STATUS OF PY-ORBIT: BENCHMARKING AND NOISE CONTROL IN PIC CODES [*]

J.A. Holmes, S. Cousineau, A. Shishlo, Oak Ridge National Laboratory, Oak Ridge, TN, USA

Abstract

PY-ORBIT is a broad collection of accelerator beam dynamics simulation models, written primarily in C++, but accessed by the user through Python scripts. PY-ORBIT was conceived as a modernization, standardization, and architectural improvement of ORBIT, a beam dynamics code designed primarily for rings. Although this goal has been substantially achieved, PY-ORBIT has additional capabilities. A major consideration in high intensity beam dynamics codes, such as PY-ORBIT and ORBIT, is the simulation of space charge effects. Computational space charge simulation is, of necessity, accompanied by noise due to discretization errors, which can compromise results over long time scales. Discretization errors occur due to finite step sizes between space charge kicks, due to graininess of the numerical space charge distribution, and due to the effects of spatial grids embedded in certain solvers. Most tracking codes use space charge solvers containing some or all of these effects. We consider the manifestation of discretization effects in different types of space charge solvers with the object of long time scale space charge simulation.

PY-ORBIT

PY-ORBIT [1] is a collection of computational beam dynamics models for accelerators, designed to work together in a common framework. It was started [2] as a "friendly" version of the ORBIT Code [3], written using publicly available supported software. Users run the code using Python scripts and the higher-level routines are in Python. The computationally intensive portions are written in C++, except for the Polymorphic Tracking Code, PTC [4], which is linked to PY-ORBIT and written in Fortran. The C++ routines and PTC are wrapped to make them available at the Python level. PY-ORBIT accommodates multiprocessing through MPI. The only additional software required by PY-ORBIT is the FFTW fast Fourier transform library [5]. The code is finding an increasing number of users and the source code is publicly available via Google Codes [1]. It is not difficult for users to develop extensions to PY-ORBIT, and it is possible for users to obtain permission to add new routines to the publicly available code. Recently, researchers at CERN and at GSI have added new methods to PY-ORBIT.

At present, PY-ORBIT is capable of performing most calculations that ORBIT does for rings and transfer lines. Many of the most widely used methods in ORBIT have been ported to PY-ORBIT and benchmarked. Single particle tracking methods include ORBIT's native symplectic tracker, PTC tracking, and a 3D field tracker. It is planned to add the option to use linear and second order tracking by matrices from the MAD codes [6,7]. Space charge models include ORBIT's longitudinal, 2D potential and direct force methods, a full 3D (not parallel) solver, and uniform charge density 3D ellipses. A 2.5D solver has recently been added and tested by Hannes Bartosik of CERN. ORBIT's longitudinal impedance model has also been ported to PY-ORBIT, and we are now in the process of porting the transverse impedance model. Other methods that have been ported from ORBIT include injection, foil and painting, RF cavities, collimation, apertures, and many diagnostics.

Some capabilities have been developed in PY-ORBIT that are not included in the original ORBIT code. Routines have been developed for linac modeling, including RF cavities, magnets, and 3D full Particle-in-Cell (PIC) and elliptical space charge models. A detailed set of atomic physics routines has been developed for laser stripping applications, and special maps have been developed for nonlinear optics studies. One ORBIT code package that has not been ported to PY-ORBIT is the self-consistent electron cloud model.

Detailed documentation of PY-ORBIT is, at best, incomplete. In the downloaded source code there are many examples that demonstrate the use of models in scripts. Some of the methods are documented in Google Code wikis. When in doubt, the user is advised to contact one of the developers for detailed answers. Finally, some of the nice features of PY-ORBIT come from the flexibility of Python. For example, the bunch class is extendable. The basic bunch contains only the 6D coordinates of each macroparticle. It is easy, however, for the user to add various properties, such as a particle index, spin, species, ionization number, excited state, etc. Because of this flexibility, it is easy and convenient to work with PY-ORBIT.

SPACE CHARGE MODELING OVER LONG TIMES

Particle-tracking simulations for accelerators involve following particle distributions over time. Space charge physics has been successfully incorporated into particle-tracking studies of linacs, transfer lines, accumulator rings, and rapid cycling synchrotrons (RCS). These space charge models have allowed the successful simulation of phenomena that would have been impossible otherwise. Even so, the evaluation of space charge effects is typically

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

Copyright © 2014 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-173-1

the most computationally intensive part of the simulation. With the emergence of ever-higher beam intensities, it is now necessary to incorporate space charge effects into simulations of storage rings [8]. Calculations for storage rings require tracking beams for far longer times than those in linacs, accumulator rings, or rapid cycling synchrotrons. These long time scales place severe requirements on the speed and accuracy of the physics models and call for innovative methods of solution. At long time scales, there are also other effects besides space charge, such as lattice imperfections and nonlinearities, wake forces, neutral gas ionization and scattering, electron cloud interaction, intrabeam scattering, beam loss, and others, that may be critical. These effects are often relatively unimportant at short or intermediate time scales. To study them in the presence of space charge, it will be necessary to use an accurate depiction of the space charge forces.

Space charge models attempt to account for the inter-particle Coulomb force, and their evaluation requires the solution of Poisson's equation. The first issue in space charge simulation is to choose a model that contains physics sufficient to address the problem. The tools of choice in most space charge simulations are fast Poisson solvers of various dimensionalities that directly use the tracked particle distribution to obtain the space charge force. The speed of fast solvers scales as order O(~M), where M is the number of macroparticles. The most popular of these solvers are the PIC methods in which the particle charges are distributed to a selected set of mesh points. This is followed by the solution of the resulting potential or forces at the mesh points using fast Fourier transforms (FFTs), and then the interpolation of the forces to the particle locations. Many PIC methods are described in Refs. [9,10]. Another order O(~M) method is the fast multipole method (FMM) [11], which expands the individual particle potentials as multipoles at the centers of a collection of square gridded cells containing the particles. These expansions are then shifted and accumulated through a hierarchy of coarser "parent cells" and the resulting totals are converted to Taylor series expansions as they are shifted backwards through the hierarchy of "child cells". The result is a set of local Taylor series expansions in each initial cell for the potential and force due to the particles in distant cells. The method uses pairwise force evaluations for nearby particles to eliminate grid-based discretization effects completely. The FMM solves the N-body force evaluation to machine precision when enough terms are retained in the multipole and Taylor series expansions. Even so, there are discretization effects due to the time step and the numerical particle distribution.

For long time scale calculations, the appropriate choice is usually a 2D or 2.5D transverse solver. The latter choice is necessary when transverse properties vary longitudinally along the beam or when transverse impedances are of interest. These solvers can be used in conjunction with a separate 1D longitudinal model, since longitudinal evolution typically occurs much more slowly. However, some 2.5D solvers also incorporate the longitudinal force. The above models are valid only for long bunches in which the bunch length greatly exceeds the beam pipe radius, as is normally the case in proton storage rings. For short bunches where the longitudinal and transverse dimensions are comparable, such as those found in linacs, it is necessary to use full 3D space charge models. Computational requirements rise steeply with the dimensionality of the model, so it is important to adopt the simplest model that contains the necessary physics.

Important issues that determine the applicability of space charge models are their ability to represent the beam distribution, nonphysical effects associated with the algorithm, and their computational speed. Before discussing the pros and cons of specific approaches, it is important to consider the nonphysical effects. These effects are the result of discretization errors due to finite time step size, the coarseness of the computational particle distribution, and the use of finite spatial grids. Time discretization is a feature of any space charge simulation, regardless of beam representation. While space charge forces act continuously in classical dynamics, simulations apply them as impulses, separated by single particle transport. At the very least, it is necessary to include many space charge evaluations per betatron or synchrotron oscillation. Further discretization errors occur when the tracked particles are used directly to provide the charge/current distribution. Real accelerator bunches typically have orders of magnitude more particles than bunches used in simulations. This results in an increased graininess of the force distribution and an increased potential for large binary collisions in simulations. Both of these effects introduce noise, or diffusion, into the particle evolution. The problem of the enhanced binary collisions is often handled by introducing artificial smoothing parameters into the inter-particle force Green's functions. A final source of discretization in many PIC methods relates to the use of spatial meshes. However, because of the particle binning and force evaluation algorithms used in PIC models, gridding actually provides a smoothing of the local space charge forces. Before realizing this to be the case, we implemented an FMM solver in ORBIT in order to eliminate grid-based numerical effects. However, we discovered that, unless we introduced a smoothing parameter to the pairwise Coulomb force function, numerical diffusion was stronger than in the grid-based FFT methods. However, through the adjustment of the smoothing parameter, we were able to achieve answers and emittance growth rates comparable to those of FFT methods. Figure 1 compares RMS emittance evolution from FFT and FMM simulations with 10^5 macroparticles for a KV distributions and a uniform focusing lattice. The period for one bare tune betatron oscillation is 50 meters, and the space charge tune shift is one third of the bare tune. The results are shown for 100 bare tune betatron oscillations, or 5000 meters. The FMM calculations are carried out alternatively with no smoothing and with an interparticle force smoothing parameter of 0.03 mm. Clearly, numerical scattering leads to emmitance growth in the unsmoothed FMM calculation, and it is necessary to include the smoothing parameter to eliminate the

ISBN 978-3-95450-173-1

Copyright © 2014 CC-BY-3.0 and by the respective authors

scattering. The grid and binning procedures provide this smoothing in the FFT approach. The FMM approach was abandoned because it provided no physics advantage over FFT methods and also because it ran a factor of ten to fifty times more slowly on sample problems, depending on parameter settings.

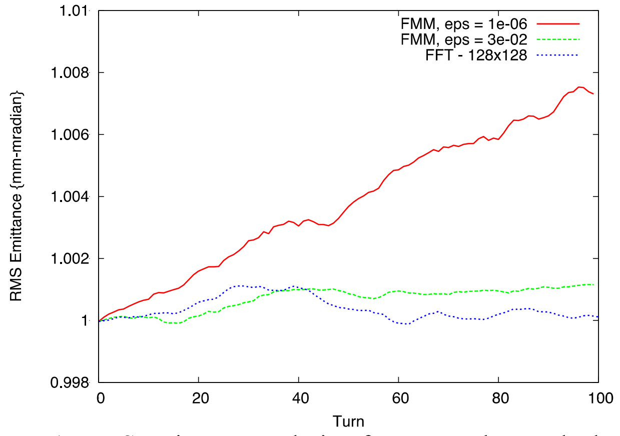

Figure 1: RMS emittance evolution for FFT and smoothed and unsmoothed FMM calculations.

In statistical and plasma physics collisions have been long characterized in terms of diffusion operators. In Ref. [12], a Fokker-Planck analysis was used to characterize entropy growth due to collisions and other stochastic processes in particle beams. More recently, this work was adapted to develop and successfully test an empirical scaling law for the collision frequency and subsequent emittance growth due to space charge in computer simulations [13]. The work in Ref. [13] was carried out for 2D space charge in FODO channels, and the results are therefore applicable to the case of PIC simulation in storage rings. From the standpoint of space charge modeling over long time scales, the essential result is that the numerical collision frequency scales as

$$\nu \propto \frac{N^2}{M}(1-\frac{\Delta}{\lambda})$$

Here N is the beam intensity or number of physical particles, M is the number of macroparticles, Δ is the macroparticle size (smoothing length or grid spacing in PIC methods), and λ is a cutoff parameter. The collision frequency ν is directly related to the entropy and emittance growth of the beam. Thus, the rate of emittance growth increases as the square of the intensity and inversely with the number of macroparticles. The effect of finite grid or smoothing parameter size reduces the growth. Figure 2a shows the effect of grid smoothing on emittance growth for the same case as in Fig. 1, except that now the calculation is carried out for 1000 bare tune oscillations. Figure 2b shows the reduction in emittance growth due to increasing number of macroparticles.

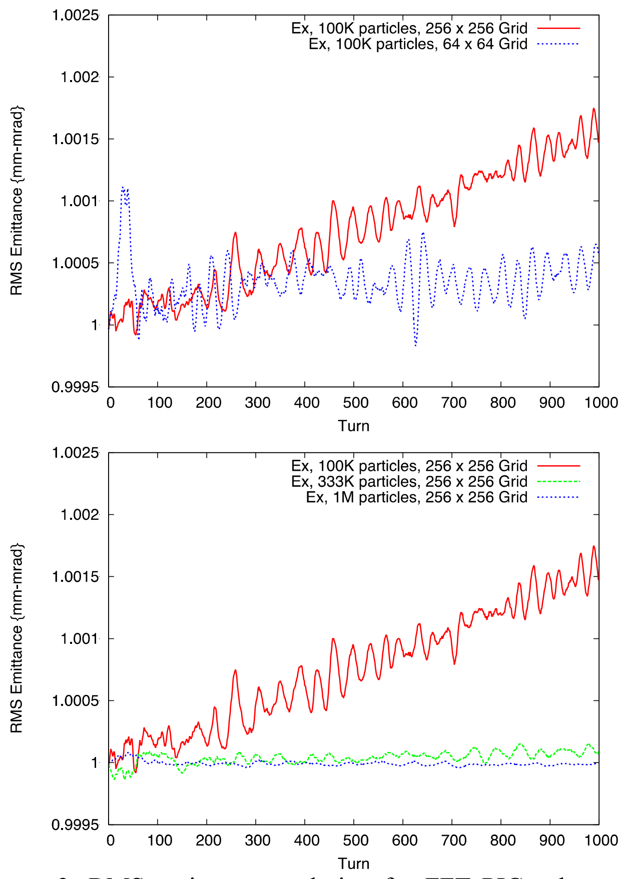

Figure 2: RMS emittance evolution for FFT PIC solver with a) two different grid sizes and b) different numbers of macroparticles.

Given the results of Ref [13], we are now prepared to consider the choice of space charge model for long time scale simulations. Direct use of the distribution of tracked macroparticles to obtain the space charge forces has been the method of choice in previous simulations of linacs, transfer lines, accumulator rings, and RCS. These methods are preferred in that they provide a faithful representation of the beam distribution, which is often complex in real accelerators. In these previous applications it was possible to use enough macroparticles M to suppress the stochastic numerical emittance growth and carry out the calculation with a moderate amount of computer resources. Such simulations could typically be done on small clusters or even individual workstations. However, time scales for storage rings are orders of magnitude longer than those for linacs and small rings, and the above scaling law suggests that the number of macroparticles necessary to limit numerical emittance growth could become prohibitively large. Thus, the direct extension of PIC solution techniques to storage ring applications is likely to be extremely expensive from a computational standpoint, and may demand the use of massively powerful parallel supercomputers.

Thus we arrive at the following question: Is there any alternative to massively parallel computing for studying the effects of space charge in storage rings? The answer

Copyright © 2014 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-173-1

may be some simplified smooth or analytic representation of the space charge force. One drawback of such an approach is that it implies a limited ability to represent the actual beam distribution. One extreme example, called the envelope or particle core model, represents the space charge distribution as a uniform ellipsoidal core that propagates according to the envelope equation. The tracked particles feel the space charge force due to the propagating core. This force is linear inside the core. Envelope models are computationally fast and easy to apply. They have been used to support theoretical studies of halo generation by mismatched beams and also the approach to the half integer resonance. One limitation of such models is that a constant emittance is specified in the envelope and thus constrains the evolution of the core. Accordingly, envelope models are simple, but far from realistic. One can envision intermediate classes of methods for handling space charge through the use of analytic or smoothed distributions, where the parameters are fitted to those of a tracked macroparticle bunch beam. Such hybrid methods could, in principle, enjoy the speed and simplicity of envelope models, have parameters that evolve with the tracked beam distribution, and eliminate the noise associated with the graininess of the particle distribution.

BENCHMARKING OF SPACE CHARGE MODELS

As with all numerical simulation tools, it is essential to benchmark space charge routines. During its fifteen years of existence, ORBIT has been extensively benchmarked, including all its space charge models. Benchmark tests have included comparisons with analytic results, with experimental observations, and with other codes. PY-ORBIT has been thoroughly benchmarked with ORBIT as models have been completed and tested. From this standpoint, we place the same confidence in the models in PY-ORBIT that we have in ORBIT.

There are also formal benchmark tests for space charge routines. An excellent suite of benchmarks on several time scales involves space charge induced resonance trapping, and can be found in Ref. [14]. ORBIT has successfully completed eight of the nine benchmarks on this site, and the final case is under study. Some of the longer time scale tests, as posed on the site, are accompanied by significant numerically-induced emittance growth. Because of this, results in some cases are very sensitive to the parameters used in the calculation. This sensitivity is illustrated in Fig. 3, which shows the emittance evolution with resonant trapping and detrapping over one synchrotron oscillation for two initially close test particles. The particle plotted in red is trapped on both halves of the oscillation, while the blue particle escapes trapping on the second half oscillation.

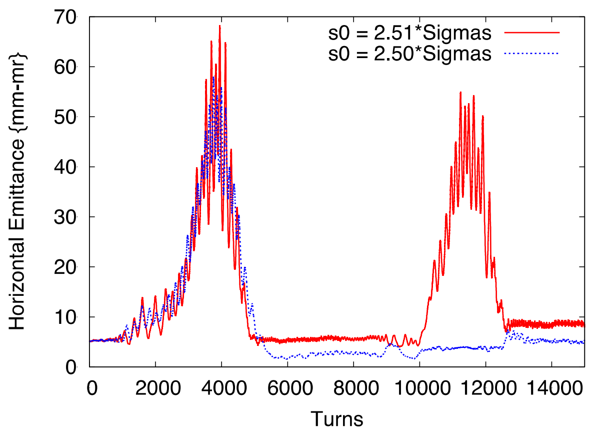

Figure 3: Emittance evolution for two nearly identical particles in resonance trapping benchmark.

CONCLUSION

PY-ORBIT has matured to the point that it can be applied to almost all the problems accessible to ORBIT. The source code is publicly available and easy to use via Python scripts. Users are welcome to develop their own models in PY-ORBIT, either in Python for high-level models or C++ for computationally intensive routines. PY-ORBIT code is carefully benchmarked with ORBIT before being made available to the public.

In considering the choice of space charge model for a given application, the relevant issues involve representation of the beam, numerical fidelity with physics, and computational speed. For short to intermediate time scales occurring in linacs, transfer lines, accumulators, and RCS, FFT-based PIC methods of appropriate dimensionality have been the choice. For storage rings, which typically have long bunch length, 2D and 2.5D models are appropriate. For applying such models on long time scales, it is necessary to account for the effects of numerical collisionality on entropy and emittance growth. These effects were studied in Ref. [13] and a simple empirical scaling law was obtained. This law provides guidance on the necessary number of macroparticles, grid spacing, and therefore computer resources to address a given problem. If, for some applications, the necessary resources are prohibitive, it is worthwhile to consider whether a simpler, fast, analytically based approach might suffice.

Benchmarking is always an essential step in validating new code. PY-ORBIT has been carefully benchmarked with ORBIT during its development, and ORBIT has been extensively benchmarked with analytic, experimental, and other code results. Finally, sensitivity of results must be considered in any benchmarking activity, especially when long time scales are involved. If small differences in input can lead to large differences in output, and if different codes agree on this, then care must be taken on interpreting the meaning of the benchmark results.

Copyright © 2014 CC-BY-3.0 and by the respective authors

ACKNOWLEDGMENT

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

REFERENCES

[1] Pubicly available at Google Codes by typing: svn checkout https://py-orbit.googlecode.com/svn/trunk py-orbit --username youraccount@gmail.com
[2] A. Shishlo, T. Gorlov, J. Holmes, in Proceedings of the International Computatonal Particle Accelerator Conference (ICAP 09), San Francisco, 2009.
[3] J.A. Holmes, S. Cousineau, V.V. Danilov, S. Henderson, A. Shishlo, Y. Sato, W. Chou, L. Michelotti, F. Ostiguy, in The ICFA Beam Dynamics Newsletter, Vol. 30, 2003.
[4] E. Forest, E. McIntosh, F. Schmidt, ``Introduction to the Polymorphic Tracking Code: Fibre Bundles, Polymorphic Taylor Types and "Exact Tracking"", CERN-SL-2002-044 (AP), KEK Report 2002-3.
[5] FFTW Home Page. http://www.fftw.org
[6] H. Grote, F. C. Iselin, CERN/SL/90-13 (AP) (Rev. 5) (1996).
[7] MAD-X Home Page, http://frs.home.cern.ch/frs/Xdoc/mad-X.html
[8] Space Charge 2014, CERN, May 20-21, 2014, https://indico.cern.ch/event/292362
[9] R. W. Hockney and J. W. Eastwood, "Computer Simulation Using Particles", Institute of Physics Publishing, Bristol, 1988.
[10] J. Demmel, NSF-CBMS Short Course on Parallel Numerical Linear Algebra, especially Lectures 24-27, http://www.cs.berkeley.edu/~demmel/cs267-1995/
[11] L. Greengard and V. Rokhlin, "A Fast Algorithm for Particle Simulations", Journal of Computational Physics 73, (1987), 325.
[12] Jurgen Struckmeier, Phys. Rev. E 54, 830, (1996).
[13] O. Boine-Frankenheim, I. Hofmann, J. Struckmeier, S. Appel, Nucl. Instr. and Meth. in Phys.Research A., in press.
[14] G. Franchetti, Code Benchmarking on Space Charge Induced Trapping, http://web-docs.gsi.de/~giuliano/

Copyright © 2014 CC-BY-3.0 and by the respective authors