ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THMBCM018

ADVANCEMENTS IN BEAMLINE DIGITAL TWIN AT BESSYII*

S. Vadilonga[1†], G. Günther[1], S. Kazarski[1], R. Ovsyannikov[1], S. Sachse[1], W. Smith[1]
[1]Helmholtz Zentrum Berlin, Berlin, Germany

Abstract

This presentation reports on the status of beamline digital twins at BESSY II. To provide a comprehensive beamline simulation experience we have leveraged BESSY II's x-ray tracing program, RAY-UI, widely used for beamline design and commissioning and best adapted to the requirements of our soft X-ray source BESSY II. We created a Python API, RayPyNG, capable to convert our library of beamline configuration files produced by RAY-UI into Python objects. This allows to embed beamline simulation into Bluesky, our experimental controls software ecosystem. All optical elements are mapped directly into the Bluesky device abstraction (Ophyd). Thus beamline operators can run simulations and operate real systems by a common interface, allowing to directly compare theory predictions with real-time results. We will discuss the relevance of this digital twin for process tuning in terms of enhanced beamline performance and streamlined operations. We will shortly discuss alternatives to RAY-UI like other software packages and ML/AI surrogate models.

INTRODUCTION

In the synchrotron community, simulations have long been potent tools, primarily utilized during feasibility studies, design, and the commissioning and operation of machines and beamlines. However, the untapped potential of simulations during subsequent commissioning and operational phases remains substantial. Currently, numerous facilities are focusing on digital twins, aiming to integrate these virtual replicas across various domains, including machines, beamlines, and experiments [1, 2]. This paper serves as a conduit to share our journey and outline our upcoming trajectory in the context of beamline digital twins at the BESSY II facility.

RAYPYNG: A MODERN APPROACH TO BEAMLINE SIMULATIONS

At BESSY II, we utilize RAY-UI for simulating the beamlines, which is an X-ray tracer with an integrated graphical user interface [3]. The RAY-UI software is specifically designed for precise simulations of X-ray beamlines, enabling users to model the behavior of X-rays as they interact with optical elements like mirrors, lenses, monochromators, and other components. This simulation capability plays a crucial role in designing and optimizing beamlines, predicting experiment performance, and facilitating the commissioning and operation of the beamlines.

While a graphical user interface is helpful when setting up a beamline for the first time, more advanced simulations necessitate integration into a programming language. We opted for Python due to its extensive user community and user-friendly nature, leading to the development of RayPyNG. RayPyNG reads the beamline configuration files generated by RAY-UI and returns a Python object for each beamline component, along with methods for manipulation. As a subsequent step, we developed a simulation backend API for RAY-UI, offering a straightforward way to interact with RAY-UI through Python, read beamline configuration files, initiate simulations, and export results.

Other simulation engines are certainly possible, and we plan to explore alternatives in the future, such as RAY-X (the successor of RAY-UI, currently under development, see [4]), or machine learning surrogate models. Finally, RayPyNG provides a suite of tools and a straightforward syntax for configuring complex simulations. For more information, please refer to the online documentation. [5].

A BEAMLINE DIGITAL TWIN

At BESSY II, an increasing number of beamlines are embracing Bluesky, a Python library that manages experiment control and scientific data collection, enabling instrument control regardless of software and hardware configurations [6–8]. A key component in this ecosystem is Ophyd, a hardware abstraction layer designed to encapsulate the control layer within a high-level interface. Ophyd enhances efficiency by consolidating individual signals into coherent logical devices.

We have developed a Python library called RayPyNG-bluesky [9], which seamlessly integrates the Python objects representing beamline components created by the RayPyNG library into Ophyd. The prerequisites are minimal: install RAY-UI on the same computer where Bluesky is installed and create a beamline file using RAY-UI. With just two lines of code, you can set up simulations within Bluesky:

```
bl = "my_beamline.rml"
RaypyngOphydDevices(RE=RE, rml_path=bl)
```

Ophyd devices will share the same names as defined in the XML file (see Figure 1), with the prefix *rp_*. For example, if a mirror is named *m1* in the XML file, it will be accessible in Bluesky as *rp_m1*. Behind the scenes, a trigger detector is created and added to the detector list for all plans using simulated devices. This detector manages the initiation of simulations and the distribution of simulation results to the various detectors specified in the Bluesky plan. This approach offers the advantage of preserving the user experience for Bluesky users. For instance, when scanning the translation along the x-axis of a mirror (M1) and checking

* Work funded by BMBF and Land Berlin
† simone.vadilonga@helmholtz-berlin.de

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THMBCM018

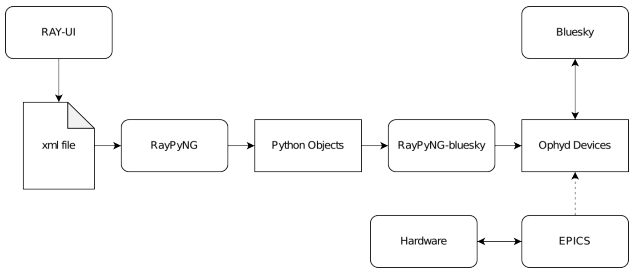

Figure 1: Workflow used to create a digital twin: The beamline and its components are configured in RAY-UI and saved as an *XML* file. RayPyNG reads this file and generates Python objects for each element in the beamline. These Python objects are used to create Ophyd devices, which can then be employed by Bluesky to perform simulations. In the near future, it will be possible to update the values of simulated Ophyd devices to reflect the current beamline status.

the X-ray intensity after passing through the same mirror using detectorM1, provided that both the real and simulated hardware are correctly configured, the process is as follows:

```
# scan using real hardware
RE(scan[detM1],M1.tx,-1,1,20))

# scan using simulated hardware
RE(scan[rp_detM1],rp_M1.tx,-1,1,20))
```

OUTLOOK

We are committed to advancing our initiatives on both the hardware and software fronts. As part of a comprehensive network restructuring effort, we will relocate our simulations to a centralized server. Currently, these simulations are conducted on the beamline computer—an approach fraught with potential issues. The risk of encountering computational limitations leading to operational bottlenecks necessitates a shift towards centralization. By consolidating computational resources onto a robust central server, we anticipate a marked improvement in operational efficiency.

Our simulation engine, RAI-UI, has the capability to compute diverse X-ray sources. Regarding undulators, RAI-UI employs a streamlined model. However, for higher precision, it draws upon the output files generated by another program, WAVE, predominantly coded in Fortran [10]. Recently, it started the development of WavePyNG, a Python API designed to interface with WAVE. This strategic move towards integration within the digital twin framework holds the promise of on-the-fly simulations. The dependency on pre-established undulator simulations with fixed parameters could thus be circumvented.

Simultaneously, our machine-learning group is working on a surrogate model for RAI-UI. Leveraging machine learning methods, the objective is to replace RAI-UI simulations with those derived from the surrogate model. The intrinsic flexibility of integrating alternative simulation engines into RayPyNG ensures a straightforward transition. Notably, this would result in simulations executed in milliseconds, a substantial leap from the existing tens-of-seconds timeframe.

Presently, the digital twin lacks awareness of the precise positions of beamline element motors. Manual alignment of these positions is required to correlate with their virtual counterparts. To address this limitation, we are actively devising the requisite code to facilitate the synchronization of simulated motor positions with their real-world counterparts. Overcoming the challenge of accurately mapping nominal beamline motor positions to their real positions constitutes a multifaceted endeavor. Our machine-learning group is actively engaged in tackling this objective.

ACKNOWLEDGEMENTS

The authors thank the Bluesky developers and the Bluesky community for their continuous support. Additionally, special thanks are extended to Stefan Schäfers (HZB) for his dedicated efforts in seamlessly integrating WavePyNG into the digital twin project.

REFERENCES

[1] L. Rebuffi *et al.*, "Advanced optical simulation tools in the OASYS suite and their applications to the design of last generation synchrotron radiation beamlines", in *J. Phys.: Conf. Ser.*, vol. 2380, 2022, p. 012065. doi:10.1088/1742-6596/2380/1/012065

[2] M. Rakitin *et al.*, "Introduction of the Sirepo-Bluesky interface and its application to the optimization problems", in *Proc. SPIE 11493, Advances in Computational Methods for X-Ray Optics V*, 2020, p. 1149311. doi:10.1117/12.2569000

[3] P. Baumgärtel *et al.*, "RAY-UI: New features and extensions", in *AIP Conference Proceedings*, 2019, p. 060034. doi:10.1063/1.5084665

[4] https://github.com/hz-b/rayx

[5] https://raypyng.readthedocs.io/

[6] R. Müller *et al.*, "Modernization of Experimental Data Taking at BESSY II", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 65–69. doi:10.18429/JACoW-ICALEPCS2019-MOCPL02

[7] R. Müller *et al.*, "Experimental Data Taking and Management at BESSY II and HZB: The Upgrade Process", presented at ICALEPCS'23, Cape Town, South Africa, 2023, paper MO2AO04, this conference.

[8] D. Allan, T. Caswell, S. Campbell, and M. Rakitin, "Bluesky's Ahead: A Multi-Facility Collaboration for an a la Carte Software Project for Data Acquisition and Management", *Synchrotron Radiat. News*, vol. 32, no. 3, pp. 19–22, 2019. doi:10.1080/08940886.2019.1608121

[9] https://raypyng-bluesky.readthedocs.io/

[10] M. Scheer, "WAVE - A Computer Code for the Tracking of Electrons through Magnetic Fields and the Calculation of Spontaneous Synchrotron Radiation", in *Proc. ICAP'12*, Rostock-Warnemunde, Germany, Aug. 2012, paper TUACC2, pp. 86–88.