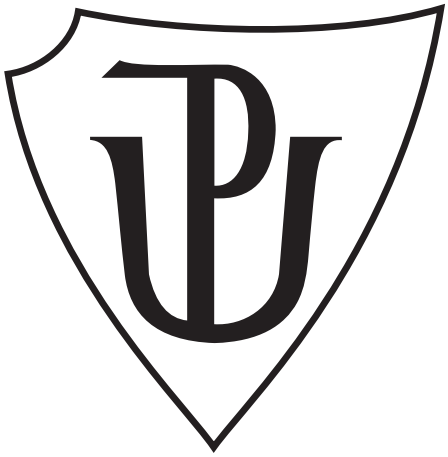
Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra experimentální fyziky

BAKALÁŘSKÁ PRÁCE

Vizualizace drah částic v magnetickém poli urychlovače



Vypracoval	**Tomáš Komárek**
Studijní program	B1701 Fyzika
Studijní obor	Aplikovaná fyzika
Forma studia	Prezenční
Vedoucí diplomové práce	Mgr. Tomáš Sýkora, Ph.D.
Termín odevzdání práce	srpen 2013

Bibliografická identifikace

Jméno a příjmení autora	Tomáš Komárek
Název práce	Vizualizace drah částic v magnetickém poli urychlovače
Typ práce	Bakalářská
Pracoviště	Katedra experimentální fyziky
Vedoucí práce	Mgr. Tomáš Sýkora, Ph.D.
Rok obhajoby práce	2013
Abstrakt	Maticová optika je metoda popisu chování optických systémů. Je vhodná jak pro popis optiky světelných paprsků, tak i nabitých částic. V této práci se budu věnovat právě nabitým částicím. Při popisu chování v urychlovači částic je každá částice v daném bodě popsána vektorem. Magnetické soustavy jsou pak popsány transformačními maticemi, které transformují vektor vstupní částice na výstupní. Všechny výpočty probíhají relativně k tzv. „nominální částici“. Dráha této částice je přímka v případě fokusujících a defokusujících systémů, ale pokud je svazek ohýbán dipólem, musíme si ověřit její dráhu jiným způsobem. Cílem této práce je implementovat program, který v reálném čase provádí výpočty maticové optiky podle toho, jak uživatel mění vstupní parametry optických prvků. Software bude mít grafické rozhraní a bude vykreslovat dráhy částic společně s prvky, které jejich dráhy ovlivňují.
Klíčová slova	Urychlovač částic, maticová optika, CERN, LHC
Počet stran	35
Počet příloh	1
Jazyk	Anglický

Bibliographical identification

Autor's first name and surname	Tomáš Komárek
Title	Visualization of particle tracks in an accelerator magnetic field
Type of thesis	Bachelor
Department	Department of Experimental Physics
Supervisor	Mgr. Tomáš Sýkora, Ph.D.
The year of presentation	2013
Abstract	Matrix optics is an approach to describe behavior of optical systems. It is suitable for both optics of light rays as well as charged particles. In my thesis, I will focus on the latter. In case of particle accelerator, every particle at given position is described by a vector. Magnetic systems are described by a matrix which transforms incoming particle vector to an outcoming one. All computations are relative to the so called "nominal particle". Its track is simple straight line in case of focusing and defocusing systems, but when the beam is bent by a dipole, we have to make sure it goes as expected in other ways than the matrix optics. The goal of this thesis is to implement a computer program which performs matrix optics computations in realtime as parameters of optical elements are changed by the user. The software will have a graphical interface and will draw particle tracks together with the beamline elements.
Keywords	Particle accelerator, matrix optics, CERN, LHC
Number of pages	35
Number of appendices	1
Language	English

Prohlášení

Prohlašuji, že jsem předloženou diplomovou práci vypracoval samostatně pod vedením Mgr. Tomáše Sýkory, Ph.D. a že jsem použil zdrojů, které cituji a uvádím v seznamu použitých pramenů.

V Olomouci dne 20. srpna 2013

.....

Tomáš Komárek

Poděkování

Chtěl bych tímto poděkovat své rodině za dlouhodobou podporu, svým spolužákům za starost a zájem o průběh práce a především pak Mgr. Tomáši Sýkorovi, Ph.D. nejen za jeho ochotu kdykoli řešit nastalé komplikace.

Contents

Introduction

Particle accelerators range from fairly simple devices to highly complex ones. One of the simple accelerators are present in classic vacuum screens of older televisions, where static electric field accelerates electrons. However, energies achieved this way are very low compared to accelerators with oscillating magnetic field.

There are two general designs of accelerators: linear and circular. They are suitable for accelerating different types of particles, generally circular accelerators are better for accelerating heavier particles, but light ones suffer a lot of energy loss due to synchrotron radiation. I will focus on particle transport in the LHC[1] circular accelerator when the particles are already accelerated. The interaction point used as a starting point for my simulation will be the one inside the ATLAS detector.

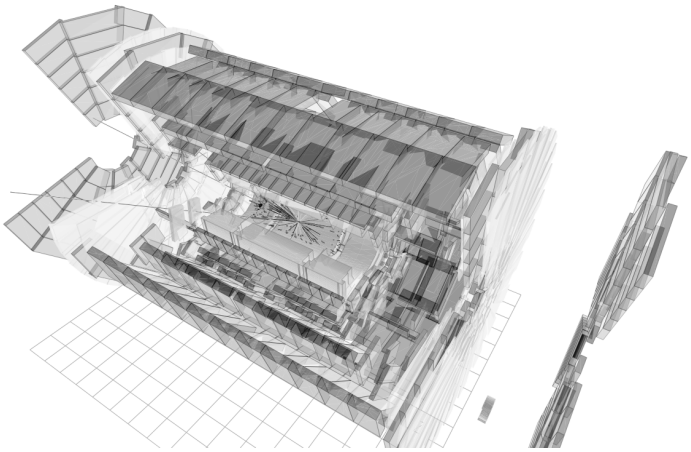


Figure 1: The ATLAS detector

The LHC is currently the largest circular accelerator in the world. Its circumference is 27 km and is capable of reaching energies up to 7 TeV per proton, that is 14 TeV per collision.

There are many types of elements in the beamline, but given some conditions (that will be discussed later), it is enough to implement only quadrupole magnets, dipole magnets, drift space (vacuum with no field) and apertures. I expect all other elements to exactly compensate for any energy loss caused by synchrotron radiation and other inaccuracies, therefore I consider them to behave like free drift space.

[1] Large Hadron Collider in CERN

My goal is to create a computer simulation with interactive visualization of particles and twiss file[2] import capabilities. Currently, no other software offers direct visualisation capabilities with realtime simulation as user changes beamline parameters. Final verification of the simulation software will be done with beamline setup of the LHC and data from the MAD-X simulation software[3], but any other accelerator geometry can be imported when corresponding twiss files are provided.

[2]twiss file is a file describing beamline geometry and magnet settings

[3]See `http://frs.web.cern.ch/frs/Xdoc/mad-X.html` or [3]

Chapter 1

The theory of accelerator optics

The fundamental principle of particle movement in accelerator field is described by the Lorentz force. When a particle with electric charge q moves through the magnetic field $\vec{B}$ with velocity $\vec{v}$, the resulting force is

$$\vec{F} = q(\vec{E} + \vec{v} \times \vec{B}). \tag{1.1}$$

As the velocity approaches c, the magnetic field gets much more effective at generating force on the particle than achievable electric field. The electric field $\vec{E}$ is neglected and we only consider the transverse magnetic field component.

To keep particles circulating in the accelerator, we need bending forces to keep the particles on a closed circular trajectory and restoring forces that keep the particles near the design orbit. Based on how strong field is applied, we can use weak or strong focusing.

1.1 Weak and strong focusing

1.1.1 Weak focusing

When the magnetic field is not depending on the azimuthal angle, we obtain weak focusing. The design orbit has radius

$$\rho = \frac{mv}{eB} \quad [1].$$

In an accelerator with weak focusing, the magnetic field creates angle independent restoring forces. For small deviations, those forces rise linearly with the horizontal or vertical deviation from the design orbit. The restoring forces therefore cause harmonic oscillations around the design orbit, called betatron oscillations. However, with the weak focusing the betatron oscillation frequency is lower than the revolution frequency. This way, the wavelength of the oscillations is larger than the circumference of the orbit and there are large deviations from the design orbit. [1].

1.1.2 Strong focusing

Strong focusing is based on a different principle than the weak focusing. Instead of continuous magnetic field, series of focusing and defocusing elements are used. This leads to overall focusing, because the particles enter the focusing elements generally further from the ideal orbit than they enter the defocusing ones. This principle is illustrated on the Figure 1.1.

There are two ways to achieve strong focusing. The first is to use combined function magnets, which do both bending and focusing/defocusing. The other is to separate those functions in standalone focusing and bending elements. In large modern accelerators, magnets with separated functions are usually used.

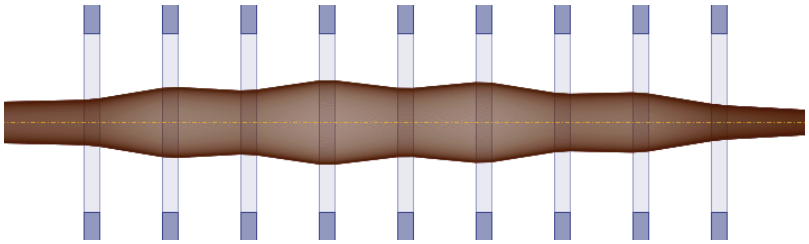


Figure 1.1: The effect of alternating focusing and defocusing elements of the same field strength realized on my simulation software

1.2 Magnet types

There are several types of magnets used in particle accelerators with strong focusing. An ideal dipole magnet (Figure 1.2) has homogeneous field in the gap, which causes the particle to run along an arc in the horizontal plane. In the ideal case, there are no effects on the movement in the vertical plane.

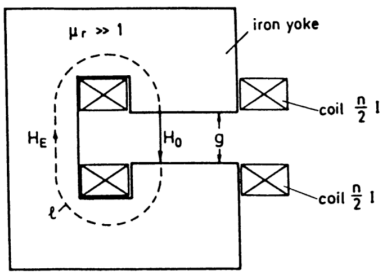


Figure 1.2: Dipole magnet cross section [1]

The quadrupole magnet (Figure 1.3) is used for focusing and defocusing. It has hyperbolically shaped inner contour. The intensity of the horizontal and vertical magnetic field components produced increases linearly for small horizontal and vertical deviations respectively. This causes the quadrupole to behave as a focusing element in the horizontal plane and defocusing in the vertical plane, and vice versa for reversed electric current or particle charge.

The combined function magnet is combining the dipole and quadrupole role. It can be viewed as a quadrupole with an offset from the center, as shown on the Figure 1.4.

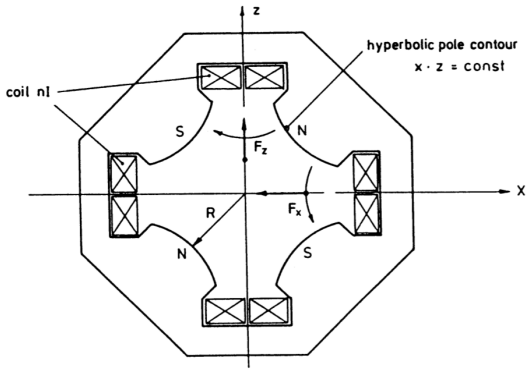


Figure 1.3: Quadrupole magnet cross section [1]

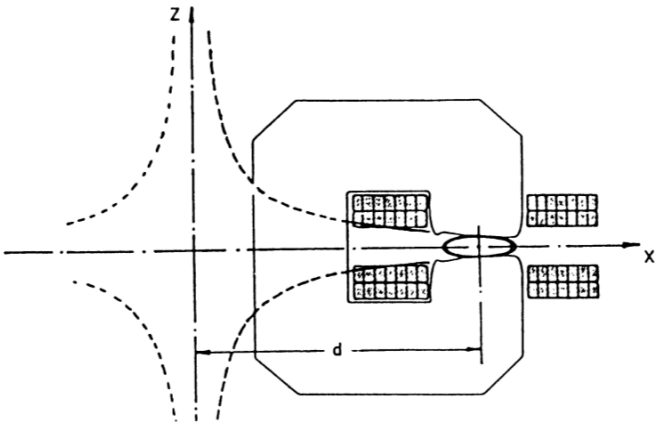


Figure 1.4: Combined function magnet cross section [1]

There are some other beamline element types, such as the sextupole, collimators, acceleration cavities etc., but they are not important for my simulation, at least in the part of the beamline I am focused on. I expect those elements to compensate for any energy loss caused by synchrotron radiation, imperfections of some elements realization and other inaccuracies, so I consider them to behave like a free drift space[1] in my simulation.

1.3 Matrix representation

Under certain considerations, one can represent particles by vectors and magnets by matrices applied to them. This approach makes use of the so called nominal particle and nominal trajectory[2], all computations are relative to them.

The general formula for transporting particle through a magnet in two dimensions takes the form

$$\begin{pmatrix} x \\ x' \\ \frac{\Delta p}{p_0} \end{pmatrix}_S = \overbrace{\begin{pmatrix} C & S & D \\ C' & S' & D' \\ 0 & 0 & 1 \end{pmatrix}}^{M_x} \begin{pmatrix} x \\ x' \\ \frac{\Delta p}{p_0} \end{pmatrix}_{S_0}, \tag{1.2}$$

[1]Drift space is a section, where no field affects the particles

[2]Trajectory of the nominal particle is the design trajectory with no oscillations

where x is the transverse distance from the nominal particle trajectory, x' is the rate at which x is changing along the trajectory (change of transverse position per meter) and $\frac{\Delta p}{p_0}$ is relative deviation of the particle momentum p from the considered nominal particle momentum p_0 (where $\Delta p = p - p_0$). S_0 and S are the starting and ending points of the transport through the element described by the matrix. The x' can be viewed as a deviation angle, because $x' = \tan(\theta) \approx \theta$ for $|\theta| \ll 1$.

1.3.1 Limitations and advantages

In order to be able to use the matrix representation, one must consider some limitations. The most important one is that the nominal trajectory must go exactly through the center of all beamline elements and be perpendicular to them (except for rectangular dipole, but the trajectory must be symmetrical in respect to the dipole). Without information about the nominal trajectory, we cannot use dipoles, only set of quadrupoles, where we know the nominal trajectory is a straight line. When dipoles are being used, we must make sure the trajectory goes through the centers of used elements in other ways; matrix optics cannot be used without the nominal trajectory being precomputed. Here we also consider the perpendicular movements along the x and the y axes as mutually independent.

The coordinate system used is shown on Figure 1.5. It is a local coordinate system bound with the nominal particle, where x, s is the horizontal plane and y, s is the vertical plane.

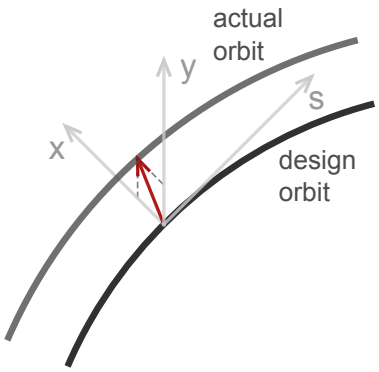


Figure 1.5: Local coordinate system used

Given those constraints, some advantages arise. We need not to know the exact parameters of the particle, like its charge or mass. Knowing that it is identical to the nominal particle (and its parameters can be unknown as well) is enough. In case of a different momentum, all we need to know is the relative deviation $\frac{\Delta p}{p_0}$ from the momentum of the nominal particle. Apart from that, we of course need to know its relative position x and the angle x'.

1.3.2 Transformation matrices of various beamline elements

Drift space

Drift space is an area without any field, where all particles travel along straight lines. The transformation matrices here have a very simple form.

$$M_x = M_y = \begin{pmatrix} 1 & l & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.3}$$

As seen in the Equation 1.3, the drift space representation is not much different from an identity matrix. The only difference here is the l in the first row and second column, describing the length of the drift space. Based on rules of matrix multiplication, it is easy to see that when fitted into Equation 1.2, the only change is $x_S = x_{S_0} + l \cdot x'_{S_0}$, which corresponds to the way x' is defined.

Quadrupole

Quadrupole magnets are used for their focusing and defocusing capabilities. We introduce the momentum independent quadrupole coefficient $k\,[m^{-2}]$ and define the quadrupole to be horizontally focusing and vertically defocusing for $k > 0$. The coefficient φ is defined as $\varphi = l\sqrt{|k|}$, where l is the length of the quadrupole.

For $k > 0$ the matrices are:

$$\begin{aligned} M_x &= \begin{pmatrix} \cos(\varphi) & \frac{1}{\sqrt{|k|}}\sin(\varphi) & 0 \\ -\frac{1}{\sqrt{|k|}}\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix}, \\ M_y &= \begin{pmatrix} \cosh(\varphi) & \frac{1}{\sqrt{|k|}}\sinh(\varphi) & 0 \\ \frac{1}{\sqrt{|k|}}\sinh(\varphi) & \cosh(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \end{aligned} \tag{1.4}$$

For vertical focusing and horizontal defocusing ($k < 0$), the matrices are simply interchanged.

From the transformation matrices in the Equations 1.4, we can see that the last column introduces no dependance on the momentum, the effects of an ideal quadrupole are therefore momentum independent.

Sector dipole magnet

A sector dipole magnet is a magnet, where the nominal particle trajectory enters and leaves the front and rear face in a perpendicular direction, as shown on the Figure 1.6.

The sector dipole magnet here has no effect in the vertical plane but bends the nominal particle in the horizontal plane by the angle $\varphi = \frac{l}{\rho}$, where l is the length of the circular trajectory and ϱ is the arc radius. It behaves as a drift space in the vertical plane.

$$M_x = \begin{pmatrix} \cos(\varphi) & \rho\sin(\varphi) & \rho(1-\cos(\varphi)) \\ -\frac{1}{\rho}\sin(\varphi) & \cos(\varphi) & \sin(\varphi) \\ 0 & 0 & 1 \end{pmatrix} \quad M_y = \begin{pmatrix} 1 & l & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.5}$$

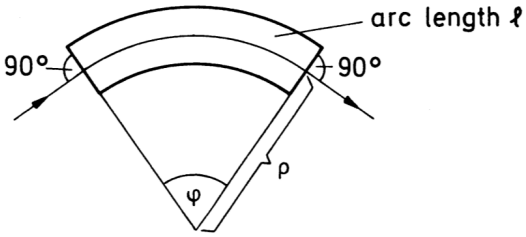


Figure 1.6: Sector dipole magnet [1]

We can see from the transformation matrices in the Equation 1.5 that the horizontal one is identical to the drift space from the Equation 1.3. The sector dipole magnet has a weak horizontal focusing effect, because the particles entering with larger radius travel longer and those with smaller radius travel shorter distance through the magnetic field, but their trajectories all have the same curvature radius, therefore the bending angle varies.

Rectangle dipole magnet

A rectangle dipole magnet has a rectangular shape, which causes the particles to enter and leave the magnetic field under an angle, as shown on the Figure 1.7.

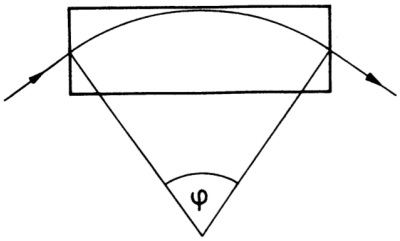


Figure 1.7: Sector dipole magnet [1]

The rectangle dipole magnet has no horizontal focusing effect (all particles parallel to the nominal trajectory travel the same distance through the field), but due to the entering and leaving angle, we have to take some imperfections of the magnetic field at the edges into account. The main reason is known as the "fringe fields"[3], which cause a weak vertical focusing. The M_y matrix for the rectangle dipole from the Equation 1.6 is therefore no longer equivalent to drift space.

$$M_x = \begin{pmatrix} 1 & \rho\sin(\varphi) & \rho(1-\cos(\varphi)) \\ 0 & 1 & 2\tan(\varphi/2) \\ 0 & 0 & 1 \end{pmatrix} \quad M_y = \begin{pmatrix} \cos(\varphi) & \rho\sin(\varphi) & 0 \\ -\frac{1}{\rho}\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.6}$$

As seen in the M_x matrices for both types of dipole magnets in the Equation 1.5 and Equation 1.6, the components in the third column describe how the dipole magnet effect depends on the momentum of the particle.

[3]Fringe field is an area at the edges, where the field is no longer homogeneous. Due to the trajectory not being perpendicular at the edges, a non-zero component affecting the behavior in the vertical plane arises. [2]

Chapter 2

Realization of the simulation software

The implementation language was chosen to be C++ with the Qt framework[1], using the Qt Creator IDE. All development was focused on a 64-bit Linux platform and all testing took place there. However, nothing should prevent the code from being successfully compiled and run on the Microsoft Windows™ platform or any other supported by the Qt framework.

Before I started working on the final software product, I implemented some isolated simulation cases to verify and evolve the overall design as well as learn C++ and Qt well enough. The source code of all evolution stages mentioned together with binaries are available in the appendix (CD).

2.1 Isolated single dipole implementation

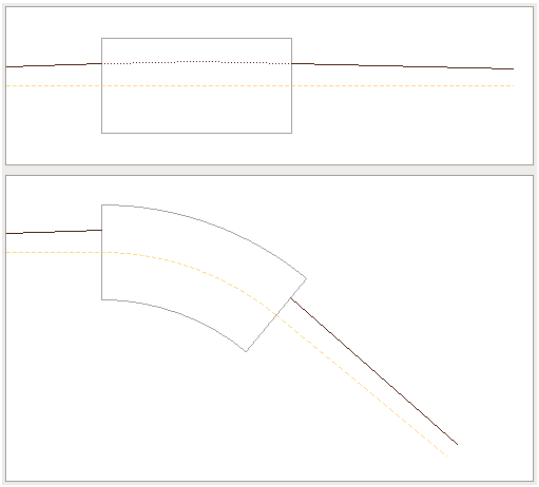


Figure 2.1: Graphical output of the SingleDipole program

My first attempt to implement the matrix optics approximation simulation was a single sector dipole magnet, able to transport only one particle at a time. The internal design of the "SingleDipole" application is not very advanced and extending

[1] `http://qt-project.org/`

the software to contain more than one element would be unnecessarily difficult. The main focus here was to learn basics of Qt.

On the Figure 2.1 we can see an example of the graphical output, where the top part is drawn relative to the nominal trajectory and the bottom one is using an absolute coordinate system. The simulation covers only the horizontal plane, as in the vertical plane the particle behaves just as in a drift space (see Equation 1.5).

2.2 Series of quadrupole magnets

The next step was to implement a beamline capable of containing many elements instead of a hard-coded single matrix operation. I also made it possible to have elements of various types in the beamline, even though only quadrupoles were used at this stage of development. The "QuadSeries" application is not based on the SingleDipole application, it is written from scratch instead.

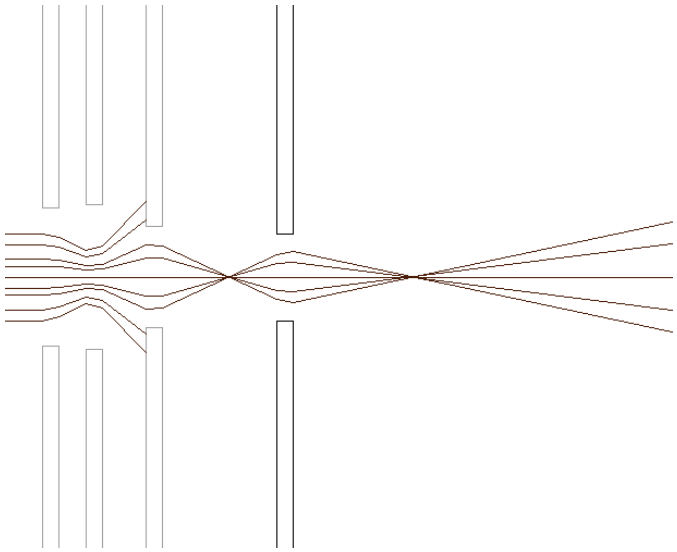


Figure 2.2: Graphical output of the QuadSeries program

On the Figure 2.2 we can see a sample output of the QuadSeries application. There is still only single projection of the particle to the horizontal or vertical plane, but more magnets and more particles are already supported.

The internal design and implementation of the beamline containing the elements proved to be good and robust, so I based the development of the final simulation software on the QuadSeries code. The implementation details will not be discussed here, because the basic structure is the same as in the final simulation software.

2.3 The EasyTracker simulation software

The final version of my matrix optics simulation software is called "EasyTracker". As mentioned above, it is an evolution of the QuadSeries application. It includes many improvements over the previous versions:

- renders both horizontal and vertical projection simultaneously,
- the nominal trajectory is automatically shown,
- completely redesigned user interface (especially regarding magnet settings) – beamline elements can be now chosen by clicking and multiple can be edited at once,
- improved visual appearance with anti-aliasing of particle tracks and better aperture indication,
- longitudinal axis with labels,
- coordinates can now be obtained by clicking any point in the visualization,
- added support for both dipole and quadrupole magnets as well as a plain drift space with apertures,
- a "terminal plane" element, which ends the simulation,
- correct circular and rectellipse[2] aperture implementation,
- particle generator for easy generation of many particles with similar properties,
- twiss file import capability,
- export of generated image in the PNG format,
- export of particle data on the terminal plane or aperture hit in the CSV format.

However, only the positions relative to the nominal trajectory are displayed. The absolute positions would be counterproductive anyway, now the user can have high resolution in the transverse plane and see all the details close to the nominal trajectory at once.

2.3.1 Basic structure and internal logic

The C++ language features object oriented capabilities and the internal logic of my simulation software is based on them. Brief description of the most important classes follows.

`MainWindow`

- class that handles the top level GUI[3] window, implements slots connected to GUI elements such as buttons and menus
- contains drawing area where beamline elements and particle tracks are rendered

[2]The "rectellipse" shape will be described later, see Figure 2.4
[3]Graphical User Interface

`Beamline`

- class that contains beamline elements stored in a double linked list, this list is kept in the correct order based on the positions of those elements
- contains public methods for adding/changing/removing/drawing beamline elements, reading information about elements and finding nearest element to given position (for `MagnetEditor`)

`Magnet`

- class that represents general beamline element contained in the `Beamline` class
- contains public methods for setting and getting parameters of the element, drawing it to given pixmap and transporting given particle through it
- is not used directly, but other classes are derived from it, thus they inherit its public methods and are compatible with the beamline element container in the `Beamline` class

`QuadrupoleMagnet, DipoleRectMagnet, DriftSpace, TerminalPlane`

- classes derived from the `Magnet` class
- they reimplement some virtual methods of the `Magnet` class, especially the ones related to transport (some contain a matrix representing the element, constructed based on their parameters)
- they are directly used in the `Beamline` class

`Particles`

- class containing a list of particles to be transported
- public methods for adding/changing/deleting particles and reading them

`settings`

- class for storing global settings, such as the size and scale of the drawing area and colors
- public methods for reading and changing settings

`Transporter`

- class that reads the contents of the `Beamline` and the `Particles` classes
- performs transport and draws results on given pixmap provided by `MainWindow`
- contains a code for data export to the CSV format

`ScaleSettings`

- class for handling the "Scale settings" GUI dialog

`MagnetEditor, ParticleGenerator`

- classes for editing beamline elements and particles, they handle the related GUI dialogs

`TwissImporter`

- class that handles the import of twiss files, containing beamline data

Block diagram

How the classes work together is described on Figure 2.3. Rectangle shapes are classes bound with GUI, rounded rectangles are all other classes and ellipses are files. All arrows indicate the data flow direction.

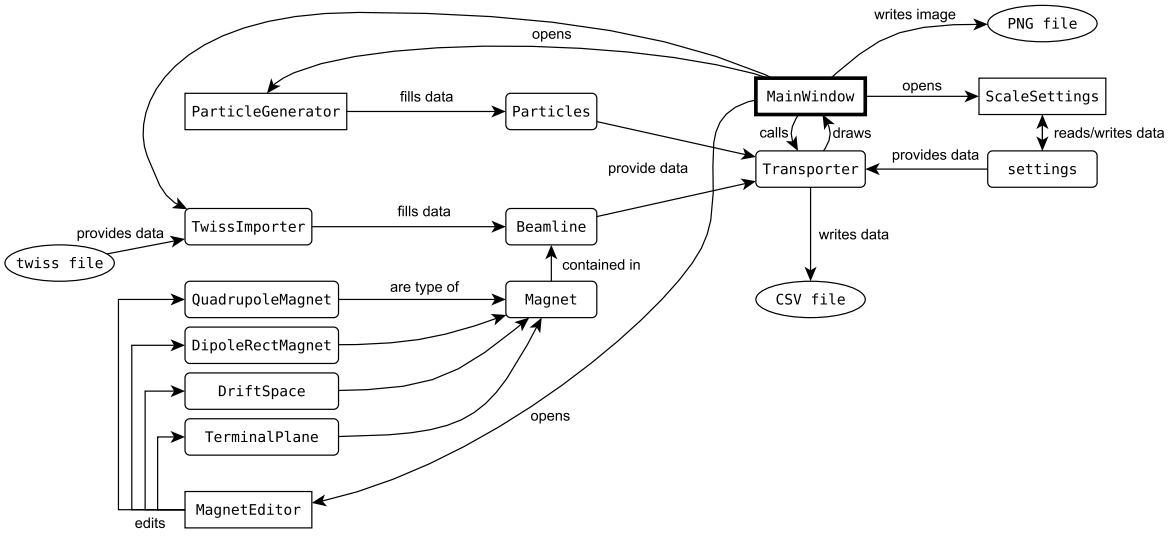


Figure 2.3: Block diagram describing internal relations of classes

2.3.2 Format of data in twiss files

Twiss files are data files containing formatted information about the beamline optical properties and apertures. They are generated by the MAD-X simulation software[4].

Considered my needs, there are many redundant information in the twiss files. In the import routine implemented in the `TwissImporter` class, I consider all unknown beamline elements (other than quadrupole, dipole and drift space) to be a drift space. Error message is printed to `stderr` if the parameters describing the effect of those elements are not zero. Except for some initial information, there is total of 30 columns. I'm interested only in those labeled `KEYWORD`, `S`, `L`, `K0L`, `K1L`, `APERTYPE`, `APER_1`, `APER_2`, `APER_3`, `APER_4`.

Meaning of those labels is explained in the Table 2.1.

[4]See `http://frs.web.cern.ch/frs/Xdoc/mad-X.html` or [3]

[5]Interaction Point

label	explanation
KEYWORD	type of the element
S	distance of the rear end of the element from the IP[5]
L	length of the element
K0L	dipole bending angle
K1L	$k \cdot$ L (for quadrupole)
APERTYPE	type of the aperture
APER_1	parameter of the aperture
APER_2	parameter of the aperture
APER_3	parameter of the aperture
APER_4	parameter of the aperture

Table 2.1: Explanation of twiss file column labels

If the aperture type is NONE, the aperture parameters are not important. If it is CIRCLE, only the first parameter is important and describes the radius of the circle. But the RECTELLIPSE aperture type is slightly more complicated.

Rectellipse aperture

The rectellipse is a special type of aperture which is shaped as an intersection of a rectangle and an ellipse. Mapping of the parameters is shown on the Figure 2.4.

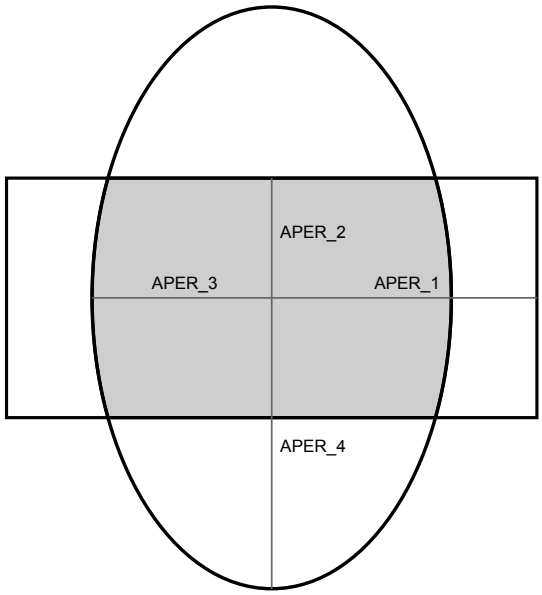


Figure 2.4: The rectellipse shape

This way, simple rectangular or elliptical shapes can be described (if the other shape is large enough that the ellipse or rectangle fits in it) as well as apertures, where the elliptical and rectangular shapes really intersect.

2.3.3 Short usage guide

The main window of the EasyTracker simulation software has two main parts: the side panel and the draw area. The main window can be seen on the Figure 2.5.

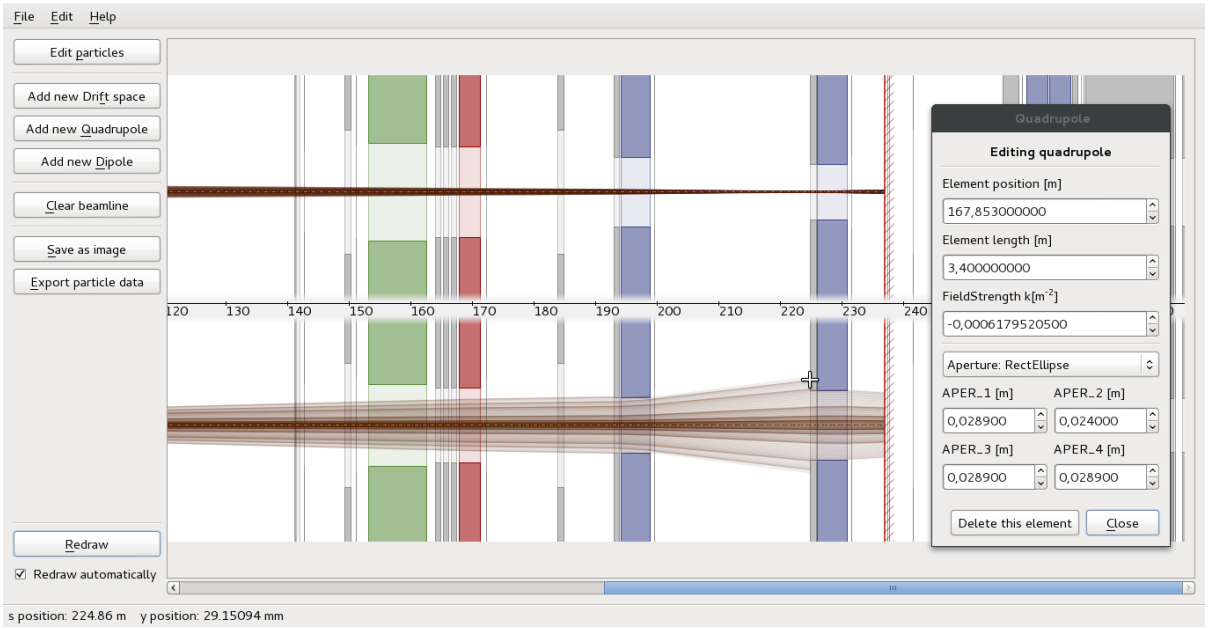


Figure 2.5: Overview of the main window

The side panel of the main window contains buttons to open particles editor, add beamline elements, clear the whole beamline and to export either the image or the particle data. On the bottom is a button which calls simulation and redrawing to be run and a checkbox which forces the same action after any change of input.

In the draw area, beamline elements together with the particle tracks are shown. They are displayed in two cross sections, on the top the horizontal plane (along axes x, s) is shown, while on the bottom is the vertical one (y, s). Colors can be customized, on default dipoles are green, quadrupoles are blue and drift spaces with apertures are gray. Clicking any point in the draw area causes the coordinates of the point to be shown in the statusbar. Clicking while holding down the Ctrl key opens the editing dialog for a nearest magnet, which becomes highlighted (by red color on default). The editing window can be seen on the right side just behind the terminal plane.

When the EasyTracker software is started, there are basically two ways to go. One is to start adding beamline elements manually, the other is to import a twiss file contents. Particles can be added through the Edit particles menu or button, both open the particle generator. Particles can be generated in sets or added/edited one by one. The current particle layout is graphically illustrated, as can be seen on the Figure 2.6. If the particle has non-zero transverse speed, the direction vector is indicated as well.

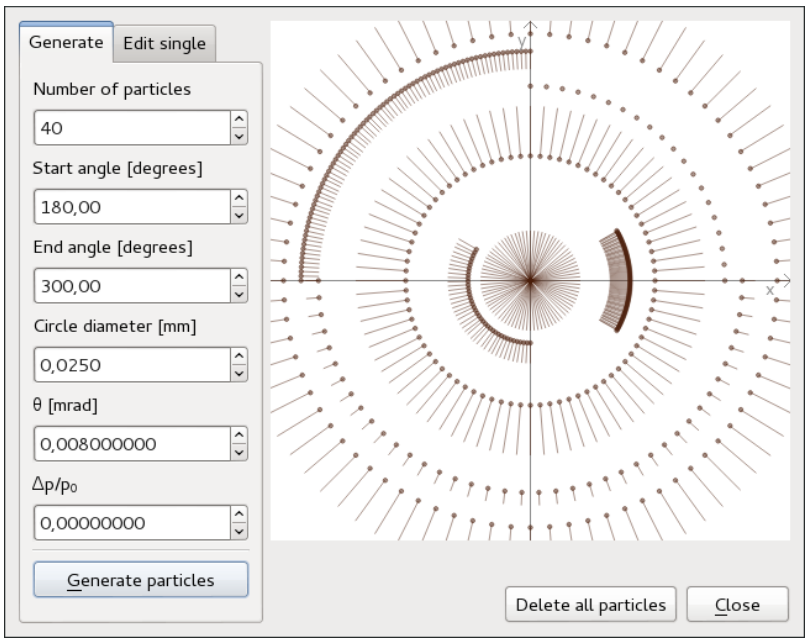


Figure 2.6: Demonstration of use of the particle generator

The visualization range and scale can be edited through the Edit menu, as can be all coloring settings. Twiss files can be imported through the File menu.

When transporting large numbers of particles, it can be useful to untick the "Redraw automatically" checkbox, as it will prevent from brief freezing while parameters are changed. The simulation is quite fast thanks to the relatively simple representation by matrix operations, but transporting more than 10000 particles can take a few seconds to be simulated and rendered.

On the other hand, using several hundred particles and setting higher transparency of particle tracks often results in nice images and helps to see behavior of specific groups of particles very well, just as in the vertical plane on the Figure 2.5.

Known issues

The software is not ideal yet and has a few shortcomings. One of them is that when the zoom is too large, particles from the top part (x, s plane) can run through the axis area and be drawn at the bottom part (y, s plane) and vice versa. This is a known issue and will be fixed in future, as the development of the software will continue.

Chapter 3

Results and verification

3.1 Verifying basic properties

I started with just a simple visual verification at first.

Single quadrupole

The quadrupole magnet should focus in one and defocus in the other axis.

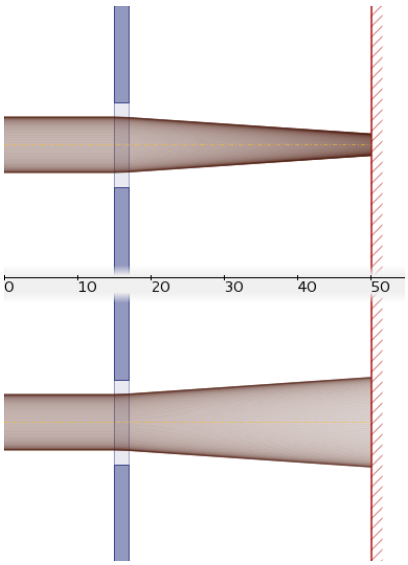


Figure 3.1: Effect of a single quadrupole

On the Figure 3.1, we can see that for $k = 0.009\,\mathrm{m}^{-2}$ and the beam diameter of 30 mm, the quadrupole behaves as expected. It corresponds to the way $k > 0$ was defined as vertically defocusing and horizontally focusing. For negative k the effects in the horizontal and vertical plane are interchanged.

Set of quadrupoles

When a series of quadrupole magnets with alternating focusing and defocusing effect are used, the beam should be focused both in the horizontal and vertical axis.

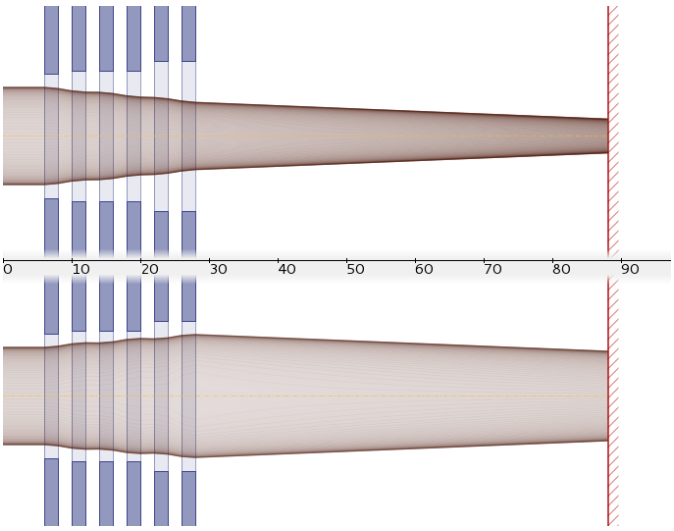


Figure 3.2: Effect of a quadrupole series

The Figure 3.2 shows how the beam is focused in both the horizontal and the vertical plane. All quadrupoles have the $|k| = 0.012\,\mathrm{m}^{-2}$, the sign alternates. Beam diameter 30 mm is the same as in the previous case.

Single dipole

When the beam goes through a rectangular dipole, the effect should be exactly the same as for the nominal particle in the horizontal plane, however in the vertical plane a weak focusing effect should show up.

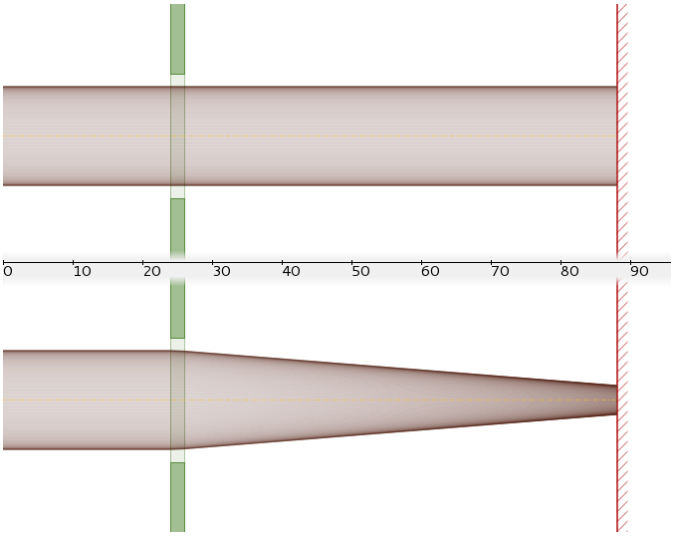


Figure 3.3: Weak focusing effect of a dipole

The Figure 3.3 shows that the beam is focused only in the vertical plane. The bending angle is set to $\varphi = 0.15\,\mathrm{rad}$ to make the focusing easily visible. Beam diameter is still 30 mm.

A dipole bending particles with different momentum

The bending effect of the dipole depends on the momentum of the particle. Trajectory of the particle with a higher momentum is bent less than the nominal particle.

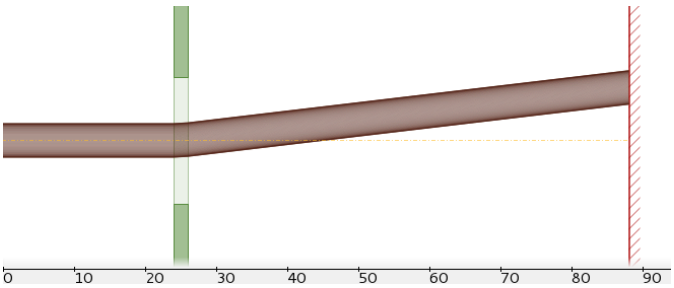


Figure 3.4: Effect of a dipole on a beam with higher energy than the nominal particle

Figure 3.4 shows the effect for $\frac{\Delta p}{p_0} = 0.05$ and the bending angle $\varphi = 5\,\text{mrad}$. It causes the beam to deviate by approximately 16 mm on a 62 m distance. Figure 3.5 shows how the effect can be compensated by another dipole bending the opposite angle $\varphi = -5\,\text{mrad}$, but she offset gained between them remains. We can get rid of the offset by using more than two dipoles, as shown on the Figure 3.6. However, by compensating the deviation angle, we get back to the original nominal trajectory direction as well.

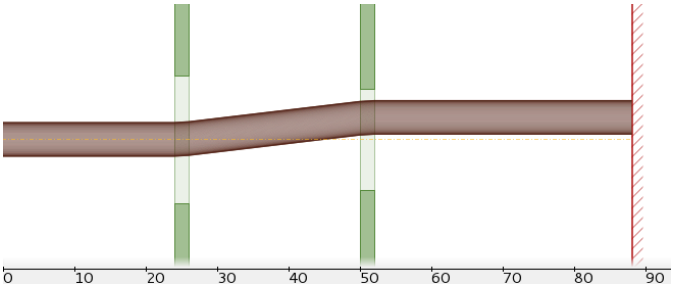


Figure 3.5: Compensation of the deviation angle by another dipole

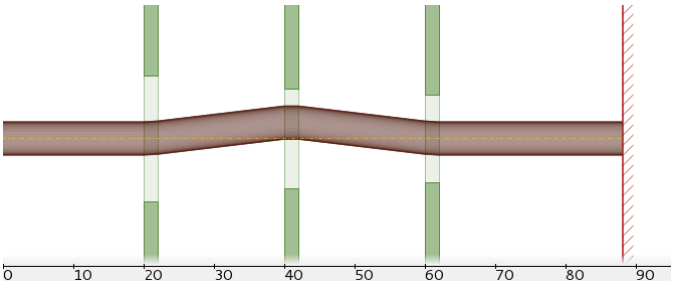


Figure 3.6: Complete elimination of the effect by two other dipoles

Apertures

Apertures are expected to block parts of the beam that are too far away from the nominal trajectory.

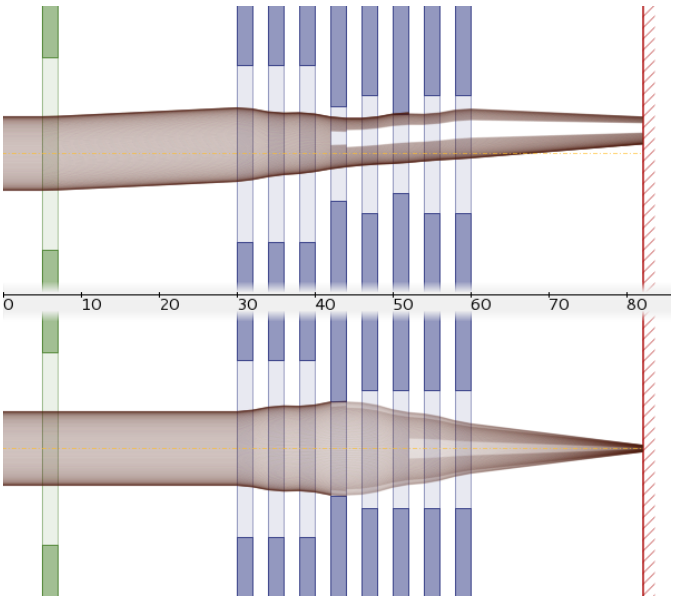


Figure 3.7: Apertures cutting parts of the beam envelope away

On the Figure 3.7 we can see that apertures work as expected. They stop parts of the beam envelope, which do not fit into the circular or rectellipse shape.

Importing twiss file

After importing the twiss file, I expect to see the beamline profile made of various magnets and apertures.

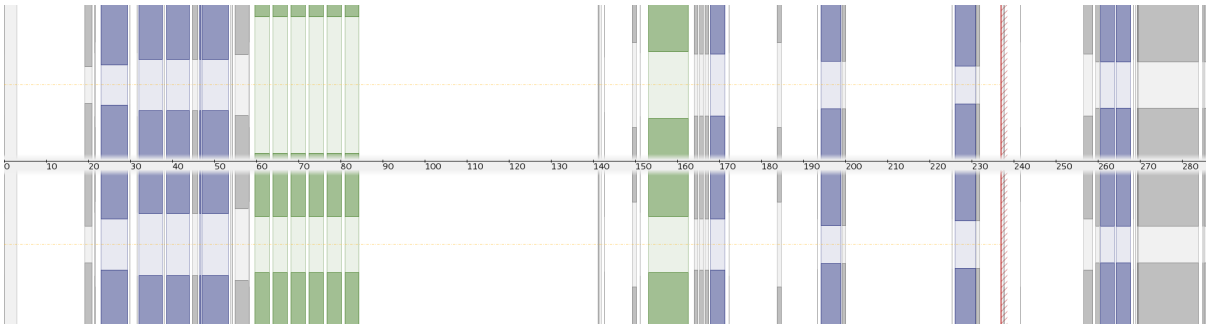


Figure 3.8: The EasyTracker software after a twiss file was imported

The imported beamline is shown on the Figure 3.8. Currently, the import is hardcoded to stop at 300 m by the `#define STOPPOSITION 300.0` statemnet in the `TwissImporter` class.

Transporting particles through the imported beamline

I tried to transport some particles just for the first check that everything works fine. The results are shown on the Figure 3.9, with a transverse range ±25 mm. The beam is focused horizontally near the terminal plane, but vertically it is not to aid forward detectors located there. The particles range up to ±0.1 mm and ±0.05 mrad deviation from the nominal particle in the interaction point.

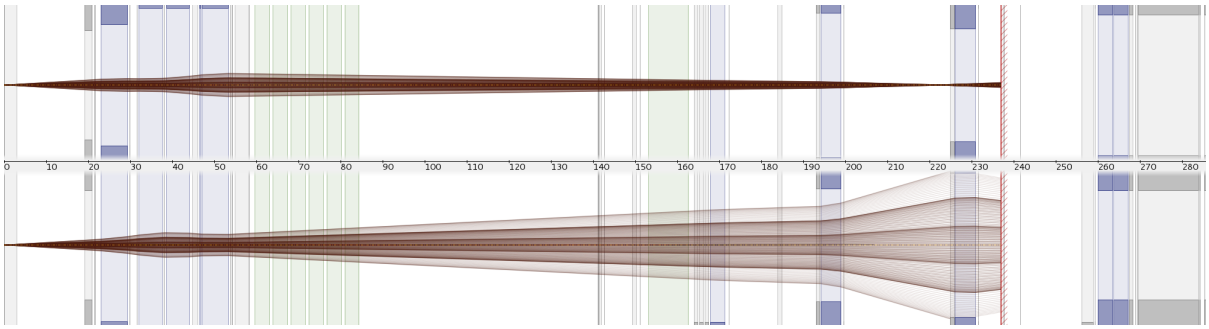


Figure 3.9: Transport of particles through the imported beamline

3.2 Comparison of data with the MAD-X software

I received a set of precomputed data obtained by the MAD-X simulation software. I chose some particles with different parameters for correctness evaluation of my simulation software. The comparison will be done for a terminal plane at $s = 236.888\,\mathrm{m}$.

In all cases, the `alfaTwiss1.txt` file is used. The file is contained in the appendix.

3.2.1 Particles with the same momentum as the nominal particle

As the first case, I verified transport of particles with the $\frac{\Delta p}{p_0}$ being zero. θ is here the deviation angle from the nominal particle trajectory and φ is the angle of direction of the particle in the transverse plane, as explained on the Figure 3.10.

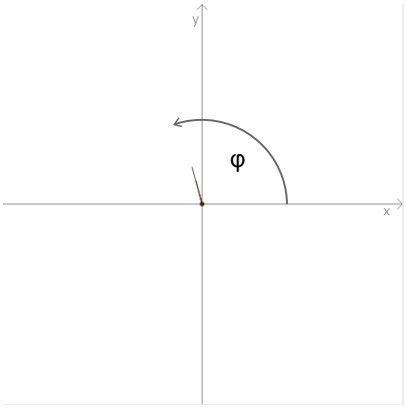


Figure 3.10: Explanation of the φ parameter (shown in the particle generator)

Simulation for $\varphi = 0\,\mathrm{rad}$

Table 3.1 shows the first comparison. The first particle is identical to the nominal particle, the rest has an increasing deviation angle θ. Visualization of this simulation run is shown on the Figure 3.11.

	EasyTracker		MAD-X	
θ	x_0[m]	y_0[m]	x[m]	y[m]
0.0	0	0	0.0000000791	0
0.2	-0.000199161	0	-0.0001990806	0
0.5	-0.000497901	0	-0.0004978202	0
0.8	-0.000796642	0	-0.0007965597	0

Table 3.1: Particles with $\frac{\Delta p}{p_0} = 0$ and $\varphi = 0\,\text{rad}$

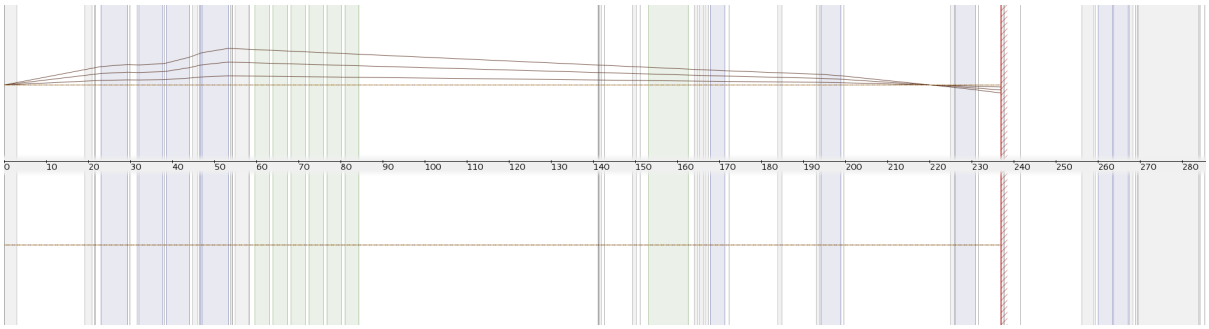


Figure 3.11: Visualisation of the simulation data in Table 3.1

Simulation for $\varphi = 0.52358\,\text{rad} = 30\,^\circ$

Table 3.2 shows the second comparison. Visualization of this simulation is shown on the Figure 3.12.

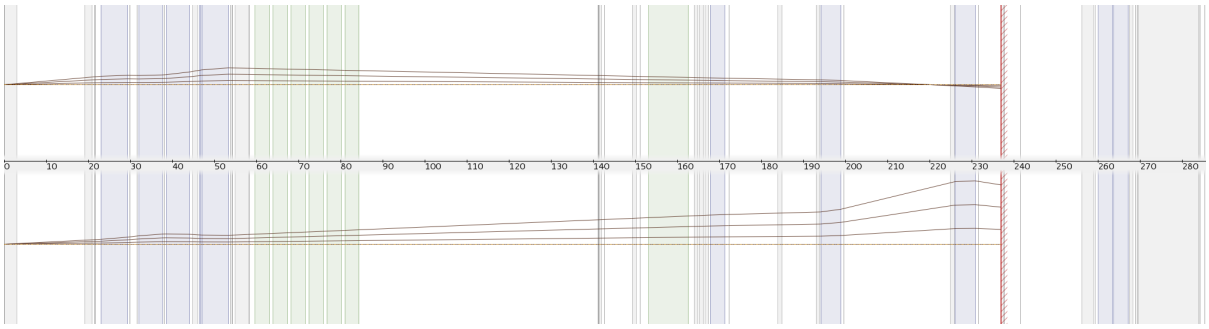


Figure 3.12: Visualisation of the simulation data in Table 3.2

Simulation for $\varphi = 1.83253\,\text{rad} = 105\,^\circ$

Table 3.3 shows the last comparison for $\frac{\Delta p}{p_0} = 0$. For $\theta = 0.8$ the particle hit an aperture at $s = 225.8\,\text{m}$. Both EasyTracker and MAD-X detected the hit and reported shorter final distance, but MAD-X reports coordinates $-99, -99$ in such cases. My software does not throw away that information and writes the coordinates of the hit. Visualization of this simulation run is shown on the Figure 3.13.

[1] $-99, -99$ coordinates are returned by MAD-X on aperture hit

	EasyTracker		MAD-X	
θ	x_0[m]	y_0[m]	x[m]	y[m]
0.0	0	0	0.0000000791	0
0.2	-0.00017248	0.00279370	-0.0001724002	0.002793722
0.5	-0.00043120	0.00698424	-0.0004311191	0.006984304
0.8	-0.00068992	0.01117480	-0.0006898379	0.011174890

Table 3.2: Particles with $\frac{\Delta p}{p_0} = 0$ and $\varphi = 0.52358\,\text{rad}$

	EasyTracker		MAD-X	
θ	x_0[m]	y_0[m]	x[m]	y[m]
0.0	0	0	0.0000000791	0
0.2	0.0000515339	0.00539728	0.0000516	0.005397326
0.5	0.0001288350	0.01349320	0.0001289131	0.01349332
0.8	0.0000620255	0.02265560	-99	-99[1]

Table 3.3: Particles with $\frac{\Delta p}{p_0} = 0$ and $\varphi = 1.83253\,\text{rad}$

Conclusion of the first part of the simulation verification

All simulation cases for $\frac{\Delta p}{p_0} = 0$ proved the EasyTracker software to simulate correctly. The relative error was in most cases under 0.05%, the absolute error was usually about $0.1\mu\,\text{m}$. Based on this, I consider the EasyTracker as a precise simulation tool given the condition $\frac{\Delta p}{p_0} = 0$.

3.2.2 Particles with different momentum compared to the nominal particle

In the second part of the verification process, I checked particles with a lower momentum than the one of the nominal particle, $\frac{\Delta p}{p_0} < 0$.

The nominal particle in the twiss file used has the energy $E = 3.5\,\text{TeV}$. Here I will use particles with energy $E = 3.3\,\text{TeV}$, therefore $\frac{\Delta p}{p_0} = -0.057142857$.

Simulation for $\varphi = 1.83253\,\text{rad} = 105\,^\circ$ and $\frac{\Delta p}{p_0} = -0.057142857$

	EasyTracker		MAD-X	
θ	x_0[m]	y_0[m]	x[m]	y[m]
0.0	-0.000335568	0	0.0000609	0
0.2	-0.000284034	0.00539728	0.0001355152	0.004954171
0.5	-0.000206733	0.01349320	0.0002474828	0.01238543
0.8	-0.001573690	0.02265560	0.0003594503	0.01981668

Table 3.4: Particles with $\frac{\Delta p}{p_0} = -0.057142857$ and $\varphi = 1.83253\,\text{rad}$

Table 3.4 shows the simulation results for particles, which have different momentum than the nominal particle has. It is easy to see that the results are precise in the vertical

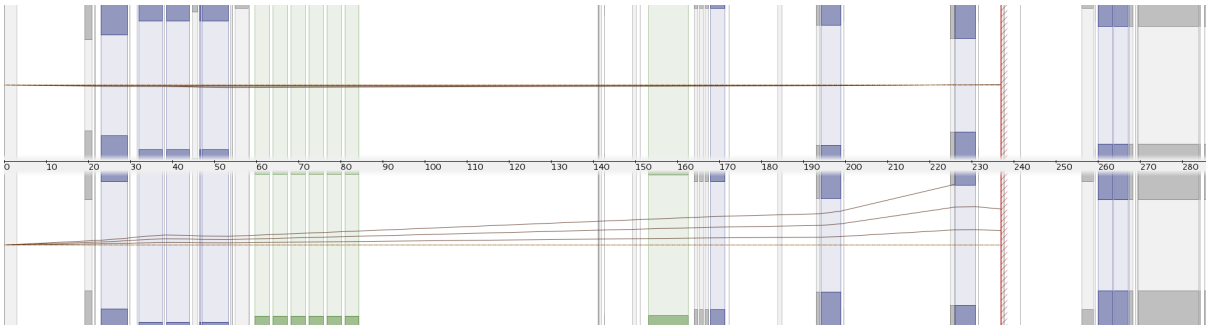


Figure 3.13: Visualisation of the simulation data in Table 3.3

plane, but completely wrong in the horizontal plane. Moreover for the $\theta = 0.8$, the EasyTracker made an aperture hit at $s = 225.8\,\mathrm{m}$, where the MAD-X did not (it does for $\theta = 0.9$).

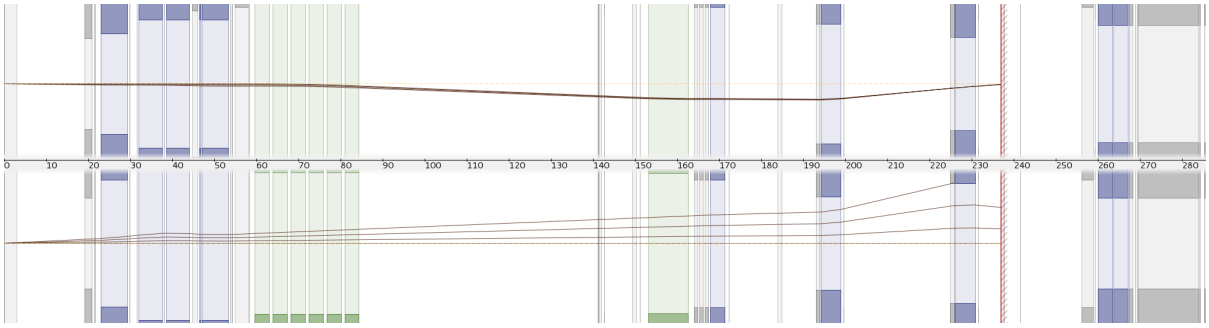


Figure 3.14: Visualisation of the simulation data in Table 3.4

Conclusion of the second part of the simulation verification

After long searches for the source of the problem I failed to find one. It could be caused by incorrect implementation of the dipole magnet, incorrect parsing or wrong understanding of the dipole strength parameter from twiss files, or even by me having wrong twiss files (not corresponding to the beamline with which the MAD-X simulations were run). Anyway, the inaccuracies are most likely caused by dipoles, because they are the only elements included that have momentum dependent effect. The only other option would be that some other elements I ignore can no longer be neglected when simulating with variable momentum.

The EasyTracker is therefore currently not usable for tracking particles with different momentum than the nominal particle momentum.

Summary

During the work on my thesis, I studied the matrix optics approximation describing movement of particles in the magnetic field of a particle accelerator. Elements interacting with the particle (as well as drift space) can be described by a matrix, which operates on a vector describing the particle at a given point in space. Despite its shortcomings, it provides a powerful and very fast way to simulate particle movement.

I designed and implemented a simulation software based on this approximation and evolved it through several stages of development, while learning how to use the Qt framework on the way. The first stage was an isolated dipole, meant to test the very basic concept of the matrix optics. This implementation was abandoned, as it did not provided needed scalability. Instead, new software implementing series of quadrupoles was written from scratch. This concept proved to be well designed and served as a base for the final piece of simulation software.

The software is capable of importing twiss files, data files describing the beamline optical properties. The most important feature of the software is the user's ability to change properties of magnets and see how the trajectories change right away. Called EasyTracker, it is meant to provide an intuitive graphical interface for the simulations. Beamline elements can be selected for editing by clicking them while holding down the Ctrl key.

For verification of the simulation, I used data from the MAD-X simulation software. The simulation was successfully verified for particles with the same momentum as the nominal particle. The results in this case are very precise. However, I failed to verify simulation of particles with different momentum. I suspect that it is caused by some issue with dipoles, whether it would be wrong implementation of the transport routine, wrong understanding of twiss files or mismatch between the MAD-X data and the twiss file used.

The simulation software is currently suitable for visualization of particles with the same momentum as the nominal particle. The development of the software will continue with the future goal of stepping out of the matrix optics approximation and hopefully providing a complete transport simulation capability.

Bibliography

[1] SCHMÜSER, P.: *Basic course on accelerator optics*. Aarhus, Denmark: CERN ACCELERATOR SCHOOL, September 1986.

[2] WILLE, K.: *The Physics of Particle Accelerators: An Introduction*. Oxford University Press, 2000

[3] HERR, W.: *A MAD-X Primer*, available online: `http://cds.cern.ch/record/744163/files/p505.pdf`

List of used symbols

$\vec{B}$	magnetic induction
$\vec{v}$	velocity
$\vec{F}$	force
$\vec{E}$	electric field
E	particle energy
q	electric charge
c	speed of light
ρ	trajectory radius
m	mass
e	elementary charge
x, y	transverse coordinates
x', y'	change of transverse coordinates per meter/approximated deviation angle
p	momentum of particle
p_0	momentum of nominal particle
$\frac{\Delta p}{p_0}$	relative momentum difference
M_x	matrix describing an element in the horizontal plane
M_y	matrix describing an element in the vertical plane
S_0	point before transport
S	point after transport
s	longitudinal coordinate
θ	deviation angle
l	length of beamline element
k	quadrupole strength coefficient
φ	dipole bending angle, later particle direction in the transverse plane
IP	Interaction Point
GUI	Graphical User Interface

Appendix

Contents of enclosed CD-ROM

```
alfaTwiss1.txt
alfaTwiss2.txt
Binaries-Linux-x86_64
Bp_Komarek.pdf
EasyTracker_Source
QuadSeries_Source
SampleImage_1.png
SampleImage_2.png
SampleImage_3.png
SingleDipole_Source

./Binaries-Linux-x86_64:
EasyTracker
QuadSeries
SingleDipole

./EasyTracker_Source:
beamline.cpp
beamline.h
constants.h
dipolerectmagnet.cpp
dipolerectmagnet.h
driftspace.cpp
driftspace.h
EasyTracker.pro
EasyTracker.pro.user
icon.png
images.qrc
magnet.cpp
magneteditor.cpp
magneteditor.h
magneteditor.ui
magnet.h
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
Makefile
particlegenerator.cpp
particlegenerator.h
particlegenerator.ui
particles.cpp
particles.h
quadrupolemagnet.cpp
quadrupolemagnet.h
scalesettings.cpp
scalesettings.h
scalesettings.ui
settings.cpp
settings.h
terminalplane.cpp
terminalplane.h
transporter.cpp
transporter.h
twissimporter.cpp
twissimporter.h

./QuadSeries_Source:
beamline.cpp
beamline.h
constants.h
magnet.cpp
magnet.h
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
particles.cpp
particles.h
quadrupolemagnet.cpp
quadrupolemagnet.h
QuadSeries.pro
```

```
QuadSeries.pro.user
scalesettings.cpp
scalesettings.h
scalesettings.ui
settings.cpp
settings.h
transporter.cpp
transporter.h

./SingleDipole_Source:
main.cpp
mainwindow.cpp
mainwindow.h
mainwindow.ui
paintjob.cpp
paintjob.h
SingleDipole.pro
SingleDipole.pro.user
transport.cpp
transport.h
```