Code description: pyORBIT

H. Bartosik

with input from members of the space charge working group

ABP-CWG meeting, 2.3.2017

Outline

- Introduction
- Code features & description of the physics modeling
- Details about space charge modules
- Code implementation
- Impact on CERN studies
- Example of application (use case)
- Future plans and needs

Introduction to pyORBIT

- **General description**
  - Macro particle simulation code mainly developed by. S. Cousineau, J. Holmes and A. Shishlo (SNS Oak Ridge) with contributions from colleagues at CERN and GSI
  - Computational beam dynamics models for accelerators, main focus on **space charge effects and related issues** (H- injection with foil, collimation, …) for **single bunch**
  - Modernized version of the ORBIT* code with Python as user interface
  - Open source, available at GitHub https://github.com/PyORBIT-Collaboration

- **Licensing policy**
  - MIT License (open source)

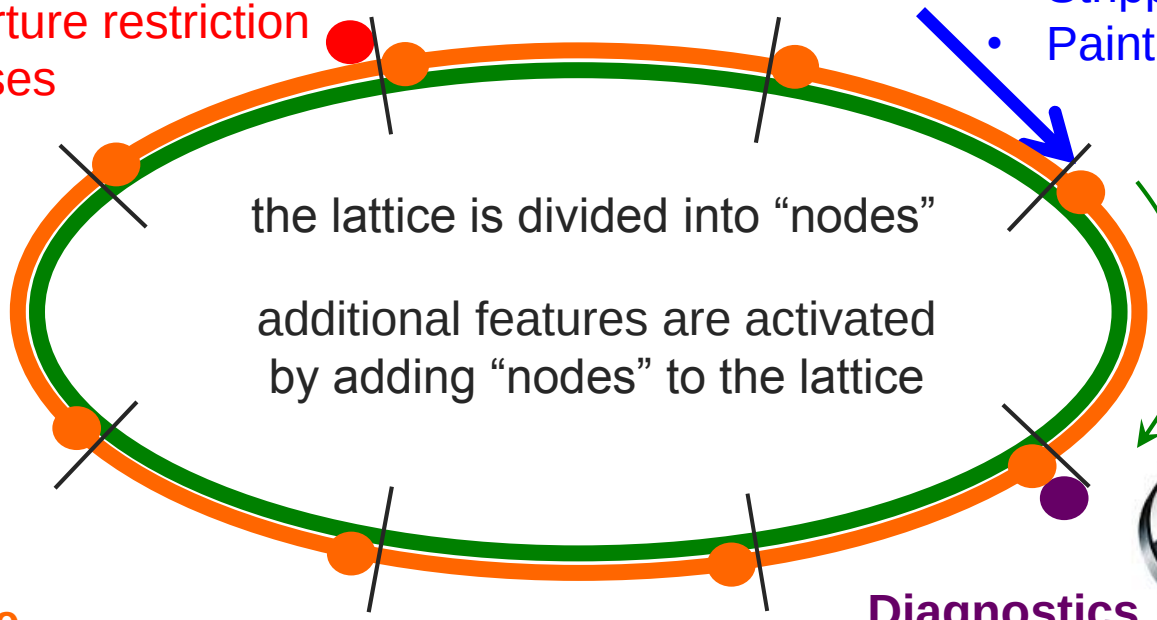
- **Documentation**
  - Mainly provided in form of code examples demonstrating/testing individual modules
  - Some documentation available on WIKI page
  - PTC interface described in notes available at space charge working group website

* ORBIT was developed at Oak Ridge National Laboratory (ORNL) for designing the Spallation Neutron Source (SNS) accelerator complex

pyORBIT – overview of relevant features

Collimation
- interaction of particles with matter
- Aperture restriction
- Losses

Injection
- Injection bump collapse
- Stripper foil scattering
- Painting



Tracking
- Teapot
- PTC
- Matrizes

Space charge
- Direct space charge (analytical or FFT PIC)
- Indirect space charge (FFT PIC with perfect conducting wall)
- Longitudinal space charge

Diagnostics
- Emittance
- Bunch twiss parameters
- Particle phase advances
- …

Tracking

- **Teapot lattice**
  - The symplectic Teapot tracker from ORBIT is included in pyORBIT
  - Each machine element is represented as a “lattice node”
  - Contains all standard elements (standard magnets, RF cavities, kickers, …)
  - Allows for acceleration and time varying fields, alignment errors
  - Lattice definition can be directly imported from MAD8 and MADX (“MAD parsers”)
  - Space charge, diagnostic and collimation nodes can be inserted as additional nodes
- **PTC**
  - An interface to PTC is available (i.e. PTC can be used for tracking)
  - PTC lattices can be called in the form of a “PTC flat file” (generated directly with MADX taking into account **multipole and alignment errors**)
  - Time varying fields, acceleration and double harmonic RF systems through “PTC tables”
  - Space charge, diagnostic and collimation nodes can be inserted in form of “child nodes”
- **Linear tracking with matrices**

Analytical (frozen) space charge solver

- **Transverse force due to space charge is derived from the closed expression for electric field of a two-dimensional Gaussian (Basetti Erskine* formula)**
  - Transverse bunch dimensions obtained from local beta-functions, emittances, dispersion and momentum spread (optics functions from PTC)
  - Space charge force is weighted for each particle with the line density at its longitudinal position: from analytical line density profile (Gaussian, parabolic, constant, …) or user defined (array input)
  - Beam parameters (intensity, transverse emittance, momentum spread, line density) and lattice parameters (TWISS parameters at location of space charge nodes) can be updated by the user whenever needed (otherwise truly frozen)
  - Not self consistent! (always transverse Gaussian)
  - No coherent motion!
- **Interest**
  - "Noise free" → allows using small number of macro particles to represent the beam
  - Studies of single particle dynamics in presence of space charge (e.g. Frequency maps)

* M. Bassetti and G.A. Erskine, Closed expression for the electrical field of a two-dimensional Gaussian charge, CERN-ISR-TH/80-06

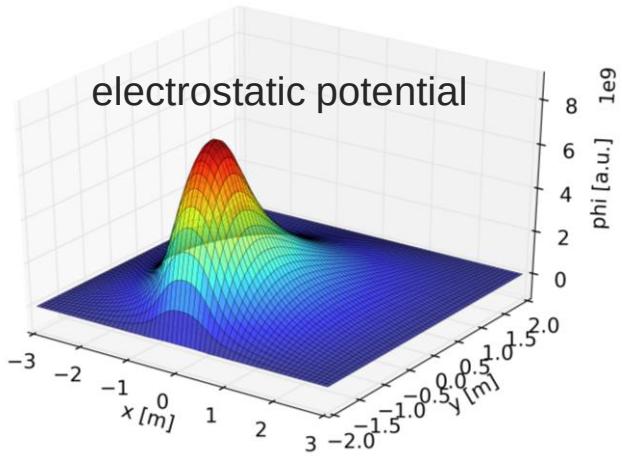
2D FFT PIC solver

- **The 2D FFT Particle in Cell Poisson solver is a standalone building block used for several space charge modules in pyORBIT**
  - Solves Poisson's equation in beam frame ($1/\gamma^2$ factor)

$$\Delta\Phi = -\frac{\rho}{\varepsilon_0\gamma^2}$$

  - *Convolution (FFT) Method** (charge density and Green function are Fourier transformed to frequency domain and multiplied → potential is obtained by inverse transform)

- **Perfect conducting boundary can be added (indirect space charge)**
  - Circular, elliptical, rectangular or arbitrary geometry defined by the user
  - Potential on the grid is modified by adding free space potential such that the sum of the two potentials becomes zero on the boundary in a least squares sense **

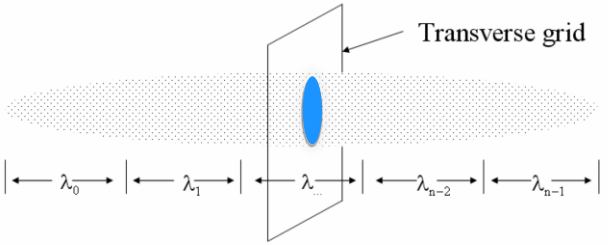


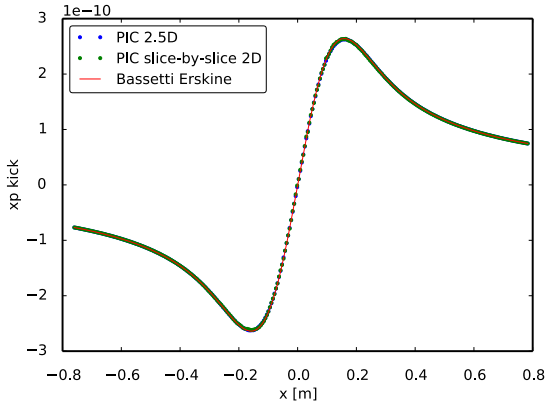
* R.W. Hockney and J.W. Eastwood, "Computer Simulation Using Particles", IOP Publishing Ltd, Adam Hilger,p.211,(1988)

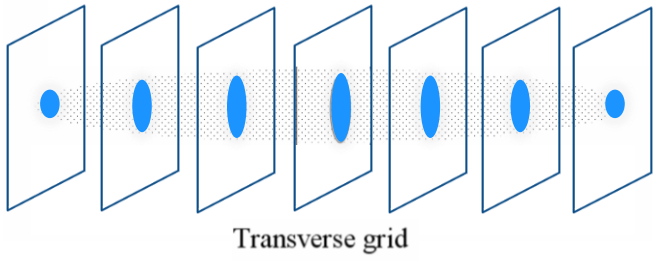
** F. W. Jones, A Method for incorporating image forces in multiparticle tracking with space charge, EPAC2000, (Vienna)

PIC models (the most commonly used)

- **“2.5D” space charge solver**
  - Transverse distribution is binned onto **single** 2D grid
  - Potential is obtained with 2D FFT Poisson solver
  - Transverse kicks are weighted by local line density λ obtained from longitudinal 1D grid
  - Longitudinal space charge can be included (line density derivative, transverse uniform beam)
  - Perfect conducting boundary conditions can be used
- **“Slice-by-slice 2D” space charge solver**
  - Particles are binned to a 3D grid
  - Potential is solved on each 2D grid slices
  - Grid slices “feel” the nearest neighbors only
  - Transverse and longitudinal kicks from gradient of potential on 3D grid
  - Perfect conducting boundary conditions can be used

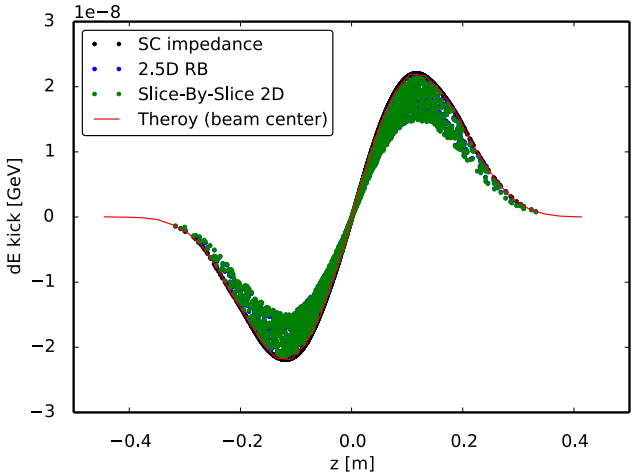






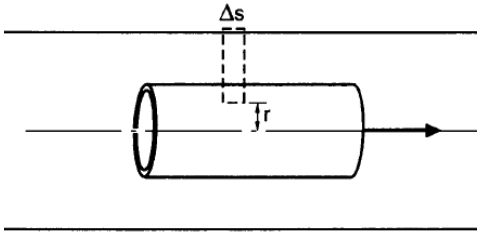
Longitudinal space charge

- **Longitudinal space charge impedance**
  - From line density derivative evaluated once per turn
  - Transverse uniform round beam in circular pipe (A. Chao)
  - Evaluated on axis → no transverse dependence
- **Longitudinal kicks from Slice-By-Slice 2D module**
  - Derivative of potential between 2D slices
  - Arbitrary boundary conditions can be used



longitudinally Gaussian bunch with transverse KV distribution in circular pipe

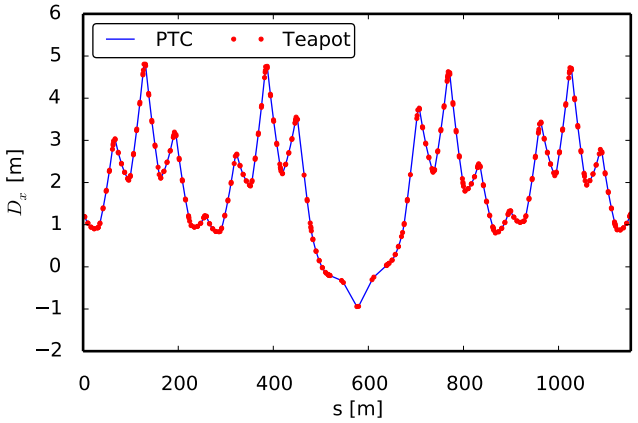
- **Longitudinal kicks from 2.5D module (special version)**
  - Calculated from line density derivative assuming transverse uniform beam
  - Transverse dependence: for each particle integration from actual radial position r to beam pipe a after calculating beam size b of macroparticle distribution (R. Baartman)

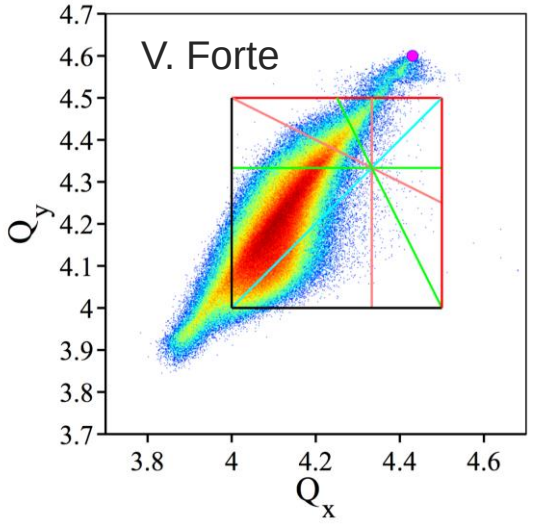


$$E_s(r, s-\beta ct) = -\frac{1}{2\pi\varepsilon_0\gamma^2}\lambda'(s-\beta ct)\begin{cases}\log\frac{b}{a}+\frac{1}{2}-\frac{r^2}{2a^2} & \text{if } 0<r<a\\ \log\frac{b}{r} & \text{if } a<r<b\end{cases}$$

Diagnostic modules

- **Bunch TWISS analysis**
  - Calculates statistical moments and correlations
  - Provides statistical beam TWISS parameters, dispersion, beam emittance (with or without dispersive part, …)
- **Bunch tune analysis**
  - Transforms the transverse phase space coordinates into normalized phase space at a given location in the lattice
  - Calculates the phase advance per turn for each particle (and actually not the tune!)
- **Losses**
  - Information on lost particles including loss location
  - Aperture limitations can be installed on pyORBIT level, but also apertures from PTC are taken into account





Code implementation

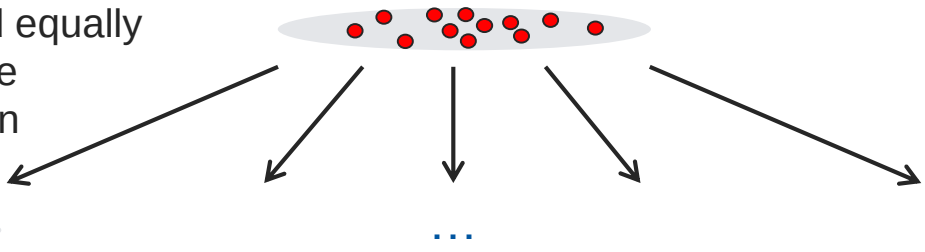
- **Programming language(s)**
  - User interface via Python (simulation runs are actually Python scripts)
  - Computationally intensive code of pyORBIT is written in C++, some functions in Python
  - PTC is written in Fortran
  - Wrapper functions make C++ routines and PTC tracking available at Python level
  - **Parallelization is based on MPI** (more details on next slide)

- **Prerequisites to run the code**
  - Python 2.6.6, C++, Fortran (if using PTC)
  - Can be compiled on Linux, MAC OSX and Windows
  - The source code and the compiled executable are available on AFS: /afs/cern.ch/project/LIUsc/space_charge/Codes/py-orbit

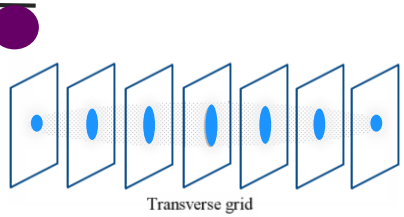
- **Where is the code run?**
  - LXPLUS (mostly for quick checks and debugging)
  - Space charge cluster (presently only 20 boxes with 16 Cores each, networked for MPI)
  - Was tested on Bologna HPC cluster
  - HPC cluster at CERN?

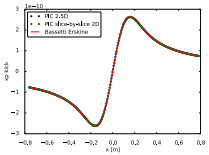
Parallelization strategy

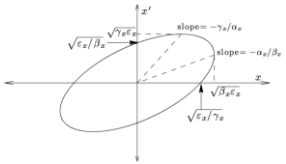
The particles are distributed equally between all CPUs during the initialization of the simulation

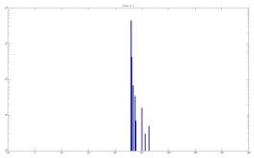


s position along machine circumference









example for one lattice node

Space charge node with PIC solver:

- Each CPU bins its particles to its own grid
- *Grids are synchronized with MPI*
- Each CPU solves electric potential for a subset of the 2D grids
- *Solution is propagated to other CPUs with MPI*
- Each CPU applies kicks for its own particles

Diagnostics node:

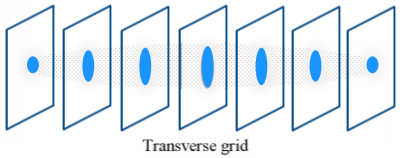
- *In some cases requires MPI communication* (e.g. emittance calculation)

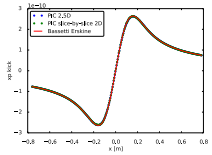
Tracking through the lattice node:

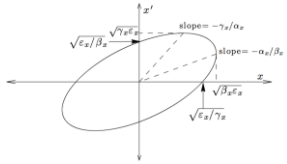
- Each CPU tracks its own particles

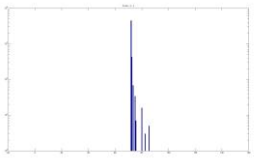
Collimation node:

- Each CPU removes its lost particles from bunch









Impact on CERN studies

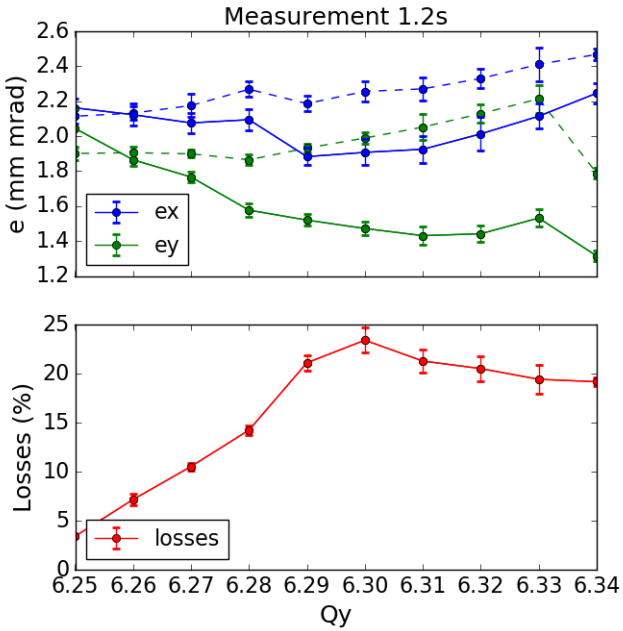
- **Understanding and modeling of space charge in LHC injectors is essential for LIU**
  - Tight requirements on emittance growth and losses

- **PSB: optimizing beam brightness (fast beam behavior at low energy)**
  - Prediction of PSB brightness curve with Linac4 with optimized working point
  - H^- charge exchange injection with painting and capture in double RF on the ramp
  - Impact of falling chicane bump including multipoles generated by eddy currents, beta-beat compensation, effect of resonances on final emittance, emittance blow-up from stripper foil with lower current from Linac4, …

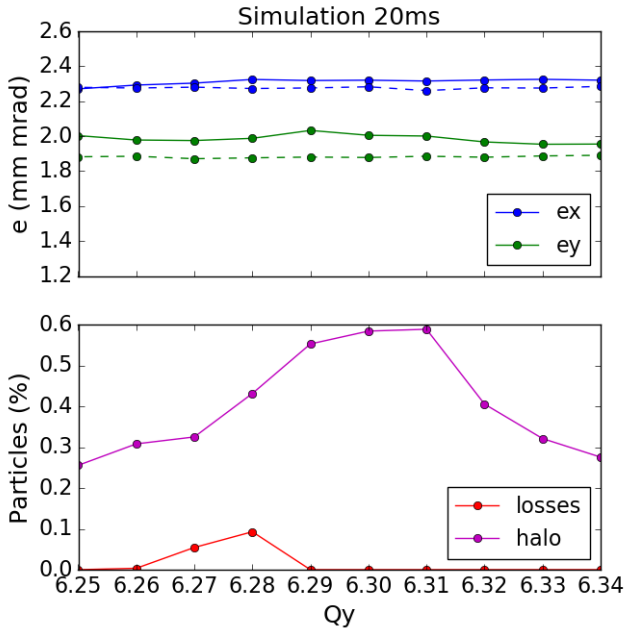
- **PS & SPS: emittance and loss evolution on injection plateau (fast & long-term)**
  - Understand performance limitation due to machine and space charge resonances
  - Quantitative simulations of losses and blow-up as function of working point for optimization of working point and machine settings
  - Beam halo and tail creation
  - Fast blow-up at injection (PS in particular)
  - Prediction of achievable brightness at 2 GeV PS injection energy

Example usage: PS (the most challenging …)

- **Studies of the 8th order structural space charge driven resonance at Qy=6.25**
  - Measurements of emittance and losses after 1.2 s storage time as a function of Qy
- **PIC model for short term behavior (runs for different tunes, …)**
  - 1 run: 1700 nodes, grid 64x64x20, 200k macro particles, **13 days for 10k turns (16 CPUs)**



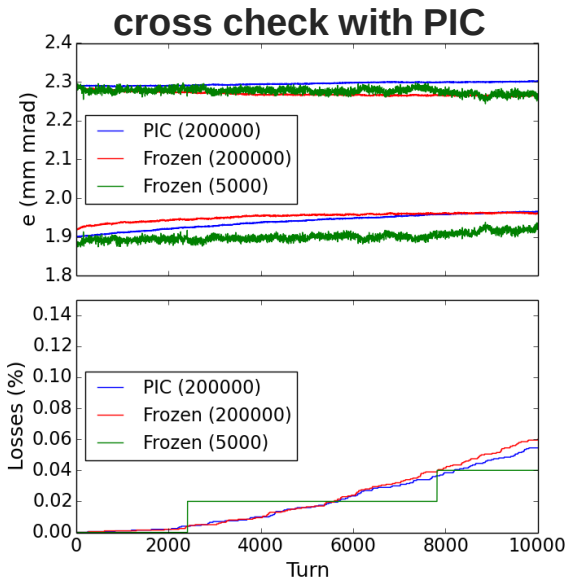
F. Asvesta, M. Serluca et al.

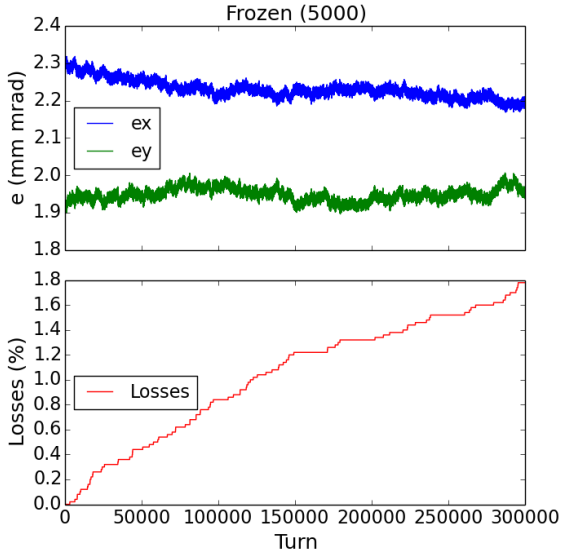


F. Asvesta

Example usage: PS (the most challenging …)

- **Studies of the 8th order structural space charge driven resonance at Qy=6.25**
  - Measurements of emittance and losses after 1.2 s storage time as a function of Qy
- **PIC model for short term behavior (runs for different tunes, …)**
  - 1 run: 1700 nodes, grid 64x64x20, 200k macro particles, **13 days for 10k turns (16 CPUs)**
- **Analytical ("frozen") model for long term behavior (runs for different tunes, …)**
  - 1 run: 1700 nodes, 5k macro particles, **14 days for 300k turns (16 CPUs)**





300k turns
(0.6s in PS)

F. Asvesta

Future plans and needs

- **Maintenance, extension and further development**
  - pyORBIT is actively maintained and further developed by colleagues from SNS
  - Minor development at CERN for special features needed for LIU studies (mainly by fellows and PhD students)
- **Performance improvement?**
  - No active work in this direction on pyORBIT code side
- **Is the performance in general adequate to the present needs?**
  - Performance could be faster … substaintal part of computation time is spent on PTC tracking
- **What type of hardware resources would be best suited for the physics case?**
  - Multi-core boxes properly networked for MPI
  - Presently using CERN “space charge cluster”: 20 boxes with 16 cores each ☹
  - Medium-term plan is to use Bologna HPC cluster
  - Long-term hope is to get HPC cluster at CERN

… presently computing resources not adequate

www.cern.ch