ISBN: 978-3-95450-253-0 ISSN: 2673-5571 doi:10.18429/JACoW-HB2023-TUA2I2

COMMUNITY MODELING TOOLS FOR HIGH-BRIGHTNESS BEAM PHYSICS

C. Mitchell * †, A. Huebl, J. Qiang, R. Lehe, M. Garten, R. Sandberg, J-L. Vay
Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Abstract

Pushing accelerator technology toward operation with higher intensity hadron beams is critical to meet the needs of future colliders, spallation neutron sources, and neutrino sources. To understand the dynamics of such beams requires a community effort with a comprehensive approach to modeling, from the source to the end of the beam lifetime. One needs efficient numerical models with high spatial resolution and particle statistics, insensitivity to numerical noise, and the ability to resolve low-density halo and particle loss. To meet these challenges, LBNL and collaborators have seeded an open ecosystem of codes, the Beam pLasma & Accelerator Simulation Toolkit (BLAST), that can be combined with each other and with machine learning frameworks to enable integrated start-to-end simulation of accelerator beamlines for accelerator design. Examples of BLAST tools include the PIC codes WarpX and ImpactX. These codes feature GPU acceleration and mesh-refinement, and have openPMD standardized data I/O and a Python interface. We describe these tools and the advantages that open community standards provide to inform the modeling and operation of future high-brightness accelerators.

INTRODUCTION

The modeling of charged-particle beams plays a critical role in accelerator design and operation. The needs of high-intensity and high-brightness hadron facilities pose a special challenge to accelerator beam dynamics modeling, due in part to the broad range of physics effects at interplay. For example, high spatial resolution and good particle statistics are required to model and predict low-density beam halo formation [1–3], to understand intensity-dependent beam loss mechanisms [4], to understand space charge induced emittance growth [5], and to model and mitigate collective instabilities [6–8]. For storage rings and colliders, the modeling of self-consistent collective effects in beams for large turn numbers must address the challenges of simulation artifacts (noise) and long computation times.

* This work was supported by the Director, Office of Science of the U.S. Department of Energy under Contracts No. DE-AC02-05CH11231 and DE-AC02-07CH11359. This material is based upon work supported by the CAMPA collaboration, a project of the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of High Energy Physics, Scientific Discovery through Advanced Computing (SciDAC) program. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award HEP-ERCAP0023719.
† ChadMitchell@lbl.gov

To address these challenges effectively, a community approach is required. In addition to sharing expertise regarding physics and algorithm development, collaborative code development based on shared standards and common code interfaces can enhance existing workflows. For example, in start-to-end modeling of complex multi-stage accelerator facilities, one typically must chain multiple simulation codes, involving different numerical models, input and output formats, and user expertise. In addition, one needs reliable methods to benchmark, exchange data, and compare results from multiple simulation codes.

To model intense beams with improved fidelity over the entire beam lifetime with reasonable computing times, to enable large ensembles of simulation runs for optimization and for training of machine learning (ML) models, and to prepare for future Exascale computing systems, software must be developed and/or modernized to take advantage of state-of-the-art computer hardware, including GPUs. Examples of collaborative multi-institute code development efforts include the Collaboration for Advanced Modeling of Particle Accelerators (CAMPA) [9], the Exascale Computing project [10, 11], and HEP SciDAC-5 [12].

GOALS OF A COMMUNITY ECOSYSTEM

One primary community goal is to understand and exploit the physics of high intensity beams for improved accelerator design, using full-physics 6-D computer simulations of entire accelerator systems that incorporate all components (including any conventional and advanced concepts sections) and all pertinent physical effects and that execute quickly and reliably (“end-to-end virtual accelerators", EVAs) [13]. Furthermore, these EVAs should be able to leverage modern computing infrastructure such as HPC clusters and GPU computing and fully integrate AI/ML tools to maximize efficiency for practical applications.

One approach to this goal is through an ecosystem of interoperable modeling tools with various levels of integration and fidelity. Figure 1 illustrates a proposed example of such an ecosystem [14]. Individual components are represented as boxes with dependencies building on top of each other. The fundamental building blocks are implementations of solvers/numerical schemes, efficient I/O, parallelization, and a performance portability layer (*Math & Computer Science libraries*). Depending on this functionality, domain-specific libraries for RF, beam, or plasma modeling can be implemented (*Accelerator & Beam Toolkits*) that then act as toolkits to be combined into concrete applications. Compatibility, development productivity, usage, and quality assurance

ISBN: 978-3-95450-253-0 ISSN: 2673-5571 doi:10.18429/JACoW-HB2023-TUA2I2

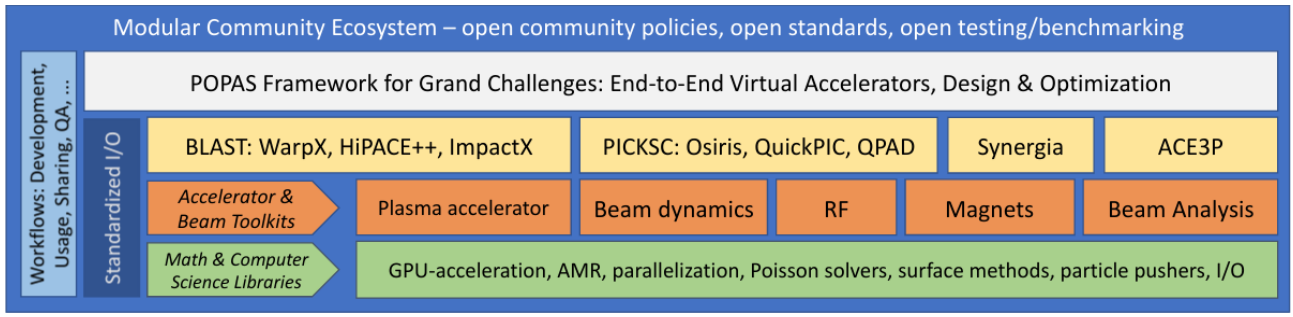

Figure 1: Diagram of a possible envisioned *ecosystem* for particle beam and accelerator modeling. Vertical components are software dependencies, which in most cases are software libraries. Open policies, standards, standardized I/O, and common workflows steer and connect the ecosystem. Reproduced from Ref. [14].

can be coordinated with standardization (e.g., for I/O and data layouts) as well as common workflows (e.g., shared continuous benchmarks and practices). The top level represents an overarching framework for optimization workflows (e.g., CAMPA's Platform for Optimization of Particle Accelerators at Scale aka POPAS) that combine application components as needed for the study of complex integrated research questions.

The Role of Standards

The goals just described are often challenging to implement in practice because of differing conventions among beam dynamics codes, variations in file format, and differences in implementation details. Standardization is a way to solve such incompatibilities and greatly simplify vertical software integration [15]. Examples of particle accelerator community standards for data exchange and common inputs include the Open Standard for Particle-Mesh Data (openPMD) [16] and the Particle-In-Cell Modeling Interface (PICMI) [17]. Discussions are ongoing regarding an Accelerator Modeling Interface (AMI) standard for lattice description. In the community ecosystem of Fig. 1, these standards are implemented in a unified Python application interface layer and will be leveraged for activities concerning unified simulation input and control.

openPMD is a metadata standard for scalable I/O and exchange of particle and mesh-based data, implemented on popular scientific file formats such as HDF5 and ADIOS. The openPMD standard is developed as an open community project and released in versions (current version 1.1.0). Included are data-processing [18] and visualization frameworks such as openPMD-viewer [19], and ParaView/VisIt.

PICMI & AMI address the challenge of a unified simulation design by defining a Particle-In-Cell Modeling Interface (PICMI) standard and an Accelerator Modeling Interface (AMI) standard (built as an extension of PICMI for accelerators) that establish conventions for the naming and structuring of input files for, respectively, PIC simulations at large and PIC-based accelerator simulations. The goal of the standards is to propose a set (or dictionary) of standardized names and definitions that can be used in input scripts, with as few changes as possible between codes.

BEAM, PLASMA & ACCELERATOR SIMULATION TOOLKIT

One example of such an integrated ecosystem approach is given by the Beam, Plasma & Accelerator Simulation Toolkit (`BLAST` [20]), a suite of open source particle accelerator modeling codes. This suite includes the particle-in-cell codes `WarpX` and `ImpactX`, codes for modeling beam-beam and electron cloud effects (BeamBeam3D, POSINST) and codes for modeling plasmas and advanced accelerator concepts (FBPIC, HiPACE++, Wake-T). Originally developed under the name Berkeley Lab Accelerator Simulation Toolkit, `BLAST` was renamed in 2021 to reflect international contributions from LIDYL (CEA, France), SLAC (USA), LLNL (USA), DESY (Germany), UHH (Germany), HZDR (Germany), Radiasoft (USA), CERN (Switzerland) and more; `BLAST` development involves deep collaboration among physicists, applied mathematicians and computer scientists.

With the emergence of the first Exascale Computing supercomputers, modeling codes that were originally designed for parallel CPU-powered machines need to undergo a fundamental modernization effort. This became necessary, as compute nodes are now equipped with accelerator hardware such as GPUs (and potentially FPGAs in the future). Selected as application for the Department of Energy Exascale Computing Project, the `BLAST` code `WARP` [21, 22] underwent a complete rewrite from Fortran to modern C++ resulting in its successor `WarpX` [23]. Building on the momentum of this transition to form a more cohesive Accelerator Toolkit, the specialized plasma wakefield acceleration code `HiPACE++` [24] and beam dynamics code `ImpactX` [25, 26] have recently undergone similar modernization.

Software Design

A central goal of the modernization of `BLAST` is modularity for efficient code reuse and tight integration for coupling, i.e., in hybrid particle accelerators with

ISBN: 978-3-95450-253-0 ISSN: 2673-5571 doi:10.18429/JACoW-HB2023-TUA2I2

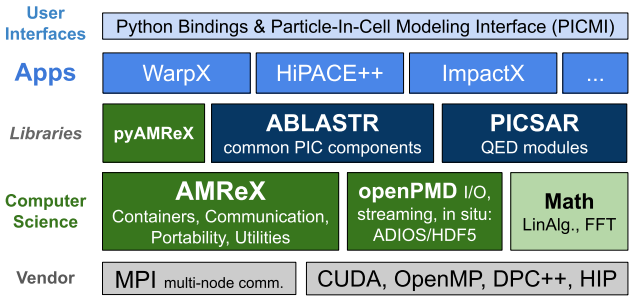

Figure 2: Design of the BLAST software stack. Modularization enables code sharing and tight coupling.

conventional and advanced (plasma) elements. Figure 2 shows the design of BLAST's software dependencies, with upper components depending and sharing lower blocks in the schema. Shared code, common application programming interfaces (APIs) and data standards ensure composability and connection to the AI/ML and data science ecosystems. Performance-critical routines are implemented and reused in modern C++, using a single-source approach to program both CPUs and GPUs via a performance-portability layer in AMReX [27]. The newly introduced ABLASTR library collects common particle-in-cell (PIC) routines.

Python high-level interfaces are used for user efficiency and to provide standardized APIs to data science and AI/ML frameworks, which are mostly driven from the same language. Documentation and examples are developed in lock-step with documentation and published on https://impactx.readthedocs.io. Examples and test cases are continuously run against expected results.

All development is carried out in the open using open source licenses, contributable code repositories, code reviews, regular releases and change logs [26]. The community reports open "issues" for feature requests, bug reports, etc. Installation for users and developers is supported by package managers and HPC modules.

IMPACTX

The code IMPACT-Z [28] is widely known in the accelerator modeling community. It has been applied to studies of halo formation and coupling resonance in high intensity beams, studies of the microbunching instability in high brightness electron linacs, beam dynamics in the SNS linac, the JARPC linac, the RIA driver linac, the CERN superconducting linac, the LEDA halo experiment, the Proton Synchrotron at CERN, and others. The code, which is written in Fortran90 with MPI, runs on both single-processor and multi-processor systems [29].

ImpactX [25, 26] is a GPU-capable C++ successor to the code IMPACT-Z [29], built on the AMReX software framework [27], for modeling relativistic charged particle beams in linacs or rings. Leveraging expertise and models in IMPACT-Z [28] and MaryLie [30, 31], this new simulation code is built from the ground up to take GPU-accelerated computing, mesh-refinement for

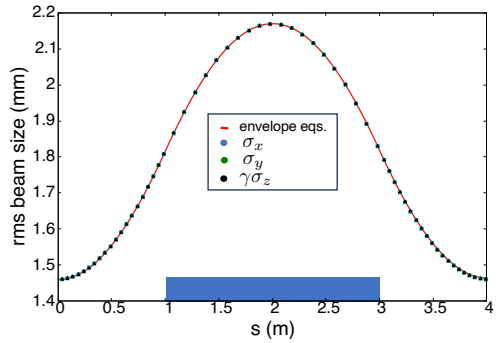

Figure 3: Evolution of the rms beam size $\sigma_x = \sigma_y = \gamma\sigma_z$ over a single period in the focusing channel used for the Kurth beam distribution test. (Line) RMS envelope equations. (Points) Values from ImpactX. The blue rectangle denotes the region of nonzero focusing.

space-charge effects, load balancing, and coupling AI/ML frameworks into consideration. ImpactX design relies on open community standards for I/O and data interfaces. Generalized and reused from WarpX via the ABLASTR library are GPU-accelerated routines for charge deposition, beam statistics, Poisson solve, profiling, warning logging, Unix signal handling, build/installation logic, among others. ImpactX can be executed in two ways: a traditional executable reading a text input file or driven from Python. For an example of the latter usage, see [25].

Model Assumptions

Similar to IMPACT-Z, tracking is performed with respect to the path length variable s, and space charge is included using a second order operator splitting [28]. All tracking methods are symplectic by design, and maps are used where possible for efficient particle pushing. Maps applied during tracking for most thick elements are accurate through linear order (with respect to the reference orbit), with support for soft-edge elements. Exact nonlinear maps are available for some elements (drift, ideal sector bend, regions of uniform field). Symplectic integrators for non-ideal, nonlinear elements are under development.

The code supports 3D space charge for bunched beams. As in IMPACT-Z, the space charge fields are treated as electrostatic in the bunch rest frame. In particular, no retardation or radiation effects are included. Unlike IMPACT-Z, the 3D space-charge fields are computed with an iterative Multi-Level Multi-Grid (MLMG) Poisson

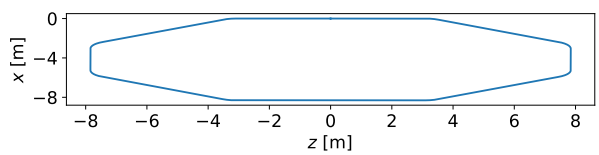

Figure 4: The reference orbit in the IOTA lattice benchmark, as produced by ImpactX, showing the storage ring floorplan.

ISBN: 978-3-95450-253-0 ISSN: 2673-5571 doi:10.18429/JACoW-HB2023-TUA2I2

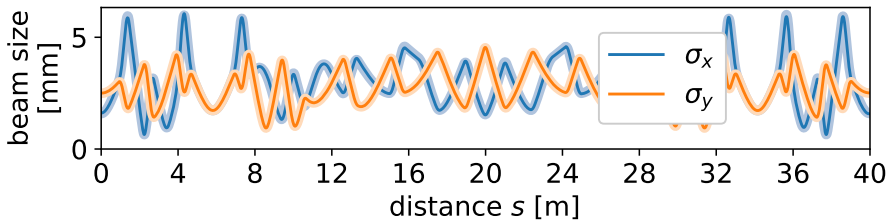

Figure 5: Evolution of the rms beam size in the IOTA lattice benchmark. Thin dark lines: `ImpactX`, light bold lines: `IMPACT-Z` results.

solver [27], providing new support for adaptive mesh refinement. This is expected to allow users to resolve high spatial density gradients within a high-intensity beam (such as sharp beam edges) at reduced computational cost.

Benchmarking and Validation

The code `ImpactX` is continuously benchmarked (after every code change) against a suite of >20 test problems, designed to validate each feature of the code. The goal of these tests is to cover all implemented functionality and verify that computed results are within expected precision, independently of the compute hardware used. Following such *test-driven development* eases the entry burden for accelerator scientists adding new functionality to the project, since automated testing will inform them if unexpected side-effects of changed code would change benchmarked physics results. Tests and examples also add a solid body of documented examples to the project. Detailed validation of the code using standard space charge benchmark tests is described elsewhere in these proceedings [32].

NUMERICAL EXAMPLES

Kurth beam in a periodic focusing channel

This benchmark of high intensity space charge is described in detail in [32]. The test case consists of a 10 nC proton bunch with a kinetic energy of 2 GeV propagating in a periodic channel comprised of alternating drift spaces and constant linear focusing sections (k = 0.7/m). The bunch has a 6D distribution of Kurth type [33], providing an exact, self-consistent solution of the Vlasov-Poisson equations with 3D space charge. The bunch and the external focusing are radially symmetric in the bunch rest frame. The bunch has an rms unnormalized emittance of 1 μm in each plane, yielding a (depressed) phase advance per period of 74°. Fig. 3 illustrates the matched beam size over a single period, comparing the values from ImpactX against the rms envelope equations, showing good agreement. See [32] for additional details.

Fermilab IOTA Storage Ring

This benchmark is described in [25], and is reproduced here for illustration. The test case is a model of the bare (linear) lattice of the Fermilab IOTA storage ring (v. 8.4) [34], with optics configured for operation with a 2.5 MeV proton beam. An rms-matched proton beam with an unnormalized emittance of 4.5 µm propagates over a single turn. The second moments of the particle distribution after a single turn are checked to coincide with the initial second moments of the particle distribution, to within the level expected due to numerical particle noise.

In Fig. 4, the reference orbit indicating the global beam position within the ring is shown. Figure 5 shows the rms beam size evolution as a function of path length over a single turn. The thin dark lines are from `ImpactX`, while the light bold lines in the background are from `IMPACT-Z`. The results of the two codes are in excellent agreement.

CONCLUSION

`ImpactX` is under active development, and several capabilities remain to be ported from `IMPACT-Z`. Future plans include detailed code performance and scaling studies, detailed exploration of benchmark tests with mesh refinement, the implementation in ImpactX of 2D and/or 2.5D space charge models appropriate for long or unbunched beams, and the implementation of additional collective effects (including resistive wall wakefields and CSR models).

REFERENCES

[1] R. L. Gluckstern, "Analytic Model for Halo Formation in High Current Ion Linacs", *Phys. Rev. Lett.*, vol. 73, p. 1247, 1994. doi:10.1103/PhysRevLett.73.1247

[2] J. Qiang and R. D. Ryne, "Beam halo studies using a three-dimensional particle-core model", *Phys. Rev. Spec. Top. Accel. Beams*, vol. 3, p. 064201, 2000. doi:10.1103/PhysRevSTAB.3.064201

[3] Dong-O Jeon, "Evidence of a halo formation mechanism in the spallation neutron source linac", *Phys. Rev. Spec. Top. Accel. Beams*, vol. 16, p. 040103, 2013. doi:10.1103/PhysRevSTAB.16.040103

[4] J. Eldred, V. Lebedev, K. Seiya, and V. Shiltsev, "Beam intensity effects in Fermilab Booster synchrotron", *Phys. Rev. Accel. Beams*, vol. 24, 044001 (2021). doi:10.1103/PhysRevAccelBeams.24.044001

[5] T. Yasui *et al*, "Transverse emittance growth caused by space-charge-induced resonance", *Phys. Rev. Accel. Beams*, vol. 23, 061001 (2020), doi:10.1103/PhysRevAccelBeams.23.061001

ISBN: 978-3-95450-253-0 ISSN: 2673-5571 doi:10.18429/JACoW-HB2023-TUA2I2

[6] J. Qiang, "Three-dimensional envelope instability in periodic focusing channels", *Phys. Rev. Accel. Beams*, vol. 21, p. 034201, 2018. doi:10.1103/PhysRevAccelBeams.21.034201

[7] T. Zolkin, A. Burov, and B. Pandey, "Transverse mode-coupling instability and space charge", *Phys. Rev. Accel. Beams*, vol. 21, p. 104201 (2018). doi:10.1103/PhysRevAccelBeams.21.104201

[8] E. Métral, "General mitigation techniques for coherent beam instabilities in particle accelerators", *Eur. Phys. J. Plus*, vol. 137, p. 47, 2022. doi:10.1140/epjp/s13360-021-02264-4

[9] Consortium for Advanced Modeling of Particle Accelerators, https://campa.lbl.gov/about-campa/

[10] Exascale computing project, https://www.exascaleproject.org/

[11] L. Fedeli, A. Huebl, *et al.*, "Pushing the frontier in the design of laser-based electron accelerators with groundbreaking mesh-refined particle-in-cell simulations on exascale-class supercomputers", in *Proc. SC'22*, Los Alamitos, CA, USA, Nov. 2022. doi:10.1109/SC41404.2022.00008

[12] Scientific Discovery through Advanced Computing, https://www.scidac.gov/about.html

[13] "General accelerator R & D program, accelerator and beam physics roadmap", *DOE Accelerator and Beam Physics Roadmap Workshop*, Sep. 2022, https://science.osti.gov/hep/Community-Resources

[14] J.-L. Vay, "Collaboration for advanced modeling of particle accelerators", 2023 SciDAC Principal Investigator (PI) Meeting, Rockville, MD, USA, Sep. 2023.

[15] D. Sagan, M. Bertz *et al*, "Simulations of future particle accelerators: issues and mitigations", *J. Instrum.*, 16(10):T10002, oct 2021.

[16] A. Huebl *et al*, "openPMD 1.0.0: A meta data standard for particle and mesh based data", 2015. doi:10.5281/zenodo.33624

[17] PICMI: Standard input format for particle-in-cell codes, https://picmi-standard.github.io

[18] A. Huebl *et al*, "openPMD-api: C++ & Python API for scientific I/O with openPMD", 2018, https://github.com/openPMD/openPMD-api, 10.14278/rodare.27

[19] R. Lehe, A. Huebl, S. Jalas, *et al*, "openPMD-viewer: Python visualization tools for openPMD files", 2016, https://github.com/openPMD/openPMD-viewer

[20] BLAST code suite, https://blast.lbl.gov

[21] J.-L. Vay, D. P. Grote, R. H. Cohen, and A. Friedman, "Novel methods in the particle-in-cell accelerator code-framework warp", *Comput. Sci. & Disc.*, vol. 5, p. 014019, 2012. doi:10.1088/1749-4699/5/1/014019

[22] A. Friedman, *et al.*, "Computational methods in the warp code framework for kinetic simulations of particle beams and plasmas", *IEEE Trans. Plasma Sci.*, vol. 42, pp. 1321-1334, 2014. doi:10.1109/TPS.2014.2308546

[23] A. Myers *et al*, "Porting WarpX to GPU-accelerated platforms", *J. Parallel Comput.*, vol. 108, p. 102833, 2021. doi:10.1016/j.parco.2021.102833

[24] S. Diederichs *et al*, "HiPACE++: A portable, 3D quasi-static particle-in-cell code", *Comput. Phys. Commun.*, vol. 278, p. 108421, 2022. doi:10.1016/j.cpc.2022.108421

[25] A. Huebl *et al*, "Next generation computational tools for the modeling and design of particle accelerators at exascale", in *Proc. 2022 North American Particle Acc. Conf. (NAPAC'22)*, Albuquerque, NM, USA, Aug. 2022, pp. 302–306. doi:10.18429/JACoW-NAPAC2022-TUYE2

[26] A. Huebl, C. E. Mitchell, R. Lehe, J. Qiang, *et al*, "ECP-WarpX/impactx: 22.08", 2022. doi:10.5281/zenodo.6954923

[27] W. Zhang *et al*, "MReX: a framework for block-structured adaptive mesh refinement"A, *J. Open Source Softw.*, vol. 4, no. 37, p. 1370, 2019. doi:10.21105/joss.01370

[28] J. Qiang, R. Ryne, S. Habib, and V. Decyk, "An object-oriented parallel particle-in-cell code for beam dynamics simulation in linear accelerators", *J. Comput. Phys.*, vol. 163, pp. 434–451, 2000. doi:10.1006/jcph.2000.6570

[29] IMPACT-Z github repository, https://github.com/impact-lbl/IMPACT-Z

[30] A. Dragt *et al*, *MARYLIE 3.0 user's manual*, Dept. of Physics and Astronomy, University of Maryland, College Park, MD, USA. https://www.physics.umd.edu/dsat/

[31] R. D. Ryne *et al.*, "Recent progress on the MaryLie/IMPACT beam dynamics code", in *Proc. ICAP'06*, Chamonix, France, Oct. 2006, pp. 157–159.

[32] C. Mitchell *et al*, "ImpactX modeling of benchmark tests for space charge validation", presented at HB'23, Geneva, Switzerland, Oct. 2023, paper THBP16, these proceedings.

[33] C.E. Mitchell, K. Hwang, and R.D. Ryne, "Kurth Vlasov-Poisson solution for a beam in the presence of time-dependent isotropic focusing", in *Proc. IPAC'21*, Campinas, SP, Brazil, May 2021, pp. 3213–3216. doi:10.18429/JACoW-IPAC2021-WEPAB248

[34] S. Antipov *et al*, "IOTA (Integrable Optics Test Accelerator): facility and experimental beam physics program", *J. Instrum.*, vol. 12, p. T03002, 2017. doi:10.1088/1748-0221/12/03/T03002