ISBN: 978-3-95450-261-5 ISSN: 2673-7000 doi: 10.18429/JACoW-NAPAC2025-MOP028

# FACILITY-SCALE DIFFERENTIABLE VIRTUAL ACCELERATOR AT FERMILAB*

N. Kuklev†, M. Wallbank, N. Banerjee, J. Jarvis, A. Romanov
Fermi National Accelerator Laboratory, Batavia, IL, USA



*Abstract*

As the design complexity of modern accelerators grows, there is more interest in using advanced simulations and algorithms that have fast execution time or yield additional insights. One notable example are the gradients of physical observables with respect to design parameters, which are broadly useful in optimization and uncertainty analysis. The IOTA/FAST facility has been working on implementing and experimentally validating an end-to-end virtual accelerator test stand that is both fast and gradient-aware, allowing for rapid prototyping of new software and experiments with minimal beam time costs. We describe the selection and benchmarking of both physics and ML codes for linac and ring simulation, including obtaining parameter gradients with autodiff. We will also show the development of generic interfaces between surrogate and physics-based sections, and how the control interface is exposed as either a deterministic discrete event simulator or a fully asynchronous EPICS/ACNET soft IOC. We will also discuss challenges in model calibration and uncertainty quantification, as well as future plans to extend modelling to other Fermilab accelerators like PIP-II and Booster.

## INTRODUCTION

Particle accelerator projects face increasing performance demands, resulting in tighter tolerances on lattice accuracy and stability. Achieving these tolerances requires both more advanced diagnostics and more commissioning time, which contributes to rising costs and lower beam availability.

A common approach to reduce commissioning risks is to implement realistic versions of key accelerator systems as simulations and use them to validate commissioning procedures. A recent successful example of this approach is the Advanced Photon Source Upgrade Project (APS-U), which had a commissioning plan that included advanced ELEGANT simulations, detailed task scheduling, and many validation studies, including within a dedicated EPICS control network [1–3].

So far most commissioning simulation projects have focused on the specific needs and quirks of their respective machines, avoiding complexity through implementing only the physics models and control system behaviors relevant to a particular project. This approach is not scalable, especially for a research facility like FAST/IOTA, since our machines are in constant flux and undergoing upgrades as the various experiments and elements are installed. Moreover, Fermilab main complex is currently undergoing several major upgrades, PIP-II (Proton Improvement Plan II) and ACORN (Accelerator Controls Operations Research Network). These will result in a mix of ACNET and EPICS controls systems, with brand new beamlines and devices having to interface with those over 40 years old.

Given the above complexity, significant validation and practice will be needed to ensure smooth PIP-II commissioning and operation. We require a tool capable of testing classic methods (such as optics measurements via LOCO [4, 5], orbit feedback, multi-wire measurements, and many others), but also amenable to training and validating a new generation of machine learning (ML) tools. Notable examples of the latter include Bayesian optimization (BO) [6–9], reinforcement learning (RL) [10], and phase space reconstruction [11]. ML methods are especially promising to address issues that have previously relied on operational experience or incomplete, qualitative simulations. This paper presents our recent progress in development of a virtual accelerator (digital twin) toolkit as a potential unified facility-scale solution for Fermilab and potentially other accelerators. We will first describe the key requirements and resulting architectural details. Then, we will show use cases and our first upcoming project, the IOTA proton injector commissioning.

## VIRTUAL ACCELERATOR

Throughout this paper, we will use the term '(surrogate) model' to describe the specific (approximate) implementation of an accelerator or system. For example, an ImpactX simulation and a neural network (respectively) that can track beam with space charge. We will use the term 'virtual accelerator' to describe a framework that exposes and coordinates both standard and surrogate models to form a complete simulated control and physics environment. In the industry, the term 'digital twin' is also popular. One can think of virtual accelerators as backend engines for a virtual test stand, whereby various programs and users can interact with the models in a configurable environment with an appropriate mix of control systems, simulation fidelities, and computational requirements.

Our plan for virtual accelerator development has three main goals aligned with both the Fermilab project priorities and the more global challenges described in the recent DOE GARD accelerator and beam physics roadmap [12, 13]:

- To provide a robust virtual test stand for testing tools and procedures, including commissioning and operational applications as well as research codes.

* This manuscript has been authored by FermiForward Discovery Group, LLC under Contract No. 89243024CSC000002 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.
† nkuklev@fnal.gov

ISBN: 978-3-95450-261-5 ISSN: 2673-7000 doi: 10.18429/JACoW-NAPAC2025-MOP028

- To enable training and validation environments for ML-based surrogates and computational methods (differentiable modelling), including integration with both standard physics codes and ML frameworks.
- To generate a near real-time model of the accelerator that is adaptively calibrated against incoming data (real-to-virtual) and provides accurate prediction and uncertainty quantification of key lattice and beam parameters (virtual-to-real), as well as best settings to achieve desired state.

## ARCHITECTURE

While interlinked, above goals place unique requirement on the software architecture. For instance, to ensure compatibility of several simulation codes, data structures are necessary that can keep track of and convert phase space coordinates. To enable model calibration, a bidirectional data flow interface is necessary along with a capability to assimilate experimental readings. Fundamentally, most of these challenges are variants of either optimization/fitting (for real or model parameters) and/or doing simulations quickly. Our core implementation of these tasks is called Apollo - Accelerator Pipeline for Optimization and Low-Latency Operation. Figure 1 shows the general 3 layer architecture of Apollo.

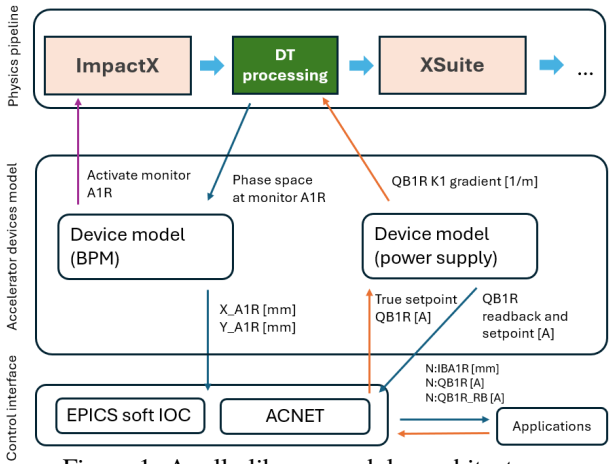


Figure 1: Apollo library modular architecture.

### *Physics Pipeline*

The first layer of Apollo serves to create or wrap various standard and ML-based simulation code into a pipeline that can be portably executed as a single unit. After a large review of active codes, we chose to directly support two: ImpactX [14] and Xsuite [15]. Key determining factors were availability of Python bindings, active development community, and enough physics models to cover most of the elements we expect to use. Of special importance is support for intense space charge and other collective effects. Other Fermilab groups also had good experiences with these codes. Direct support means that Apollo can 'synthesize' both simulation commands and lattices on the fly from a common shared format, transparently handling implementation details like dipole edges and coordinate transformations. For a couple other codes like TraceWin [16], a wrapper class is available to run pre-made simulations but it has no synthesis capabilities. For custom models, a black-box interface specification can be used. For differentiable models, we created our own tracking code that can generate whole beamlines in either Jax or PyTorch. We also support arbitrary Jax and PyTorch-based modules using the standard parameter interface.

### *Device Models*

Second layer of Apollo is responsible for determining how to convert parameters and outputs between the physics pipeline and the real world. For example, calculating the magnet gradient based on the virtual power supply current, or beam position from the phase space. Device interface is kept generic, resembling that of discrete event simulators, and allowing for complex time-dependent implementation logic. For example, the standard power supply model includes hysteresis, ADC noise, and overshoot behavior. Active devices determine which simulation outputs are requested, minimizing unnecessary computations.

### *Control Interface*

Final layer of Apollo is responsible for presenting the devices to clients through a control system. It runs asynchronously to the simulation loop, ensuring prompt network communication with multiple clients. We use external libraries for EPICS implementation, with CA interface through 'caproto' and PV access through 'p4p'. Both present devices as configurable sets of records with appropriate fields (including limits and other customary EPICS hints) and support for monitoring and scanning. For ACNET, we have recently added pure-Python implementations of GETS32 and SETS32 protocols to simulate frontends as well as a virtual Data Pool Manager (DPM) gRPC endpoint to simulate central control services.

### *System Design*

Throughout Apollo we imposed additional requirements to ensure the implementation remains flexible, portable, and yet robust to misconfigurations or bugs. Namely, each layer must be independently serializable and executable (i.e., in cluster environment). Parameters and return values should be validated against specifications at every step. This is accomplished by using 'pydantic' library and a set of declarative inputs and output for each model and device. Pydantic enables serialization to and from json, yaml, and other formats. For example, a model might declare phase space output through OpenPMD standard [17], or specify one of the supported longitudinal and transverse coordinate types for automatic conversion.

We avoid complexity of nested data containers by using a single shared namespace per layer, and a global state object that carries both current values and history. This ensures

ISBN: 978-3-95450-261-5 ISSN: 2673-7000 doi: 10.18429/JACoW-NAPAC2025-MOP028



synchronized modification, but does limit scalability, especially in Python. We mitigate this by offloading execution to distributed frameworks like Ray and Dask along with shared in-memory caching services like Redis, while keeping the coordinator thread free.

## DIFFERENTIABLE SIMULATIONS

Modern ML frameworks have made it feasible to implement complex physics models with automatic differentiation. Differentiable models can be run just like regular simulations, but in addition to output values yield the gradients of outputs with respect to a designated set of parameters, such as lattice gradients or beam shape. They are useful in accelerator design, optimization, model calibration, and machine learning. Apollo implements a special pipeline type if all component models can do gradient propagation, with support for PyTorch and Jax. Both ecosystems also have packages for NN training/serving, RL, and Bayesian inference (Pyro [18], NumPyro [19]), and can be used for calibration and surrogate modelling [20]. Uniquely for PyTorch, we support uncertainty-aware modules like Gaussian processes.

Because Apollo prioritizes fast execution, we developed a new accelerator tracking code called JACC (Jax for ACCelerators). It implements standard 6x6 first order matrices as well as select second order matrices and drift-kick integrators through bends, quads, multipoles, and RF elements. There is also a single-bunch space charge implementation and implicit vectorization capability consistent with Jax 'vmap' style. Uniquely, whole beamline can be compiled to both Jax and PyTorch backends as a single unit, with pruning and element fusion where possible. For pure numeric simulations, Jax typically has better performance. For development and debugging, PyTorch eager-style execution is significantly easier to work with.

## USE CASES

### *IOTA Proton Injector*

IOTA proton injector (IPI) project aims to add a 2.5MeV proton injector alongside existing 150MeV electron injection line [21,22]. Scientifically, this will add a capability to study space charge dominated beams and their control methods using a well understood storage ring with lattice accuracy comparable to light sources. Hardware is currently being installed, with commissioning set to begin in the fall of 2025. Diagram of the IPI is shown in Fig. 2.

IPI will be the first machine to use Apollo to simulate commissioning and infer beam/beamline parameters, such as the beam phase space coming from the RFQ and element misalignments. Full MEBT section lattice is already implemented in both standard and differentiable formats, and work is ongoing to construct an open-source RFQ tracking model.

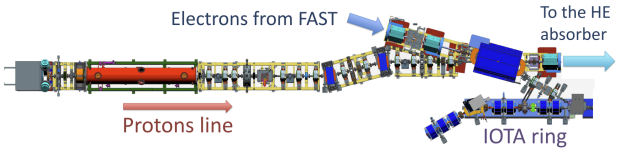


Figure 2: IOTA proton injector layout and integration with existing electron beamlines. Proton line consists of the duoplasmotron source (left end, gray), LEBT (yellow, short), RFQ (red), and MEBT (yellow, long) sections. MEBT contains a dogleg, debuncher, and various beam diagnostics.

### *FNAL Main Complex in the PIP-II Era*

Supporting all main complex machines is the long-term goal for Apollo. This is made difficult due to the complex physics (SRF cavities, ramping machines, impedances, feedback systems) and a large number of codes used by various departments. For PIP-II, we started adding advanced cavity and RFQ models to better match the reference TraceWin implementation. In the Booster, for key beam power limiting factors like beam losses, we are training data-driven surrogates because simulations have so far not achieved quantitative agreement. We are also studying performance of forward mode differentiation for multi-turn differentiable tracking. Main Injector already uses Xsuite simulations extensively, and we expect no major roadblocks integrating them into Apollo.

## CONCLUSION

Robust virtual simulation and testing environments are important for ensuring efficient design, commissioning, and operation of the new generation of particle accelerators. They enable development of many other capabilities, and thanks to differentiable codes are a powerful inference and model calibration tool in their own right. In this paper we have described our development of a differentiable virtual accelerator framework 'Apollo'. It is our hope that using a modern software stack, robust system design practices, and a modular architecture will cover the majority of use cases at Fermilab and beyond, creating a first generic virtual accelerator framework. We are planning on using Apollo for IOTA IPI commissioning in the fall of 2025, and through this early experimental testing will discover and address any issues well ahead of PIP-II commissioning. Future work will focus on improving user experience, integration with live data readout, as well as extending the differentiable capabilities with advanced impedance, cavity, and feedback models.

## REFERENCES

[1] V. Sajaev, "Commissioning simulations for the Argonne Advanced Photon Source Upgrade lattice", *Phys. Rev. Accel. Beams*, vol. 22, no. 4, p. 040 102, 2019. doi:10.1103/PhysRevAccelBeams.22.040102

[2] V. Sajaev, "Improvements to the commissioning simulations of the APS Upgrade storage ring", in *Proc. IPAC'23*, Venice, Italy, pp. 3112–3115, 2023. doi:10.18429/JACoW-IPAC2023-WEPL006

ISBN: 978-3-95450-261-5 ISSN: 2673-7000 doi: 10.18429/JACoW-NAPAC2025-MOP028

[3] V. Sajaev *et al.*, "Commissioning of the Advanced Photon Source Upgrade – The first swap-out injection-based synchrotron light source", in *Proc. IPAC'25*, Taipei, Taiwan, Jun. 2025, paper TUCD2, to be published.

[4] D. Vilsmeier, R. Singh, and M. Bai, "Inverse modeling of circular lattices via orbit response measurements in the presence of degeneracy", *Phys. Rev. Accel. Beams*, vol. 26, no. 3, p. 032 803, 2023. doi:10.1103/PhysRevAccelBeams.26.032803

[5] A. Romanov *et al.*, "Correction of magnetic optics and beam trajectory using LOCO based algorithm with expanded experimental data sets", 2017, arXiv:1703.09757 [physics.acc-ph]. doi:10.48550/arXiv.1703.09757

[6] N. Kuklev, M. Borland, G. I. Fystro, H. Shang, and Y. Sun, "Online accelerator tuning with adaptive Bayesian optimization", in *Proc. NAPAC'22*, Albuquerque, NM, USA, pp. 842–845, 2022. doi:10.18429/JACoW-NAPAC2022-THXD4

[7] N. Kuklev, M. Borland, G. Fystro, H. Shang, and Y. Sun, "Robust adaptive Bayesian optimization", in *Proc. IPAC'23*, Venice, Italy, pp. 4428–4431, 2023. doi:10.18429/JACoW-IPAC2023-THPL007

[8] R. Roussel *et al.*, "Bayesian optimization algorithms for accelerator physics", *Phys. Rev. Accel. Beams*, vol. 27, no. 8, p. 084 801, 2024. doi:10.1103/PhysRevAccelBeams.27.084801

[9] N. Kuklev *et al.*, "High efficiency multi-objective Bayesian algorithm for APS-U nonlinear dynamics tuning", in *Proc. IPAC'25*, Taipei, Taiwan, Jun. 2025, paper THPM102, to be published.

[10] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning", *Found. Trends Mach. Learn.*, vol. 11, no. 3–4, pp. 219–354, 2018. doi:10.1561/2200000071

[11] R. Roussel *et al.*, "Efficient six-dimensional phase space reconstructions from experimental measurements using generative machine learning", *Phys. Rev. Accel. Beams*, vol. 27, no. 9, p. 094 601, 2024. doi:10.1103/PhysRevAccelBeams.27.094601

[12] Accelerator beam physics research roadmap, https://science.osti.gov/hep/-/media/hep/pdf/2022/ABP_Roadmap_2023_final.pdf

[13] S. Nagaitsev *et al.*, "Accelerator and beam physics research goals and opportunities", 2021, arXiv:2101.04107 [physics.acc-ph]. doi:10.48550/arXiv.2101.04107

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

[15] G. Iadarola *et al.*, "Xsuite: An integrated beam physics simulation framework", in *Proc. HB'23*, Geneva, Switzerland, pp. 73–80, 2024. doi:10.18429/JACoW-HB2023-TUA2I1

[16] D. Uriot and N. Pichoff, "Status of TraceWin code", in *Proc. IPAC'15*, Richmond, VA, USA, May 2015, pp. 92–94. doi:10.18429/JACoW-IPAC2015-MOPWA008

[17] C++ and python api for scientific i/o with openpmd, https://github.com/openPMD/openPMD-api

[18] E. Bingham *et al.*, "Pyro: Deep universal probabilistic programming", *J. Mach. Learn. Res.*, vol. 20, no. 28, pp. 1–6, 2019.

[19] J. Bradbury *et al.*, *JAX: Composable Transformations of Python+ NumPy Programs*, version 0.4.8, 2023. http://github.com/google/jax

[20] N. Kuklev, "Differentiable beam optics optimization and measurement", in *Proc. IPAC'23*, Venice, Italy, pp. 3108–3111, 2023. doi:10.18429/JACoW-IPAC2023-WEPL004

[21] D. Edstrom *et al.*, "IOTA proton injector beamline installation", in *Proc. IPAC'23*, Venice, Italy, pp. 1737–1739, 2023. doi:10.18429/JACoW-IPAC2023-TUPA183

[22] J. Wieland and A. Romanov, "Injection simulations of space charge dominated proton beams in IOTA", in *Proc. IPAC'25*, Taipei, Taiwan, Jun. 2025, paper TUPB020, to be published.