SERPENTINE: A NEW CODE FOR PARTICLE TRACKING

Stephen Molloy*, Stewart Boogert, Royal Holloway, University of London, UK

Abstract

Serpentine is a Python library, written for the purpose of simulating charged particle accelerators. It has been written to allow for the simulation of both rings and single-shot machines in a light-weight way (i.e. without requiring significant computational resources for typical calculations, such as the determination of transfer matrices, or matching of Twiss parameters), and has been structured to be highly modular (i.e. allowing extension of the simulations to include effects not already included in the base installation). Through the use of the Universal Accelerator Parser (UAP), Serpentine has no need for a new lattice representation, and allows access to any lattice format understood by UAP. The operation of this code on several complex accelerator designs is demonstrated.

INTRODUCTION

Computer simulations of charged particle accelerators is fundamental to their construction, and the complexity of these codes has increased not only with advances in computer hardware, but also with the understanding of the physics underlying the performance of these machines.

Several codes, each with its own specialisation, now dominate the field, and almost all current and planned machines are described, either in whole, or in part, using the formats provided by this small subset of packages. While these packages cover almost all of the calculations necessary for design and analysis of these machines, the authors know of no code that operates in a way that allows easy extension of complex features if necessary, while being extremely lightweight in its default state.

Serpentine, the code introduced in this paper, aims to fill this niche.

ADVANTAGES OVER ALTERNATIVES

As mentioned, numerous others packages exist for the purpose of modelling the dynamics of accelerated beams, so it is important to make it clear in which ways the code presented here provides an advantage over these alternatives.

1. Light–weight. Since many of the most common operations required of beamline modelling tools are based on simple matrix manipulations, the default state of this code can be very light computationally.

2. Python–based. Presently, Python is an extremely popular choice of language, and is used to solve a wide range of differing software problems. This large variety of different uses means that a lot of complex data manipulation and calculation problems have already been solved by the wider community, and may simply be imported into the working environment in order to gain that functionality.

3. Free. While other codes rely on proprietary packages for their operation, Serpentine, through its reliance on Python, does not require the purchase of a license to operate.

4. Object–oriented. As a design philosophy, the authors believe that an object–oriented approach has many advantages over a more linear, functional, style. This is a matter of taste, but it is believed that a significant fraction of our users will see a benefit from the ability to use a more modern programming paradigm.

5. Modular. Serpentine has been authored in such a way as to allow easy addition of different elements, more complex physics, or extenstion to different scenarios. For example, it is very simple to inherit from an existing class in order to design a completely new accelerator element.

6. Flexible. The class structure of Serpentine allows simple manipulation and alteration of accelerator designs in a way that is not possible with any other codes known to the authors.

7. Control. A Python module for interfacing with the EPICS control system is available, and, since many accelerators are controlled using this system, this raises the interesting possibilty of controlling an accelerator from an environment identical to that in which simulations were conducted.

PHYSICS

Since this is a beta release of this code, most effort was directed at ensuring the correctness of the physics on which the rest of the code is built, and optimisation for speed was not a primary consideration.

Note that, although future extension to multi-species physics is planned in the near future, Serpentine currently performs all physics based on an assumption that it is electrons that are traversing the machine.

Twiss Propagation and R–matrices

The transfer matrices across each element are calculated and stored with that beamline element object upon instantiation of the Serpentine object holding the lattice description, so calculations of these matrices for individual elements is not necessary, and a determination of the matrix

* stephen.molloy@rhul.ac.uk

between any two points on the beamline can be performed by simple multiplication of the appropriate matrices.

Since the Twiss parameters are purely linear functions of an accelerator lattice, they can be calculated directly from the R-matrices stored with each beamline element. The initial Twiss parameters required for the propagation calculation are stored as an attribute of the Serpentine object.

Tracking

Tracking involves two loops: an outer loop around the elements, and an inner loop around each of the particles.

Although each element already contains an object specifying the transfer matrices, this is only applicable to a particle with the design momentum, so, when looping around the particles, the transfer matrices are recalculated in order to take account of any deviation from that value, and the 6D position vector is transformed by the recalulated matrices. After tracking has been performed for all particles, the transfer matrices are then recalculated for the design momentum, and the outer loop advances to the next element.

Linear Elements

The particle dynamics within the linear elements can be computed by considering the Hamiltonian for the motion of the particles within the EM field.

The Hamiltonians used are based on the usual, curvilinear, reference frame that follows the synchronous particle, where the longitudinal coordinate has been replaced by time, t. This means that the particle coordinates are described in a Hamiltonian formalism in which the position coordinates, x, y, and z, represent the distance along each of the curvilinear degrees of freedom from the synchronous particle in metres, x' and y' are the transverse momenta normalised by total momentum of the particle (and are given in units of radians), and the conjugate partner to z is the particle's fractional momentum error, $\frac{\Delta P}{P}$.

In order to preserve the symplecticity of the resulting equations of motion – the property that the phase space is conserved by these transformations – it is important that, if the equations should be truncated to some order, that the Taylor series truncation is performed on the Hamiltonian itself, and not on the resulting equations of motion. This is due to the fact that Hamilton's equations are guaranteed to preserve symplecticity, while limiting a Taylor series may break this. In other words the equations of motion are always constructed perfectly from the Hamiltonian, however the Hamiltonian itself may be an approximation.

Correctors are the only linear elements not dealt with in precisely this way. Instead they are treated as a drift of half of their length, followed by the appropriate angular kick, and then another drift of half of their length. This is, obviously, an approximation to their true action, however, for the small kicks normally imparted by these devices, this will give a result that is very close to exact.

Higher Order Magnets

While the motion through linear elements may be calculated by solving the equations of motion that result from analysing the Hamiltonian, no closed form solution exists for motion through higher-order elements. Thus these components require a different approach.

Currently, the only higher order element implemented in Serpentine is the thin sextupole (i.e. a sextupole field where it is assumed that the action of the field occurs entirely at the longitudinal centre of the magnet). Thus all sextupoles are modelled as a field–free drift space for half of the length of the sextupole, followed by a kick based on the integrated strength of the field at the location of the particle, and then another drift region of hlaf the length.

While this is obviously a simplification of the physics, it should be noted that motion through a sextupolar field is not soluble in closed form. In addition, by performing a Yoshida factorisation based on the Lie transform of the Hamiltonian [1], it is possible to show that this approximation (that of simplifying the action to a drift–kick–drift), yields results of similar accuracy to a tenth-order truncation of the Taylor series of the Hamiltonian.

Although future releases of this code will include a higher order Yoshida factorisation, and will thus yield even more accurate results for higher order fields, reliable results may still be achieved with the present algorithm.

Acceleration

This beta release of Serpentine does not yet support acceleration of beams, so it is not possible to perform simulations of machines that involve acceleration elements. Due to the fact that this rules out the use of this code for many important simulations, it is expected that this will be changed in the next release of the code.

AN EXAMPLE: BEAM-BASED ALIGNMENT AT ATF2

ATF2 [2] is a test facility currently operating at KEK, Japan. It uses a linac to accelerate a 1.3 GeV electron beam, which is then injected into a damping ring, before extraction into a line designed to vertically focus the beam to 35 nm.

This example tracks the beam from the exit of the damping ring to the dump at the end of the extraction line. The Twiss parameters are shown in figure 1, as well as a comparison with those calculated by Lucretia (an alternative tracking code) in figure 2.

Algorithm

This code implements an algorithm known as *Beam Based Alignment (BBA)*, and demonstrates its effectiveness in determining the offset of a quadrupole focusing magnet.

For a bunch moving through a quad with an offset, the bunch will experience a steering force in addition to the fo-

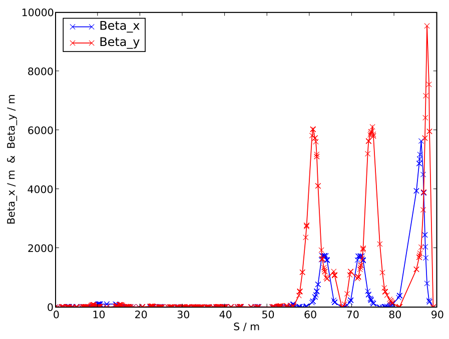

Figure 1: Twiss parameters calculated using Serpentine.

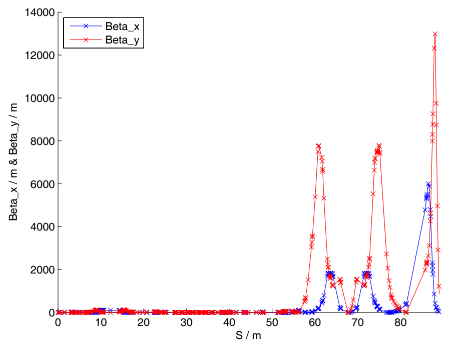

Figure 2: Twiss parameters calculated using Lucretia.

cussing effect, leading to a negative impact on the beam quality. It is, therefore, important to ensure that the beam travels through the magnetic centre of each element, and so it is necessary to deploy a technique to measure any misalignments.

BBA uses a corrector magnet to change the trajectory of the beam through the quadrupole, and measures its position at two BPMs: one located immediately downstream of the quadrupole (the Quad BPM), and another a few metres downstream of that (the witness BPM).

After this, the strength of the quad is dropped by 50%, and the corrector sweep is repeated.

Since a beam travelling through the centre of the quadrupole should not have its trajectory altered by its magnetic field, the corrector setting for which there is no change in position due to the alteration of the quadrupole field, is the setting for which the beam is travelling through its magnetic centre.

Results

Figure 3 shows the x and y orbits before and after the misalignment of the quad.

The simulation performs the corrector sweeps mentioned previously, along with the alteration of the focussing strength of the quad, and records the values from the Quad and witness BPMs. Once the two sweeps have been completed, two straight lines are fit, and their crossing point calculated.

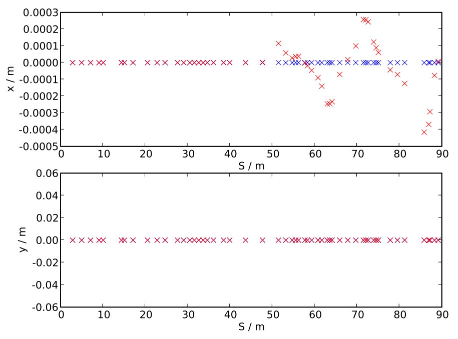

Figure 3: The beam position at each BPM due to a perfect machine (blue), and the misaligned machine (red).

The results are shown in figure 4, and it can be seen that this algorithm does a good job of finding the misalignment given to the quadrupole. The reason the result from the fit (0.104 mm) is not exactly equal to the value given to the quad (0.100 mm) is due to the fact that BPM is located adjacent to the quad, not at precisely the same location, and so the positive angle of the trajectory results in this algorithm slightly over-estimating the results. This is an expected effect that may be eliminated by estimating the magnitude of the angle induced by the corrector field.

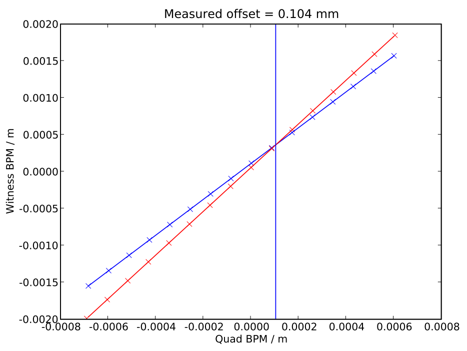

Figure 4: Results of the BBA algorithm.

This example demonstrates the ability of Serpentine to successfully handle common tasks in an accelerator simulation.

REFERENCES

[1] A. Wolsi, "Nonlinear Single-Particle Dynamics in High Energy Accelerators",

`http://pcwww.liv.ac.uk/~awolski/main_teaching_nonlineardynamics.htm`

[2] A. Seryi, et al., "ATF2 Commissioning", Invited talk at PAC09, May 2009