical axis.(11) The refractive power of the lens is given by

$$\frac{1}{f} = \frac{q^2}{8mE}\int B^2 \, \mathrm{d}z \tag{9}$$

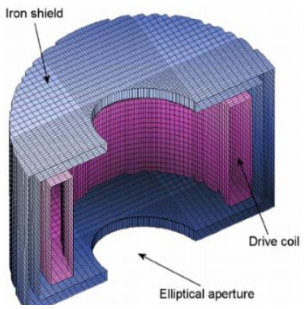

Figure 2: Cutaway view of a solenoid lens with elliptical apertures showing the conformal mesh structure for the finite- element field calculation. The elliptical axes are rotated so that the exit beam is parallel to the x - y axes of the simulation.

1.3.2.1.4 Quadrupole lenses

Electrostatic and magnetic quadrupoles are often used as focusing elements in LEBT systems in addition to einzel lenses and solenoids. The electrostatic quadrupole consists of four hyperbolic electrodes placed symmetrically around the beam axis with positive potential *Vquad* on the electrodes in the $+x$ and $-x$ directions and negative potential $-Vquad$ on the electrodes in the $+y$ and $-y$ directions. The potential is given by $V = \frac{x^2-y^2}{a^2} V_{quad}$ where a represents the radius of the quadrupole. Regarding to this the electrostatic field is given by:

$$\vec{E} = \frac{-2V_{quad}}{a^2} x\hat{x} + \frac{2V_{quad}}{a^2} y\hat{y} \tag{10}$$

while analyzing the formula (10), the conclusion is reached that the quadrupole focuses the beam in the x direction while it defocuses it in the y direction. Refractive power of these systems on x and y axis are (respectively)

$$\frac{1}{f_x} = k \sin\ (kL) \tag{11}$$

$$\frac{1}{f_y} = -k \sinh\ (kL) \tag{12}$$

where $k^2 = \frac{V_{quad}}{aV_0}$ and L represents the effective length of the quadrupole.

Magnetic quadrupole has almost the same construction as the electrostatic one where the magnet poles are made to be hyperbolic and windings are coiled in such a way that every other pole has magnetic flux in to the beam and every other out of the beam. The magnetic field in such a system is

$$\vec{B} = \frac{B_T}{a} x\hat{x} + \frac{B_T}{a} y\hat{y} \tag{13}$$

where B_T is the magnetic field density at the pole tip. Velocity of the positive particles is $\vec{v} = v_z\hat{z}$ and the force on them is $\vec{F} = \frac{qB_T v_z(-x\hat{x}+y\hat{y})}{a}$ which is again focusing on the *x* direction and defocusing on the *y* direction. The force leads to the same refractive power as in (equations for fx and fy) but the value of k is $k_B^2 = \frac{qB_T}{pa}$.

Quadrupole lenses are typically used as doublets or triplets for solutions that are focusing in both transverse directions. Quadrupoles can also be used for transforming asymmetric beams such as slit beams from a Penning ion source, for example, into a round beam. (11)

1.3.3 Beam emittance

Emittance is defined as the six-dimensional volume limited by a contour of constant particle density in the (x, p_x, y, p_y, z, p_z) phase space. This volume obeys the Liouville theorem[3] and is constant in conservative fields. With practical accelerators, a more important beam quality measure is the volume of the envelope surrounding the beam bunch. This is not conserved generally, only in the case where the forces acting on the particles are linear. Regarding the continuous (or long pulse) beams, where the longitudinal direction of the beam is of a lesser interest, transverse distributions (x, x_0) and (y, y_0) are used instead of the full phase-space distribution for simplicity. Also for these distributions the envelope surrounding the distribution changes when nonlinear forces[4] act on the particles. The size and shape of the transverse distribution envelope are important quality measures for beams because most complex ion optical devices such as accelerators have an acceptance window in the phase space within which they can operate. (11)

1.3.3.1 Emittance ellipse

For calculation and modelling purposes, a simple shape is needed to model the ion beam envelope in (x, x_0) phase space. Real well-behaved ion beams usually have Gaussian distributions in both x and x_0 directions. Because the contours of two-dimensional (2D) Gaussian distributions are ellipses, it is an obvious solution to use the ellipse as the model in 2D phase spaces (and ellipsoids in higher dimensions). The equation for an origin-centered ellipse is

$$\gamma x^2 + 2\alpha x x' + \beta x'^2 = \varepsilon \tag{14}$$

where the scaling

$$\beta\gamma - \alpha^2 = 1 \tag{15}$$

is chosen. The ε represents 2D transverse emittance while α, β, γ represent Twiss parameters defining the ellipse rotation and aspect ratio. Area of the ellipse can be calculated as

$$A = \pi\varepsilon = \pi R_1 R_2 \tag{16}$$

where R_1 and R_2 represent the length of the big and the small axis of the ellipse (the radii of the ellipse). Because of the connection between the area of the ellipse and ε, there is sometimes confusion about whether to include π in the above formula for quoted emittance values. The unit of emittance is mostly written as π mm mrad. This is used as an emphasis that the emittance number is in relation to the radii and not the area of the ellipse. (5,11)

[3] The Liouville equation describes the time evolution of the phase space distribution function.

[4] Non-idealities of beamline elements, for example

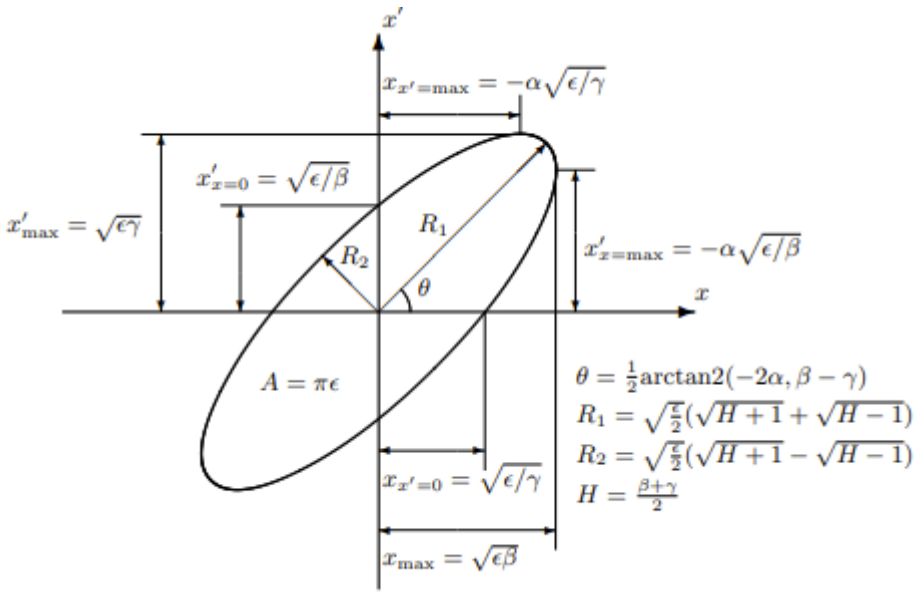

Figure 3: emittance ellipse geometry with the most important dimensions

1.3.3.2 R.m.s emittance

There are numerous ways to fit an ellipse to particle data. Often, a minimum-area ellipse containing some fraction of the beam is wanted (e.g. $\varepsilon_{90\%}$), but unfortunately this is difficult to produce in a robust. A well-defined way to produce the ellipse is by using a statistical definition known as the r.m.s. emittance,

$$\varepsilon_{r.m.s} = \sqrt{\langle x'^2 \rangle \langle x^2 \rangle - \langle xx' \rangle^2} \tag{17}$$

where the values for $\langle x'^2 \rangle$, $\langle x^2 \rangle$ and $\langle xx' \rangle$ can be calculated as:

$$\langle x'^2 \rangle = \frac{\iint x^2 I(x,x') dx dx'}{\iint I(x,x') dx dx'} \tag{18}$$

$$\langle x^2 \rangle = \frac{\iint x'^2 I(x,x') dx dx'}{\iint I(x,x') dx dx'} \tag{19}$$

$$\langle xx' \rangle = \frac{\iint xx' I(x,x') dx dx'}{\iint I(x,x') dx dx'} \tag{20}$$

I(x,x') describes the magnitude of the beam current at the differential area dxdx' of phase space at the point (x.x'). Twiss parameters can be calculated from these particle distributions as

$$\alpha = -\frac{\langle xx' \rangle}{\varepsilon} \tag{21}$$

$$\beta = \frac{\langle x^2 \rangle}{\varepsilon} \tag{22}$$

$$\gamma = \frac{\langle x'^2 \rangle}{\varepsilon} \tag{23}$$

As it is assumed that the emittance distribution at the origin is centered we can say the values for $\langle x' \rangle$ and $\langle x \rangle$ are 0. When measuring the emittance there are additional difficulties that are caused by background noise and amplifier offset in the I (x, x') data. (11,16)

1.3.3.3 Amount of beam inside the emittance ellipse

To better understand the r.m.s. emittance the amount of beam enclosed by the ellipse can be considered. This depends on the particle distribution shape. For real measured distributions, there is no direct rule. For theoretical known distributions, this can be calculated. The two most used model distributions used for beams are the Bi-Gaussian and the Kapchinskij–Vladimirskij (KV) distribution.(13) The Bi-Gaussian distribution orientated along the axis is given by:

$$\mathrm{I}(x, x') = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x{}^2} + \frac{x'^2}{\sigma_{x'}{}^2}\right)} \tag{24}$$

σ_x and $\sigma_{x'}$ are standard deviations of the distribution in the x and x' directions. In practice the distribution can be additionally rotated by an angle.

The KV distribution has a uniform beam density inside an elliptical phase space given by:

$$\mathrm{I}(x, x') = \mathrm{f}(x) = \{\frac{1}{\pi\varepsilon}, if\ \gamma x^2 + 2\alpha x x' + \beta x'^2 \leq \varepsilon\ 0, \qquad otherwise \tag{25}$$

1.3.3.4 Normalization of emittance

The transverse emittance defined in (x, x_0) space has the property that it is also dependent on the longitudinal beam velocity. If the beam is accelerated and p_z increases, then $x_0 = \frac{p_x}{p_z}$ decreases. (17) This effect is eliminated by normalizing the velocity to the speed of light c, which gives

$$x'_n = \frac{p_x}{p_{z1}}\frac{v_{z1}}{c} = \frac{v_x}{c} = \frac{p_x}{p_{z2}}\frac{v_{z2}}{c} \tag{26}$$

at non relativistic velocities. The normalized emittance can be calculated from regular emittance with:

$$\varepsilon_n = \varepsilon\frac{v_z}{c} \tag{27}$$

1.3.3.5 Emittance from plasma temperature

An ion beam formed by letting charged particles from a plasma be emitted from a round aperture has an emittance defined by the plasma ion temperature *T* and the aperture radius *r*, assuming that the acceleration to velocity v_z does not add aberrations. This minimum emittance can be calculated by using (27) (28) (29) and (30) and using a particle distribution defined by a circular extraction hole and Gaussian transverse ion distribution, i.e.

$$\mathrm{I}(x, x') = \frac{2}{\pi r^2}\sqrt{r^2 - x^2}\sqrt{\frac{m}{2\pi kT}}e^{-\frac{m(x'v_z)^2}{2kT}} \tag{28}$$

and after normalization the resulted r.m.s. emittance becomes

$$\varepsilon_{rms,n} = \frac{1}{2}\sqrt{\frac{kT}{m}}\frac{r}{c} \tag{29}$$

the calculation with a slit-beam extraction gives

$$\varepsilon_{rms,n} = \frac{1}{2}\sqrt{\frac{kT}{3m}}\frac{\omega}{c} \tag{30}$$

In the round aperture case, the emittance of the beam is linearly proportional to the plasma aperture radius as well as the beam current is roughly proportional to the area of the plasma aperture. Scaling of the aperture size does not therefore change the beam brightness and the brightness in the first approximation is given as $B = \frac{I}{\varepsilon_{n,x}\varepsilon_{n,y}}$. (9,11,18)

1.3.3.6 Emittance from solenoidal magnetic (B) field

In electron cyclotron resonance (ECR) and microwave ion sources, there is a strong solenoidal magnetic field at the plasma electrode location, where the beam formation happens and this has a strong influence on the beam quality. As the particles exit the solenoidal magnetic field, they receive an azimuthal thrust. The emittance of the beam can be calculated outside the solenoid by considering the particle coordinates far away, where the azimuthal particle motion has completely changed to radial motion and this is explained as:

$$r' = \frac{v_r}{v_z} = \frac{v_\theta}{v_z} = \frac{qBr_0}{2mv_z} \tag{31}$$

In that case the r.m.s. emittance of the beam can be calculated from this radius and the radius of the constant current density beam at the extraction. The r.m.s. emittance and the normalized emittance are then stated as (respectively):

$$\varepsilon_{rms} = \frac{1}{4}r'^{r_0} = \frac{qBr_0^2}{8mv_z} \Rightarrow \varepsilon_{rms,n} = \frac{qBr_0^2}{8mc} \tag{32}$$

In the case of ECR ion sources (ECRIS) the effects of the magnetic field dominate the emittance compared to the effect of the ion temperature. That is the result of high magnetic fields in the device itself. But in the case of practice, the formula given here is not able to predict the emittance values as the emittances of higher charge state ions have lower emittances than their counterparts (the lower charge state ions). This is what contradicts the formula (32) and the only solution is that the particles are not extracted from uniform plasma. This may also be considered that the higher charged particles are closer to the axis at the extraction aperture so their emittance is lower. (19)

1.3.4 Space charge

Ion beam charge density can be defined as:

$$\rho = \frac{J}{v} = \frac{I}{Av} \tag{33}$$

and it plays a major role in beam extraction systems where we can find high current densities and low velocities compared to other parts of the whole accelerator system. Space charge induces forces which lead to the increase of divergence and emittance. This effect lowers when the magnetic force that is generated by the beam particles compensates the “blow-up”, but the effect of the magnetic field is negligible at velocities much lower than the speed of light ($v \ll c$). (11,20)

1.3.4.1 Space charge effects on beam

If the assumption is that the beam is cylindrical with a constant current density with a radius of r and it propagates with constant velocity of v_z, then the electric field is given by the Gauss law and it is:

$$E = \{\frac{I}{2\pi\varepsilon_0 v_z}\frac{r}{r_{beam}^2}, \quad if\ r \le r_{beam}\ \frac{I}{2\pi\varepsilon_0 v_z}\frac{1}{r}, \ otherwise \tag{34}$$

the potential inside a beam tube is therefore:

$$\phi = \{\frac{I}{2\pi\varepsilon_0 v}\left[\frac{r^2}{2r_{beam}^2} + log\ log\ \left(\frac{r_{beam}}{r_{tube}}\right) - \frac{1}{2}\right], if\ r \le r_{beam}\ \ \frac{I}{2\pi\varepsilon_0 v}\ log\ log\left(\frac{r}{r_{tube}}\right),$$

$$otherwise \tag{35}$$

The equation for the constant current density electric field given by the equation (number here) is a linear value and it does not cause emittance growth but it does cause increase of the divergence of the beam. A particle at the beam boundary experiences the force:

$$F_r = qE_r = ma_r = \frac{qI}{2\pi\varepsilon_0 r v_z} \tag{36}$$

in that case the particle trajectory is calculated as

$$a_r = \frac{d^2r}{dt^2} = \frac{d^2r}{dz^2}\frac{d^2z}{dt^2} = v_z^2\frac{d^2r}{dz^2} \Rightarrow \frac{d^2r}{dz^2} = \frac{1}{v_z^2}a_r = \frac{qI}{2\pi\varepsilon_0 r v_z^3}a_r = K\frac{1}{r} \tag{37}$$

With the implementation of simple differential calculus and assuming the divergence is small, and that $\frac{\mathrm{d}r}{\mathrm{d}z} = 0$ at $z = 0$ the projectile trajectory is: (21)

$$\frac{\mathrm{d}r}{\mathrm{d}z} = \sqrt{2K\ \log\log\frac{r}{r_0}} \Rightarrow z = \frac{r_0}{\sqrt{2K}}\mathrm{f}\left(\frac{r}{r_0}\right)\ where\ \mathrm{f}\left(\frac{r}{r_0}\right) = \int_{y=1}^{\frac{r}{r_0}} \frac{\mathrm{d}y}{\sqrt{\log\ y}}. \tag{38}$$

Function f cannot be calculated analytically but can be estimated numerically as an estimation of divergence. E.G. a parallel zero-emittance beam of $^{181}Ta^{20+}$ accelerated with 60 kV has an initial radius of r_0 = 15 mm. The size of a 120 mA beam after a drift of 100 mm can be solved from $\mathrm{f}\left(\frac{r}{r_0}\right) = 1.189$, which gives r = 20 mm.

With practical drifting low-energy beams, a more realistic model for the beam distribution is Bi-Gaussian, for example. This kind of distribution leads to nonlinear space-charge forces, which cause emittance growth in addition to increase of beam divergence. Computer simulations are required to estimate these effects. (11)

1.3.4.2 Space charge compensation

The potential well of the beam formed by the accelerated charged particles acts as a trap for oppositely charged particles in areas where there are no external electric fields to drain the created charges. These particles compensate for the charge density of the beam, decreasing the depth of the potential well and therefore also decreasing the magnitude of the beam space-charge effects described in chapter 1.3.1. This process is called space charge compensation.

The most abundant process for the production of compensating particles is the ionization of the background gas within the beam. In the case of positive ion beams, the electrons produced in the background gas ionization are trapped in the beam, while slow positive ions are repelled to the beamline walls. In the case of negative ion beams, the compensating particles are the positive ions created in the gas. The creation rate of the compensating particles can be estimated with

$$\frac{\mathrm{d}n_c}{\mathrm{d}t} = \Phi n_{gas} \sigma_i \tag{39}$$

in which Φ represents the flux of particles in the beam, while n_{gas} represents the gas density and the σ_i ionization cross-section. If the creation rate is high enough, the space-charge compensation is finally limited by the leakage of compensating particles from the potential well as the compensation factor approaches 100%. The compensation factor achieved in a real system is difficult to estimate accurately because it depends on the lifetime of the compensating particles in the potential well. Most important processes affecting the lifetime are:

i. leakage of particles at the beamline ends, which can be limited with accelerating einzel lenses or magnetic fields
ii. recombinative processes
iii. scattering processes leading to ejection of particles from the potential well.

Assuming that the creation rate of compensating particles is high, the time-scale for achieving full compensation is

$$\tau = \frac{\rho_{beam}}{e\frac{\mathrm{d}n_c}{\mathrm{d}t}} = \frac{Q}{v n_{gas} \sigma_i} \tag{40}$$

Where *Q* stands for the charge state of the beam and v is the velocity of the beam. (22,23)

1.3.5 Beam formation

The assumption that was used in the previous chapters is that the ion beam is formed by accelerating the plasma particles and colliding them with the plasma electrode aperture which then for a product has the formation of the quasi neutral plasma and the un-neutralized beam. In the following chapter the physics of beam formation is analyzed in more detail. (9)

1.3.5.1 Space charge limited emission

In the first acceleration gap, where the beam is formed, the space-charge forces acting on the beam are largest. The situation can be evaluated in one dimension by assuming a beam starting with zero velocity with Poisson equation

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}z^2} = -\frac{\rho}{\varepsilon_0} = -\frac{J}{\varepsilon_0}\sqrt{\frac{m}{2q\phi}} \tag{41}$$

where z defines the location, ϕ is the gap potentials, J represents the beam current density and ε_0 is the permittivity of vacuum. The emission surface is at $\phi = 0$ $(at\ z = 0)$ and the extraction surface is at the potential $\phi = V$ $(at\ z = d)$. For *J*=0 potential distribution is linear. As the emission current density increases, the electric field at the emission surface decreases until it becomes zero. At that point the emission current is at the maximum level, for which Eq. (55) can be solved with the boundary condition $\frac{\mathrm{d}\phi}{\mathrm{d}z} = 0$ $(at\ z = 0)$. This condition is known as space charge limited emission, and the resulting limit for the maximum emission current density can be calculated using the following equation, which is known as the Child–Langmuir law (24):

$$J_{max} = \frac{4}{9}\varepsilon_0\sqrt{\frac{2q}{m}}\frac{V^{\frac{3}{2}}}{d^2} \tag{42}$$

The plasma ion sources are typically operated in emission-limited mode, i.e. the potential difference between the plasma electrode and puller electrode is made sufficiently large to handle the beam space charge. The law in the form shown here is not strictly valid for ion source plasma extraction because of the effects of plasma neutralization and higher starting velocity of particles in plasma extraction. The physics of the space-charge limit is still valid and the Child–Langmuir law can be used to estimate it. In any system, the maximum extractable current is dependent on the geometry, the emission current density and the voltage via the space-charge limit. In the space-charge-limited region, the current is proportional to $V^{\frac{3}{2}}$. This leads to the definition of the beam purveyance as:

$$P = \frac{I}{V^{\frac{3}{2}}} \tag{43}$$

which is the proportionality constant describing the system. As long as the emission is space-charge limited, the beam purveyance is roughly constant. When the voltage is further increased and the beam emission is no longer space-charge-limited, the beam purveyance decreases. (11)

1.3.5.2 Electrode geometry

The space-charge forces try to blow up the beam, as was shown above. This happens especially in the first acceleration gap because of the low velocity of the beam. To counteract the space-charge forces in the transverse direction, the electrodes can be shaped in such a way that the electric field in the first gap is not only accelerating but also focusing. In the case of space-charge limited surface emitted electrons, there is a perfect solution providing a parallel electron beam accelerated from the cathode. The solution is to have a field shaping electrode around the cathode (at cathode potential) in a 67.5 ◦ angle with respect to the emitting surface normal. This geometry is known as Pierce geometry. For plasma ion sources, there is no such magic geometry because the ions do not start from a fixed surface, but from plasma with varying starting conditions. (5,11,25)

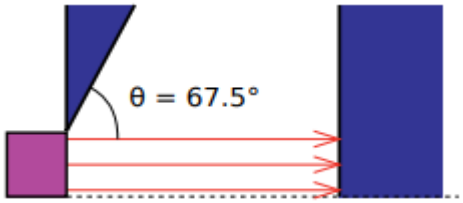

Figure 4: Perfectly parallel extraction of space-charge limited surface emission electrons using the Pierce geometry

1.3.5.3 Positive ion plasma extraction

In the case of ion plasma extraction, the beam formation is more complicated than in the case of surface emitted electrons described above. The ions are born in the quasi neutral plasma and get extracted into the un-neutralized beam. It is obvious that the extraction cannot be modelled without considering the neutralizing effect of the plasma. The simplest description of the necessary transition layer or plasma sheath was given by Bohm for an ion-electron plasma (26). The ions are assumed to arrive from the bulk plasma into the sheath with velocity v_0. The charge density of ions can be calculated by assuming a quasi-neutral situation $\rho_0 = \rho_i = \rho_e$ at the bulk plasma in the plasma potential $\phi = \phi_P$, where $\phi = \phi_{wall} = 0$ is the plasma electrode potential. Using ion continuity $\rho_0 v_0 = \rho_i v_i$ and energy conservation $\frac{m_i v_i^2}{2} = \frac{m_i v_0^2}{2} - q_i(\phi - \varphi_P)$, the ion density becomes

$$\rho_i = \rho_0 \sqrt{1 - \frac{q_i(\phi - \phi_P)}{m_i v_0^2}}. \tag{44}$$

The electrons are assumed to be in thermal equilibrium and therefore they follow the Boltzmann distribution:

$$\rho_e = \rho_0 e^{\frac{e(\phi - \varphi_P)}{kT_e}}. \tag{45}$$

As the potential is described by the Poisson equation the potential can be calculated through:

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = -\frac{\rho_0}{\varepsilon_0}\left[\sqrt{1 - \frac{q_i(\phi - \phi_P)}{m_i v_0^2}} - e^{\frac{e\phi}{kT_e}}\right]. \tag{46}$$

An important feature can be observed from the equation: the shielding condition $\frac{d\phi}{dx} = 0\ (at\ x = 0)$ is only fulfilled when the space charge is non-negative. The necessary condition is:

$$v_0 \geq v_B = \sqrt{\frac{kT_e}{m_i}} \tag{47}$$

is known as the Bohm sheath criterion and v_B as the Bohm velocity. The criterion sets a low velocity limit for ions arriving at the sheath edge and in most cases the equation holds with equality (27). The Poisson equation (46) is impossible to solve analytically, and often a numerical approach or approximations are used even in the presented one-dimensional case.

The situation at the plasma extraction is simple according to the model. Positive ions flow from quasi neutral bulk plasma into the extraction sheath with velocity v_B. The compensating electron density, which is defined by the potential, is equal to the ion density in bulk plasma and decays exponentially towards the extraction. Far enough in the extraction, the compensation becomes (essentially) zero. From the model, it is obvious that there is no well-defined boundary between neutralized plasma and the un-neutralized extraction. Often, such a boundary would be useful for judging the focusing action of the electric field close to the plasma electrode and for communicating about the plasma sheath shape. Therefore, an equipotential surface at ϕ_{wall} in the case of positive ion extraction is often chosen as an artificial 'boundary', known as the plasma meniscus. This choice works as a thought model even though in reality there is no such boundary.

The process of beam formation varies not only with the extraction electric field strength and shape, but also with the properties of the plasma, i.e. plasma density, electron and ion temperatures. (11,28)

1.3.5.4 Negative ion plasma extraction

The negative ion plasma extraction model is similar to the previously mentioned positive ion extraction model. The bulk plasma is at positive plasma potential ϕ_P and it is separated from the plasma electrode at $\phi = \phi_{wall} = 0\ V$ by a plasma sheath. Assumption is that the negative ions, which are either volume or surface produced, are born near the wall potential and then are extracted from a uniform plasma volume. These charges form a potential well and counteract the formation of a saddle point at the extraction aperture. The potential deviates from zero going into the bulk plasma due to the plasma potential and towards the extraction due to the acceleration voltage. This potential structure causes positive ions from the bulk plasma to be accelerated towards the extraction, having energy $e\phi_P$ at zero potential. These ions propagate until they are reflected back into the plasma by the increasing potential in the extraction. The potential well acts as a trap for thermal positive ions. The negative ions and electrons are accelerated from the wall potential towards the bulk plasma and more importantly towards the extraction (29).

The negative ion plasma sheath from the zero potential towards the extraction is described by the Poisson equation which is:

$$\Delta\phi = -\frac{\rho}{\varepsilon_0}; while\ \rho = \rho_{neg} + \rho_f + \rho_{th} \tag{48}$$

In the equation (48) ρ_{neg} represents the space charge density of negative particles, while ρ_f and ρ_{th} represent the space charge density of fast positive ions and trapped thermal positive ions respectively. This model allows several different negative ion species to be extracted from the source and many positive ion species that are used as compensating plasma particles. The thermal ions are prone to the Maxwellian velocity distribution where for each particle the distribution is:

$$\rho_{th} = \rho_{th,0} e^{-\frac{e\phi}{kT_i}}. \tag{49}$$

$\rho_{th,0}$ stands for the space charge density of thermal ion species at the wall potential while T_i is the corresponding thermal ion temperature. Fast ions are decelerated and turned back into plasma by extraction voltage. The space-charge distribution of the fast ions is defined by the virtual cathode formation and it is:

$$\rho_f = \rho_{f,0}\left(1 - \frac{e\phi}{E_i}\right)\ at\ e\phi < E_i. \tag{50}$$

$\rho_{f,0}$corresponds to the space charge density of the fast ions at the wall potential while E_i is the corresponding kinetic energy which should have the value around $e\phi_P$ as the particles are flowing with the bulk plasma. The condition for quasi neutrality of the plasma is that:

$$\rho_{neg} + \rho_f + \rho_{th} = 0\ at\ \phi = 0\ V \tag{51}$$

Plasma sheath acts similarly to the positive ion extraction in the negative ion extraction. The smallest beam emittance is achieved with an extraction field optimized for the plasma density of the ion source. The biggest difference from the positive ion case is that, with a positive puller electrode voltage, also electrons will be extracted from the plasma in addition to the negative ions. Depending on the ion source, the amount of co-extracted electrons may be as high as 100–200 times the amount of negative ions extracted or as low as 1 as is the case in caesiated surface production H^- sources. In the cases where the amount of electrons is high, the electrons need to be dumped in a controlled manner, as soon as possible, to avoid unnecessary emittance growth due to the additional space charge. Often the electron beam current is so high that the dumping cannot be done at the full beam energy required by the application. To be stated differently, the electron beam has to be dumped on an intermediate electrode at lower potential than ground. (30–33)

2 Materials and methods

In the following chapter of the paper materials that have been used as well as the methods used while designing the system as well as gathering the results will be discussed. The chapter will be based on having a short explanation of program languages used such as C++ and Python and for what they have been used, as well as the package IBSimu and the benefits it has. Once the explanation of the technical part has been finished, the chapter will base upon the issue of constructing geometries and providing all the iterations up until the final version of the code which is used for gathering and analyzing the results.

2.1 Short description of programing languages

2.1.1 C++

C++ is a general-purpose programming language created by Bjarne Stroustrup as an extension of the C programming language, or "C with Classes". The language has expanded significantly over time, and modern C++ now has object-oriented, generic, and functional features in addition to facilities for low-level memory manipulation. It is almost always implemented as a compiled language. As a programing language it was designed with an orientation toward system programming and embedded, resource-constrained software and large systems, with performance, efficiency, and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including desktop applications, video games, servers (e.g. e-commerce, web search, or databases), and performance-critical applications (e.g. telephone switches or space probes). (34)

2.1.2 Python

Python is an interpreted high level, general purpose programming language. Its language constructs as well as its object-oriented approach aim to help programmers write a clear and logical code. Python is dynamically-typed. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3. Python 2 was discontinued with version 2.7.18 in 2020.(35)

2.1.3 MAD - Methodical Accelerator Design

MAD-X is a project with a long history, aiming to be at the forefront of computational physics in the field of particle accelerator design and simulation. Its scripting language is de facto the standard to describe particle accelerators, simulate beam dynamics and optimize beam optics at CERN.

MAD-X is the successor of MAD-8 and was first released in June, 2002. It offers most of the MAD-8 functionalities, with some additions, corrections, and extensions. The most important of these extensions is the Polymorphic Tracking Code (PTC) of E. Forest.

MAD-X is released for the Linux, Mac OS X and Windows platforms for 32 bit (on demand) and 64 bit architectures. The source code is written in C, C++, Fortan77 and Fortran90. (36)

2.1.4 Ion Beam Simulation (IBSimu)

Ion Beam Simulator or IBSimu is an ion optical computer simulation package for ion optics, plasma extraction and space charge dominated ion beam transport using Vlasov iteration. The code has several capabilities for solving electric fields in a defined geometry and tracking particles in electric and magnetic fields. The code is a constructed as a C++ library for maximal versatility and openness. IBSimu is usable in Linux and Windows. IBSimu is released under GNU General Public License. Notable features of the software are:

→ Solid geometry definition using 2D DXF files, 3D STL files, mathematical formulation, etc.
→ Finite Difference Method solver for 1D, 2D and 3D Poisson equations with edge smoothing.
→ Particle trajectory iteration in self-consistently calculated electric and imported magnetic fields.
→ Space charge density calculation from trajectories.
→ Vlasov iteration for self-consistent simulation of high space-charge beams.
→ Nonlinear plasma models for positive and negative ion extraction.
→ Interactive diagnostic tools.
→ Object-oriented and highly customizable. (37,38)

2.2 Description of the code

The following chapter will be used as a brief introduction to custom commands which are used to receive and analyze the data. C++ script is used as a main program for simulating and running the problem of low energy beam extraction while the Python scripts are used to analyze the data and receive necessary values for previously defined important variables which are used for matching to the RFQ.

2.2.1 C++ script

The beginning of the script introduces libraries which are custom written and came as the part of the package of IBSimu. Afterwards the constants are defined. They are used throughout the main program and they are:

Table 4: list of constants which are used throughout the main program

CONSTANT	VARIABLE	VALUE
Mass of helium atom in atomic mass units	Mass_4he_in_u	4.002602
Elementary charge	e0	1.60217662e-19
Speed of light	c0	299792458

After defining the constants in the program, the solid objects within the phase space are defined such as electrodes, solenoids and other optical elements that are within the simulation area. They are defined using the "bool" command whose syntax is as

"bool variable {logical statement}".

As the case is within this program logical statement is equations of straight lines or ellipses and circles defined by analytical geometry. Within this subprogram it is possible to define new variables which are used.

Subprogram that follows the definition of geometries is the most important part of the program and it consists of few different important aspects. In this subprogram following are defined:

→ Size of the mesh[5]

Using the command with its syntax

"Geometry geom (simulation space[6], definition of the simulation mesh, location of the coordinate origin, size of the mesh)"

the full description of the area within which the particles are traveling is defined.

→ Solids are defined as geometries and then afterwards can be used as electrodes

Command:

"Solid * sn = new FuncSolid(solid n); geom.set_solid (m, sn);"

is used to define previously plotted solids as functions that are representing electrodes (in the case of the problem discussed within this paper) or other objects defined with "bool". It is of upmost importance to clarify that "sn", "n" are numbers which are from 1 to max defined number within C++, as well that the number "m" is, for user defined solids, greater than 7 since the numbers from 1 to 6 are reserved for minimum x, maximum x, minimum y, maximum y, minimum z, maximum z boundaries.

[5] Area within which the simulation is taking place and the area in which the solid elements are defined and presented

[6] It can be 2D, 3D, cylindrical, etc.

→ Electrodes potential defined and given a value

Code for defining the potential to the now already defined solids is

"geom.set_boundary(m, Bound(BOUND_DIRICHLET/NEUMANN), value));".

Number "m" is the same number for the solids defined previously (or axis boundary conditions) while the boundary condition is either Dirichlet or Neumann. Neumann condition means constant first derivative of the potential with respect to the unit outward normal of the surface while Dirichlet means constant potential on the surface. Neumann condition is used to define the boundaries of the simulation box while Dirichlet is used for the boundary conditions of the electrodes. Value is the value of potential of either the solids previously defined or the box boundaries (such as x minimum, x maximum, etc.)

→ Poisson equation is solved

Main part of the subroutine is consisted of solving the Poisson equation. After all the boundary conditions and solids have been defined they are mapped with the function:

"geom.build_mesh()".

Function which solves the equation is:

"EpotBiCGSTABSolver solver(geom)",

while this function is used within this program it is important to underline that there are multiple more solvers for the Poisson equation included in the package but the one stated above is the most efficient.

The function "EpotBiCGSTABSolver solver" is not sufficient to solve the Poisson equation alone. It is necessary to define the mesh-based scalar field, space charge electric field, solving the interaction of the magnetic field which is produced, calculation of the interaction of electric field at a location from the electric potential data. These parameters are initialized with the following:

"EpotBiCGSTABSolver solver(geom);

EpotField epot(geom);

MeshScalarField scharge(geom), scharge_ave(geom);

MeshVectorField bfield;

EpotEfield efield(epot);

field_extrpl_e efldextrpl[6] = { FIELD_EXTRAPOLATE,
FIELD_EXTRAPOLATE, FIELD_SYMMETRIC_POTENTIAL,
FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE,
FIELD_EXTRAPOLATE };

efield.set_extrapolation(efldextrpl);"

→ Plasma is defined

Plasma is defined by using:

"InitialPlasma init_plasma(AXIS_X, 0.55e-3);"

in which we define the plasma on the x axis on the distance of 0.5 mm from the axis. The values of the plasma potential is defined as a standard variable as well as the electron temperature within the plasma which are as well defined with a function:

"solver.set_initial_plasma(Up, &init_plasma)"

→ Particles are defined

Main part and the main point of the paper is yet to be defined. The particles are defined within the Vlasov iteration loop after the initialization of plasma. They are defined with:

"pdb.add_2d_beam_with_energy (number of particles, current density,

Charge of the particles, particle mass, starting energy, parallel temperature,

Transverse temperature and the starting coordinates of the beam x1, y1, x2, y2)"

The measuring units for some of the values for particle definition are:

- Current density – $\frac{A}{m^2}$
- Charge of the particles – in electron charges
- Particle mass – in atomic units
- Starting energy – in electron volts (eV)
- Parallel temperature – in electron volts (eV)
- Transverse temperature – in electron volts (eV)

→ Files necessary for further result analyzation are outputted within this subprogram

Within the subroutine codes for exporting the particle description at the exit of the extraction system and the emittance values are written. For exporting the particles the code:

"std::ofstream fileOut("name of the file");

for(size_t k = 0; k < pdb.size(); k++) {

Particle2D &pp = pdb.particle(k);

fileOut << std::setw(12) << pp.IQ() << " ";

fileOut << std::setw(12) << pp.m() << " ";

fileOut << std::setw(12) << pp.location() << " ";

fileOut << std::setw(12) << pp.velocity() << " ";

fileOut << "\n";"

The code prints out the values for the charge of the particles, mass of the particle, location in the mesh (this is a control variable to filter the particles that have not reached the exit of the extraction system) as well as the velocity of the particles. Location and velocities are presented as Cartesian coordinates.

While the particles are used for later analysis the emittance file is there to provide the crosscheck of the emittance value as well as the twiss parameters value between the values calculated by the Python script and the values received directly from the main C++ code. Code for the emittance file is given as:

```
"TrajectoryDiagnosticData tdata;

std::vector<trajectory_diagnostic_e> diagnostics;

diagnostics.push_back( DIAG_Y );

diagnostics.push_back( DIAG_YP );

pdb.trajectories_at_plane( tdata, AXIS_X, geom.max(0), diagnostics );

Emittance emit( tdata(0).data(), tdata(1).data() );

ofstream dout( "filename.txt", ios_base::app );

dout << emit.alpha() << " "
<< emit.beta() << " "
<< emit.epsilon() << "\n";

dout.close();"
```

→ Plots are defined

At the end of the subroutine plots for the phase space and the emittance are defined with implementing the functions which call upon the libraries which are used for plotting within the IBSimu package.

The main part of the code is used to call up the subroutines which were previously explained as well to run debugging and ignoring some errors of the C++ compiler which can be caused by the IBSimu package.

2.2.2 Python script

As previously stated multiple times Python is used as a tool for analyzing the presenting the results and finally for matching to the RFQ acceptance. Two scripts for Python are used.

1) First script is used for calculation of the emittance, plotting the emittance and calculating the twiss parameters.

The script consists of already familiar modules. Main part of the script is the one that loads in the data from the particles file from the output in the main code. When using the script, it filters out only the particles that reach the end of extraction. When the data is filtered plot of the emittance is plotted and from within the value of the statistical emittance is calculated. Using the data, the twiss parameters are calculated. Using the definitions and equations from the chapter (**Error! Reference source not found.**) the normalized beam emittance is c alculated. The data calculated is then compared to the theoretical values of the beam emittance gained through equations (27).

2) Second script is used for matching to the RFQ

From the same set of data used in the first script the values for the RFQ (data taken from 1)) are imported (matching is explained in the chapter 3.8). Then the emittance is rotated to match to the RFQ.

Multiple iterations of the main C++ code have been used while the scripts in Python have been written and reused for different sets of data received from the particles outputted as part of the main code.

2.3 ECRIS – Nanogan and Supernanogan

As part of the simulation the suitable ion source had to be chosen. For the purpose of the paper and for the possible solution for a project within the University of Sarajevo in cooperation with CERN named "Sarajevo LINAC Project" the source Nanogan from the company Pantechnik has been chosen to simulate. As the company does not disclose the information of geometries of the ion source and some of the important components of the particles multiple sources have been used.

2.3.1 About the source – Nanogan

Nanogan is an ECR ion source, very compact and with good performance, which the magnetic circuit is entirely made with permanent magnets both for the radial and longitudinal fields, so the total electrical power is extremely low. The weight of the source is 11kg. Its performance is the best in its category, allowing the production of beam currents of 20 eμA of Ar8+. Nanogan can run with RF power up to 100W depending on the element and charge state needed. The maximum extracting voltage is 20 kV. This ion source is working in several laboratories and can also be installed inside electrostatic accelerators, like single-end or even Tandems terminals (39).

2.3.2 About the source – Supernanogan

Supernanogan is an ECR ion source, reliable and with high performance, which the magnetic circuit is entirely made with permanent magnets both for the radial and longitudinal fields, so the total electrical power is extremely low. The source includes 220kg of permanent magnets and 300kg of lead protection. Its performance is the best of its category, allowing the production of beam currents of 200 eμA of Ar8+ and C4+. Supernanogan can run with RF power up to 600W at 14.5 GHz depending on the element and charge state needed. The maximum extracting voltage is 30 kV. This ion source is working in several laboratories and is the reference source for Hadron therapy, the ultimate cancer treatment method. Supernanogan can be used in any kind of accelerators, i.e. RFQ, LINAC, Synchrotrons, Cyclotrons, etc.

Both of the sources are ECR sources and have similar components but the main difference is the one that Supernanogan can produce multiple heavy ions while the Nanogan is mostly used for producing ions which are lighter. For the purpose of this paper the combination of both sources was used (40).

2.4 Theoretical calculations

Before presenting the simulation and the results of the simulation it is necessary to present the theoretical calculations to have a reference to tune the simulations so the approximate results should be reached. The results of the theoretical calculations will be used for calculating the absolute and relative error after the presentation of simulation results.

To introduce the theoretical calculations it is necessary to review one of the previous chapters (Beam emittance on page 10). Beam emittance is defined as the six-dimensional volume limited by a contour of constant particle density in the (x, p_x, y, p_y, z, p_z) phase space. As mentioned in the previous chapters the emittance can be calculated from plasma temperature as well as from the magnetic field. For this instance, the calculations from the magnetic field have been used and the reasoning behind that is the one that the ECR sources use magnetic field to confine the beam. Formulas that are used are mentioned previously (32) and are (for geometrical and normalized emittance):

$$\varepsilon_{rms} = \frac{qBr_0^2}{8m_n v_z} \tag{52}$$

$$\varepsilon_{rms,n} = \frac{qBr_0^2}{8m_n c} \tag{53}$$

As part of the theoretical calculations only the formula for normalized beam emittance will be used (53) and that result will be compared with the one received from the simulations.

After listing the formulas, it is necessary to discuss the magnetic field. As it is known magnetic field (as it is explained in the introduction) can be separated into two components axial and radial. The one that has the most effect on the particles and their trajectory is the radial field which can be given as B_{ECR}. This magnetic field is calculated by the following:

$$B_{ECR} = \frac{m}{q}\omega_{RF} = \frac{m}{q}2\pi f \tag{54}$$

Where m is the mass of electrons, q is the charge and f is the frequency at which the source operates on.

In this case the following parameters were used:

For mass the mass of electron is used – $m = 9.10938356 * 10^{-31}$ kg

For the mass in formula for the normalized beam emittance the mass of helium nucleus is used as – $m_n = 4.002602 * u = 6.6422 * 10^{-27}$ kg

For charge the charge of electron is used and multiplied by 2 since it is the case for He^{2+} – $q = 1.60217662 * 10^{-19}$ C

For frequency the value is – $f = 10\ GHz = 10^{10}$Hz

The value of r_0 is used as the aperture of the source and the value is – $r_0 = 0.002$ m

The speed of light is used as – $c = 299792458\ \frac{\text{m}}{\text{s}}$

Mass of electron is used for the magnetic field since the electrons are the particles that with their movement as plasma create a confining magnetic field. After listing all the values using the python script the following is calculated for magnetic field:

$$B_{ECR} = 0.357057 \text{ T}$$

As for the emittance value of the magnetic field at the plasma chamber wall is double the amount of the B_{ECR}, as well as the contribution of the double of amount of electron charge in He^{2+} so the formula for normalized emittance shifts into:

$$\varepsilon_{rms,n} = \frac{qBr_0^2}{2m_n c} \tag{55}$$

Including the previous values in the formula (55) above the value of the normalized beam emittance is as following:

$$\varepsilon_{rms,n} = 0.05746 \text{ mm mrad}$$

2.5 Parameters for matching to the RFQ

To match to the RFQ ion beam has to meet the conditions which are necessary to transport the beam particles through the RFQ. Those conditions can be divided into longitudinal and transverse conditions. Longitudinal parameter is energy of particles and the energy spread, while transverse is the transverse emittance and the beam ellipse shape and orientation.

The 750 MHz RFQ (hope you describe it in the introduction) has the following matching conditions:

- Twiss parameters (alpha and beta)
- Beam emittance (geometrical)

Parameters stated above are prone to numerical change and some of them depend on type of the RFQ and the type of the particle.

Beam parameters are not the only conditions that have to be met. RFQ itself has a parameter called RFQ acceptance and it is referred as a maximum emittance that a beam transport system is able to transmit.

In conclusion the twiss parameters, beam emittance have to be tuned as beam parameters while the beam emittance has to match to the acceptance for it to be a successful match.

In the paper the following values for beam parameters and RFQ acceptance have been used for matching to the 750 MHz RFQ, as they could be found in "750 MHz radio frequency quadrupole with trapezoidal vanes for carbon ion therapy" written by Vittorio Bencini, Hermann W. Pommereneke, and others. The beam parameters are as follows (41):

Table 5: list of values of twiss parameters and beam emittance

Parameter	**Value**
Alfa []	0.3
Beta [mm/mrad]	0.01
Beam emittance (ε) [pi*mm*mrad]	0.2

3 Results and discussion

The following chapter will introduce the step-by-step analysis of the problem for matching the beam to the RFQ. It will consist on few subchapters which will include the discussion of key steps which lead to the final version of the code and an explanation on what has impacted the code with all the accompanying programs and codes used to better define the values.

3.1 Tutorial version

The program started as a code which was prewritten as a tutorial code on the IBSimu website (reference here). The code was written for one thousand particles of Hydrogen 1^+ which had the current density of the beam of 50 $\frac{A}{m^2}$. The starting simulation space was a 2D space in a xy plane with the values on x going from 0 to 120mm (on the plot it is given as 0.12 m as the scale is in meters) and the values on y which are going from 0 to 50mm (on the plot it shows as 0.05m because of the incidence previously mentioned). On the plot (Figure 5) it is visible that there are 3 electrodes present (blocks colored in blue) which were defined with mathematical formulas corresponding to type of the line edge is defined. The value of the voltages was -3 kV, -14 kV and -1 kV (respectively from left to right in the plot). On the plot with the green lines the equipotential lines have been presented. Trajectory of particles are represented by red lines and it can be seen that the large number of particles are leaving residue on the 3rd electrode, from the left, and it can cause damage caused by precipitate on the electrode which would impact the electric field produced by it. In this version of the code neither one of the plots were given as it is the case for the ones that will be presented in the following chapters (42).

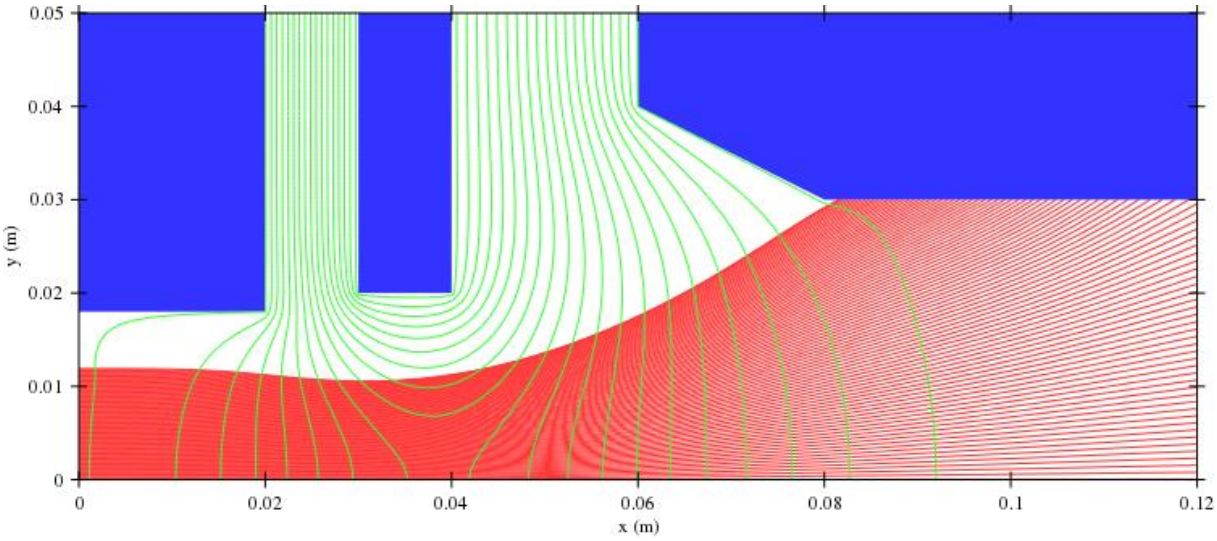

Figure 5: Simulation space for the tutorial code given by the IBSimu Tutorial

3.2 Discussion about parameters

In the chapter (cross reference to the chapter where the code is explained) the code which introduces the parameters to the simulation is explained but the values which it introduces are not mentioned.

Condition on which the execution efficiency of the simulation is based on is the number of particles. The number of particles of course has an impact on the final result but the best optimal solution for the number of particles, which best suits optimal efficiency and it does not have a great impact on the final result, was found to be between 5 000 and 10 000 particles. The number of particles in simulations which were run was dependent on the amount of conditions the simulation was executed with. The number of particles in the earlier iterations of the code were larger while the number was reduced due to the lack of computing power in the later iterations of the code since it did no major impact on the final results.

The values of mass and charge represent the values of which type of the beam the simulation is based on. In the case of the simulation analyzed in this paper it is a case of a $^{4}He^{2+}$ in the final version of the code. The beam in that case has a mass of 4.002603 atomic units[7] and a charge of +2 elementary charge[8].

Values of current density, parallel and transverse energy are dependent on the source while the value of the start energy is dependent on the value of the plasma potential plus the type of the source which is used. Current density is the amount of charge per unit time that flows through a unit area of a chosen cross section. Current density is a variable which depends on the source since it can fluctuate between some values which are tied on which ions is the source producing. For the same source there are multiple values for ion beam density. For the Nanogan source the values of maximum beam current are presented in the table (Table 6). (39)

Table 6: Beam intensity for various charge states given in electric μA

Ion / Q	1	2	4	6	8	9	12	14
H	1000							
He	1000	100						
Ar	300		140	45	20	5		
Xe							10	5
Ta					10		10	5
Au			10	9	8	6		2

[7] Atomic unit has a value of $1.67377 * 10^{-27}$ kg
[8] Elementary charge is $1.60217 * 10^{-19}$ C

Parallel and transverse temperatures are values which represent the temperature of electrons in the plasma which provide and the value of these temperatures are given in eV (electron volts). The value of parallel temperature is in most cases (and in the case of this paper) is 0eV since it does not impact the emittance. On the other hand, the value of transverse has a great impact on the emittance and this value provides an initial ionization to the injected gas. The value for this parameter is dependent on the temperature of the plasma where the value can be in the interval between 0.4 eV and 1 eV. (43)

The plasma potential is the value which then would decrease the number of cold electrons (which are not magnetically confined) which tend to escape more rapidly from plasma than the ions because of their higher mobility. As a result of that a positive plasma potential builds up to retard the escape of electrons and to accelerate the escape of the ions. It was discovered during operation of ECR sources that the production of high charge state ions can be substantially enhanced by adding a light support or mixing gas to the ECR plasma. There are many parameters involved in the ECR plasma. Plasma potential is a function of the related parameters: (44)

$$V_P = \mathrm{f}(n_e, n_i, n_0, T_e, T_i, m_i, B, \Omega_w)$$

where:

n_e – electron density distribution

n_i – ion density distribution

n_0 – neutral particle density distribution

Ω_w – plasma chamber configuration and wall condition

T_e – electron temperature

T_i – ion temperature

m_i – ion mass

B – magnetic field of the source

Plasma potential of an ECR source presented in the simulation is 20 V while the temperature of the electrons in the plasma is 10 eV. The starting energy is then calculated as the the plasma charge times the value of the charge.

Confinement of the starting beam is just a geometrical preset which depends on the source. In Nanogan the aperture at the exit of the source is at 4 mm so the starting beam confinement should be at the minimum at 4 mm but it can go into higher values since the aperture at the exit will confine the beam afterwards. (44)

3.3 Addition of electrodes

In addition to the tutorial code the problem of constructing the adequate electrodes immerged. The tutorial code is just a starting point for the problem since it does not provide a layout and voltage for the electrodes which would represent a layout for an ECR source while it could represent a layout for another source which is not an ECR source since the 2 cm aperture is not a value of aperture which is used for an ECR source. For the following reason the layout should be built and constructed as a realistic setup already present and commissioned as a part of the low energy particle beam accelerator. For the problem of this paper the layout of the ECR ion sources at CNAO facility in Pavia, Italy. (45) The ECR source that the low energy transport is used for is Supernanogan which is similar to the Nanogan source and the main difference is that the Supernanogan can produce heavier ions (e.g. Carbon 4^+) while the Nanogan source is more used for production of lower mass ions (e.g. Helium 2^+ on which the simulation will be based.

As it can be seen the difference between the two setup is that on the paper there are 4 electrodes

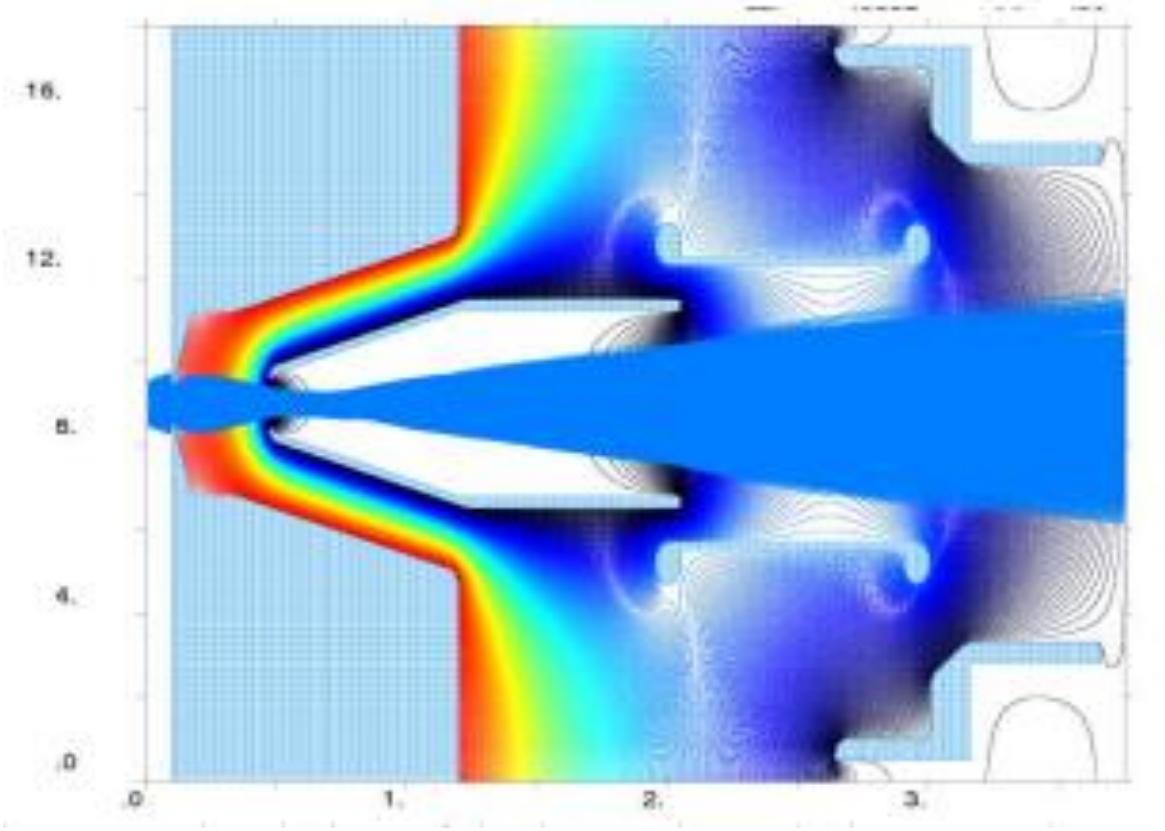

Figure 6: Desired appearance of the setup

(body of the source and plasma electrode, puller electrode and a focusing electrode which are used to form an electric field which then focuses the beam) while in the tutorial code there are only 3 electrodes visible. The changes had to be done to the tutorial code but the problem surfaced when the electrodes had to be designed as in the paper. The program Adobe Ilustrator was used to extract the dimensions and positions of the electrodes. After aligning the electrode borders as in (Figure 7) the measurements and coordinates were extracted to an Excel sheet and afterwards the equations for the electrodes were extracted.

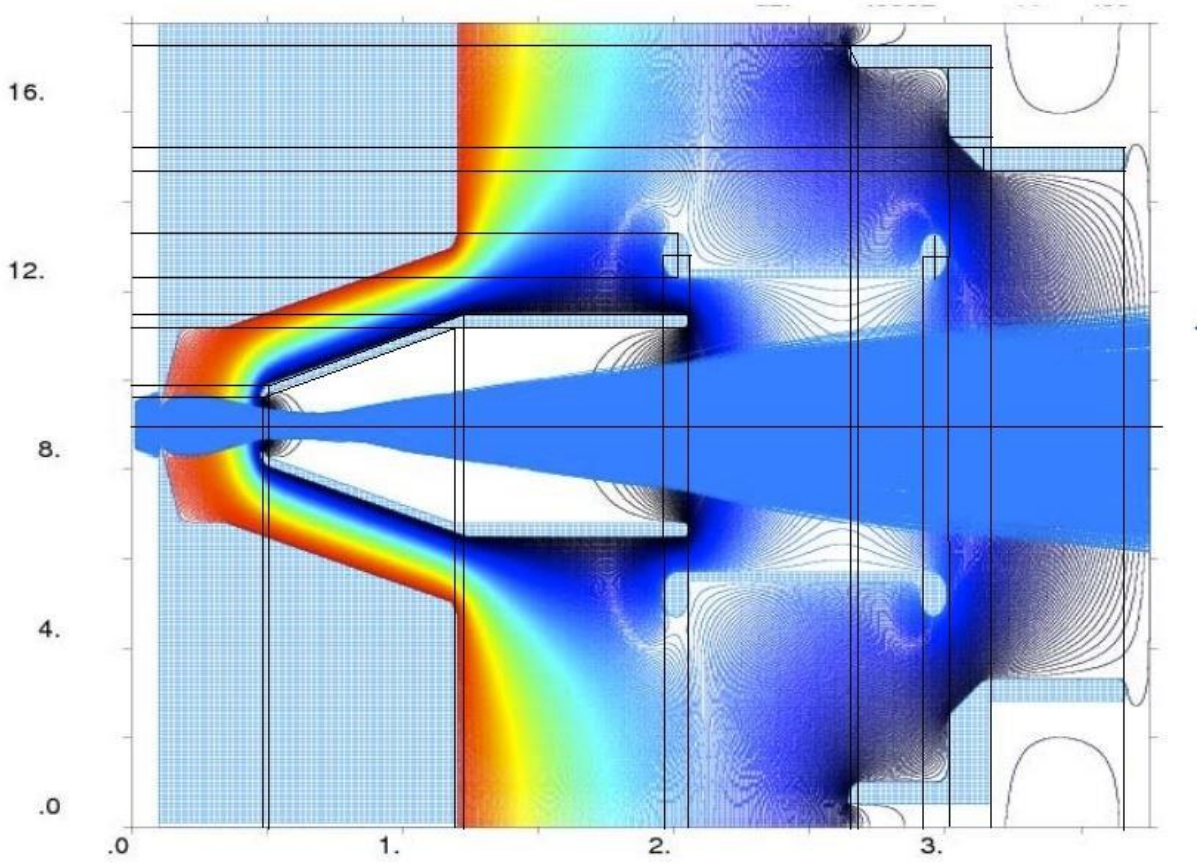

Figure 7: Process of extracting the values for the electrode geometries

Equations were written as geometrical equations with the following formulas:

- Distance between two points: $d = \frac{y_1+y_2}{2}$
- For a straight line: $y - y_1 = \frac{y_2-y_1}{x_2-x_1}(x - x_1)$
- For the ellipse: $b^2x^2 + a^2y^2 = a^2b^2$

After calculating the equations of the straight and curved lines they are inserted in the main code within the bool command (C++ script). The simulation spaced is halved for the better performance and better execution where the full space presented as in figure (Figure 12) will be discussed in chapter 0. The look of the simulation space after implementing the electrodes look as in (Figure 8).

Upon reconstructing the electrodes, the problem in adequate voltages occurred. Since the paper on which the electrodes were based on, did not provide the values for voltages of each electrode. Issue with the voltages got solved by providing the data from the paper: “Double einzel lens extraction for the JYFL 14 GHz ECR ion source designed with IBSimu” (46).

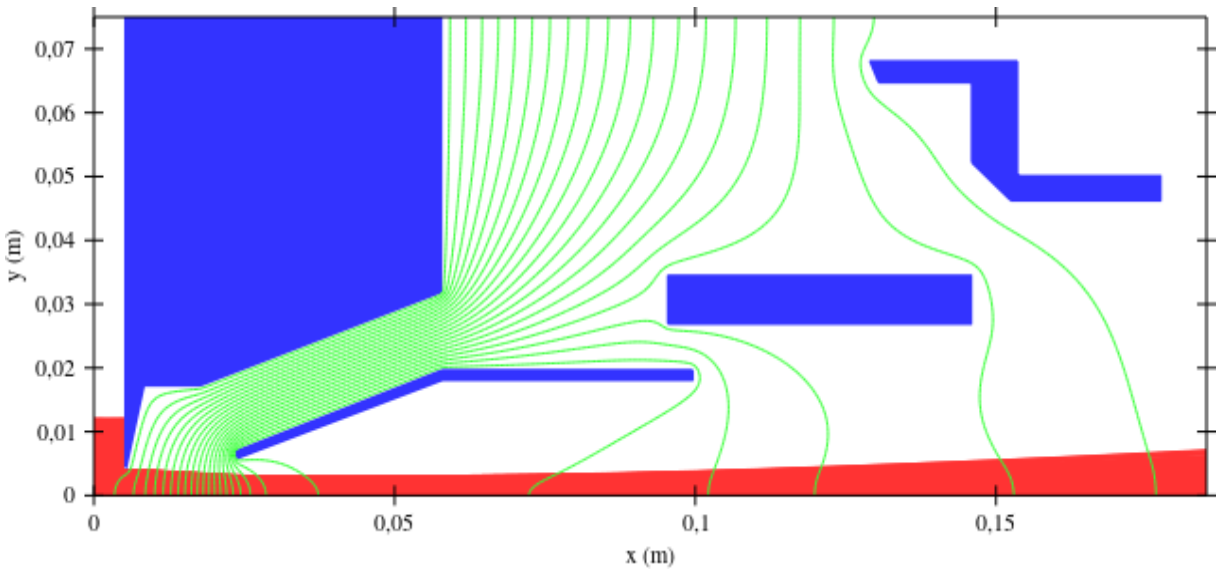

Figure 8: Electrodes shape and potential after gathering the values

Even with the parameters where the values for potentials were:

- Body of the source – 10kV
- Puller electrode – 0V
- Focusing electrode – 15kV
- Ground electrode – 0V

It can be seen that the beam has not changed drastically so the further investigation had to be done. After the consolation from the author of the IBSimu (Taneli Kalvas, Ph.D.) and engineer from company Pantechnik (Arun Annaluru, Ph.D.) the following voltages have been reached in the final version of the code:

- Body of the source – 0V
- Puller electrode – -25.50kV
- Focusing electrode – -45.0kV
- Ground electrode – -30kV.

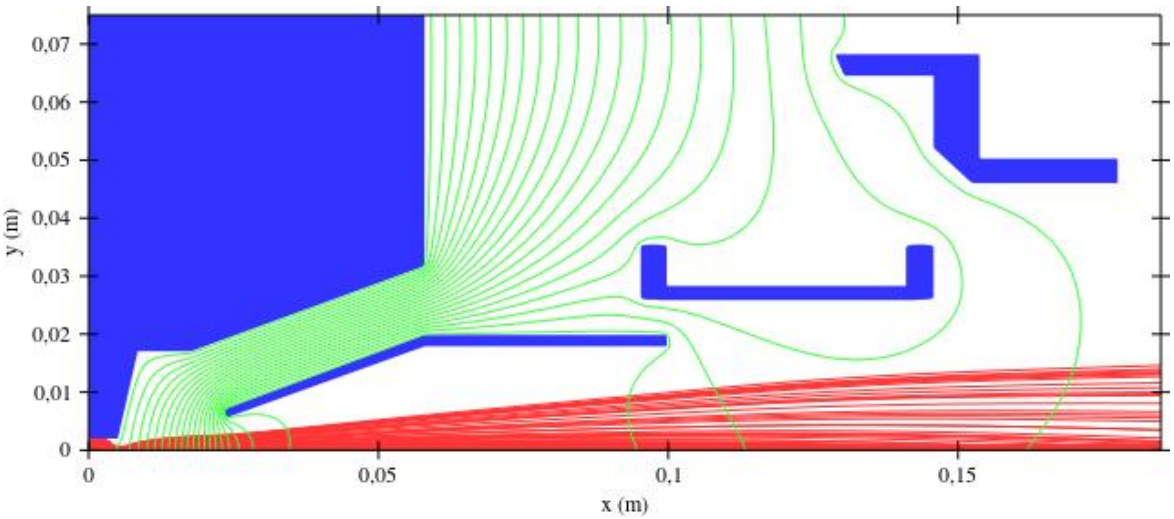

Figure 9: First version of the electrodes as depicted in the CNAO paper.

What is different to the electrode potential from the paper is that now the source is on 0V while other electrodes are relative to the amount of the potential of the source. Since the source of Nanogan is at the working voltage between 20kV and 30kV for it to be at 0V other electrodes have to have the value of their voltage minus the working voltage of the source (e.g. if the source is at 24kV if we reduce it down to 0 other electrodes have to be their value minus the value of 24kV).

There is a noticeable difference between the electrodes of the tutorial simulation and the simulation for the Nanogan source. In the final version of the code the electrodes were slightly differed to the setup in this chapter but in the chapter (3.7) will the changes be discussed and the reason for implementing them.

3.4 Expansion of the code

Once the electrodes were reconstructed from the paper the issue of printing out the results and controlling the output particles emerged. Since the original code did not output any data except the simulation are plot the original code had to be expanded.

Firstly, the part of the code dedicated to the particle output has been made which then dumped the tracked particles coordinates (x, y and z coordinates) and their velocities (per velocity component v_x, v_y, and v_z) to a file for further analysis using Python.

Second addition to the code was as well an addition to the plot after the code execution. The addition was that of an interactive menu which then had options to plot the particle distribution in the phase space[9] for either x, y or z axis. The plots consist of the points which correspond to each of the particles which reached the desired point and their x coordinate is their position (either x, y or z) and y coordinate is the corresponding angle. The plots received then can be analyzed further with additional tools which are available with the program.

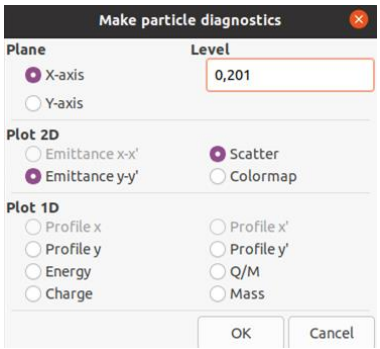

Figure 10: appearance of the pop-up menu that is the result of the code expansion

[9] In dynamical system theory, a phase space is a space in which all possible states of a system are represented, with each possible state corresponding to one unique point in the phase space.

The pop-up menu can be used to plot the distribution of the particle beam characteristics (such as energy, charge, mass, etc.). After choosing to plot the emittance figure (number here) can be seen. On the figure values of Twiss parameters: alpha, beta, gamma and geometrical beam emittance can be found. Such values can be used in further discussions and beam analysis for the beam to proceed to further matching to the RFQ.

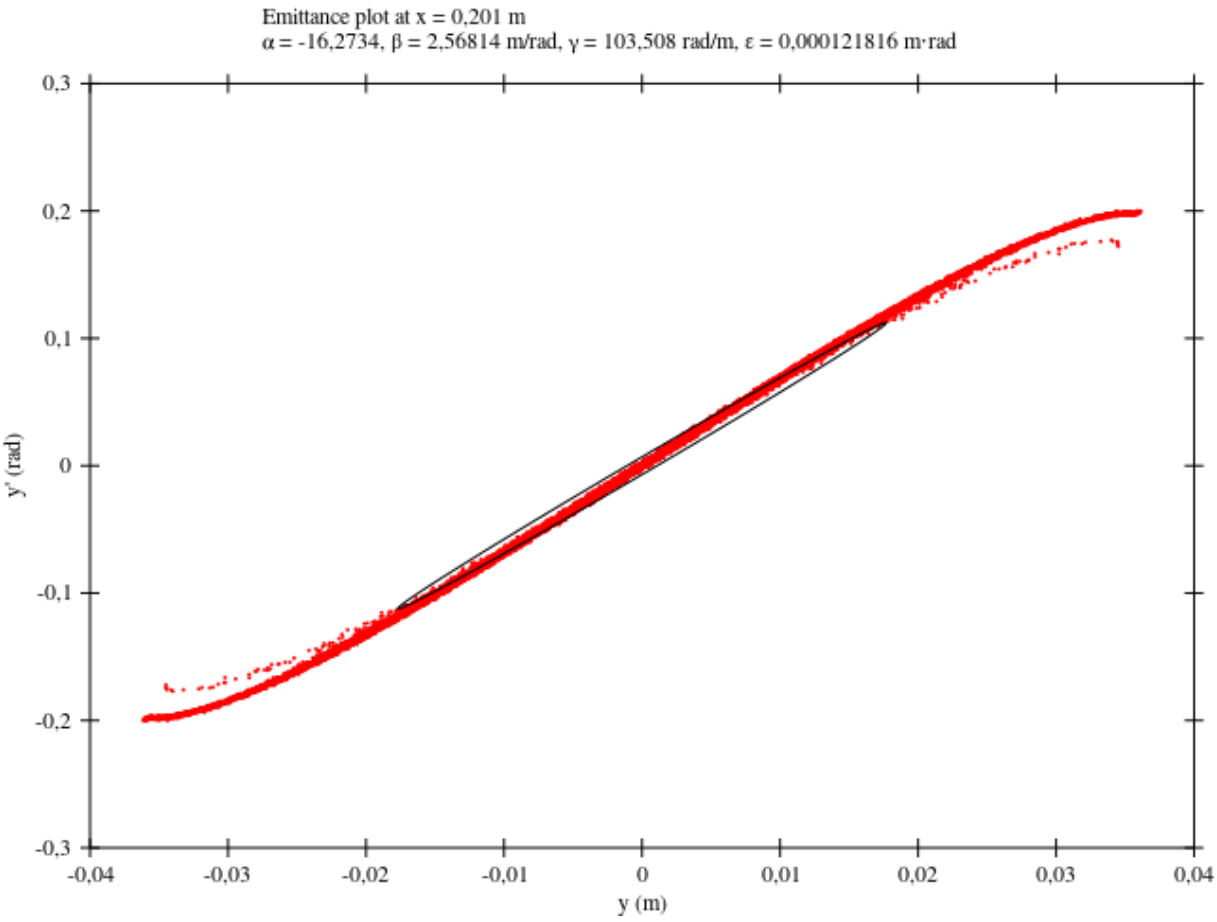

Figure 11: Beam emittance plot within the IBSimu program

Additional expansions of the code are presented as the addition of the plasma potential

Addition of the plasma potential to the simulation made it more accurate. After the addition of the plasma it had to be confined to the area where it has impact. The confinement area is the source. Plasma is defined with two values which are:

- For plasma potential 20V
- For electron temperature (from the electron temperature paper reference here) 10eV

3.5 Control script for calculating emittance from the extraction parameters

The result of the expansion of the code is the output file referenced in the chapter (cross-reference for the previous chapter here). Since the IBSimu package provides only the final values for emittance, alpha, beta and gamma and the values of the particle parameters[10], for the detailed analysis the additional script had to be made. For the purpose the custom made script was made in Python in which all the necessary further calculations were made.

The script is written to assist in calculation of the percentage of particles which would suit the acceptance of the RFQ using the values from the output file. Using the script, the tracking data was sorted as every row represented a particle while the columns had the values of mass, charge, position and velocity. In the continuation of the script the file was filtered to retain the particles which x value was the maximum and equal to the max value of x in the simulation space. Furthermore, the file was expanded to include:

- Values of the angle of particle trajectory with respect to direction of the beam which was calculated with the following formula: $1000 \times \arctan\left(\frac{v_y}{v_x}\right)$ which value is in mrad (milirad)
- Values of the position of the particle on the y axis in mm

As a mean to control the main simulation in the calculations of the beam parameters (alpha, beta, gamma and beam emittance) in the script the calculation for these beam parameters was performed from the output file. All the equations used are present in the theoretical introduction of the paper, to be exact the chapter (number here).

The script includes the emittance plot as well which always corresponds to the one the program plots. The reason is that both were made from the output file but the difference is that the IBSimu is plotting internally and is automatic while the Python script needs an input (in the version of the output file received from IBSimu) of the particles and hand written code to operate.

The RFQ acceptance is imported to this script which uses it to calculate the percentage of the particles that achieve the acceptance. The script is important to the simulation since the IBSimu program is not equipped to calculate the matching while the script could be looked at as the add on to the program where it would not work without the main program. It is possible to integrate the script and use it in the main program and write it in C++. The issue would arise on the optimization of the code since the main purpose of the code is to calculate the trajectory and the simulation is already heavy and requires a lot of computation power while the integration of the would lower the efficiency of the simulations where the program would require more time to operate. The solution of an add-on script is a better solution since it quickens the process of the main simulation while the lack would be only the required knowledge to write it in a different programing language to the main program written in IBSimu package.

[10] Mass, charge, coordinates and velocity

3.6 Optimization of beam parameters

Once the code expansion was finished, the script which analyzes the output was created and the electrodes were constructed only parameters left to be defined are beam parameters. Since the beam which was meant to be produced is $^{4}He^{2+}$ the mass and the charge are known. As the simulation is rather a one of an ECR source the beam current density is known and ion source physics have defined the temperature (longitudinal and transverse) of the ions. To manipulate the beam, the geometry and voltages of electrodes (body of the source, puller, plasma and focusing electrodes) had to be changed.

3.6.1 First iteration of the code

In the first iteration of the parameters the values of voltages and the beam parameters are presented in the table (Table 7).

Table 7: Table of beam parameters and values for electrode voltages in the first iteration of the code

Parameter	Value
Number of particles	10 000
Mass (in atomic units)	4
Charge (in values of elementary charge)	1
Beam current density (in $\frac{A}{m^2}$)	80
Starting energy of the particles (in eV)	20 000
Parallel temperature (in eV)	0
Transverse temperature (in eV)	1

Parameter	Value
Body of the source (in kV)	20
Puller electrode (in kV)	-1
Focusing electrode (in kV)	2
Ground electrode (in kV)	0
Beam start confinement (in mm)	X axis = (0.0 – 0.0)
	Y axis = (0.0 – 12)

Provided values in the table (Table 7) simulate the plot and emittance as on (Figure 12).

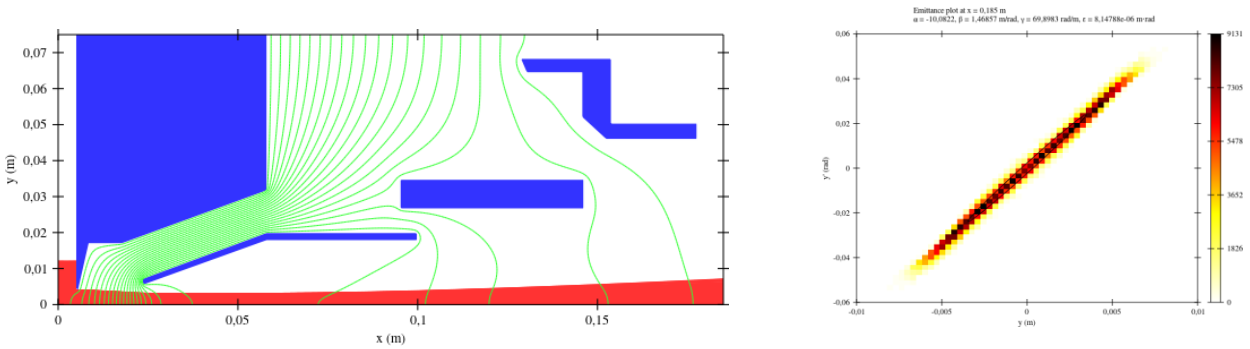

Figure 12: Plot for the simulation space (left) and beam emittance (right) received through the simulation for the first iteration of the code

On the results from the figure (number here) the values for emittance are presented in the table (Table 8).

Table 8: Value of the beam emittance and normalized beam emittance for the first iteration of the simulation

Beam emittance (in mm mrad)	8.14788
Normalized emittance (in mm mrad)	0.37742

For the presented results it can be seen that the result of the simulation is not realistic. From figure (Figure 12) on the left side the simulation space is presented and it can be seen that the beam of particles is flat and that is not realistic which means some of the parameters from table (Table 7) are not correct so further changes to the parameters are supposed to be made.

3.6.2 Second iteration of the code

Following the first iteration of the code the values for voltages are changed to match the suggestions from Taneli Kalvas[11] and Arun Annaluru[12]. To addition of the voltage changes further changes to the beam parameters were made. Since the value from the first iteration were not regular as they were taken from the tutorial code further investigation was required. It was discovered that the values for some of the parameters were higher than expected and those values were corrected. Following the further research in the Nanogan source (for which the simulation is loosely based on) the yield of He^{1+} ions is negligible and for the source commission as a laboratory equipment for further research the change to ions of He^{2+} was decided. He^{2+} is used more for Ion beam analysis then his counterpart in He^{1+}.

[11] Kalvas Taneli, Staff Scientist at University of Jyväskylä, Finland

[12] Research and development engineer at Pantechnik

The values of beam parameters are presented in the table (Table 9).

Table 9: Values for beam parameters in the second iteration of the code

Parameter	Value
Number of particles	10 000
Mass (in atomic units)	4
Charge (in values of elementary charge)	2
Beam current density (in $\frac{A}{m^2}$)	80
Starting energy of the particles (in eV)	50
Parallel temperature (in eV)	0
Transverse temperature (in eV)	1

Parameter	Value
Body of the source (in kV)	0
Puller electrode (in kV)	-21
Focusing electrode (in kV)	-18
Ground electrode (in kV)	-20
Beam start confinement (in mm)	X axis = (0.0 – 0.0)
	Y axis = (0.0 – 12)

Values from the table (Table 9) after the simulation present the plots seen in the figure (Figure 13).

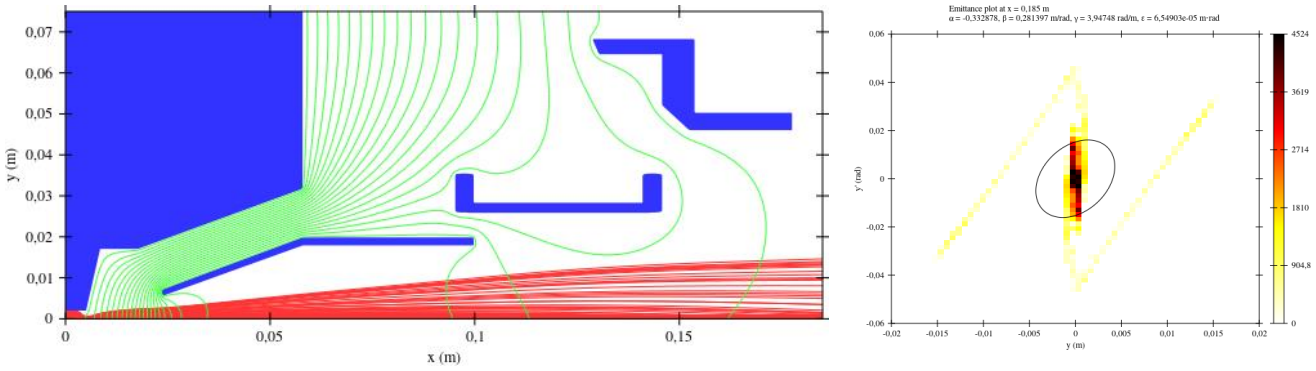

Figure 13: Plot for the simulation space (left) and beam emittance (right) received through the simulation for the second iteration of the code

The values of beam emittance from the second iteration of the code are presented in the table (Table 10)

Table 10: Value of the beam emittance and normalized beam emittance for the second iteration of the simulation

Beam emittance (in mm mrad)	65.4933
Normalized emittance (in mm mrad)	0.214843

From the data presented in the figure (Figure 13) and from tables (Table 10) the space of the simulation has a more realistic look while the beam emittance has taken a non-elliptical shape which leads that the simulation is not accurate with these parameters. As a reason to approve the parameters is the beam emittance as well. The beam emittance provided in the second iteration of the code is higher than expected and has to be lower for the source to have a successful matching to the RFQ. Few more iteration are required for the simulation to be processed by the follow-up script and for the setup to be matched to the RFQ.

3.6.3 Third iteration of the code

The beam emittance of the second iteration of the code (Figure 13) was not elliptical so the third iteration of the code will repair this issue while trying to retain the beam parameters since they are the theoretical values which in practice do not change. After the further review changes to the voltages of the electrodes and the addition of the plasma potential in the simulation. Explanation of the addition of plasma potential is in the chapter (3.4).

Values for the beam parameters and the electrode voltages of the third iteration are presented in the table (Table 11).

Table 11: Values of the beam parameters and electrode voltages for the third iteration of the code

Parameter	Value
Number of particles	10 000
Mass (in atomic units)	4
Charge (in values of elementary charge)	2
Beam current density (in $\frac{A}{m^2}$)	80
Starting energy of the particles (in eV)	50
Parallel temperature (in eV)	0
Transverse temperature (in eV)	1

Parameter	Value
Body of the source (in kV)	0
Puller electrode (in kV)	-26
Focusing electrode (in kV)	-25
Ground electrode (in kV)	-30
Beam start confinement (in mm)	X axis = (0.0 – 0.0)
	Y axis = (0.0 – 12)

After simulating the beam parameters which are stated in the table (Table 11) the plots in figure (Figure 14) have been obtained.

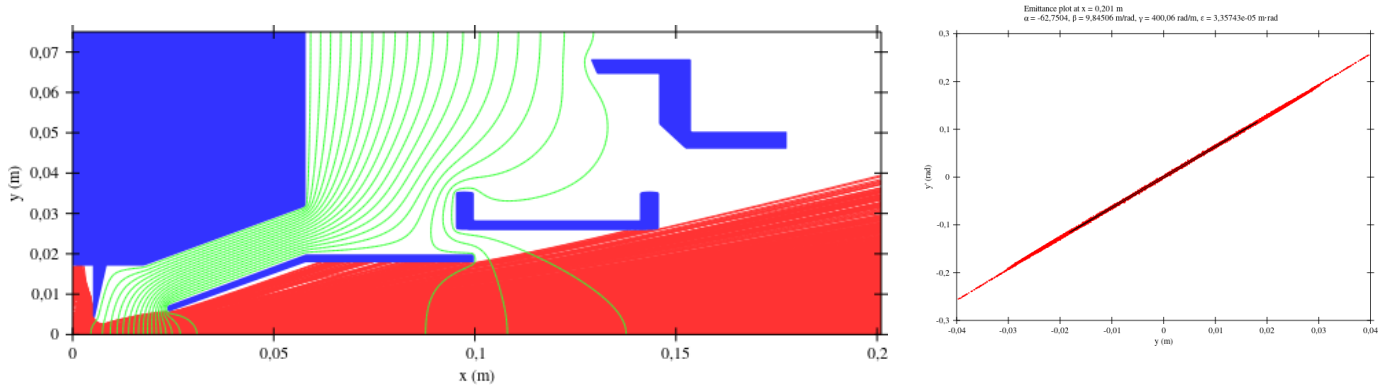

Figure 14: Plot for the simulation space (left) and beam emittance (right) received through the simulation for the third iteration of the code

From the plots in figure (figure number here) the values for emittance have been acquired and they are presented in the table (Table 12).

Table 12: Value of the beam emittance and normalized beam emittance for the third iteration of the simulation

Beam emittance (in mm mrad)	58.483
Normalized emittance (in mm mrad)	0.22364

After the analysis of the figure (Figure 14) and the table (Table 12) it is deduced that the simulation plot is taking a realistic shape after the third iteration. Beam emittance is still to improve on. The beam parameters are set for a standard He^{2+} beam and it is on the electrode shape to be altered further for the code to reach the final version. The beam is currently (in this iteration) leaving the residue on the electrodes which is not acceptable since the exchange of the electrodes would be required before the desired time. This affects the performance of the source. In this setup of electrodes on the equipotential lines we do not see the plasma effect on the beam which means that the geometry must be refined to benefit this effect. Further on the simulation area was expanded a few millimeters and it proved that it has effect since the realistic system is 0.201 m long (the extraction system based on 3 electrodes).

3.7 Final version of the code

Once the third iteration of the code was finished there were a couple of minor errors to be corrected. One of those errors was the geometry of the electrodes which contributed to the lack of plasma effects on the beam which then led to nonrealistic plots of beam emittances and to beamlines which hit the electrodes.

The final version of the code was different to the previous iteration (the third iteration) in a way that on the first electrode a small edge was added to contribute to having the plasma effect on the beam. It contributed by bending the potential to be more in tune to the plasma effect on the beam. In the figure (Figure 15) can be clearly seen that the plasma effect is present now which was not the case in the third iteration of the code (3.6.3). The values for the voltages on these electrodes were lowered as the source can operate on the voltages between 21 kV and 30kV. Relative to this all the other voltages were lowered

Other difference to the third iteration is a small change in the beam parameters. Instead of the beam current density being 80 $\frac{\mathrm{A}}{\mathrm{m}^2}$ it is raised to 96 $\frac{\mathrm{A}}{\mathrm{m}^2}$. The value of the starting energy of the ions is changed to 10 eV instead of 50 eV while other parameters remained the same. The change to the beam starting confinement in y_{max} is for the purpose of easier compiling of the program and does not affect the outcome to the final result. The reason for that is the aperture of the body of the source which confines the beam to 4 mm weather the beam confinement is larger or less than the previous value. Quick note is that the value that represents the number of particles is changed as well but this number only has an influence on compile time of the simulation so since the input data is more hardware heavy then in the previous iteration, the number is lowered from 10 000 to 5 000.

Table 7: Values of the beam parameters and electrode voltages for the final version of the code

Parameter	Value
Number of particles	5 000
Mass (in atomic units)	4
Charge (in values of elementary charge)	2
Beam current density (in $\frac{A}{m^2}$)	96
Starting energy of the particles (in eV)	10
Parallel temperature (in eV)	0
Transverse temperature (in eV)	1

Parameter	Value
Body of the source (in kV)	0
1st electrode (in kV)	-28.5
2nd electrode (in kV)	-24.5
3rd electrode (in kV)	-30
Beam start confinement (in mm)	X axis = (0.0 – 0.0)
	Y axis = (0.0 – 10)

As a output of the simulation with the parameters from table (table number here) the simulation space and the emittance plot are presented on figure (number here).

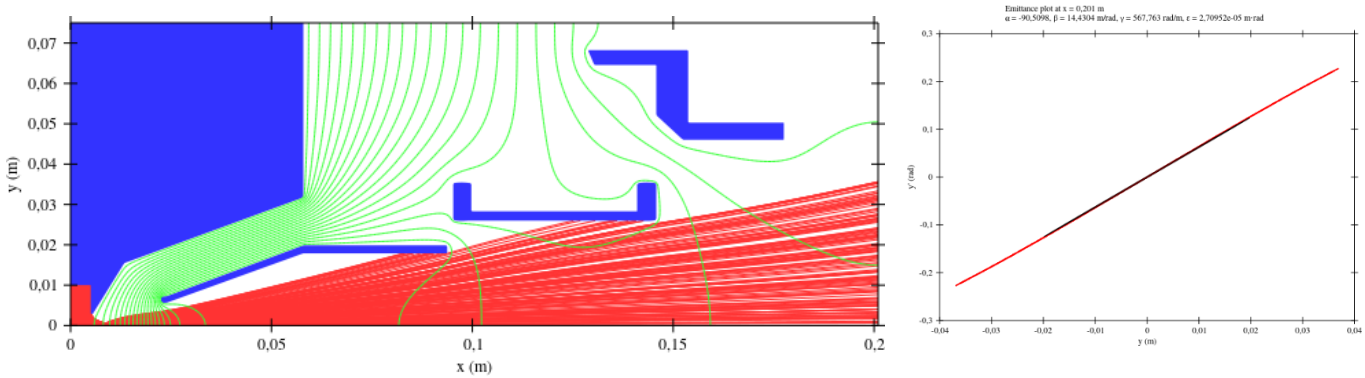

Figure 15: Plot for the simulation space (left) and beam emittance (right) received through the simulation as the final version of the code

The values of geometrical and normalized beam emittance are given in the table (number here).

Table 8: Value of the beam emittance and normalized beam emittance for the final version of the code

Beam emittance (in mm mrad)	26.0448
Normalized emittance (in mm mrad)	0.0341558
Relativistic beta	0.00543
Relativistic gamma	1.0000147

The data recovered from table (Table 8) and figure (Figure 15) it can be seen that there are many improvements to the third iteration of the code. The small addition to the first electrode proved significantly valuable since the effect of the plasma is visible through the shape of equipotential lines. There are few notable deficiencies which is the precipitate of particles on the electrodes which will, in time, cause the system to perform less efficient but the solution to this would be the exchange of the electrodes which happens after the lifetime is exceeded. This is improved after the company which produces the source provides the full setup data since the problem in this paper is solved while gathering data from different papers (41,45,46). The value of normalized beam emittance that can be found in chapter (2.4) is the same order of magnitude in correspondence to the value found in the table (Table 8). The value of the geometrical beam emittance is within the acceptable margin of error for the type of the ECR source simulated within the problem.

The setup presented in this final version of the code represents the best suitable option to start the matching procedure with the RFQ. The process for matching to the RFQ will be explained within the chapter (3.8).

3.8 Matching to the RFQ

Matching of to the RFQ successes the simulation presented in the chapter (3.7). The process of matching consists of few steps.

First step is writing a MAD-X code where the defining of two solenoids which would focus the beam. Solenoids of 0.2 m of length are used and their position is defined. The code also has to contain the values of twiss parameters to function correctly. Output of the code are the values of k factors (which are used afterwards to write a transfer matrix) and a file which can be used to read out parameters such as a transfer matrix, k values, mass of the particles etc.

After defining the code used to apply solenoids to the simulation a small compilation is needed and as an output for the values received as part of the final version of the code (3.7). For the calculation of the plot for beam emittance the transfer matrix is used.

Transfer matrix is a final result of multiplying multiple drift matrices and transfer matrices of both solenoids. The result for the problem discussed as combining 3 drift matrices (drift matrices are defining the space between objects within the simulation, between the simulation area and the first solenoid, between two solenoids and the final one between the 2nd solenoid and the entrance to the RFQ). Drift matrices are combined with the matrices of the solenoid which there are two for both solenoids. The transfer matrix is then calculated by multiplying the drift matrices and solenoid matrices with each other in a reverse order (from the RFQ to the starting area of the simulation).

Drift matrices are defined as:

$$M_D = \begin{pmatrix} 1 & L & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & L \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{56}$$

where the value of L is the length in between the elements of the system. Values for L are:

- Between the simulation space and the first solenoid the value is 0.3 m
- Between the solenoids the value is 0.1 m
- Between the second solenoid and the entrance to the RFQ the value is 0.3 m

The matrices are defined as M_{D1}, M_{D2} and M_{D3} respectively.

A Solenoid matrix can be defined as a product of two matrices which are a global focusing matrix in both xx' and yy' plans and a rotation matrix. These are defined as:

$$M_{\text{rotation}} = \begin{pmatrix} \cos kL & 0 & \sin kL & 0 \\ 0 & \cos kL & 0 & \sin kL \\ -\sin kL & 0 & \cos kL & 0 \\ 0 & -\sin kL & 0 & \cos kL \end{pmatrix} \tag{57}$$

Variables k and L are:

- k is read as an output from MAD-X
- L is the length of the solenoid

$$M_{\text{focusing}} = \begin{pmatrix} \cos kL & \frac{\sin kL}{k} & 0 & 0 \\ -k \sin kL & \cos kL & 0 & 0 \\ 0 & 0 & \cos kL & \frac{\sin kL}{k} \\ 0 & 0 & -k \sin kL & \cos kL \end{pmatrix} \tag{58}$$

The solenoid matrix is calculated as:

$$M_{\text{solenoid}} = M_{\text{rotation}} \times M_{\text{focusing}} \tag{59}$$

The final value of the transfer matrix is:

$$M_{\text{transfer}} = M_{D3} \times M_{\text{solenoid2}} \times M_{D2} \times M_{\text{solenoid1}} \times M_{D3} \tag{60}$$

Calculations for the transfer matrix were done by using MS Excel as it represented the most time efficient way to calculate them. The reason behind that is that MS Excel functions provide an easier way to exchange the values for the parameters and elements represented in the problem.

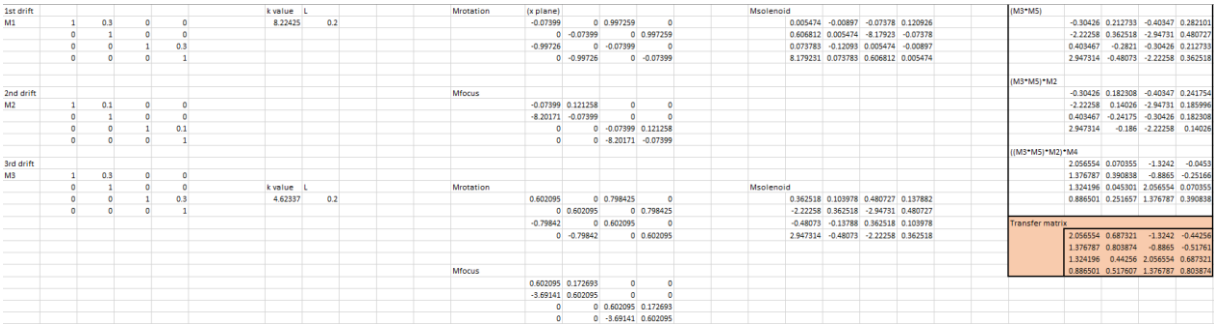

Figure 16: Calculation of the transfer matrix done in MS Excel

After the calculation the matrix is transferred to the Python script so it can be used to transform the original vector of position and the angle of particles (y,y') into the new one which would represent the rotation of the particles. The equation for the vector of the new position for an individual particle is gathered from the equation:

$$Vector_{newposition} = M_{transfer} \times Vector_{oldposition} \tag{61}$$

The value for the vector of position and angle has to be 4x1 where the values are:

$$Vector_{position} = \begin{pmatrix} y \\ y' \\ z \\ z' \end{pmatrix} \tag{62}$$

Since the simulation is symmetrical in the z and y planes the values of z and z' are equal to the ones of y and y'.

Within the Python script the values for the vector position of original particles is extracted from the file which is the output of the IBSimu code. Since there is no argument for an angle in the output file a new column had to be added. The process of addition is explained within chapter (2.2.2) which for has a result of expanding the file with 2 additional sets of variables. The vector of new position is then calculated within the script which then gives a rotated beam emittance

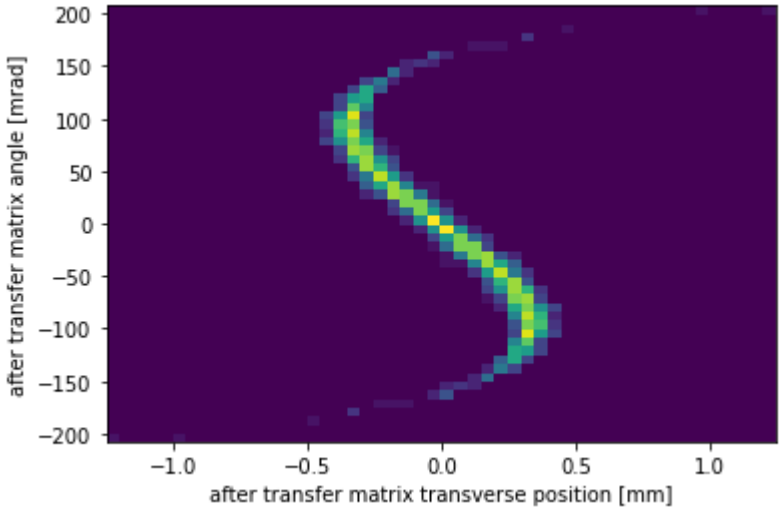

Figure 17: rotated beam emittance plot which is calculated within a python script

plot as seen on figure (Figure 17).

After rotating the particles and receiving the new emittance plot the acceptance of the RFQ is defined. It is a ellipse which is presented on the figure (Figure 18).

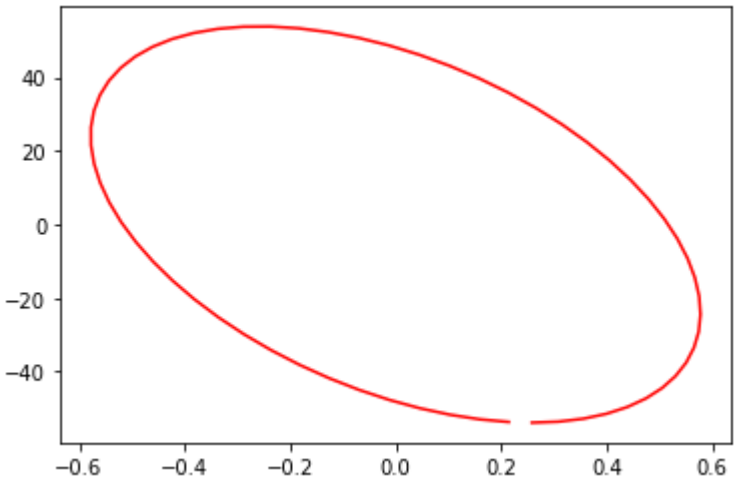

Figure 18: acceptance plot for the RFQ

The final part of the simulation would be to see if the matching to the RFQ is successful. To verify that the two plots (Figure 17 and Figure 18) have to be overlapped and the percentage of the particles included within the acceptance. The plot can be seen within the figure (Figure 19).

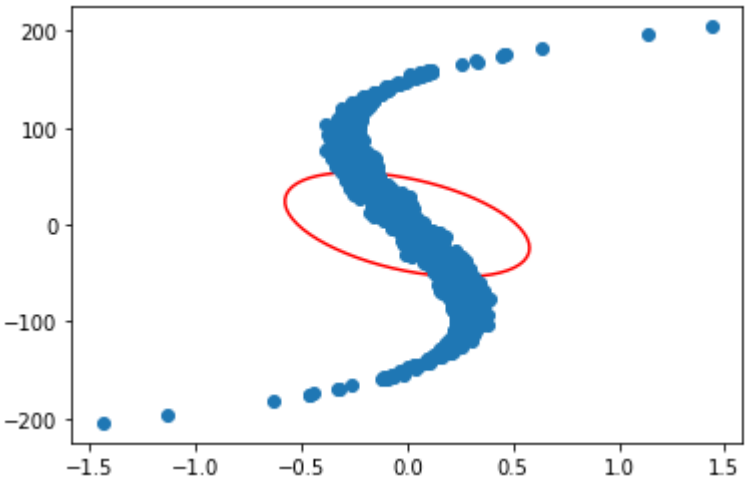

Figure 19: overlap of the beam emittance and the acceptance

The percentage of the particles contained within the acceptance of the RFQ can be obtained by overlapping the two ellipses. The plot seen in figure (Figure 19) is enlarged to just only the acceptance ellipse and it can be seen on the figure (Figure 20).

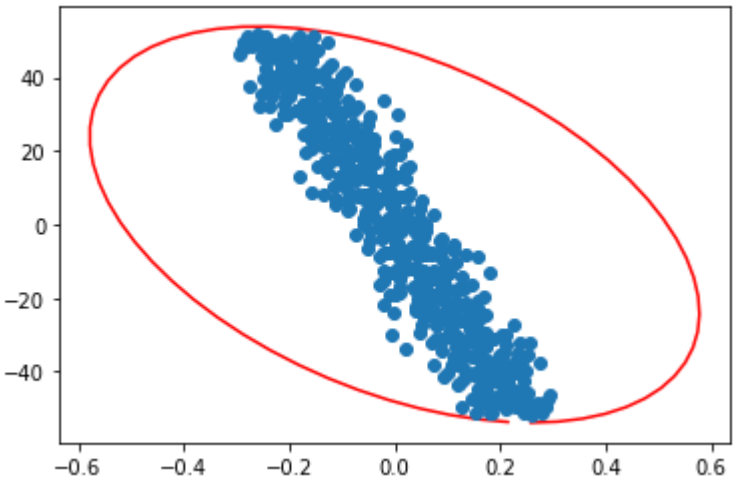

Figure 20: enlarged acceptance ellipse with the beam emittance ellipse within

The number of particles included in the acceptance ellipse of the RFQ is 71.32% of all particles that reached the end of the trajectory.

3.9 Future addition to the code and possible outcomes

Since the simulation and the matching for a halved setup is finished a discussion for future addition to the code and possible outcome can be discussed.

First addition to the code would be the expansion of the space on which the simulation is taking place. Since if the simulation areas are compared from figure (Figure 6) and the figure (Figure 15) it can be seen that the area on the figure (Figure 15) is the area halved of the original one. The expansion is done by moving the plot space by its height upward and axially symmetrically mapping the electrodes. Second part would be moving the source of ions and plasma values to the new spot (which now would be the original value plus the height of the simulation space). The new figure of the simulation space can be seen on figure (Figure 21).

The reason why this expansion is not added to the final version of the code is the emittance plot received from IBSimu which has an error which was not managed to be debugged by the user and the issue was forwarded to the author of the package (Kalvas Taneli).

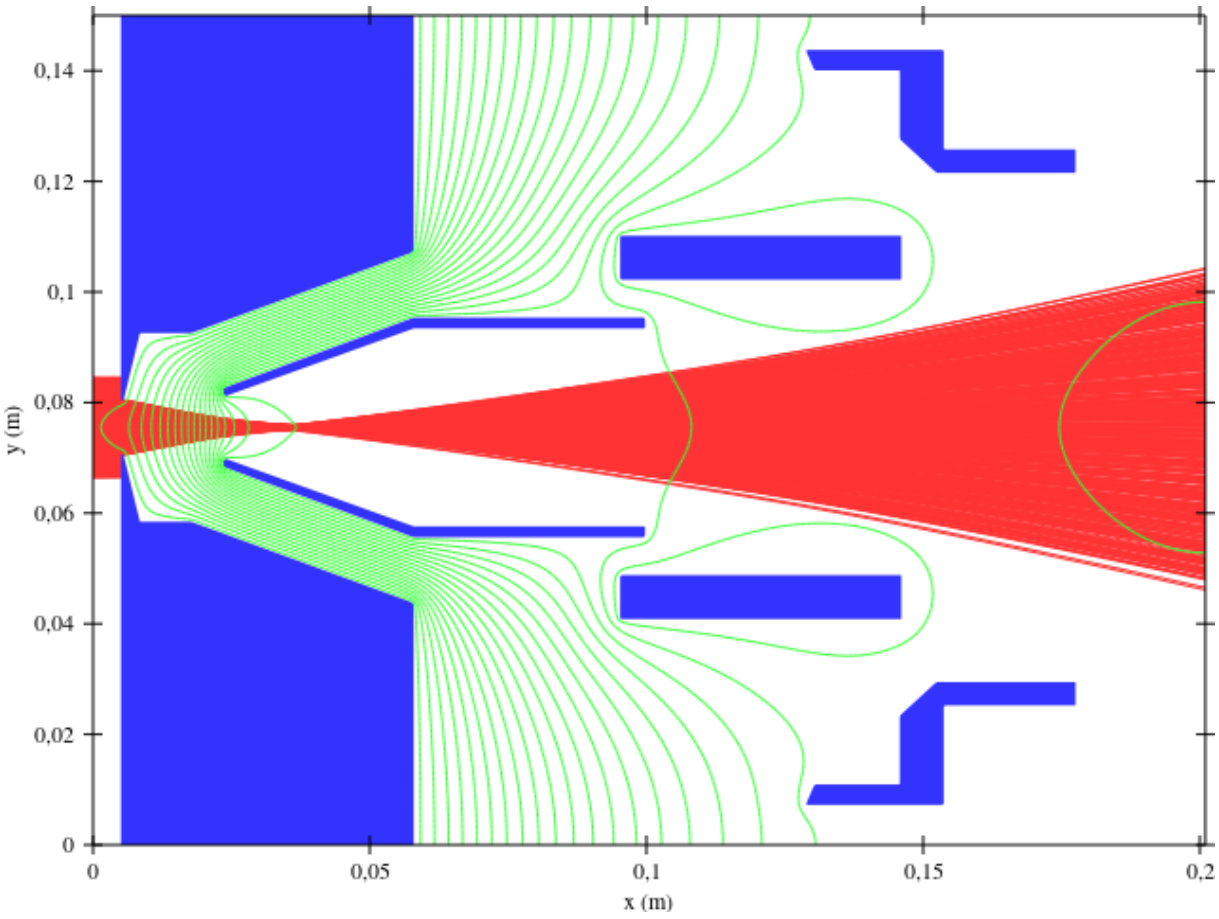

Figure 21: simulation area after expansion of the area to most closely resemble the area of simulation from the CNAO paper

The second addition would be to observe values of electrode potentials and beam parameters and exchange them with the values of the electrode potentials and beam parameters from the author of the setup which the simulation is based on. The values included in the final version of the code (3.7) are correct, but they are gathered from different sources. The acquired values would improve the percentage of particles which can be accepted to the RFQ.

The third addition would be the addition of multiple particle beams to the simulation. The real case of ion beams generated by the source are the ones where multiple beams are generated by the source. When having He^{2+} beams the source produces He^{1+} ions as well. The He^{1+} beam would be present in the simulation and they would dissipate while entering the RFQ and only He^{2+} ions would remain. The simulation, if the particles of He^{1+} are added, would look like as seen on figure (Figure 22).

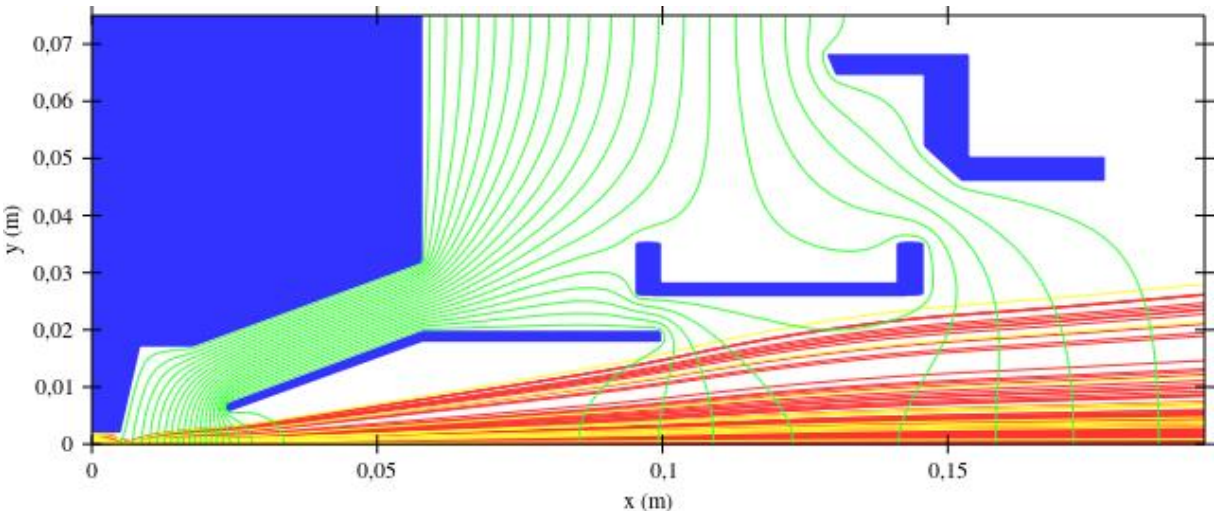

Figure 22: simulation if He^{1+} ions are added to the beam

With both of these future additions to the code this simulation could be used to fully simulate a source. These simulations afterwards could be used as a base for the commission of an ion source. The ion source then would be used as a part of a system which then could be used for multiple application ranging from ion beam analysis to creation of radioisotopes which would be used for medical purposes.

3.10 Analysis of the parameters

From the chapter (number of the first iteration) to the chapter (number for the final iteration) changes made are significant. From the electrode potential to some of the parameters. Few of them would be analyzed in the following subchapter.

The value of the electrode potential was influenced by multiple sources. Another addition which was done and is not present at the papers which the simulation is based on is a small part attached to the puller electrode which is just a few millimeters long and has enhanced the effects of plasma potential within the source. Overall the electrodes were the hardest parameters to optimize since they consist of multiple factors where both their geometries and potentials are closely related to the focus of the beam and they had to be changed together or the results would not have been changed.

Second parameter to be discussed is the value of current density. Since every ion beam differs on which ion it consists. The beam, as it was previously mentioned, is a beam of $^4He^{2+}$ but the reality is that the beam is not uniform. Since to reach the ions of $^4He^{2+}$ the atoms of helium have to be ionized and for them to reach their 2nd ionization they have to go through the 1st ionization. The result of the ionization of the gas is that in the plasma there is a mixture of $^4He^{2+}$ and $^4He^{+}$ ions where when the extraction process starts both are pulled towards the exit and the beam in the extraction system is constructed of majorly $^4He^{+}$ ions while the rest is of $^4He^{2+}$ ions. $^4He^{+}$ disperse in the extraction system while the $^4He^{2+}$ stay and continue to the RFQ. The value of $96\,\frac{A}{m^2}$ is reached when the values of the current density of both helium ions combine where the value for the $^4He^{2+}$ is $16\,\frac{A}{m^2}$ while the value for $^4He^{+}$ ions is $80\,\frac{A}{m^2}$. The total value is the output from the source where it can be seen that most of the current dissipates in the extraction system.

Third parameter for analysis is starting energy of the ion beam. For the case of the simulation of the source conducted in this paper the value is 10 eV. The value for this parameter can range from 1 to 50 eV while the one mostly used for $^4He^{2+}$ ions is 10 eV. This value is reached by multiplying plasma potential and the charge of the beam where the plasma potential for the source is at 20 V. This value of course as well varies from the source itself and the cold electrons have impact on it so that is why the value is not fixed and it can be in range of multiple electronvolts.

Electron temperature is one of the values which depends on the ionization of the beam in this case the value is at 10 eV where it is the temperature of electrons reached in the second ionization of the helium. This value has an impact on the plasma temperature and that itself impacts the beam in its early stages (impacts the starting energy and transverse temperature). The addition of this parameter with its correct value proved to be one of the biggest contributions (with the addition of plasma potential) to the final result where it impacted the beam particles to be matched in a greater percentage to the RFQ, while when this parameter was not included the matching was done with less of a number of particles.

Discussion for other parameters is discussed in different subchapters and it can be summarized as:

- The beam confinement is just a geometrical setup on where the beam starts and the value for it has to meet at least the aperture of the source (in this case 4 mm)
- The value for transverse temperature is dependent on the plasma temperature and in ECR sources it is in range of 0.4 eV to 1 eV and it has a strong impact on emittance while the value of parallel temperature is 0 because it does not affect the beam itself and has no impact on the emittance
- Number of particles is just a number which helps with the time in which the simulation is executed. It does impact it but the values of few thousand particles is sufficient to run the simulation with little to no variety

4 Conclusion

Ion source simulated in the paper (and the steps) are a theoretical calculation with data gathered from different sources. The value of percentage of the particles which match the RFQ acceptance is good and the setup presented within this paper can be used further in ion beam analysis. Of course, the percentage of the particles matching to the RFQ acceptance can be improved if the parameters of electrodes (geometries and voltages) are improved in a sense where they would be gathered from a same source and not a combination of multiple sources.

The simulation was analyzed and constructed using multiple programming languages and programs ranging from C++, Python, Fortran and many other, the geometries were reconstructed using third party software (Adobe Illustrator), transfer matrix was calculated using MS Excel, data was analyzed using Python as a programing language which presents the best way to analyze and process data. IBSimu code was shaped differently to the original with few custom additions. This was necessary to improve the performance and make it easy for the data to be processed and analyzed afterward.

In conclusion the package IBSimu is excellent in simulating ion beams and simulating different extraction systems but from the tutorial code multiple additions are necessary for the code to have realistic beams and functions for beam analysis. Package, on the other hand, is not good at matching to the RFQ where different scripts are used to supplement the lack of it. For this purpose, Python, as one of the best tools for data analysis, is more than enough. Python can be used as well to be an error checker for the package. Another small deficiency of the IBSimu package is the electrode geometries which follow the strict rules of analytical geometry and the data for the shape of electrodes is not easily obtained since the institutions which use the source are not able to provide necessary information due to companies which produce the source are keeping the data for themselves.

The final conclusion regarding the accuracy of the IBSimu simulations looking back on all results from the chapters 3.7 and 3.8 are:

- IBSimu simulations give a very elongated ellipse, which is probably not realistic; their ends are also folded and bended, which leads to an assumption that there are nonlinear effects present and the values for twiss parameters (alpha and beta). Improvements to the twiss parameters and elimination of the nonlinear effects would lead to a larger percentage of particles accepted to the RFQ;

- Despite of that having two solenoids is enough to match the beam to RFQ

5 References

1. Wille K. The Physics of Particle Accelerators An Introduction. Oxford; 2000. p. 315.

2. Bryant PJ. A BRIEF HISTORY AND REVIEW OF ACCELERATORS. 1928;

3. Gurney R, Condon E. Wave mechanics and radioactive disintegration. Nature. 1928;122(3073):(Sep).

4. Cipra BA. An introduction to the Ising model. Am Math Mon. 1987;(Dec).

5. Wangler TP. An Introduction to the Physics of High Energy Accelerators.

6. Wideröe R. Some memories and dreams from the childhood. Europhys News. 1984;(15(2):9-11).

7. Paul W. Early days in the development of accelerators. In: InProc Int Symposium in Honour of Robert R Wilson. Fermilab; 1979. p. 25–688.

8. Janes G, Levy R, Bethe H, Feld B. New type of accelerator for heavy ions. Phys Rev. 1966;(May).

9. Wolf B. Handbook of Ion Sources. Wolf B, editor. CRC Press; 1995. 544 p.

10. Thomae R, Gough R, Keller R, Leung KN, Schenkel T, Aleksandrov A, et al. Beam measurements on the H- source and low energy beam transport system for the Spallation Neutron Source. Rev Sci Instrum. 2002;73(5):2016.

11. Kalvas T, Tarvainen O, Ropponen T, Steczkiewicz O, Ärje J, Clark H. IBSIMU : A three-dimensional simulation software for charged particle opticsa) IBSIMU : A three-dimensional simulation software for charged particle optics a …. 2014;703(2010):2008–11.

12. Heddle DWO. Electrostatic Lens Systems. Electrostatic Lens Systems. CRC Press; 2000.

13. Helmut L. Applied Charged Particle Optics. Springer; 2008. 131 p.

14. Wollnik H. Optics of charged particle. Esevier; 2012.

15. Kumar V. Understanding the focusing of charged particle beams in a solenoid magnetic field. Am J Phys. 2009;77(8):737–41.

16. Stockli MP, Welton RF, Keller R. Self-consistent, unbiased root-mean-square emittance analysis. Rev Sci Instrum. 2004;75(5 PART II):1646–9.

17. Edition S. Charged Particle Optics.

18. Elmaghraby E. Radiation Interaction with Matter. Radiation Synthesis of Materials and Compounds. 2013. 403–421 p.

19. Leitner D. Ion beam properties and their diagnostics for ECR ion source injector systems. In: 14th Beam Instrumentation Workshop. 2010.

20. Ed IGB. The Physics and Technology of Ion Sources Second , Revised and Extended Edition. 2004.

21. Humphries S. Charged particle beams. Courier Corporation; 2013.

22. Humphries S, Russell S, Carlsten B, Earley L, Ferguson P. Circular-to-planar transformations of high-perveance electron beams by asymmetric solenoid lenses. Phys Rev Spec Top - Accel Beams. 2004;7(6):1–10.

23. Chauvin N. Space-charge effect. arXiv Prepr arXiv14107991. 2014;

24. Child CD. Discharge from hot CaO. Phys Rev (Series I). 1911;(32(5):492).

25. CERN. Accelerators [Internet]. 2020 [cited 2020 Sep 12]. Available from: https://home.cern/science/accelerators

26. Bohm D. Minimum ionic kinetic theory for a stable sheath. In: Guthrie A, Wakerling R, editors. Electrical, The Characteristics of Fields, Discharges in Magnetic. McGraw-Hill; 1947.

27. Chapman B, Vossel J. Glow discharge processes: sputtering and plasma etching. Phys Today. 1981;34(7):62.

28. D IW, Instruments KA. The ion optics of low-energy ion beams. 1984;34.

29. Kalvas T, Tarvainen O, Clark H, Brinkley J, Ärje J. Application of 3D code IBSimu for designing an H-/D- extraction system for the Texas A&M facility upgrade. In: AIP Conference Proceedings. American Institute of Physics; 2011. p. 439–48.

30. Kuo T, Yuan D, Jayamanna K, McDonald M, Baartman R, Schmor P, et al. On the development of a 15 mA direct current H− multicusp source. Rev Sci Instrum. 1996;67(3):1314(March).

31. Keller R, Cheng D, DiGennaro R, Gough R, Greer J, Leung K, et al. Ionsource and low-energy beam-transport issues with the front-end systems for the Spallation Neutron Source. Rev Sci Instrum. 2002;73.

32. Midttun, Kalvas T, Kronberger M, Lettry J, Pereira H, Schmitzer C, et al. A new extraction system for the Linac4 H? ion source. Rev Sci Instrum. 2012;83(2).

33. Kalvas T, Welton RF, Tarvainen O, Han BX, Stockli MP. Simulation of H- ion source extraction systems for the Spallation Neutron Source with Ion Beam Simulator. Rev Sci Instrum. 2012;83(2).

34. Stroustrup B. C++ programming language. 1997.

35. Kuhlman D. A Python Book: Beginning Python, Advanced Python, and Python Exercises. 2012.

36. CERN Accelerator Beam Physics Group, \relax CERN Accelerator Beam Physics Group. MAD - Methodical Accelerator Design. 2014.

37. Kalvas T. IBSimu.

38. Kalvas T, Tarvainen O, Ropponen T, Steczkiewicz O, Rje J, Clark H. IBSIMU: A three-dimensional simulation software for charged particle optics. Rev Sci Instrum. 2010;81(2).

39. Pantechnik. Nanogan. 2013.

40. Pantechnik. Supernanogan. 2013.

41. Bencini V, Pommerenke HW, Grudiev A, Lombardi AM. 750 MHz radio frequency quadrupole with trapezoidal vanes for carbon ion therapy. Phys Rev Accel Beams. 2020;23(12).

42. Kalvas T. Vlasov2d tutorial code of IBSimu [Internet]. 2017. Available from: http://ibsimu.sourceforge.net/vlasov2d/index.html

43. Puligundla P, Mok C. Microwave- and radio-frequency- powered cold plasma applications for food safety and preservation [Internet]. Advances in Cold Plasma Applications for Food Safety and Preservation. Elsevier Inc.; 2020. 309–329 p. Available from: http://dx.doi.org/10.1016/B978-0-12-814921-8.00011-6

44. Ranjini K, Nabhiraj PY, Das SK, Mallik C, Bhandari RK. Estimation of electron temperature in 14 . 45 GHz ECR ion source plasma by analysis of Bremsstrahlung spectra. 2007;45(December):965–8.

45. Ciavola G, Gammino S, Celona L, Maimone F, Galatà A, Pullia M, et al. COMMISSIONING OF THE ECR ION SOURCES AT CNAO FACILITY. :415–7.

46. Toivanen V, Kalvas T, Koivisto H, Komppula J, Tarvainen O. Double einzel lens extraction for the JYFL 14 GHz ECR ion source designed with IBSimu. J Instrum. 2013;8(5).

6 Addition

6.1 IBSimu code

```
#include "epot_bicgstabsolver.hpp"
#include "particledatabase.hpp"
#include "geometry.hpp"
#include "func_solid.hpp"
#include "epot_efield.hpp"
#include "meshvectorfield.hpp"
#include "ibsimu.hpp"
#include "error.hpp"
#include "particlediagplotter.hpp"
#include "geomplotter.hpp"
#include "config.h"
#ifdef GTK3
#include "gtkplotter.hpp"
#endif

// physics constants:
double constexpr mass_4He_in_u = 4.002602;
double constexpr e0           = 1.60217662e-19;
double constexpr c0           = 299792458;
using namespace std;

// puller
bool solid1( double x, double y, double z )
{
        float etx1 = 0.02355;  // tilted electrode starting x; originally
: 0.02355
  float etx2 = 0.05794;  // tilted electrode ending x
  float etx3 = 0.09375; // straight part; originally : 0.09375

    return((x >= etx1 - 0.001 && x <= etx1 && y >= 0.005629 && y <=
0.007216)||(x >= etx1 && x <= etx2 && y >= -0.00296 + 0.36*x && y <= -
0.00163 + 0.371*x) || (x >= etx2 && x <= etx3 && y >= 0.0178984 && y <=
0.01986574));
}

// focus electrode
bool solid2( double x, double y, double z )
{
    return( (x >= 0.09525 && x <= 0.099748 && y <= 0.030692 + sqrt
(2.268e-5-0.36*pow((x-0.097367),2.0)) && y >= 0.030692 - sqrt (2.268e-5-
0.36*pow((x-0.097367),2.0))) || (x >= 0.09821 && x <= 0.1429 && y <=
0.0283104 && y >= 0.026) || (x >= 0.1410229 && x <= 0.1457854 && y <=
0.030692 + sqrt (2.268e-5 - 0.36*pow((x-0.143404165),2.0)) && y >=
0.030692 - sqrt (2.268e-5 - 0.36*pow((x-0.143404165),2.0))));
}

// ground
bool solid3( double x, double y, double z )
{
    return(( x >= 0.13017 && x <= 0.1537229 && y <= 0.06826 && y >=
0.06456)       ||      ( x >= 0.12859 && x <= 0.13017 && y >= -2.33 * x +
0.36830 && y <= 0.06826)       ||      (x >= 0.145785 && x<= 0.1537229 &&
y>= 0.0521229 && y <= 0.0645583)       || (x>=0.145785 && x<=
0.1523999981 && y <= 0.0521229 && y>= -0.92 * x + 0.1862455) || (x >=
0.1537229 && x<= 0.1775354 && y >= 0.0460374 && y <= 0.0502708333) || (x
>= 0.1523999981 && x <= 0.1537229 && y >= 0.0460374 && y <=0.0521229));}
```

```
// plasma
bool solid4( double x, double y, double z)
{
        return((x <= 0.00503 && y >= 0.01)||(x >= 0.00503 && x <= 0.02997 && y >=
1.605 * x - 0.00600625) || ( x >= 0.00847 && x <= 0.01773 && y >= 0.017056769)
|| ( x >= 0.01173 && x <= 0.05794 && y >= 0.3653 * x + 0.01058));
}

//all the +0.0025 or 0.0015 are newly added

void simu( int *argc, char ***argv )
{
    // size, origin, mesh size
    Geometry geom( MODE_2D, Int3D(403,151,1), Vec3D(0,0,0), 0.0005 );


    Solid *s1 = new FuncSolid( solid1 );
    geom.set_solid( 7, s1 );
    Solid *s2 = new FuncSolid( solid2 );
    geom.set_solid( 8, s2 );
    Solid *s3 = new FuncSolid( solid3 );
    geom.set_solid( 9, s3 );
    Solid *s4 = new FuncSolid( solid4 );
    geom.set_solid( 10, s4);

    float fHV=-25.5e3;
    geom.set_boundary( 1,Bound(BOUND_NEUMANN,  0.0e3) ); //xmin
    geom.set_boundary( 2,Bound(BOUND_NEUMANN,  0.0e3) );//xmax
    geom.set_boundary( 3,Bound(BOUND_NEUMANN,     0.0  ) );//ymin
    geom.set_boundary( 4,Bound(BOUND_NEUMANN,     0.0  ) );//ymax
    geom.set_boundary( 7,Bound(BOUND_DIRICHLET,  -23.0e3) );// puller
    geom.set_boundary( 8,Bound(BOUND_DIRICHLET, fHV) );//focus electrode
    geom.set_boundary( 9,Bound(BOUND_DIRICHLET, -24.0e3) );//ground electrode
    geom.set_boundary(10,Bound(BOUND_DIRICHLET, 0.0e3) );//body of the source
    geom.build_mesh();
    // NEUMANN: dEf/dx
    // DIRICHLET


    EpotBiCGSTABSolver solver( geom );
                InitialPlasma init_plasma( AXIS_X, 0.513e-3 );
                //plasma potential [V]:
                double Up=20.0;
                solver.set_initial_plasma( Up, &init_plasma );
                //electron temperature in plasma
                double Te=10.0;

                EpotField epot( geom );
    MeshScalarField scharge( geom ); // space-charge electric field
    MeshVectorField bfield;
    EpotEfield efield( epot );
    field_extrpl_e efldextrpl[6] = { FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE,
                                     FIELD_SYMMETRIC_POTENTIAL, FIELD_EXTRAPOLATE,
                                     FIELD_EXTRAPOLATE, FIELD_EXTRAPOLATE };
    efield.set_extrapolation( efldextrpl );

    ParticleDataBase2D pdb( geom );
    bool pmirror[6] = { false, false, true, false, false, false };
    pdb.set_mirror( pmirror );
```

```
    for( size_t i = 0; i < 20; i++ ) {
                        if( i == 1 ) {
                  double rhoe = pdb.get_rhosum();
                  solver.set_pexp_plasma( -rhoe, Te, Up );
              }
          solver.solve( epot, scharge );
          efield.recalculate();
          pdb.clear();



// Helium 2+
// arguments for particle generation function:
// - No of particles,
// - beam current density [A/m2], 80 (He+), 96 (He+ and He2+)
// - particle charge [e],
// - mass [u]
// - mean energy E (eV), eg. 10 eV (plasma potential*charge, can be 1-50 eV)
(25)
// - parallel temperature [eV] Tp = 0.0 eV (0.1)
// - transverse temperature [eV] Tt = 1 eV (28) - has large impact on emittance
                        pdb.add_2d_beam_with_energy( 5000, 96.0, 2.0,
mass_4He_in_u,
                                        10.0, 0.0, 1.0,
                                        0.0, 0.0,
                                        0.0, 0.01 );
          pdb.iterate_trajectories( scharge, efield, bfield );
    }

// Write output file containing all particles
    std::ofstream fileOut( "particles_out.txt" );
    for( size_t k = 0; k < pdb.size(); k++ ) {
      Particle2D &pp = pdb.particle( k );
      // Plot particle I, m, coordinates, velocities
      fileOut << std::setw(12) << pp.IQ() << " ";
      fileOut << std::setw(12) << pp.m() << " ";
      fileOut << std::setw(12) << pp.location() << " ";
      fileOut << std::setw(12) << pp.velocity() << " ";
      fileOut << "\n";
}
fileOut.close();

// calculate emittance:
                TrajectoryDiagnosticData tdata;
                std::vector<trajectory_diagnostic_e> diagnostics;
                diagnostics.push_back( DIAG_Y );
                diagnostics.push_back( DIAG_YP );
//pdb.trajectories_at_plane( tdata, AXIS_X, geom.max(0)-geom.h(), diagnostics );
                        pdb.trajectories_at_plane( tdata, AXIS_X, geom.max(0),
diagnostics );
                        Emittance emit( tdata(0).data(), tdata(1).data() );

                        // Output, append
                        ofstream dout( "emit.txt", ios_base::app );
                        dout << emit.alpha() << " "
                                        << emit.beta() << " "
                                        << emit.epsilon() << "\n";
                        dout.close();
```

```
#ifdef GTK3
    GTKPlotter plotter( argc, argv );
    plotter.set_geometry( &geom );
    plotter.set_epot( &epot );
    plotter.set_scharge( &scharge );
    plotter.set_particledatabase( &pdb );
    plotter.new_geometry_plot_window();
    plotter.run();
#endif
}

int main( int argc, char **argv )
{

        int i;
        for(i=0;i<argc;i++) {printf("arguments in main: %s\n",argv[i]);}

    try {
        ibsimu.set_message_threshold( MSG_VERBOSE, 1 );
        ibsimu.set_thread_count( 2 );
      simu(&argc, &argv); // with graphics
                        //simu(argc, argv); // no graphics
    } catch ( Error e ) {
        e.print_error_message( ibsimu.message( 0 ) );
        exit( 1 );
    }

    return( 0 );
}
```

6.2 Python script

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np from math
import pi,atan2
%matplotlib inline
```

```
df = pd.read_csv("particles_out.txt", delim_whitespace=True, header=None)
df.columns=["iq","m","x","y","z","vx","vy","vz"]
df['tang']=[1000*atan2(vy,vx) for vy,vx in zip(df['vy'],df['vx'])] # better
df['tpos']=df["y"]*1000 # convert position to [mm]
df_filtered = df[df['x'] == 0.201]
df_filtered.head()
df.columns=["iq","m","x","y","z","vx","vy","vz"]
df['tang']=[1000*atan2(vy,vx) for vy,vx in zip(df['vy'],df['vx'])]
df['tpos']=df["y"]*1000 # convert position to [mm]
```

```
accRFQ=0.2 # [pi*mm*mrad] normalized, at 15 keV/u
betrel=0.005675 # 15 keV/u
accRFQ=accRFQ/betrel #print(accRFQ)
alphRFQ=0.3 # []
betaRFQ=0.01 # [mm/mrad]
```

```
# helper functions to plot the RFQ acceptance ellipse:
# calculate ellipse tilt angle
# in radians
def calc_theta(a,b,g):
    #return 0.5*np.arctan(-2*a/a(b,g)b-g))
    return 0.5*atan2(-2*a,(b-g))


## calculate semi-major and semi-minor axes
def calc_axes(e,a,b,g):
    h=0.5*(b+g)
    majax=np.sqrt(e/2)*(np.sqrt(h+1)+np.sqrt(h-1))
    minax=np.sqrt(e/2)*(np.sqrt(h+1)-np.sqrt(h-1))
    return majax,minax

# plot the ellipse using parameteric plot
def get_ellipse(emm,alpha,beta,gamma):
    t=np.arange(0,6.293185,0.1) # returns evenly spaced values (for 0.1) within
a given interval [0-2pi].
    ang=calc_theta(alpha,beta,gamma)
    #print("ellipse tilt angle is [rad]: ",ang)
    majax,minax=calc_axes(emm,alpha,beta,gamma)
    #print('ellipse axes are: ', majax, minax)
    #print('theta is the rotation angle of the ellipse that we put in at the
begining', angdeg)
    x1=[majax*np.cos(t)*np.cos(ang)-minax*np.sin(t)*np.sin(ang)][0]
    xp1=[majax*np.cos(t)*np.sin(ang)+minax*np.sin(t)*np.cos(ang)][0]
    #plt.plot(x1,xp1,'o')
    return x1,xp1
```

```
xel,xpel=get_ellipse(accRFQ,alphRFQ,betaRFQ,gamRFQ)

# mirror the particle distribution
ypos=[] # negative x,x'
yvel=[]
for y,yp in zip(df_filtered["tpos"],df_filtered["tang"]):
        ypos.append(y)
        ypos.append(-y)
        yvel.append(yp)
        yvel.append(-yp)

plt.hist2d(ypos,yvel,bins=(50,50))
plt.xlabel('transverse position [mm]')
plt.ylabel('angle [mrad]')
```

```
plt.plot(ypos,yvel,'o', label='beam')
import matplotlib.colors as mcolors
colors = [(1,0,0,c) for c in np.linspace(0,1,100)]
cmapred = mcolors.LinearSegmentedColormap.from_list('mycmap', colors, N=256
plt.hist2d(ypos,yvel, bins=50, cmap=cmapred, zorder=7

# RFQ acceptance ellipse:
plt.plot(xel,xpel,'r-', label='RFQ acceptance') # axes labels
plt.xlabel('position [mm]')
plt.ylabel('angle [mrad]')
plt.grid()
plt.xlim(-17,17)
plt.ylim(-75,75)
plt.legend()
```

```
# function:
def stemitt(x,xp,npart):
xm2=np.sum(pow(x,2))/npart - pow(np.sum(x)/npart,2
xpm2=np.sum(xp*xp)/npart - pow(np.sum(xp)/npart,2
xxpm=np.sum(x*xp)/npart - (np.sum(x)*np.sum(xp))/pow(npart,2)
#print('mixed term=',xxpm)
rmsemitt=pow(xm2*xpm2-pow(xxpm,2),0.5)
#print('emittance=',rmsemitt)
return rmsemitt
print(type(ypos))
em=stemitt(np.asarray(ypos),np.asarray(yvel),len(ypos)) print('{}
[mm*mrad]'.format(em))

npart=len(ypos)
ypos=np.asarray(ypos)
yvel=np.asarray(yvel)
beta=(np.sum(pow(ypos,2))/npart - pow(np.sum(ypos)/npart,2))/em
gama=(np.sum(pow(yvel,2))/npart - pow(np.sum(yvel)/npart,2))/em
alpha = -(np.sum(ypos*yvel)/npart - (np.sum(ypos)*np.sum(yvel))/pow(npart,2))/em
print(alpha,beta,gama)

# calculate normalized emittance
for vx in zip (df_filtered["vx"]):
 a=df_filtered["vx"]

c=299792458
e=2.94879*pow(10,-5)
avgv=np.sum(a)/len(a)
relbeta=avgv/c
relgama=1/np.sqrt(1-pow((avgv/c),2))
print ('relativistic parameters =', relbeta, relgama, relbeta*relgama)
normE=relgama*relbeta*e
print ("normalized emittance = {} [mm*mrad]".format(normE))

#Transfer matrix
TransfMatrix = np.array([[-0.1506867438,0.02430629197,-
0.7994759258,0.1289582267],[-0.5661202565,-0.1363516383,-3.003578847,-
0.7234203191],[0.7994759258,-0.1289582267,-
0.1506867438,0.02430629197],[3.00357884,0.7234203191,-0.5661202565,-
0.1363516383]])
print ("matrix is")
print (TransfMatrix)
for x in zip(df_filtered["tpos"]):
    b = df_filtered["tpos"]
for y in zip (df_filtered["tang"]):
    a = df_filtered["tang"]
print (a)
print (b)
```

```
ypos1=[]
yang1=[]
for j in range(len(a)): posVec = [[b[j]],[a[j]],[b[j]],[a[j]]]
newPosVec = np.array
[[TransfMatrix[0,0]*b[j]+TransfMatrix[0,1]*a[j]+TransfMatrix[0,2]*b[j]+TransfMat
rix[0,3]*a[j]],[TransfMatrix[1,0]*b[j]+TransfMatrix[1,1]*a[j]+TransfMatrix[1,2]*
b[j]+TransfMatrix[1,3]*a[j]],[TransfMatrix[2,0]*b[j]+TransfMatrix[2,1]*a[j]+Tran
sfMatrix[2,2]*b[j]+TransfMatrix[2,3]*a[j]],[TransfMatrix[3,0]*b[j]+TransfMatrix[
3,1]*a[j]+TransfMatrix[3,2]*b[j]+TransfMatrix[3,3]*a[j]]])
      ypos1.append
      (TransfMatrix[0,0]*b[j]+TransfMatrix[0,1]*a[j]+TransfMatrix[0,2]*b[j]+Tran
      sfMatrix[0,3]*a[j])
      ypos1.append (-
      (TransfMatrix[0,0]*b[j]+TransfMatrix[0,1]*a[j]+TransfMatrix[0,2]*b[j]+Tran
      sfMatrix[0,3]*a[j]))
      yang1.append
      (TransfMatrix[1,0]*b[j]+TransfMatrix[1,1]*a[j]+TransfMatrix[1,2]*b[j]+Tran
      sfMatrix[1,3]*a[j])
      yang1.append (-
      (TransfMatrix[1,0]*b[j]+TransfMatrix[1,1]*a[j]+TransfMatrix[1,2]*b[j]+Tran
      sfMatrix[1,3]*a[j]))

plt.hist2d(ypos1,yang1,bins=(50,50))
plt.xlabel('after transfer matrix transverse position [mm]')
plt.ylabel('after transfer matrix angle [mrad]')
plt.plot(xel,xpel,'r-', label='RFQ acceptance')
plt.plot(ypos1,yang1, 'o')

# compute percentage of the beam particles inside the acceptance ellipse
pacc=[]
aacc=[]
for x,y in zip(ypos1,yang1):
 tiltang=calc_theta(alphRFQ,betaRFQ,gamRFQ)
 a,b = calc_axes(accRFQ,alphRFQ,betaRFQ,gamRFQ)
 comp1=pow(x*np.cos(tiltang)+y*np.sin(tiltang),2)/pow(a,2)
 comp2=pow(x*np.sin(tiltang)+y*np.cos(tiltang),2)/pow(b,2)
 if comp1+comp2<1.0:
        pacc.append(x)
        aacc.append(y)

plt.plot(xel,xpel,'r-', label='RFQ acceptance')
plt.plot(pacc,aacc,'o')
print("fraction of accepted particles:",len(pacc)/len(ypos1))
```