CERN

Update on Status and Recent Developments in PTC-PyORBIT

Hannes Bartosik, Haroon Rafique (BE-ABP-HSI)

haroon.rafique@cern.ch

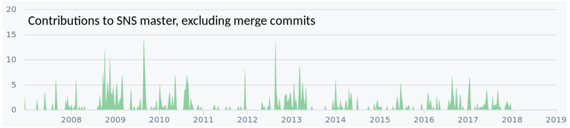
19.03.19

Table of Contents

PTC-PyORBIT

- **What is it?**: PTC (Polymorphic Tracking Code) is a well established and relied upon particle tracker. ORBIT (Objective Ring Beam Injection and Tracking) is also capable of tracking, but provides the framework for injection, space charge, e-cloud etc. PyORBIT is Python-wrapped ORBIT, adding MPI, PIC space charge, and more (see reference in acknowledgments).
- **What can it do?**: For CERN machines it is well suited to space charge studies for the PSB, PS, and SPS.
- **Why do we use it?**: PTC tracking is reliable, python scripting is easy to use, space charge models are sufficient, CERN does not have it's own space charge code at the moment. Can use MAD-X lattice errors and port to PTC.

History



- ‘CERN’ PyORBIT forked from SNS master in 2017.
- Numerous ’branches’: frozen (analytical) space charge, acceleration, development.
- Now merged into one (CERN) master branch. Aim to merge back into SNS master.

git speak: A "fork" is a copy of a repository. A "branch" is an active line of development. "Merge" refers to bringing together two branches, where one branch can be another repository.

PyORBIT Git Activity

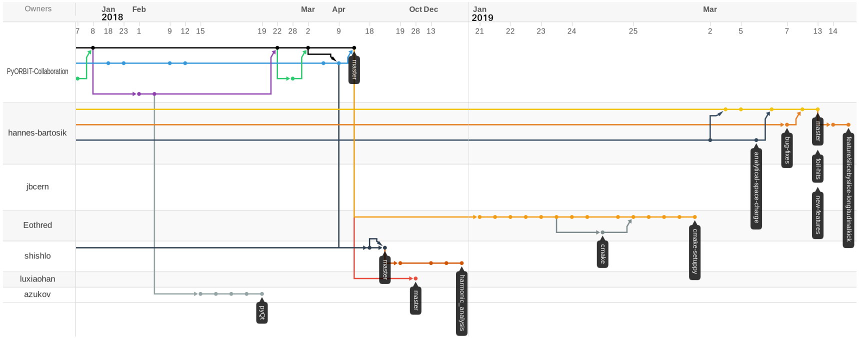


Figure: Recent network for PyORBIT forks.

PyORBIT user

/afs/cern.ch/user/p/pyorbit/public contains:

- Virtual environment for PyORBIT.
- Soft link to most recent PyORBIT build.
- Alternate GFortran PyORBIT build.
- Documentation.
- Examples repository.

All information is centralised here. User installation of PTC/PyORBIT no longer necessary if you have access to AFS (and CVMFS).

New Structure

/afs/cern.ch/user/p/pyorbit/public/PyOrbit_env

PyORBIT virtual environment - Python v 2.7 which currently includes numpy, scipy, ipython, matplotlib, imageio, h5py. **Please ask us if you require something more.**

/afs/cern.ch/user/p/pyorbit/public/PyOrbit_env/py-orbit

Soft link to latest version so the user never has to change their PyORBIT directory. Versions currently available: py-orbit_20190303/ (latest) py-orbit_frozenSC_20170731/ ...

GFortran vs iFort

GFortran compiled PTC PyORBIT available at:

/afs/cern.ch/user/p/pyorbit/public/PyOrbit_env_gcc_gfortran

On HPC-Batch this version appears to be the same as iFort, however on HTCondor the iFort version (previous slide) is faster.

How to access the ‘latest’ version

We will maintain all versions from hereon in for completeness and back-compatibility. The soft link will always point to the latest version.

Soft link to latest version:

/afs/cern.ch/user/p/pyorbit/public/PyOrbit_env/py-orbit

Installation scripts available (designed for lxplus/afs).

First install environment: 01_install_PyOrbit_env.sh
Next install required version:
02a_install_PyORBIT_20170731_frozenSC.sh
02b_install_PyORBIT_20190307.sh
02c_install_PyORBIT_20190313.sh

Running PyORBIT

The user DOES NOT have to install PyORBIT, simply use the following scripts to use the maintained PyORBIT versions accessible via AFS. The installation scripts are provided should you wish to install on another system.
PyORBIT can be run using mpirun. The following script is provided in the examples repository to do this for you. Please note all scripts are made for the Bash shell. Items worth noting:

- The method of calling the setup_environment.sh script `. setup_environment.sh` not `./setup_environment.sh`.
- The environment variable `FI_PROVIDER=sockets` must be set on lxplus due to the upgrade to CC7.
- This script is run as: `$> ./START_local.sh pyOrbit.py 4` where pyOrbit.py is your simulation script, and 4 indicates the number of MPI processes.

Running PyORBIT: `START_local.sh`

```bash
#!/bin/bash
set -o errexit
if [ ! -n "$1" ]
  then
    echo "Usage: `basename $0` <name of the SC script> <N CPUs>"
    exit $E_BADARGS
fi
if [ ! -n "$2" ]
  then
    echo "Usage: `basename $0` <name of the SC script> <N CPUs>"
    exit $E_BADARGS
fi

export FI_PROVIDER=sockets
. setup_environment.sh

mpirun -np $2 ${ORBIT_ROOT}/bin/pyORBIT $1
```

Environment Setup: `environment_setup.sh`

The simulation_setup.sh script (provided in examples repository) activates all requirements and sets up environment variables:

```
#!/bin/bash
pyOrbit_dir=/afs/cern.ch/user/p/pyorbit/public/PyOrbit_env/py-orbit

source ${pyOrbit_dir}/customEnvironment.sh
echo "customEnvironment done"
source ${pyOrbit_dir}/../virtualenvs/py2.7/bin/activate
echo "python packages charged"
source ${pyOrbit_dir}/../setup_ifort.sh
echo "ifort charged (necessary for running)"

ORBIT_ROOT_fullpath=`readlink -f ${ORBIT_ROOT}`
. ${ORBIT_ROOT}/../CheckGitStatus.sh ${ORBIT_ROOT_fullpath}
```

Environment Setup

Where will this work?

- LXPlus
- HTCondor
- HPC-Batch (SLURM)

Documentation

/afs/cern.ch/user/p/pyorbit/public/pyorbit_examples_repository/PyOrbit_documentation

- Code_Description_H_Bartosik_ABP-CWG: Code description talk.
- Notes_on_PTCOrbit_M_Fitterer: PTC-ORBIT notes (NB. not PTC-PyORBIT).
- PTCORBIT_pyORBIT_benchmark_JB_Lagrange.pdf: Self-explanatory.
- PTC-ORBIT-A_Molodozhentsev.pdf: PTC based notes for PTC-ORBIT (NB. not PTC-PyORBIT).
- Python_Shell_for_the_ORBIT_Code_A_Shishlo.pdf: ICAP paper on PyORBIT.
- CERN_PTC-PyORBIT_Update: This talk.

Collection of talks, papers, manuals that could be useful.

Examples Repository

GitHub examples repository is available - should work out of the box on AFS.

Examples repository:

Pull/fork from gitlab: https://gitlab.cern.ch/pyorbit/pyorbit_examples
Copy from pyorbit user: /afs/cern.ch/user/p/pyorbit/public/pyorbit_examples_repository/

We endeavour to create examples for all CERN machines.
The following slides are given as a guideline for new users.

Create a Lattice

- **Step 1:** Have a MAD-X description of the lattice in a suitable format.
- **Step 2:** Create a PTC flat file using PTC inside MAD-X.
- **Step 3:** Read the flat file in PyORBIT to create a lattice.
- **Step 4:** Check optics etc.

Create a Lattice

- **Step 1:** Have a MAD-X description of the lattice in a suitable format.

Suitable Format:[1]

- "No elements should be split if they are not split in the machine. Splitting magnets increases the number of integration steps without increasing the order of the symplectic integrator and with it the precision."
- "All markers should be removed. These slow down the simulation as they are unnecessary nodes."
- "Cavities should be defined using `no_cavity_totalpath`."

[1] M. Fitterer, Notes on PTC-ORBIT, 2012

Create a Lattice

- **Step 2:** Create a PTC flat file using PTC inside MAD-X.

```
ptc_create_universe;
ptc_create_layout,time=true, model=2, exact=true, method=6, nst=3;
ptc_script, file="./PS/resplit.ptc";
ptc_script, file="./print_flat_file.ptc";
select, flag=ptc_twiss, clear;
select, flag=ptc_twiss, column=name, betx, px, bety, py, disp3, disp3p,
 ↪   disp1, disp1p;
ptc_twiss, icase=5, no=4, deltap_dependency, closed_orbit, file,
 ↪   table=ptc_twiss;
ptc_end;
```

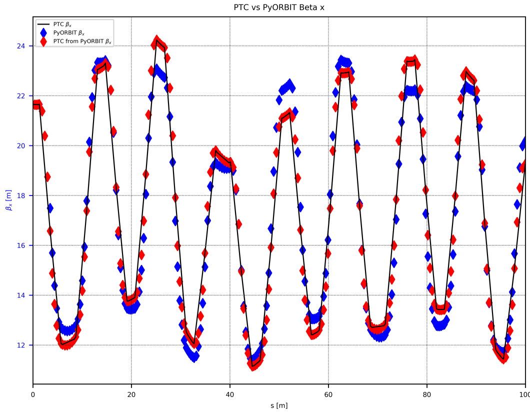
Create a Lattice

- **Step 3:** Read the flat file in PyORBIT to create a lattice.

```
PTC_File = 'PTC-PyORBIT_flat_file.flt'
Lattice = PTC_Lattice("PS")
Lattice.readPTC(PTC_File)
readScriptPTC('PTC/fringe.ptc')
readScriptPTC('PTC/time.ptc')
readScriptPTC('PTC/ramp_cavities.ptc')
```

Create a Lattice

- **Step 4:** Check optics etc.

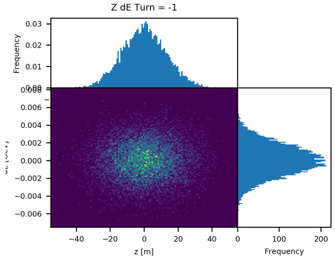


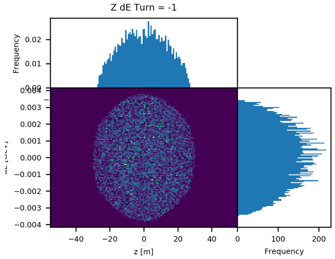
Create a Bunch

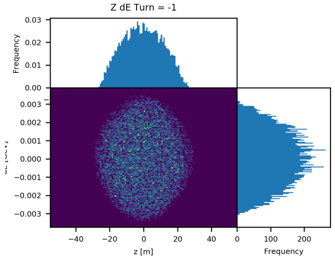
- The user may specify their own distribution, or use the (CERN made) `pyOrbit_GenerateInitialDistribution.py` PyORBIT library. This gives the options for: matched, Poincaré, 3D Gaussian, tomo, and other distributions.
- The user must specify transverse emittances, bunch length, momentum spread ($\frac{\Delta p}{p}$). Unless using tomo method - in which case transverse emittances only.
- It is recommended to verify the longitudinal motion of the bunch before running large simulations.

Create a Bunch

3D Gaussian, Matched, Tomo



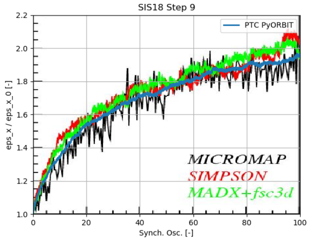




Space charge

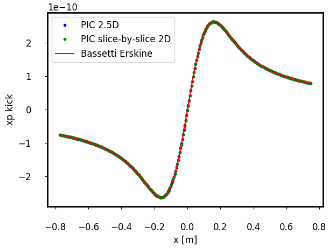
Various examples available:

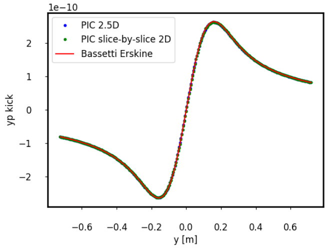
SIS18 Benchmark.
Tests in examples repository.

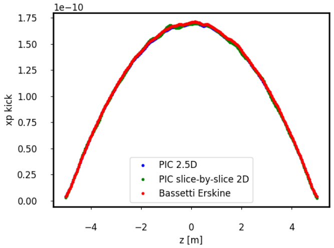


Space Charge

Space charge kick comparison for Gaussian distribution.

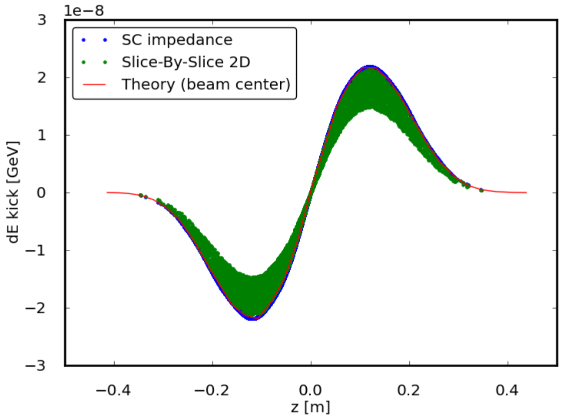






Space Charge

Longitudinal space charge kick for $2 \cdot 10^6$ protons in a KV distribution.



CERN Libraries

CERN authored PyORBIT scripts exist to make simulations simpler:

- Write PTC ramp tables.
- MPI helpers.
- Suppress PTC output.
- Generate initial distribution.
- Save (restore) bunch as (from) mat file.
- Calculate tunespread.[2]
- Print lattice functions from PTC within PyORBIT (useful if these change on a turn-by-turn basis).
- Output dictionary (dumps user specified parameters every turn into a single output file).
- Particle output dictionary (dumps data for specific particles - useful for Poincaré sections).

[2] Uses A. Oeftiger's spacecharge_tunespread scripts

Common Pitfalls

- The user must include a call to the PTC script ‘time.ptc’ in their PyORBIT code to ensure that PTC use the correct units.
- For ramping of magnets or accelerator cavities a PTC table must be constructed.
- When using pickle (for HTCondor - this allows resuming simulations that have failed mid-way) it is important to remove the pickle file before starting a fresh simulation.

Distribution from Tomo

Note that the the examples repository contains everything required to do the following.

- **Step 1:** Change tomo dat file directory to ./
- **Step 2:** Run tomo executable (via python script).
- **Step 3:** Analyse data and reformat (via python script). Produces image that should be manually checked to remove outliers and centre the bunch.
- **Step 4:** Produce output in PyORBIT expected .mat file (via python script).
- **Step 5:** Read .mat file in simulation using function `generate_initial_distribution_from_tomo`

Distribution from Tomo

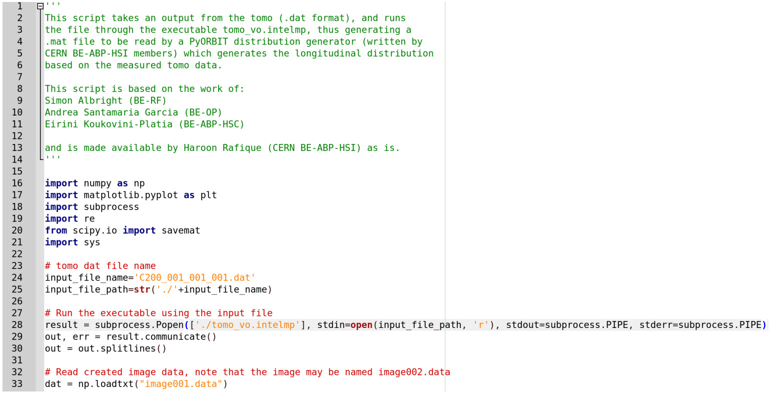
- **Step 1:** Change tomo dat file output directory to './'.

```
MD1=MD4224_48b_BCMS
1536135630
170
1706.43
1
1268.0000000000007
0.5
-4.007615230460919
! Version 2
!
! Data Format:
! Input data file =
pipe
! Directory in which to write all output =
/tmp/
! Number of frames in input data =
150
! Number of frames to ignore =
0
! Number of bins in each frame =
1000
```

```
MD1=MD4224_48b_BCMS
1536135630
170
1706.43
1
1268.0000000000007
0.5
-4.007615230460919
! Version 2
!
! Data Format:
! Input data file =
pipe
! Directory in which to write all output =
./
! Number of frames in input data =
150
! Number of frames to ignore =
0
! Number of bins in each frame =
1000
```

Distribution from Tomo

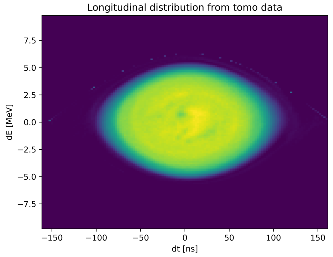
- **Step 2:** Run tomo executable (via python script).

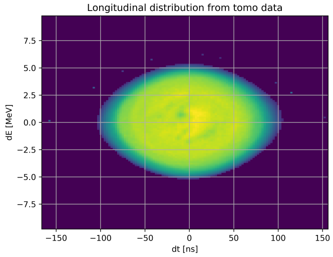


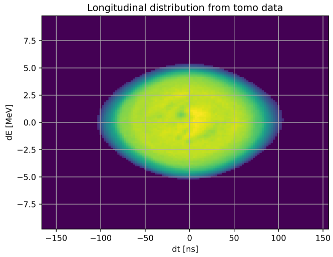
```python
'''
This script takes an output from the tomo (.dat format), and runs
the file through the executable tomo_vo.intelmp, thus generating a
.mat file to be read by a PyORBIT distribution generator (written by
CERN BE-ABP-HSI members) which generates the longitudinal distribution
based on the measured tomo data.

This script is based on the work of:
Simon Albright (BE-RF)
Andrea Santamaria Garcia (BE-OP)
Eirini Koukovini-Platia (BE-ABP-HSC)

and is made available by Haroon Rafique (CERN BE-ABP-HSI) as is.
'''

import numpy as np
import matplotlib.pyplot as plt
import subprocess
import re
from scipy.io import savemat
import sys

# tomo dat file name
input_file_name='C200_001_001_001.dat'
input_file_path=str('./'+input_file_name)

# Run the executable using the input file
result = subprocess.Popen(['./tomo_vo.intelmp'], stdin=open(input_file_path, 'r'), stdout=subprocess.PIPE, stderr=subprocess.PIPE)
out, err = result.communicate()
out = out.splitlines()

# Read created image data, note that the image may be named image002.data
dat = np.loadtxt("image001.data")
```

Distribution from Tomo

- **Step 3:** Analyse data and reformat (via python script). Produces image that should be manually checked to remove outliers and centre the bunch.

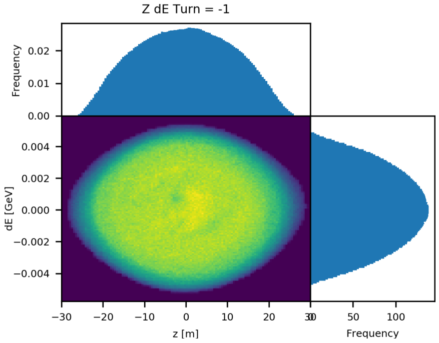






Distribution from Tomo

- **Step 4:** Produce output in PyORBIT expected .mat file (via python script).
- **Step 5:** Read .mat file in simulation using function `generate_initial_distribution_from_tomo`



HTCondor

- The user does not have to use the ‘getenv’ flag, instead sourcing ‘setup_environment.sh’ method will work.
- Example submission scripts are provided in the examples repository.
  /afs/cern.ch/work/p/pyorbit/public/pyorbit_examples/Job_Submission/HTCondor
  https://gitlab.cern.ch/pyorbit/pyorbit_examples/tree/master/Job_Submission/HTCondor

HPC-Batch (SLURM)

The user should now be able to use PyORBIT on AFS from HPC-Batch using the same setup_environment.sh script.
Important things to remember:

- As PyORBIT has it's own MPI version in the virtual environment, the user must specify the SLURM MPI version using 'module load mpi/mvapich2/2.2' in the SLURM submission script **after** initialising the virtual environment.
- If the user simulation is stored on \bescratch then the be-short or be-long queues must be used as batch-long or batch-short nodes do not have access to \bescratch.
- Example submission scripts are provided in the examples repository.
  /afs/cern.ch/work/p/pyorbit/public/pyorbit_examples/Job_Submission/HPC-Batch(SLURM)
  https://gitlab.cern.ch/pyorbit/pyorbit_examples/tree/master/Job_Submission/HPC-Batch(SLURM)

Summary

- The CERN fork of PTC-PyORBIT has been updated to merge both the analytical space charge branch with the acceleration branch.
- A new user has been created with a new home for PTC-PyORBIT on AFS.
- Inside this users `public/`, there exists a soft link that always points to the current working version of PTC-PyORBIT (CERN fork).
- There are a myriad of examples and documentation here.
- A new user simply has to clone the PyORBIT_Examples repository to their user space (AFS accessible) and they can run the examples as is. These examples should provide a quick and easy method of using PyORBIT.
- Job submission scripts and instructions are provided for HTCondor and HPC-Batch.

Conclusion 1

What did we do?

Unified the CERN effort on PTC-PyORBIT. Codes were merged, a clean new home has been provided for the code at CERN, scripts have been written to make it easy to use, and structured examples have been developed (more ongoing) and provided.

Why did we do it?

PTC-PyORBIT is our current workhorse for space charge simulations. We perform a great deal of space charge simulations in order to advance our understanding of how the beam behaves in our accelerators.

Conclusion 2

Why should you care?

New users (possibly you):

- No longer have to perform a literature review to understand the many parts of the code. Documentation and thorough examples are provided.
- Don't have to worry about setting up their environment, finding or installing the latest version, or writing basic space charge simulations.
- Should be running their own simulations in minutes/hours rather than days/weeks.
- Can use HTCondor and HPC-Batch (SLURM) without worrying about the virtual environment.

Acknowledgements

This work was built upon the efforts of many:

- **ORBIT** originally developed by J. Galambos, S. Danilov, D. Jeon, J. Holmes, D. Olsen @ SNS, and A. Luccio, J. Beebee-Wang @ BNL (SNS/ORNL/AP Technical Note Number 011 Rev. 1, July 16, 1999, Proceedings of IPAC99, New York, NY, THP82, 1999.
- **PyORBIT** developed by A. Shishlo, J. Holmes, T. Gorlov @ ORNL Proceedings of ICAP09, San Francisco, CA, THPSC052, 2009.
- **PTC** originally developed by Etienne Forest (KEK) *CERN-SL-2002-044-AP*, with updates from F. Schmidt (BE-ABP-HSI), P. Skowronski (BE-ABP-HSS), A. Molodozhentsev (KEK).

Acknowledgements

- **Help with compilation, installation, and environment setup**: F. Schmidt (BE-ABP-HSI).
- **Distribution from tomo**: S. Albright (BE-RF), E. Koukovini-Platia (BE-ABP-HSC), A. Santamaria Garcia (BE-OP).
- **Development & PS Simulations**: F. Asvesta (BE-ABP-HSI), A. Huschauer (BE-ABP-HSS).
- **Frozen space charge**: H. Bartosik (BE-ABP-HSI).
- **PTC-ORBIT to PTC-PyORBIT development and benchmarking**: JB. Lagrange (BE-ABP-HSI).
- **SIS18 Benchmark**: H. Rafique (BE-ABP-HSI).
- **PSB simulations**: E. Koukovini-Platia (BE-ABP-HSC), F. Antoniou (BE-ABP-HSI), T. Prebibaj (BE-ABP-HSI).
- **Improving PTC for PyORBIT** Piotr Skowronski (BE-ABP-HSS).