Flight Path Simulation Application

A flight simulator for charged particle transport

Ulf Bylander
Abstract

Flight Path Simulation Application

Ulf Bylander

CTF3 is a test facility for a new CLIC high energy linear collider. For this beam steering and beam focusing is vital. Because physically running a beamline and changing setup is expensive and takes much effort it is beneficial to use a simulator for the beamline. The transportation of the beam through the beamline can be represented with matrix multiplications and for this reason MATLAB is a fitting environment to simulate in. A Flight Path Simulator was written in MATLAB and was successfully implemented and tested for the CALIFES beamline of the two-beam test stand that is part of the CTF3 facility.
# Table of contents

Abstract .......................................................................................................................... 3

Introduction ....................................................................................................................... 6
  High Energy Beams ........................................................................................................ 6
  CLIC/CTF3 ...................................................................................................................... 6
  Flight Path Simulator .................................................................................................... 6

Objective ............................................................................................................................ 7

Theory: Linear Optics ....................................................................................................... 7
  Beam Representation ..................................................................................................... 7
  Matrix Representation .................................................................................................. 8
    Equation of motion ..................................................................................................... 8
    Quadrupole ................................................................................................................ 8
    Dipole ......................................................................................................................... 10
    Kick ............................................................................................................................. 12
  Beam Diagnostics ......................................................................................................... 12
  Other multipoles .......................................................................................................... 12
  Beam Bunch Distribution ............................................................................................... 12
    Distribution Moments ............................................................................................... 14
  Beam Propagation ......................................................................................................... 14
    Transport of first moment ......................................................................................... 14
    Transport of second moment ..................................................................................... 15
    Transporting the beam .............................................................................................. 15

Method ............................................................................................................................... 15
  MATLAB ....................................................................................................................... 15
  Application .................................................................................................................... 16
    Data structure ............................................................................................................ 16
      Component list ........................................................................................................ 16
      Initial Beam .............................................................................................................. 16
      Transported beam data ........................................................................................... 16
    Algorithms .................................................................................................................. 17
      Component list creation ......................................................................................... 17
      Transportation Algorithm ........................................................................................ 17
      Graphical Representation ......................................................................................... 18

Results ............................................................................................................................... 18
  Features ......................................................................................................................... 18
### Accuracy

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTF3 two-beam test stand comparison</td>
<td>20</td>
</tr>
<tr>
<td>Other_</td>
<td>21</td>
</tr>
</tbody>
</table>

### Discussion

<table>
<thead>
<tr>
<th>Discussion</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTF3 test comparison</td>
<td>22</td>
</tr>
<tr>
<td>Limitations</td>
<td>22</td>
</tr>
<tr>
<td>Alternative solutions</td>
<td>22</td>
</tr>
<tr>
<td>Further development</td>
<td>22</td>
</tr>
</tbody>
</table>

### Conclusion

<table>
<thead>
<tr>
<th>Conclusion</th>
<th>23</th>
</tr>
</thead>
</table>

### References

| References | 23 |

### Appendix

<table>
<thead>
<tr>
<th>Appendix</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>User manual</td>
<td>23</td>
</tr>
<tr>
<td>Component text file format</td>
<td>23</td>
</tr>
<tr>
<td>Beam text file format</td>
<td>25</td>
</tr>
<tr>
<td>GUI</td>
<td>25</td>
</tr>
<tr>
<td>The CTF3 two-beam test stand beamline</td>
<td>26</td>
</tr>
</tbody>
</table>

---
Introduction

High Energy Beams

In the field of particle physics one studies the processes of the fundamental particles. Many of these processes require high energies to occur and a particle accelerator is a way of obtaining these energies. High energy beams are by necessity charged in order to be accelerated, since accelerators use electrical forces to accelerate their beams. Charged beams can also be manipulated by magnetic and electric forces and can especially be steered and focused with magnets. Running a particle accelerator is very expensive and changing settings can take a lot of effort. That’s why a simulator can be a great help to save both time and money. The theory behind beam steering can also be used for other beamlike applications, charged or not. Examples for other applications are controlling X-rays, proton beams and LASERS.

CLIC/CTF3

CLIC (Compact Linear Collider) is a project studying a future electron-positron collider. CLIC is an international collaboration spanning over 30 countries and involving over 70 institutes, one of them being Uppsala Universitet. The collider will be operating at a centre-of-mass energy range of 0.5-3 TeV higher than the previous highest energy lepton collider, which operated at 209 GeV as highest point. To get these energy levels at a reasonable price the acceleration gradient, which is the energy gained by the beam per meter, must be very high. The currently used technology is limited to lower accelerations, whereas CLIC utilizes a secondary drive beam to induce accelerations in the range of 100 Mev/m in a main beam.

CTF3 is a test facility for a CLIC accelerator (CLIC Test Facility 3). The facility houses a two-beam test stand to experiment with the drive beam solution for a CLIC accelerator. It was built and is operated by CERN, CEA Saclay, and Uppsala Universitet under the department of high energy physics.

![Qualitative diagram of the CTF two-beam test stand.](image)

Flight Path Simulator

In order to conveniently tune the beam line virtually an interactive flight path simulator is highly desirable. The purpose of the simulator is to calculate the trajectory, or flight path, of a beam traveling through a setup of magnets, also called a beam line. The flight path simulator will need to be able to load a setup in the form of a list of components. Each component describes one of five types of steering magnets or an empty section. The simulator will also need to load a starting state of a beam and then transport the beam through the magnets to calculate how the beam is affected.
Objective

The objectives of the project can be condensed to:

- Load beam data
- Load component data
- Transport beam data
- Plot transported beam data
- Interactive graphical user interface

Theory: Linear Optics

In this section we will be looking at the mathematical theory behind the application: Linear optics. Linear optics might be recognized from its origin as the theory of how light rays behave in lenses. Its applications have been discovered to be wider than just handling light rays and covers areas such as error propagation and specifically propagation of charged beams. Linear optics is used for particle accelerator computations and its software all over the world today, much because of its tested accuracy and compatibility with computers.

Beam Representation

A single particle in a beam is represented with its deviation from a reference particle in an ideal trajectory. The deviations commonly of interest are: the space displacement from the ideal trajectory, the rate of change for the displacement, the difference in arrival time and the difference in momentum. These properties are arranged in a vector denoted respectively (note that there are two space dimensions)

\[
\vec{x} = \begin{pmatrix} x \\ x' \\ y \\ y' \\ \tau \\ \delta \end{pmatrix}
\]

Figure 2, A particle traveling in the Z direction. The particles motion can be completely described by two parameters.

Example 1: a particle 0.002m of x-centre, going slightly angled from the centre, and 1 nanosecond ahead of a bunch would be
Matrix Representation

How a beam is affected when moving through most elements found in an accelerator is beneficially expressed as a matrix transforming the \( \mu \)-vector (Eq. 1) for the beam. The simplest case is an empty part of an accelerator called a drift sector or drift space. This section should only affect the displacement of the particle and every length unit of drift space should add \( x' \) to the displacement. Looking at only \( x \) and \( x' \):

\[
\begin{pmatrix}
1 & L \\
0 & 1
\end{pmatrix}
\begin{pmatrix}
x \\
x'
\end{pmatrix}
= \begin{pmatrix} x + Lx' \\
x'
\end{pmatrix}
\]

Eq. 2

Considering the complete state representation, the drift matrix will be:

\[
D = \begin{pmatrix}
1 & L & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & L & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

Quadrupole

To focus the beam a quadrupole magnet is used. The quadrupole magnet creates a magnetic field with a gradient in the transverse directions through the beam line. This causes particles further away from the reference trajectory to get a larger kick in \( x' \), acting like a lens in light optics. This results in a focusing or defocusing effect on the beam depending on the sign of the gradient. Note that a quadrupole has opposite gradient signs in perpendicular directions, so focusing in one direction means defocusing in the other. This is caused by Maxwells equation \( \vec{\nabla} \times \vec{B} = 0 \).

Approximating the magnet as thin we can consider it a thin lens for demonstration purposes.

\[
\begin{array}{c}
\text{Figure 3, two particles traveling in the } z \text{ direction. They are focused by a lens to a focus point.}
\end{array}
\]
Looking at the magnet as a thin lens and knowing it should increase \( x' \) linearly with \( x \) we get the form, where \( f \) is the focal length:

\[
Q = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
-\frac{1}{f} & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{f} & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

For a thick quadrupole magnet one needs the equation of motion for a charged particle within a magnet. The equation of motion determines the trajectory of a particle traveling through the magnetic field of a magnet and is as follows\(^1\) (for a complete discussion see Wille, K. (2005))

\[
x''(L) + \left( \frac{1}{R^2} + k \right) x(L) = 0
\]

Where \( L \) is the traveling direction, \( k \) is a magnetic field gradient constant and \( R \) is a bending radius constant. A quadrupole does not bend the beam so \( 1/R = 0 \). The solution to this equation for values of \( k > 0 \) and \( 1/R = 0 \) is:

\[
x = x_0 \cos(\sqrt{k}L) + x'_0 \frac{1}{\sqrt{k}} \sin(\sqrt{k}L)
\]

\[
x' = -x_0 \sqrt{k} \sin(\sqrt{k}L) + x'_0 \cos(\sqrt{k}L)
\]

And for \( k < 0 \);

\[
x = x_0 \cosh(\sqrt{k}L) + x'_0 \frac{1}{\sqrt{k}} \sinh(\sqrt{k}L)
\]

\[
x' = x_0 \sqrt{k} \sinh(\sqrt{k}L) + x'_0 \cosh(\sqrt{k}L)
\]

Here \( k \) is proportional to the magnetic field gradient and the energy of the beam. This means that \( k \) will have opposite signs for the \( x \) and \( y \) direction, leaving us with a complete transfer matrix\(^2\):

\[
Q = \begin{pmatrix}
\cos(\sqrt{k}L) & \frac{1}{\sqrt{k}} \sin(\sqrt{k}L) & 0 & 0 & 0 & 0 \\
-\sqrt{k} \sin(\sqrt{k}L) & \cos(\sqrt{k}L) & 0 & 0 & 0 & 0 \\
0 & 0 & \cosh(\sqrt{k}L) & \frac{1}{\sqrt{k}} \sinh(\sqrt{k}L) & 0 & 0 \\
0 & 0 & \sqrt{k} \sinh(\sqrt{k}L) & \cosh(\sqrt{k}L) & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

Where \( L \) is the length of the magnet and \( k \) is derived from the magnets and beams physical properties; \( k = \frac{\partial B}{\partial x} / B \rho \), \( B \rho \) being a measurement of the energy of the beam.
Dipole
The main purpose of a dipole magnet is to steer the reference trajectory of the beam. This is achieved with a constant magnetic field that increases/decreases $x'$ uniformly over all $x$.

Sector bend
There are two types of thick dipoles and the first one is called a sector bend. To derive its matrix we must first consider its effects on the beam. The sector bend has a weak focusing effect and a dispersion effect in addition to its bending.

The focusing effect only applies to the bending plane, the other is untouched. The equation of motion again is:

$$x''(L) + \left(\frac{1}{R^2} + k\right)x(L) = \frac{\Delta p}{kp}$$

Where $L$ is the traveling distance, $k$ is a magnetic field gradient constant, $R$ is the bending radius for the magnet, and $p$ is the momentum. Often $k$ is zero for a dipole and $R \neq 0$, and assuming for this discussion that $\Delta p = 0$ gives us the following solutions:
For the dispersion we consider $\Delta p \neq 0$ and look at two charged particles in a uniform magnetic field. One of the particles has a higher velocity than the other. Both particles will adopt circular orbits but the faster particles orbit will have a bigger radius. In this way, a bending magnet will bend particles with less momentum more efficiently. This is accounted for in the 5th and 6th rows and columns of the transfer matrix $^2$:

$$
SB = \begin{pmatrix}
\cos \left( \frac{1}{R} L \right) & R \sin \left( \frac{1}{R} L \right) & 0 & 0 & 0 & R \left( 1 - \cos \left( \frac{1}{R} L \right) \right) \\
-\frac{1}{R} \sin \left( \frac{1}{R} L \right) & \cos \left( \frac{1}{R} L \right) & 0 & 0 & 0 & \sin \left( \frac{1}{R} L \right) \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
-\sin \left( \frac{1}{R} L \right) & -R \left( 1 - \cos \left( \frac{1}{R} L \right) \right) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
$$

![Figure 6](image-url)

Two particles traveling to the right. Their different energies cause their trajectories to divert.

**Rectangular bend**

The Rectangular bend is easier to manufacture and is modeled as a sectorbend multiplied with an extra focusing matrix on each side. This extra focus matrix is needed to represent the fact that the inner side of a beam hits the magnetic field of the magnet before the outer side, and exits the field earlier in the same manner.

$$
E = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
\tan \left( \frac{1}{2R} \right) & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & -\tan \left( \frac{1}{2R} \right) & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
$$
With the complete sector bend transfer matrix$^2$,

$$RB = E \cdot SB \cdot E$$

Figure 7, three particles traveling through a rectangular bend. The inner most particles encounter the bend earlier then the outermost.

**Kick**

A thin dipole is commonly called a steerer and and is used for local displacement of the beam trajectory. A kicker should only increase or decrease $x'$ directly and cannot be expressed as a matrix like the other components. There are some ways to get around this as can be seen in the methods section of this report.

**Beam Diagnostics**

One needs to employ a series of different methods to acquire the distribution data. A Beam position Monitor is one of the more common tools and is used to find $\mu_x$ and $\mu_y$. A common type of BPM measures the beams center-of-mass by measuring variations in an electric field held over the beam. The $\sigma$-matrix is difficult to measure and requires multiple measurements of the cross section of the beam under different settings on a screen. Such a screen, which often contains fluorescent material, intercepts the beam and is viewed by a camera.

**Other multipoles**

There are more magnet types used to steer a charged beam, for example sextupoles and octopoles. But their effect on the beam is more subtle and won’t be needed in this application.

**Beam Bunch Distribution**

It is not practical to follow every single particle in a beam bunch since they number in the millions. It is easier to consider the distributions of the particle properties instead. This is easiest described
with an example. Consider a series of small detectors arranged at different $x$ coordinates. Every
detector count how many particles pass their coordinate and after a time, or an integration over time,
the number of counts on each detector will have a certain distribution. This distribution tells us how
many particles has a specific $x$ displacement, or more accurately how big proportion of particles is
within an interval of $x$ displacement.

The complete distribution over all properties is

$$
\Phi = \Phi(x, x', y, y', \tau, \delta)
$$

Integrating this over an interval tells us how big proportion of the particles populates that subspace.

Example:

To get how many particles are within $x=0$ and $x=1$ we integrate over all other properties and form
$x=0$ to $x=1$;

$$
\int_{x'}^{x=1} \int_{y'}^{y=0} \int_{\tau}^{\tau=1} \int_{\delta}^{\delta=0} \Phi(x, x', y, y', \tau, \delta) dx' dx' dy' dy' d\tau d\delta
$$

This distribution is often modeled as a Gaussian distribution, since the beam is affected by many
different random forces. This means that any single property has a one dimensional Gaussian
distribution;

$$
\varphi(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2}}
$$

Where $\mu$ is the mean value or first moment;

$$
\mu_x = \langle x \rangle
$$

The first moment can be seen as the centre of mass for the beam property. In addition, the
complete distribution is a six dimensional Gaussian distribution whereas an n-dimensional
Gaussian distribution has the following form:

$$
\Phi(\vec{x}) = \frac{1}{2\pi^{n/2}\sqrt{|\sigma|}} e^{-\frac{1}{2} \sum_{i,j=1}^{n} (\sigma^{-1})_{ij} (x_i - \mu_i)(x_j - \mu_j)} \quad \text{Eq. 3}
$$

Where we have introduced the $\sigma$-matrix;

$$
\sigma = \begin{pmatrix}
\sigma_{xx} & \sigma_{xx'} & \cdots \\
\sigma_{x'x} & \sigma_{x'x'} & \cdots \\
\vdots & \vdots & \ddots
\end{pmatrix}
$$

And where $\sigma$ is the square of the variance or the second moment;

$$
\sigma_{ij} = \langle (x_i - \mu_i)(x_j - \mu_j) \rangle
$$

The second moment can be seen as the width of the beam.
Figure 8, qualitative shape of a normalized Gaussian distribution. This distribution has $\mu=0$ and $\sigma=1$.

Distribution Moments

The beam can be modeled as a multivariable Gaussian distribution as seen in the previous section, Eq. 3.

This means we only need the $\sigma$-matrix and a $\mu$-vector to describe a beam. In the case of a beam with six variables will look like the following:

$$
\hat{\mu} = \begin{pmatrix}
\mu_x \\
\mu_x' \\
\mu_y \\
\mu_y' \\
\mu_z \\
\mu_z'
\end{pmatrix}
$$

$$
\sigma = 
\begin{pmatrix}
\sigma_{xx} & \sigma_{xx'} & \sigma_{xy} & \sigma_{xy'} & \sigma_{xt} & \sigma_{xt'} \\
\sigma_{xx'} & \sigma_{xx''} & \sigma_{xy'} & \sigma_{xy''} & \sigma_{xt'} & \sigma_{xt''} \\
\sigma_{xy} & \sigma_{xy'} & \sigma_{yy} & \sigma_{yy'} & \sigma_{yt} & \sigma_{yt'} \\
\sigma_{xy'} & \sigma_{xy''} & \sigma_{yy'} & \sigma_{yy''} & \sigma_{yt'} & \sigma_{yt''} \\
\sigma_{xt} & \sigma_{xt'} & \sigma_{yt} & \sigma_{yt'} & \sigma_{tt} & \sigma_{tt'} \\
\sigma_{xt'} & \sigma_{xt''} & \sigma_{yt'} & \sigma_{yt''} & \sigma_{tt'} & \sigma_{tt''}
\end{pmatrix}
$$

Beam Propagation

Transport of first moment

Transferring the first moment with a transfer matrix:

$$
\hat{x}_i = \sum_j M_{ij} x_j
$$

$$
\hat{\mu}_i = \langle \hat{x}_i \rangle = \sum_j M_{ij} \langle x_j \rangle = \sum_j M_{ij} \mu_j
$$

We see that the first moment transfers the same way as a $\mu$-vector;

$$
\hat{\mu} = M\hat{\mu}
$$
Transport of second moment

The second moment
\[ \tilde{\sigma}_{ij} = \langle \tilde{x}_i \tilde{x}_j \rangle = \sum_k M_{i,k} \sum_l M_{j,l} \langle \tilde{x}_k \tilde{x}_l \rangle = \sum_k M_{i,k} \sum_l M_{j,l} \langle \tilde{x}_k \tilde{x}_l \rangle = \sum_k M_{i,k} \sum_l M_{j,l} \sigma_{kl} \]

Where the last step in matrix form takes the following shape:
\[ \tilde{\sigma} = M \sigma M^T \]

Transporting the beam

A beam consisting of a large collection of particles can be modelled as a Gaussian distribution and can be represented with a \( \mu \)-vector and a \( \sigma \)-matrix.

\[ \begin{align*}
\tilde{\mu} &= \begin{pmatrix} \mu_x \\ \mu_y \\ \vdots \end{pmatrix} \\
\sigma &= \begin{pmatrix} 
\sigma_{xx} & \sigma_{yx} & \sigma_{xy} & \vdots \\
\sigma_{yx} & \sigma_{yy} & \sigma_{xy} & \vdots \\
\sigma_{xy} & \sigma_{yx} & \sigma_{yy} & \vdots \\
\vdots & \vdots & \vdots & \ddots 
\end{pmatrix}
\end{align*} \]

Components in an accelerator can be modelled as transfer matrices.

\[ D = \begin{pmatrix} 
1 & L & 0 & \vdots \\
0 & 1 & 0 & \vdots \\
0 & 0 & 1 & \vdots \\
\vdots & \vdots & \vdots & \ddots 
\end{pmatrix} \]

How the \( \mu \)-vector and \( \sigma \)-matrix changes when passing through a component is calculated with matrix multiplication.

\[ \begin{align*}
\tilde{\mu} &= M \tilde{\mu} \\
\tilde{\sigma} &= M \sigma M^T
\end{align*} \]

Method

In this section, the method of solving the stated problems will be described. First let’s recall the purpose and challenges of this application. The application will need to load the data for the components and beam from separate text files. The application then must calculate the trajectory of the beam through the given components. Finally the application will draw this information into graphs. The application will only need to handle a limited amount of different sorts of components and the calculations will be handled with matrices.

MATLAB

MATLAB is a well suited language, since the transportation of the beam can be modelled as matrix transformations and MATLAB is based on matrix calculations. MATLAB also is a familiar environment for the specific users at CTF3 and the application may be available for students in the
future, who also is familiar with MATLAB. It is a familiar language for me as well and has a good support base. An application in MATLAB will not be limited to any specific platforms. A drawback that can be discussed is that it is licensed software.

Application

MATLAB as a language is not strictly object oriented but has something called a struct that is used in the application. A struct is a simpler form of a class that has no methods or functions but a series of fields containing a structs different data. Each field is identified with a name and is created by assigning the field a value, for example struct.name = ‘Jan’.

Another datatype used in the application is a cell. A cell has fields like a struct but the fields has an index instead of a name, which is useful for storing series of matrices and handling them in index loops.

The data for the application is separated in three parts: A list of the components in the beam stored as an array of structs, the initial data of the beam also stored as a struct and the transported beam data stored in two large cells.

The transported beam data is calculated with the transport algorithm explained below and represented graphically in the GUI.

Data structure

Component list

The component list is an array of structs that contains the information of each component in the beam line. The list is loaded from a text file which lists its type, length and appropriate variable such as magnetic field gradient. The information saved in each struct is the transfer matrix, name, length and the cumulative transfer matrix. The length is needed to map the data correct in the trajectory direction. The cumulative matrix for a component list member is the product of all previous list members transfer matrices and is useful for quicker calculations.

Initial Beam

The beam data is stored in a struct with three fields. The fields contain the \( \mu \)-vector, the \( \sigma \)-matrix and a BPM identifier for the beam. The BPM identifier is used to note where the beam centroid is originated. The beam data is loaded from a textfile containing the data for the three previously mentioned fields with the addition of an emittance value if it is needed. The initial beam data is used to calculate the transported beam data but is otherwise untouched through the process.

Transported beam data

The transported beam data contains the \( \mu \)-vectors and \( \sigma \)-matrices of the transported beam in a cell each. The transported beam data also needs a vector to keep track of the mapping; a vector that tells us which cell element goes to what position in the beam line. The transported beam data is calculated from the initial beam data and the component list using the transportation algorithm. To plot the data it is necessary to first extract the data out of these cells, which can be done with a simple extraction method.
Algorithms

Component list creation
A component list is loaded from a text file. The text file lists a components type, length and two physical quantities needed to define the component. A matrix representation is then constructed for every entry in the text file and added to a component list. The matrix constructor reads what type of component should be constructed and builds an appropriate matrix. These matrices are the transfer matrices discussed in previous section with an added row and column in order to handle the effect of a kick magnet. The cumulative matrices is then calculated and added to the list as its own field.

Handling the kick
A dipole magnet that only affects the trajectory angle of the beam cannot be expressed in the 6 by 6 matrix format. In order to handle this in the application an extra row and column is added. The kick matrix will be an identity matrix except for column 7 where it will have a kick value θ in row 2 and another in row 4. The beam μ-vector has an added element set to 1 and the beam sigma matrix has an extra row and column set to zero. The transfer matrix will then only affect the μ-vector.

Transportation Algorithm
The algorithm iterates through every component on the list. If it is not a drift section; it notes the length of the component and jumps to the next section. Otherwise it divides the drift section into several shorter segments and step-by-step calculates the σ-matrix and μ-vector for those points and finally notes the points coordinates into a vector. The calculated σ-matrices are stored in cells in a larger struct for later use. The cumulative matrix is used for the calculation of every new section in the iteration to avoid buildup of errors.

Figure 9, flowchart for component loading function.
Figure 10, flowchart for beam transportation calculation.

Graphical Representation

For a graphical representation three plots are used. The first and second plot presents the flight path over the extent of the beam and the third plot presents the cross section. A point along the beam must first be selected to be able to plot the cross section. The flight path drawn by first drawing the centerline from the x and y values in the μ-vectors of the transported beam data, secondly two lines are drawn corresponding to x+√σx and x-√σx to get a sense of the width of the beam. More precisely to get one standard deviation of the beam distribution.

Results

In this section we will look at the finished application. The application has been showing generally good results and stability. It can successfully load both component data and beam data from text files. The beam translation calculation is successfully implemented and the calculated data is successfully plotted into the GUI. The GUI is useful and understandable, but there is need for a manual on how to represent the data in the text files. Some extra features were added such as cross section plot.

Features

The GUI features eight elements and a file menu. These elements are two buttons, three graphs, two tables, and a checkbox. The first button is an update button to tell the application to calculate or recalculate the transportation of the beam and plot that. The second button tells the application to calculate the cross section and plot that. The lowermost graphs are the plots for the flight paths of the
beam; the x direction above the y direction. At the bottom of the first graph is a qualitative model of the beam line drawn. The model consists of elevated boxes for focusing quadrupoles, lowered boxes for defocusing quadrupoles, midlevel boxes for other components and a big box with data for a BPM component. The third and topmost graph is for the cross section. The rightmost table presents the component list currently active and the last table show the $\sigma$-matrix for the beam at the current cross section. The checkbox is for activating “datacursor” mode, which allows the user to select a data point in the beam graphs. This data point determines where the cross section is calculated.

User defined components are also possible. The component data load algorithm will look for other component functions if the type in the text file loaded doesn’t match any of the preexisting ones. If the function found takes the right input and gives the right output it is used. An example user defined component is included in the application package.

Figure 11, graphical user interface for application.
The accuracy of the application was tested by running a similar application written in FORTRAN and comparing the data results. The beamline used in the two applications was the CTF3 two-beam test stand.

The trajectory calculated by the MATLAB application can be seen in figure 13 and the corresponding data for the FORTRAN application can be seen in figure 14.

Figure 13, CTF3 two-beam test stand data from the MATLAB application with data points a, b, c, and d.
The data compared is the magnitude of the sigma values at two different points in the beam. Points ‘a’ and ‘c’ marks the very beginning of the beam. At point ‘a’ the FORTRAN application presents a sigma value of just above 1.8 mm and the MATLAB application presents a sigma value of just above 1.8 mm as well. At point ‘c’ the FORTRAN application presents a sigma value of (only looking at the magnitude) just below 1.8 mm and the MATLAB application presents the same sigma value at about 1.9 mm. Points ‘b’ and ‘d’ are 10 meters downstream just after the last quadrupole triplet but before the bending magnet. At point ‘b’ the FORTRAN application presents a sigma value of just above 1.2 mm and the MATLAB application also presents a sigma value of just above 1.2 mm. At point ‘d’ the FORTRAN application presents a sigma value of just about 1 mm and the MATLAB application presents a sigma value of just about 1 mm. So the applications agree with each other.

Other

The application was not expected to have any instabilities and showed none. The application was tasked to load a list of 100 components elements and showed no problems; the application has a wide tolerance for user errors and will in most cases tell the user what is wrong. The biggest danger would be error multiplication through the transportation of the beam; this problem is avoided with the cumulative matrices. The Application opens a component setup file of 100 in 6 milliseconds and draws the same setup in 181 milliseconds.
Discussion

CTF3 test comparison

Even though there are slight differences in the data from both applications, one can conclude that there are no systematic errors in either of them. The slight differences can come from the mapping of the beam distance might not be the same in the beginning. For example the first point in the FORTRAN application may correspond to a small distance before the first quadrupole while the first point in the MATLAB application is just after the first quadrupole. Unfortunately, the access to real beam data was limited and a comparison to the real beam could not be made. This would be needed to further investigate the accuracy of the application.

Limitations

There are naturally some limitations to this application that can be expanded upon. The user can’t edit components and beam data already loaded in the GUI in the current application and the input formats, especially for the beam data, is a bit complicated. There are only four types of component defined in the application code and even though the users can define their own components, more components already in the application might be needed in the future. Examples of components to add would be sextupoles and octopoles. The application cannot handle non-linear components, since they cannot be expressed as transfer matrices. The beam representation is only an approximation of the real beam and should be taken into consideration when using the application. The real beam may exhibit asymmetric cross sections and nonlinear dynamics.

Alternative solutions

Calculating the flight path can be done via beta functions instead of $\sigma$-matrices, but that would be another level of approximation and further from the real physical system. Also the specific transportation algorithm used for calculation can come in different forms. This application uses a quite straight forward calculation process with predefined drift sections; an alternative algorithm could for example assume a drift section after each physical magnet or use a recursive method for transportation. The former example method was not used for user clarity purposes and the second would probably have problems with buildup errors. Instead of using a beam distribution one could use a small group of particles directly follow them or use a larger group of particles in a stochastic process to get a statistical result by sampling different points in the beam line for data histogram. This is needed with some components that cannot be expressed as matrices, usually because of non-linearities.

Further development

An editor to edit components in the applications component list would be helpful. In its current state the user has to edit the text file loaded and reload the file. In the same way an editor for the initial beam data would have been a great addition. With the editors a save function would be practical to save the edited data. The GUIs layout could be made better looking and more components could come with the application. The variables compared used in the cross section could be changeable by the user to look at other data sections. More components can be added and the ability to rotate components and displace them from the center of the beamline.
Due to the symmetries in the beam data, a beam can be expressed in a more compact way using only three parameters: emittance, alpha and beta. The x and y part of the sigma matrix can then be calculated using these three parameters. The ability to express the beam data in these parameters would make a great addition. A matching process where the user defines a desired beam-output and an optimal setup for the magnets is calculated to achieve that.

A test comparing results to real data would be necessary to ensure accurate results.

Conclusion
Linear optics is a robust theory for beam translation and works well with MATLAB. The calculations took place well within practical time limits and yielded reasonable results. The users were satisfied with the application and it may even be used in future courses by other students.

References


Appendix

User manual

Component text file format
The text files containing the setup of the components and other components must have a specific format. This format has four parameters for every entry. The parameters are separated with commas and every new entry is written on a new line. A line can be commented by writing "%%" in the beginning of that line.

The parameters, in the order it should be written, are: the components type, the components length in meters, primary physical quantity of the component (exactly what depends on component type), and secondary physical quantity (also dependent on component type)

Q: Quadrupole. Primary quantity is magnet field gradient in Tesla per meter.
   If the quantity is negative, then an x-defocusing quadrupole is created. If the quantity is positive, then an x-focusing matrix is created instead. Secondary quantity is the beam energy in MeV.

QF: Quadrupole focusing in x plane. Primary quantity is magnet field gradient in Tesla per meter.
    Secondary quantity is the beam energy in MeV.
QD: Quadrupole defocusing in x plane. Primary quantity is magnet field gradient in Tesla per meter. Secondary quantity is the beam energy in MeV.

D: Drift (empty) section, primary and secondary quantity is unused but entering zero is recommended.

SB: Sector bend magnet. Primary quantity is magnetic field strength in Tesla. Secondary quantity is the beam energy in MeV.

RB: Rectangular bend magnet. Primary quantity is magnetic field strength in Tesla. Secondary quantity is the beam energy in MeV.

Kick: Kicker magnet in x direction. Primary quantity is kick angle in milliradians.

Kickx: Same as Kickx.

Kicky: Kicker magnet in y direction. Primary quantity is kick angle in milliradians.

Kickxy: Kicker in both x and y direction. Primary quantity is kick angle in milliradians for x direction. Secondary quantity is kick angle in milliradians for y direction.

BPM: Beam Position Monitor, puts a beam position monitor marker between the components before and after. The BPM component has an identity matrix to not disturb the beam and is only used when drawing the qualitative beam line with boxes.

Example:

```plaintext

%% Two-Beam test stand beam line @ CTF3

Q, 0.226, 0.57, 200
D, 0.224, 0, 0
Q, 0.226, -1.14, 200
D, 0.224, 0, 0
Q, 0.226, 0.57, 200
D, 2.8, 0, 0
Q, 0.226, 0.57, 200
D, 0.28, 0, 0
Q, 0.226, -1.14, 200
D, 0.28, 0, 0
Q, 0.226, 0.57, 200
D, 3.642, 0, 0
Q, 0.226, 0.57, 200
D, 0.28, 0, 0
```

24
Beam text file format

The textfile for the beam also has a specific format that must be followed.

It is necessary to have at least one commented line in the beginning of the file. This line will be read as a note by the application and displayed in the GUI. This note could be used for describing for example BPM data.

The first line not commented is where the $\mu$-vector of the beam is written. The $\mu$-vector has six parameters separated with a comma. The parameters are, in order of entry: $x$, $x'$, $y$, $\tau$, and $\delta$.

The six following line will contain the six rows of the $\sigma$-matrix. Every element separated with a comma and every row on a new line.

The units that should be used is meters, seconds and electron volts.

Example

```% Beam data exiting Califes
0,0,0,0,0,0
3.5081e-6,2.5484e-8, 0, 0,0,0
2.5484e-8,1.0618e-9, 0, 0,0,0
 0, 0, 3.5081e-6,-2.5484e-8,0,0
 0, 0,-2.5484e-8, 2.5484e-8,0,0
 0, 0, 0, 0,1,0
 0, 0, 0, 0,0,0
```

GUI

In the gui, press file -> open magnet setup to load magnet list.
Press file -> open beam data to load beam data.
Press update to calculate the beam flight path.
Check the Data Cursor box to select data points.
With a data point selected, press cross-section to get a cross section of the beam and a $\sigma$-matrix.
The CTF3 two-beam test stand beamline

The beamline for the CTF3 two-beam test stand has the following components\(^4\) with their corresponding lengths and the test data used in the simulator. A beam energy of 200 MeV was assumed for the test.

<table>
<thead>
<tr>
<th>type</th>
<th>Length in meters</th>
<th>Tesla/m or Tesla</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quadrupole, focus</td>
<td>0.226</td>
<td>0.57</td>
</tr>
<tr>
<td>Drift (in lens triplet)</td>
<td>0.224</td>
<td></td>
</tr>
<tr>
<td>Quadrupole, defocus</td>
<td>0.226</td>
<td>-1.14</td>
</tr>
<tr>
<td>Drift (in lens triplet)</td>
<td>0.224</td>
<td></td>
</tr>
<tr>
<td>Quadrupole, focus</td>
<td>0.226</td>
<td>0.57</td>
</tr>
<tr>
<td>Drift section</td>
<td>2.8</td>
<td></td>
</tr>
<tr>
<td>Quadrupole, focus</td>
<td>0.226</td>
<td>0.57</td>
</tr>
<tr>
<td>Drift (in lens triplet)</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>Quadrupole, defocus</td>
<td>0.226</td>
<td>-1.14</td>
</tr>
<tr>
<td>Drift (in lens triplet)</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>Quadrupole, focus</td>
<td>0.226</td>
<td>0.57</td>
</tr>
<tr>
<td>Drift (through experimentation tank)</td>
<td>3.642</td>
<td></td>
</tr>
<tr>
<td>Quadrupole, focus</td>
<td>0.226</td>
<td>0.57</td>
</tr>
<tr>
<td>Drift (in lens triplet)</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>Quadrupole, defocus</td>
<td>0.226</td>
<td>-1.14</td>
</tr>
<tr>
<td>Drift (in lens triplet)</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>Quadrupole, focus</td>
<td>0.226</td>
<td>0.57</td>
</tr>
<tr>
<td>Drift (to bend)</td>
<td>0.653</td>
<td></td>
</tr>
<tr>
<td>Rectangular Bend</td>
<td>0.471</td>
<td>0.3087</td>
</tr>
<tr>
<td>Drift (to monitor)</td>
<td>0.869</td>
<td></td>
</tr>
</tbody>
</table>