ISBN: 978-3-95450-248-6 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2025-THPS053

INTEGRATING COMMUNITY CODES FOR ACCELERATOR DESIGN AND OPTIMIZATION*

N.M. Cook †, C. Hall, J. Edelen, E. Carlin, P. Moeller, R. Nagler, D. Bruhwiler
RadiaSoft, Boulder, Colorado, USA
A. Huebl, C. Mitchell
Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Abstract

Advances in fidelity and performance of accelerator modeling tools, in tandem with novel machine learning capabilities, have prompted community initiatives aiming to realize "virtual test stands" that can serve as true analogues to physical machines. Such efforts require integrated, end-to-end modeling capabilities with support for parametric optimization and benchmarking. We present the ongoing development of an integrated Sirepo application to support the holistic modeling of accelerators. Our approach leverages existing modeling workflows, such as the Light Source Unified Modeling Environment (LUME), as well as community I/O frameworks, such as openPMD, to provide a toolbox for constructing and modeling beamlines. Users can build and test simulations using different community modeling tools, as well as connect individual tools to produce end-to-end simulations. We discuss some specific beamline modeling demonstrations as well as ongoing efforts to support code-agnostic design and development.

INTRODUCTION

Computer modeling is essential to advancing particle accelerator design and operation. Advances in the scale and sophistication of accelerator facilities have merited commensurate advances in modeling complexity. State-of-the-art modeling tools now permit the inclusion of multi-scale modeling techniques, machine learning enhanced algorithms, and execution across heterogeneous architectures up to and including exascale supercomputers. New community efforts are aiming to realize digital twins that can serve as true analogues to the physical machine, necessitating additional inputs to characterize features unique to specific facility implementations.

This paper outlines the development of a prototype platform for interchangeable modeling of accelerator beamlines based on the Sirepo platform [1], featuring support for the specification, execution, and analysis of beamline simulations using MAD-X, `elegant`, and ImpactX. Demonstrations of these features were made using touchstone facility models, including the IOTA lattice and PIP-II medium energy beam transport (MEBT) section at Fermilab as well as a ring-based demonstration via the BESSY-II synchrotron.

A SHARED BEAMLINE EDITOR

We first generated a Sirepo-based interface that enables code-agnostic beamline editing. Our approach was to identify a common set of shared elements and create a simple schema to represent lattice objects in a code-agnostic format. The resulting schema[1] is leveraged specifically by the Canvas app when building lattices in the editor.

Figures 1, 2, and 3 depict the lattice editor developed during this work, which shows the IOTA ring at Fermliab, and the available lattice elements that can be used to modify an accelerator design. As with other Sirepo editors, and to maintain consistency with community design practices, elements can be grouped recursively into lines, and subsequently joined to create a full accelerator design. The interface permits individual element modifications, along with modifications to lines through manual re-arrangement or via drag-and-drop capabilities within the beamline editor.

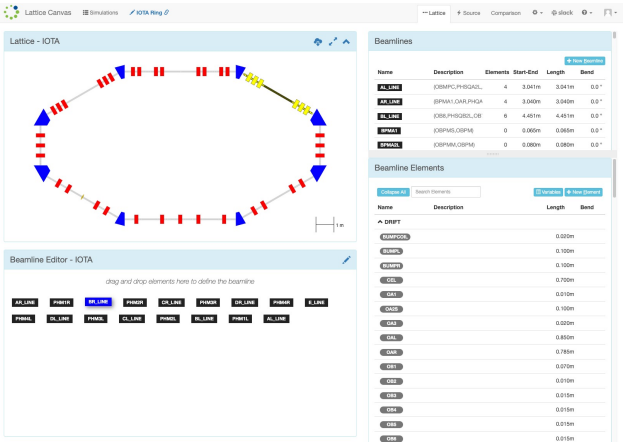

Figure 1: Canvas application illustrating the lattice editor panel for the IOTA storage ring at Fermilab.

For each shared element, a subset of the total possible attributes of that element are preserved by the shared Canvas schema, so as to avoid inconsistencies or loss of information during exchange with different codes. Figure 3 depicts attributes for a quadrupole magnet that can be defined and shared to create the element model.

PARTICLE DISTRIBUTION EXCHANGE

While much of the accelerator lattice description may be an independent representation many accelerator simulations

* Work supported by the U.S. D.O.E.,Office of Science, Office of High Energy Physics under Award Number(s) DE-SC0024814.
† ncook@radiasoft.net

[1] `https://github.com/radiasoft/sirepo/blob/master/sirepo/package_data/static/json/canvas-schema.json`

ISBN: 978-3-95450-248-6 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2025-THPS053

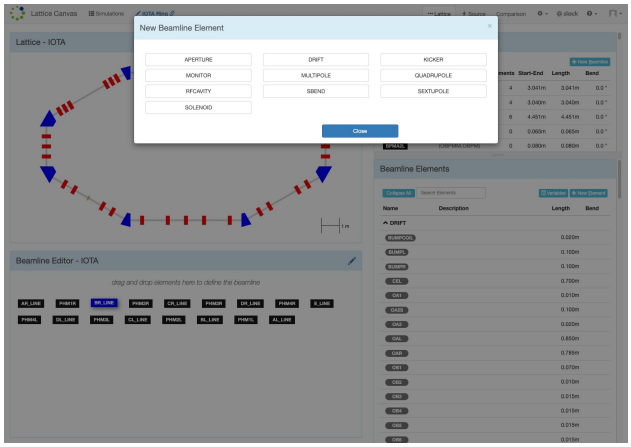

Figure 2: The Sirepo Canvas application supports the design of lattices contained a subset of common elements shared across `elegant`,ImpactX, and MadX.

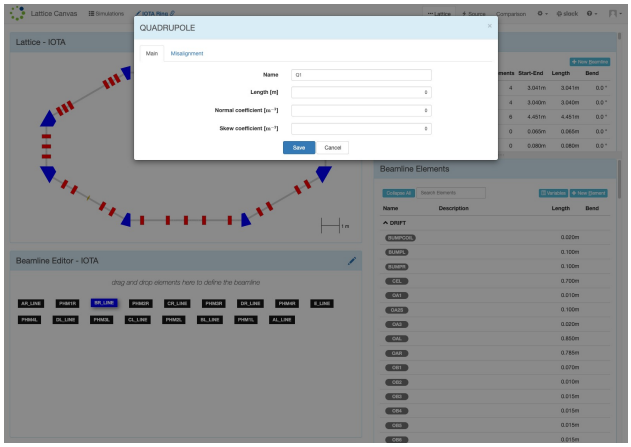

Figure 3: Within each element type, a subset of common attributes are supported to enable appropriate exchange between the executable codes.

serve primarily as particle tracking programs and require a particle distribution description. Therefore it is essential that the exchange of particle data information is facilitated alongside the lattice exchange. At least one ongoing project to standardize beam information exists in the openPMD [2]-BeamPhysics standard[2]. We are primarily concerned with the exchange of particle data between codes, however, so while the openPMD standard is supported, and used internally, in many cases for data transfer, it is not a substitution for particle data exchange.

For the particle data exchange we have developed a `SwitchYard` module — implemented in Python — to read, write, and store particle data from a variety of codes. The core `SwitchYard` maintains a set of readers and writers for all known codes and stores particle data in an internal format. Maintaining an internal storage format for the data exchange is essential so that for N supported codes we need only develop $2N$ reader and writers, not N^2 transformations.

[2] `https://github.com/openPMD/openPMD-standard/blob/upcoming-2.0.0/EXT_BeamPhysics.md`

However, this is only used as an in-memory presentation. For cases where the particle data should be stored and a particular code is not requested we default to using the openPMD standard. For reading and writing formatted files, where available we make use of existing libraries. For the common SDDS standard we prefer the use of pysdds[3] a purely-Python implementation of SDDS.

COMPARATIVE EXECUTION

Our existing beam specification API has been extended to support file-based import of particle information, capturing relevant details across multiple formats. As discussed in the Task 2 accomplishments, the resulting tool leverages two community standards: the SDDS file format common to `elegant`, and the openPMD standard in use by many other community codes.

New particle distributions can be generated in memory via a Python-based API that creates beam distributions following prescriptions outlined in the ImpactX code. These include many standard distributions, including Gaussian, K-V, Waterbag, and thermal distributions, among others. To guide the distribution process, the API extracts the relevant Courant-Synder parameters (when possible) at the starting location of the lattice, and supplies those to enable automated matching of the distribution. Figure 4 shows an example of generating a new Gaussian distribution to the match a lattice.

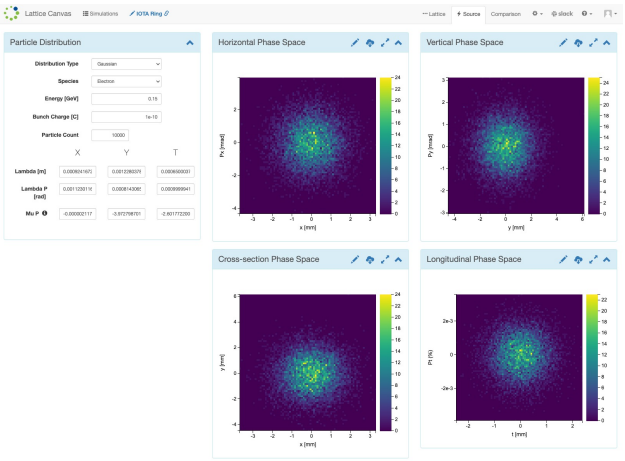

Figure 4: Depiction of the Canvas source generation tab depicting the creation of a new Gaussian particle distribution matched to the lattice at the s=0 injection point.

The interface supports execution across three different codes from the shared lattice. Our strategy leverages existing Sirepo support for containerized environment including common accelerator codes and dependencies, which have been extended to include ImpactX support. Configuration of the lattice and source produce external files, which are then coupled to their corresponding executables and run in the pre-configured environment in a separate section of the interface.

[3] `https://github.com/nikitakuklev/pysdds`

ISBN: 978-3-95450-248-6 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2025-THPS053

We then developed a set of common diagnostics that provide holistic beamline analyses in tandem with beam-specific visualizations. Each diagnostic supports (optional) comparisons where possible.

Figure 5 depicts a TWISS diagnostic displaying linearized values of the Courant-Synder parameters β_x and β_y compared across all three codes. In a similar vein, statistical beam parameters σ_x and σ_y are also provided in a corresponding diagnostic. Figure 5 displays this diagnostic for `elegant` and ImpactX simulations. These composite visualizations provide a full s-dependence in a single plot.

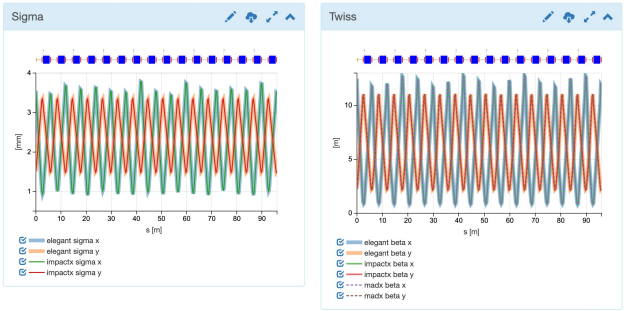

Figure 5: Snapshot of the lattice TWISS and beam sigma diagnostics from a simulation of the BESSY-II synchrotron for `elegant`, MAD-X, and ImpactX. Good agreement is seen in the $\beta_{x,y}$ parameters for all codes while $\sigma_{x,y}$ are in agreement for `elegant` and ImpactX simulations.

Additional phase space projections are computed and displayed for the user, providing more detailed snapshots of the beam at varying points along the lattice. Two-dimensional projections of the horizontal, vertical, and longitudinal phase space, as well as the X-Y cross-sectional projection can be produced. The phase space plots are designed to be output at the beginning and end of the lattice by default; users can add additional diagnostic locations by adding a "MARKER" element to the lattice, which are then accessible in the list phase space outputs. Users can follow the evolution of each phase space projection for each code, enabling the identification of location-specific changes in the projected performance across models. Figure 6 depicts the horizontal phase space plotted at $s = 0$ (the default location) and $s = 62.85$m (the location of the 6th "MARKER" in the lattice).

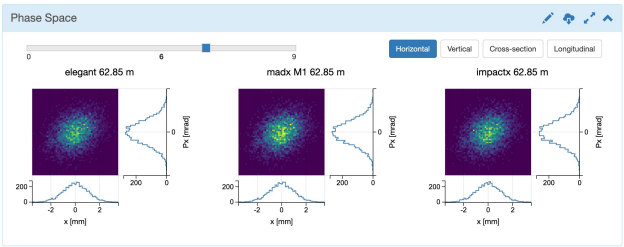

Figure 6: The phase space diagnostic depicting horizontal longitudinal phase spaces from a BESSY-II simulation with `elegant`, MAD-X, and ImpactX.

Figure 7 provides another example of comparative diagnostics from a simulation of the MEBT section of the PIP-II linac.

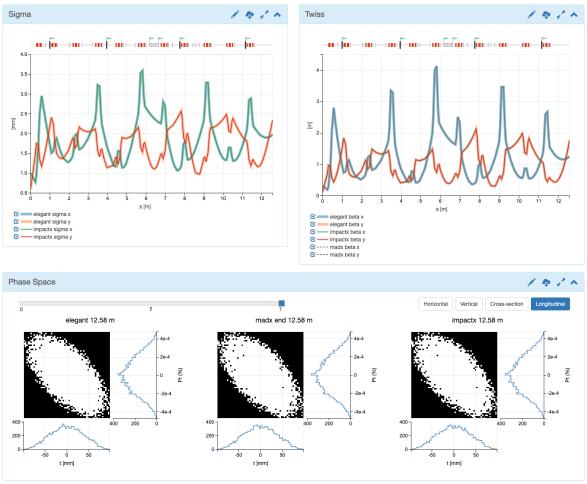

Figure 7: Canvas application displaying diagnostic outputs from a simulation of the MEBT section of the PIP-II linac.

CONCLUSION

We have developed a baseline API and corresponding software interface that supports the design, execution, and comparative analysis of accelerator simulations across multiple community codes, including MAD-X, elegant, and ImpactX. The developed interface supports code-agnostic definition of the lattice, using a shared set of common elements with code-specific translation. The interface supports comparative visualization of lattice parameters, beam properties, and beam phase space cross sections along the length of the accelerator. Shared file structures and input-output operations were adopted to streamline coordination and code-agnostic information exchange.

ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. D.O.E.,Office of Science, Office of High Energy Physics under Award Number(s) DE-SC0024814.

REFERENCES

[1] M. S. Rakitin *et al.*, "*Sirepo*: an open-source cloud-based software interface for X-ray source and optics simulations", *J. Synchrotron Radiat.*, vol. 25, no. 6, pp. 1877–1892, Nov. 2018. doi:10.1107/S1600577518010986

[2] A. Huebl *et al.*, "Openpmd 1.1.0: base paths for mesh- and particle-only files and updated attributes (1.1.0)", *Zenodo*, 2018. doi:10.5281/zenodo.1167843