HZB BESSY II Light Source

HARMONIC CAVITY SIMULATIONS

Comparison pyAT, Pelegant & mbtrack/mbtrack2

Teresia Olsson,

AT Workshop, 2-3 October 2023

Code benchmarking

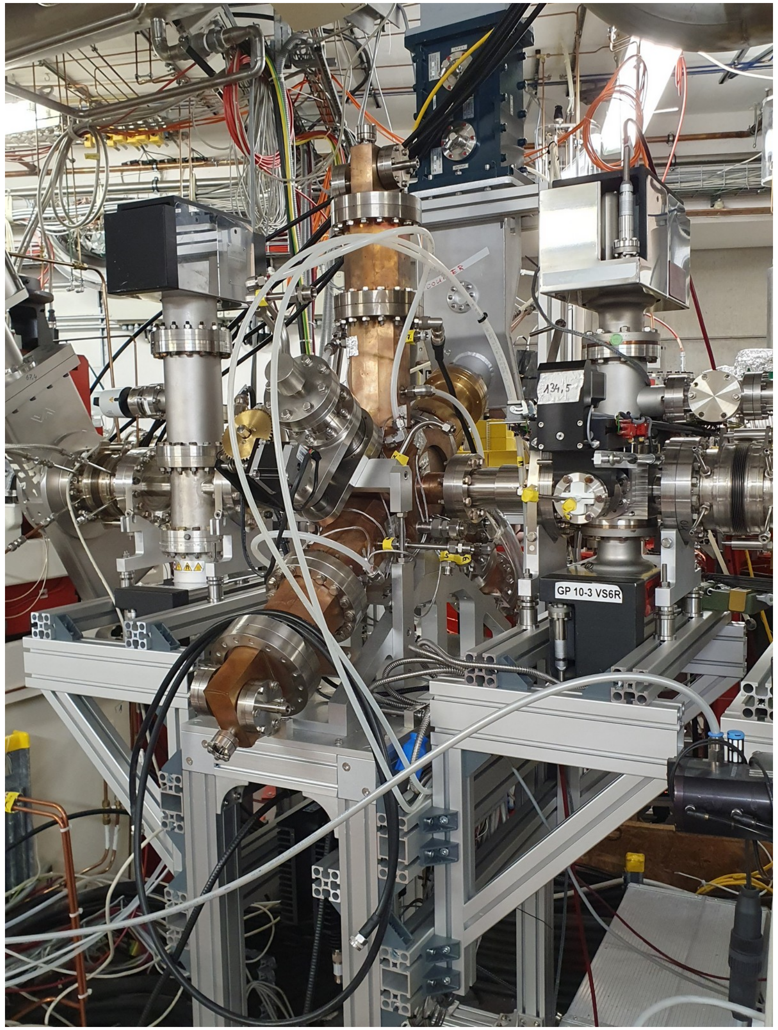
INTRODUCTION

WHY SIMULATE HARMONIC CAVITIES?

- Harmonic cavity = cavity operated at harmonic of the main RF $\rightarrow$ change the RF potential seen by the beam.
- Can be passive/active and normal/superconducting.

- HCs already used in many machines: mainly to increase lifetime.
- Interest for HCs has increased due to MBA upgrades $\rightarrow$ lifetime, stability, intrabeam scattering mitigation.

- But HCs can also cause problems: transient beam loading, longitudinal instabilities.



3rd order active normal conducting harmonic cavity from ALBA installed in BESSY II

IMPORTANT USER CASES

- Transient beam loading when operating with gaps in the fill pattern.
- Longitudinal coupled-bunch instabilities driven by the HC impedance or caused by flattening of the potential well.
- Effect on intrabeam scattering.
- Effect on transverse + longitudinal single bunch instabilities.
- Effect on transverse coupled-bunch instabilities.
- RF feedback behaviour $\rightarrow$ especially of interest for active HCs but also Robinson instability.
- Injection studies with HC $\rightarrow$ change of RF bucket, accumulation, missed shots.

Often require at least 10 000 particles per bunch and tracking for > 50 000 turns (depends on damping times)

BENCHMARKING OF CODES

- Four codes chosen:
  - pyAT
  - Pelegant
  - mbtrack (C++ version available at MAX IV)
  - mbtrack2 (Python version available at Soleil)

- MAX IV 3 GeV ring case → only 176 bunches.
- Requirements: 10 000 particles per bunch + 150 000 turns.
- Looked at:
  - User-friendliness
  - Physics results
  - Execution time

Work in progress so this is just my personal first impressions

BENCHMARKING

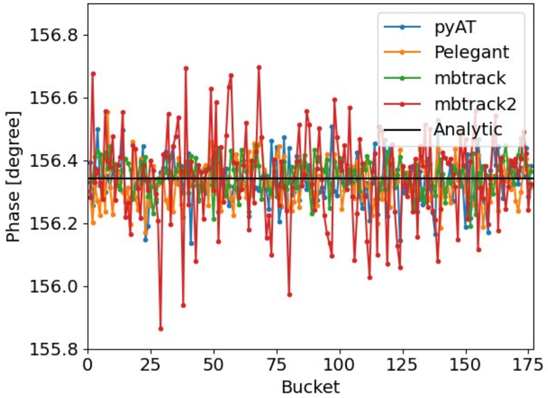
USER-FRIENDLINESS

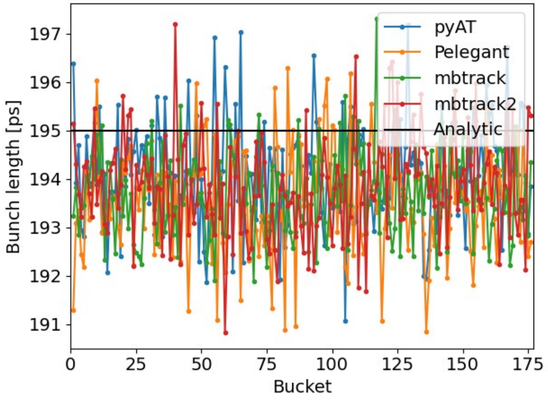
	pyAT	Pelegant	mbtrack	mbtrack2
Installation	Easy but need to choose which of three parallelisation options to use at compile time	Complicated to build from source but precompiled binaries exist	Very easy	Easy unless HD5F and h5py not already built with MPI support
Documentation	Good, manual and example scripts	Good, manual and example scripts + **user forum**	Manual exist, but not so detailed	Good, manual included as part of example notebooks
Setting up simulation	Easy, but a bit complicated if you want to use MPI	Several steps required: creating beam, constructing one turn map, setting up cavities correctly RF feedback complicated	Very easy, but need to insert lattice parameters manually	Very easy, lattice parameters can be imported directly from pyAT but in a questionable way
Change settings	Quick	Quick, but new fill pattern has to be generated separately	Quick, but new fill pattern has to be generated separately (just text file)	Quick
Flexibility	Good	Very complicated to modify or add features	Difficult	Good
Parallelisation	Multiprocessing, openMP, MPI Number of particles need to be multiple of number of bunches, but no restriction for number of processes	No restrictions, but relative number of particles per bunch give bunch charge	Number of process = number of bunches → need to oversubscribe cores	Number of process = number of bunches → need to oversubscribe cores
Output analysis	Easy	Data stored in SDDS so need tools to access	Easy, but confusing due to conventions for head/tail of train and phase on pos slope of sin wave	Easy

PHYSICS RESULTS SO FAR

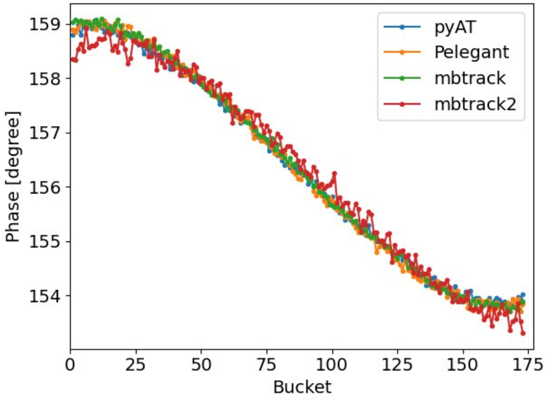
- Simulations with MC without beam loading + passive HC with beam loading.

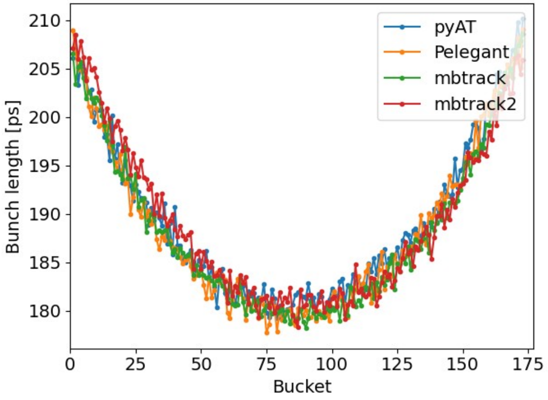
176 bunches



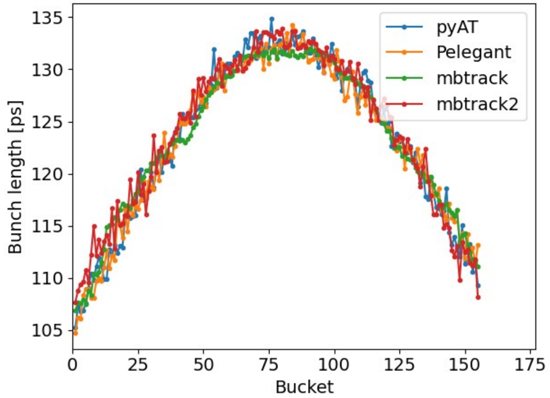


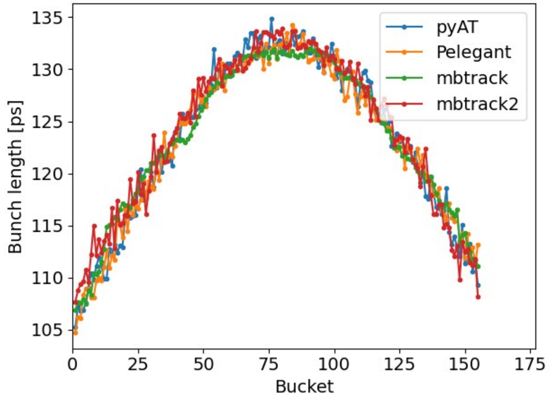
173 bunches





155 bunches





Results agree between all codes, but mbtrack2 results are slightly different and a bit more noisy

Next step to compare results with beam loading in MC and then with active HC

HZB BESSY II Light Source

EXECUTION TIME

- For machines with many bunches execution time is crucial.
- Also important if one wants to include a lot of effects for a self-consistent simulation.
- No optimisation at all done → only followed the installation instructions provided in documentation.
- For codes with builds that required root permissions or dependencies not existing on the cluster containers were used → this can effect the execution time.

10 000 particles per bunch, 176 bunches, 150 000 turns

10 000 particles, 176 bunches, 150 000 turns	**pyAT**	**Pelegant (container)**	**mbtrack**	**mbtrack2 (container)**
1 node	26.1 min	4.19 h	25.8 min	Not able to run because of oversubscribing
2 nodes	20.9 min	> 10 h	> 10 h	Not able to run because of oversubscribing
3 nodes	15.6 min	> 10 h	7.7 h	Not able to run because of oversubscribing
4 nodes				47.5 min

DISCUSSION TOPICS

BEAM DYNAMICS MODEL

- All codes do one-turn map = linear matrix + classic radiation + quantum diffusion + non-linear effects.
- PyAT: self-consistent model → calculated directly from the lattice.
- Pelegant:
  - Self-consistent for uncoupled lattice
  - Radiation model can cause problems with orbit and equilibrium emittance.
- mbtrack/mbtrack2:
  - Separation of transverse/longitudinal → not symplectic except for uncoupled lattice without dispersion at the observation point.
  - mbtrack2 uses average optics functions in one-turn map when importing lattice from pyAT → Does this give correct physics?
  - Radiation model can cause problems with orbit and equilibrium emittance.

- For purely longitudinal studies of HCs this is not a problem, but we also want to study the effect of HCs on the transverse plane → **when are these models not valid anymore?**

CAVITY CONVENTIONS

- Conventions for longitudinal plane is a mess…

- Two cavity conventions are most common:
  - Sine wave $\rightarrow$ for cavities without beam loading $V \sin(\omega_{RF} t + \phi)$
  - Cos wave $\rightarrow$ for cavities with beam loading (phasors) $V \cos(\omega_{RF} t + \psi)$

- AT/pyAT use a different one: $-V \sin(\omega_{RF}(t - TimeLag/c))$

Parameter is called TimeLag, but it is actually a lag in cτ and not time.

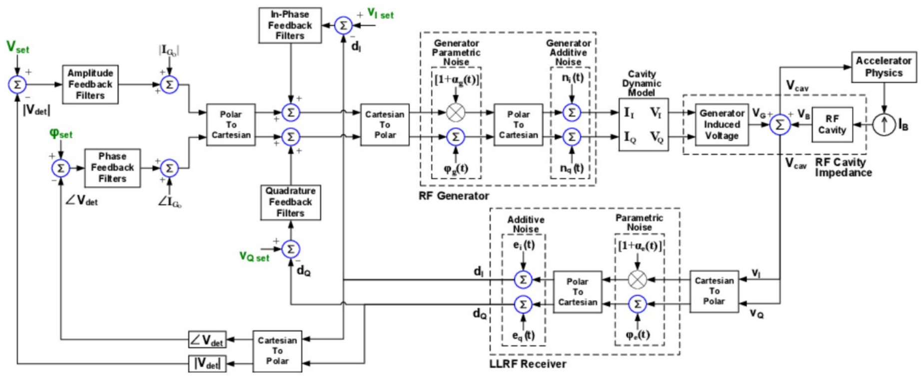
- This convention causes a lot of headache and wasted time $\rightarrow$ how easy will it be to implement a full model of the RF feedback in this convention?

- Is this the time to change it to avoid future problems?

RF FEEDBACK

- For MCs and active HCs RF feedback must be included to keep RF voltage constant with beam loading.
- Two different type of feedback implementation exists:
  - “Compensation” scheme: from beam loading directly calculate required generator phasor
  - Model of feedback system → requires filter coefficients

Feedback model in Pelegant



- The two implementations are useful for different user cases:
  - Studies of settings and behaviour at equilibrium.
  - Behaviour of RF feedback, effect of feedback on instability thresholds, effect of RF noise etc.
- One parameter is common: averaging length/update interval → not numerically robust = difficult to find good settings and machine/simulation dependent.

HZB BESSY II Light Source

CONCLUSIONS

- pyAT has huge potential for being able to simulate all collective effects with a fully self-consistent single particle dynamics model.
- pyAT allows for full flexibility $\rightarrow$ code can easily be modified and extended if required.

SUGGESTIONS FOR THE FUTURE

- pyAT should be separated from Matlab AT to allow for full use of Python functionality and avoid legacy issues, e.g. modernisation of the integrators should be discussed.
- Two options for the RF feedback should be implemented to cover all user cases.
- We should come together and start to join tools: middlelayer, simulators, virtual accelerators, digital twins.
- We need to start to value our codes $\rightarrow$ strategy & resources for long-term maintenance and support.
- We should build a user community around pyAT $\rightarrow$ user meetings, user forum etc.
- Build connections with the supercomputing community to optimise the parallel performance of our codes.

Thank you to MAX IV for allowing me to borrow their cluster resources