
UKRI
Science and Technology Facilities Council
ISIS Neutron and Muon Source
Welcome

Benchmarking Experience with PTC-PyORBIT for High-Intensity Synchrotron Studies

How we can confidently utilise ‘AI’ to support benchmarking

Haroon Rafique, Chris Warsop

7th ICFA Space Charge Workshop

02.10.26

UKRI Science and Technology Facilities Council
ISIS Neutron and Muon Source

Contents

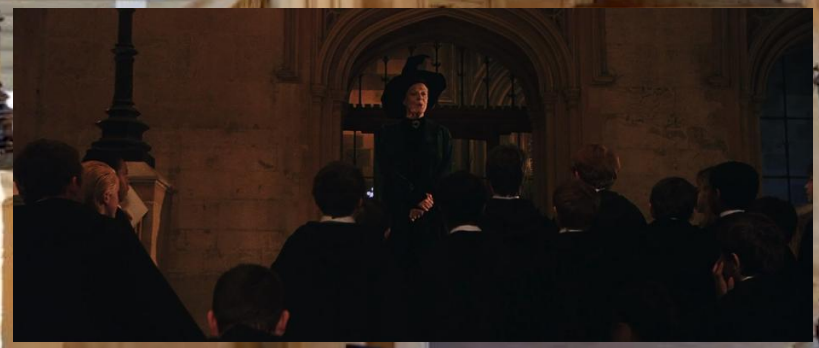
What is PTC / PyORBIT?

What do we use it for?

History

Compiling PTC-PyORBIT3

Benchmarking



Science and Technology Facilities Council

ISIS Neutron and Muon Source

What is PTC-PyORBIT?

PTC

- **Single-particle accelerator optics/tracking code** developed principally by Etienne Forest (**KEK**), with major CERN collab/integration by Frank Schmidt et. al.
- Fortran 90 symplectic integrator / 'kick code' in similar element-by-element tradition to TEAPOT, SixTrack.
- Element models integrated symplectically with user selectable integration order / slicing, allowing controllable numerical accuracy.
- Built around Forest's Fully Polymorphic Package (FPP)
  - Variables can represent ordinary floating point numbers or Taylor Series
  - Enables high-order transfer maps, differential algebra/TPSA, Lie methods and normal form analysis useful for tracking and non-linear optics
- Detailed geometrical representation of accelerator
  - **3D magnet placement / orientation**
  - **Access to optics quantities inside thick elements**
  - Sector and rectangular bends, recirculators and other non-trivial beam-line topology
  - Forward and backward propagation
- Integrated into MAD-X as MAD-X/PTC, giving MAD-X access to PTC tracking, higher order maps, and nonlinear optics. Also integrated into BMAD.

ORBIT

- **Macro-particle simulation code** developed by **SNS** @ ORNL (+ CERN, GSI) for **single bunch particle tracking** with **physics processes**

PTC-ORBIT

- 2010 developed @ KEK (E. Forest, A. Molodeozhentsev)

PyORBIT

- Python wrapping of (C++) ORBIT: used **python 2.7** with custom C++ and MPI bindings, original code uses **TEAPOT** tracker, PTC integration carried over from PTC-ORBIT
- ESS developed Python 3.0 version (referred to as PyORBIT3) in 2023
  - Still under active development and usage by multiple facilities

PTC-PyORBIT

- 2017 (ish) forked by CERN (+ **PTC** library) for LIU (H. Bartosik, JB. Lagrange, et. al.)
- Benchmarked on CERN injectors for various **space charge** studies
- @ ISIS:
  - ORBIT code used for beam dynamics, later TEAPOT-PyORBIT (python 2.7)
  - PTC-PyORBIT (python 2.7) used since 2020, running on SCARF (HPC)

PTC provides the accelerator model and tracking,
PyORBIT provides the macro-particle beam and collective effects

What do we use PTC-PyORBIT for?

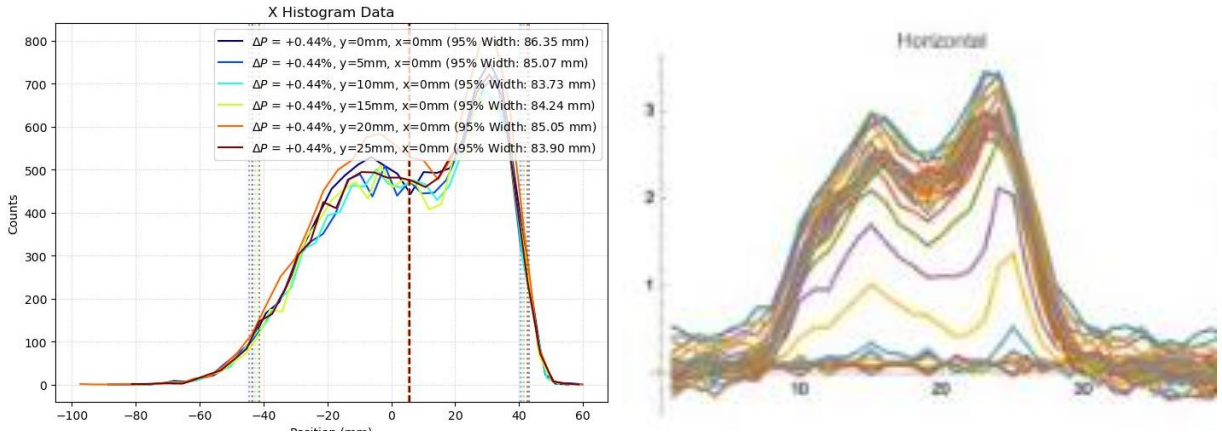
Goals:

- Understand the accelerator to **improve operation (reduce losses / increase efficiency)**
- Develop end-to-end **digital twin** for operational optimisation and training humans to operate

PTC-PyORBIT used for ISIS RCS:

- Operational lattice model development
  - Defining the bare lattice
  - Misalignment studies
  - Injection painting studies
  - Tune control
  - Misalignments
- High intensity studies
  - Resonance crossing studies
  - Non-linear dynamics

Benchmark: Code vs Measurement behaviour



Simulated (left) and measured (right) horizontal profiles for similar painting parameters

PTC-PyORBIT is one of our tools for high intensity RCS beam dynamics studies

2025: Python 2.7 obsolescence

- Can no longer run legacy PTC-PyORBIT on newer systems (required for ‘security’)



2025: Python 2.7 obsolescence

- Can no longer run legacy PTC-PyORBIT on newer systems (required for ‘security’)
- Need to fit PTC (Fortran90 wrapped in C++ wrapped in python) into PyORBIT3





LLMs and Agentic AI

	Large Language Model (LLM)	Agentic AI
What is it?	Model trained on text/code to understand context and **generate** useful **response**	System using LLM for reasoning and planning to **take actions towards a goal**
Typical interaction	**Prompt -> Response**	**Goal -> Plan -> Act -> Inspect -> Adapt -> Repeat**
Can it use tools?	Not inherently	Usually: **code execution, files, web, APIs, databases, other models/agents**
State / workflow	Primarily responds to current context	Can **maintain task state** and **execute multi-step workflows**
Accelerator Example	"**What is** the SIS18 frozen space charge benchmark?"	"**Plan** a new repository for the complete SIS18 benchmark **using** PTC-PyOBIT3, **based on** the completed PTC-PyORBIT2.7 SIS18 benchmark repository, **using** the website reference plots to produce side-by-side outputs. **Use** the superpowers brainstorming **skill**, the existing repository is here …, the agent should **start with** step 1 and **not progress until** the user confirms the result."

LLMs:

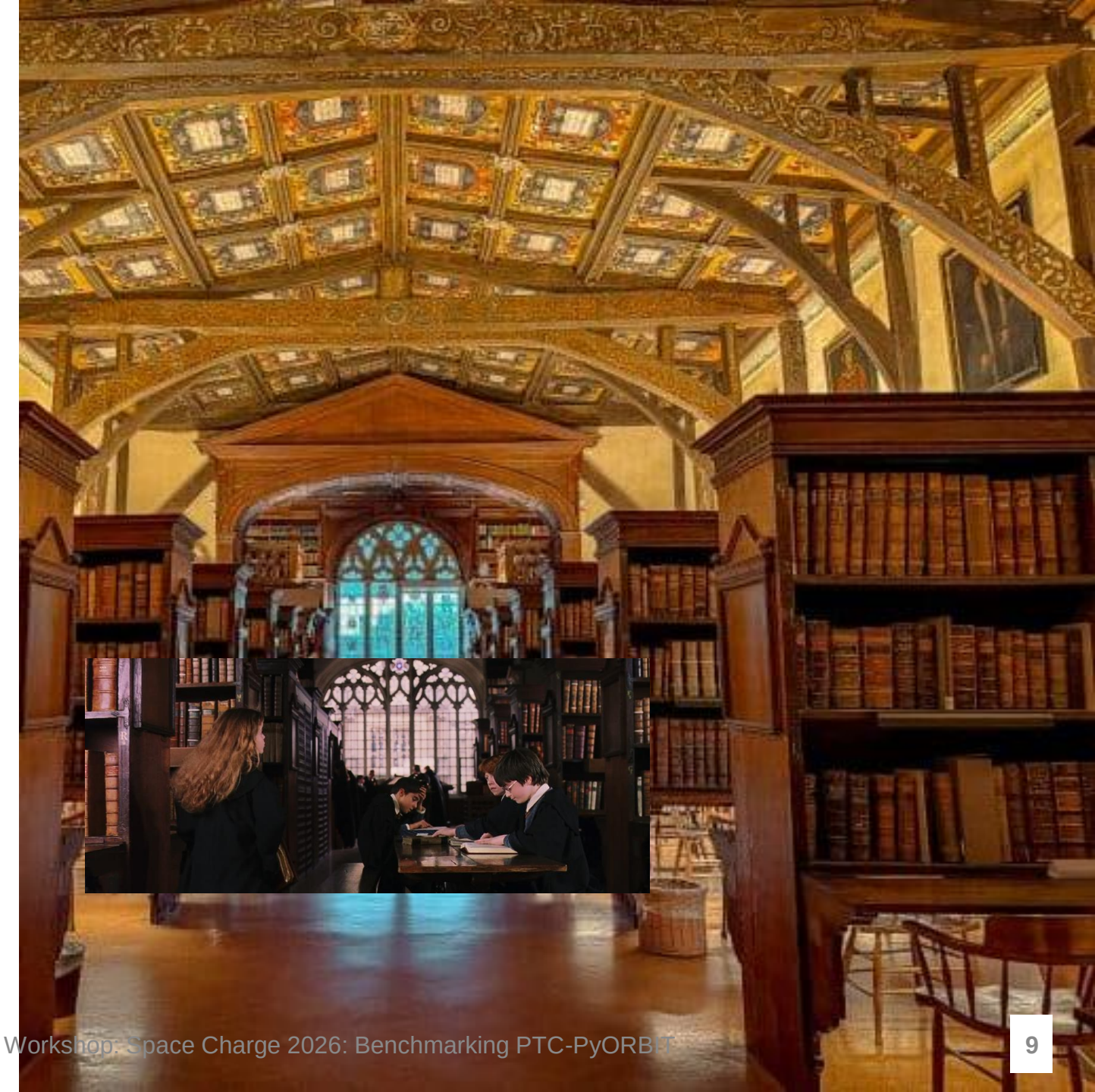
- ChatGPT
- Gemini
- Meta AI / Muse
- Copilot
- Claude
- Grok
- Deepseek
- Qwen / Qianwen
- Doubao
- Le Chat
- Kimi

Agentic AI:

- ChatGPT Work / Codex
- Gemini Spark
- Muse
- Claude Cowork
- Copilot + Agents
- Manus
- Genspark Super Agent
- Perplexity
- GitHub Copilot coding agents
- Cursor
- Devin / Cognition
- Z.ai / GLM
- Google Antigravity

Human vs AI

- Learned meson build
- Support from ISIS Scientific Computing (Dr. Sam Tygier)
- StackOverflow
- Debugging with LLMs (browser based)
- Successful compilation after intermittent effort over months
- Runtime segmentation fault
  - Didn’t have time/resource to resolve



Science and Technology Facilities Council

ISIS Neutron and Muon Source

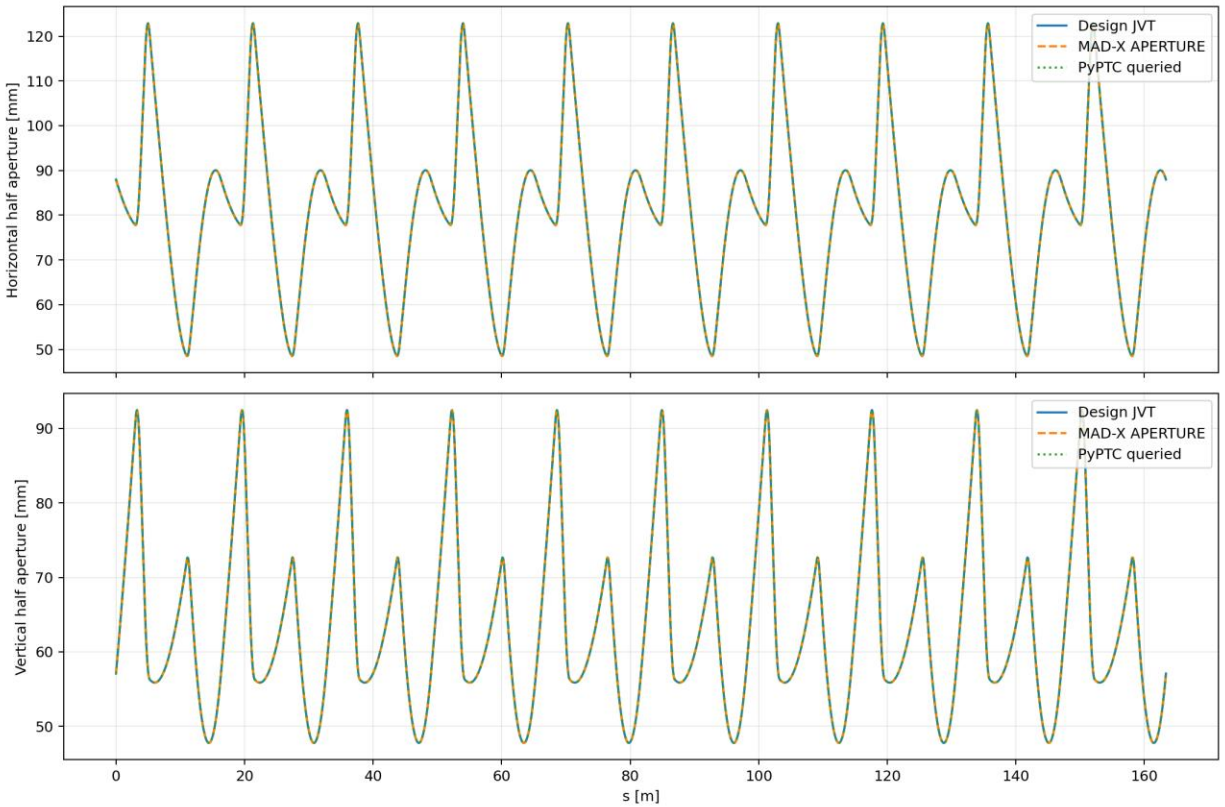
Human vs **AI**



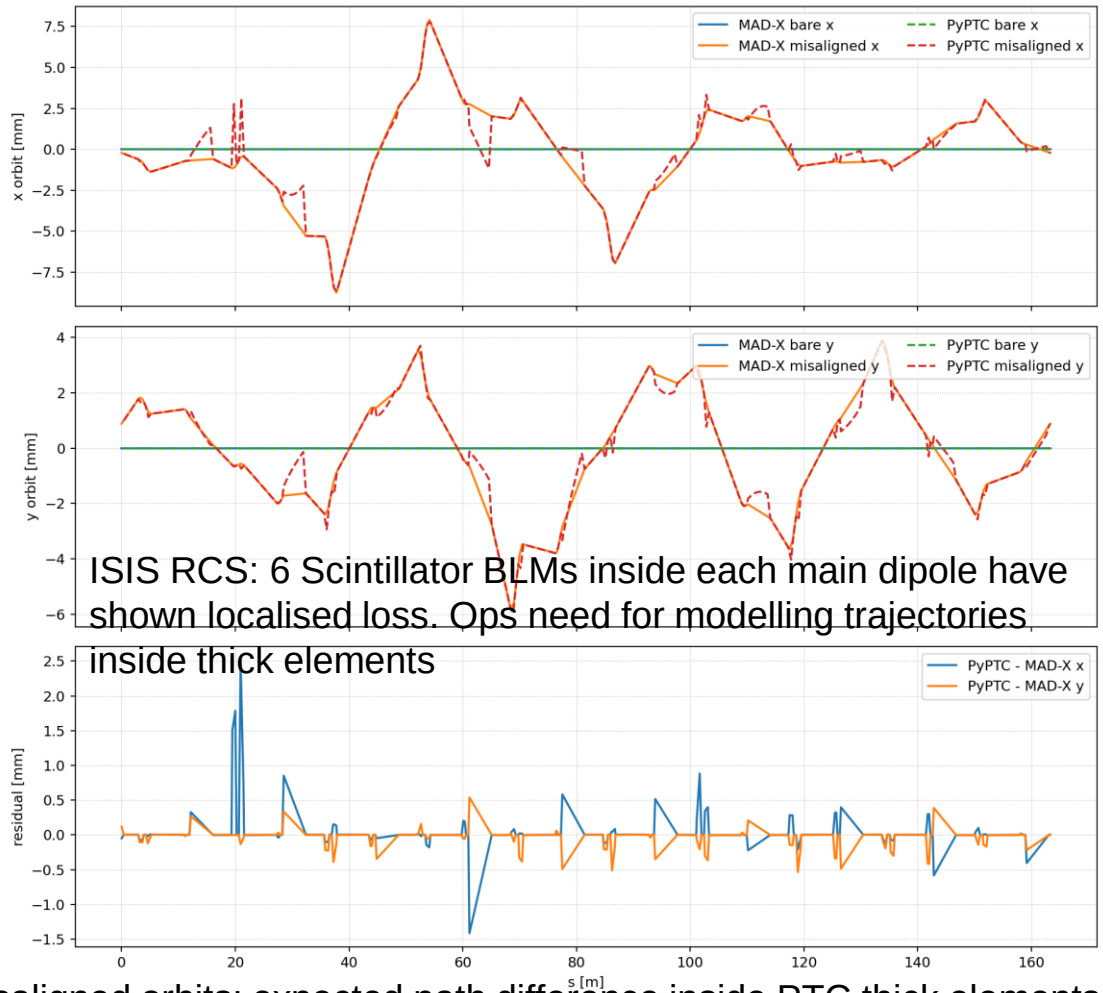
- Using agentic command line interface (CLI)
- **Iterative** approach needed
  - AI is **not intelligent**
- Developed **good practice**
  - AI requires robust and rigid safeguards, **rules**, objectives, resources (repos, documentation, instruction)
- Able to achieve more than initial intention
  - **Wrapped** PTC in python standalone (pyPTC)
  - **Compiled** PTC-PyORBIT3
  - **Exposed** more PTC **functionality** to PyORBIT
  - **Developed** better **tests/benchmarks**
- Made mistakes
  - More than a human (?) but different types
  - Human must remain authority
  - **Regular guidance → Documented rules**

PyPTC

- Standalone python library wrapping PTC
- Provides independent tracking/benchmarking asset
- GitHub Repo: https://github.com/HaroonRafique/PTC/tree/master/PyPTC



Apertures agree between design, MAD-X, pyPTC



ISIS RCS: 6 Scintillator BLMs inside each main dipole have shown localised loss. Ops need for modelling trajectories inside thick elements

Misaligned orbits: expected path difference inside PTC thick elements

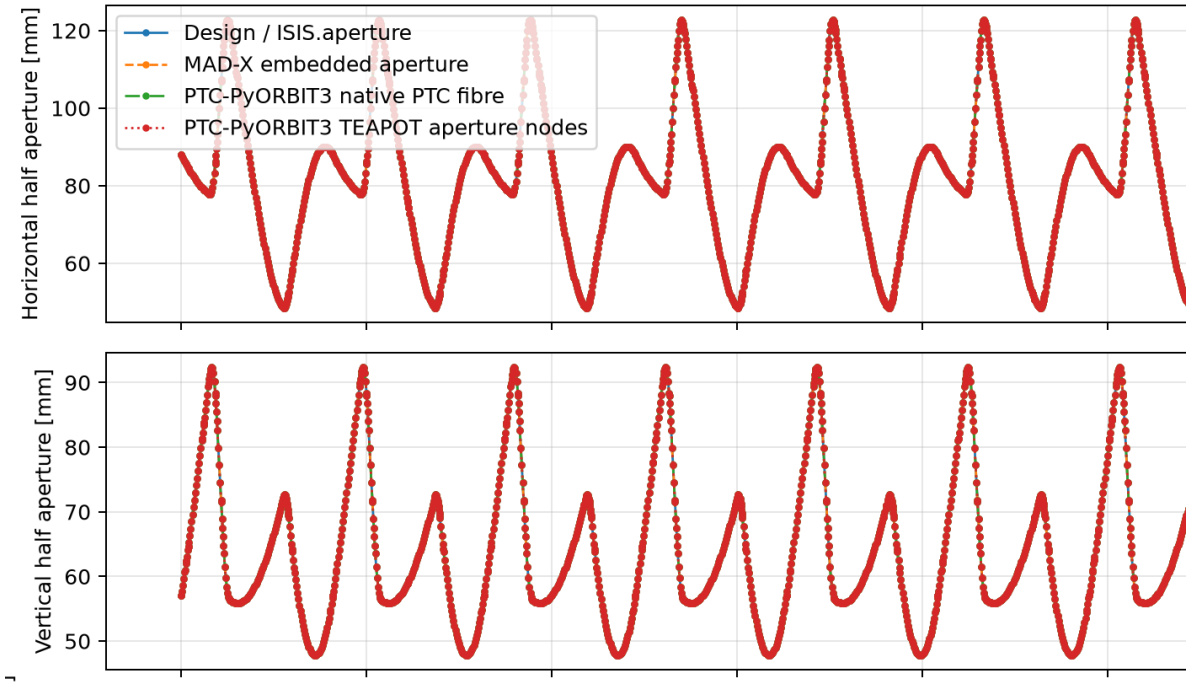
Science and Technology Facilities Council

ISIS Neutron and Muon Source

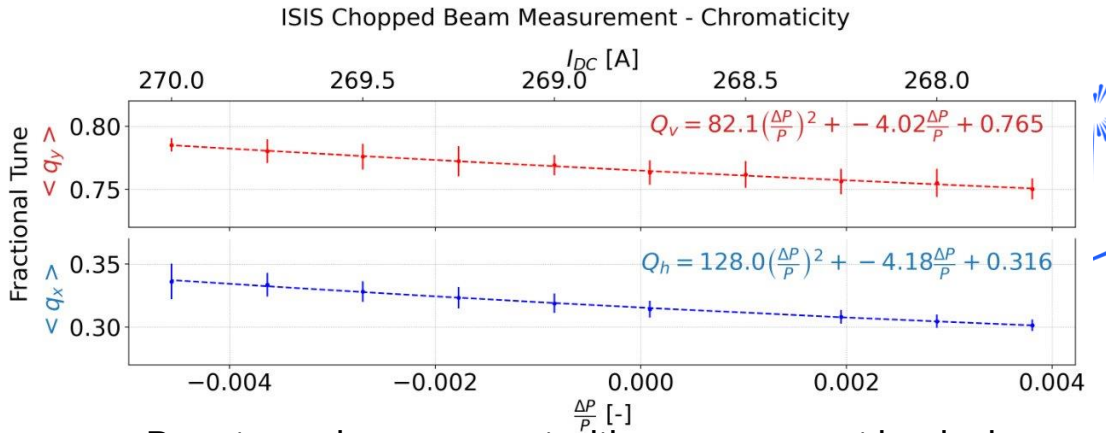
Benchmark: Code vs Code consistency

PTC-PyORBIT3 Benchmarking

- Aperture consistent between all codes
- Chromaticity behaviour similar between model and measurement

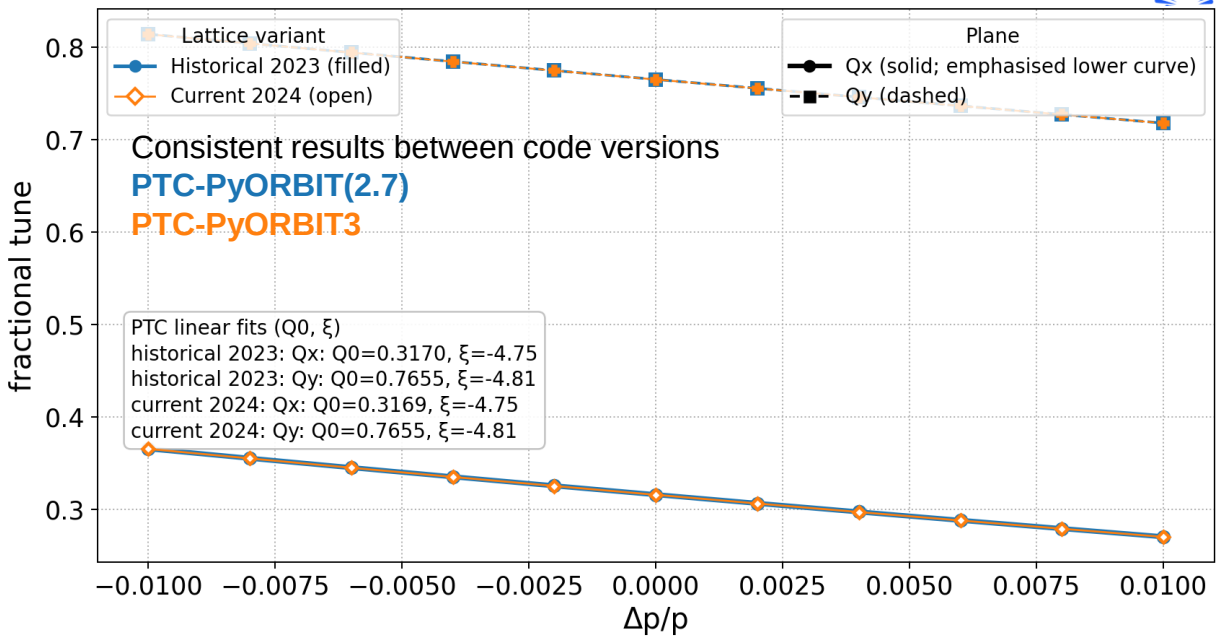


Apertures agree between design, MAD-X, PTC, TEAPOT



Bare tunes in agreement with measurement by design

Benchmark: Code vs Measurement Behaviour



Benchmark: Code vs Code consistency

SIS18 Frozen Space Charge Benchmark



THBW01 Proceedings of HB2006, Tsukuba, Japan

CODE BENCHMARKING ON SPACE CHARGE INDUCED PARTICLE TRAPPING

G. Franchetti, I. Hofmann, and S. Machida*
GSI, 64291 Darmstadt, Germany
*CCLRC/RAL/ASTeC, Chilton, Didcot, Oxon, U.K.

Benchmarking of Trapping particles induced by space charge and nonlinear resonances

G.Franchetti*, S. Machida+, I.Hofmann*
*GSI, +RAL

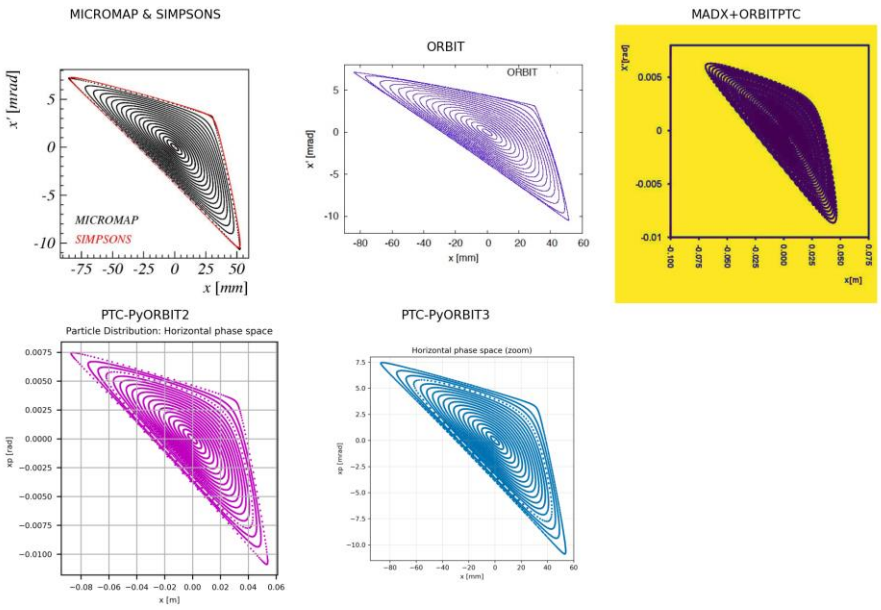
HB2006, 1st June 2006, Tsukuba

http://www-linux.gsi.de/~giuliano/research_activity/trapping_benchmarking/main.html

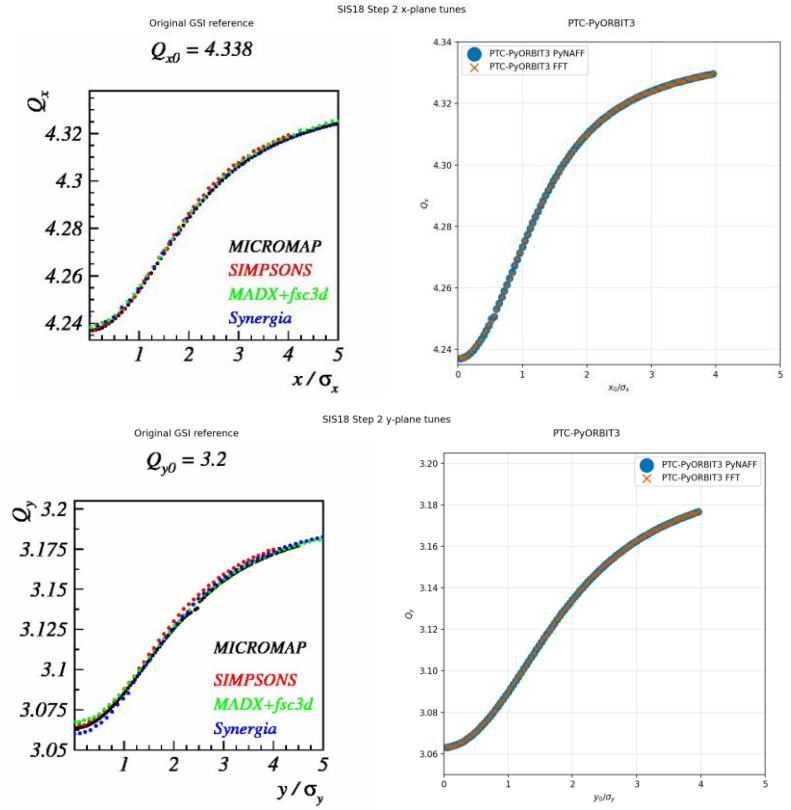
HB2006 1.6.2006 G. Franchetti 1

Science and Technology Facilities Council
ISIS Neutron and Muon Source

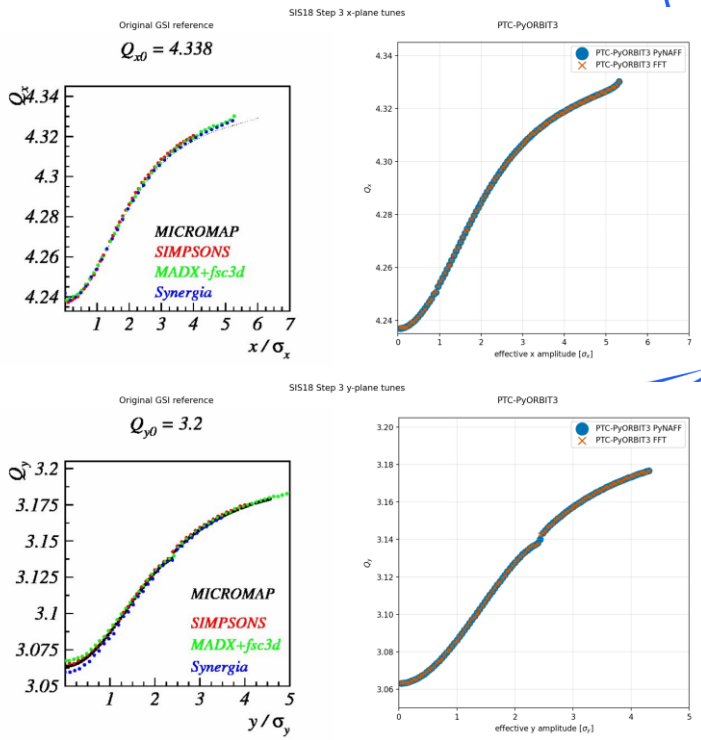
PTC-PyORBIT3: SIS18 frozen space charge benchmark



Step 1: validate that all codes reproduce the same third-order-resonance Poincare topology and dynamic-aperture boundary with frozen space charge



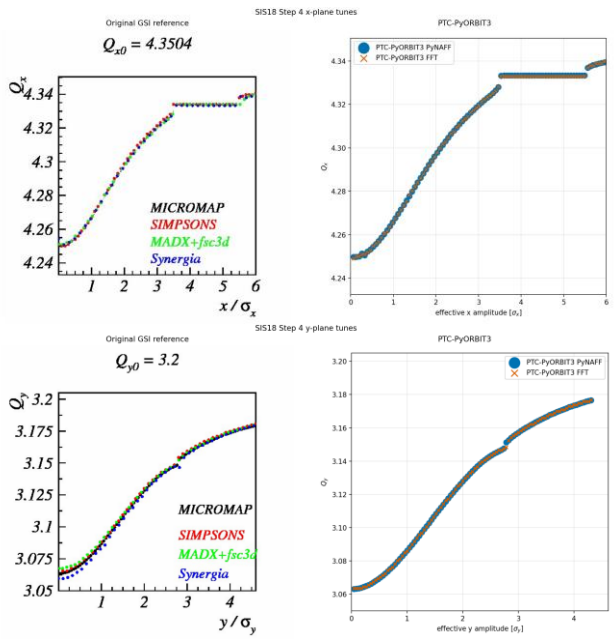
Step 2: Sextupoles off, benchmark amplitude dependent tune shift from frozen space charge



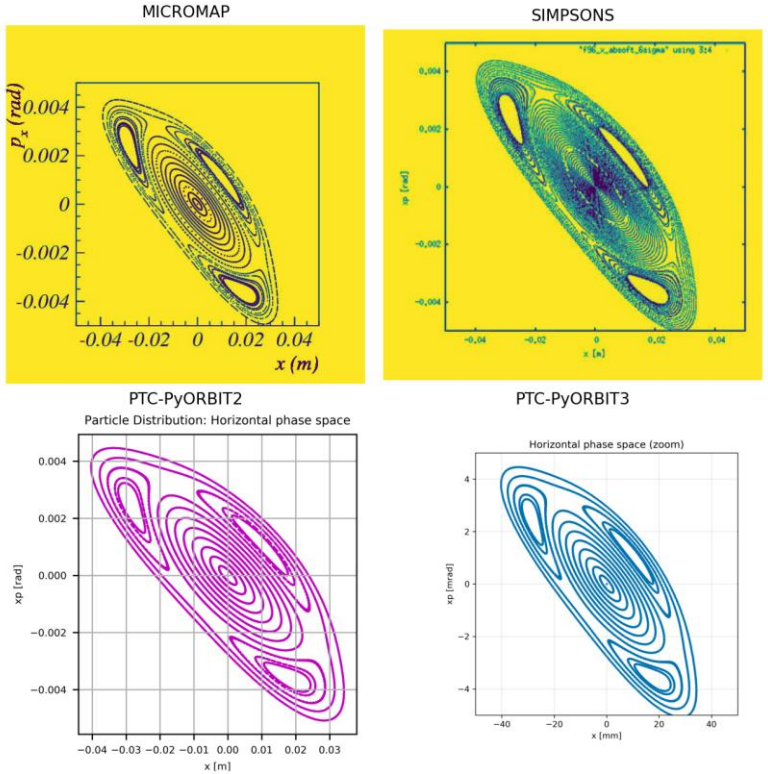
Step 3: Sextupoles on at Q_x=4.338 to verify the combined space charge and non-linear detuning before islands enter sampled range

[SIS18 Benchmark: G. Franchetti, S. Machida, I. Hofmann](#) [HB06 Paper](#)

PTC-PyORBIT3: SIS18 frozen space charge benchmark



Step 4: Move closer to 3rd order resonance at Q_x = 4.3504 to expose amplitude dependent tune structure associated with islands

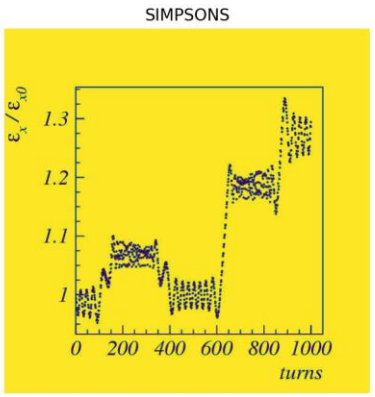


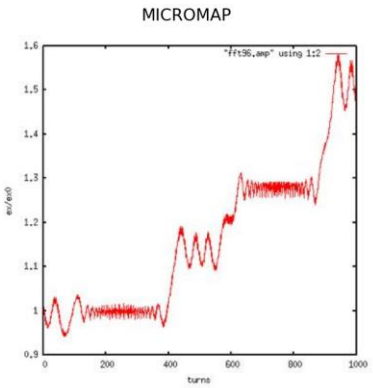
Step 5: Compare space charge plus sextupole phase space topology, including central orbit, physical extent, and three stable islands

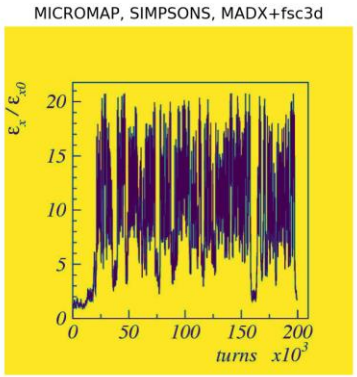
Step 6: Follow 1 particle through slow 15,000 turn synchrotron oscillation to test adiabatic resonance trapping through its normalised horizontal invariant

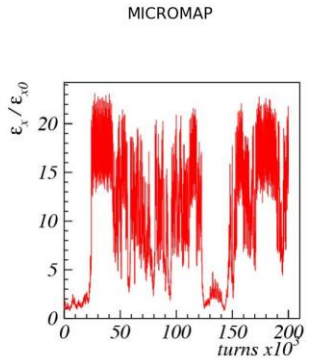
[SIS18 Benchmark: G. Franchetti, S. Machida, I. Hofmann](#) [HB06 Paper](#)

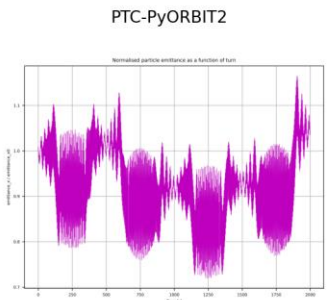
PTC-PyORBIT3: SIS18 frozen space charge benchmark

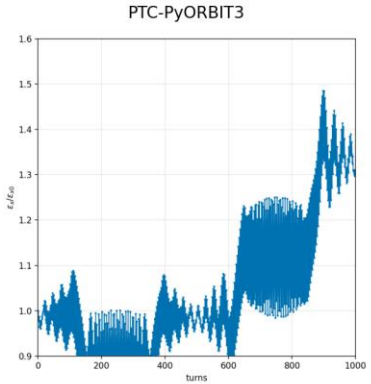


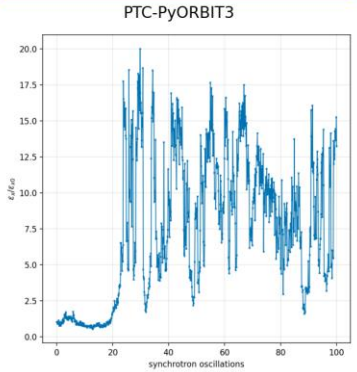


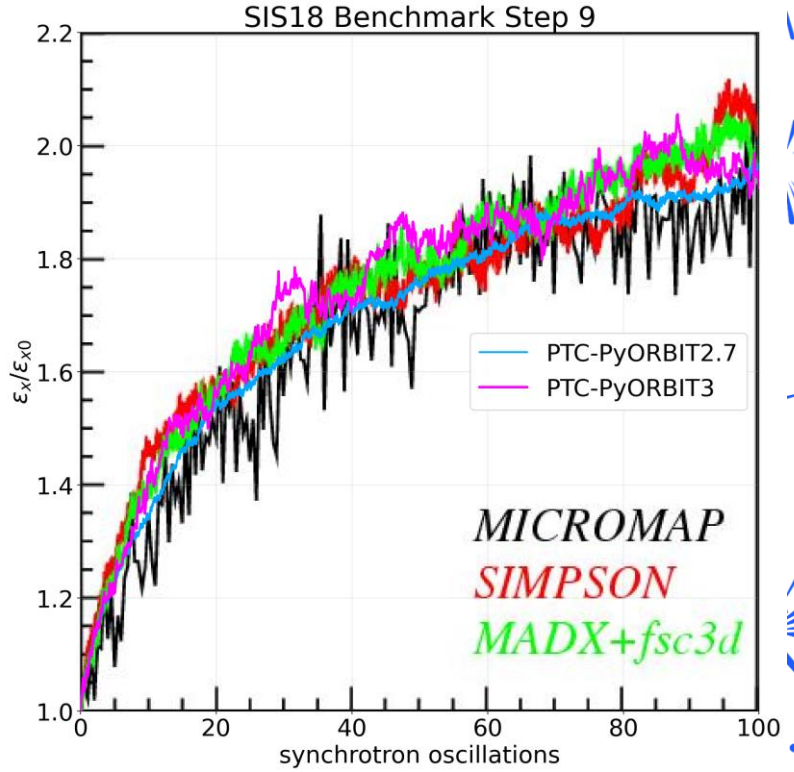












Step 7: Increase synchrotron tune to $Q_s = 10^{-3}$ to show transition from adiabatic trapping to fast scattering

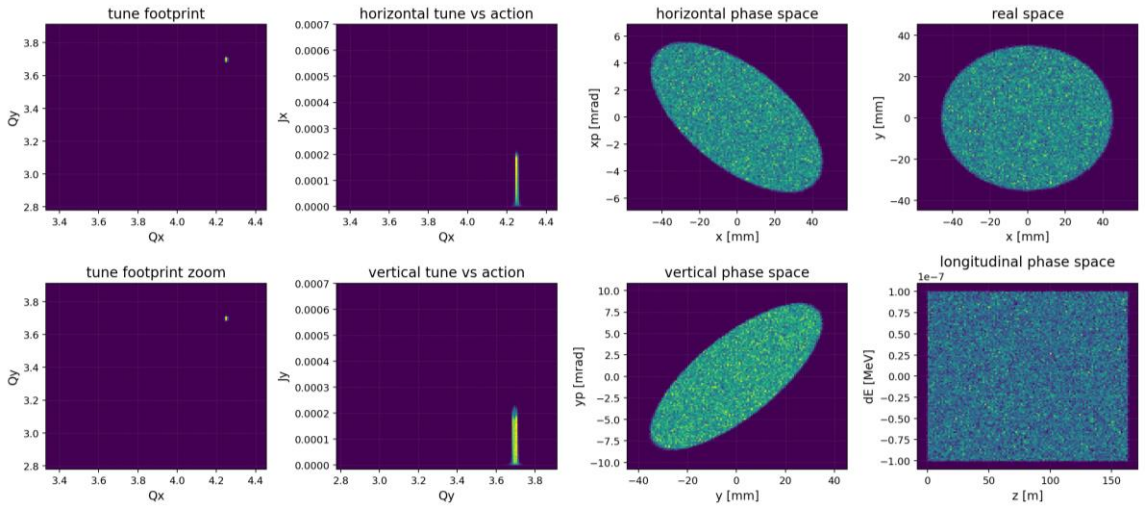
Step 8: Extend fast-regime single particle case to long storage and compare resulting scattering/trapping action evolution across codes

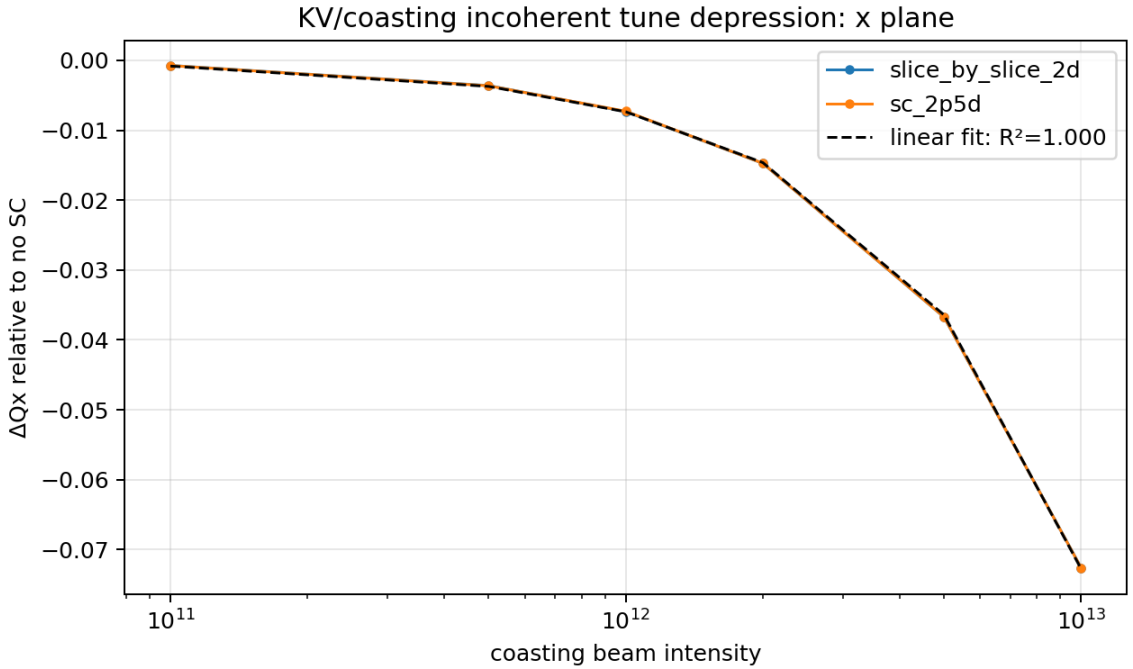
Step 9: Track a full Gaussian bunch at $Q_s = 10^{-3}$ to benchmark long term transverse RMS emittance growth from the collective trapping/scattering dynamics

Science and Technology Facilities Council
ISIS Neutron and Muon Source

SIS18 Benchmark: G. Franchetti, S. Machida, I. Hofmann HB06 Paper

PTC-PyORBIT3: PIC space charge benchmark





- Coasting KV (Kapchinskij-Vladimirskij) distribution
  - *Exact self consistent solution of the Vlasov-Poisson system for a beam in linear periodic focusing, with its self-generated space charge field remaining linear and preserving the distribution.*
  - No longitudinal variation -> no variation in line density
  - Expect a tune depression, but no tune spread
  - Tune depression linearly dependent on Intensity

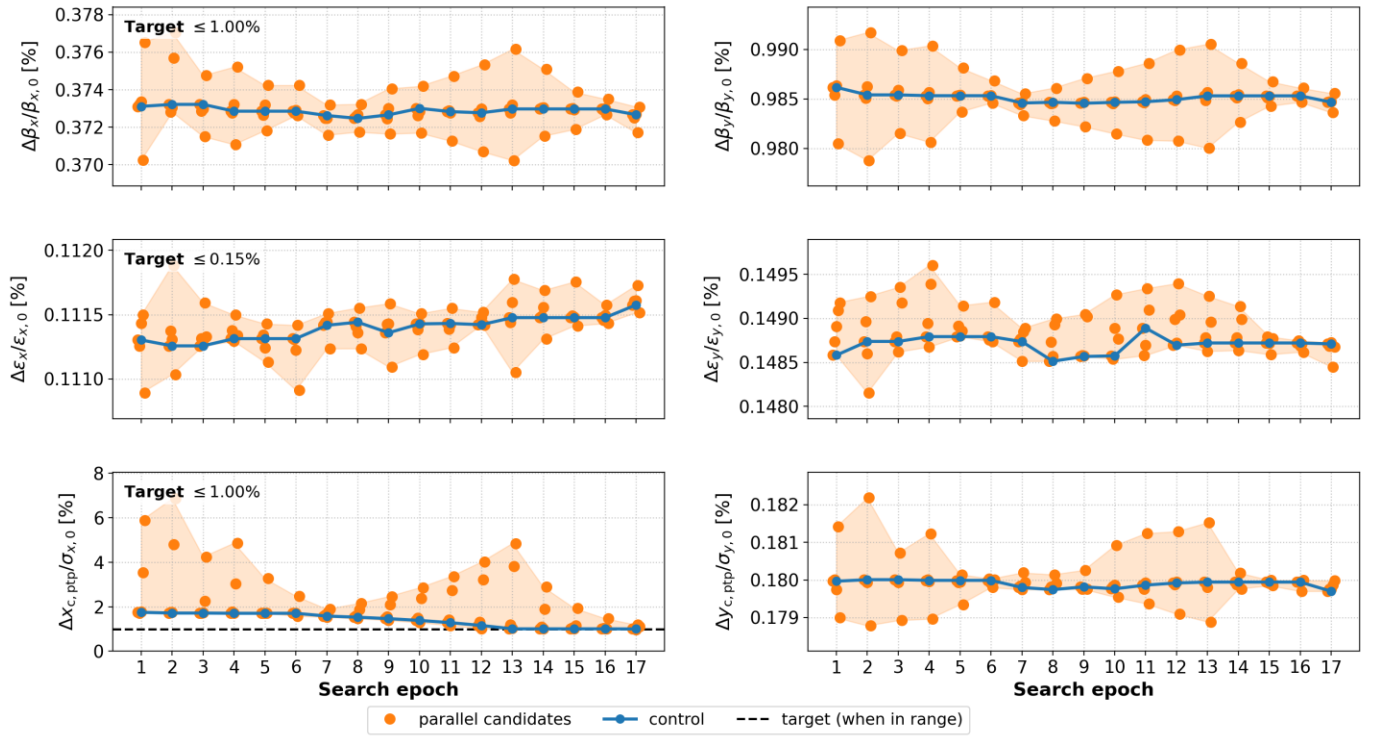
Science and Technology Facilities Council
ISIS Neutron and Muon Source

Agentic AI approaches to benchmarking

Things Agentic AI does well (with the correct guidance)

- Code porting
- Automated testing
- Implement existing well documented benchmarks
- Applying working methods to new environments

- Convergence testing
  - How many macro-particles are needed to resolve behaviour?
  - What space charge grid size best balances prediction and run time?
  - Needs well defined metrics
- Test: Can I perfectly match a KV distribution?
  - Used to perform convergence on initial distribution to minimise; beta-beating, emittance growth, and centroid motion



PHYSICAL REVIEW ACCELERATORS AND BEAMS **24,** 044201 (2021)

Computation of the matched envelope of the Danilov distribution

A. Hoover,[1] N. J. Evans,[2,*] and J. A. Holmes[2]
[1]*Department of Physics and Astronomy, University of Tennessee, Knoxville, Tennessee 37966, USA*
[2]*Oak Ridge National Laboratory, One Bethel Valley Road, Oak Ridge, Tennessee 37831, USA*

Science and Technology Facilities Council
ISIS Neutron and Muon Source

Lessons from Agentic AI usage

- Set rules, boundaries, goals, dependencies, expectations in AGENTS.md: keep it focussed
  - Rule: Read the AGENTS.md, WORK_DIARY.md, and README.md files before any execution, refer to these files and store important information in the relevant file to minimise dependency on context windows
- Work in Git repositories; Agents delete a lot of useful things that have to be retrieved
  - Rule: All completed code changes and generation of input/output artefacts should be committed as separate git commits, the user should be prompted for confirmation before committing any files larger than 1MB
- Store all prompts and summaries of agent outputs in a WORK_DIARY.md: record decisions, process and reasons behind them, constraints that may change in time or with further development
  - Rule: All prompts and summaries of Agent outputs must be stored in the timestamped WORK_DIARY.md for provenance, the Agents.md should be updated with rules and constraints as provided by the user, and the README.md file should be kept up to date by the agent without prompting as part of each prompt resulting in file changes.
- Don’t rely on the ‘context window’: store plans in .md files
  - Rule: Store complete plans in .md files, refer to these in the README.md, and AGENTS.md, including their status. Ensure that these plan .md files are updated as part of their implementation.
- Use Superpowers and Skills
  - Rule: Use the superpowers skills in all execution, this can be found in <directory>
- Don’t use agents to run simulations
  - Rule: All simulations should be run by the user, the agent should implement all simulations with smoke tests to debug using small particle numbers, turns, grid sizes, etc. to minimise unnecessary usage

Summary

- Porting legacy codes to modern frameworks seems to be a recurring theme in my career
  - Complex, tedious, **time consuming**, requires dedicated commitment

Summary

- Porting legacy codes to modern frameworks seems to be a recurring theme in my career
  - Complex, tedious, **time consuming**, requires dedicated commitment
- **Evidence based approach** to understanding, porting, wrapping, compiling, and benchmarking Acc. Phys./S.C codes with Agentic AI
  - **LLM**s are well suited to such language-based problems, not to be trusted with much else
  - **Agentic AI** (LLMs that can execute commands to achieve goals) are better suited
    - Requires very clear boundaries, rules, babysitting
    - Copilot – humans must define correct, provide feedback, direct investigation, be authority



Summary

- Porting legacy codes to modern frameworks seems to be a recurring theme in my career
  - Complex, tedious, **time consuming**, requires dedicated commitment
- **Evidence based approach** to understanding, porting, wrapping, compiling, and benchmarking Acc. Phys./S.C codes with Agentic AI
  - **LLM**s are well suited to such language-based problems, not to be trusted with much else
  - **Agentic AI** (LLMs that can execute commands to achieve goals) are better suited
    - Requires very clear boundaries, rules, babysitting
    - Copilot – humans must define correct, provide feedback, direct investigation, be authority
- Working with code-aware copilots, we can focus more on the science
  - New approaches and methods to be considered: **trust** and **provenance** are key
  - Plan to use similar approach to systematically benchmark suite of codes at ISIS

Summary

- Porting legacy codes to modern frameworks seems to be a recurring theme in my career
  - Complex, tedious, **time consuming**, requires dedicated commitment
- **Evidence based approach** to understanding, porting, wrapping, compiling, and benchmarking Acc. Phys./S.C codes with Agentic AI
  - **LLM**s are well suited to such language-based problems, not to be trusted with much else
  - **Agentic AI** (LLMs that can execute commands to achieve goals) are better suited
    - Requires very clear boundaries, rules, babysitting
    - Copilot – humans must define correct, provide feedback, direct investigation, be authority
- Working with code-aware copilots, we can focus more on the science
  - New approaches and methods to be considered: **trust** and **provenance** are key
  - Plan to use similar approach to systematically benchmark suite of codes at ISIS
- pyPTC available as standalone access to polymorphic tracking
- PTC-PyORBIT3 compiled with meson build, benchmarking ongoing
- PTC-PyORBIT3 SIS18 benchmark completed

Question: What does the future of modelling look like?

Haroon.Rafique@stfc.ac.uk



Haroon Rafique He/Him
Senior Accelerator Physicist at STFC

Connect with me on LinkedIn

Science and Technology Facilities Council

ISIS Neutron and Muon Source



2024 RAL Open Days: ISIS Target Station 2 Accelerator Zone #WheresWally #WheresWaldo

PTC_PyORBIT3_SIS18_Benchmark / AGENTS.md

Preview | Code | Blame | 130 lines (117 loc) · 7.61 KB | Raw

SIS18 PTC-PyORBIT3 Benchmark Agent Instructions

Purpose

This repository reproduces the nine-step SIS18 PTC-PyORBIT benchmark using the current PTC-enabled PyORBIT3 stack. The scientific obj
is to validate space-charge-induced trapping, tune behaviour, and long-term emittance evolution against the historical benchmark, not to
preserve Python-2 code.

Read `README.md`, `BENCHMARK_PLAN.md`, and the detailed Superpowers documents under `docs/superpowers/` before non-trivial work.

Authoritative sibling repositories

- `../PyORBIT3`, branch `PTC`: runtime and PTC integration source of truth.
- `../PTC`: PTC source of truth. Do not edit PTC Fortran sources for this benchmark unless the user explicitly requests it.
- `../ptc_pyorbit3_examples`: the required reference for Python structure, configuration, MPI, PTC tracking, diagnostics, plotting, and ou
conventions. Do not import it at runtime or alter its unrelated worktree.
- `/home/hr/Repositories/PTC_PyORBIT_SIS18_Benchmark`: read-only historical result artifacts only. Resolve it through `SIS18_REFERENCE_RO`
never write to it or copy its outputs into this repository.
- `/home/hr/Repositories/superpowers`: required development methodology.

Required development method

- Before plan changes, read and follow `skills/writing-plans/SKILL.md` in the Superpowers repository.
- For implementation, use `skills/executing-plans/SKILL.md` task by task.
- Before production Python changes, follow `skills/test-driven-development/SKILL.md`: add one focused test, run it and observe the
expected failure, then implement the smallest passing change.
- For every test/build/runtime/physics failure, follow `skills/systematic-debugging/SKILL.md` before attempting a fix.