# EVALUATING PYORBIT AS UNIFIED SIMULATION TOOL FOR BEAM-DYNAMICS MODELING OF THE ESS LINAC

C. Zlatanov, J. F. Esteban Müller, Y. Levinsen, N. Milas
European Spallation Source ERIC, Lund, Sweden

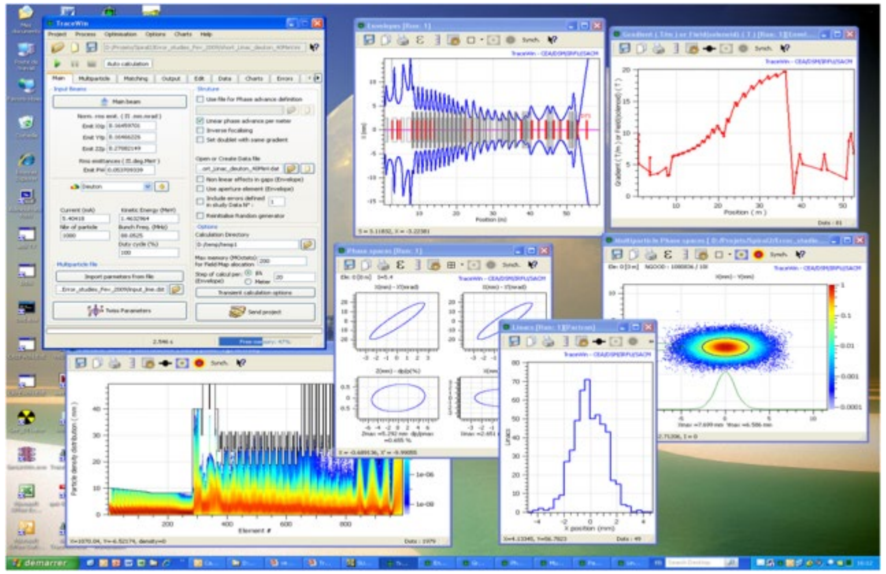
A. Zhukov, A. P. Shishlo
ORNL, Oak Ridge, Tennessee, USA

2023-10-10 HB2023

# Current status of beam-dynamics simulations at ESS

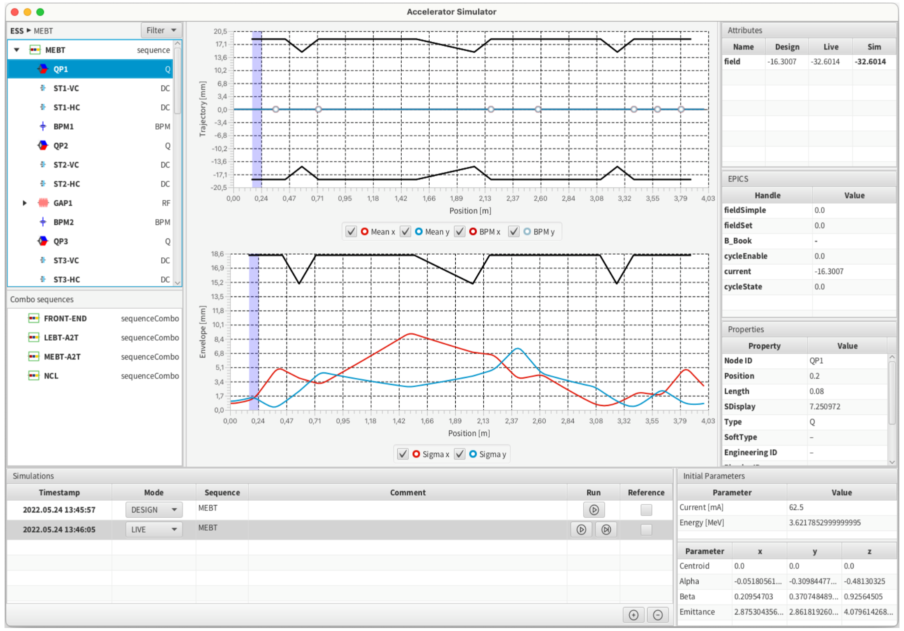
**TraceWin**: for precise offline simulations

- Envelope model
- Multiparticle tracker
- Feature-rich
- Graphical user interface



**Open XAL**: for fast simulations and control room beam-physics applications

- Online envelope model
- EPICS integration
- Java and JavaFX for user interfaces



# Shortcomings of this approach

- We need to maintain 2 sets of lattice files
- TraceWin can't easily be extended and is closed source
- Users of the simulation framework would prefer a Python API to exploit its advanced data analysis and visualization libraries
- We want to be able to simulate synchrotrons for other projects

ESS neutrino Super Beam plus

International MUON Collider Collaboration

→ We have been assessing the option to use one code for all beam-dynamics simulations

# Alternative solutions

- Develop a new framework from scratch

  → Estimated effort ~2-3 person-year

- Extend Open XAL with a multiparticle tracker and good Python bindings

  → Java performance limitations

- Find an existing code that fulfills at least partially our requirements and that can be extended

  → Simplest solution if we find a good match

# PyORBIT

Open-source code written in Python and C++

Developed for SNS at ORNL

Features:

- Multiparticle tracker for linacs and synchrotrons
- 3 different algorithms for space charge
- Numerical integrator for tracking particles under arbitrary electromagnetic fields
- Routines for beam-coupling impedance and beam-matter interaction
- MPI integration for parallel computing

# Porting to Python 3

Existing version:

- Compatible only with Python 2.7 (or older), deprecated in 2020
- Complicated build system: users need to compile a Python interpreter that loads PyORBIT library as builtin modules

We released a new version on https://github.com/PyORBIT-Collaboration/pyorbit3

- Ported PyORBIT to Python 3
- Improved build mechanism: can generate a pip wheel for simpler deployment

# Lattice description (files)

PyORBIT supports MAD-X, SAD, and a custom XML format

We are exploring the option to generate our lattice in Python

| Pros | Cons |
| --- | --- |
| ▪ No need to learn a new syntax<br>▪ Can programmatically describe machine sections, auto-generate signal names for integration with the control system, etc.<br>▪ Takes advantage of IDE features such as auto-completion, syntax highlighting, error detection, and display documentation<br>▪ Deployment via pip | ▪ Every time lattice is modified may require reinstalling the pip package<br>▪ Switching between different lattices at runtime can become more cumbersome |

# Lattice description in Python

## Example

```python
from typing import List
from orbit.lattice import AccNode
from orbit.py_linac.lattice import LinacAccLattice, Sequence
import lattice_builder as builder

# Instantiating a Lattice element
lattice = LinacAccLattice("ESS Lattice")
frequency = 352.21e6
maxDriftLength = 0.005

accSeqs: List[Sequence] = []
thinNodes: List[AccNode] = []

# Adding the DTL sequence
dtl_seq = builder.addSequence(lattice=lattice, accSeqs=accSeqs, name="DTL", length=33.1528945,
                              bpmFrequency=7.0442e+08, start_position=0)

# Adding a cavity element
dtl_tank_1 = builder.addCavity(sequence=dtl_seq, name="DTL-010:EMR-Cav-001", amplitude=0.003,
                               phase=-35.0, frequency=frequency, position=0)

# Adding a quadrupole magnet
builder.addQuad(sequence=dtl_seq, name="DTL-010:BMD-PMQ-001", length=0.05, field=-59.9444,
                aperture=0.02, aprt_type=1, pos=0.025, maxDriftLength=maxDriftLength)

# Adding an RF gap to a cavity element
ttfs_element = builder.TTFs(beta_max=0.99, beta_min=1e-06, polyT=[1.07731, -0.00257565, -1.17398e-05])

builder.addRFGap(sequence=dtl_seq, cavity=dtl_tank_1, name="DTL-010:EMR-Cav-001:G1",
                 amplitude_factor=0.781302, length=0.0752368, mode=0, EzFile="?", aperture=0.02,
                 aprt_type=1, pos=0.0626084, ttf=0.780607, ttfs_element=ttfs_element)

# Performs sanity checks and add drifts
builder.processNodes(lattice=lattice, sequence=dtl_seq, thinNodes=thinNodes, maxDriftLength=maxDriftLength)

lattice.initialize()
```

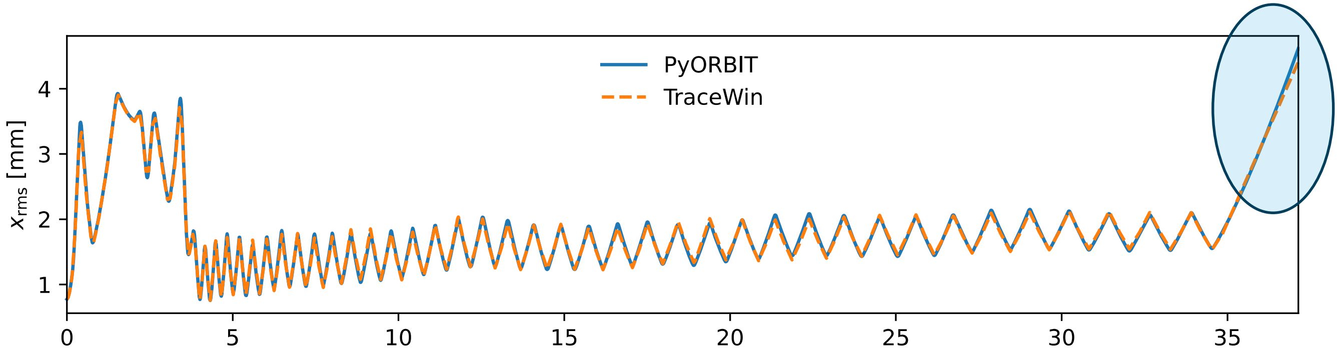
# Benchmark against TraceWin

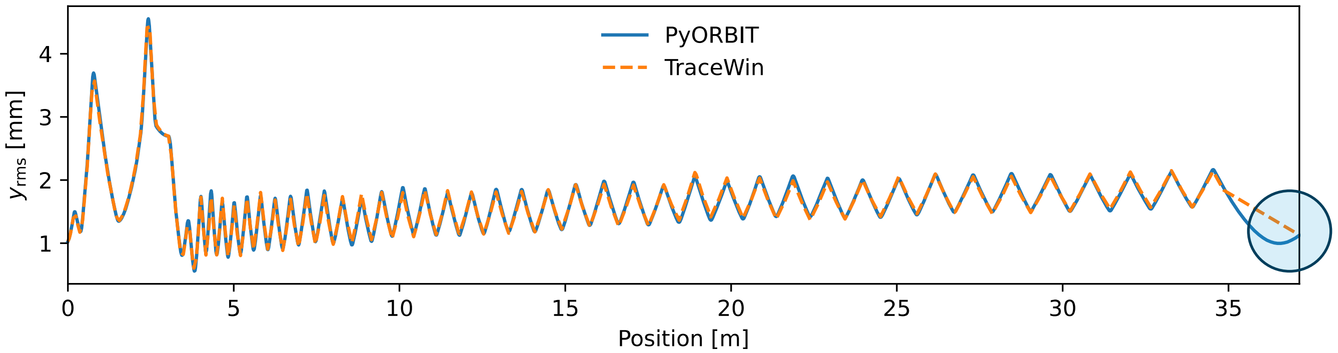
## ESS linac – from MEBT to DTL tank 4

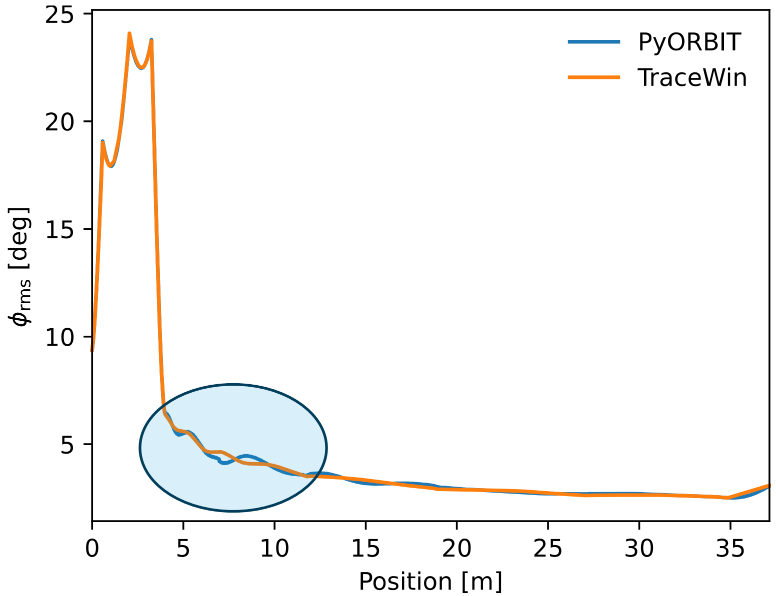
3D space-charge model (Poisson solver)

Same initial distribution (~$10^6$ protons)

Design parameters




Small differences probably due to different meshing strategies in the space-charge solver

# Envelope tracker

**Purpose:** fast simulations for control room applications and quick analisys

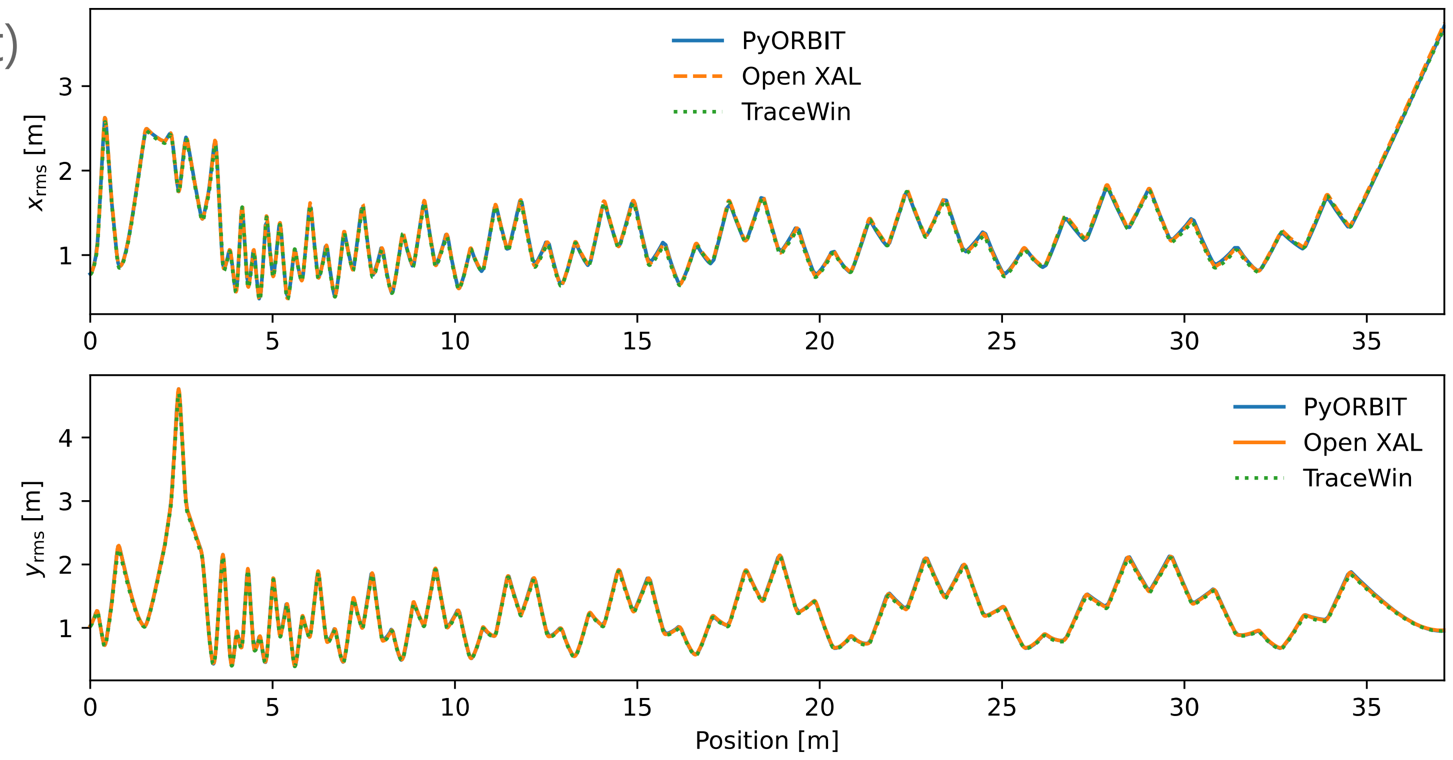
Prototype including RF cavities and quadrupoles

No bends, no space-charge (yet)

Almost perfect agreement

Runtime ~10 ms

(multiparticle ~4 min)



# Control system integration

## EPICS

The ESS control system is based on EPICS

We developed a PyORBIT extension to add EPICS PVAccess support based on p4p

```
from orbit.epics import channels

# Adding EPICS channels to a quadrupole magnet
channels.addMagnetChannels(node=quad_1, current_channel="MEBT-010:PwrC-PSQV-001:Cur-R",
                           current_set_channel="MEBT-010:PwrC-PSQV-001:Cur-S",
                           convFactor=-0.162)

# Adding EPICS channels to an RF cavity
channels.addCavityChannels(node=cavity_1, amplitude_channel="MEBT-010:RFS-LLRF-101:SPRampingA",
                           amplitude_set_channel="MEBT-010:RFS-DIG-101:AI0-SMonAvg-Mag",
                           phase_channel="MEBT-010:RFS-DIG-101:RFCErrLimSBCmp1",
                           phase_set_channel="MEBT-010:RFS-LLRF-101:SPRampingPhase")
```

Code at https://gitlab.esss.lu.se/ess-crs/pyorbit-epics

# Summary and future steps

PyORBIT fulfills our requirements for beam-dynamics simulations for both offline and online applications

Good agreement with our current models

Only reasonable amount of software development required

Plan:

- More exhaustive benchmark (with misalignments, mismatches, etc.)
- Complete implementation of envelope tracker (bends, space-charge)
- Finish EPICS integration (diagnostics)
- Study GPU optimization of multiparticle tracker
- UI framework

# Thank you!

2023-10-10