TESTS OF COMMISSIONING SIMULATION TOOLS IN OPERATION OF STORAGE RING LIGHT SOURCES

K. Paraschou*, I. Agapov, E. Musa, Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany
T. Hellert, Lawrence Berkeley National Laboratory, Berkeley, CA, USA
N. Carmignani, L. Carver, S. Liuzzo, J.-L. Pons, S. White
European Synchrotron Radiation Facility, Grenoble, France

Abstract

The design of 4th-generation synchrotron light sources employs such strong quadrupolar and sextupolar magnets that the motion of electrons in the storage ring is extremely sensitive to the misalignment and the field imperfections of magnets. As a result, it has become common practice to simulate the steps followed during the commissioning of a light source to compare the sensitivity to different types of lattice errors from a beam dynamics point of view and inform the design of the light source. Although there is significant overlap in the algorithms used to commission and to simulate the commissioning of a light source, they typically consist of separate implementations. This work describes the latest developments in the pySC software for commissioning simulations to allow the use of its tools in a real control system environment for storage rings. Finally, it reports on tests done at the European Synchrotron Radiation Facility Extremely Brilliant Source (ESRF-EBS) to apply these tools to tune the storage ring.

INTRODUCTION

To achieve small emittances, 4th-generation storage-ring light sources need strong quadrupole magnets to focus the particle beams to the needed level. This in turn implies that strong sextupole magnets are needed to correct the natural chromaticity that is generated by the quadrupole magnets. The realistic alignment of such magnets will result in strong feed-down effects, and the calibration errors, either in magnets or diagnostic devices, will make a "perfect" correction impossible. It is therefore an important part during the lattice design process to perform commissioning simulations under realistic errors. This can inform the design in the form of what commissioning procedures are required, how strict the specifications must be in terms of alignment, field calibration, field imperfections, beam position monitor specifications, and others. One of the first examples of such studies can be found in Ref. [1].

The Toolkit for Simulated Commissioning was developed as a tool to model lattices with realistic errors and a set of commissioning algorithms that can be used to simulate the commissioning process [2, 3]. It is written in the MATLAB language and uses Accelerator Toolbox (AT) [4, 5] as its particle tracking engine. pySC is a Python implementation and extension of this idea [6], able to use either Accelerator Toolbox (AT) or Xsuite [7] as its tracking engine. The recent development of pySC was motivated by commissioning studies for PETRA IV [8–10], but the same tools are intended to be useful in the control room. This paper describes the recent reorganization of pySC around reusable tuning applications and reports first tests of these applications at the European Synchrotron Radiation Facility Extremely Brilliant Source (ESRF-EBS) [11, 12].

* konstantinos.paraschou@desy.de

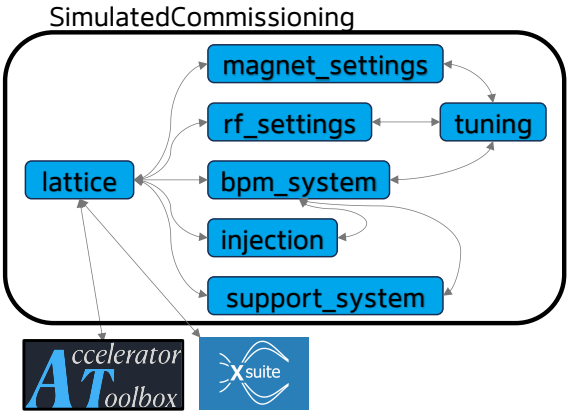

Figure 1: Diagram of the SimulatedCommissioning object.

RECENT pySC DEVELOPMENTS

Earlier pySC development focused on reproducing the simulated commissioning functionality of the original Toolkit for Simulated Commissioning in Python. In the recent development the focus shifted to reorganizing the tuning applications so that they can be reused in real accelerator environments. The reason for this is first and foremost for testing of the algorithms, but also for potential use in operation or in interfacing with other software, such as the Python Accelerator Middle Layer [13]. In order to do that, the pySC software is reorganized to provide a user interface closer to a real control system.

The *SimulatedCommissioning* object is the central state container of a pySC simulation. It is now re-implemented as a structured model that groups the lattice, active and design set-points, support and diagnostic systems, RF and injection state, and the tuning namespace. During initialization, references between these subsystems are propagated, random-number generation is seeded and the stored set-points are sent to the simulated lattice. The same object can also be serialized to JSON, copied, and exposed through a lightweight server. The main objects that make up the SC structure are described below and visualized in Fig. 1. They expose the

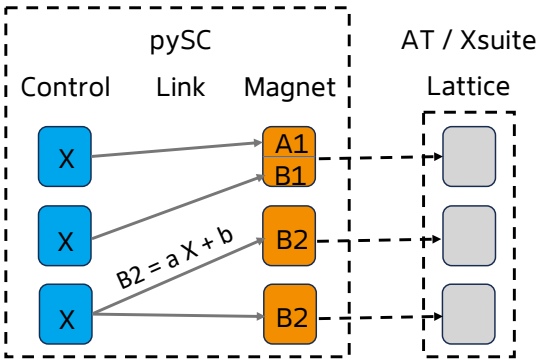

Figure 2: Diagram of the control-link-magnet structure that handles lattice errors and communication with a particle tracking engine.

different subsystems in a way that mirrors the usual organization in accelerator operations.

SC.lattice This object serves as the interface of the *SC* object with the particle tracking engine, which can either be an AT or Xsuite lattice. It provides two lattices, one without any errors applied for calculating model parameters, and the "active" lattice for simulating the commissioning process.

SC.magnet_settings This object manages the settings of magnets through a graph of "controls", "magnets", and "links", an example of which is shown in Fig. 2. A "control" is a variable for the user to change, like the strength of a corrector magnet. A "magnet" stores all multipolar fields associated to a magnet. For example, sextupole magnets in ESRF-EBS have coils to control independently the strength of the normal sextupolar field, the skew quadrupolar field, and the normal and skew dipolar fields. The "link" describes the relation between a "control" setpoint and a field component of a "magnet", including errors. Finally, the information in the "magnet" object is sent to the *SC.lattice* object to be propagated in the particle tracking engine. The point of this object is to expose all "controls" and associate them with names to mimic the behavior in a real accelerator control system environment, where each control variable is associated to a name that can be accessed and manipulated with "get" and "set" functions.

SC.bpm_system This object holds all information about BPMs, including names, noise and calibration errors, alignment errors and reference orbits which are typically produced by beam-based alignment measurements. Through here, the user can request BPM readings of the orbit, or of turn-by-turn data from an injected or kicked particle beam.

SC.support_system Here, the alignment of all elements (and BPM offsets) is handled. It also takes care to propagate the misalignment of supporting structures to individual elements. For example, it handles the propagation of tunnel floor misalignment to displacement of girders and subsequently to the displacement of elements on the girders.

SC.rf_settings Similar to the "magnet_settings" object, the errors of rf cavities are handled here (voltage, frequency and phase), as well as setpoints for each cavity separately.

SC.injection The errors associated to the injection process are managed here, i.e. both systematic and shot-by-shot errors of the injected beam (assumed to be coming from the upstream injector systems in reality).

SC.tuning This object is the entry point to simulate high-level procedures which include:

1. First-turn steering (applicable for the first N turns)
2. Beam-based alignment (BBA) of a Beam Position Monitor (BPM) to a nearby magnet, either based on orbit measurement or on the first turn trajectory.
3. Tune, global coupling, orbit, rf frequency, rf phase, and optics correction.

Finally, the configuration of `pySC` has also been moved toward a file-based description. A YAML file can define the simulator settings, identify magnet families and controls used by the correction routines with regular expressions, and specify numerical values for the different kinds of errors.

REUSABLE TUNING APPLICATIONS

The main recent change is the introduction of reusable applications in `pySC.apps` which implement the procedures of orbit correction, dispersion measurement, orbit response matrix (ORM) measurement, BPM noise measurement and BBA without depending on a specific accelerator backend. These applications rely on the implementation of an "abstract" interface, which must provide the following specific set of functions:

1. "`get_orbit()`", to acquire the orbit read by the BPMs,
2. "`get(name)`", to acquire the set-point (SP) of a control variable with a given name,
3. "`set(name, value)`", to instead set the SP,
4. "`get_many(names_list)`", to acquire many SP,
5. "`set_many(names_values_dict)`", to set many SP,
6. "`get_rf_main_frequency()`", to acquire the SP of the main rf system's frequency, and
7. "`set_rf_main_frequency(value)`", to set it.

In the simulation use case, the `SC.tuning` object instantiates an interface on-the-fly that forwards calls back to the `SC` object. For use in an accelerator control room, the interface must be provided. In practice, control systems that can work with Python, such as EPICS [14], TANGO [15] or DOOCS [16], typically have such functionality already and assembling the interface object required for `pySC.apps` is straightforward.

Single-step algorithms such as orbit correction are implemented as simple function calls. On the other hand, multi-step procedures like the ORM or BBA measurements are implemented as Python generator objects. While it is not usually needed for commissioning simulations, in real accelerator measurements one may have to take into account hysteresis effects of magnets or long-term drifts. For example, with the use of generators, during the measurement of

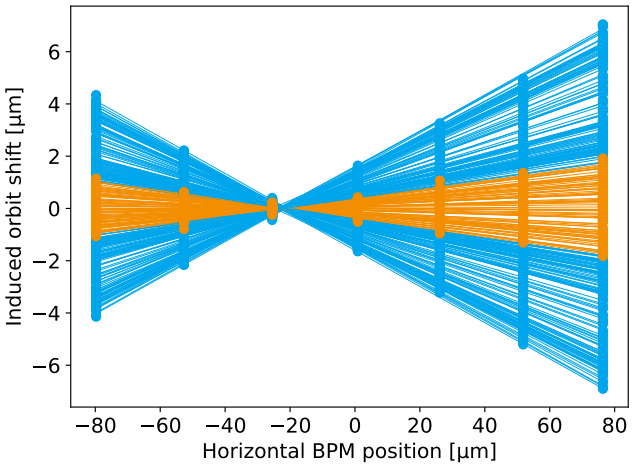

Figure 3: Beam-based alignment measurement at ESRF-EBS with accepted (blue) and rejected (orange) points.

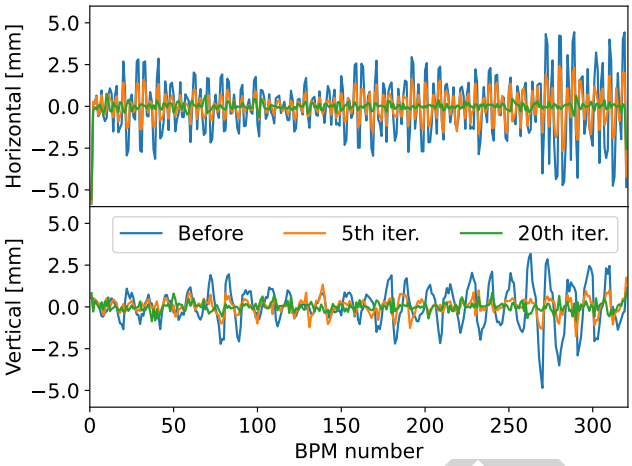

Figure 4: First-turn steering at ESRF-EBS.

an ORM, the user can correct unwanted or unexpected orbit changes.

As part of the applications, `pySC.apps` provides a `ResponseMatrix` object mainly used for the orbit correction algorithms but can also be reused for any other similar least-squares problem. In the case of orbit correction, it supports BPM and corrector weights and calculates the pseudo-inverse matrix through either the truncated singular value decomposition or Tikhonov regularization. Additionally, it can also constrain the sum of the corrections or include the response of the rf system's frequency to the orbit. Finally, it also supports correction with the MICADO algorithm [17].

TESTS AT ESRF-EBS

The applications were tested at ESRF-EBS during machine development time in November 2025 and January 2026. All tests used an interface (as specified above) to the Tango control system of ESRF-EBS, to access the integrated normalized strength of the magnets.

Closed orbit correction In the first set of tests, the algorithms of orbit correction and beam-based alignment were tested. The orbit correction used an ORM pre-computed with pySC and the ideal ESRF-EBS model in AT. It was loaded from a file and the names of the model controls were mapped to the corresponding Tango device names. The reference orbit was read from the control system and the pySC application was then used to calculate and apply corrections. The orbit was corrected to a satisfactory level using any of the pseudo-inversion methods listed before.

BPM to Quadrupole BBA The classic BBA algorithm [18] is implemented and an example of a measurement at ESRF-EBS is visible in Fig. 3. The estimated offset between the BPM and the nearby quadrupole was found to agree with the result of an independent measurement using the operational tools of ESRF-EBS.

First-turn steering In a second set of tests, all magnets were set to their design strengths and the rf cavities were turned off, to bring the machine to a state similar to that during a restart of the storage ring. It is important to note that the state is still not comparable to that during the first commissioning, as all magnets are known to respond properly and the beam-based alignment corrections were not removed for the sake of time. Trajectory correction was then performed using `pySC` and a model response matrix computed with the model lattice, as was done for the case of the closed orbit correction. The correction converged within 20 iterations, as also shown with recorded trajectories in Fig. 4. After convergence, the rf cavities were switched on. Beam was stored only after the betatron tunes were scanned to avoid low order resonances.

ORM and dispersion measurement After beam accumulation was possible, the next step of the tests was to correct the linear optics using the pyLOCO [19] code, which is a Python implementation of the LOCO [20] method for optics correction. Here, `pySC` was used to measure the ORM, the dispersion function and the noise of BPMs as input for pyLOCO. It was known that, due to hysteresis in the corrector magnets, the orbit needs to be corrected with the same corrector whose response was being measured after each step. The flexibility provided by the implementation of the measurements as Python generators made it trivial to customize the measurement without needing to modify the source code of `pySC`. The analysis of the measurements are not discussed here but can be found in Ref. [19].

CONCLUSION

The `pySC` software was extended with an interface that mimics a real control system. This enabled the conversion of tools used in simulated commissioning to applications that can either be used in real operation or commissioning, or interfaced to other applications. Care was taken to ensure that these applications can be further customized by the user for operations that may need to be done in real operation but would otherwise be irrelevant in simulations. The basic ingredients needed for commissioning a storage ring were made into such applications and were subsequently tested successfully during machine development time at ESRF-EBS. The development of `pySC` will continue and include more applications and tests in real storage rings.

REFERENCES

[1] V. Sajaev, "Commissioning simulations for the argonne advanced photon source upgrade lattice", *Phys. Rev. Accel. Beams*, vol. 22, p. 040102, 2019. doi:10.1103/PhysRevAccelBeams.22.040102

[2] T. Hellert, P. Amstutz, C. Steier, and M. Venturini, "Toolkit for simulated commissioning of storage-ring light sources and application to the advanced light source upgrade accumulator", *Phys. Rev. Accel. Beams*, vol. 22, p. 100702, 2019. doi:10.1103/PhysRevAccelBeams.22.100702

[3] T. Hellert, C. Steier, and M. Venturini, "Lattice correction and commissioning simulation of the advanced light source upgrade storage ring", *Phys. Rev. Accel. Beams*, vol. 25, p. 110701, 2022. doi:10.1103/PhysRevAccelBeams.25.110701

[4] A. Terebilo, "Accelerator toolbox for matlab", Stanford, CA, USA, SLAC National Accelerator Laboratory, SLAC-PUB-8732, 2001. doi:10.2172/784910

[5] W. A. H. Rogers, N. Carmignani, L. Farvacque, and B. Nash, "pyAT: A Python Build of Accelerator Toolbox", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017, pp. 3855–3857. doi:10.18429/JACoW-IPAC2017-THPAB060

[6] L. Malina *et al.*, "Python library for simulated commissioning of storage-ring accelerators", in *Proc. ICALEPCS'23*, Cape Town, South Africa, Oct. 2023, pp. 1637–1642. doi:10.18429/JACoW-ICALEPCS2023-FR2AO05

[7] G. Iadarola *et al.*, "Xsuite: An Integrated Beam Physics Simulation Framework", in *Proc. HB'23*, Geneva, Switzerland, Oct. 2023, pp. 73–80. doi:10.18429/JACoW-HB2023-TUA2I1

[8] C. G. Schroer *et al.*, "PETRA IV: upgrade of PETRA III to the ultimate 3d x-ray microscope. conceptual design report", Hamburg, Germany, Deutsches Elektronen-Synchrotron DESY, 2019. doi:10.3204/PUBDB-2019-03613

[9] I. Agapov *et al.*, "Beam dynamics performance of the proposed PETRA IV storage ring", Aug. 2024. doi:10.48550/arXiv.2408.07995

[10] R. Bartolini *et al.*, "Status of the PETRA IV accelerators upgrade", presented at IPAC'26, Deauville, France, May 2026, paper THV2001, this conference,

[11] P. Raimondi *et al.*, "Commissioning of the hybrid multibend achromat lattice at the european synchrotron radiation facility", *Phys. Rev. Accel. Beams*, vol. 24, p. 110701, 2021. doi:10.1103/PhysRevAccelBeams.24.110701

[12] P. Raimondi *et al.*, "The extremely brilliant source storage ring of the european synchrotron radiation facility", *Commun. Phys.*, vol. 6, p. 82, 2023. doi:10.1038/s42005-023-01195-z

[13] Python Accelerator Middle Layer developers, "Python accelerator middle layer", 2026. `https://pyaml.readthedocs.io/en/latest`

[14] L. R. Dalesio *et al.*, "The experimental physics and industrial control system architecture: past, present, and future", *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 352, no. 1–2, pp. 179–184, 1994. doi:10.1016/0168-9002(94)91493-1

[15] A. Götz *et al.*, "The Tango Controls Collaboration Status in 2021", in *Proc. ICALEPCS'21*, Shanghai, China, Aug. 2021, pp. 544–549. doi:10.18429/JACoW-ICALEPCS2021-WEAR01

[16] L. Fröhlich *et al.*, "The Evolution of the DOOCS C++ Code Base", in *Proc. ICALEPCS'21*, Shanghai, China, Aug. 2021, pp. 188–192. doi:10.18429/JACoW-ICALEPCS2021-MOPV027

[17] B. Autin and Y. Marti, "Closed-orbit correction of A.G. machines using a small number of magnets", CERN, Geneva, Switzerland, CERN Note CERN-ISR-MA-73-17, 1973. doi:10.17181/CERN-ISR-MA-73-17

[18] X. Huang, *Beam-based correction and optimization for accelerators*. Boca Raton, FL, USA: CRC Press, 2020. doi:10.1201/9780429434358

[19] E. Musa, I. Agapov, J. Keil, S. Liuzzo, and K. Paraschou, "PyLOCO: A Python Framework for Linear Optics Correction in Storage Rings", presented at IPAC'26, Deauville, France, May 2026, paper WEP5011, this conference,

[20] J. Safranek, "Experimental determination of storage ring optics using orbit response measurements", *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 388, no. 1–2, pp. 27–36, 1997. doi:10.1016/S0168-9002(97)00309-4