# Python Simulation of Linear and Non-Linear Accelerator Elements

Oscar Elmqvist Sandvik

July 27, 2016

**Abstract**

For this thesis I have programmed and tested a particle simulator. It was coded in the Python programming language. In this simulator, both linear and non-linear elements can be used. The design philosophy was to make the program object-oriented for easy addition of new element types with a modular result. The Python language allowed for a smooth implementation and kept the code clear. In order to benchmark and test the code's correctness, its results have been compared with the results of TraceWin, a widely used particle-tracking package, with good agreement for the essential parts.

# Contents

# 1 Introduction

Particle accelerators have become well-established in both research and industry. The vast range of experiments using accelerators have provided us with a deep understanding of nature. For every generation that is built, the machines get bigger, more complex and more expensive. Software for simulating particle dynamics are used to improve and better understand accelerators. Particle simulators are not only useful as design tools but also during operation where they play an integral part in modern control systems. With these arguments for the utility of simulators, my thesis will delve into the theory and programming behind the Differential-Algebra-Tracker code (DAT-code) that I have made and tested for this thesis. The DAT-code was coded under the GNU General Public License version 3 [1], and has been made freely available on my Github repository: `https://github.com/OscarES/Differential-Algebra-Tracker` The code that I have written will be used for non-linear simulations at ESS and parts of it will be used in the "ESS Linac Simulator" program. The inspiration for this thesis stems from a summer project that I did during 2015 where I was introduced to non-linear particle simulations. That project's code is available on `https://github.com/OscarES/lie` and parts of it has been used in this thesis. The work made during this thesis can be used to simulate accelerators made with both linear and non-linear elements. Because of the open-source license it possible to use the code when coding other particle simulators. The object-oriented programming design makes it easy to extend the software with new accelerator elements. Since the user interface is abstracted away from the simulation code it is easy to write a new interface. Most importantly the software that I have written is both a robust and advanced particle simulator which works when compared against a popular proprietary particle simulator.

# 2 Theory

This section discusses accelerator physics starting with the Lorentz force equation and then introduces both the linear and non-linear dynamics that it creates. The linear and non-linear treatments are based on Wolski's book "Beam Dynamics in High Energy Particle Accelerators" [2].

## 2.1 Hamiltonian dynamics

A particle in an accelerator has properties such as mass, charge and energy. The particle's position is determined by the coordinates $x$, $y$ and $z$. Its momentum has the components $p_x$, $p_y$ and $p_z$. The time it has spent traveling through the accelerator is denoted by $t$.

A function called the Hamiltonian determines the dynamics of the particle and is denoted $H(x, p_x, y, p_y, z, p_z; t)$. The Hamiltonian determines the dynamics

through Hamilton's equations. For the $x$ axis they are

$$\frac{dx}{dt} = \frac{\partial H}{\partial p_x}, \tag{1}$$

$$\frac{dp_x}{dt} = -\frac{\partial H}{\partial x}. \tag{2}$$

Coordinates and momenta that evolve according to Hamilton's equations are called canonical variables. Since it is very hard to keep track of the time for each particle along an accelerator it is more desirable to keep track of the longitudinal coordinate along the trajectory of the reference particle. The reference particle is the particle that perfectly follows the trajectory specified by the accelerator design. A coordinate $s$ is defined as the distance travelled along the trajectory of the reference particle with the beginning of the accelerator as origin. The coordinate $s$ allows us to rewrite Hamilton's equations into

$$\frac{dx}{ds} = \frac{\partial H}{\partial p_x}, \tag{3}$$

$$\frac{dp_x}{ds} = -\frac{\partial H}{\partial x}. \tag{4}$$

A particle in an accelerator is affected by the Lorentz force

$$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}), \tag{5}$$

where $q$ is the charge of the particle, $\mathbf{E}$ is the electric field at the particle's position, $\mathbf{v}$ is the velocity of the particle and $\mathbf{B}$ is the magnetic field. This force determines how the Hamiltonian, $H$, will look. The calculations from the Lorentz force into a useful Hamiltonian are rather long and can be found in chapter 2.2 in [2]. During the calculations the coordinates and momenta are transformed into a new set of canonical variables. These new variables are

$$\begin{pmatrix} x \\ x' \\ y \\ y' \\ z \\ \delta \\ s \end{pmatrix}, \tag{6}$$

where $s$ was defined previously, $\delta$ is a particle's energy deviation from the reference particle defined as

$$\delta \equiv \frac{E}{cP_0} - \frac{1}{\beta_0}, \tag{7}$$

where $E$ is the kinetic energy of a particle, $c$ is the speed of light, $P_0$ is the reference particle's total momentum and $\beta_0$ is the speed of the reference particle divided by $c$. The transversal coordinates $x$ and $y$ are how far away from the reference particle the particle is in their respective dimensions. The $y$ axis is defined as being in the opposite direction of gravity and the $x$ axis is perpendicular to both $y$ and $s$ in a right hand sided system. The new transversal "momenta"

$x'$ and $y'$ are the original momenta divided by the reference particle's total momentum $P_0$. The coordinate $z$ is the longitudinal advance compared to the reference particle which is tangential to $s$ and it is defined as

$$z \equiv \frac{s}{\beta_0} - ct, \tag{8}$$

where the speed of the reference particle is included in $\beta_0$ and $c$ is just the speed of light in vacuum. The coordinates $x$, $y$ and $z$ are measured in meters. The new "momenta" are unitless.

After calculations found in chapter 2.2 in [2] the following Hamiltonian is acquired from the Lorentz force

$$H = \frac{\delta}{\beta_0} - \sqrt{(\delta + \frac{1}{\beta_0} - \frac{q\phi}{cP_0})^2 - (x' - a_x)^2 - (y' - a_y)^2 - \frac{1}{\beta_0^2\gamma_0^2}} - a_z, \tag{9}$$

where $\phi$ is the scalar potential. The parameter $P_0$ is the total momentum of the reference particle. The parameters $a_x$, $a_y$ and $a_z$ are the components of the vector potential

$$\mathbf{a} = \frac{q}{P_0}\mathbf{A}, \tag{10}$$

where $\mathbf{A}$ together with $\phi$ give rise to the electric and magnetic fields

$$\mathbf{E} = -\nabla\phi - \frac{\partial \mathbf{A}}{\partial t}, \tag{11}$$

$$\mathbf{B} = \nabla \times \mathbf{A}. \tag{12}$$

Last but not least $\gamma_0$ is defined as

$$\gamma_0 = \frac{1}{\sqrt{1 - \beta_0^2}} \tag{13}$$

Different elements of an accelerator have different potentials $\phi$ and $\mathbf{A}$ and when they are known the particle dynamics can be calculated.

## 2.2 Transfer matrices

Given equation 9 together with the electric and magnetic fields in an accelerator element, we can calculate the particle dynamics inside the element. For linear elements, where the forces in the accelerator elements are linear, the canonical variables after an element can be determined.

Starting with the simplest element, the drift element, we can first calculate the particle dynamics and then solve the equations of motion. In a drift element both $\phi$ and all the components of $\mathbf{a}$ are 0. This means that the Hamiltonian will be

$$H = \frac{\delta}{\beta_0} - \sqrt{(\delta + \frac{1}{\beta_0})^2 - x'^2 - y'^2 - \frac{1}{\beta_0^2\gamma_0^2}}. \tag{14}$$

By making the paraxial approximation, which is that power series expansions of a Hamiltonian can be truncated at low orders for the transversal momenta, we can expand equation 14 to second order

$$H = -1 + \frac{x'^2}{2} + \frac{y'^2}{2} + \frac{\delta^2}{2\beta_0^2\gamma_0^2} + O(3). \tag{15}$$

The paraxial approximation is valid because the transversal momenta are much smaller than the longitudinal momentum ($x'$ and $y'$ are small), which is the case in almost every point of an accelerator. By dropping constant terms, which won't affect the dynamics because the dynamics come from differentiating the Hamiltonian, and terms of order three or higher we end up with

$$H = \frac{x'^2}{2} + \frac{y'^2}{2} + \frac{\delta^2}{2\beta_0^2\gamma_0^2}. \tag{16}$$

By using this Hamiltonian in Hamilton's equations we get the following dynamics

$$\frac{dx}{ds} = \frac{\partial H}{\partial x'} = x', \tag{17}$$

$$\frac{dx'}{ds} = -\frac{\partial H}{\partial x} = 0, \tag{18}$$

$$\frac{dy}{ds} = \frac{\partial H}{\partial y'} = y', \tag{19}$$

$$\frac{dy'}{ds} = -\frac{\partial H}{\partial y} = 0, \tag{20}$$

$$\frac{dz}{ds} = \frac{\partial H}{\partial \delta} = \frac{\delta}{\beta_0^2\gamma_0^2}, \tag{21}$$

$$\frac{d\delta}{ds} = -\frac{\partial H}{\partial z} = 0. \tag{22}$$

The drift element has the length $L$ along the $s$ axis. Given the starting point of the drift element, denoted with the subscript 0, and the end point of the drift element, denoted with the subscript 1, the equations of motion can be solved. The solutions are

$$x_1 = x_0 + Lx'_0, \tag{23}$$

$$x'_1 = x'_0, \tag{24}$$

$$y_1 = y_0 + Ly'_0, \tag{25}$$

$$y'_1 = y'_0, \tag{26}$$

$$z_1 = z_0 + \frac{L}{\beta_0^2\gamma_0^2}\delta_0, \tag{27}$$

$$\delta_1 = \delta_0. \tag{28}$$

Writing the six canonical variables in vector form makes it possible to construct a transfer matrix, $M$, which evolves variables at the start of an element into variables at the end of an element. For the drift element, the evolution will be

$$\begin{pmatrix} x_1 \\ x_1' \\ y_1 \\ y_1' \\ z_1 \\ \delta_1 \end{pmatrix} = M_{\text{drift}} \begin{pmatrix} x_0 \\ x_0' \\ y_0 \\ y_0' \\ z_0 \\ \delta_0 \end{pmatrix} = \begin{bmatrix} 1 & L & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & L & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \frac{L}{\beta_0^2\gamma_0^2} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_0 \\ x_0' \\ y_0 \\ y_0' \\ z_0 \\ \delta_0 \end{pmatrix}. \tag{29}$$

Particles will drift away from the trajectory of the reference particle in drift elements. Therefore focusing of the particles is needed. To do this quadrupole elements are used. A quadrupole element is a magnet element where there are four magnetic poles which produces a force which grows linearly with the transversal position deviations from the reference particle. It does not contain an electric field and thus once again the scalar potential $\phi$ can be set to zero. The magnet structure creates a magnetic field scaled with charge and the reference particle's total momentum

$$\mathbf{b} = \frac{q}{P_0}\mathbf{B} = (ky, kx, 0) \tag{30}$$

where $k$ is the strength of the magnetic field. The vector potential for this field is

$$\mathbf{a} = (0, 0, -\frac{k}{2}(x^2 - y^2)). \tag{31}$$

The Hamiltonian for the quadrupole thus becomes

$$H = \frac{\delta}{\beta_0} - \sqrt{(\delta + \frac{1}{\beta_0})^2 - x'^2 - y'^2 - \frac{1}{\beta_0^2\gamma_0^2}} + \frac{k}{2}(x^2 - y^2). \tag{32}$$

By doing the paraxial approximation it becomes

$$H = -1 + \frac{x'^2}{2} + \frac{y'^2}{2} + \frac{\delta^2}{2\beta_0^2\gamma_0^2} + \frac{k}{2}(x^2 - y^2) + O(3). \tag{33}$$

Truncating constant terms and terms with order three or higher it becomes

$$H = \frac{x'^2}{2} + \frac{y'^2}{2} + \frac{\delta^2}{2\beta_0^2\gamma_0^2} + \frac{k}{2}(x^2 - y^2). \tag{34}$$

The equations of motion become

$$\frac{dx}{ds} = \frac{\partial H}{\partial x'} = x', \tag{35}$$

$$\frac{dx'}{ds} = -\frac{\partial H}{\partial x} = -kx, \tag{36}$$

$$\frac{dy}{ds} = \frac{\partial H}{\partial y'} = y', \tag{37}$$

$$\frac{dy'}{ds} = -\frac{\partial H}{\partial y} = ky, \tag{38}$$

$$\frac{dz}{ds} = \frac{\partial H}{\partial \delta} = \frac{\delta}{\beta_0^2 \gamma_0^2}, \tag{39}$$

$$\frac{d\delta}{ds} = -\frac{\partial H}{\partial z} = 0. \tag{40}$$

There are three possible solutions to these equations. If $k$ is 0 the solution is the same as that for the drift element. If $k$ is positive the following transfer matrix represents the solution to the equations

$$M_{\mathrm{QF}} = \begin{bmatrix} \cos(KL) & \sin(KL)/K & 0 & 0 & 0 & 0 \\ -K\sin(KL) & \cos(KL) & 0 & 0 & 0 & 0 \\ 0 & 0 & \cosh(KL) & \sinh(KL)/K & 0 & 0 \\ 0 & 0 & K\sinh(KL) & \cosh(KL) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \frac{L}{\beta_0^2\gamma_0^2} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{41}$$

where $K$ is $\sqrt{|k|}$. It has the "QF" subscript meaning quadrupole focusing because it will make $x$ smaller. Unfortunately it will make $y$ larger. In $z$ and $\delta$ it will act just as a drift element. If $k$ is negative the solution will instead be

$$M_{\mathrm{DF}} = \begin{bmatrix} \cosh(KL) & \sinh(KL)/K & 0 & 0 & 0 & 0 \\ K\sinh(KL) & \cosh(KL) & 0 & 0 & 0 & 0 \\ 0 & 0 & \cos(KL) & \sin(KL)/K & 0 & 0 \\ 0 & 0 & -K\sin(KL) & \cos(KL) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \frac{L}{\beta_0^2\gamma_0^2} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{42}$$

Here it will defocus in $x$, and focus in $y$. Often the magnetic strength is measured by the field gradient $G$ [T/m]. The following equation calculates $K$ from $G$

$$K = \begin{cases} -\sqrt{|\frac{G}{B\rho}|} & \text{if } qG > 0 \\ \sqrt{|\frac{G}{B\rho}|} & \text{if } qG < 0 \end{cases}, \tag{43}$$

where

$$B\rho = \frac{m_0 c \beta_0 \gamma_0}{q}, \tag{44}$$

where $m_0$ is the mass of the particles. This $K$ from $G$ comes from the Transfer Matrices section in [3].

A solution to the problem that quadrupoles only focus in one dimension is to alternate focusing and defocusing quadrupoles in a structure called FODO-lattice. The "F" in FODO stands for focusing quadrupole, the "O" stands for drift and "D" stands for defocusing. The alternation between focusing and defocusing keeps the beam from growing in size. See Chapter 3.13.3 in [4] for a worked example.

By using a similar method of starting with a Hamiltonian and deriving the equations of motion the transfer matrices for dipole magnets and solenoids can be constructed. But if one wants to calculate a transfer matrix for a sextupole (a magnet element with six magnetic poles) this method will discard some of the sextupoles most essential dynamics. Instead another method called "Lie transform" will be used later in this report to construct a "Transfer map" for the sextupole which allows one to calculate the canonical variables at the end of the element. The Lie transform can and will be used to calculate other higher order magnetic elements as well.

## 2.3 Plotting the impact on the beam

When plotting particle parameters phase space plots are commonly used, where in each plot position against momentum along a certain axis are plotted. Two example phase space plots are shown in Figure 1.

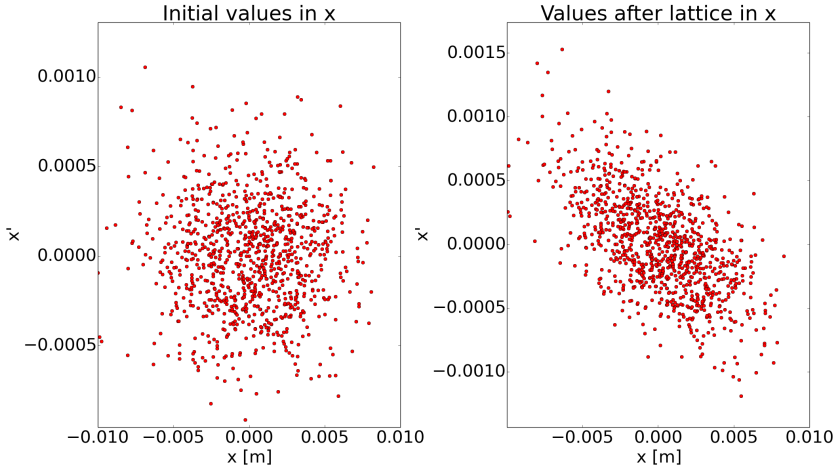


Figure 1: Example of phase space plots for a bunch of 1000 particles along the $x$ axis before and after an accelerator lattice.

## 2.4 Space-Charge

In a particle beam there can be around 10 billion particles. For heavy particles, such as protons which have a speed that is not highly relativistic, the particles will repel each other. This is known as the space-charge effect. This repelling force will cause the particles to spread out in all directions. Thus the effect is three dimensional. It is hard to account for it in simulations as one needs to find a theoretical description of the effect which makes calculation of it for all particles feasible. Lighter particles, such as electrons, with the same energy as heavy particles will have a much higher relativistic gamma value. The current of the particles will create a magnetic field that will attract particles together. This

will compensate for the space-charge effect and one can in most cases neglect the space-charge effect for lighter particles.

Three ways of simulating the space charge can be found in Chapter 12 in [2], Chapter 7.2 and 7.3 in [5] and in [6].

## 2.5 Non-Linear Elements

In Chapter 9 in [2] the full Hamiltonian for the sextupole element is stated. By doing the paraxial approximation and clearing the unnecessary terms it can be written as

$$H_{\text{sextupole}} = \frac{x'^2}{2} + \frac{y'^2}{2} + \frac{k}{6}(x^3 - 3xy^2). \tag{45}$$

This Hamiltonian is just for the variables $x$, $x'$, $y$ and $y'$. For the dynamics of $z$ and $\delta$ one just have to add the term $\frac{\delta^2}{2\beta_0^2\gamma_0}$ to the Hamiltonian. If the Hamiltonian in equation 45 is used directly in Hamilton's equations some of the equations will have a quadratic dependence on $x$ or $y$. This makes it very hard to solve the equations right away. In order to solve this problem the Lie transform will be introduced. What constitutes the core of the Lie transform is the Lie operator.

The Lie operator $: g :$ (also called the Poisson bracket) for any function $g(x_i, p_i)$ is defined as

$$: g :\equiv \sum_{i=1}^{n} \frac{\partial g}{\partial x_i}\frac{\partial}{\partial p_i} - \frac{\partial g}{\partial p_i}\frac{\partial}{\partial x_i}, \tag{46}$$

where the i index is for which dimension ($x_1$ is $x$, $x_2$ is $y$ and $x_3$ is $z$). If $g$ is set to $H$ and one applies the operator on $x$ and $p_x$ ($n = 1$) we get

$$: H : x = -\frac{\partial H}{\partial p_x}, \tag{47}$$

$$: H : p_x = \frac{\partial H}{\partial x}, \tag{48}$$

since

$$\frac{\partial p_i}{\partial x_i} = 0, \frac{\partial x_i}{\partial p_i} = 0, \frac{\partial p_i}{\partial p_i} = 1, \frac{\partial x_i}{\partial x_i} = 1. \tag{49}$$

Also, the cross-dimensional derivatives will be zero. The results for the Lie operator $: H :$ acting on $x$ and $p$ (multiplied by -1 on the right hand side) are the classical Hamilton's equations of motion, which give the derivatives with respect to time

$$\dot{x} = \frac{\partial H}{\partial p}, \tag{50}$$

$$\dot{p} = -\frac{\partial H}{\partial x}. \tag{51}$$

If we write the canonical variables $x$, $x'$, $y$, $y'$, $z$ and $\delta$ as a vector

$$\bar{v} = (x, x', y, y', z, \delta), \tag{52}$$

we can write $:H:$ acting on $\bar{v}$ as

$$\frac{\partial \bar{v}}{\partial t} = -:H:\bar{v}. \tag{53}$$

By doing a transformation that makes $s$ take the place of $t$ the equation becomes

$$\frac{\partial \bar{v}}{\partial s} = -:H:\bar{v}. \tag{54}$$

The solution to this differential equation can be written as

$$\bar{v}(s) = \mathrm{e}^{-s:H:}\bar{v}(0), \tag{55}$$

where the exponential factor is a Taylor expansion similar to the normal Taylor expansion of $\mathrm{e}^{-x}$. This expression is called a Lie transform and for $\mathrm{e}^{:f:}g$ one says "the $f$ Lie transform of $g$". The justification for the Lie transform can be found in Chapter 9 in [2]. The transform in Equation 55 then becomes

$$\bar{v}(s) = \bar{v}(0) - s:H:\bar{v}|_{s=0} + \frac{s^2}{2!}(:H:)^2\bar{v}|_{s=0} - \cdots . \tag{56}$$

At the end of an element of length $L$, the equation will be

$$\bar{v}(L) = \mathrm{e}^{-L:H:}\bar{v}(0) = \bar{v}(0) - L:H:\bar{v}|_{s=0} + \frac{L^2}{2!}(:H:)^2\bar{v}|_{s=0} - \cdots . \tag{57}$$

There is however one problem here: that the Taylor expansion has infinite terms. When should it be truncated? If the alternating terms does not cancel each other out the series could diverge. It was found that truncating after order 5 gives accurate results since the factorial denominator will increase rapidly and that if $L$ is kept less than 1 m it will damp the higher order terms. If one does not mind the extra computational time the more terms in the Taylor expansion the more exact the results will be. Also symplecticity, which will be introduced in Section 2.7, is better conserved when keeping more terms. In Chapter 9.2 in [2] the error for a particle in a sextupole element is calculated to be $k^m L^{m+1} x_i^{2m-1}$, where $m$ is the order after one truncates and $x_i$ is a canonical variable of a particle.

The algebra for calculating a Lie transform becomes rather long and can be left to the Python library Sympy [7] which is used in my program. By truncating at the fourth order the transfer map functions for a sextupole after the Lie transform become

$$x(L) = -\frac{1}{6}kL^3(x_0'x_0 - y_0'y_0) - \frac{1}{4}kL^2(x_0^2 - y_0^2) + Lx_0' + x_0 \tag{58}$$

$$x'(L) = \frac{1}{6}kL^3(\frac{1}{2}kx_0^3 + \frac{1}{2}kx_0y_0^2 - x_0'^2 + y_0'^2) - \frac{1}{2}kL^2(x_0'x_0 - y_0'y_0) - \frac{1}{2}kL(x_0^2 - y_0^2) + x_0' \tag{59}$$

$$y(L) = \frac{1}{6}kL^3(x_0'y_0 + y_0'x_0) + \frac{1}{2}kL^2x_0y_0 + Ly_0' + y_0 \tag{60}$$

$$y'(L) = \frac{1}{6}kL^3(\frac{1}{2}kx_0^2y_0 + \frac{1}{2}ky_0^3 + 2x_0'y_0) + \frac{1}{2}kL^2(x_0'y_0 + y_0'x_0) + kLx_0y_0 + y_0' \tag{61}$$

$$z(L) = z_0 \tag{62}$$

$$\delta(L) = \delta_0 \tag{63}$$

## 2.6 Radiofrequency Cavities

Radiofrequency cavities (rf-cavities) are used to increase a beam's energy and thus the velocities of the particles. They accomplish this by maintaining an externally created electric field which accelerates the particles when the field has the correct phase. Rf-cavities are made of several cylindrical cells and each cell is one half wavelength long. As a particle exits a cell and enters the next, the second cell's field switches from being de-accelerating to accelerating for the incoming particle. This places a huge demand on the timing of the particles. The theory for the radiofrequency cavities in my program are based on Chapter 3.6 in [2].

The Hamiltonian for a rf-cavity is

$$H = \frac{p_x^2}{2} + \frac{p_y^2}{2} + \frac{\delta^2}{2\beta_0^2\gamma_0^2} + \frac{\alpha}{4\pi}\cos(\phi_0)k^2(x^2+y^2) - \frac{\alpha}{\pi}\sin(\phi_0)kz + \frac{\alpha}{2\pi}\cos(\phi_0)k^2z^2, \tag{64}$$

where $\phi_0$ is the phase of the electric field when the particles enter the cavity. $k$ is the wavenumber. Due to the boundary conditions imposed on $k$, it was calculated to be

$$k = \frac{p_{01}}{a}, \tag{65}$$

where $a$ is the radius of the cavity and $p_{01}$ is the point where the Bessel function $J_0$ has its first zero ($p_{01} \approx 2.405$). The parameter $\alpha$ in Equation 64 is defined as

$$\alpha = \frac{qV_0}{P_0 c}, \tag{66}$$

where $V_0$ is the cavity voltage defined by

$$V_0 = LE_0T. \tag{67}$$

Here, $L$ is the length of the cell, $E_0$ is the amplitude of the electric field and $T$ is the "Transit-Time Factor" (TTF). The TTF takes into account that the electric field will change while the particles are travelling through the cell, and is defined as

$$T = \frac{2\pi\beta_0}{k^2L^2}\sin(\frac{kL}{2\beta_0}). \tag{68}$$

By entering the Hamiltonian from Equation 64 into Hamilton's equations (see Equations 50 and 51) and solving them one will obtain a linear transfer map

$$\bar{x}(L) = M_{\mathrm{rf}}\bar{x}_0 + \bar{m}_{\mathrm{rf}}, \tag{69}$$

where $M_{\mathrm{rf}}$ is

$$M_{\mathrm{rf}} = \begin{bmatrix} c_\perp & s_\perp & 0 & 0 & 0 & 0 \\ -\omega_\perp^2 s_\perp & c_\perp & 0 & 0 & 0 & 0 \\ 0 & 0 & c_\perp & s_\perp & 0 & 0 \\ 0 & 0 & -\omega_\perp^2 s_\perp & c_\perp & 0 & 0 \\ 0 & 0 & 0 & 0 & c_\parallel & \frac{1}{\beta_0^2\gamma_0^2}s_\parallel \\ 0 & 0 & 0 & 0 & -\beta_0^2\gamma_0^2\omega_\parallel^2 s_\parallel & c_\parallel \end{bmatrix} \tag{70}$$

and $\bar{m}_{\mathrm{rf}}$ is

$$\bar{m}_{\mathrm{rf}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ (1-\cos(\omega_{\parallel} L))\frac{\tan(\phi_0)}{k} \\ \beta_0^2\gamma_0^2\omega_{\parallel}\sin(\omega_{\parallel} L)\frac{\tan(\phi_0)}{k} \end{pmatrix}. \tag{71}$$

The transversal and parallel components of $M_{\mathrm{rf}}$ and $\bar{m}_{\mathrm{rf}}$ are

$$c_{\perp} = \cos(\omega_{\perp} L), \tag{72}$$

$$s_{\perp} = \frac{\sin(\omega_{\perp} L)}{\omega_{\perp}}, \tag{73}$$

$$\omega_{\perp} = k\sqrt{\frac{\alpha\cos(\phi_0)}{2\pi}}, \tag{74}$$

$$c_{\parallel} = \cos(\omega_{\parallel} L), \tag{75}$$

$$s_{\parallel} = \frac{\sin(\omega_{\parallel} L)}{\omega_{\parallel}}, \tag{76}$$

$$\omega_{\parallel} = \frac{k}{\beta_0\gamma_0}\sqrt{\frac{\alpha\cos(\phi_0)}{\pi}}. \tag{77}$$

While the particles' $\delta$ will be changed by $M_{\mathrm{rf}}$ and $\bar{m}_{\mathrm{rf}}$, the reference energy and momenta will not be changed. Therefore, the reference energy and momenta are changed after the evaluation inside the rf-cavity. The $\delta$ of the particles should be once again centered around 0 or the point they where initially centered around. However since $x'$ and $y'$ have $P_0$ in their definitions, they need to be rescaled with the quota of the old and new momentum in order to be correct. The coordinates are rescaled by

$$\bar{x}_1 = M_{\Delta P}\bar{x}_0 + \bar{m}_{\Delta P}, \tag{78}$$

where $M_{\Delta P}$ is

$$M_{\Delta P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{P_0}{P_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{P_0}{P_1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{P_0}{P_1} \end{bmatrix} \tag{79}$$

and $\bar{m}_{\Delta P}$ is

$$\bar{m}_{\Delta P} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{\beta_1}(\frac{\gamma_0}{\gamma_1} - 1) \end{pmatrix}. \tag{80}$$

## 2.7 Symplecticity

A transfer matrix $M$ is symplectic if the following equation holds

$$M^T S M = S, \tag{81}$$

where

$$S = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}. \tag{82}$$

This comes from the definition in Section 3.1 in [8]. If the area of the phase space is defined as

$$A = \bar{v}_1^T S \bar{v}_2, \tag{83}$$

it remains constant when $\bar{v}$ is transformed by a symplectic transfer matrix. A quick calculation verifies this

$$(\bar{v}_1^T M^T) S (M\bar{v}_2) = \bar{v}_1^T (M^T S M)\bar{v}_2 = \bar{v}_1^T S \bar{v}_2 = A. \tag{84}$$

The fact that the area of the phase space is constant throughout a symplectic system is called Liouville's theorem. Thus symplecticity means that Liouvilles theorem is respected. For particles which are spread out over the phase space area, Liouville's theorem requires the particle density in the phase space to be constant.

For non-linear systems, such as a sextupole, the definition of the symplectic condition is made by replacing the transfer matrix $M$ in Equation 81 with the Jacobian, $J$, of the system

$$J = I + S\widetilde{H}\delta s, \tag{85}$$

where $\widetilde{H}$ is a 2n x 2n (n is the number of dimensions) matrix with elements

$$\widetilde{H}_{jk} = \frac{\partial^2 H}{\partial v_j \partial v_k}|_{s=s_0}. \tag{86}$$

With this definition of $J$ the non-linear Lie transform of $H$ will be symplectic if [2]

$$J^T S J = S. \tag{87}$$

Since the determinant of $S$ is one it can be shown that if the system is symplectic then the determinant of $J$ is one. Thus checking the determinant of $J$ can be a way to check if the system is symplectic or not. One has to be aware though that a system does not have to be symplectic just because the determinant of $J$ is 1.

## 2.8 Symplectic Integrators

One major drawback of the Lie transform discussed in Section 2.5 is that symplecticity is lost when truncating the polynomial. However if one Lie transforms drift and kinetic terms of the Hamiltonian separately, then the resulting polynomials are not infinite and there is no need to truncate the polynomials and lose symplecticity. If we use the following approximation

$$\mathrm{e}^{-L:H:} = \mathrm{e}^{-L:H_d+H_k:} \approx \mathrm{e}^{-L:H_d:}\mathrm{e}^{-L:H_k:}, \tag{88}$$

then the equations for the particles remain symplectic. One would, in effect, have an element of just the kinetic dynamics followed by a drift element. Since the kinetic elements only hold the dynamics from the magnet, it is physically still applicable over the same length in the accelerator. Thus, even though both the drift and kinetic parts have length $L$, the total length is still $L$. This technique comes from Chapter 10 in [2].

## 2.9 Twiss parameters

So far all the equations above have described how a single particle behaves. When accelerators have millions of particles travelling through them, it can be more interesting to look at the beam as a whole rather than a collection of individual particles. The so called Twiss parameters (also referred to as Courant–Snyder parameters) can be used to describe the beam as a whole. The transversal size of a beam oscillates through the accelerator and the first Twiss parameter that is introduced, $\beta(s)$, is defined as the local amplitude function of the oscillation. For a periodic accelerator, the equations of motion are solved by inserting the trial function

$$x(s) = A\sqrt{\beta(s)}\cos(\Phi(s) + \phi), \tag{89}$$

where $A$ is a constant, $\Phi(s)$ is the phase, and $\phi$ is the initial phase. This beta function determines the size of the beam. The next Twiss parameter, $\alpha(s)$ is defined as

$$\alpha(s) \equiv -\frac{1}{2}\frac{\partial\beta(s)}{\partial s}. \tag{90}$$

The third parameter, $\gamma(s)$, is

$$\gamma(s) \equiv \frac{1+\alpha^2(s)}{\beta(s)}. \tag{91}$$

The three Twiss parameters, together with $x$ and $x'$, define a constant (for energy-conserving elements) called emittance, which has the symbol $\epsilon$, and is defined as

$$\epsilon \equiv \gamma(s)x^2(s) + 2\alpha(s)x(s)x'(s) + \beta(s)x'^2(s). \tag{92}$$

This is the emittance for one particle, the root mean square (rms) beam emittance is defined in Chapter II.5 in [9] as

$$\epsilon_{rms} = \sqrt{\sigma_x^2\sigma_{x'}^2 - \sigma_{xx'}^2}, \tag{93}$$

where $\sigma_x$ is the rms beam width, $\sigma_x'$ is the rms of all particles' $x'$ values and $\sigma_{xx'}$ is the correlation.

The Twiss parameters can be defined in the same way for the other axes. The theory for this section is based on Wille's book Chapter 3.7 and 3.8 [4]. Another introduction to the Twiss parameters can be found in Chapter 4 in [2].

### 2.10 Envelope calculations

The Twiss parameters at the end of an element can be calculated by matrix multiplications in a similar manner as the canonical coordinates. The derivation of how the transfer matrix should look is rather lengthy and can be found in chapter 3.10.2 in [4]. The equation for the new Twiss parameters is

$$\begin{pmatrix} \beta_x \\ \alpha_x \\ \gamma_x \\ \beta_y \\ \alpha_y \\ \gamma_y \\ \beta_z \\ \alpha_z \\ \gamma_z \end{pmatrix} = T \begin{pmatrix} \beta_{x0} \\ \alpha_{x0} \\ \gamma_{x0} \\ \beta_{y0} \\ \alpha_{y0} \\ \gamma_{y0} \\ \beta_{z0} \\ \alpha_{z0} \\ \gamma_{z0} \end{pmatrix} = \begin{bmatrix} T_x & 0 & 0 \\ 0 & T_y & 0 \\ 0 & 0 & T_z \end{bmatrix} \begin{pmatrix} \beta_{x0} \\ \alpha_{x0} \\ \gamma_{x0} \\ \beta_{y0} \\ \alpha_{y0} \\ \gamma_{y0} \\ \beta_{z0} \\ \alpha_{z0} \\ \gamma_{z0} \end{pmatrix}, \tag{94}$$

where the $T$ is a 9 x 9 matrix where the Twiss parameters of each axis only depend on the Twiss parameters in that axis (if the axes are uncoupled), which is why three 3 x 3 submatrices $T_x$, $T_y$ and $T_z$ can be written. The $T_x$ submatrix is defined as

$$T_x = \begin{bmatrix} M_{0,0}^2 & -2M_{0,0}M_{0,1} & M_{0,1}^2 \\ -M_{0,0}M_{1,0} & M_{0,0}M_{1,1} + M_{0,1}M_{1,0} & -M_{0,1}M_{1,1} \\ M_{1,0}^2 & -2M_{1,0}M_{1,1} & M_{1,1}^2 \end{bmatrix} \tag{95}$$

Here $M_{i,j}$ is the matrix element on row $i$ and column $j$ in the corresponding linear element's submatrix for that axis calculating the canonical variables in that axis. The submatrices $T_y$ and $T_z$ are constructed in the same way. Thus every M-matrix with uncoupled axes has a corresponding T-matrix. The purpose of this section and the previous one is that they allow beam parameters to be calculated through linear elements which can be described by matrices. In Section 4.2 these calculations and parameters will be calculated for a beam which goes through a FODO-lattice.

# 3 Simulation Code

In my code I have implemented the elements described above as separate classes. Every element object is placed in data structures called lists. When sending particles through a lattice, the particles are sent through each of the elements and are assigned new positions and momenta at the end of each element. When dealing with the space-charge effect, the elements are split and at each slice the particles get a kick which resembles the space-charge effect.

## 3.1 Overview and Structure

Each implemented element is coded as a class in Python. This means that one element has its own parameters describing how it behaves and how it wants to calculate particles' new positions. In the final phases of programming it was found that the structure of the program is similar to the structure proposed in one of C. K. Allen's papers [10].

Each element class has a function which takes input particles and calculates the particles' exit parameters. The particles are stored as NumPy arrays and all particles are combined into a single NumPy array. The array which holds all particles is thus passed through each of the elements and has its particle data calculated.

The "accelerator.py" file contains all the code for the things mentioned above. This is the core of the program where all the accelerator physics take place. The particle distributions are created in a code named "particleFactory.py". The code "facility.py" provides a handle for user interfaces to interact with the "accelerator.py" code and the other codes. Results are plotted in the "plotting.py" code. The "qtinterface.py" code provides a graphical user interface (GUI) for the user and is the main way of interacting with the simulator. This GUI is shown in Figure 2. There is also a code for saving and loading particles, lattices and other data called "IOHandler.py". Functions doing relativistic calculations are defined in a code "relativity.py". A graph showing the programs' dependencies is shown in Figure 3. An arrow pointing toward a program means that the program at the tail uses the program at the head.

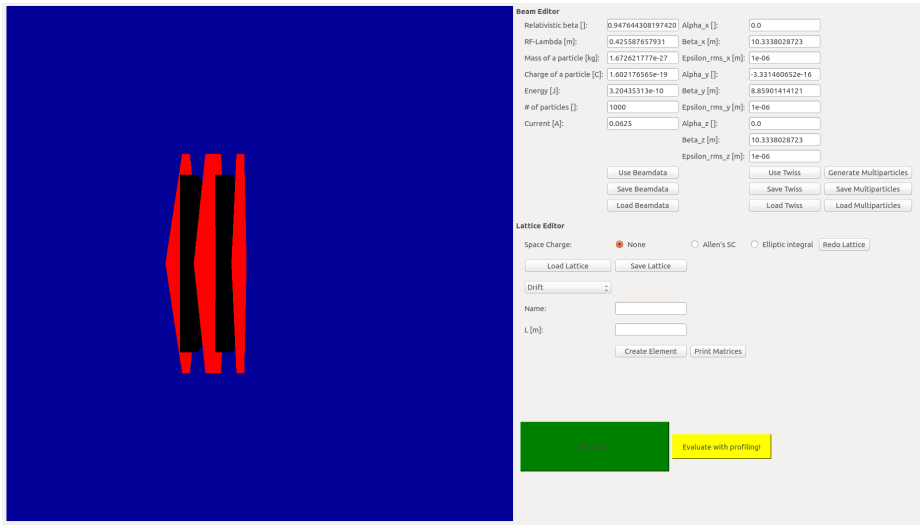


Figure 2: The GUI showing a constructed FODO lattice. The red and black pattern on the left side is a graphical representation of the accelerator lattice. In the top-right corner is a beam editor where a user can set which beam of particles that should be used. Below the beam editor is the lattice editor where a user can construct an accelerator lattice. The green and yellow buttons in the lower-right corner are clicked in order to start the simulation.

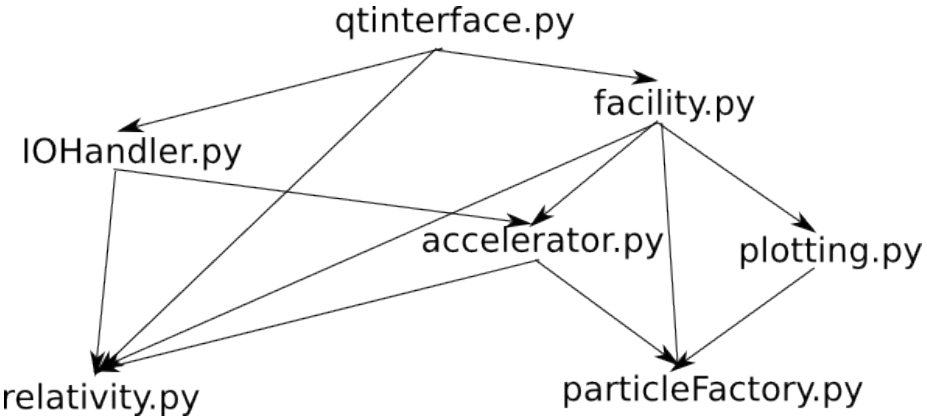


Figure 3: The program structure.

## 3.2 Linear Elements Implementation

For the linear elements, their matrices are constructed for each element. When it comes to evaluation, the canonical variables of $\bar{v}$ are multiplied by the matrix. For $s$ the length of the element is just added to its current value.

## 3.3 Non-Linear Elements Implementation

For the non-linear elements, their Hamiltonians are used in Lie transforms which have been implemented using the Sympy Python library [7]. When the Lie transform is done the transfer map functions are tested to see that they are close to full symplecticity by checking that the difference between the the determinant of $J$ and 1 is less than $10^{-6}$. The result of the symplecticity check is printed. After the symplecticity check the transfer map functions are lambdified, which is a Sympy tool that makes evaluation of functions much faster.

## 3.4 Symplectic Integrator Implementation

Since the symplectic integrator technique is based on the Lie transform approach it was easy to implement. Non-linear Hamiltonians but without the drift element terms in them were created for each non-linear element. Elements were then created with these Hamiltonians according to the previous section. After an element was created with one of these Hamiltonians a drift element would be created with the same length as that of the non-linear element. In this way the drift and kinetic parts were split. Another way of doing the symplectic integrator approach was to start with half the drift element, then the Hamiltonian element and then a second drift element.

## 3.5 Space Charge

The elements are split up in multiple slices. This is because the dynamics of space charge change over very small distances and therefore require small displacements between each evaluation. In my code the user can set how many slices there are per element. I tried using 1000 slices per element since if more were used the program would become too slow. Between each slice the particle were "kicked" by a space-charge matrix. The space-charge matrix is calculated based on the particles before the kick is applied. The form of the matrix depends on which space charge implementation was used.

## 3.6 RF-cavity

The matrices and vectors from Section 2.6 are calculated. When the particle parameters are evaluated they are multiplied and added with these matrices and vectors. The new beam energy and beam relativistic beta is added to a list called "beamdata" in the code.

### 3.7 Graphical User Interface (GUI)

The GUI was coded in the Python library PyQt5 [11] which is the Python implementation of the Qt user interface [12]. It provided an easy way to construct a manageable interface separating the simulation code from GUI code by having the "facility.py" as a middleman. PyQt5 also made it possible for my program to only use the Python programming language, which is a great advantage since large codes written in several languages can quickly become hard to manage.

## 4 Testing

### 4.1 Result comparison

When testing my program particles were simulated in both my program and another program called TraceWin [13]. After both programs had simulated the particles arrays for every parameter were made. Each array held the values of a parameter for all particles. Thus there was an array holding all $x$ parameter values calculated by my program and another array containing all $x$ parameter values calculated by TraceWin. A new array with the difference between the $x$ parameter values from my program and the $x$ parameter values from TraceWin was created. This difference array was created for the other parameters as well. When the difference arrays had been created for each parameter the standard deviation for each of them was taken. The standard deviation is a statistical term for how much values in an array deviate from the mean value and is denoted by $\sigma$. It is defined for an array $a$ with $N$ elements as

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(a_i - \mu(a))^2}, \tag{96}$$

where $\mu(a)$ is the mean value of $a$. The unit of the standard deviation is the same as the unit for the elements in the array $a$. The smaller the standard deviations for each array of difference between the resulting parameters of the two programs the better the agreement between the two programs.

In the Python library NumPy [14] there is a function called *std* which calculates the standard deviation for an array. The $\sigma$ for coordinate $x$ was calculated by

$$\sigma = std([x_{1,DAT} - x_{1,TW}, x_{2,DAT} - x_{2,TW}, \ldots, x_{N,DAT} - x_{N,TW}]), \tag{97}$$

for $N$ particles. By calculating $\sigma$ for each parameter I could measure how close the result in my program and TraceWin were. When I did multiple simulations in my program and compared the results with each other the same technique of calculating $\sigma$ values was used.

## 4.2 FODO-lattice

Particles with an energy of 2 GeV were sent through a so called FODO-lattice, in both my program and in TraceWin. The energy of 2 GeV was used since this is the energy of the beam that will be used at ESS [15]. The FODO-lattice consists of a half-quadrupole focusing in $x$, then a drift element, then a full length quadrupole defocusing in $x$, then a drift and finally a half quadrupole focusing in $x$. The parameters for the FODO-lattice can be seen in Table 1.

Table 1: The FODO-lattice.

| Element | $L$ [m] | $G$ [T/m] |
|---|---|---|
| Focusing Quadrupole | 0.05 | 100 |
| Drift | 0.10 | - |
| Defocusing Quadrupole | 0.10 | -100 |
| Drift | 0.10 | - |
| Focusing Quadrupole | 0.05 | 100 |

The resulting phase space plot in the $xx'$-plane is shown in Figure 4, where the Twiss parameters make up the ellipses in the two plots. The theory and parameters for the ellipses come from chapter 3.8 in [4]. The minor axis of the ellipse has the length of $\sqrt{\epsilon\beta}$ and the major axis of the ellipse has the length of $\sqrt{\epsilon\gamma}$. The ellipse is tilted clockwise by an angle

$$\Psi = \frac{1}{2}\arctan(\frac{2\alpha}{\gamma - \beta}). \tag{98}$$

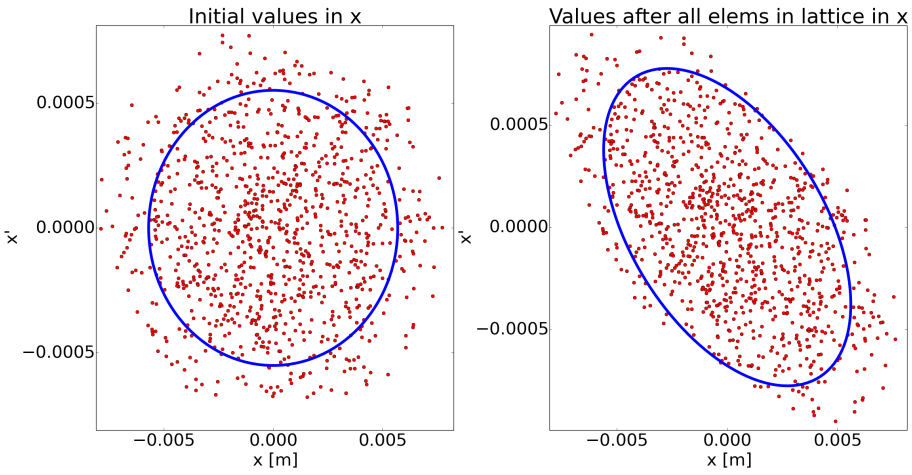


Figure 4: Simulation in my program of the FODO-lattice specified in Table 1.

The $\sigma$ values for each parameter of the results can be seen in Table 2. As the $\sigma$ values are extremely small it can be said that the two programs produce the

same results. However the $z$-value is not nearly as small as the other $\sigma$ values due to a small difference in the transfer matrices for $z$ and $\delta$ between my program and TraceWin. This difference comes from the fact that TraceWin uses a $dp/p$ parameter instead of the $\delta$ parameter and TraceWin has a different definition of the transfer matrix in the $z$-plane, see the "Transfer Matrices" chapter in [3].

Table 2: The $\sigma$ of the differences in each parameter between a FODO-lattice simulation in TraceWin and my program.

| Parameter | $\sigma$ |
|---|---|
| $x$ | 1.654e-10 m |
| $x'$ | 1.520e-11 |
| $y$ | 1.425e-10 m |
| $y'$ | 1.465e-11 |
| $z$ | 0.157 m |
| $\delta$ | 0.000 |

In Figure 4 the ellipses made by the Twiss parameters match the shapes made by the particles. This means that the Twiss parameter calculations are most likely correct. Not all particles are within the ellipses and this is due to the fact that particles are created in "particleFactory.py" with a Gaussian distribution and then shaped by Twiss parameters.

## 4.3 Sextupole simulation

How can one know that the non-linear dynamics of one's simulation are correct? Since the calculations quickly become very costly, the next best is to compare one's results with other results. For this I have used an example from Figure 9.1 in [2] shown in Figure 5. In this example 1024 particles are sent through a circular ring 1000 laps. A lap in a periodical circular ring was simulated by a one-turn with a phase of $\nu = 0.246 \times 2\pi$ radians. This would rotate the phase spaces of the transversal dimensions by the phase $\nu$ each turn. This simulates all the quadrupoles and drifts that the particles travel through. After each lap, they encounter a sextupole element with length 0.1 m and a strength K = 4000 $\text{m}^{-3}$ where the Lie transform has used an order of 5. The particles in this test started in the $xx'$-plane between -4 mm and +4 mm in $x$, and -4e-3 and +4e-3 in $x'$. In the other dimensions, the particle parameters were set to zero since in [2] the parameters were only defined in $x$ and $x'$. At the end of the 1000 laps, the phase space plot for $x$ and $x'$ was plotted.

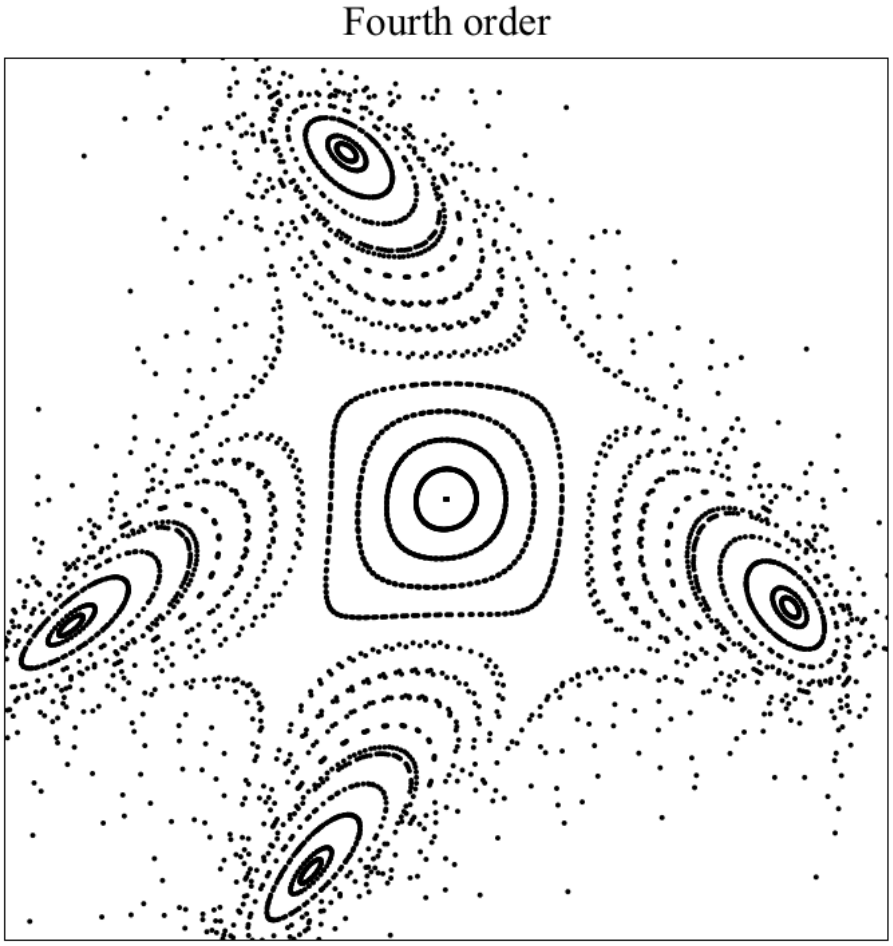


Figure 5: Reference simulation of a sextupole in a ring: $\nu = 0.246 \times 2\pi$; sextupole: L = 0.1 m, K = 4000 $\mathrm{m}^{-3}$, order = 5.

The resulting phase space plot from my program is shown in Figure 6. In my simulation the particle parameters started in an even grid within the above specified limits. Since some of the particles were unstable they were set to zero and deemed lost when the coordinates started to grow too large. The limit for particles were if

$$\sqrt{x^2 + x'^2 + y^2 + y'^2 + z^2 + \delta^2} > 1. \tag{99}$$

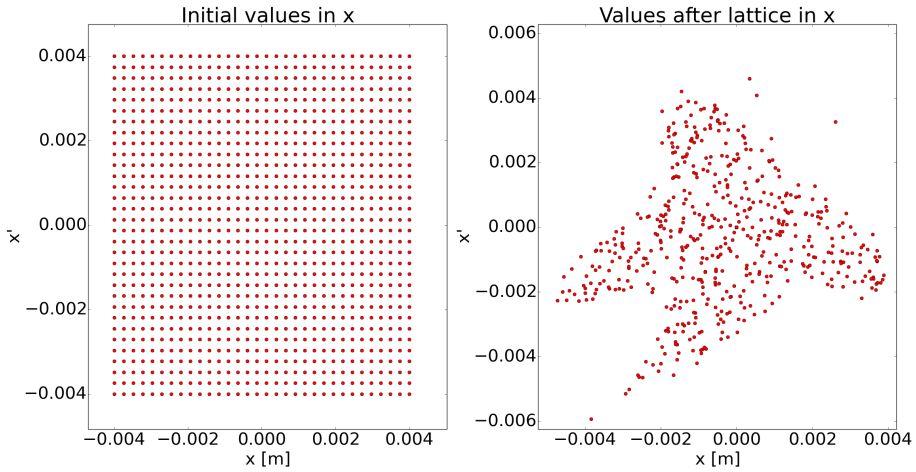


Figure 6: Simulation of a sextupole in a ring: $\nu = 0.246 \times 2\pi$; sextupole: L = 0.1 m, K = 4000 $m^{-3}$, order = 5. The determinant of $J$ is 3.89e-03 away from 1. The number of lost particles were 510.

By comparing Figure 5 and 6 one can see that they have the same "bent cross" shape but the internal structure is more chaotic in my results. The reason for this is probably that the starting parameters for the particles are not the same. The starting parameters which gives the results seen in Figure 5 are within the same range as the initial parameters in Figure 6 but it is unclear how they are distributed within that range.

The sextupole was also tested with the symplectic integrator technique by replacing the sextupole element in the previous setup with an element with only the kinetic terms of the Hamiltonian followed by a drift element with the length of the original sextupole. The results for this can be seen in Figure 7, where the same initial particles as in Figure 6 has been used. The final parameter values in the $x$-plane for the particles in Figure 6 and 7 w compared with the method described in Section 4.1 which gave the results seen in Table 3. It can be seen that the $\sigma$ values are much larger than those for the FODO-lattice found in Table 2. This means that the two techniques don't produce the same result. Since so many particles are lost there is a risk that a particle is lost with the normal Lie transform approach but not with the symplectic integrator approach and this would mean that the $\sigma$ value would be extra large because of that particle.

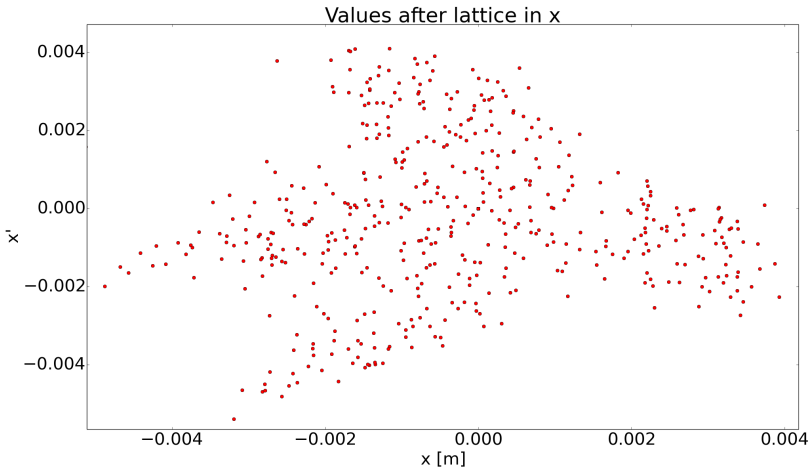


Figure 7: Simulation of a sextupole in a ring calculated with a symplectic integrator: $\nu = 0.246 \times 2\pi$; sextupole: L = 0.1 m, K = 4000 m$^{-3}$, order = 5; drift: 0.1 m. The determinant of $J$ for the kinetic part of a sextupole is equal to 1 with more than 6 digits and thus the calculations are called fully symplectic. The number of lost particles were 546.

Table 3: The $\sigma$ of the differences of the parameters in the $x$-plane for the two techniques of simulating a sextupole.

| Parameter | $\sigma$ |
|---|---|
| $x$ | 1.441e-3 m |
| $x'$ | 2.293e-3 |

## 4.4 Octupole simulation

Octupoles are magnetic elements which have eight magnetic poles and are also non-linear accelerator elements. Here there were no example to compare the results with. Instead the setup for the sextupole test in Section 4.3 was used with the sextupole replaced by an octupole. In Figure 8 the results for this setup are shown, where the same initial particles as in Figure 6 has been used.

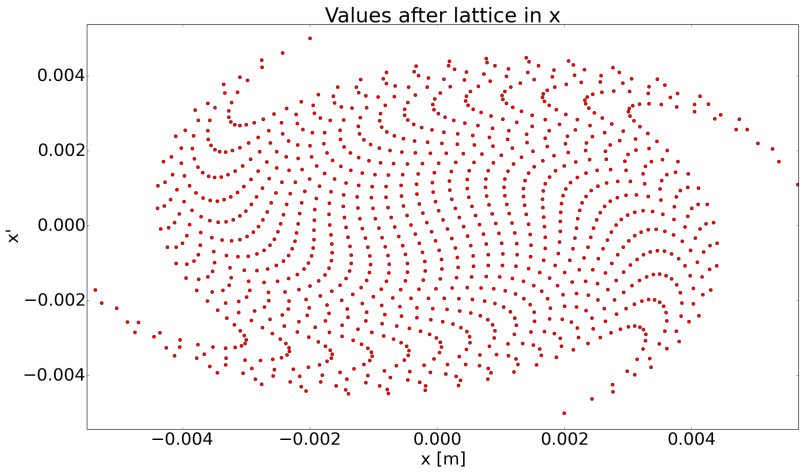


Figure 8: Simulation of an octupole in a ring: $\nu = 0.246 \times 2\pi$; octupole: L = 0.1 m, K = 4000 m$^{-4}$, order = 5. The determinant of $J$ for an octupole element is equal to 1 with more than 6 digits and thus the calculations are called fully symplectic.

This setup with the same initial particles was also simulated with the symplectic integrator technique in two ways. The first way was by having an element with only the kinetic terms of the octupole Hamiltonian followed by a drift element with the length of the real octupole. The result for this is shown in Figure 9.

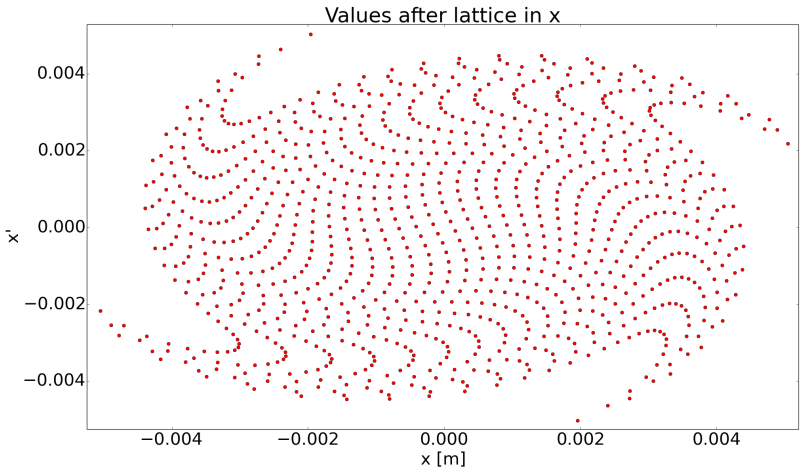


Figure 9: Simulation of an octupole in a ring calculated with a symplectic integrator: $\nu = 0.246 \times 2\pi$; octupole: L = 0.1 m, K = 4000 m$^{-4}$, order = 5; drift: 0.1 m. The determinant of $J$ for the kinetic part of an octupole element is equal to 1 with more than 6 digits and thus the calculations are called fully symplectic.

The second way when doing the symplectic integrator technique was to do the same as the first one but splitting the drift element in two equally long elements and placing one before the kinetic octupole element and the other after the kinetic octupole element. The results for this are shown in Figure 10.

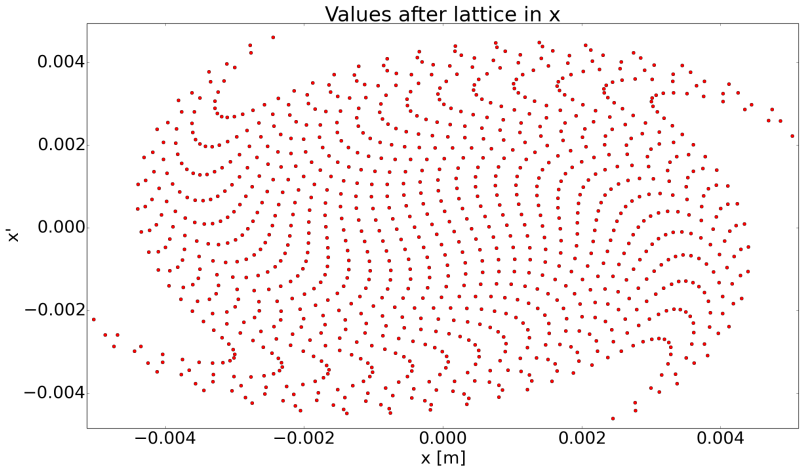


Figure 10: Simulation of an octupole with split drifts in a ring calculated with a symplectic integrator: $\nu = 0.246 \times 2\pi$; drift: 0.05 m; octupole: L = 0.1 m, K = 4000 $\mathrm{m}^{-4}$, order = 5; drift: 0.05 m. The determinant of $J$ for the kinetic part of an octupole element is equal to 1 with more than 6 digits and thus the calculations are called fully symplectic.

Table 4: The $\sigma$ of the differences in the phase space for $x$ between the values in Figure 8 and 9.

| Parameter | $\sigma$ |
|---|---|
| $x$ | 1.084e-05 m |
| $x'$ | 1.092e-05 |

Table 5: The $\sigma$ of the differences in the phase space for $x$ between the values in Figure 8 and 10.

| Parameter | $\sigma$ |
|---|---|
| $x$ | 4.686e-05 m |
| $x'$ | 4.950e-05 |

Table 6: The $\sigma$ of the differences in the phase space for $x$ between the values in Figure 9 and 10.

| Parameter | $\sigma$ |
|---|---|
| $x$ | 1.532e-05 m |
| $x'$ | 1.567e-05 |

There are now three simultion results that can be compared to each other. The final parameter values for the particles in Figure 9 and 10 was then compared with the final parameter values in Figure 8 by using the method described in Section 4.1. In the Tables 4, 5 and 6 the results are compared with each other. It is shown in the tables that the three different techniques have results which are very close to each other due to the small $\sigma$ values in all of the tables. This means that one can use each of the three techniques to simulate an octupole.

## 4.5 Radiofrequency cavities

To test the rf-cavity in my program, particles with energy 2 GeV were sent through a single rf-cavity. Since the rf-cavity has the most interesting results in the $z\delta$-plane only the $z$ and $\delta$ parameter values will be calculated. The parameters for the rf-cavity were: $L = 1$ m, $a = p_{01}/\pi$, $E_0 = 10^9$ V/m and $\phi_0 = \pi/4$. The beam energy was increased from 2000.00 MeV before the rf-cavity to 2474.37 MeV after the rf-cavity. The resulting phase space in the $z\delta$-plane is shown in Figure 11.

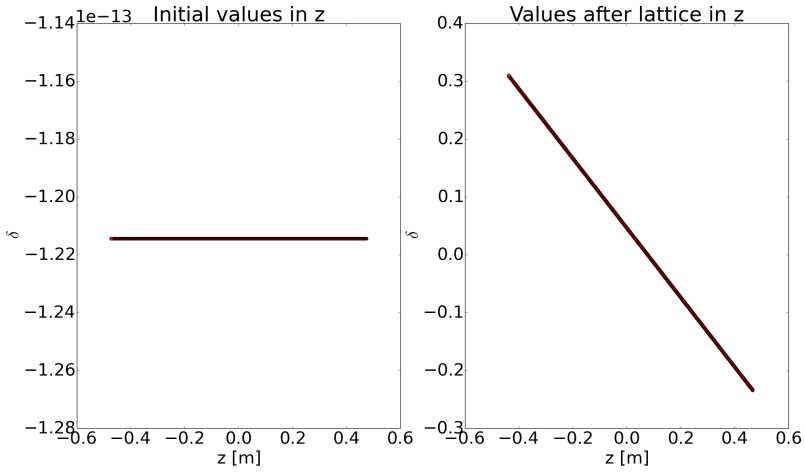


Figure 11: The phase space in $z\delta$-plane before and after a rf-cavity with: $L = 1$ m, $a = 0.766$ m, $E_0 = 10^9$ V/m and $\phi_0 = \pi/4 = 0.785$.

The particles with the lowest and highest initial $z$ parameter values were then calculated by only using the equations described in Section 2.6. The particle with the lowest $z$ parameter value is called particle 1 and the particle with the highest $z$ parameter value is called particle 2. In Table 7 the results for the particles from both my program and direct calculations are shown.

Table 7: The results for the two particles with both my program and direct calculations.

| Initial parameter | Values for particle 1 | Values for particle 2 |
|---|---|---|
| $z$ | -0.473 m | 0.474 m |
| $\delta$ | -1.214e-13 | -1.214e-13 |
| My program's results | | |
| $z$ | -0.439 m | 0.467 m |
| $\delta$ | 0.310 | -0.234 |
| Direct calculation results | | |
| $z$ | -0.439 | 0.467 |
| $\delta$ | 0.310 | -0.234 |

It is seen that my program and direct calculations match perfectly. The reason for this is that my program and the calculations are based on the same equations and thus will produce the same results when the parameters are the same.

# 5 Discussion

It has been shown here that my program works for simulating both linear and non-linear particle dynamics.

However, it does not seem possible to simulate the space charge effect in my program using the theoretical basis in [2], [5] and [6]. The theoretical basis in [6] has been used succesfully in another program, see [15]. A problem is that for an accurate simulation one needs to slice each element into many parts, since the forces arising from space-charge will change over very small distances. This means that instead of just doing the calculations for an element once, one would need to do them a thousand times, which is very demanding to do on a normal computer if the code isn't properly optimized. It may be possible to overcome this problem by having a theoretical description which only requires that the particles be calculated once per element, or perhaps if the particle distribution was elliptically symmetric or even a KV-distribution. Another problem for space charge is that it couples the axes since the forces depend on the three dimensional distances between the particles.

For calculating the slicing of elements a numerical differentiation technique known as the "leapfrog algorithm" could have been used when using Hamiltonians. The leapfrog algorithm steps through time or the longitudinal coordinate $s$ and calculates the position based on the velocity half a step length before. A thorough introduction to this technique is found in Chapter 9-6 in [16]. This source doesn't use the Hamiltonian formalism but with some calculations the technique can be used with Hamiltonians as well. The leapfrog algorithm would have demanded a rewrite of most of my code, but might have had better results, since it is a very well established technique in mathematics and other areas.

The approximations made in this thesis were that the paraxial approximation

first used in Section 2.2 holds, that the polynomial after a Lie transform could be truncated after a certain order and that the Hamiltonian could be split up for the symplectic integrator approach. The paraxial approximation was the easiest to justify, since the momentum in the $z$-direction was larger than the other momenta by several orders of magnitude. The polynomial truncation was trickier, but by calculating the Jacobian and the Jacobian's determinant it was verified that it was sufficiently close to 1 since the furthest the determinant of the Jacobian deviated from 1 was in the third decimal. That the Hamiltonian split in the symplectic integrator worked was demonstrated by comparison with the normal Hamiltonian. For the sextupole when using the symplectic integrator technique the calculations were fully symplectic but not when the normal Hamiltonian was used. This also shows the usefulness of the symplectic integrator technique.

Another way of dealing with non-linear elements could have been used. This is to linearize them so that they fit into the matrix formalism. Since every element would have had matrices then the evaluation might have gone quicker but it is questionable if it would have given better results.

Although I have written much code and simulated many aspects of accelerators, I continue to discover new effects that one must recognize when writing simulation software. One need to provide many tools for the user if they should wish to simulate an accelerator through the software. There are many approaches one can take when simulating accelerators and I find my program to work well for combining two approaches (the linear and non-linear). It still needs to be further optimized for speed before it is capable of simulating large machines such as ESS and the LHC. This is because the major drawback of my program is that it is quite slow when simulating a long accelerator of more than 1000 elements when having more than 1000 particles in the beam.

After this thesis my program will be used to simulate non-linear particle effects at ESS. It can also be used for teaching accelerator physics to students since the interface is quite user friendly and the code is written to be as clear as possible. For the code itself I hope that it will be a good basis if others want to write their own simulators. I would like to get the space-charge effect properly simulated as well.

# 6 End note

This master thesis has been enjoyable and I would like to continue in this area of research. The challenge of combining physics with programming has been interesting. I have deepened my knowledge of both areas as well as my knowledge of accelerator physics. Since this thesis has touched several fields of science I have learned the difficulties and benefits of working in an interdisciplinary environment.

# 7 Acknowledgements

I wish to thank my deputy supervisor Emanuele Laface and primary supervisor Mats Lindroos. I also wish to thank Ben Folsom for proofreading this report. The ICS team at ESS has been very helpful as well.

# References

[1] Inc. Free Software Foundation. Gnu general public license, version 3. `https://www.gnu.org/licenses/gpl-3.0.html`, 2007. Accessed: 2016-06-28.

[2] Andrzej Wolski. *Beam Dynamics in High Energy Particle Accelerators*. Imperial College Press, 2014.

[3] Didier Uriot and Nicolas Pichoff. *TraceWin documentation*. CEA/SACLAY - DSM/Irfu/SACM, 2011.

[4] Klaus Wille. *The Physics of Particle Accelerators: an introduction*. Oxford University Press, 2000.

[5] Christopher K. Allen. *Theory and Technique of Beam Envelope Simulation*. Los Alamos National Laboratory, 2002.

[6] P. M. Lapostolle. *Effets de la Charge D'Espace Dans un Accelerateur Lineaire a Protons*. CERN, 1965.

[7] SymPy Development Team. Sympy. `http://www.sympy.org/en/index.html`, 2016. Accessed: 2016-05-17.

[8] Alex J. Dragt. *Lie Methods for Nonlinear Dynamics with Applications to Accelerator Physics*. University of Maryland, College Park, 2015.

[9] S. Y. Lee. *Accelerator Physics*. World Scientific Publishing Co. Pte. Ltd., 2004.

[10] Christopher K. Allen. A software engineering approach to particle beam simulation. 1999.

[11] Riverbank Computing. Pyqt5. `https://www.riverbankcomputing.com/software/pyqt/intro`, 2016. Accessed: 2016-05-17.

[12] The Qt Company. Qt. `https://www.qt.io/`, 2016. Accessed: 2016-06-01.

[13] CEA France. Tracewin. `http://irfu.cea.fr/Sacm/en/logiciels/index3.php`, 1998. Accessed: 2016-06-01.

[14] NumPy Developers. Numpy. `http://www.numpy.org/`, 2016. Accessed: 2016-05-18.

[15] E. Laface, M. Eshraqi, and R. Miyamoto. *Space Charge and Cavity Modeling for the ESS Linac Simulator*. Proceedings of IPAC2013, Shanghai, China. ESS, Lund, Sweden, 2013.

[16] R. B. Leighton R. P. Feynman and M. Sands. *The Feynman Lectures on Physics*. Basic Books, 2011.