Particle tracking in circular accelerators using the exact Hamiltonian in SixTrack

Mattias Fjellström
matfje-7@student.ltu.se

Dep. of Engineering Sciences and Mathematics
Luleå University of Technology

Supervisors: Riccardo De Maria (CERN) and Johan Hansson (LTU)

December 2013

Abstract

Particle motion in accelerators is in general complex. Tracking codes are developed to simulate beam dynamics in accelerators. SixTrack is a long lived particle tracking code maintained at CERN, the European Organization for Nuclear Research.

A particle accelerator consists of a large number of magnets and other electromagnetic devices that guide the particle through the accelerator. Each device defines its own equation of motion, which often cannot be solved exactly. For this purpose, a number of approximations are introduced in order to facilitate the solution and to speed up the computation.

In a high-energy accelerator, the particle has small transverse momentum components. This is exploited in the small-angle approximation. In this approximation the equations of motion are expanded to a low order in the transverse momentum components. In low-energy particle accelerators, or in tracking with large momentum deviations, this approximation is invalid.

The equations of motion of a particle passing through a field-free region in an accelerator, a so called drift space, has been implemented in the SixTrack code. The equations of motion are derived from the exact Hamiltonian, keeping the non-linear term unexpanded. This solution of the drift is called the exact drift space. Previously, the drift space has been solved using the small-angle approximation. This solution of the drift is called the expanded drift space. The new implementation is a step towards a more realistic, and more general, tracking code. The drift space contains the bulk of the small-angle approximation in a tracking code, it is therefore the most important element to address.

The new drift space implementation is applied in two simulation studies on the Large Hadron Collider (LHC). In the first, particle losses in the collimation system of the machine are studied. The collimation system is a collection of protective devices, used to protect the rest of the accelerator from particles spiraling out of the machine. The application of the exact drift space in this simulation shows a small, but insignificant, variation compared to the expanded drift. Of the total 14×10^6 tracked particles, about 12×10^6 are absorbed in the collimators for each model. The total number of particles lost in other locations of the ring are about 12×10^3 for both models. The most dangerous losses are losses in the superconducting magnets, called cold losses. For the exact drift, the number of cold losses were 4471. This is a short increase from the expanded drift, where the number of cold losses were 4446. These results do not show that the exact drift space is necessary in collimation studies for the LHC. It should still be an improvement to consider for future machine protection studies.

The second simulation study on the LHC is an investigation of the tune variation as a function of the momentum deviation of the particle. The tune is a measure of the number of oscillations a particle makes during one complete turn around the accelerator. The number of oscillations must avoid certain values to not induce a resonance in the motion, causing the motion to be unstable. The momentum deviation, δ, is a measure of the momentum of a particle compared to an ideal reference particle. The horizontal– and vertical tunes were calculated for a range of values for δ, both with the exact– and expanded drift space. As expected the deviation between the models grows with a larger momentum deviation. The maximum differences in the simulation were obtained for $\delta = -4 \times 10^{-3}$, where the exact model results in a tune value larger by 3×10^{-5} for the horizontal tune and 1.5×10^{-5} for the vertical tune. These tune shifts are small, and for regular tracking simulations in the LHC they are insignificant. However, in simulations where very high-order resonance effects are considered, these tune shifts could start to become important.

Preface

This Master of Science project has been carried out as a Technical Student in the BE-ABP-LCU (Beams department, Accelerators and Beams Physics, LHC Commissioning and Upgrade) section at CERN (the European Organization for Nuclear Research) in Geneva, Switzerland, under the supervision of Riccardo de Maria.

I would like to thank all the people who have helped me with this work. Most of all I wish to thank my supervisor at CERN, Riccardo, for his guidance in this project and for introducing me to the world of accelerator physics and also for recommending me to use Linux - which will be a permanent tool in the rest of my career. I would also like to thank my supervisor at LTU, professor Johan Hansson, for proof reading this manuscript and for offering some of the most interesting physics courses at LTU, without which I might not have ended up at CERN.

Thanks also to Pascal Hermes, now working on his PhD at CERN, for the help with the collimation simulations and for providing the tools needed for analyzing the results.

This work concludes my years as a student in Engineering Physics and Electrical Engineering with specialization in Computational Methods and Physics at LTU.

Mattias Fjellstrom,
Geneva, December 2013

Contents

Chapter 1

Introduction and motivation

There are many particle accelerators in the world. Most of these are small scale accelerators used for medical diagnosis or for industrial purposes. Although fewer, the particle accelerators used in physics research are larger and more complex. The largest particle accelerator in operation today is the Large Hadron Collider [1] at CERN. Thousands of scientists around the world rely on the successful operation of this machine.

Of paramount importance in the design and operation of a particle accelerator is the need to accurately predict, and possibly correct, the behavior of the machine in various scenarios. Miscalculations can lead to a halt in construction or operation of the accelerator, adding unwanted costs and delaying potential physical discoveries. SixTrack [2] is a particle tracking code which, turn by turn, simulates the motion of particles in a circular accelerator to predict the machine behavior and performance.

This chapter aims to introduce the reader to the accelerator complex at CERN, and to the problem of particle tracking in accelerators. The process of particle tracking in accelerators is discussed together with common approximations employed in tracking codes. The chapter concludes by presenting the aim of this thesis along with a brief outline of the following chapters.

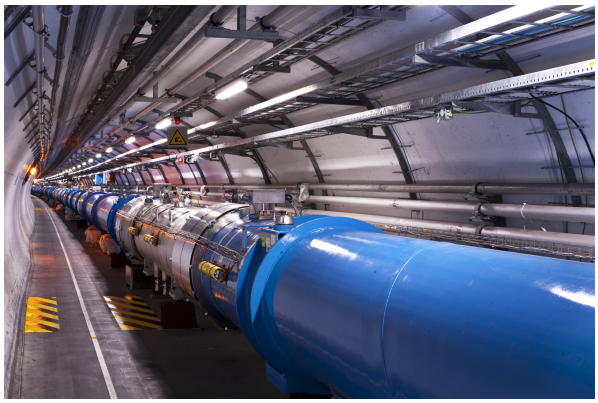
1.1 The CERN accelerator complex

CERN (the European Organization for Nuclear Research*) was founded in 1954. Since then, the struggle for new scientific discoveries has brought a cascade of increasingly expensive large scale accelerators and other experimental equipment to CERN. The crown jewel in the CERN accelerator complex is the Large Hadron Collider (LHC), see Figure 1.1. The LHC, measuring close to 27 km in circumference, is located at the border between Switzerland and France. LHC was built in the same tunnel as the previous large scale accelerator, the Large Electron Positron collider (LEP) [3]. The accelerator complex is a chain of accelerators, see Figure 1.2. Each accelerator increases the energy of the beam of particles before injecting the beam into the next accelerator in the chain.

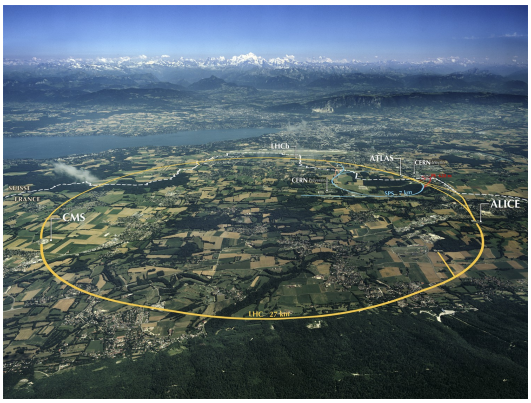
When LHC is used for proton-proton collisions, the first step in the chain is the extraction of protons from a container of hydrogen gas. The electrons are stripped away from the hydrogen, leaving the protons bare. The protons are accelerated in a linear accelerator, Linac 2, to an energy of 50 MeV. The beam is then injected into the PS Booster which further ramps up the energy to 1.4 GeV. Then the beam is injected into the PS (Proton Synchrotron) which accelerates the particles to 25 GeV. The next step is the SPS (Super Proton Synchrotron) which is the second largest accelerator at CERN. Here the protons reach an energy of 450 GeV, which is the required energy for injection into the LHC. LHC in turn accelerates the particles to a record energy of 7 TeV†. During operation there are two beams circulating in opposite directions. Each beam

*The acronym CERN originally comes from the French name *Conseil Européen pour la Recherche Nucléaire*.

†However, the LHC has not yet reached this energy.



(a) View from inside the LHC-tunnel [4].



(b) Aerial view of the LHC [5].

Figure 1.1: The Large Hadron Collider (LHC) at CERN crossing the Swiss-French border.

consists of a large number of bunches and each bunch contains on the order of 10^{11} protons. Under nominal conditions, the two beams will circulate inside the LHC for many hours with this energy. Eventually, too many particles are lost in collisions or due to other effects and the beam is dumped to give place to a new beam.

A lesser part of the LHC-operation is spent on collisions between lead ions, or between lead ions and protons. The lead ions are obtained from a pure sample of lead heated to around 500°C. The most prominent lead ion obtained from this process is Pb^{29+}, which are extracted and accelerated

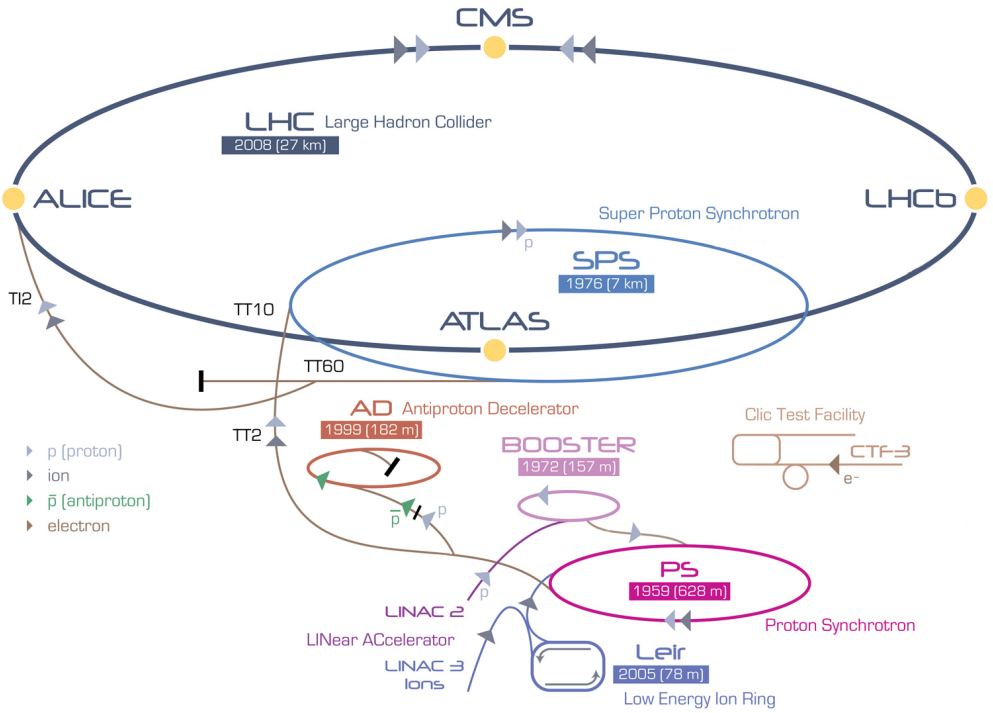


Figure 1.2: An overview of the CERN accelerator complex [6] showing the path of protons and ions in the injection stage to the LHC. The accelerators are not drawn to scale. The four large experiments (ATLAS, ALICE, CMS and LHCb) are marked by yellow circles along the LHC.

to around 5.4 MeV/u[‡]. They are then sent through a carbon foil which further strips away electrons to produce Pb^{54+}. These ions are accelerated to an energy of 72 MeV/u in LEIR (Low Energy Ion Ring). The next step is the PS which accelerates the ions to 5.9 GeV/u and then sends the ions through another foil before sending them on to the SPS. The ions are fully stripped of electrons to Pb^{82+} in the second foil. SPS accelerates the ions to 177 GeV/u and then sends them to the LHC, which brings them up to a final energy of 2.76 TeV/u for collisions.

The LHC started up its operation on September 10, 2008. After some minor set backs, it could operate almost without problems between 2010 and 2013. From the end of February 2013 it has been shut down for maintenance work in what is called the Long Shutdown 1 (LS1). The maintenance work includes replacement of several of the 15-meter long superconducting dipole magnets. LS1 stretches into 2015, whereupon LHC will continue its operation until LS2, planned in 2018. LHC will later undergo a larger upgrade to what is referred to as the High Luminosity LHC (HL-LHC) [7], increasing the design value of the luminosity by a factor of 10. This will result in a greater potential for precision measurements in the experiments. For a summary of the most important numerical parameters of the LHC, see Table 1.1.

Table 1.1: Nominal values for the LHC machine [8].

Quantity	**Value**
Circumference	26659 m
Dipole operating temperature	1.9 K
Number of magnets	9593
Number of main dipoles	1232
Number of RF-cavities	8 per beam
Nominal energy, protons	7 TeV
Nominal energy, ions	2.76 TeV/u
Peak magnetic dipole field	8.33 T
Minimum distance between bunches	7 m
Design luminosity	10^{34} cm^{-2}s^{-1}
Number of bunches per proton beam	2808
Number of protons per bunch	1.1×10^{11}
Number of turns per second	11 245
Number of collisions per second	6×10^{8}

Located around the LHC ring are four large particle collision detectors. Two of these are general purpose detectors; ATLAS (A Toroidal LHC ApparatuS) [9] and CMS (Compact Muon Solenoid) [10]. These two experiments did a joint presentation on the 4th of July 2012, announcing the discovery of a candidate to the long sought Higgs boson [11, 12]. The other two detectors are ALICE (A Large Ion Collision Experiment) [13] and LHCb (LHC beauty)[14]. ALICE investigates heavy ion collisions to study a state of matter known as quark-gluon plasma. LHCb studies particles containing b-quarks and searches for an explanation to the matter and antimatter asymmetry in the universe.

There are also other accelerators at CERN. The Antiproton Decelerator, which technically decelerates instead of accelerates particles, is responsible for providing low energy antiprotons to experiments studying antimatter. The Compact Linear Collider (CLIC) [15] is a proposed future linear collider for collisions of electrons and positrons at energies of several TeV. Equipment and technology for this collider is currently being tested at the CLIC Test Facility (CTF3) at CERN.

A new linear accelerator to be used in the initial proton acceleration stage in the LHC injection cycle is Linac 4. It will be put into operation after LS2. Linac 4 will replace the current Linac 2, which is used for the same purpose, but Linac 4 will provide a higher beam energy.

[‡]eV/u is the energy in electron-volts per nucleon.

1.2 Particle tracking

Simulating particle motion in accelerators is important to understand and predict the behavior of the machines. In the nominal LHC-operation with protons, the total stored beam energy is about 360 MJ. This is a large amount of energy and the desire to control this energy in a safe and predictable way is obvious.

Particle tracking codes are used to evaluate the long term stability of accelerators and to simulate particle losses at collimators or in the superconducting magnets, which can cause the magnet to quench. A quench is the process during which the magnet loses its superconducting state. Simulations can identify potential dangers and suggest machine settings to avoid these.

A single particle tracking code as SixTrack simulates the motion of a number of particles, each particle tracked individually through the accelerator. Collective effects in a full beam of particles are in general considered using specific simulation tools.

The process of particle tracking is simply to bring a particle with a set of initial coordinates through each magnet in the accelerator. The effect of the magnet on the motion of the particle is contained in what will be referred to as a map. Each magnet can be thought of as a map which maps the initial coordinates at the entrance of the magnet to the final coordinates at the exit of the magnet. If z_i denotes the initial coordinates, z_f the final coordinates and $\mathcal{M}$ the map of a magnet, this mapping can be illustrated as

$$z_f = \mathcal{M} \cdot z_i.$$

For a complete turn around the accelerator, passing through n magnets, the mapping can be written as

$$z_f = \mathcal{M}_n \cdot \mathcal{M}_{n-1} \cdots \mathcal{M}_2 \cdot \mathcal{M}_1 \cdot z_i.$$

The mathematical form of these maps varies depending on the approximations used in their derivations. In a linear case they can be expressed in matrix form, and all the maps for the complete turn can be combined into a single matrix. Generally, the individual maps are nonlinear and a simple combination of maps might not be possible.

Once the particle has traversed the full accelerator, the process is repeated. Tracking codes check if the particle at any point in the accelerator touches the beam pipe in which it travels. If it does, the particle is lost.

A particle tracking code does not simulate the collisions of particles in particle detectors. Particle detectors are no more than single points as far as the tracking code is concerned.

1.3 Problem formulation

The motion of a particle through an accelerator is in general complex, and several approximations are used in a tracking code. Except for the complexity in deriving the equations of motion, it can be beneficial to introduce approximations for other reasons. One concern is the time spent on simulations, which can be in the order of days for simulations involving millions of particles. The simulation time can be reduced using these approximations. Three common approximations will be explained in the following.

1. The thin-lens approximation. This approximation involves replacing the actual magnets, which will be referred to as thick magnets, with infinitely thin lenses. The thin lenses have the same integrated strength as the original magnets. In the lowest order approximation a thick magnet is replaced by a single thin lens located at the midpoint of the original magnet. A higher order approximation involves more than one thin-lens for each thick magnet. The lowest order approximation is valid when particles do not considerably change their trajectories while passing through the magnet, which is typical for high energy accelerators.

2. The hard-edge approximation. This is an approximation concerning the behavior of the magnetic field at the edge of magnets. The approximation assumes that the magnetic field

of a magnet is constant inside of the body of the magnet, but ends abruptly at its edges. In reality, this behavior is forbidden by Maxwell's equations. The field has to change continuously, with no abrupt jump at the edge of the magnet. The field at the edges of the magnets are referred to as the fringe field. For smaller accelerators, the fringe fields can have a big impact on the motion of a particle.

3. The small-angle approximation. In the derivation of the equations of motion for a particle in an accelerator it is assumed that the transverse momentum of the particle is much smaller than the longitudinal momentum, $p_{\text{tot}} \ll p_{x,y}$. This allows for the expansion of the equations of motion to first order in the transverse momenta, greatly simplifying the equations and allowing for computations involving linear matrices. In combination with the thin-lens approximation, the small-angle approximation is for the most part applied for a drift space, which is a field-free component of accelerators where particles simply drift by without changing its momentum. This approximation will be addressed in this thesis.

SixTrack employs all of the above approximations, but include options to avoid the thin-lens approximation and the hard-edge approximation by inclusion of thick element maps and special edge-focusing elements to handle fringe field effects in dipoles. To address the small-angle approximation, the equations of motion using the exact Hamiltonian formulation for a drift space has been implemented in the SixTrack code.

1.4 Existing solutions

A number of particle tracking codes exist. Some include measures to avoid the approximations discussed above.

The Methodical Accelerator Design code (MAD-X) [16] is mainly used for the design of an accelerator lattice, but includes capabilities for particle tracking. The small-angle approximation is addressed by the use of the exact Hamiltonian for the particle motion through a drift space, similar to the solution addressed in this thesis.

Another code, the Polymorphic Tracking Code (PTC) [17], has tools to avoid all the approximations mentioned above but at a considerable cost in term of execution speed. PTC is a library of tracking routines and not a pre-built simulation tool like SixTrack. This is inconvenient when performing a variety of tracking simulations using different accelerator lattices and settings. In each case, the simulation has to be built as a Fortran program on its own. In SixTrack, a simulation is initiated by a few simple input-files. There is no need to access the Fortran code unless new functionality has to be implemented.

1.5 Aim and purpose

The aim of this thesis is to introduce new physics to the particle tracking code SixTrack. The new physics is in the motion of the particle through a drift space, a field-free region of an accelerator. The purpose is to remove the small-angle approximation from the code. To evaluate the impact of the small-angle approximation, the newly implemented physics will be compared to the old through realistic case studies in the LHC. This improvement is one step towards a more realistic tracking code, improving all future simulation studies using SixTrack.

1.6 Method

SixTrack is a simulation tool built up from about 70 000 lines of Fortran code. Fortran is a high-speed computational programming language. A majority of the code is written in the Fortran 77 standard. The new implementations will use the same Fortran standard, as no features of modern Fortran will be needed.

In order to implement new functionality into an existing large simulation code it is important to get a complete overview of the code. Changes are then introduced gradually and tested. Once all new features are introduced and the code is successfully compiled, the new functionality should be benchmarked with existing codes.

The process of benchmarking is twofold. Firstly, to check that the results of a simulation do not deviate significantly from the results of a trusted code. Depending on the implementation, the results could be expected to be identical. Secondly, benchmarking can show improvements compared to existing codes. Keeping the benchmark tests as simple as possible, facilitates the identification of potential problems and causes to differences in the results.

1.7 Thesis structure

A brief overview of the following chapters in this thesis:

- Chapter 2 will introduce the reader to various theoretical topics needed to understand the results of a tracking code. These topics include Hamiltonian mechanics with the derivation of the accelerator Hamiltonian, symplectic integration and particle dynamics in accelerators. In the end of the chapter the new equations of motion are derived and explained in detail.
- Chapter 3 describes the SixTrack code in some detail, explaining the process of building SixTrack from the code. The input to, and output from, SixTrack will also be described.
- Chapter 4 provides a description of the implementation of the new physics in the code. It also presents benchmark tests of the new implementations. SixTrack is compared to MAD-X and PTC. The total simulation time is measured and compared between the codes as well as between the new implementation and the existing one.
- Chapter 5 describes two case studies on the LHC. The first is of the LHC collimation system. Particles hitting the collimators located around the accelerator ring are scattered. In the scattering process, the particles can undergo large changes in the transverse momentum. This is a case when the small-angle approximation should be avoided for an increased accuracy of the simulated trajectory. The second case study is of the tune variation as a function of the momentum deviation. The tune is the number of oscillations of the particle during one turn in the accelerator. The value of the tune should not deviate too much from a desired working point, as this can induce resonances in the motion.
- Chapter 6 discusses the new impact of the new physics in SixTrack. Other possible simulation studies which could benefit from the new physics are proposed. A few observations and suggestions for SixTrack concludes the discussion.

A number of appendices provide additional information, which has been moved out from the regular chapters to not lead the reader astray. These include the full derivation of the accelerator Hamiltonian, a short description of the implementation of an additional element, the exact thin dipole, and additional details of the SixTrack build process.

Chapter 2

Theoretical background

This chapter aims to introduce the reader to particle dynamics in an accelerator. Most concepts are introduced without a thorough derivation, but references to these are provided. The new physics implemented in the SixTrack code will also be presented in detail at the end of the chapter.

2.1 Coordinate system

To describe particle motion in an accelerator it is convenient to introduce a special curvilinear coordinate system as shown in Figure 2.1. This coordinate system differs from a global coordinate system $(\hat{X}, \hat{Y}, \hat{Z})$, as shown in the figure.

It is assumed that a design orbit exists, described by $\vec{r}_0(s)$. The design orbit can be thought of as the ideal closed orbit followed by a reference particle with constant energy in a uniform and constant magnetic bending field. The path-length, s, is the time-like variable measuring the distance along the design orbit from a chosen origin. The path of the tracked particle is described by $\vec{Q}(x, y, s, t)$, where the transverse coordinates x and y are specified relative to the design orbit.

A circular trajectory is characterized by the bending radius $\rho(s)$, or the inverse of the bending radius, $h(s) = 1/\rho(s)$. In an ideal case the bending radius is constant, in which case $h(s) \equiv h = 1/\rho$. The inverse bending radius is denoted as h_x and h_y for bending in the horizontal and vertical plane, respectively.

Three unit vectors are used to describe the trajectory of a particle relative to the design orbit. These are the unit tangent vector, $\vec{e}_s$, the unit normal vector, $\vec{e}_N$, and the unit binormal vector,

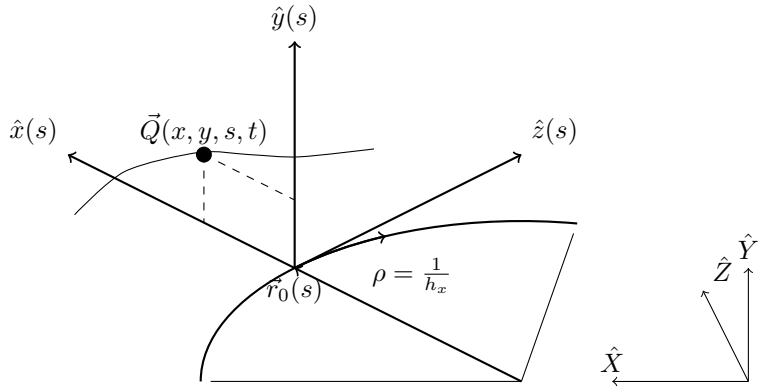


Figure 2.1: Moving reference frame $(\hat{x}, \hat{y}, \hat{z})$ parametrized by $s(t)$. The trajectory of a particle Q can be described by the coordinates (x, y, s, t).

$\vec{e}_B$. However, it is advantageous to introduce new unit vectors defined as

$$\begin{aligned} \vec{e}_x(s) &= \begin{cases} +\vec{e}_N(s), & \text{orbit in horizontal plane.} \\ -\vec{e}_B(s), & \text{orbit in vertical plane.} \end{cases} \\ \vec{e}_y(s) &= \begin{cases} +\vec{e}_B(s), & \text{orbit in horizontal plane.} \\ +\vec{e}_N(s), & \text{orbit in vertical plane.} \end{cases} \end{aligned} \tag{2.1}$$

This definition results in $\vec{e}_x(s) \times \vec{e}_y(s) = \vec{e}_s(s)$. This means $\{\vec{e}_x(s), \vec{e}_y(s), \vec{e}_s(s)\}$ represents a right handed orthonormal system with $\vec{e}_x(s)$ always in the horizontal plane and $\vec{e}_y(s)$ always in the vertical plane. The position of the tracked particle at a time t can now be written as

$$\vec{Q}(x, y, s, t) = \vec{r}_0(s(t)) + x \cdot \vec{e}_x(s(t)) + y \cdot \vec{e}_y(s(t)). \tag{2.2}$$

2.2 The accelerator Hamiltonian

This section introduce the basic framework which will be used to derive the accelerator Hamiltonian. The motion of a particle in an accelerator is derived from this. This Hamiltonian is thus the heart of a particle tracking code. For a deeper discussion of Hamiltonian mechanics than this section can provide, refer to Goldstein [18].

2.2.1 Hamiltonian mechanics

The Newtonian formulation of classical mechanics can briefly be summarized in the equation

$$\sum_i F_i = \frac{d(mv)}{dt}.$$

This equation states that the sum of forces acting on a dynamical system is equal to the time rate of change of the mechanical momentum of the system.

Another formulation of classical mechanics is the Lagrangian formulation. The Lagrangian L of a dynamical system is defined as

$$L \equiv T - V.$$

T is the kinetic energy and V is the potential energy of the system. This equation is true if the potential V is velocity-independent. A similar expression holds when the potential does depend on velocities. In that case it is replaced by a generalized potential U [18]. The Lagrangian can be shown to obey the set of equations

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = 0, \qquad (j = 1, \ldots, n). \tag{2.3}$$

These equations are known as the Euler-Lagrange equations of motion. The coordinates q_j are called the n generalized coordinates of the system. By calculating the Euler-Lagrange equation for each generalized coordinate q_j, a set of n second order differential equations are obtained. These can be solved together with a set of $2n$ initial conditions or boundary values.

A third formulation of classical mechanics is the Hamiltonian formulation. Starting with the Lagrangian L of a dynamical system, a set of coordinates q_i are chosen and a corresponding set of conjugate momenta p_i are calculated by

$$p_i \equiv \frac{\partial L(q_j, \dot{q}_j, t)}{\partial \dot{q}_i}, \qquad (i = 1, \ldots, n). \tag{2.4}$$

The set q_i are usually the same set of n generalized coordinates used in the Lagrangian. The resulting set of coordinate pairs (q_i, p_i) are known as canonical variables. The Hamiltonian is defined in terms of these variables, as

$$H = \sum_i \dot{q}_i p_i - L(q, \dot{q}, t). \tag{2.5}$$

If the forces acting on the system are conservative forces*, and the equations defining q_i do not explicitly depend on time, the Hamiltonian can be expressed as $H = T + V$. In this case the Hamiltonian is an expression for the total energy of the system.

The equations of motion for the dynamical system can now be derived using the Hamiltonian and a set of equations known as Hamilton's equations of motion. In Cartesian coordinates (x, y, z), with the mechanical momentum (p_x, p_y, p_z) as the conjugate momenta and the time t as independent variable, these equations are expressed as

$$\begin{aligned} \frac{dx}{dt} &= +\frac{\partial H}{\partial p_x}, & \frac{dy}{dt} &= +\frac{\partial H}{\partial p_y}, & \frac{dz}{dt} &= +\frac{\partial H}{\partial p_z}, \\ \frac{dp_x}{dt} &= -\frac{\partial H}{\partial x}, & \frac{dp_y}{dt} &= -\frac{\partial H}{\partial y}, & \frac{dp_z}{dt} &= -\frac{\partial H}{\partial z}. \end{aligned} \tag{2.6}$$

For n pairs of canonical coordinates this results in $2n$ first order differential equations. As in the Lagrangian case, these require $2n$ initial conditions to be solved.

2.2.2 Phase space

The $2n$ canonical variables are used to described the evolution of the system in phase space. The coordinates of this space are the $2n$ canonical variables. A single point in phase space completely describes the state of the system. As the system evolves in time, the point in phase space will move and trace out a curve.

2.2.3 Canonical transformation

A transformation from an old set of canonical coordinates (q_i, p_i) to a new set (Q_i, P_i) is called a canonical transformation. The canonical transformation may also involve the independent variable t. The transformation can be expressed as

$$Q_i = Q_i(q, p, t), \qquad P_i = P_i(q, p, t), \qquad i = 1, \ldots, N. \tag{2.7}$$

Here, q and p in the argument of Q_i and P_i represent the full set of old variables. If the transformation does not explicitly depend on the independent variable it is called a restricted canonical transformation. A canonical transformation also involves a transformation of the Hamiltonian H. The transformed Hamiltonian will be denoted by K.

The purpose of a canonical transformation is often to simplify the problem at hand. An example of this is to transform from Cartesian coordinates to spherical coordinates if the problem has spherical symmetry. In accelerator physics it is common to transform to a set of coordinates that obtain small values. It is then possible to expand nonlinear equations of motion in terms of these coordinates.

The new set of canonical coordinates (Q_i, P_i) must obey Hamilton's equations expressed with the transformed Hamiltonian K, as

$$\begin{aligned} \frac{dQ_i}{dt} &= +\frac{\partial K}{\partial P_i}, \\ \frac{dP_i}{dt} &= -\frac{\partial K}{\partial Q_i}. \end{aligned} \tag{2.8}$$

*A force is called conservative if the work done when moving a body from a starting point to an end point does not depend on the path taken between the points.

Table 2.1: Generating functions for canonical transformations [18]. The old set of coordinates is denoted as (q, p) and the new set as (Q, P). The old Hamiltonian is H and the new Hamiltonian is K.

Generating function	Coordinates	Hamiltonian
$F = F_1(q, Q, t)$	$p_i = +\frac{\partial F_1}{\partial q_i}$, $P_i = -\frac{\partial F_1}{\partial Q_i}$	$K = H + \frac{\partial F_1}{\partial t}$
$F = F_2(q, P, t) - Q_i P_i$	$p_i = +\frac{\partial F_2}{\partial q_i}$, $Q_i = +\frac{\partial F_2}{\partial P_i}$	$K = H + \frac{\partial F_2}{\partial t}$
$F = F_3(p, Q, t) + q_i p_i$	$q_i = -\frac{\partial F_3}{\partial p_i}$, $P_i = -\frac{\partial F_3}{\partial Q_i}$	$K = H + \frac{\partial F_3}{\partial t}$
$F = F_4(p, P, t) + q_i p_i - Q_i P_i$	$q_i = -\frac{\partial F_4}{\partial p_i}$, $Q_i = +\frac{\partial F_4}{\partial P_i}$	$K = H + \frac{\partial F_4}{\partial t}$

One way of achieving a canonical transformation is through a generating function F. A generating function is an arbitrary function of the old and new canonical coordinates and the independent variable. The generating function acts like a bridge from the old set of coordinates and the old Hamiltonian, to the new set of coordinates and the new Hamiltonian. Four possible generating functions are listed in Table 2.1.

An important point to note is that the solution to Hamilton's equations of motion (see Equation (2.6)) for a step in the independent variable, is itself a canonical transformation [19].

2.2.4 Particle motion in an accelerator

For particle tracking in accelerators the independent variable t is usually substituted for s, the path-length of the design trajectory. It exists a number of different common choices of which set of canonical variables to use instead of the Cartesian coordinates and the mechanical momentum. In this thesis the variables are chosen in agreement with SixTrack, which uses $(x, p_x, y, p_y, \sigma, p_\sigma)$. The definitions of these variables will be introduced below.

A relativistic particle of charge q moving in an electromagnetic field characterized by the electric field $\mathbf{E}$ and the magnetic field $\mathbf{B}$, experiences the Lorentz force

$$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B}). \tag{2.9}$$

The electric and magnetic fields are derived from the electromagnetic scalar potential ϕ and the electromagnetic vector potential $\mathbf{A}$ as

$$\begin{aligned} \mathbf{E} &= -\nabla\phi - \frac{\partial \mathbf{A}}{\partial t}, \\ \mathbf{B} &= \nabla \times \mathbf{A}. \end{aligned} \tag{2.10}$$

The Lagrangian for a relativistic particle of charge q in an electromagnetic field is [20]

$$L = -m_0 c^2 \sqrt{1 - \frac{|\mathbf{v}|^2}{c^2}} - q\phi + q\mathbf{v} \cdot \mathbf{A}. \tag{2.11}$$

The accelerator Hamiltonian can be derived using the relation between the Lagrangian and the Hamiltonian from Equation (2.5). The full derivation of the accelerator Hamiltonian can be found in Appendix B. The result is

$$\begin{aligned} &H \equiv H(x, p_x, y, p_y, \sigma, p_\sigma; s), \\ &H = p_\sigma - (1 + h_x x)\left(\sqrt{(1+\delta)^2 - (p_x - a_x)^2 - (p_y - a_y)^2} + a_s\right), \end{aligned} \tag{2.12}$$

where $\delta \equiv \delta(p_\sigma)$ is the momentum deviation of the particle with respect to the reference particle, and h_x is the horizontal inverse bending radius (see Figure 2.1). The elements of the magnetic vector potential has been normalized as

$$a_x = \frac{q}{P_0} A_x, \qquad a_y = \frac{q}{P_0} A_y, \qquad a_s = \frac{q}{P_0} A_s, \tag{2.13}$$

where P_0 is the momentum of the reference particle. In general, both the inverse bending radius h_x and the components of the magnetic vector potential vary with s.

The transverse position coordinates (x, y) are the transverse displacements as in Figure 2.1. The canonical momentum variables (p_x, p_y) are given by

$$\begin{aligned} p_x &= \frac{1}{P_0}\left(\frac{mv_x}{\sqrt{1-\frac{|\mathbf{v}|^2}{c^2}}} + qA_x\right), \\ p_y &= \frac{1}{P_0}\left(\frac{mv_y}{\sqrt{1-\frac{|\mathbf{v}|^2}{c^2}}} + qA_y\right). \end{aligned} \tag{2.14}$$

The longitudinal coordinates (σ, p_σ) are defined as

$$\begin{aligned} \sigma &= s - \beta_0 ct, \\ p_\sigma &= \frac{1}{\beta_0}\frac{E-E_0}{P_0 c}. \end{aligned} \tag{2.15}$$

The longitudinal position coordinate σ, is a measure of the delay in arrival time at a position s for the tracked particle relative to the reference particle. It is also a measure of the longitudinal separation of the particle from the center of the bunch. The longitudinal momentum coordinate p_σ is the energy difference ($\Delta E = E - E_0$) between the tracked particle and the reference particle scaled by $\beta_0 P_0 c$, where β_0 is the speed of the reference particle.

The Hamiltonian in Equation (2.12) neglects synchrotron radiation effects that becomes relevant when the particle trajectories are bent at energies several order of magnitude larger than the rest mass of the particle. It should also be noted that this Hamiltonian does not take into account the interactions between particles in a bunch. Additional terms for collective effects are needed for a complete treatment.

It is usually assumed that the magnetic field from a single magnet in an accelerator is only localized to the extent of the magnet. This is the hard edge approximation. The components of the vector potential are different depending on the type of electromagnetic elements, the terms A_x, A_y and A_s will differ and give rise to different Hamiltonians for each element. For many common elements the terms A_x and A_y are zero, greatly simplifying the calculations of the equations of motion.

The chosen set of canonical variables and the corresponding Hamiltonian gives the following set of Hamilton's equations

$$\begin{aligned} \frac{dx}{ds} &= +\frac{\partial H}{\partial p_x}, & \frac{dy}{ds} &= +\frac{\partial H}{\partial p_y}, & \frac{d\sigma}{ds} &= +\frac{\partial H}{\partial p_\sigma}, \\ \frac{dp_x}{ds} &= -\frac{\partial H}{\partial x}, & \frac{dp_y}{ds} &= -\frac{\partial H}{\partial y}, & \frac{dp_\sigma}{ds} &= -\frac{\partial H}{\partial \sigma}. \end{aligned} \tag{2.16}$$

These are the equations used for the derivation of transfer maps in SixTrack.

2.3 Accelerator physics

Particle accelerators vary in size from very short accelerators applied in hospitals for medical diagnosis to very large accelerators used for fundamental particle physics research. The design process of an accelerator in the size of the LHC [1] at CERN requires a thorough understanding of many areas of physics. This section provides an overview of some of the common elements of an accelerator. Important concepts of accelerator physics will also be introduced. Readers familiar with accelerators can skip this section. For a thorough introduction to accelerator physics, see Lee [21].

2.3.1 Components of an accelerator

This section provides a brief overview of the most common devices found in an accelerator. The collection of magnets constituting the accelerator will be referred to as the accelerator lattice.

Drift space

A drift space is a field free region of the accelerator located between magnets and other elements. A drift space provides no focusing, no bending and no acceleration of the beam. A particle entering a drift space simply drifts through without any change of momentum. This can be compared to a light ray passing through vacuum in between two lenses in a optical system.

In particle tracking codes using the thin-lens approximation, a drift space is the most common element in the lattice.

Dipole magnet

The purpose of a dipole magnet is to bend the beam around the accelerator lattice. The total bending angle of all the dipoles in a circular accelerator must add up to 2π so that the beam can circulate the lattice. The cross-section of a superconducting dipole magnet is shown in Figure 2.2a. The field direction inside the dipole magnet is shown in Figure 2.3a.

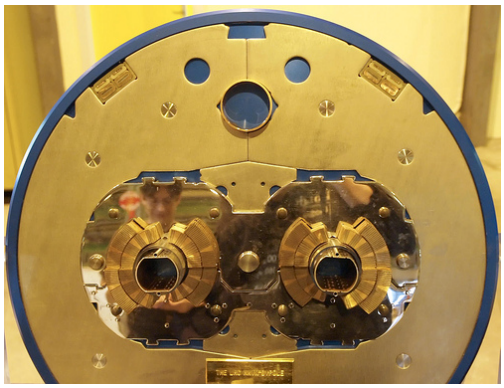
Quadrupole magnet

The purpose of a quadrupole magnet is to focus the beam. An example of a quadrupole is shown in Figure 2.2b. The field direction inside a normal quadrupole is shown in Figure 2.3b.

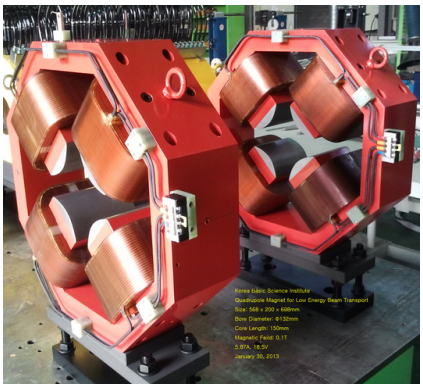
A quadrupole focuses the beam in one plane, and defocuses the beam in the other plane. A horizontally focusing quadrupole is often referred to as simply a focusing quadrupole. A vertically focusing quadrupole is often referred to as a defocusing quadrupole.

Higher order magnets

Apart from dipoles and quadrupoles there are also higher order magnets. These are used for various reasons. Sextupole magnets are mainly used for chromaticity compensation [22]. Higher order magnets can be used to correct nonlinear behavior, or to introduce nonlinearities on purpose to create a desired beam behavior.

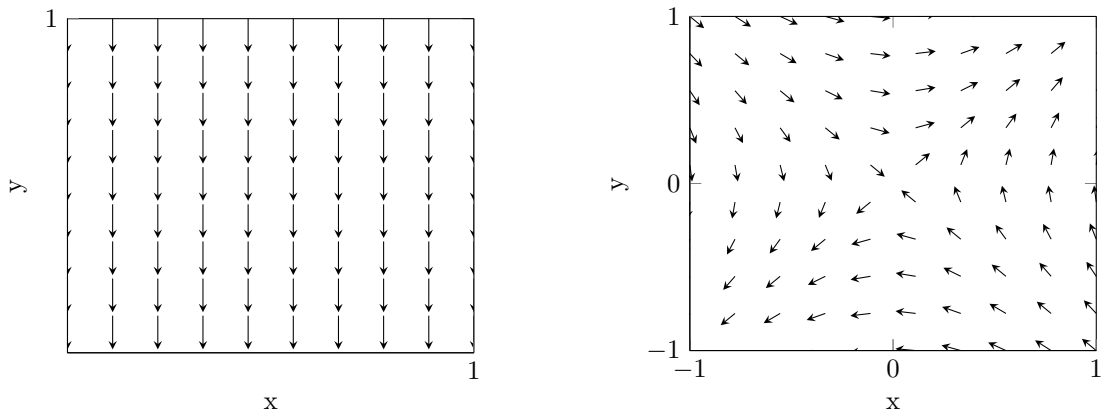


(a) Dipole magnet



(b) Quadrupole magnet

Figure 2.2: (a) Cross-section of a superconducting dipole magnet of the LHC. The two beam-pipes can clearly be seen. (b) Two normal quadrupole magnets, non-superconducting.



(a) Dipole field **(b)** Quadrupole field

Figure 2.3: A view of the direction of the magnetic field inside (a) a dipole magnet and (b) a quadrupole magnet.

RF-cavity

An RF-cavity (Radio Frequency cavity) is used to provide longitudinal acceleration and focusing of the beam. In electron storage rings an RF-cavity must provide enough energy to the beam to account for lost energy due to synchrotron radiation. Protons are less subject to synchrotron radiation than electrons since they have a mass nearly 2000 times greater.

In the LHC, the RF-systems are responsible for bringing the protons from an energy of 450 GeV to an energy of 7 TeV.

The longitudinal focusing is necessary since the particle beam consists of a number of individual bunches, each bunch containing a large number of particles. This bunch structure is important for many processes in and around the particle acceleration. If no longitudinal focusing is provided, the bunch structure is lost.

2.3.2 Transverse dynamics

A particle moving without oscillations through the center of each magnet in an accelerator is following the design orbit. In reality particles are slightly disturbed from this perfect orbit, and due to the focusing in the accelerator they oscillate around it.

An accelerator is often built in a repetitive pattern. Long sections are repeated over and over in the accelerator, with minor differences at locations of particle detectors or injection and extraction regions. This periodic nature of the accelerator allows the transverse motion of particles through drifts, dipoles and quadrupoles to be studied through the linearized Hill's equation [21]

$$\begin{aligned} x'' + K_x(s)x = 0, \quad K_x(s) = h_x^2 - K_1(s), \\ y'' + K_y(s)y = 0, \quad K_y(s) = K_1(s). \end{aligned} \tag{2.17}$$

$K_{x,y}$ are focusing functions determined by the properties of dipoles and quadrupoles. In the following only the horizontal plane will be considered. The treatment of the vertical plane is similar. The general solution of Hill's equation can be written as

$$x(s) = \sqrt{\epsilon}\sqrt{\beta(s)}\cos(\psi(s) + \phi). \tag{2.18}$$

ϵ and ϕ are constants determined by initial conditions. $\beta(s)$ is the envelope function of transverse oscillations. It is a periodic function determined by the focusing properties of the quadrupoles in the lattice. $\psi(s)$ is the phase advance function of the transverse oscillations. The relation between $\beta(s)$ and $\psi(s)$ is

$$\psi(s) = \int_0^s \frac{ds}{\beta(s)}. \tag{2.19}$$

The number of complete transverse oscillations performed during one turn around the accelerator is called the tune, Q

$$Q = \frac{1}{2\pi} \oint \frac{ds}{\beta(s)}. \tag{2.20}$$

The horizontal tune is denoted as Q_x, and the vertical tune as Q_y. Differentiating the general solution in Equation (2.18) gives

$$\begin{aligned} x(s) &= \sqrt{\epsilon}\sqrt{\beta(s)}\cos(\psi(s)+\phi), \\ x'(s) &= -\frac{\sqrt{\epsilon}}{\sqrt{\beta(s)}}\left(\alpha(s)\cos(\psi(s)+\phi)+\sin(\psi(s)+\phi)\right), \end{aligned} \tag{2.21}$$

where $\alpha(s) = -\frac{\beta'(s)}{2}$ and $\gamma(s) = \frac{1+\alpha(s)^2}{\beta(s)}$. Solving Equation (2.21) for ϵ gives

$$\epsilon = \gamma(s)x(s)^2 + 2\alpha(s)x(s)x'(s) + \beta(s)x'(s)^2. \tag{2.22}$$

The value of ϵ remains constant throughout the motion [21]. It describes the shape of an ellipse in the (x, x') phase space[†]. Throughout the motion this ellipse will change shape and orientation depending on the value of $\beta(s)$. The area of this ellipse, $\pi\epsilon$ is called the emittance.

2.3.3 Dispersion

The field inside a dipole is adjusted to the design energy E_0 and momentum P_0. Particles with a momentum different than the design momentum follow a different path through the magnet. The momentum deviation from the design momentum is denoted as δ. A particle with $\delta > 0$ will be bent less than the design particle, while a particle with $\delta < 0$ will be bent more. For the horizontal case, the position of the particle can be written as

$$x(s) = x_0(s) + x_D(s) = x_0(s) + D(s)\delta. \tag{2.23}$$

Here $x_0(s)$ is the position of a particle with the design momentum, and $D(s)$ is called the dispersion function. The path length for a particle with $\delta \neq 0$ thus deviates from the design. The ratio of the relative change in path-length ($\Delta L/L$) to the relative momentum deviation is called the momentum compaction factor [22]

$$\alpha_c = \frac{\Delta L/L}{\delta} = \frac{1}{L_0} \oint \frac{D(s)}{\rho(s)} ds, \tag{2.24}$$

where $\rho(s)$ is the bending radius.

2.3.4 Longitudinal dynamics

The longitudinal motion of particles are dictated by RF-cavities. The basic principle of synchrotron motion is explained with the help of Figure 2.4. In this discussion it is assumed that the particles have relativistic velocities ($v = c$), but the main points are still valid in the realistic case ($v < c$).

An RF-cavity is designed to provide enough energy to the particles to account for losses due to synchrotron radiation. This is achieved by timing the particle arrival at the cavity with a specific phase in the oscillating voltage. Let this phase be ψ_0 (see Figure 2.4). The corresponding voltage at this phase is V_0.

A particle with a larger momentum than the design momentum ($\delta > 0$) takes a longer path-length around the accelerator due to dispersive effects. The particle will then arrive at the RF-cavity after the ideal particle. It will encounter a lower voltage in the cavity and thus be accelerated less than the ideal particle. In this way the particle will get closer to the design momentum in the next turn around the accelerator. The opposite is true for a particle with $\delta < 0$. This particle has

[†]Generally, x' is not the conjugate momentum of x. Therefore the phase space of (x, x') is not the correct phase space to look at from a Hamiltonian point of view. However, x' and p_x are closely related.

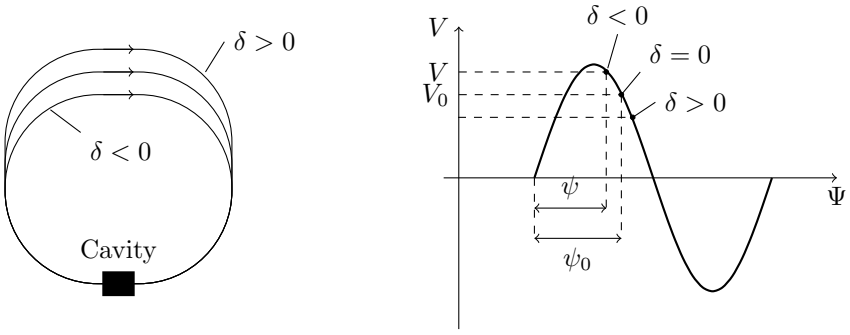


Figure 2.4: The principle of synchrotron oscillations for relativistic particles ($v = c$) [22]. The effect of a momentum deviation different from zero is illustrated.

a shorter path-length and will arrive at the cavity ahead of the ideal particle. Now the particle sees a higher voltage and is accelerated more than the ideal particle. This particle will gain in on the ideal particle during the next turn. In this way the RF-cavity focuses the beam in the longitudinal direction. The bunch structure of the particle beam can thus be kept.

2.4 Symplectic integration

Numerical integration of differential equations is a vast topic. Common integration schemes include the Euler- and Runge-Kutta schemes. In what follows, an integration scheme will be referred to as an integrator.

Symplectic integrators belong to a larger class of integrators called geometric integrators. An integration step can be thought of as a mapping or a transformation from an initial set of coordinates to a new set of coordinates. A property of all geometric integrators is that they are canonical transformations. This is why the concept of a symplectic integrator is important in the Hamiltonian approach in accelerator physics.

2.4.1 The symplectic condition

It is possible to check weather a given transformation, or integration step, is symplectic. The symplectic condition is an equation which must be satisfied for a given transformation to be symplectic. The symplectic condition can be stated as [19]

$$\mathbf{M}^T\mathbf{S}\mathbf{M} = \mathbf{S}, \tag{2.25}$$

where $\mathbf{M}$ is the Jacobian matrix of the transformation and $\mathbf{S}$ is the symplectic matrix. For a transformation from a set of coordinates $(x_1, p_1, x_2, p_2, \ldots, x_N, p_N)$ to a new set of coordinates $(X_1, P_1, X_2, P_2, \ldots, X_N, P_N)$ the Jacobian matrix is defined as

$$\mathbf{M} = \begin{bmatrix} \frac{\partial X_1}{\partial x_1} & \frac{\partial X_1}{\partial p_1} & \cdots & \frac{\partial X_1}{\partial x_N} & \frac{\partial X_1}{\partial p_N} \\ \frac{\partial X_2}{\partial x_1} & \frac{\partial X_2}{\partial p_1} & \cdots & \frac{\partial X_2}{\partial x_N} & \frac{\partial X_2}{\partial p_N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial X_N}{\partial x_1} & \frac{\partial X_N}{\partial p_1} & \cdots & \frac{\partial X_N}{\partial x_N} & \frac{\partial X_N}{\partial p_N} \end{bmatrix}.$$

The symplectic matrix $\mathbf{S}$ is a $2N \times 2N$ block diagonal matrix of the form

$$\mathbf{S} = \begin{bmatrix} 0 & 1 & & & \\ -1 & 0 & & & \\ & & \ddots & & \\ & & & 0 & 1 \\ & & & -1 & 0 \end{bmatrix}. \tag{2.26}$$

All the left-out entries are zeros. Note that the form of the symplectic matrix $\mathbf{S}$ depends on which order the coordinates are defined in. If the coordinates are grouped as $(x_1, x_2, \ldots, x_N, p_1, p_2, \ldots, p_N)$ the correct form of the symplectic matrix is

$$\mathbf{S} = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}.$$

I_n is the $n \times n$ unit matrix, where $2 \cdot n = N$. An important point is that for a number of Jacobian matrices $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_n$ the product

$$\mathbf{M}_1 \cdot \mathbf{M}_2 \cdots \mathbf{M}_n = \mathbf{M}_{\text{total}},$$

will also satisfy the symplectic condition in Equation (2.25), assuming each matrix individually satisfy this condition.

2.4.2 Example of symplectic integration

To see the effect of a symplectic integrator, consider the Hamiltonian for a simple pendulum

$$H(x, p) = \frac{p^2}{2} + \cos x. \tag{2.27}$$

The equations of motion are

$$\begin{aligned} \dot{x} &= +\frac{\partial H}{\partial p} = p, \\ \dot{p} &= -\frac{\partial H}{\partial x} = \sin x. \end{aligned} \tag{2.28}$$

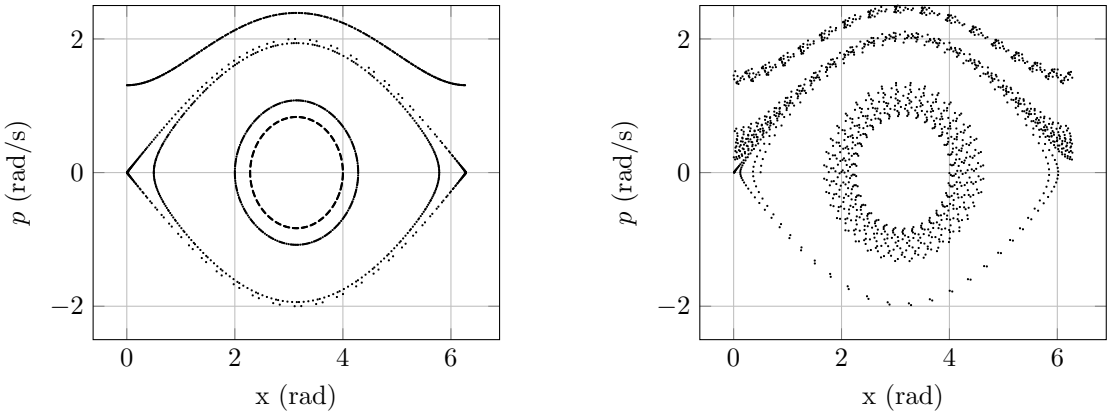
Here x is the angle coordinate and p is the canonical momentum. The mass of the pendulum is set to unity, and the length is chosen to be $\frac{1}{g}$, where g is the acceleration due to gravity. This gives $mgl = 1$ for the pendulum.

The simplest symplectic integrator for this Hamiltonian is to split the motion in three parts. Let the integration step size be h. First, the pendulum drifts with constant momentum for a step of $h/2$. At this location the pendulum receives a kick which changes the momentum. The kick is an instantaneous change in momentum, corresponding to the change of momentum for the total step h. During the kick, the position x of the pendulum stays constant. Then, the pendulum performs another drift with constant momentum for a step of $h/2$. From Equation (2.28), keeping the momentum constant, the map for the drift part for a step of $h/2$ is

$$\begin{aligned} x &\to x + \frac{h}{2} \cdot p, \\ p &\to p. \end{aligned} \tag{2.29}$$

The corresponding map for the kick part, keeping the position constant, is

$$\begin{aligned} x &\to x, \\ p &\to p + h \cdot \sin x. \end{aligned} \tag{2.30}$$



(a) Symplectic **(b)** Runge-Kutta

Figure 2.5: Phase space trajectories for a simple pendulum. The integration was performed with a step size of $h = 0.01$ and for 6000 iterations. The same set of initial conditions were used in both cases.

Iterating this map yields the motion of the pendulum. The result is shown in Figure 2.5a. For comparison the result for an asymplectic fourth-order Runge-Kutta scheme is shown in Figure 2.5b. Each integrator was iterated for 6000 iterations with a step size of $h = 0.01$. In both cases, a set of five different initial conditions were used. Each initial condition gives rise to a trajectory in phase space.

The result for the Runge-Kutta scheme shows nonphysical behavior. The phase space trajectories are making spirals, which either grows towards infinity or shrinks down to zero. In the symplectic case the trajectories in phase space repeat for each oscillation of the pendulum. This is the expected result for the simple pendulum when no damping forces are applied. The result for the Runge-Kutta scheme can be improved by decreasing the step size h, but at the cost of more computational steps.

The Jacobian matrices for the simple pendulum mappings in Equation (2.29) and (2.30) are

$$\mathbf{M}_{\text{drift}} = \begin{bmatrix} 1 & \frac{h}{2} \\ 0 & 1 \end{bmatrix}, \quad \mathbf{M}_{\text{kick}} = \begin{bmatrix} 1 & 0 \\ h\cos x & 1 \end{bmatrix}. \tag{2.31}$$

The product of the matrices for the whole transformation is

$$\mathbf{M}_{\text{total}} = \mathbf{M}_{\text{drift}} \cdot \mathbf{M}_{\text{kick}} \cdot \mathbf{M}_{\text{drift}} = \begin{bmatrix} 1 + \frac{h^2}{2}\cos x & h + \frac{h^3}{4}\cos x \\ h\cos x & 1 + \frac{h^2}{2}\cos x \end{bmatrix}. \tag{2.32}$$

Applying this result in the symplectic condition in Equation (2.25) shows that this transformation is symplectic.

2.4.3 Symplectic integration in particle tracking

The drift-kick-drift symplectic integrator described for the simple pendulum has applications in particle tracking. In the lowest order of the thin-lens approximation the motion through a magnet is approximated by a drift-kick-drift scheme. A drift space is a field-free region which does not change the momentum of the particle. The kick represents the integrated strength of the magnet, located at the midpoint of the original magnet.

A higher order symplectic integrator can be built up of more than one kick, separated by drift spaces. In SixTrack, this build up of the integrator has to be performed at the input stage of the simulation. This can be done automatically by other tools such as MAD-X.

The sort of spiraling behavior in phase space as in Figure 2.5b is a sign of asymplecticity. In such a case the total energy of the system is not conserved. When performing particle tracking

with asymplectic maps, this same spiraling behavior will appear. This behavior is not observed in a real accelerator. When implementing new maps in a tracking code, this behavior usually results from an error in the code or an error in the equations. It is therefore a helpful guide during the implementation.

2.5 Exact Hamiltonian

In order to track particles for a large number of turns using a tracking code it is of vital importance to use accurate equations. The accelerator Hamiltonian in Equation (2.12) is itself an approximation, neglecting synchrotron radiation and collective effects. Taking the accelerator Hamiltonian as the starting point there are commonly further approximations performed. From now on, the accelerator Hamiltonian in Equation (2.12) will be referred to as the exact Hamiltonian.

In the small-angle approximation, the total momentum of the tracked particle is assumed to be much larger than the transverse momentum. The accelerator Hamiltonian contains a term of the form

$$\sqrt{(1+\delta)^2 - (p_x - a_x)^2 - (p_y - a_y)^2}.$$

The total momentum is contained in the $(1+\delta)$ term. If $p_x, p_y \ll (1+\delta)$, this term is expanded as

$$(1+\delta)\sqrt{1 - \frac{(p_x - a_x)^2 + (p_y - a_y)^2}{(1+\delta)^2}} \approx (1+\delta)\left(1 - \frac{1}{2}\frac{(p_x - a_x)^2 + (p_y - a_y)^2}{(1+\delta)^2}\right). \tag{2.33}$$

The Hamiltonian using this expansion will be referred to as the expanded Hamiltonian. This greatly simplifies the derivation of the equations of motion and the approximation is valid in many circumstances. However, when there are large transverse momentum variations this approximation is unacceptable and it is necessary to keep the square root term non-expanded. As long as the use of the exact Hamiltonian does not increase simulation times drastically, it should also be considered of general interest to use it.

The reasons for expanding the square root term is twofold. The transverse momenta are usually very small terms, and solving the equations of motion with the exact Hamiltonian is only possible in very few situations. One situation where it is possible is for the motion through a drift space.

In this section, the tracking map for a drift space using the full square root term is presented. This map has been implemented in SixTrack. Previously, SixTrack has only been using the expanded Hamiltonian for the drift space. Another case where it is possible to solve the exact Hamiltonian is for a thin dipole magnet. This is presented in Appendix C.

2.5.1 The exact drift space

A drift space is a field free region of an accelerator, located between magnets and other devices. In a field free region the components of the magnetic vector potential are all zero, $A_{x,y,s} = 0$ and $a_{x,y,s} = 0$. From Equation (2.12) the exact Hamiltonian for a drift space region is

$$H(x, p_x, y, p_y, \sigma, p_\sigma; s) = p_\sigma - \sqrt{(1+\delta)^2 - p_x^2 - p_y^2}. \tag{2.34}$$

Hamilton's equation of motion for this Hamiltonian is

$$\begin{aligned}
x' &= +\frac{\partial H}{\partial p_x} = \frac{p_x}{\sqrt{(1+\delta)^2 - p_x^2 - p_y^2}}, & p_x' &= -\frac{\partial H}{\partial x} = 0, \\
y' &= +\frac{\partial H}{\partial p_y} = \frac{p_y}{\sqrt{(1+\delta)^2 - p_x^2 - p_y^2}}, & p_y' &= -\frac{\partial H}{\partial y} = 0, \\
\sigma' &= +\frac{\partial H}{\partial p_\sigma} = \left(1 - \frac{\beta_0}{\beta_z}\right), & p_\sigma' &= -\frac{\partial H}{\partial \sigma} = 0.
\end{aligned} \tag{2.35}$$

The prime denotes derivation with respect to s. Integrating Equation (2.35) for a step of length L in the independent variable s gives the transfer map for the exact drift space

$$\begin{aligned} x &\to x + x'L, & p_x &\to p_x, \\ y &\to y + y'L, & p_y &\to p_y, \\ \sigma &\to \sigma + \left(1 - \frac{\beta_0}{\beta_z}\right) L, & p_\sigma &\to p_\sigma. \end{aligned} \tag{2.36}$$

The following definitions have been introduced

$$x' = \frac{p_x}{p_z} \qquad y' = \frac{p_y}{p_z} \qquad \beta_z = \beta \cdot \frac{p_z}{1+\delta}, \tag{2.37}$$

where $p_z = \sqrt{(1+\delta)^2 - p_x^2 - p_y^2}$. Denoting the old coordinates with a subscript of 1, and the new coordinates with a subscript of 2, the Jacobian matrix of this transformation is

$$\begin{aligned} \mathbf{M} &= \begin{bmatrix} \frac{\partial x_2}{\partial x_1} & \frac{\partial x_2}{\partial p_{x1}} & \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial p_{y1}} & \frac{\partial x_2}{\partial \sigma_1} & \frac{\partial x_2}{\partial p_{\sigma 1}} \\ \frac{\partial p_{x2}}{\partial x_1} & \frac{\partial p_{x2}}{\partial p_{x1}} & \frac{\partial p_{x2}}{\partial y_1} & \frac{\partial p_{x2}}{\partial p_{y1}} & \frac{\partial p_{x2}}{\partial \sigma_1} & \frac{\partial p_{x2}}{\partial p_{\sigma 1}} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial p_{x1}} & \frac{\partial y_2}{\partial y_1} & \frac{\partial y_2}{\partial p_{y1}} & \frac{\partial y_2}{\partial \sigma_1} & \frac{\partial y_2}{\partial p_{\sigma 1}} \\ \frac{\partial p_{y2}}{\partial x_1} & \frac{\partial p_{y2}}{\partial p_{x1}} & \frac{\partial p_{y2}}{\partial y_1} & \frac{\partial p_{y2}}{\partial p_{y1}} & \frac{\partial p_{y2}}{\partial \sigma_1} & \frac{\partial p_{y2}}{\partial p_{\sigma 1}} \\ \frac{\partial \sigma_2}{\partial x_1} & \frac{\partial \sigma_2}{\partial p_{x1}} & \frac{\partial \sigma_2}{\partial y_1} & \frac{\partial \sigma_2}{\partial p_{y1}} & \frac{\partial \sigma_2}{\partial \sigma_1} & \frac{\partial \sigma_2}{\partial p_{\sigma 1}} \\ \frac{\partial p_{\sigma 2}}{\partial x_1} & \frac{\partial p_{\sigma 2}}{\partial p_{x1}} & \frac{\partial p_{\sigma 2}}{\partial y_1} & \frac{\partial p_{\sigma 2}}{\partial p_{y1}} & \frac{\partial p_{\sigma 2}}{\partial \sigma_1} & \frac{\partial p_{\sigma 2}}{\partial p_{\sigma 1}} \end{bmatrix} = \\ &= \begin{bmatrix} 1 & \frac{L}{p_z} + \frac{Lp_x^2}{p_z^3} & 0 & \frac{Lp_xp_y}{p_z^3} & 0 & \frac{\beta_0}{\beta}\frac{Lp_x(1+\delta)}{p_z^3} \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{Lp_xp_y}{p_z^3} & 1 & \frac{L}{p_z} + \frac{Lp_y^2}{p_z^3} & 0 & \frac{\beta_0}{\beta}\frac{Lp_y(1+\delta)}{p_z^3} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{\beta_0}{\beta}\frac{Lp_x(1+\delta)}{p_z^3} & 0 & -\frac{\beta_0}{\beta}\frac{Lp_y(1+\delta)}{p_z^3} & 1 & \frac{L}{p_z}\frac{\beta_0^2}{\beta^2}\left(\frac{(1+\delta)^2}{p_z^2} - 1 + \frac{\beta'}{\beta_0}\frac{1+\delta}{p_z}\right) \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{2.38}$$

Using this Jacobian in the symplectic condition in Equation (2.25), it can be seen that the transformation is symplectic.

The transverse momenta of the particle, p_x and p_y, do not change inside the drift space. This can be understood physically since there are no magnetic fields inside the drift, thus no forces will be acting on the particle. It can also immediately be seen using Hamilton's equations and noticing that the Hamiltonian in Equation (2.34) is independent of the x- and y-coordinates. The longitudinal momentum coordinate p_σ does not change either, since this requires a change in energy of the particle.

Of course, the idea that particles pass through a drift space without any change in motion is only valid when considering single particles. Collective effects in a bunch of particles will slightly change the motion of each particle. This small effect is neglected in this derivation.

The drift space map is important in particle tracking. It is not only used for a pure drift space between magnets or other devices. As in the drift-kick-drift integration scheme (see Section 2.4) the drift map is also used in the build up of symplectic integrators for the magnets themselves. This makes the map in Equation (2.36) an important improvement for a large part of the accelerator lattice.

The introduction of the exact drift space in SixTrack has been a long foreseen implementation [23].

2.5.2 The expanded drift space

Performing the expansion of the Hamiltonian as in Equation (2.33), results in the expanded Hamiltonian for a drift space

$$H = p_\sigma - \delta + \frac{1}{2}\frac{p_x^2 + p_y^2}{1+\delta}. \tag{2.39}$$

Constant terms in the Hamiltonian have been neglected since these do not contribute to the motion. The equations of motion are derived through Hamilton's equations

$$\begin{aligned} x' &= +\frac{\partial H}{\partial p_x} = \frac{p_x}{1+\delta}, & p_x' &= -\frac{\partial H}{\partial x} = 0, \\ y' &= +\frac{\partial H}{\partial p_y} = \frac{p_y}{1+\delta}, & p_y' &= -\frac{\partial H}{\partial y} = 0, \\ \sigma' &= +\frac{\partial H}{\partial p_\sigma} = 1 - \frac{\beta_0}{\beta}\left(1 + \frac{1}{2}\frac{p_x^2 + p_y^2}{(1+\delta)^2}\right), & p_\sigma' &= -\frac{\partial H}{\partial \sigma} = 0. \end{aligned} \tag{2.40}$$

Integrating the solutions for a step of L in the independent variable s results in the transfer map for the expanded drift space

$$\begin{aligned} x &\to x + x'L, & p_x &\to p_x, \\ y &\to y + y'L, & p_y &\to p_y, \\ \sigma &\to \sigma + \left(1 - \frac{\beta_0}{\beta}\left(1 + \frac{1}{2}\frac{p_x^2 + p_y^2}{(1+\delta)^2}\right)\right) \cdot L, & p_\sigma &\to p_\sigma. \end{aligned} \tag{2.41}$$

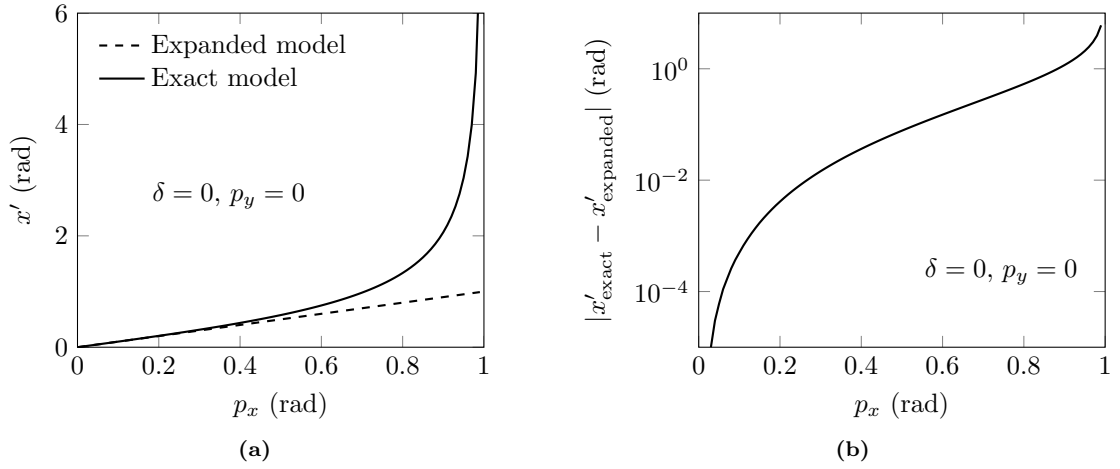
Compare the transverse motion (x, y) in Equation 2.41 with that of the exact drift space in Equation (2.36). The difference between the exact drift space and the expanded drift space is in the definition of x' and y'. For the exact drift space the definition of (x', y') is

$$\begin{aligned} x' &= \frac{p_x}{\sqrt{(1+\delta)^2 - p_x^2 - p_y^2}}, \\ y' &= \frac{p_y}{\sqrt{(1+\delta)^2 - p_x^2 - p_y^2}}, \end{aligned} \tag{2.42}$$

while for the expanded drift space the corresponding definition is

$$\begin{aligned} x' &= \frac{p_x}{1+\delta}, \\ y' &= \frac{p_y}{1+\delta}. \end{aligned} \tag{2.43}$$

The main thing to notice is the coupling between x' and y' in Equation (2.42), which is not present in Equation (2.43). The behavior of x' for the exact and the expanded case is shown in Figure 2.6. The value of p_y and δ are set to zero for this comparison.



(a) **(b)**

Figure 2.6: Comparison of x' for the exact and expanded model of the drift space. The expanded model is, as expected, only valid at low values of the transverse momentum p_x.

Chapter 3

SixTrack

SixTrack [2] is a single particle six-dimensional symplectic tracking code used for studies of particle dynamics in circular accelerators. It has been used for the LHC in dynamic aperture studies, tune optimization and collimation studies. The core program is a single executable file generated by about 70 000 lines of Fortran code. SixTrack is based on the four-dimensional tracking code RACETRACK [24], by including the longitudinal degrees of freedom and extending post-analysis capabilities. SixTrack has been under continued development during the past twenty years. Future improvements of SixTrack include GPU-implementation of computational heavy parts, implementation of new elements and tracking of particles with different mass and charge states [25].

This chapter introduces SixTrack, from the process of building the executable file from the source code, to explaining the input and output of a simulation.

3.1 Purpose of a particle tracking code

The purpose of a particle tracking code is to analyze beam properties and the long-term stability of a particle accelerator. The individual position of a single particle at a certain time is not of interest, but rather the overall properties of a beam of particles and the long term behavior of a collection of single particles.

A single particle tracking code tracks individual particles around the accelerator. The actual number of tracked particles can be large, but collective effects due to interactions between the particles are not included in such a code.

The six dimensions of a tracking code refers to the six degrees of freedom of a tracked particle. The horizontal and vertical motion counts for four degrees of freedom. The longitudinal coordinates related to the energy and position along the accelerator counts as the last two degrees of freedom. A four-dimensional code excludes the longitudinal degrees of freedom. An example of this is the code RACETRACK.

Many different tracking codes exists. Most tracking codes are designed for a specific purpose, or a specific machine. All codes contain approximations to a certain degree. Some codes might not be applicable to single pass systems such as a linear accelerator (Linac) and some codes are constructed for tracking in small accelerator rings. SixTrack was constructed and used mostly for tracking in large circular accelerators, in particular the LHC.

Another well used and long lived code is MAD-X [16], which has tracking capabilities included but is mainly used in the design process of accelerators. MAD-X includes capabilities of tracking with PTC (Polymorphic Tracking Code) [17].

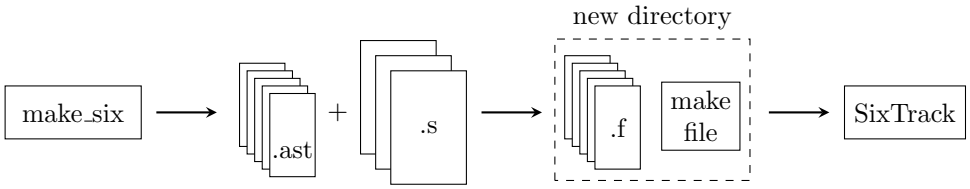


Figure 3.1: SixTrack build process. The `make_six` file generates an `.ast` file for each Fortran file to be produced. The `.ast` files produce the Fortran files from the source code in the three `.s` files, `sixtrack.s`, `lielib.s` and `dabnew.s`. The Fortran files are linked and compiled to produce the SixTrack executable file.

3.2 Code structure and building SixTrack

The process of producing the executable file for SixTrack is illustrated in Figure 3.1. The complete source code* is contained in three files, `sixtrack.s`, `lielib.s` and `dabnew.s`. These files can be used to generate a specific version of SixTrack by specifying a number of flags. A flag is a logical switch to include certain functionalities. Using the gfortran compiler on Linux, the build of SixTrack is initiated through the `make_six` file as illustrated in Listing 3.1.

Listing 3.1: Command to build SixTrack on Linux using the gfortran compiler.

```
1 ./make_six gfortran [list of flags]
```

A full list of the currently available flags can be found in Appendix E. In the first stage of the build process, a number of ASTUCE-files (file ending with `.ast`) are generated. These files contain a blueprint of what the Fortran files will contain. The `.ast` files together with the correct content from the `.s` files produce the required Fortran files. An intermediate step is the application of the Differential Algebra extension of Fortran (DAFOR) to convert special expressions in the `.s` files starting with `*FOX`, to Fortran code [26].

The Fortran source files contain a subset of the source code available in the `.s` files. In the regular version of SixTrack there are five Fortran files generated. A description of these files are listed below.

track.f This is the file containing the tracking maps. The list of element maps include dipole, quadrupole, sextupole, octupole and magnets of higher order, RF-cavity, solenoid, multipole, beam-beam lens, crab cavity and more. There are subroutines for tracking in both four– and six dimensions along with different versions for thin-lens tracking and thick-lens tracking. Some tracking maps are only available in six-dimensional thin-lens tracking.

sixve.f This file contains the main program along with a large number of subroutines for closed orbit calculation, acceleration and deceleration of the particle, input parsing, optics calculations, post processing and other tasks.

sixvefox.f SixTrack has the capability to calculate the lattice optics and closed orbit using a differential algebra approach. Subroutines for this purpose are located in this file.

lielib.f This file contains subroutines by E. Forest for normal form calculations [27].

dabnews.f This file contains the differential algebra subroutines by Berz [26]. The subroutines in this file are called from `sixvefox.f` in the closed orbit calculation.

The Fortran files are compiled and linked to produce the executable file.

After building SixTrack through `make_six` with a set of flags a new directory is created. The name of this directory depends on the set of flags. Building SixTrack using the gfortran compiler with the `collimat` flag produces the directory `SixTrack_coll_gfortran_O4`. The digit at the end

*The SixTrack source code can be obtained through SVN as `svn co http://svn.cern.ch/guest/SixTrack`

of the directory name is related to the level of optimization used by the compiler. A value of 0 (zero) corresponds to no optimization. This is useful when debugging the code. The level of optimization can be changed in the `make_six` file.

The produced directory contains the executable file, the Fortran source code and a new makefile. Changes to the code can now be made directly in the Fortran files, and the new makefile is used to compile and link these files to produce the new executable. There is no need to re-build SixTrack through `make_six`. However, it is required to apply changes directly in the `.s` files when adding new permanent features to the code. The `.s` files will overwrite any changes made directly in the Fortran files the next time `make_six` is run.

There are a few special statements available in the `.s` files, which simplifies the code structure. These statements are summarized below:

`+cd` ***name-of-block*** This can be used to define a block of code, which can then be included at as many locations as desired. Changes to the code in the block are made at a single location, and the changes are included everywhere the block is used. The name of the block is limited to 8 characters.

`+ca` ***name-of-block*** This is used to insert a previously defined block of code at a desired location.

`+if` ***(conditional statement)*** If certain parts of the code only should be included if a specific flag is used in the building of SixTrack, this can be accomplished with the `+if` statement. This statement is terminated by a `+ei` statement.

3.3 Structure of a SixTrack simulation

Below follows a simplified overview of the steps in a tracking simulation with SixTrack. Two files, `fort.2` and `fort.3` are required as input. After running SixTrack, a large number of output files called `fort.#` (where `#` is a number, `#` > 3), are created.

The first step is to generate the input data which defines the simulation. This can be done automatically from other codes such as MAD-X or done by hand following the SixTrack manual closely. When SixTrack is launched with the input data the following steps are performed:

- The simulation settings are read from `fort.3` along with the description of the accelerator elements from `fort.2`.
- All necessary parameters and variables are initialized depending on the settings in `fort.3`.
- The four– and six dimensional closed orbits are computed as well as all the optical parameters of the lattice (α, β, γ, ...). These calculations can be performed using a DA-approach.
- The tracking is performed using the settings specified in `fort.3`. At this stage the decision between tracking with thick or thin lenses is done.
- The tracking data is processed and analyzed according to the specified settings in `fort.3`. If desired, a number of plots of the analyzed data are also generated.

Output is written to the screen, and to a number of output files, during the entire simulation. The output data is used for post processing in SixTrack or with other tools.

3.4 Input to SixTrack

The main input data needed to perform particle tracking are located in two files. The `fort.2` file contains the description of every single element contained in the lattice along with the definition of the positions of each element within the lattice. The `fort.3` file contains all the other settings defining the simulation such as the total number of particles to be tracked, the initial positions of the particles and the number of turns to be simulated.

There are also a few other input files which are used for specific tasks. The `fort.12` file is used when prolonging a simulation by specifying all relevant coordinates of the tracked particles. The `fort.32` can be created in the first stages of the simulation. It contains a complete description of the accelerator structure and parameters. In a subsequent run the full description can be read from this file to the program.

The input in `fort.2` and `fort.3` are divided into a number of different blocks, each block defining a set of parameters. Each block begins with a unique four-letter keyword and ends with a line starting with the keyword `NEXT`. Between these two rows, a varying number of lines of input data are located. Only the first four letters of the opening line of a block are important. After these there may be other characters to clarify the purpose of the input block. For instance, a line beginning with `TRAC` or with `TRACKING PARAMETERS` are both valid. In the latter case the purpose of the block is more clear.

Below, the most common input blocks are described in detail. A few additional blocks are described in Appendix D. For a complete reference of all the available input blocks and corresponding settings, see the SixTrack user manual [2].

3.4.1 FREE/GEOM

The very first line in `fort.3` must start with one of the keywords `FREE` or `GEOM`. This will dictate where the lattice is read from. The keyword `FREE` means that the lattice will be read from the same file (`fort.3`). The keyword `GEOM` means the lattice is read from `fort.2` instead. The latter is the most common way and if the lattice is large (e.g. the LHC) it is very convenient to separate the lattice description from the rest of the simulation settings.

A title of the simulation can be specified on the first row of `fort.3` using column nine and onward. An example of this is shown in Listing 3.2.

Listing 3.2: `GEOM/FREE` line of the `fort.3` input file.

```
1 GEOM     Title of simulation
```

This block is not ended by the keyword `NEXT`. However, at the very end of `fort.3`, a line with the keyword `ENDE` will mark the end of the input for SixTrack. The combination of `GEOM` (or `FREE`) with `ENDE` can be seen as a large input block containing all the other blocks. This is shown in Listing 3.3.

Listing 3.3: `GEOM` block terminated by `ENDE`.

```
1 GEOM     Example simulation ------------------------
2
3   (all other input blocks)
4
5 ENDE-------------------------------------------------
```

3.4.2 Initial values for tracking (INIT)

This block specifies the initial phase space coordinates for two particles to be tracked along with some related parameters. The structure of this input block is shown in Listing 3.4.

Listing 3.4: `INIT` input block for `fort.3`.

```
1 INITIAL VALUES ---------------------------------------
2 itra chi0 chid rat iver
3 (+ 15 lines of initial coordinates)
4 NEXT-------------------------------------------------
```

The first line of values specifies the number of particles to be tracked (`itra`), the starting phase of the initial coordinate in the horizontal and vertical phase space projections (`chi0`), the phase difference between the first and second particle (`chid`) and the emittance ratio of horizontal and vertical motion (`rat`).

SixTrack tracks pairs of particles, a regular particle and a twin particle. The `itra` controls the number of particles to be tracked. However, the actual number of particles tracked is controlled in the `TRAC`input block (see below). `itra` can be set to 0, 1 or 2. A value of 0 means the coordinates from the list in this input block are used. A value of 1 means the twin particle is ignored, while a value of 2 includes the twin particle. In the `TRAC` input block, the number of particle pairs can be controlled.

The following six lines specifies the initial phase space coordinates for the first particle. The coordinates are (x, x'), (y, y') and (σ, δ). The next six lines specifies the initial phase space coordinates for the second particle. The last three rows specifies the total energy of the reference particle along with the total energy of particle one and two. All in all, the last 15 lines specifies $(x_1, x'_1, y_1, y'_1, \sigma_1, \delta_1, x_2, x'_2, y_2, y'_2, \sigma_2, \delta_2, E_0, E_1, E_2)$.

3.4.3 Tracking parameters (TRAC)

This block controls a number of parameters directly related to particle tracking. It also provides an automatic way of specifying the initial coordinates of the tracked particles. The structure of this input block is shown in Listing 3.5.

Listing 3.5: `TRAC` input block for `fort.3`.

```
1 TRACKING PARAMETERS--------------------------------
2 numl numlr napx amp(1) amp0 ird imc
3 idy(1) idy(2) idfor irew iclo6
4 nde(1) nde(2) nwr(1) ... nwr(4) ntwin ibidu iexact
5 NEXT-----------------------------------------------
```

The parameter `numl` specifies the number of tracking turns. The three parameters `napx`, `amp(1)` and `amp0` specify the number of amplitude variations, the starting amplitude and the ending amplitude, respectively. These parameters can be used to automatically generate several starting conditions for tracking. Each starting condition will count as an individual particle, and it is possible to generate up to 64 particles this way (32 particle pairs). It is also possible to specify a number of variations in the momentum deviation δ with the `imc` parameter. It is necessary to make sure the total number of combinations of amplitude and momentum variations does not exceed 64. If it does, the simulation will end prematurely with an error.

The `iclo6` flag specifies if the closed orbit is to be calculated with the DA-package. A value of 0 switches the DA-closed orbit calculation off, a value of 1 switches the calculation on and a value of 2 switches the calculation on as well as adds the closed orbit to the initial coordinates.

During tracking, the particle coordinates after each turn are printed in the output. This can be changed by the `nwr(4)` parameter. The coordinates are only printed every `nwr(4)`:th turn.

A flag of interest in this block is `iexact`, which switches on or off the exact tracking model. This is a new flag added for the exact drift space implementation.

3.4.4 Synchrotron oscillations (SYNC)

This input block controls the behavior of RF-cavities in the accelerator lattice, which is responsible of longitudinal acceleration of particles. The structure of this input block is shown in Listing 3.6.

Listing 3.6: `SYNC` input block for `fort.3`.

```
1 SYNCHROTRON OSCILLATIONS----------------------------
2 harm alc u0 phag tlen pma ition dppoff
3 dpscor sigcor
4 NEXT-----------------------------------------------
```

The `harm` parameter is the harmonic number, followed by `alc` which is the momentum compaction factor (α_c), `u0` is the total voltage of all the cavities in the accelerator, `phag` is the phase angle of the cavity, `tlen` the total length of the accelerator, `pma` the mass of the particle, `ition` is the transition energy switch. The transition energy switch can have one of three values, -1, 0 or +1. A value of 0 turns off the synchrotron motion, while -1 and +1 means below or above the transition energy, respectively. `dppoff` is the relative momentum deviation offset. The second row specifies `dpscor` and `sigcor`, the scaling factors for the relative momentum deviation and the path length difference, respectively.

The parameters of this block is only used if the only cavity in the lattice is defined with the keyword `CAV`. A cavity defined like this is added to the lattice description in `fort.2` at the desired location in the lattice. The other way to define a cavity is in the `SING` input block (see below). In that case, the parameters of this input block is ignored.

3.4.5 Single elements (SING)

Within this block, all single elements (magnets, cavities, kickers, drifts, ...) in the accelerator lattice are defined. The order in which they are defined does not matter at this point. An example of the use of this block is shown in Listing 3.7 where four elements needed to construct a simple FODO-cell[†] with a cavity are defined.

Listing 3.7: `SING` input block for `fort.2`.

```
1 SINGLE ELEMENTS-------------------------------------
2 drift    0     0.000e+00   0.0          1.00e+00
3 qf       2    -1.400e-02   0.0          0.0
4 qd       2     1.400e-02   0.0          0.0
5 cav     12     3.000e-01   1.00e+00     1.80e+02
6 NEXT-----------------------------------------------
```

The first row of values describes a drift space of length 1 m, the next two rows describe a focusing and a defocusing quadrupole, respectively. Both quadrupoles are of zero length, with strengths of equal magnitude but opposite signs. The last line specifies an RF-cavity.

One important (and easily confusing) point is that the columns in this block do not necessarily specify the same property for each element. The fifth column in the drift definition specifies the length of the drift, while the fifth column in the cavity definition specifies the phase lag angle of the cavity. Thus, it is necessary to check the definitions for each element in the SixTrack user manual [2] when defining the elements.

3.4.6 Block input (BLOC)

This input block[‡] can be used to construct combinations of linear elements. The linear single elements include drift spaces, thick dipoles, thick quadrupoles and dipole edge effect elements. Using the elements from the `SING` input block above, only the drift space could be used in a block. This is shown in Listing 3.8. It is required to have at least one block.

[†]A FODO-cell is a construction of elements containing a focusing quadrupole, a defocusing quadrupole and either drifts or dipoles in between. This cell is often repeated throughout the entire accelerator.

[‡]Do not confuse the name of this input block (`BLOC`) with a general input block to SixTrack.

Listing 3.8: `BLOC` input block for `fort.2`.

```
1 BLOCK INPUT----------------------------------------
2 BLOC1   drift
3 NEXT-----------------------------------------------
```

In thin-lens tracking the only element contained in a block is the drift space.

3.4.7 Structure of elements (STRU)

With this input block the actual structure of all the elements and blocks are constructed. This is where the order of the elements is important. The elements from the `SING` block in Listing 3.7 can be used to construct two FODO-cells in a row with a cavity at the end as shown in Listing 3.9. The drift space is included as the block defined in Listing 3.8.

Listing 3.9: `STRU` input block for `fort.2`.

```
1 STRUCTURE INPUT-------------------------------------
2 qf      BLOC1   qd
3 BLOC1   qf      BLOC1
4 qd      BLOC1   cav
5 NEXT-----------------------------------------------
```

The starting point of the accelerator can be changed by including the keyword `GO` at the desired location.

3.5 Output from SixTrack

The following list describes the output to screen when running SixTrack with standard settings.

- Starting time and date of the simulation
- A list of all defined elements. This is a formatted table containing the same information as in `fort.2`.
- A summary of the important tracking parameters in a formatted list.
- Details of the regular 4D-closed orbit calculation.
- A detailed table of optical parameters for each element in the ring. Also the horizontal and vertical tune.
- Details of the differential algebra 6D-closed orbit calculation.
- A list of the initial coordinates of all the particles.
- Turn-by-turn print of all the coordinates of the particles.
- Details and summary of the post-processing. These include

The coordinates of each pair of particles can be sent to output files in the range `fort.69-fort.90`. For a full list of output files, see Appendix C in the SixTrack manual [2].

Chapter 4

Implementation and benchmarking

This chapter describes the details of the new implementation in SixTrack. Benchmark tests with other tracking codes will be performed to evaluate the behavior of the implementation.

4.1 Implementation details

4.1.1 Flag for exact tracking

A new logical flag called `iexact` has been introduced in order to allow the exact model to be turned on or off, depending on the needs for a simulation. This flag is specified in the `TRAC` input block in `fort.3` (see Section 3.4.3). The default value of 0 corresponds to having the exact model turned off. A value of 1 corresponds to having the exact model turned on.

In the code, the `iexact` flag is added to a new Fortran common-block named `exact`. This common-block must be included in all subroutines that need to know which model is in use. The concerned subroutines are listed in Table 4.1. The code for the common block is shown in Listing 4.1.

Listing 4.1: Code for including `iexact` to a subroutine.

```
1 integer iexact
2 common/exact/iexact
```

Table 4.1: Subroutines with access to the iexact-flag.

Name	Location	Purpose
maincr	sixve.f	main SixTrack-program
comnul	sixve.f	set common parameters and variables to zero
daten	sixve.f	parse input data from fort.2 and fort.3
umlauda	sixvefox.f	6D closed orbit calculation
thin4d	track.f	4D thin-lens tracking
thin6d	track.f	6D thin-lens tracking
thin6dua	track.f	6D thin-lens tracking with acceleration
collimate2	track.f	scattering routine for collimation

In the `sixtrack.s` source file, the code in Listing 4.1 can be included through a code block named `commonex`. Adding this code block to a subroutine is done by writing `+ca commonex` at the appropriate location in `sixtrack.s`.

The default value of `iexact` is set to 0 in the `comnul` subroutine. The specified value is parsed from `fort.3` in the `daten` subroutine. If the value is missing in `fort.3` the default value is kept. This is to avoid introducing changes to old simulations where the `iexact` flag is not set.

4.1.2 Tracking routines

The physics of the exact drift is implemented in the tracking subroutines `thin4d`, `thin6d` and `thin6dua`. The `thin4d` subroutine excludes the longitudinal degrees of freedom. Both the `thin6d` and the `thin6dua` subroutines include the longitudinal degrees of freedom. The difference between the two 6D tracking subroutines is that in `thin6dua`, acceleration of the reference particle is possible.

Due to how SixTrack is implemented there is no corresponding exact version in thick-lens tracking. In thick-lens tracking, the maps for all linear elements (drift, dipole, quadrupole and the edge-focusing element) are evaluated using matrices. For consecutive linear elements, these matrices are combined to form a single matrix. This is done to speed up the computations. The definition of (x', y') for the thick elements differ from the definition for the exact drift space. This is why the exact drift is incompatible with the current implementation of the thick-lens tracking routines.

In SixTrack the unit of (x, y) is mm, and the unit of (x', y') is mrad. The unit of the longitudinal coordinate σ is also mm. SixTrack does not explicitly use p_σ as the sixth coordinate in the code. Instead it uses the momentum deviation δ, from which p_σ can be derived. The name of the coordinates in the code can be found in Table 4.2. Below follows a description of the exact drift map calculation in SixTrack. This description is for the `thin6d`/`thin6dua` subroutines. The calculations in the `thin4d` subroutine follows the same steps, but excludes all calculations involving the longitudinal coordinate σ.

1. At the entrance of the drift, a conversion from mm to m (and from mrad to rad) is performed by multiplication of 10^{-3}.

$$(x, x', y, y', \sigma) \rightarrow (x, x', y, y', \sigma) \cdot 10^{-3}. \tag{4.1}$$

   The momentum deviation $\delta = \frac{P - P_0}{P_0}$ is a fraction and does not need to be converted. The code for this conversion is shown in Listing 4.2.

Table 4.2: Particle variables in the SixTrack tracking routines.

Variable	Description	Unit
`xv(1,j)`	x-coordinate for particle j	mm
`xv(2,j)`	y-coordinate for particle j	mm
`yv(1,j)`	x'-coordinate for particle j	mrad
`yv(2,j)`	y'-coordinate for particle j	mrad
`sigmv(j)`	σ-coordinate for particle j	mm
`dpsv(j)`	δ-coordinate for particle j	-
`rvv(j)`	$\frac{\beta_0}{\beta_{(j)}}$ for particle j , β_0 is the speed of the reference particle	-

Listing 4.2: Converting from mm to m (and mrad to rad).

```
xv(1,j) = xv(1,j)*c1m3
xv(2,j) = xv(2,j)*c1m3
yv(1,j) = yv(1,j)*c1m3
yv(2,j) = yv(2,j)*c1m3
sigmv(j)= sigmv(j)*c1m3
```

The constant `c1m3` is declared to be `1d-3`, which is the double precision notation for 10^{-3} in Fortran.

2. To calculate the value of $p_z = \sqrt{(1+\delta)^2 - p_x^2 - p_y^2}$, the values of p_x and p_y are needed. These values can be found from the angle coordinates (x', y') in the small angle approximation. From Equation (2.43),

$$p_x = x'(1+\delta) \qquad\qquad p_y = y'(1+\delta). \tag{4.2}$$

Substituting these expressions for p_x and p_y in the exact drift map in Equation (2.36) give for the transverse coordinates

$$x \to x + L \cdot \frac{x'(1+\delta)}{\sqrt{(1+\delta)^2 - x'^2(1+\delta)^2 - y'^2(1+\delta)^2}},$$
$$y \to y + L \cdot \frac{y'(1+\delta)}{\sqrt{(1+\delta)^2 - x'^2(1+\delta)^2 - y'^2(1+\delta)^2}}.$$

These expressions simplifies to

$$x \to x + L \cdot \frac{x'}{\sqrt{1 - x'^2 - y'^2}}$$
$$y \to y + L \cdot \frac{y'}{\sqrt{1 - x'^2 - y'^2}}.$$

From Equation (2.36) and Equation (2.37), and using the expressions for p_x and p_y from Equation (4.2) the map for the longitudinal coordinate can be written as

$$\sigma \to \sigma + \left(1 - \frac{\beta_0}{\beta} \frac{1+\delta}{\sqrt{(1+\delta)^2 - x'^2(1+\delta)^2 - y'^2(1+\delta)^2}}\right) \cdot L.$$

This simplifies to

$$\sigma \to \sigma + \left(1 - \frac{\beta_0}{\beta} \frac{1}{\sqrt{1 - x'^2 - y'^2}}\right) \cdot L.$$

The code for the calculations in this step is shown in Listing 4.3.

Listing 4.3: Code for the exact drift map.

```
pz=sqrt(one-(yv(1,j)**2+yv(2,j)**2))
xv(1,j)=xv(1,j)+stracki*(yv(1,j)/pz)
xv(2,j)=xv(2,j)+stracki*(yv(2,j)/pz)
sigmv(j)=sigmv(j)+(stracki-((rvv(j)/pz))*stracki)
```

`stracki` is the length of the drift.

3. Finally, at the exit of the drift, a conversion back to mm and mrad is performed

$$(x, x', y, y', \sigma) \rightarrow (x, x', y, y', \sigma) \cdot 10^3. \tag{4.3}$$

The code for this transformation is shown in Listing 4.4.

Listing 4.4: Converting from m to mm (and rad to mrad).

```
1 xv(1,j)=xv(1,j)*c1e3
2 xv(2,j)=xv(2,j)*c1e3
3 yv(1,j)=yv(1,j)*c1e3
4 yv(2,j)=yv(2,j)*c1e3
5 sigmv(j)=sigmv(j)*c1e3
```

The constant `c1e3` is declared to be `1d3`, i.e. 10^3 in the Fortran double precision notation.

The description above does not include the details of the `crlibm` flag. This flag is issued by default when building SixTrack, but it can be excluded. When using this flag, the `crlibm` mathematical library [28] is used. The only modification to the drift map is in the calculation of p_z. The code for the calculation of p_z with the `crlibm` flag is shown in Listing 4.5.

Listing 4.5: Calculating p_z when the `crlibm` flag is active.

```
1 pz=exp_rn(half*log_rn(one-(yv(1,j)**2+yv(2,j)**2)))
```

The `exp_rn` and `log_rn` are special versions of the exponential and logarithmic functions, ensuring correctly rounded results.

4.1.3 Differential algebra closed orbit and optics calculations

Apart from in the tracking subroutines, the exact drift map is implemented in the file `sixvefox.f`. This file contains subroutines performing closed orbit and optics calculations using a differential algebra approach. The calculation is performed with the subroutines The implementation of this map follows the steps outline for the regular tracking routines, with a minor difference. The momentum coordinates p_x and p_y are already available in this routine and they are directly used to calculate p_z without the need to calculate them from x' and y'. The name of the coordinates used in the code can be found in Table 4.3.

The code implementation is a bit different from the regular tracking subroutines. First of all, there is no loop over particles. Only a single particle is used. The calculations use the differential algebra (DA) package by Berz [26]. Each line of code starts with the marker `*FOX` and ends with semi-colon. This tells the DAFOR pre-processor that these lines of code should be handled by the DA package. Below follows a description of the drift map used in the 6D closed-orbit calculations

Table 4.3: Particle variables in the SixTrack DA-routines.

Variable	Description	Unit
`X(1)`	x-coordinate	mm
`X(2)`	y-coordinate	mm
`YP(1)`	p_x-coordinate	mrad
`YP(2)`	p_y-coordinate	mrad
`SIGMDA`	σ-coordinate	mm
`DPDA`	δ-coordinate	-
`RV`	$\frac{\beta_0}{\beta_{(j)}}$, β_0 is the speed of the reference particle	-

1. First, a conversion to m and rad is performed. The code for this step is shown in Listing 4.6.

Listing 4.6: Converting from mm to m (and mrad to rad) in DA routines.

```
*FOX  X(1)=X(1)*C1M3 ;
*FOX  X(2)=X(2)*C1M3 ;
*FOX  Y(1)=Y(1)*C1M3 ;
*FOX  Y(2)=Y(2)*C1M3 ;
*FOX  SIGMDA=SIGMDA*C1M3 ;
```

   As in the regular tracking subroutines, the constant `C1M3`* is declared to be 10^{-3}.

2. Next, the drift map is computed. The code for this computation is shown in Listing 4.7.

Listing 4.7: Exact drift map in DA routines.

```
*FOX  PZ=SQRT((ONE+DPDA)**2-YP(1)**2-YP(2)**2) ;
*FOX  X(1)=X(1)+EL(JX)*YP(1)/PZ ;
*FOX  X(2)=X(2)+EL(JX)*YP(2)/PZ ;
*FOX  SIGMDA=SIGMDA+(ONE-(RV/PZ))*EL(JX) ;
```

   `EL(JX)` is the length of the drift.

3. Finally, a conversion back to mm and mrad is performed. The code for this conversion is shown in Listing 4.8.

Listing 4.8: Converting from m to mm (and rad to mrad) in DA routines.

```
*FOX  X(1)=X(1)*C1E3 ;
*FOX  X(2)=X(2)*C1E3 ;
*FOX  Y(1)=Y(1)*C1E3 ;
*FOX  Y(2)=Y(2)*C1E3 ;
*FOX  SIGMDA=SIGMDA*C1E3 ;
```

   `C1E3` is declared to 10^3 as before.

After the pre-processing with DAFOR, these calculations are broken down into the constituent parts and each part is handled by the appropriate DA routine. There are special routines to handle addition, subtraction, multiplication and other elementary operations. As an example, the code generated by the DAFOR pre-processor for the calculation of `X(1)` in Listing 4.7 is shown in Listing 4.9.

Listing 4.9: Map for x after DAFOR preprocessing.

```
CALL DACOP(X((1)), ISCRDA(1+IDAA))
RSCRRI(2+IDAA) = EL(JX)
CALL DACOP(YP((1)), ISCRDA(3+IDAA))
CALL DACMU(ISCRDA(3+IDAA), RSCRRI(2+IDAA),
    ISCRDA(5+IDAA))
CALL DADIV(ISCRDA(5+IDAA), PZ, ISCRDA(6+IDAA))
CALL DAADD(ISCRDA(1+IDAA), ISCRDA(6+IDAA),
    ISCRDA(7+IDAA))
CALL DACOP(ISCRDA(7+IDAA), X((1)))
```

*Note that Fortran is case-insensitive. The two variables c1m3 and C1M3 are the same. This is contrary to modern programming languages such as Python or C++.

Table 4.4: Differential Algebra subroutines in dabnews.f

Subroutine	Purpose
`DAALL(A,I,T,O,N)`	Allocates space for I vectors A. T is the variable name. O is the order and N the number of variables.
`DACOP(A,C)`	Performs $C = A$
`DAFUN(F,A,C)`	Performs $C = F(A)$ (where F is an elementary function)
`DAMUL(A,B,C)`	Performs $C = A \cdot B$
`DASUB(A,B,C)`	Performs $C = A - B$
`DASUC(A,RA,C)`	Performs $C = RA - A$

An explanation for the different subroutines in this code is given in Table 4.4. In the first line the old value of `X(1)` is copied to the array `ISCRDA`. Then the variable `RSCRRI` is assigned the length of the drift `EL(JX)`. The third line makes a copy of the momentum p_x in `YP(1)` to the `ISCRDA` array. In the fourth line the value of p_x is multiplied by the length of the drift, and the result is saved to `ISCRDA`. Then the result of that computation is divided by `PZ` and the new result is once again stored in the array `ISCRDA`. The line second to last adds the old value of x (`X(1)`) with the result from the previous calculation, Lp_x/p_z. The last step is to copy the result back to the `X(1)` variable.

In this way, very simple expressions are built up from the elementary computational steps. This is necessary to be able to handle the differential algebra approach numerically. For an explanation of the differential algebra approach, see [29].

Variables to be used in DA computations has to be defined in a special way. For this implementation, `PZ` had to be defined. The code declaring the p_z variable is shown in Listing 4.10.

Listing 4.10: Declaring the `PZ` variable.

```
1 *FOX   D V DA INT PZ NORD NVAR ;
```

The sequence of letters `D V DA INT` specifies the type of the variable. `PZ` is the name of the variable. `NORD` and `NVAR` are related to the DA map to be produced, giving the order and the number of variables. The Fortran code generated from this is shown in Listing 4.11.

Listing 4.11: Code for the `PZ` declaration after DAFOR preprocessing.

```
1 INTEGER PZ
2 CALL DAALL(PZ, 1, 'PZ', NORD, NVAR)
```

The purpose of the `DAALL` subroutine is given in Table 4.4.

4.2 Benchmark codes

In the following benchmark tests the newly implemented code in SixTrack will be compared with two other codes with similar implementations. The first is MAD-X and the second is PTC, although PTC is activated through a MAD-X script. First, a short introduction to these two codes.

4.2.1 MAD-X

The Methodical Accelerator Design (MAD) [16] code is mainly used in the design phase of accelerator structures. The code includes tracking capabilities using the thin lens approximation and

more recently supports tracking through thick quadrupole magnets [30]. The most recent version of MAD is MAD-X, first introduced in 2002. MAD-X uses the exact Hamiltonian solution for the drift space in its tracking routine, similar to the new implementation in SixTrack.

MAD-X can be run either in an interactive mode or in a batch mode. In the interactive mode the input is given through a command line, similar to how MATLAB or Mathematica can be run. In the batch mode, all input is written to a file which is then executed with MAD-X.

MAD-X uses the canonical coordinate pairs (x, p_x), (y, p_y) and $(c\Delta t, p_t)$ This differs from SixTrack in the longitudinal coordinates. MAD-X uses the longitudinal coordinates $(c\Delta t, p_t)$, and SixTrack uses (σ, p_σ). The Hamiltonian for the set of canonical coordinates in MAD-X is [31]

$$\begin{aligned} H &\equiv H(x, p_x, y, p_y, c\Delta t, p_t; s) = \\ &= \frac{p_t}{\beta_0} - (1 + hx)\left(\sqrt{p_t^2 + \frac{2p_t}{\beta_0} + 1 - (p_x - a_x)^2 - (p_y - a_y)^2} + a_s\right). \end{aligned} \tag{4.4}$$

It should be remembered that h and $a_{x,y,s}$ in general depend on s. For a drift space, the Hamiltonian in these coordinates is

$$H(x, p_x, y, p_y, c\Delta t, p_t; s) = \frac{p_t}{\beta_0} - \sqrt{p_t^2 + \frac{2p_t}{\beta_0} + 1 - p_x^2 - p_y^2}. \tag{4.5}$$

4.2.2 PTC

The Polymorphic Tracking Code (PTC) [17] is a symplectic thick lens tracking code. PTC has an option to switch on the exact Hamiltonian. The drift space solution is the same as in MAD-X and hence similar to the new implementation in SixTrack. PTC use the same set of canonical coordinates as MAD-X, with the exception of using p_t as the fifth variable which necessitates the change $c\Delta t \to -c\Delta t$ as the sixth coordinate. The canonical coordinate pairs used by PTC is thus (x, p_x), (y, p_y) and $(p_t, -c\Delta t)$.

PTC includes an exact geometric solution of the sector bending magnet, as well as more complicated magnet geometries which are solved by compositions of different maps.

PTC attaches a local frame of reference to each individual element in the lattice most appropriate for the element at hand. The philosophy of the code is that, as far as the particle traversing a magnet is concerned it does not matter if the magnet is located on a table in an office or as part of the LHC, the tracking is done in the same way regardless. Geometric transformations are performed between different frames of reference for the elements, giving PTC the possibility to track through arbitrarily shaped accelerator lattices. This differs to the approach used by SixTrack and MAD-X. For the full details of the physics implemented in PTC refer to [32]. This sophistication in the code comes at the expense of being more complicated to handle. PTC does not have the command line interaction as MAD-X or the simple input files as SixTrack. PTC is being integrated as a part of MAD-X, which simplifies its use. However, the available features of PTC through MAD-X is still fairly limited.

PTC also has capabilities to track particles using a one-turn map instead of the traditional element-to-element tracking. This one-turn map is a high-order map which maps the coordinates for a whole turn around the accelerator lattice. A lot of information can be extracted from this one-turn map, such as the optical functions and the tunes.

4.2.3 Relation to SixTrack

There is a conversion routine implemented in MAD-X which converts the accelerator lattice used in MAD-X to the correct form needed for SixTrack. MAD-X is more suited to construct a working lattice than SixTrack, which is why SixTrack is always used in combination with MAD-X for the generation of the lattice. A small part of the input needed in `fort.3` is also generated in this process, but the majority of the content needs to be written by the user.

MAD-X can invoke tracking using the PTC tracking library. The availability of the PTC library is limited to a few standard elements. No use of the more advanced features of PTC is

possible with the current version of MAD-X. In particular, there are three different integration methods available. The three methods are

- drift-kick-drift
- matrix-kick-matrix
- delta-matrix-kick-matrix

The first is the regular drift-kick-drift integration method used in thin-lens tracking in SixTrack. This method does not include any exact elements [17]. The matrix-kick-matrix method is a fast integration method since the matrices do not depend on δ. They do not need to be recomputed every time they are used. This is achieved by including a δ-dependent correction term for each application of the matrix elements. The delta-matrix-kick-matrix method is similar in spirit but the matrices are δ-dependent. This method is similar to how SixTrack performs calculations in thick lens tracking. Both matrix methods include the use of the exact drift space.

The longitudinal coordinates of MAD-X and PTC are related to the longitudinal coordinates of SixTrack by the speed of the reference particle β_0 through [31]

$$\begin{aligned} p_t &= \beta_0 \cdot p_\sigma \\ c\Delta t &= \frac{1}{\beta_0} \cdot \sigma. \end{aligned} \tag{4.6}$$

If no acceleration of the reference particle is performed, β_0 stays constant. This allows the result of MAD-X and PTC to be compared with that of SixTrack, by the simple relation in Equation 4.6. However, SixTrack uses a different sixth coordinate in the code than in the derivation of maps. In the code, SixTrack uses the momentum deviation $\delta = \frac{P-P_0}{P_0}$. To be able to compare SixTrack to MAD-X and PTC a conversion between δ and p_t is needed. The relation is

$$(1+\delta)^2 = p_t^2 + \frac{2p_t}{\beta_0} + 1. \tag{4.7}$$

This can be seen by comparing the Hamiltonian for the exact drift in SixTrack, Equation 2.34, with the corresponding Hamiltonian in MAD-X and PTC, Equation 4.5.

4.3 Benchmark in 4D and 6D

The purpose of this benchmark test is to verify the behavior for regular tracking at realistic amplitudes. The exact model of the drift in SixTrack is compared to the exact drift in MAD-X and PTC.

4.3.1 Lattice and settings

Two simple accelerator lattices were created and used to verify the behavior of the new implementation. The first lattice was a single FODO-cell, with no cavity. This lattice is referred to as the 4D lattice. The structure of the 4D lattice is

$$\left\{ \frac{1}{2}\text{QF} \quad \text{D} \quad \text{QD} \quad \text{D} \quad \frac{1}{2}\text{QF} \right\}.$$

QF represents a thin focusing quadrupole, QD represents a thin defocusing quadrupole and D represents a drift space. The cell is symmetrically constructed, with the focusing quadrupole split in half. In the simulation this means that the focusing quadrupole is defined with half the integrated strength of the original unsplit quadrupole. In these tests the drift spaces in between the quadrupoles are of the exact kind.

The second lattice, which will be referred to as the 6D lattice, is a larger lattice consisting of 34 FODO-cells of the same kind as for the 4D-lattice. An RF-cavity is included at the end point of the lattice.

The settings for each lattice element as well as for the particles are presented in Table 4.5.

Table 4.5: Benchmark settings

	4D lattice	**6D lattice**
Number of turns	10^6	10^6
Initial coordinates	$x_1 = 0.3$ mm, $y_1 = 0.1$ mm $x_2 = 0.6$ mm, $y_2 = 0.1$ mm $\delta_{1,2} = 0$	$x_1 = 0.4$ mm $x_2 = 0.8$ mm $\delta_{1,2} = 0.01$
Quadrupole integrated strength	4.0×10^{-1}	1.4×10^{-2}
Drift length	2.0 m	1.0 m
Total cell length	4.0 m	2.0 m
Total lattice length	4.0 m	68.0 m
Cavity voltage	-	0.3 MV
Cavity harmonic number	-	1
Cavity lag angle	-	0.0 rad

4.3.2 Results

The results are presented as phase space plots. For the 4D lattice, the horizontal and vertical phase space projections are shown in Figure 4.1. In the horizontal plane, the two particles have different amplitudes, but they have the same amplitude in the vertical plane.

For the 6D lattice, the horizontal and longitudinal phase space projections are shown in Figure 4.2. The two particles follow the same path in the longitudinal phase space.

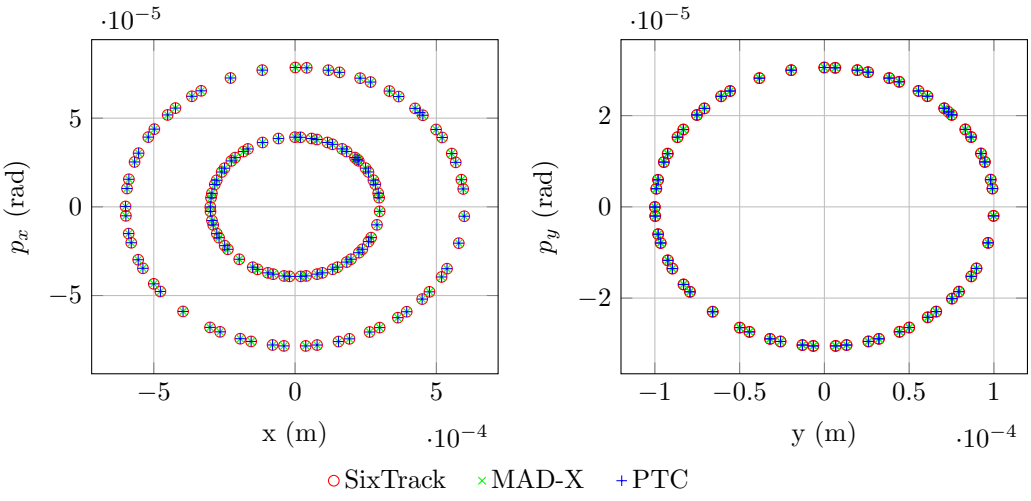


Figure 4.1: Horizontal– and vertical phase space projections of the motion.

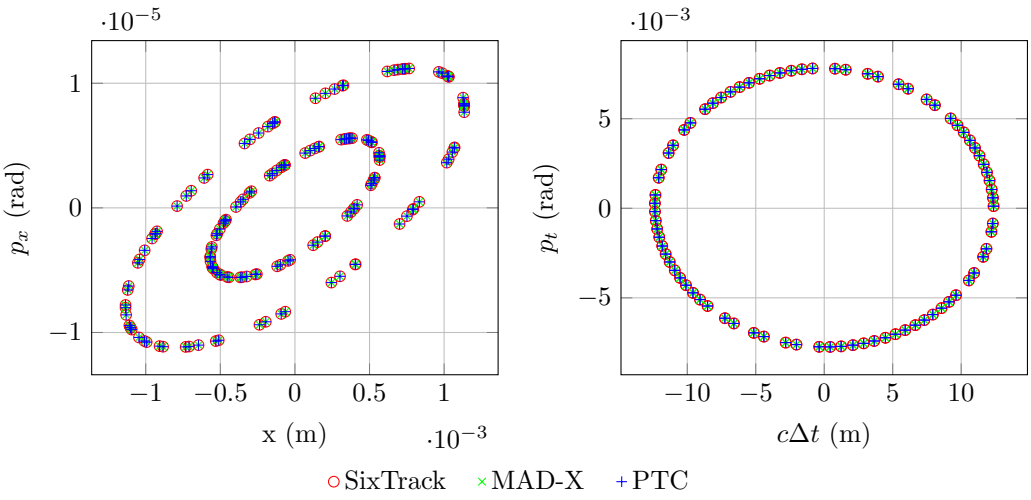


Figure 4.2: Horizontal– and longitudinal phase space projections of the motion.

4.4 Large transverse momentum

In this benchmark test, the exact model is compared to the expanded model for large transverse momentum.

4.4.1 Lattice and settings

This test was performed in 4D with the same lattice as from the test above. A single particle was tracked, with an initial horizontal angle coordinate of $x_1' = 300$ mrad, all other coordinates were set to zero. This is unrealistic because of the very large oscillations of the particle, but it serves to show the deviation between the exact– and expanded models. MAD-X was used as a reference code to verify the behavior of the exact model at large transverse momenta.

4.4.2 Results

The horizontal phase space projection of the motion is shown in Figure 4.3.

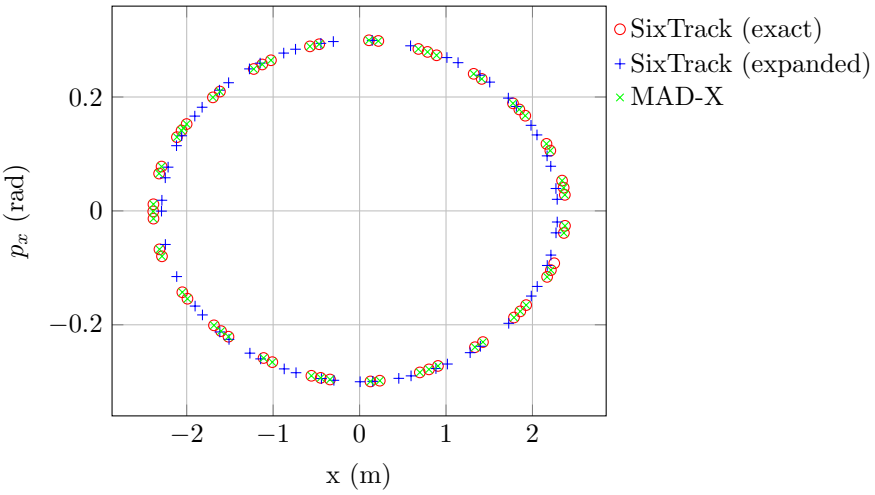


Figure 4.3: Horizontal phase space projection of the motion. The exact model of the drift is compared to the expanded model at large transverse momentum.

4.5 Computational speed

The exact drift map contains additional computational steps than the expanded drift map. This can be seen from Equation (2.36) and Equation (2.41). In this test the two different models are compared to each other in terms of speed. SixTrack is also compared to PTC, with both codes using the exact drift space.

4.5.1 Lattice and settings

For a realistic test, the full LHC lattice was used. The LHC lattice can be obtained from the CERN web [33]. The optics version used in this test was 6.503 for beam 1.

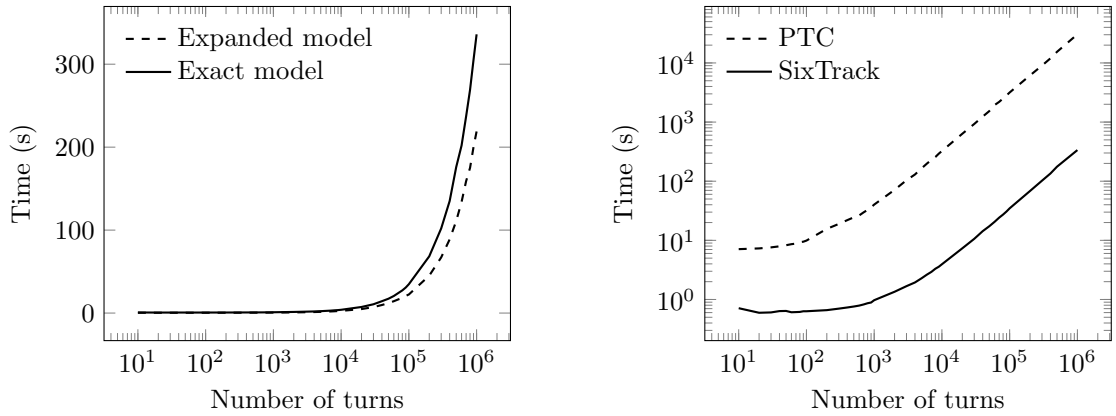
The settings for SixTrack and PTC were kept at a minimum level, so that as much of the simulation time as possible was for the actual tracking. One particle was tracked for a varying number of turns in the range $[10^1, 10^6]$. The initial coordinates for the particle were $x = 0.1$ mm, $y = 0.1$ mm and $\delta = 10^{-4}$. All other coordinates were set to zero.

For each of the two models, a total of 46 simulations were performed. The number of turns ranged from 10^1 to 10^6. For each value of turns, the simulation was repeated five times and the average simulation time was calculated.

All simulations were performed on a desktop computer running Ubuntu 13.04 with an Intel Core i7-3770 CPU at 3.40 GHz.

4.5.2 Results

The result of the comparison between the exact– and expanded drift space in SixTrack is shown in Figure 4.4a. The result of the comparison between SixTrack and PTC is shown in Figure 4.4b.



(a) Exact–, and expanded drift in SixTrack.

(b) SixTrack and PTC, using the exact drift.

Figure 4.4: Simulation time as a function of the number of turns in the LHC. In (a) a comparison of the exact and expanded model of the drift space in SixTrack. In (b) a comparison of PTC and SixTrack, both with the exact drift space.

Chapter 5

Application

In this chapter the new implementation of the exact drift space in SixTrack is applied in two case studies. In the first, the impact of the exact drift is evaluated for the LHC collimation system. In the second, the change of the transverse tunes of the LHC as a function of the momentum deviation of the particle is investigated.

5.1 Collimation

5.1.1 LHC Collimation system

The beam in an accelerator is not a static collection of particles keeping its shape turn by turn. Various beam dynamics processes and errors induce motion of the particles within the beam. Particles drift away from the center of the beam to populate what is called the beam halo. Particles located in the beam halo could move further away from the beam center, eventually hitting the inside of the beam pipe.

A collimation system is used to protect an accelerator from particles in the beam halo. When particles hit a superconducting magnet, they can cause a magnet to quench. This is when the magnet loses its superconducting state and the material in the magnet becomes resistive again. This process will cause the magnet to quickly heat up, which can lead to serious damages to the magnet.

In the LHC, the total stored beam energy can reach a value of 360 MJ*. This can be compared to the stored energy in a normal car traveling at a velocity of almost 800 km/h. This energy is two order of magnitude above the beam energy obtained in HERA or Tevatron [34]. A local loss of 4×10^7 protons in a superconducting magnet would induce an energy of about 30 mJ/cm^{-3}. This energy would be enough to cause a quench [35].

A conceptual layout of a small part of the LHC collimation system is shown in Figure 5.1. The collimation is set up as a multi-stage collimation system. The first set of collimators encountered by the beam is called the primary collimators. These collimators define the primary aperture of the collimator system. Some protons are not absorbed by these collimators, which necessitates the multi-stage system. Protons hitting the primary collimator may scatter off with large momentum-deviations or induce showers of other particles. These particles form a secondary halo which may also contribute to a possible magnet quench. Particles surviving the secondary collimators form the so called tertiary halo. This halo is handled by the tertiary collimator which is an important last protection stage located before sensitive systems of the particle detectors. There are in total eight insertion regions (IR) in the LHC. Two of these are dedicated to beam cleaning. IR3 houses the momentum cleaning and IR7 the betatron cleaning. These beam cleaning regions each contain a large number of collimators. Betatron cleaning limits the transverse extensions of the beam halo.

*Calculated for a proton beam of 7 TeV energy with 2808 particle bunches and $1.15 \cdot 10^{11}$ particles per bunch.

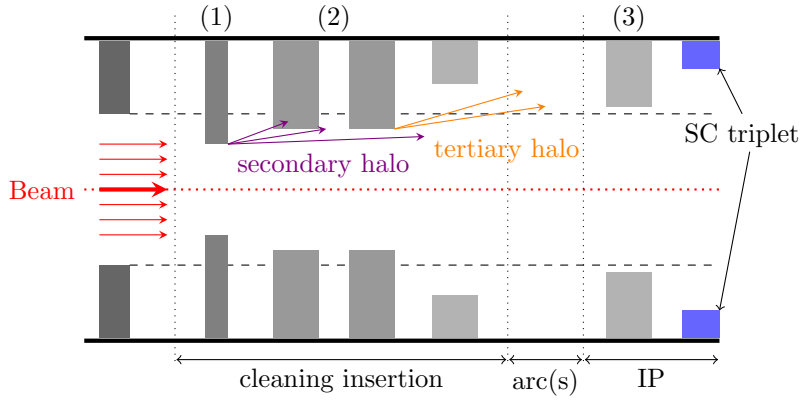


Figure 5.1: Principle of multi-stage cleaning. The three layers of collimation stages are numbered in the figure.

Momentum cleaning catch particles with a longitudinal momentum different from the reference momentum.

5.1.2 Collimation extension to SixTrack

A special version of SixTrack including subroutines for collimation is available [36]. This extension to SixTrack includes a new input block `COLL`, where the desired settings related to the collimation calculations are set. The collimators are defined in a separate database file. For each collimator the database contains its name, the nominal opening, length, material, orientation as well as the design β-function values at the location of the collimator.

The collimation addition to SixTrack allows for tracking of a large number of halo particles as well as taking into account the interaction between halo particles and arbitrarily placed collimators. When a halo particle hits the collimator jaws particle scattering occurs. The scattering effects are computed using COLLTRACK/K2 routines [36, 37].

Particle losses around the accelerator are recorded throughout the simulation. Post-processing routines create a loss map showing the locations where the losses occurs. There are three types of losses of interest, warm, cold and collimator losses. Preferably all particle losses should occur in a collimator. A cold loss is when a particle has been absorbed in a superconducting magnet. Losses of this type can cause a quench. The loss map can thus give information about locations where a quench might occur.

5.1.3 Application of drift in collimation routines

In SixTrack, a collimator of length L is located at a single point, corresponding to the mid-point of the actual collimator length, see Figure 5.2. The motion through the initial half and the final half is computed using the drift map.

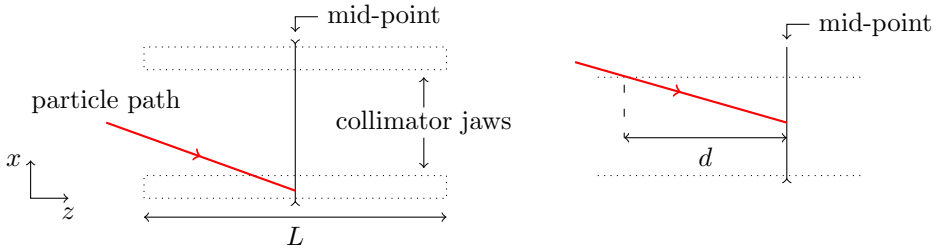


Figure 5.2: Thin lens model of a collimator in the collimation routine in SixTrack.

When the particle passes the mid-point of the collimator a check is done to see if the particle is inside the aperture limits of the collimator. If it is, the particle is transported back by half the length of the collimator. The location where the particle hits the collimator aperture is located. The scattering of this particle in the collimator material is calculated with special scattering routines.

5.1.4 Simulation of losses

A simulation of the LHC collimation system involves tracking a large number of particles with large oscillation amplitudes for a relatively few number of turns. Particles are lost around the ring, and the locations of these losses are recorded.

Simulation settings

A total number of 14.028×10^6 particles were used for each simulation case (100 sets of 6 particles $\times$ 2192 simulation runs). Each particle was tracked for 500 turns. The beam energy was set to 3.5 TeV, corresponding to the beam energy used during the 2011 LHC run. The momentum deviation was set to $\delta = 0$ in all simulations. The complete input files to SixTrack used in these simulations can be found in Appendix F. A description of the `COLL` input block for SixTrack collimation studies is also provided in the same appendix.

5.1.5 Comparison and results

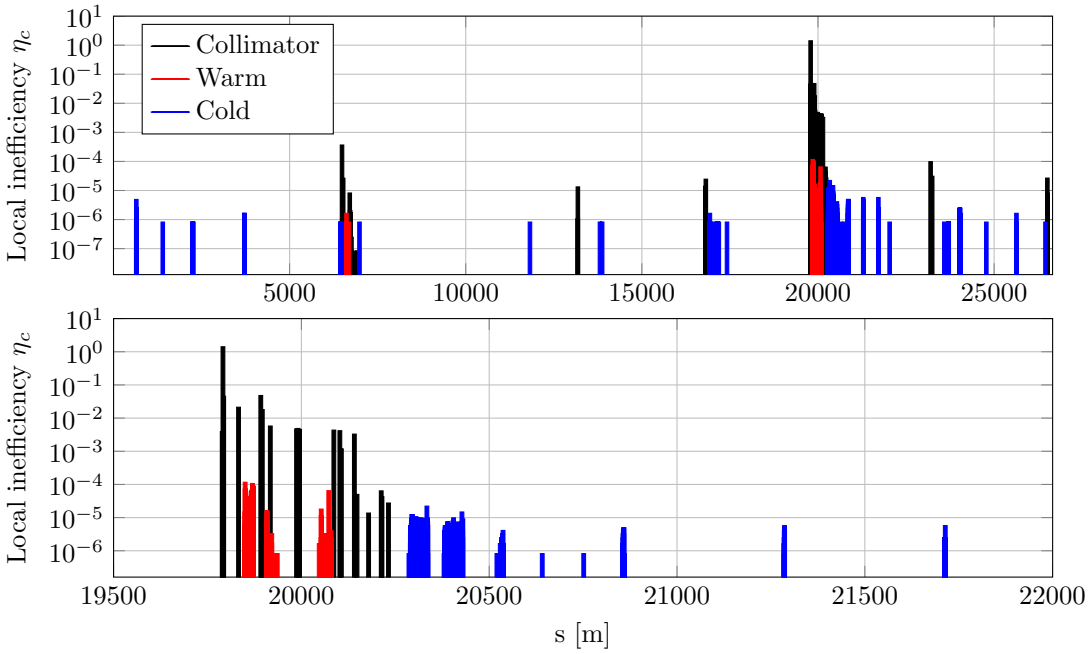
The local inefficiency is the number of particles lost in a certain bin divided by the total number of particles lost over the length of the aperture bin. This is illustrated in a loss map, where the number of lost particles is counted for a number of bins. Each bin corresponds to a certain distance along the accelerator.

The results using the exact drift space is shown in Figure 5.3a, and the results using the regular drift space is shown in Figure 5.3b. The upper plots shows the entire LHC ring, where each bin corresponds to a distance of 100 m. The lower plots in Figure 5.3a and 5.3b show a zoom on the betatron cleaning in IR7 where a majority of the losses occur, here each bin corresponds to a distance of 10 m. The three loss types shown in a loss map are collimator losses, cold losses and warm losses. A collimator loss is when a particle has been absorbed in the collimator material. This is the desired loss case, when the collimators are successfully protecting the machine. A cold loss is when a particle is lost in a superconducting magnet. Too many particles lost in the same magnet in a short time span will induce a quench. A warm loss is when a particle is lost in another element which is not kept at superconducting temperatures. These losses are not in themselves as harmful, but they can induce showers of other particles.

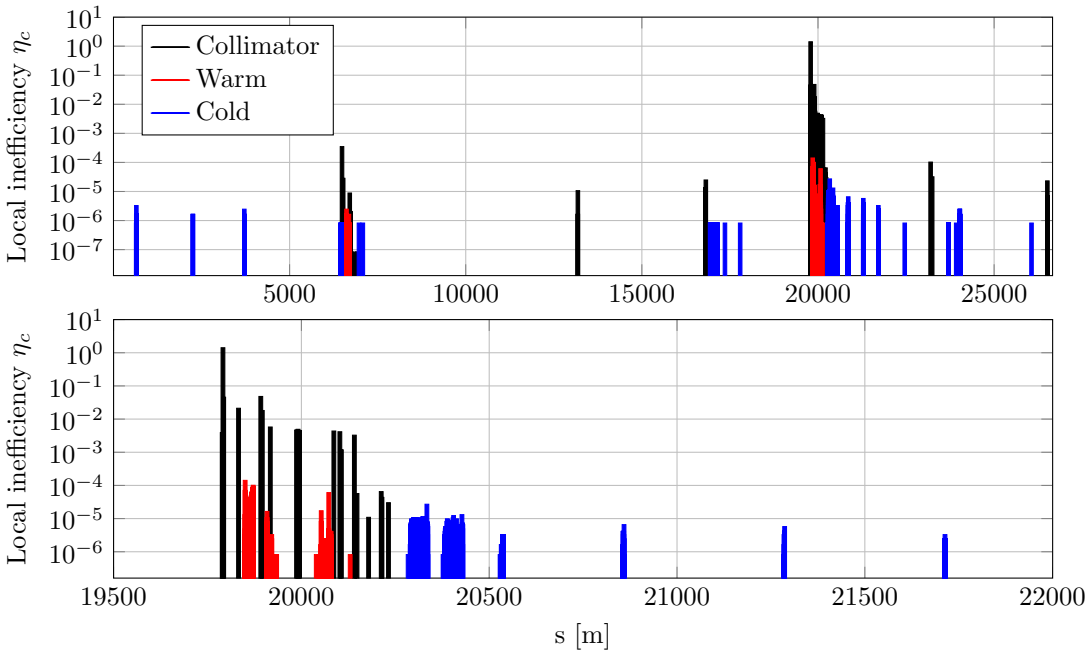
Details of the losses in the exact case is presented in Table 5.1 and details for the regular case is presented in Table 5.2.

Table 5.1: Details of losses for the collimation simulation with the exact drift space.

Exact drift space	
Total n.o. runs	2192
Total n.o. particles absorbed in collimators	1.217×10^7
Total n.o. particles lost in ring	12269
Global inefficiency	1.009×10^{-3}
Total n.o. particles lost in cold regions	4471
Total n.o. particles lost in warm regions	7798
Highest cold loss	At $s = 20334.1$ m, $\eta = 2.219 \times 10^{-5}$
Highest warm loss	At $s = 19850.3$ m, $\eta = 1.167 \times 10^{-4}$



(a) Exact drift space.



(b) Regular drift space.

Figure 5.3: Particle loss map for the LHC. The losses on the vertical scale is a measure of the local inefficiency. The upper plot shows the complete LHC circumference and the lower plot is a zoom on the betatron cleaning insertion IR7.

Table 5.2: Details of losses for the collimation simulation with the regular drift space.

Regular drift space	
Total n.o. runs	2192
Total n.o. particles absorbed in collimators	1.218×10^7
Total n.o. particles lost in ring	12392
Global inefficiency	1.017×10^{-3}
Total n.o. particles lost in cold regions	4446
Total n.o. particles lost in warm regions	7946
Highest cold loss	At $s = 20334.1$ m, $\eta = 2.708 \times 10^{-5}$
Highest warm loss	At $s = 19850.3$ m, $\eta = 1.420 \times 10^{-4}$

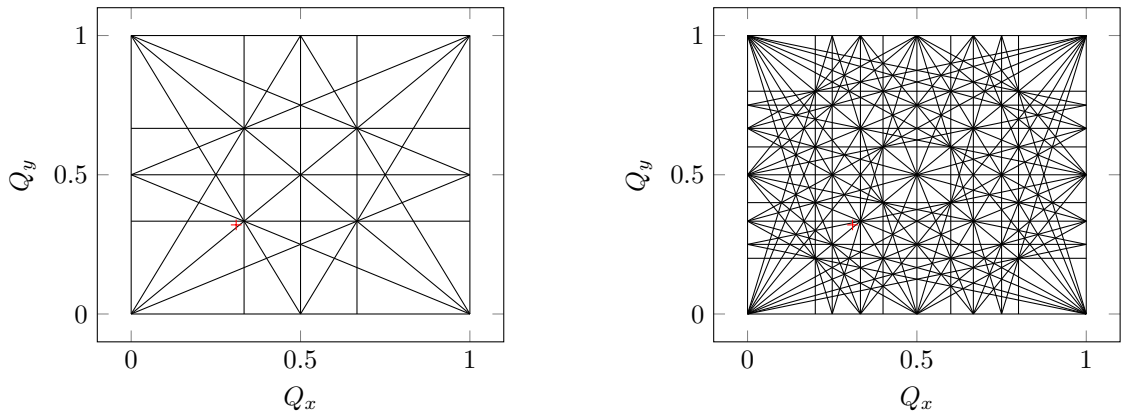
5.2 Tune shift in the LHC

5.2.1 Resonances

Multipole field errors in magnets induce resonances which should be avoided. The horizontal and vertical tunes of the machine needs to be carefully chosen. In general, the betatron oscillations in the two planes are coupled, leading to coupled resonances. The condition for a resonance is

$$m \cdot Q_x + n \cdot Q_y = p, \tag{5.1}$$

where m, n and p are integers. The order of the resonance is determined by the sum $|m| + |n|$. It is in fact the fractional part of the tune which is the important value. For a tune of 60.32, the fractional tune is 0.32. The number of complete oscillations do not affect the results. Example of 3rd and 5th order resonance diagrams are shown in Figure 5.4. The working point of the LHC (0.31,0.32) is marked out in red. A higher order diagram quickly becomes very messy and it can seem hopeless to find a working point. However, the effect of higher order resonances are very weak, and usually only up to fifth order are considered in practice [22].



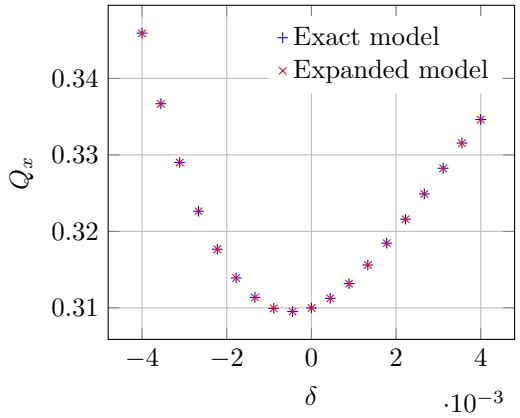
(a) 3rd order **(b)** 5th order

Figure 5.4: Resonance diagrams for 3rd and 5th order. The working point for the LHC (0.31,0.32) is marked in red. Increasing the order quickly complicates the process of finding a suitable working point.

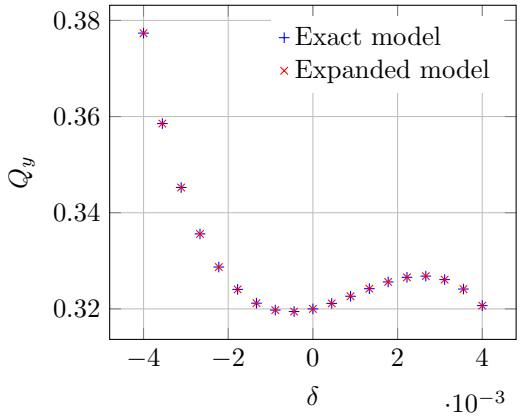
5.2.2 Simulation of tune shifts

The variation of the tunes due to a variation in the relative momentum deviation δ for the LHC is of interest. A simulation with the new implementation in SixTrack has been compared to the

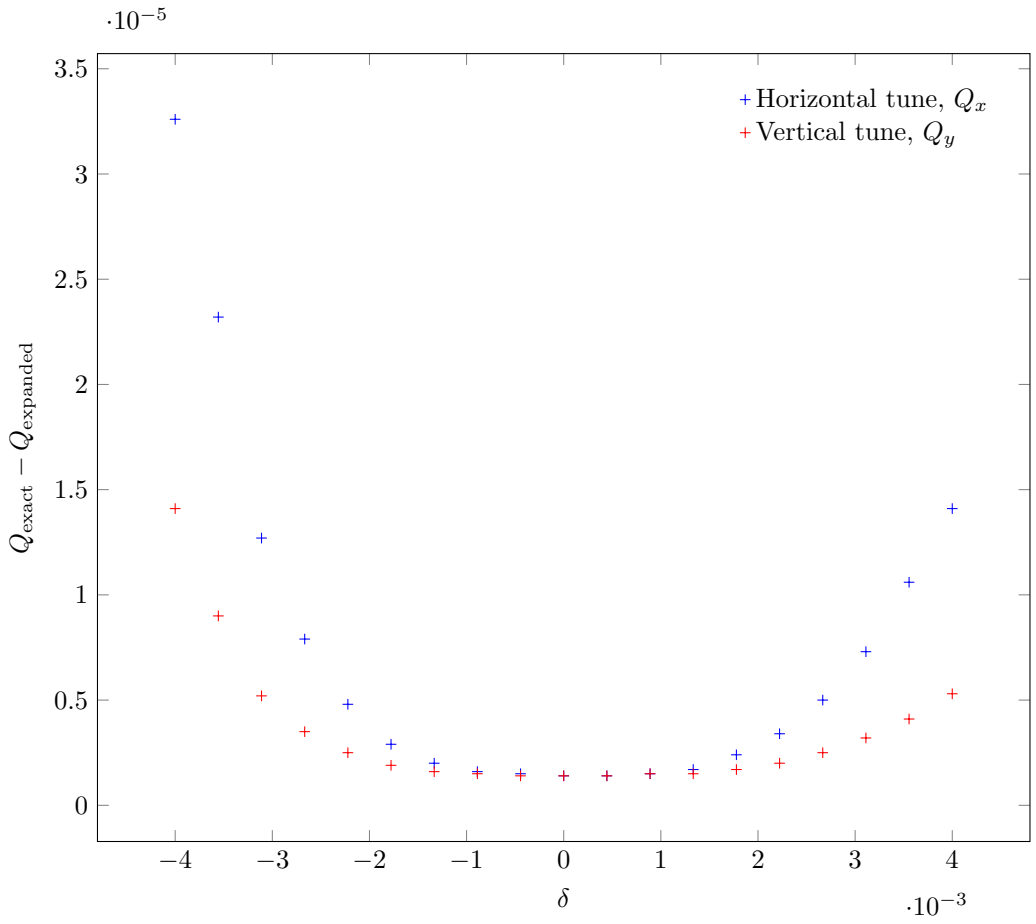
expanded drift space. For these simulations the relative momentum deviation was varied in the range $[-4, 4] \times 10^{-3}$. For each value of δ, 10^4 tracking turns were performed. The tunes were calculated through Fourier analysis of the x-coordinates for the horizontal tune, and y-coordinates for the vertical tune. The results are presented in Figure 5.5.



(a) Horizontal tune.



(b) Vertical tune.



(c) Difference between the exact– and expanded models.

Figure 5.5: Horizontal– and vertical tune variations as a function of momentum deviation δ in the LHC.

Chapter 6

Discussion and conclusions

6.1 Benchmark results

The new implementation agrees well with PTC and MAD-X. However, small initial differences in the results grows with time. This is also true between PTC and MAD-X. Initially the results agree down to the 18:th or 19:th decimal place, which is far away from what could possibly be measured in a real machine. The difference after 10^9 turns in the simple FODO-lattice grows to a few 10^{-14} m, indicating that the growth is very slow. More simulations are needed to figure out if this error grows up to measurable levels. An investigation as to why there is a difference between MAD-X and PTC could also be performed. These two codes should agree exactly since the same set of canonical variables are used and the same tracking maps.

From Figure 4.3 it is clear that the two models deviate when there is a large transverse momentum of the particle. This deviation shows up in the x-values, where the expanded model has a smaller amplitude than the exact model. This is the expected result when $\delta = 0$ and $p_y = 0$. In the expanded model, the change in x is Lp_x. In the exact model, the change in x is $Lp_x/\sqrt{1-p_x^2}$. The value of the square-root term is less than one, making the change of x larger.

The new implementation has additional computational steps to perform. It is thus expected that a simulation involving many drifts should take more time. This is seen from Figure 4.4a. Looking at the calculation of the map in Listing 4.3 one can count to eight steps for the computation of x. This can be compared to only two steps for the expanded drift space. This small number of steps for the expanded drift space comes from the fact that SixTrack keeps x' in memory instead of p_x. The increased number of steps for the exact map is due to the computation of p_z.

In the comparison between PTC and SixTrack in Figure 4.4b it is clear why SixTrack is the superior code to use in large scale simulations. SixTrack is orders of magnitude faster than PTC, even with the new "slower" exact drift implementation. The simulation speed of PTC is not appreciably improved by switching off the exact model. The simulation speed of MAD-X was tested to be approximately the same as for PTC, but this is not reported in this thesis.

6.2 Impact on collimation

The new implementation has no major impact on the resulting loss map for collimation. This is a desired result since the existing loss maps are close in agreement to measured values in the real LHC. However, it is also an indication that the exact drift map is not necessarily needed in this case. When dealing with machine protection simulations it is of utmost importance to do realistic simulations, and the exact drift implementation should still be considered.

From Table 5.1 and Table 5.2 we can see that approximately the same number of particles are absorbed in the collimators. More importantly, the number of particles absorbed in the cold regions are about the same. A total of 4471 particles in the exact case, and a total of 4446 particles in the regular case. The highest cold loss occurs at the same position in each case.

6.3 Impact on tune variation

Studying Figure 5.5 is is clear that the impact of the exact drift is small in this case. However, as expected there is a growing variation with a larger momentum deviation δ. This growth has an exponential behavior. At $\delta = -4 \times 10^{-3}$, the horizontal tune is about 3.3×10^{-5} larger with the exact drift than with the regular drift.

For the horizontal case, the tune variation over the interval $\delta = [-4, 4] \times 10^{-3}$ is in the range $Q_x \approx [0.31, 0.34]$. The corresponding vertical tune variation is $Q_y \approx [0.32, 0.38]$. In the real machine, variations of $\Delta Q \approx \pm 0.1$ are measured in nominal operation [1]. This means that the interval studied here is larger than what we could expect during normal LHC operation.

Tune variations on the order of 10^{-5} are not possible to measure. However, in simulations this is very much possible to measure and this could potentially be important for studies of high-order resonance effects.

6.4 Further simulations

The impact of the exact drift space could be further investigated in simulations with small rings. A typical small ring (e.g. LEIR or the PS Booster at CERN) contains four short straight sections and four 90° arcs. The beam energy in such a ring is much smaller than in a large scale accelerator. This means the small-angle approximation could be invalid (depending on the exact energy of the beam). Approximations such as the hard-edge approximation must also be avoided.

A continued investigation of the collimation system can be performed by running simulations with different values of the momentum deviation δ. This can be used to see at what value of δ the difference between the exact and regular model become significant. It would also be beneficial to simulate the motion of the scattered particles from collimator losses. These particles are ejected with a wide range of momentum deviations.

Implementation of an exact thin dipole was also done (see Appendix C). This dipole has not been fully tested, and therefor not discussed at any length in this thesis. It has to be combined with the exact drift space, which again makes the drift space the more important map. There is no corresponding element in MAD-X or PTC, and it can therefore not be directly benchmarked with these codes to evaluate its behavior. Future simulations can compare this implementation with the regular thin dipole. It would also be beneficial to compare this dipole with the geometrically exact solution of the sector dipole magnet [32], to see at what bending angle these two dipoles agree.

6.5 Comment on SixTrack

To quote from the developer manual for SixTrack, *"SixTrack is wonderful, but it is bloody complicated!"*. The implementation of the exact drift would have been facilitated if the code used the canonical momentum p_x and p_y instead of the angles x' and y'. It is suspicious to use the small-angle approximation of p_x and p_y in the calculation of p_z as in Listing 4.3. However, with the present set of coordinates in the code this is the only possible way to perform the implementation.

When tracking in 4D with the momentum deviation $\delta = 0$, the two sets of coordinates (x, p_x, y, p_y) and (x, x', y, y') are the same. It can then be expected that the results should agree very well in this case. This is seen from the 4D benchmarking results where even after 10^6 turns the three codes agree well. When tracking in 6D with a varying δ this perfect agreement can no longer be expected. This is due to the approximation used to calculate p_z. It is difficult to evaluate the exact impact of this, since other factors differing between the codes can contribute as well. However, from the simple 6D benchmark test reported in this thesis, the agreement is very good.

There are a lot of features in SixTrack which does not seem to be optional. The need to include at least one block of linear elements in the `BLOC` part of `fort.3` is one example. If this could be

avoided it would be possible to also implement the exact drift in the thick-lens tracking routines. However, most simulations are performed with thin-lens tracking.

Bibliography

[1] L. Evans and P. Bryant. LHC Machine. *JINST*, 3(S08001), 2008.

[2] F. Schmidt. SIXTRACK: Single particle tracking code treating transverse motion with synchrotron oscillations in a symplectic manner. Technical Report 94-56, CERN, 1994.

[3] CERN. LEP design report. Technical Report LEP-84-01, CERN, 1984.

[4] Maximilien Brice. Views of the LHC tunnel sector 3-4. Obtained from `http://cds.cern.ch/record/1211045?ln=en`. Accessed: 2013-11-06.

[5] Maximilien Brice. Aerial View of CERN taken in 2008. Obtained from `http://cds.cern.ch/record/1295244`. Accessed: 2013-11-06.

[6] CERN. TE-EPC machines. `http://te-dep-epc.web.cern.ch/te-dep-epc/machines/general.stm`. Accessed: 2013-11-06.

[7] CERN. HL-LHC: High Luminosity Large Hadron Collider. `http://hilumilhc.web.cern.ch/HiLumiLHC/index.html`. Accessed: 2013-11-06.

[8] CERN FAQ, LHC the guide. `http://cds.cern.ch/record/1165534/files/CERN-Brochure-2009-003-Eng.pdf`. Accessed: 2013-11-06.

[9] The ATLAS Collaboration. The ATLAS experiment at the CERN LHC. *JIST*, 3(S08003), 2008.

[10] The CMS Collaboration. The CMS experiment at the CERN LHC. *JIST*, 3(S08004), 2008.

[11] The ATLAS Collaboration. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys. Lett. B*, 716:1, 2012.

[12] The CMS Collaboration. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys. Lett. B*, 716:30, 2012.

[13] The ALICE Collaboration. The ALICE experiment at the CERN LHC. *JIST*, 3(S08002), 2008.

[14] The LHCb Collaboration. The LHCb Detector at the LHC. *JIST*, 3(S08005), 2008.

[15] R. W. et al Assmann. *A 3 TeV e^+e^- Linear Collider Based on CLIC Technology.* CERN, Geneva, 2000.

[16] H. Grote and F. Schmidt. MAD-X – An Upgrade from MAD8. Technical Report 2003-024, CERN-AB, 2003.

[17] E. Forest, E. McIntosh, and F. Schmidt. Introduction to the polymorphic tracking code: Fibre bundle, polymorphic taylor types and exact tracking. Technical Report 2002-044, CERN-SL, 2002.

[18] H. Goldstein, C. Poole, and J. Safko. *Classical Mechanics.* Addison Wesley, 3rd edition, 2002.

[19] J. R. Rees. Symplecticity in Beam Dynamics: An Introduction. Technical Report PUB-9939, SLAC, 2003.

[20] J. D. Jackson. *Classical Electrodynamics*. John Wiley & Sons, Inc., 3rd edition, 1999.

[21] S. Y. Lee. *Accelerator Physics*. World Scientific Publishing Co. Pte. Ltd., 3rd edition, 2012.

[22] K. Wille. *The Physics of Particle Accelerators*. Oxford University Press, 1st edition, 2000.

[23] K. Heinemann, G. Ripken, and F. Schmidt. Construction of nonlinear symplectic six-dimensional thin-lens maps by exponentiation. Technical Report 95-189, DESY, 1995.

[24] A. Wrulich. RACETRACK: a computer code for the nonlinear particle motion in accelerators. Technical Report 84-026, DESY, 1984.

[25] R. De Maria, R. Bruce, R. Calaga, L. Deniau, M. Giovannozzi, M. Fjellstrom, Y. Levinsen, E. McIntosh, A. Mereghetti, S. Redaelli, H. Renshall, A. Rossi, D. Sinuela, F. Schmidt, R. Tomas, V. Vlachoudis, G. Robert-Demolaize, D. Banfi, J. Barranco, B. Dalena, L. Lari, V. Previtali, R. Appleby, and D. Brett. Recent Developments and Future Plans for SixTrack. Technical Report CERN-ACC-2013-0060, CERN, Geneva, May 2013.

[26] M. Berz. DAFOR – Differential Algebra Precompiler Version 3, Reference Manual. Technical Report 755, MSUCLB, 1991.

[27] E. Forest. LBL differential algebra package and LieLib, unpublished. `http://cds.cern.ch/record/1295244`.

[28] CRlibm correctly rounded mathematical library. `http://lipforge.ens-lyon.fr/www/crlibm/`. Accessed: 2013-12-04.

[29] M. Berz. Differential algebraic treatment of beam dynamics to very high order including applications to spacecharge. *AIP Conferene Proceedings*, 177(1):275–300, 1988.

[30] A. Latina. Implementation of a thick quadrupole in the MAD-X tracking module. Technical Report ACC-NOTE-2013-0021, CERN, 2013.

[31] R. De Maria and M. Fjellstrom. Sixtrack physics manual (draft). Technical report, CERN, 2013.

[32] E. Forest. *Beam Dynamics: A New Attitude and Framework*. Harcourt Academic Publisher, 1st edition, 1999.

[33] CERN. Accelerator Optics. `http://cern-accelerators-optics.web.cern.ch/cern-accelerators-optics/`. Accessed 2013-11-19.

[34] R. W. Assmann. Collimators and cleaning, could this limit the LHC performance? Technical Report AB-2003-008 ADM, CERN, 2003.

[35] O. Brüning, P. Collier, P. Lebrun, S. Myers, R. Ostojic, J. Poole, and P. Proudlock. *LHC Design Report*. CERN, Geneva, 2004.

[36] G. Robert-Demolaize, R. Assmann, S. Redaelli, and F. Schmidt. A New Version of SIXTRACK with Collimation and Aperture Interface. *Proceedings of PAC05*, 2005.

[37] T. Trenkler and J.B. Jeanneret. K2: A Software Package evaluating Collimation Systems in Circular Colliders (Manual). Technical Report SL/94-56 (AP), CERN, 1994.

[38] Brian William St. Leger Montague. Basic Hamiltonian mechanics. *Proceedings of the CERN Accelerator School, 5th Advanced Accelerator Physics Course, Rhodes, Greece, CERN Yellow Report CERN-95-06*, pages 1–14, 1993.

[39] D. P. Barber, K. Heinemann, G. Ripken, and F. Schmidt. Symplectic thin-lens transfer maps for SIXTRACK: Treatment of bending magnets in terms of the exact hamiltonian. Technical Report 96-156, DESY, 1996.

[40] CERN. SixTrack - 6D Tracking Code. `http://sixtrack-ng.web.cern.ch/sixtrack-ng/` ¿ sources.

[41] CERN. LHC Collimation Project. `http://lhc-collimation-project.web.cern.ch/lhc-collimation-project/code-tracking-2012.php`. Accessed 2013-11-18.

Appendix A

Common acronyms

ALICE	A Large Ion Collision Experiment
ATLAS	A Toroidal LHC ApparatuS
CERN	Conseil Européen pour la Recherche Nucléaire
CLIC	Compact Linear Collider
CMS	Compact Muon Solenoid
IP	Interaction Point
IR	Interaction Region
LEP	Large Electron-Positron collider
LHC	Large Hadron Collider
LHCb	LHC beauty
LS	Long Shutdown
MAD	Methodical Accelerator Design
PTC	Polymorphic Tracking Code
PS	Proton Synchrotron
SC	Superconducting
SPS	Super Proton Synchrotron

Appendix B

Derivation of the accelerator Hamiltonian

The derivation in this appendix follow the steps outlined in Reference [38].

The Lagrangian for a charged particle of mass m_0 and charge q traveling in an electromagnetic field with speed v is

$$\mathcal{L} = -m_0 c^2 \sqrt{1 - \frac{|\mathbf{v}|^2}{c^2}} - q\phi + q\mathbf{v} \cdot \mathbf{A}, \tag{B.1}$$

where $\mathbf{A}$ is the electromagnetic vector potential and ϕ is the scalar potential from which the electric and magnetic fields $\mathbf{E}$ and $\mathbf{B}$ are derived as

$$\begin{aligned} \mathbf{E} &= -\nabla\phi - \frac{\partial \mathbf{A}}{\partial t}, \\ \mathbf{B} &= \nabla \times \mathbf{A}. \end{aligned}$$

B.1 Straight coordinate system

To simplify the derivation, begin by considering a straight coordinate system (x, y, z) moving in the z-direction. The Hamiltonian H is derived from the Lagrangian as

$$H = \dot{q}_i p_i - L(q, \dot{q}, t). \tag{B.2}$$

Inserting the Lagrangian $\mathcal{L}$ from Equation (B.1) in Equation (B.2) gives

$$H = \sqrt{(\mathbf{p} - q\mathbf{A})^2 c^2 + m^2 c^4}, \tag{B.3}$$

where the scalar potential has ben set to zero, $\phi = 0$. This Hamiltonian is expressed with time the t as the independent coordinate. It is favorable and intuitive to instead introduce s, the position along the accelerator, as the independent coordinate. A particle circulating in an accelerator passes an element at a certain position in the accelerator, but the time when the particle passes the element is not as clear. This change of independent coordinate is possible only if t is a monotonically increasing quantity, which is true in this case.

To perform the change of independent coordinate, consider the variation of the action S [18], which should be zero,

$$\delta S = \delta \left[\int_{t_0}^{t_1} \mathcal{L}\, dt \right] = 0. \tag{B.4}$$

Inserting the Lagrangian as expressed in Equation (B.2) gives the action

$$S = \int_{t_0}^{t_1} \left(p_x \dot{x} + p_y \dot{y} + p_z \dot{z} - H \right) dt \tag{B.5}$$

where the dot represents derivation with respect to t. Now changing the integration variable to z by, $dt = \frac{dt}{dz}dz$

$$S = \int_{z_0}^{z_1} \left(p_x \frac{dx}{dz} + p_y \frac{dy}{dz} - H \frac{dt}{dz} + p_z \right) dz. \tag{B.6}$$

Comparing this with Equation (B.5), to correctly describe the mechanics in the Hamiltonian framework with s as the independent coordinate the new Hamiltonian must be chosen as $\hat{H} = -p_z$ and the canonical coordinate pairs as (x, p_x), (y, p_y) and $(-t, H)$. Solving for the new Hamiltonian gives

$$\hat{H} = -\sqrt{\frac{E^2}{c^2} - m^2c^2 - (p_x - qA_x)^2 - (p_y - qA_y)^2} - qA_z, \tag{B.7}$$

where the old Hamiltonian H has been replaced with the total energy of the particle, E. Normalizing the transverse momentum variables as $\bar{p}_i \to \frac{p_i}{P_0}$, as well as performing the substitution $\bar{H} \to \frac{\hat{H}}{P_0}$ to make Hamilton's equations remain invariant gives

$$\bar{H}P_0 = -\sqrt{\frac{E^2}{c^2} - m^2c^2 - (P_0\bar{p}_x - qA_x)^2 - (P_0\bar{p}_y - qA_y)^2} - qA_z, \tag{B.8}$$

or by simplifying further

$$\bar{H} = -\sqrt{\frac{E^2}{P_0^2c^2} - \frac{m^2c^2}{P_0^2} - (\bar{p}_x - a_x)^2 - (\bar{p}_y - a_y)^2} - a_z. \tag{B.9}$$

Here $a_i = \frac{q}{P_0}A_i$ are the normalized components of the vector potential. By using the relation between momentum, mass and energy, $(Pc)^2 = E^2 - (mc^2)^2$, the two first terms inside the square root can be simplified as

$$\frac{E^2}{P_0^2c^2} - \frac{m^2c^2}{P_0^2} = \frac{E^2 - m^2c^4}{P_0^2c^2} = \frac{P^2c^2}{P_0^2c^2} = (1+\delta)^2, \tag{B.10}$$

where the relative momentum deviation is $\delta = \frac{P - P_0}{P_0}$ and $1 + \delta = \frac{P}{P_0}$. The Hamiltonian is then

$$H_1 = -\sqrt{(1+\delta)^2 - (p_x - a_x)^2 - (p_y - a_y)^2} - a_z. \tag{B.11}$$

The canonical variable pair $(-t, E)$ is not very convenient to use since t is an ever increasing quantity. Therefore, introduce the generating function for a canonical transformation from the old variables $(x, \bar{p}_x, y, \bar{p}_y, -t, E)$ to the new variables $\left(X, P_x, Y, P_y, \sigma = s - \beta_0 ct, p_\sigma = \frac{E - E_0}{\beta_0 P_0 c}\right)$ as

$$F_2 = xP_x + yP_y + (s - \beta_0 ct)\left(p_\sigma + \frac{E_0}{\beta_0 P_0 c}\right). \tag{B.12}$$

The new position coordinates and the new Hamiltonian is obtained from

$$\begin{aligned} Q_i &= \frac{\partial F_2}{\partial P_i}, \\ K &= H + \frac{\partial F_2}{\partial s}. \end{aligned} \tag{B.13}$$

This gives

$$X = x, \qquad Y = y, \qquad Z = = z - \beta_0 ct = \sigma. \tag{B.14}$$

The transverse momentum variables are left unchanged ($P_x = \bar{p}_x$ and $P_y = \bar{p}_y$) but the relation between the old and the new longitudinal momentum variable p_σ becomes

$$\frac{E}{P_0} = \beta_0 c \left(\frac{E_0}{\beta_0 P_0 c} + p_\sigma \right). \tag{B.15}$$

The new Hamiltonian is

$$K = p_\sigma - \sqrt{(1+\delta)^2 - (P_x - a_x)^2 - (P_y - a_y)^2} - a_z. \tag{B.16}$$

Finally, rename the variables as $K \to H$, $P_x \to p_x$ and $P_y \to p_y$ and end up with the final accelerator Hamiltonian in a straight coordinate system

$$H = p_\sigma - \sqrt{(1+\delta)^2 - (p_x - a_x)^2 - (p_y - a_y)^2} - a_z. \tag{B.17}$$

B.2 Curved coordinate system

Consider a coordinate system with a curvature in the horizontal plane, but no curvature in the vertical plane. This is the most common case in particle tracking but the final result can easily be generalized to the case with curvature in both planes. A curved reference frame is the most natural frame for a dipole magnet, whose main objective is to bend the trajectory of particles.

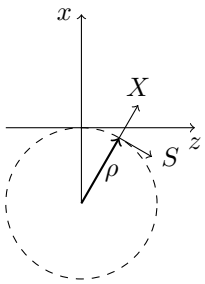


Figure B.1: Curved reference system in the x-z plane.

The coordinates in the straight coordinate system are (x, y, z) and the coordinates in the curved frame are (X, Y, S). From Figure B.1 the relation between these two sets of coordinates are

$$\begin{aligned} x &= (\rho + X)\cos(S/\rho) - \rho, \\ y &= Y, \\ z &= (\rho + X)\sin(S/\rho). \end{aligned} \tag{B.18}$$

Now, construct a generating function which gives the right coordinate transformation, and which will give the corresponding conjugate momenta in the curved reference system

$$F_3(X, p_x, Y, p_y, S, p_z) = -\left[(\rho + X)\cos\frac{S}{\rho} - \rho\right] p_x - Y p_y - \left[(\rho + X)\sin\frac{S}{\rho}\right] p_z. \tag{B.19}$$

The old coordinates (x, p_x, y, p_y, z, p_z) are related to the new coordinates (X, P_x, Y, P_y, S, P_s) through

$$x_i = -\frac{\partial F_3}{\partial p_i}, \qquad\qquad P_i = -\frac{\partial F_3}{\partial X_i}. \tag{B.20}$$

The coordinates transform to the form in Equation (B.18) as desired. The new momentum coordinates are given by

$$\begin{aligned} P_x &= p_x \cos\frac{S}{\rho} + p_z \sin\frac{S}{\rho}, \\ P_y &= p_y, \\ P_s &= p_z\left(1 + \frac{X}{\rho}\right)\cos\frac{S}{\rho} - p_x\left(1 + \frac{X}{\rho}\right)\sin\frac{S}{\rho}. \end{aligned} \tag{B.21}$$

The components of the electromagnetic vector potential are transformed as

$$\begin{aligned} A_X &= A_x \cos\frac{S}{\rho} - A_z \sin\frac{S}{\rho}, \\ A_Y &= A_y, \\ A_S &= A_z \cos\frac{S}{\rho} + A_x \sin\frac{S}{\rho}. \end{aligned} \tag{B.22}$$

Now, the procedure is the same as in the case of a straight coordinate system, except for a factor of $(1 + X/\rho)$ coming from P_S when changing the Hamiltonian from H to $-P_S$. The final result is

$$H = p_\sigma - (1 + hx)\left(\sqrt{(1+\delta)^2 - (p_x - a_x)^2 - (p_y - a_y)^2} + a_s\right). \tag{B.23}$$

$h = 1/\rho$ is the inverse bending radius. Setting $h = 0$ gives the accelerator Hamiltonian for a straight reference system as in Equation (B.17).

Appendix C

Exact dipole implementation

In this appendix, the implementation of an exact dipole kick is described. This dipole kick is dependent on the map for the exact drift in Equation 2.36. Many details are the same as in the exact drift implementation, this appendix is thus more sparse for brevity.

C.1 Equation of motion

In a curvilinear reference frame a dipole magnet with an inverse horizontal bending radius h_x is characterized by the magnetic vector potential

$$A_x = 0, \qquad A_y = 0, \qquad A_s = -B_y x \left(1 - \frac{h_x x}{2(1 + h_x x)}\right).$$

This vector potential gives rise to a pure dipole field in the y-direction. A particle with a positive charge q moving in the direction of increasing s experiences a force towards the right (towards negative x-values, see Figure 2.1).

From Equation (2.12) the Hamiltonian for a particle in a dipole magnet is

$$H = p_\sigma - (1 + h_x x)p_z + \frac{qB_0}{P_0} x \left(1 + \frac{h_x x}{2}\right),$$

where

$$p_z = \sqrt{(1+\delta)^2 - p_x^2 - p_y^2}. \tag{C.1}$$

Assuming that the magnetic field of the dipole is properly matched to the curvature of the reference trajectory, $\frac{qB_0}{P_0} = h_x$, simplifies the Hamiltonian to

$$H = p_\sigma - (1 + h_x x)p_z + h_x x \left(1 + \frac{h_x x}{2}\right). \tag{C.2}$$

In order to derive the equations of motion while keeping p_z unexpanded it is necessary to split the Hamiltonian into solvable parts. The Hamiltonian is split as

$$H = H_I + H_{II}, \tag{C.3}$$

where

$$\begin{aligned} H_I &= -\delta h_x x + \frac{1}{2}(h_x x)^2 \\ H_{II} &= -h_x x(p_z - (1+\delta)) + p_\sigma - p_z \end{aligned} \tag{C.4}$$

These parts can be solved individually [39]. The complete transfer map can be expressed as a composition of maps,

$$\mathcal{M} = T_{II}(L/2) \circ T_I(L) \circ T_{II}(L/2). \tag{C.5}$$

The map $T_{II}(L)$ is the solution to H_{II}, and is given by

$$\begin{aligned} p_x &\to \frac{1}{1+(h_xL)^2}\left[p_x + (h_xL)(1+\delta)\left(\sqrt{1-\frac{p_x^2+p_y^2-C}{(1+\delta)^2}}-1\right)\right], \\ x &\to x + (h_xL)x\cdot\frac{p_x}{p_z}, \\ y &\to y + (h_xL)x\cdot\frac{p_y}{p_z}, \\ \sigma &\to \sigma - (h_xL)x\cdot\frac{\beta_0}{\beta}\left(\frac{1+\delta}{p_z}-1\right), \end{aligned} \tag{C.6}$$

with $C = -(h_xL)^2p_y^2 + 2(h_xL)(1+\delta)p_x$. The map $T_I(L)$ is the solution to H_I, and is given by

$$\begin{aligned} p_x &\to p_x - h_x^2Lx + (h_xL)\cdot\delta, \\ \sigma &\to \sigma - (h_xL)x\cdot\frac{\beta_0}{\beta}. \end{aligned} \tag{C.7}$$

To apply this map for a dipole of length L in a first order thin-lens approximation it must be surrounded by two exact drift spaces of length $L/2$. In the original derivation [39] the contribution of the drift space is included in the map T_{II} in (C.6). In Equation C.6, the contribution of the drift space has been removed.

C.2 Implementation details

Below follows a description of the steps for the calculation of the dipole kick map in SixTrack. The steps describe the calculation for a horizontal dipole. The steps are the same for a vertical dipole, interchanging the roles of x and y (and x' and y'). Some of the details are the same as in the implementation details for the exact drift space, see Section 4.1

C.2.1 Tracking routines

Below, the implementation details for the thin-lens tracking routines are presented. This map is implemented in all three versions of thin-lens tracking.

1. At the entrance of the kick, the coordinates are converted from mm to m. This is done in the exact same way as for the drift, shown in Listing 4.2.

2. The value of h_x is calculated as in Listing C.1. In the code h_x is called `hbend`. This value is calculated from the bending angle `dki(ix,1)` and the length `dki(ix,3)`. These two properties are given in the input file `fort.2`.

Listing C.1: Calculation of h_x in tracking routines.

```
1 hbend=-dki(ix,1)/dki(ix,3)
```

   The sign is reversed, because SixTrack has the opposite definition of the bending radius in the code. This sign reversion is done so that the equations of motion can be implemented as in Equation (C.6) and Equation (C.7).

3. Next the map T_{II} in (C.6) is calculated, the code for this map is shown in Listing C.2.

Listing C.2: Code for the T_{II} map.

```
Lbend=dki(ix,3)*half
theta=Lbend*hbend
cbend=-(theta**2)*(yv(2,j)**2)+two*(theta*yv(1,j))
sqrtterm=sqrt(one-((yv(1,j)**2+yv(2,j)**2)-cbend))
yv(1,j)=(yv(1,j)+theta*(sqrtterm-one))/(one+theta**2)
pz=sqrt((one-(yv(1,j)**2+yv(2,j)**2)))
xv(2,j)=xv(2,j)+((theta*xv(1,j))*yv(2,j))/pz
sigmv(j)=sigmv(j)+(theta*(rvv(j)*(xv(1,j)*(one-one/pz))))
xv(1,j)=xv(1,j)+((theta*xv(1,j))*yv(1,j))/pz
```

In the first line half the length of the dipole is stored in `Lbend` and then half of the bending angle is calculated in `theta`. The following lines calculate the map, by breaking the calculations down in smaller steps.

4. Next, the map T_I in Equation (C.7) is performed. The code for this step is shown in Listing C.3.

Listing C.3: Code for the T_I map.

```
Lbend=dki(ix,3)
theta=hbend*Lbend
yv(1,j)=yv(1,j)-(hbend*theta*xv(1,j)-(theta*dpsv(j)))/
&(one+dpsv(j))
sigmv(j)=sigmv(j)-theta*rvv(j)*xv(1,j)
```

5. Then, the map T_{II} is applied again, as in Listing C.2.

6. Finally, a conversion back to m is performed. The same code as in Listing 4.4 is used.

C.2.2 DA routine

Below the implementation details for the DA closed orbit and optics calculations are presented. The calculations follow the same principle as in the tracking routines. For details of the special `*FOX` notation, see the implementation details for the exact drift space in Section 4.1.

1. First, a conversion from mm to m. This is the same code as for the drift space shown in Listing 4.6.

2. The inverse bending radius is assigned to the variable `HBEND`, this is shown in Listing C.4.

Listing C.4: Calculation of h_x in DA routines.

```
*FOX   HBEND=-DKI(IX,1)/DKI(IX,3) ;
```

3. The map T_{II} is then calculated as shown in Listing C.5.

Listing C.5: Code for the T_{II} map in the DA routines.

```
*FOX  LBEND=DKI(IX,3)*HALF ;
*FOX  THETA=LBEND*HBEND ;
*FOX  CBEND=-(THETA*THETA)*(Y(2)*Y(2))+TWO*(THETA*Y(1)) ;
*FOX  SQRTTERM=SQRT(ONE-Y(1)*Y(1)-Y(2)*Y(2)+CBEND) ;
*FOX  Y(1)=(Y(1)+THETA*(SQRTTERM-C1E3))/(ONE+THETA*THETA) ;
*FOX  PZ=SQRT(ONE-Y(1)*Y(1)-Y(2)*Y(2)) ;
*FOX  X(2)=X(2)+((THETA*X(1))*Y(2))/PZ ;
*FOX  SIGMDA=SIGMDA+THETA*RV*X(1)*(ONE-ONE/PZ) ;
*FOX  X(1)=X(1)+((THETA*X(1))*Y(1))/PZ ;
```

4. Then the map T_I is calculated as shown in Listing C.6.

Listing C.6: Code for the T_I map in the DA routines.

```
*FOX  LBEND=DKI(IX,3) ;
*FOX  THETA=HBEND*LBEND ;
*FOX  Y(1)=Y(1)-((HBEND*THETA)*X(1)-THETA*DPDA)/(ONE+DPDA) ;
*FOX  SIGMDA=SIGMDA-THETA*RV*X(1) ;
```

5. Then the map T_{II} is calculated again, following the same code as in Listing C.5.

6. The last step is the conversion from m back to mm. The code is the same as for the drift space, shown in Listing 4.8.

Appendix D

SixTrack input blocks

This appendix provides details on a few additional input blocks for the input file `fort.3` used in a SixTrack simulation. These blocksi, together with the blocks described in Chapter 3, are the ones that appear in most SixTrack simulations.

D.1 Comment line (COMM)

This block provides a chance to add a comment describing the simulation. This comment will appear in the output from SixTrack. This can be useful when scripts are used to automatically parse the output for post-processing.

D.2 Print selection (PRIN)

This block contains no content and is not ended by the `NEXT` keyword. By adding this block the input data will be printed to the file `fort.6`. This can be useful when automatically doing a large number of simulations where the `fort.3` file is changed in each simulation. In this case `fort.6` can be saved as a log of the input.

D.3 Iteration errors (ITER)

This block provides a way to define the maximum number of iterations for closed orbit calculations, tune variations and chromaticity corrections along with desired precision of the calculations. The structure of the input block is shown in Listing D.1.

Listing D.1: `ITER` input block for `fort.3`.

```
1 ITERATION ERRORS-----------------------------------
2 itco dma dmap
3 itqv dkq dqq
4 itcro dsm0 dech
5 de0 ded ds1 aper(1) aper(2)
6 NEXT-----------------------------------------------
```

The first line specifies the number of iterations for the closed orbit calculation (`itco`) along with the desired precision for the closed orbit displacements (`dma`) and the derivative of the closed orbit displacements (`dmap`).

The second line specifies the number of iterations for the tune adjustments (`itqv`) along with the variation magnitude of the quadrupole magnet strengths needed to achieve the desired tunes (`dkq`) and also the demanded precision of the tune values (`dqq`).

The third line specifies the number of iterations for the chromaticity corrections (`itcro`) along with the variation of the magnitude of the sextupole magnets strengths needed to achieve the desired chromaticity (`dsm0`) and also the demanded precision of the chromaticity (`dech`).

The last line specifies the variation of momentum spread for the chromaticity calculation (`de0`), the variation of momentum spread for evaluation of dispersion (`ded`), the demanded precision of the desired orbit r.m.s value (`ds1`) and also the horizontal and vertical aperture limitations in millimeters (`aper(1)` and `aper(2)`).

D.4 Linear optics calculation (LINE)

This input block controls the calculation of the linear parameters of the lattice. The structure of this input block is shown in Listing D.2.

Listing D.2: `LINE` input block for `fort.3`.

```
1 LINEAR OPTICS CALCULATION----------------------------
2 mode  number-of-blocks  ilin  ntco  E_I  E_II
3 name(1) name(2) ... name(nele)
4 NEXT-------------------------------------------------
```

In the first line, `mode` is specified as either ELEMENT or BLOCK. This will cause the linear parameters to be calculated and printed at each single element or each block of elements. The `number-of-blocks` can be used to specify at how many elements or block the linear parameters should be printed. By setting this value to zero the linear parameters will be printed for all elements or blocks in the lattice. `ilin` is a switch to select the traditional 4D closed orbit calculation or the 6D differential algrebra (DA) approach. `ntco` is a switch to select write out of linear coupling parameters.

The emittances of mode 1 and mode 2 are set in `E_I` and `E_II`, respectively.

The last row(s) are used to specify the names of the single elements or blocks where the linear parameters should be printed. The list of elements together with the optical functions will appear in the output before the tracking starts.

D.5 Post processing (POST)

This input block controls a number of post processing options available. The structure of this input block is shown in Listing D.3.

Listing D.3: `POST` input block for `fort.3`.

```
1 POST PROCESSING--------------------------------------
2 comment title
3 iav nstart nstop iwg dphix dphiy iskip iconv imad cma1 cma2
4 Qx0 Qy0 ivox ivoy ires dres ifh dfft
5 kwtype itf icr idis icow istw iffw nprint ndafi
6 NEXT-------------------------------------------------
```

The first line of input is reserved for a comment witch will appear in the output of the run. The second line specifies the amount of data to be saved and processed as well as ways to scaling the data. The third line specifies details about the FFT analysis of the tunes. The last line of input specifies which plots to produce and the number of data-files to be processed.

D.6 List of all blocks

An overview of all the available input blocks in SixTrack is given in Table D.1.

Table D.1: Input blocks for SixTrack simulations.

Block	Descriptive title
BEAM	Beam-beam element
BLOC	Block definition
CHRO	Chromaticity correction
CORR	Tune-shift corrections
COMB	Combination of elements
COMM	Comment line
DECO	Decoupling
DIFF	Differential algebra
DISP	Displacement of elements
FLUC	Random fluctuation starting number
INIT	Initial coordinates
ITER	Iteration errors
LIMI	Aperture limitation
LINE	Linear optics
MULT	Multipole coefficients
NORM	Normal form
ORBI	Orbit adjustment
ORGA	Organisation of random numbers
POST	Post-processing
PRIN	Printout selection
RESO	Resonance compensation
RIPP	Power supply ripple
SEAR	Search for resonance compensation positions
SING	Single elements
STRU	Structure input
SUBR	Sub-resonance calculation
SYNC	Synchrotron oscillations
TRAC	Tracking parameters
TUNE	Tune variation
TROM	"Phase trombone" element

Appendix E

SixTrack build flags

A list of the currently available build flags for SixTrack is provided in this appendix, as well as a brief description of how to build SixTrack.

E.1 Build

The source code can be obtained from the SixTrack web [40]. A regular build of SixTrack using the gfortran compiler on Linux is initiated as

```
./make_six gfortran
```

To include a certain flag simply add the name of the flag, or several names separated by a space as

```
./make_six gfortran flag1 flag2 flag3
```

To exclude a certain flag add a minus sign before the flag name as

```
./make_six gfortran -flag1
```

This is useful since some flags are added by default. The make_six command supports five Fortran compilers

- Intel ifort
- gfortran
- NAG nagfor
- PGI pgf90
- Lahey-Fujitsu lf95

E.2 List of all flags

The following build flags are available in SixTrack

- automatc
- beamgas
- big
- bignblz
- bnlelens
- boinc
- bpm
- collimat
- cr
- crlibm
- ctrack
- debug
- fast
- fio
- hdf5
- hhp
- iibm
- nagfor
- rvet
- small
- tilt
- time
- vvector

Appendix F

Collimation study settings

These are the settings used in the collimation study in SixTrack. The details of the simulation can be found in Section 5.1. The special collimation input block `COLL` is shown in Listing F.1, this block is part of the `fort.3` input file but for the purpose of presentation it is shown separately.

Listing F.1: The `COLL` input block in `fort.3`.

```
1  COLLIMATION-----------------------------------------------------------
2  .TRUE.
3  100 3500000
4  2 5.7 .0015 0. 0. "nothing" 1.129E-4 75.5
5  .TRUE. 12. 15.6 15.6 17.6 5.7 8.5 8.5 17.7 900. 999. 9.8 9.3 999.
6  11.8 26.0 11.8 11.8 11.8 26.0 11.8 11.8
7  0 19789.0 20150.0 1 1
8  -1.3899e-6 -9.345e-5 5.05324e-3 -1.6595e-2 2.15955e-2 -9.96261e-3 1.0
9  -1.3899e-6 -9.345e-5 5.05324e-3 -1.6595e-2 2.15955e-2 -9.96261e-3 1.0
10 1.005E-09 1.005E-09
11 .FALSE. .FALSE. 0 .TRUE. TCP.C6L7.B1 .FALSE. .TRUE. .TRUE. .TRUE.
12 0 0 0 0
13 0 0 0 0 0 0 0 0 0 0 .FALSE.
14 .FALSE. 6.003 .0015
15 0 0 .FALSE. .FALSE.
16 0 .00213 0 0.288e-3 1
17 "CollDB_V6.503_lowb_st.b1.data" 1
18 .TRUE. .FALSE. WAbsVertLowbcoll 101 1 1.
19 NEXT----------------------------------------------------------------
```

The `COLL` input block specifies all settings needed for a collimation simulation in SixTrack. A brief description of the important parameters is given below, and a complete description can be found at [41]. The first line is a logical switch to turn on (`.TRUE.`) or off (`.FALSE.`) the collimation studies. The second row specifies the number of sets of particles to be tracked, each set contains 64 particles.

The beam energy in MeV is also specified, in this case the energy is set to 3.5 TeV. The third line specifies the initial beam distribution. Settings related to the collimators are given in line four to line 15. Line 16 specifies the name of the collimator database and which beam is to be tracked (beam 1 or 2).

The main input file to SixTrack used for the collimation study (Chapter 5) is shown below. The `COLL` input block has been separated from the rest of the input for clarity. See the detailed description of the `COLL` input block below.

Listing F.2: The `fort.3` input file.

```
GEOM-----------------------------------------------
PRINT INPUT----------------------------------------
NEXT-----------------------------------------------
TRACKING PARAMETERS--------------------------------
500 0 32 0 17 0 1
1 1 0 0 0
0 0 1 1 1 20000 2 1 1
NEXT-----------------------------------------------
INITIAL COORDINATES--------------------------------
2 0 0 1
0.0
0.0
0.0
0.0
0.0
0.0
0.0
.000001
0.0
0.0
0.0
0.0
3500000.0
3500000.0
3500000.0
NEXT-----------------------------------------------
FLUCTUATION----------------------------------------
100000 1 7 3
NEXT-----------------------------------------------
ITERATION ACCURACY---------------------------------
50 0.10E-13 0.10E-14
10 0.10E-09 0.10E-09
10 0.10E-04 0.10E-05
0.10E-07 0.10E-11 0.10E-09
NEXT-----------------------------------------------
LINEAR OPTICS--------------------------------------
ELEMENT  0 1 1 3.5 3.5
NEXT-----------------------------------------------
BEAM-----------------------------------------------
0.110E+12 3.5 3.5  0.0755E+00  0.4716E-03 1 1 1
NEXT-----------------------------------------------
SYNCHROTRON OSCILLATIONS---------------------------
35640 .0003225 16 0 26658.864 938.2796 1
1 1
NEXT-----------------------------------------------
ENDE-----------------------------------------------
```