STATUS OF THE ORBIT CODE: RECENT DEVELOPMENTS AND PLANS*

J.A. Holmes, S. Cousineau, A. Shishlo, ORNL, Oak Ridge, TN 37830, U.S.A.

Abstract

We report on recent enhancements to the physics modules of the ORBIT Code and on progress toward a new implementation of ORBIT using python. We have developed the capability to track particles through general three dimensional electromagnetic field configurations. This facility has proved essential in modeling beam transport through the complicated magnetic field regions of the SNS injection chicane and injection dump line, where beam losses are high. We have also enhanced the acceleration module to provide more flexibility for synchrotron calculations and we have developed alternative multiple Coulomb and Rutherford scattering models for the stripper foil and collimation routines. Finally, progress continues on the migration of the ORBIT physics models to a python user environment. We present the status of this work.

INTRODUCTION

ORBIT [1] is a computer code designed specifically for beam dynamics calculations in high intensity accelerators. Its intended use is the detailed simulation of realistic accelerator problems, although it is equally applicable to idealized situations. ORBIT is a particle-in-cell tracking code in 6D phase space that transports bunches of interacting particles through a series of nodes representing elements, dynamic effects, or diagnostics that occur in the accelerator lattice. ORBIT has been designed to simulate real machines: it has detailed models for strip-foil injection including painting, scattering, and nuclear processes; RF focusing and acceleration; symplectic transport through various magnetic elements; alignment and field errors, closed orbit calculation, and error correction; longitudinal and transverse impedances; longitudinal, transverse, and three-dimensional space charge forces; feedback stabilization of instabilities; beam-in-gap cleaning, collimation, and limiting apertures; self-consistent electron cloud dynamics; and the calculation of many useful diagnostic quantities.

ORBIT is an object-oriented code, written in C++ with a scripting interface to facilitate interactive programming. Its basic classes are herds, which are groups of particles, and nodes, which operate on the herds. A conversion of the SuperCode [2] scripting interface to the standard scripting language, Python, is underway. ORBIT supports parallel computing using MPI. The ORBIT code is an open source, powerful, and convenient tool for studying beam dynamics in high-intensity rings.

Recent work on the ORBIT physics models includes the development of a tracker for particles in 3D magnetic field configurations. This was motivated by the necessity of detailed tracking of H^0, H^-, and H^+ beam components in the SNS injection chicane and dump line, where beam losses are large. Because an increasing number of ORBIT users are interested in synchrotron applications, we have spent some time generalizing the acceleration module. For example, in addition to acceleration, it is now capable of deceleration. The ORBIT collimation and foil scattering packages are adaptations of the K2 collimation code and the physics models are described in Ref. [3]. We have recently developed additional models for multiple Coulomb and Rutherford scattering following the presentation in Classical Electrodynamics, by J.D. Jackson [4]. Finally, work has begun on the development of a new version of ORBIT in which the SuperCode driver shell is replaced by python. This involves significant restructuring of the code architecture, and the end result should be a more flexible package at the shell level.

3D PARTICLE TRACKER

It is sometimes necessary to study the properties of particular accelerator elements with more sophisticated tools than standard accelerator models. This proved to be the case in the SNS injection and extraction systems, where the detailed properties of the chicane and septum dipoles have profound effects on the beam transport and losses [5]. Intensive studies of the field configurations of these magnets [6] were calculated using the OPERA-3D/TOSCA code [7]. In order to track charged particles through the local 3D fields produced by such calculations, we developed a leapfrog, predictor-corrector tracker in 6D phase space. This tracker was benchmarked for known field configurations and found to converge to the correct answers for reasonable parameter values. Figure 1 shows the horizontal H^0 and H^- waste beam components from 3D tracking through the original and redesigned injection dump septum magnets. This magnet was redesigned to provide a greater aperture for the extracted beams. Such calculations are critical in the ongoing effort to understand losses in SNS and to mitigate them where necessary.

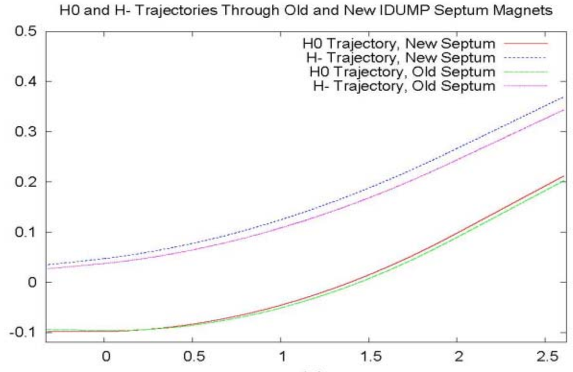

Figure 1. Horizontal H^0 and H^- waste beam trajectories through original and redesigned IDUMP septum magnets.

---

* ORNL/SNS is managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR22725.

ACCELERATION MODULE

Because ORBIT was developed with the SNS accumulator ring in mind, the original acceleration model was fairly rudimentary. As the number of ORBIT users has increased, more applications involving acceleration are being pursued. In response to the demand for a more comprehensive acceleration package, we have enhanced ORBIT's acceleration model. In particular, we reviewed and thoroughly debugged the acceleration routines and have also added some additional capabilities.

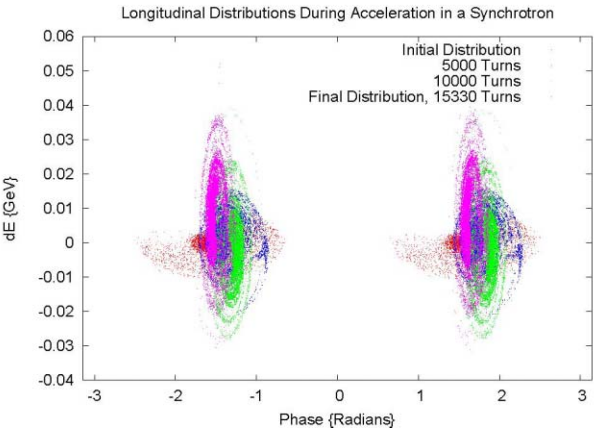

Figure 2. Longitudinal distributions during acceleration from 0.4 GeV to 3.0 GeV in a proton synchrotron.

To define the acceleration of the synchronous particle, it is necessary to provide two of the following three quantities: the synchronous particle momentum or magnetic field ($\rho B = q/p$), the RF voltage, or the synchronous particle RF phases. From any two of these, the third can be calculated. In the initial version of ORBIT, the user was required to supply the magnetic fields and RF voltages as functions of time, from which the synchronous particle phases were calculated and the acceleration carried out. While this is still the most frequently used option, we now provide the alternative possibility of specifying the RF voltages and synchronous particle phases to calculate the synchronous momentum. Some users prefer to work in a longitudinal coordinate system where the synchronous particle is centered at phase $\varphi = 0$, and we have now provided this option. Finally, we now allow for negative acceleration, so that it is possible to slow beams down as well as speed them up. Figure 2 shows the longitudinal beam distributions at four times during the acceleration of a proton beam from 0.4 GeV to 3.0 GeV in a rapid cycling synchrotron with second harmonic RF.

FOIL AND COLLIMATOR

As stated above, the ORBIT collimation and foil scattering packages are adaptations of the K2 collimation code. The physics models include beam energy loss due to interaction with electrons, small angle multiple Coulomb scattering, Rutherford scattering, elastic nuclear scattering, and inelastic nuclear scattering. These models, presented in Ref. [3], have been benchmarked against K2. However, in some recent calculations of foil scattering in SNS, we observed some deviation between the ORBIT results and the predictions of the models presented in Classical Electrodynamics, by J.D. Jackson [4], for multiple Coulomb and Rutherford scattering. Therefore, as an alternative to the K2 multiple Coulomb and Rutherford scattering models, we have included the option to use those presented in Jackson's text.

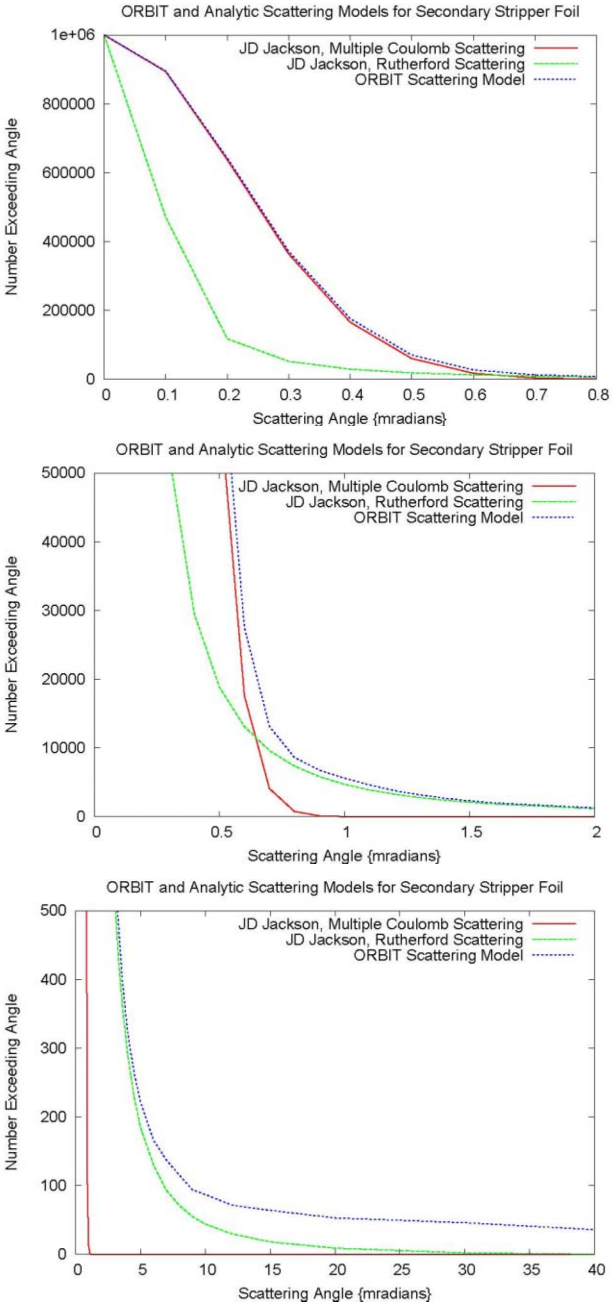

Figure 3. Distribution of scattering angles due to the secondary carbon foil of density 18 mg/cm^2.

Figure 3 shows the number of particles, from a sample of 10^6 particles, with scattering angles exceeding the indicated value for the secondary stripper foil in SNS, which consists of 18 mg/cm^2 carbon. The top figure shows small angle scattering (< 0.8 mr). In this regime, the ORBIT results agree very well with the multiple Coulomb scattering formulation in Jackson. The middle figure shows the transition from multiple Coulomb to Rutherford scattering between about 0.6 and 0.8mr. The bottom figure shows the scattering at large angles. The difference between the ORBIT and Rutherford scattering results is due to the nuclear elastic scattering in ORBIT. ORBIT takes account of inelastic nuclear scattering by removing inelastic scattered particles from the beam. For the secondary stripper foil, the ORBIT calculation yields 119 particles, about 0.01% of the beam, to inelastic scattering.

PYTHON ORBIT

The development of a version of ORBIT to run under a Python driver shell remains a work in progress. The approach is to place all the higher level code under Python, so the user will typically interact with Python ORBIT through Python shell scripts and routines. C++ code will be used for the numerically intensive operations, and will be wrapped to be accessible from the Python shell. We have begun with an organization of the higher level structure of the code at the Python level. The basic classes are bunches and elements. Bunches are collections of macroparticles defined by their properties. The list of properties can be extended in the sense that the user can add additional properties to those that are provided with the class. The elements can perform actions on bunches and are arranged in linked lists called lattices. At present, these classes have been defined and coded, and parsers have been written to read MAD [8] input files and to prepare the elements for conversion into symplectic ORBIT tracker elements. There is still much work to do in porting ORBIT's extensive physics and diagnostic capabilities to this new framework, and this effort progresses only as time permits.

CONCLUSIONS

We continue to extend and improve the physics models in ORBIT as needs dictate. Recent extensions include the development of a particle tracker for 3D fields, enhancements to the acceleration module, and the development of alternative models for multiple Coulomb and Rutherford scattering. The development of a new Python-based version of ORBIT continues slowly, as allowed by our schedules. The higher level structure of the code has been defined and written, but most of the physics modules remain to be ported.

REFERENCES

[1] J.A. Holmes, S. Cousineau, V.V. Danilov, S. Henderson, A. Shishlo, Y. Sato, W. Chou, L. Michelotti, and F. Ostiguy, *in The ICFA Beam Dynamics Newsletter*, Vol. 30, 2003.

[2] S.W. Haney, Using and Programming the SuperCode, http://www.sns.gov/APGroup/Codes/Codes.htm

[3] S. Cousineau, PhD Dissertation, Indiana University, 2003.

[4] J.D. Jackson, "Classical Electrodynamics," Wiley, (New York: 1967).

[5] J.A. Holmes, S. Cousineau, M.A. Plum, and J.G. Wang, "Computational Beam Dynamics Studies for Improving the Ring Injection and Extraction Systems in SNS", EPAC'08, Genoa, June 2008, THPC015.

[6] J.G. Wang and M.A. Plum, Phys. Rev. ST Accel. Beams 11, 014002 (2008).

[7] Vector Fields: Software for Electromagnetic Design, "TOSCA", Cobham Group Company, 2005, http://www.vectorfields.com/content/view/27/5

[8] H. Grote and F. Christoph Iselin, The Mad Program, Version 8.19, User's Reference Manual, CERN/SL/90-13, (Geneva, 1996).