ISBN: 978-3-95450-219-6 ISSN: 2226-0366 doi: 10.18429/JACoW-LINAC2024-THPB068

ADVANCEMENTS IN BACKWARDS DIFFERENTIABLE BEAM DYNAMICS SIMULATIONS FOR ACCELERATOR DESIGN, MODEL CALIBRATION, AND MACHINE LEARNING

R. Roussel[1], G. Charleux[2], A. Edelen[1], A. Eichler[3], J. P. Gonzalez-Aguilera[4], A. Huebl[2], J. Kaiser[3], R. Lehe[2] A. Santamaria Garcia[5], C. Xu[5]
[1] SLAC National Accelerator Laboratory, Menlo Park, CA, USA
[2] Lawrence Berkeley National Laboratory, Berkeley, CA, USA
[3] Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany
[4] University of Chicago, Chicago, IL, USA
[5] Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract

Many accelerator physics problems such as beamline design, beam dynamics model calibration or interpreting experimental measurements rely on solving an optimization problem that use a simulation of beam dynamics. However, it is difficult to solve high dimensional optimization problems using current beam dynamics simulations because calculating gradients of simulated objectives with respect to input parameters is computationally expensive in high dimensions. To address this problem, backwards differentiable beam dynamics simulations have been developed that enable computationally inexpensive calculations of objective gradients that are largely independent of the number of input parameters. In this work, we highlight current and future applications of differentiable beam dynamics simulations in accelerator physics, such as improving accelerator design, model calibration, and machine learning. We also describe current collaborative efforts between SLAC, DESY, KIT, and LBNL to implement fast, backwards differentiable beam dynamics simulations in Python. These tools will enable unprecedented improvements in optimization efficiency and speed when using beam dynamics simulations, leading to enhanced control and detailed understanding of physical accelerator systems.

INTRODUCTION

Solving challenging optimization problems is critical to accelerator design, interpreting experimental measurements, and calibrating physics models to realistic beam dynamics. However, current beam physics simulations can present a bottleneck in the optimization process. Most current beam dynamics simulations used in accelerator physics are "gradient-free" simulations, where only the input and output of the model is available. This limits the applicability of powerful gradient-based optimization algorithms, such as first- or second-order gradient descent, which often outperform optimization algorithms that do not use gradient information [1]. The only available methods for estimating gradients of gradient-free simulations is finite difference methods, which require higher computational costs proportional to the number of optimization parameters. As a result, optimization problems that include beam dynamics simulations are often restricted to optimizing a handful of parameters at once.

An alternative simulation paradigm to gradient-free simulations is so-called "differentiable simulations" , illustrated in Fig. 1. In addition to making conventional predictions, differentiable simulations also provide the gradient of simulations with respect to input parameters. Differentiable simulations are enabled through the use of a computational technique known as "automatic differentiation" [2]. Computational programs, such as beam dynamics simulations, can be broken down into a set of atomic operations (addition, multiplication, exponentiation etc.) for which the derivative is analytically known. Differentiable simulations track the derivative of each calculation step alongside normal computation and use the chain rule to calculate total derivatives of simulation outputs with respect to simulation inputs.

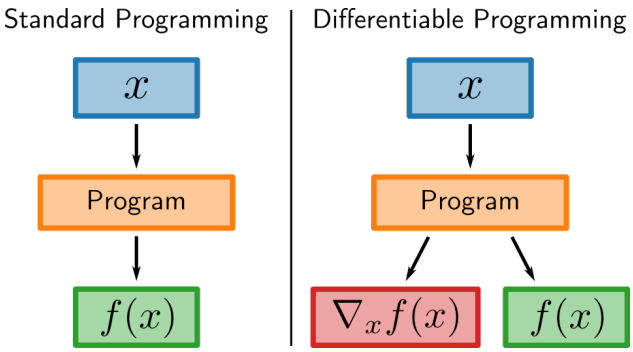

Figure 1: Block diagram showing the difference between standard programming used to model physical phenomena, which produces outputs $f(x)$ as a function of model inputs x, and differentiable programming, which produces gradients of model outputs $\nabla_x f(x)$ in addition to model outputs.

Several existing beam dynamics software packages [3–5] use a technique known as "forward differentiation" [2] to calculate arbitrary order Taylor maps of accelerator elements, and more recently, to optimize accelerator design parameters for beam dynamics under the influence of space charge [6]. However, the efficacy of this method is restricted to computing derivatives with respect to a small number of input parameters, making it impractical for optimizing problems that contain a large number of free parameters. On the other

ISBN: 978-3-95450-219-6 ISSN: 2226-0366 doi: 10.18429/JACoW-LINAC2024-THPB068

hand, "backwards" differentiable calculations (sometimes referred to "adjoint differentiation") offer the advantage of computing gradients with less computational cost given a large number of input parameters.

In both forwards and backwards cases, the novel differentiable simulations open doors to new solutions for accelerator design and tuning tasks. Some example applications that have been demonstrated using the backwards differentiable codes Cheetah [7] and Bmad-X [8] are listed in Fig. 2 and Fig. 3 respectively. First and foremost, backwards differentiation enables gradient-based optimization for lattice parameters during the design stage. The derivative information also allows efficient model calibration, also known as system identification, for the simulation model. For example, the unknown misalignments of the components can be fitted to the measured data, improving the accuracy of the simulation model. The differentiable simulations can also be coupled to other learning-based optimization algorithms. In the case of the sample-intensive reinforcement learning (RL) algorithms, the simulation serves a fast-executing training environment, removing the need for training on the real accelerators and reducing the overall training time by several orders of magnitude. The derivative information can be further used to accelerate the training of some RL algorithms. For the Bayesian optimization algorithms, the differentiable simulation can be included as a physics-informed prior mean function for the Gaussian process model. This provides extra knowledge on the optimization task and can speed up the convergence in accelerator tuning tasks [9, 10].

In this work, we describe collaborative efforts to combine efforts on implementing backwards differentiable beam dynamics simulations for solving challenging optimization problems. This enables unprecedented capabilities in solving accelerator design challenges, calibrating beam dynamics simulations to experimental measurements, and integrating physics information into machine learning workflows.

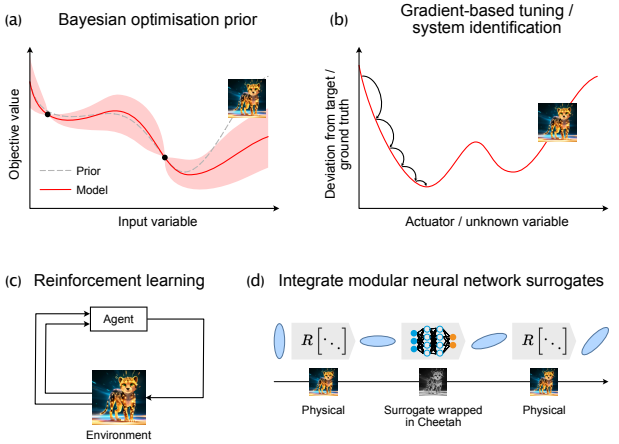

Figure 2: Example applications using the differentiable simulation code Cheetah [7]. (a) Serving as a physics-informed prior mean for BO. (b) Differentiable beam dynamics tracking for gradient-based accelerator tuning and system identification. (c) Providing fast beam dynamics environments for training RL agents. (d) Training modular neural network surrogate models within physical beam dynamics simulations.

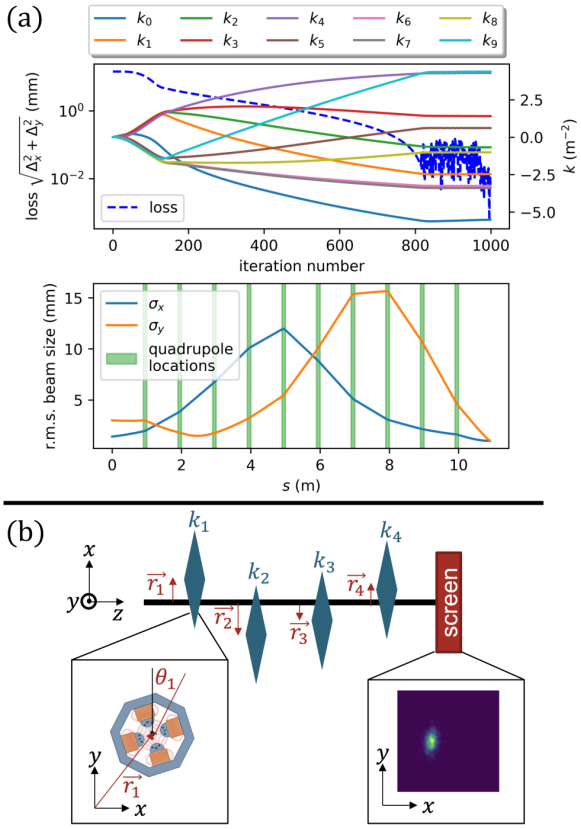

Figure 3: Example applications using the differentiable simulation code Bmad-X [8]. (a) High-dimensional optimization of beamline parameters for accelerator design. (b) Calibration of beamline parameters to experimental data. These applications have also been investigated using Cheetah.

BACKWARDS DIFFERENTIABLE TRACKING PACKAGE IN PYTHON

Bmad-X [8] is a Python package with library-agnostic particle tracking routines based on Bmad [5]. This allows the use of well established machine learning Python libraries, such as PyTorch [11], as the particle tracking back-end to enable backward mode automatic differentiation and GPU acceleration. Additionally, using PyTorch as Bmad-X back-end facilitates the integration of physics-based accelerator simulations with neural network models and gradient-based optimization routines.

The simulation code Cheetah [7] is implemented using the PyTorch framework and benefits from its optimized tensor computations, single-node GPU support, and automatic backward mode differentiation features. Cheetah consists of two main modules, `Beam` and `Element`, representing particle beams and the individual accelerator components. It currently supports two types of tracking, namely the track-

ISBN: 978-3-95450-219-6 ISSN: 2226-0366 doi: 10.18429/JACoW-LINAC2024-THPB068

ing of statistical parameters assuming a Gaussian distributed beam with `ParameterBeam` and the tracking of macroparticles using the `ParticleBeam`. For the lattice elements such as drift sections and magnets, linear transfer maps are implemented and used as the default tracking routine. Some more complex tracking methods, including space charge effects, are also implemented. Cheetah also includes several optional speed optimizations, such as intelligent transfer map reduction and lazy execution. When activated, these can, in some cases, allow the tracking to run 2 to 8 orders of magnitude faster than compared-to CPU simulation codes [7]. What is more, Cheetah offers converters from a number of other simulation codes to make integrating it with conventional non-differentiable simulations straightforward.

Bmad-X tracking routines provide higher fidelity by using the map derived from the element Hamiltonian and use a kick-drift-kick model. On the other hand, most of Cheetah tracking routines are linear transfer maps, speeding up the computation, specially when using hardware acceleration.

CURRENT AND FUTURE WORK

Extensive efforts are currently underway to unify Cheetah and Bmad-X into a single differentiable beam dynamics package. As part of this effort, Bmad-X tracking methods are being integrated into Cheetah for its upcoming version 0.7 release. This will allow the user to select one of the two different tracking methods for each individual element in a beamline. The combination of their capabilities will result in a more flexible differentiable simulator and enable the deployment of methods like phase space reconstruction to production.

The efforts are accompanied by the vectorization of Cheetah. The new vectorized implementation will allow for highly concurrent simulation of different beamline configurations and working conditions, allowing the user to take full advantage of highly parallel hardware acceleration as it is provided by GPUs. A pre-release development version of Cheetah v0.7 with vectorization has already been found to enable an up to 50-fold increase in compute speed just on CPU when multiple concurrent configurations are run. This improvement is expected to be more significant on GPU.

The next release of Cheetah further comes with a 3D space charge module. This implementation uses the usual Particle-In-Cell approach, whereby:

- the charge density of the beam is deposited on a grid
- the corresponding electric potential is obtained the same grid by solving the Poisson equation (in the case of Cheetah, by using the integrated Green's function method [12, 13])
- the corresponding force is computed from the electric potential on a grid, and then gathered from the grid on the beam particles

Importantly, because Cheetah implements each of these operations with Pytorch, the overall space-charge module is fully backward-differentiable. To our knowledge, this is the first instance of a backward-differentiable accelerator code with space-charge. Future work will explore the computational scaling of backward differentiability for simulations with space-charge, using this new tool.

All of these features are expected to release in a stable build of Cheetah soon.

CONCLUSION

In this work, we have described collaborative efforts to implement backwards differentiable beam dynamics simulations for use in accelerator physics applications. The most important avenue for future work regarding auto-differentiable beam dynamics simulators is their application. While we have explored a number of applications in prior work [7, 14, 15], we believe that the potential impact of such simulators goes far beyond these examples. As such, the dissemination of these methods through easy-to-use simulators like Cheetah will be a key part of future efforts in this direction. The application of Cheetah to novel tasks will also inform its future development.

ACKNOWLEDGEMENTS

This work is supported by the U.S. Department of Energy, Office of Science under Contract No. DE-AC02-76SF00515 and the Center for Bright Beams, NSF award PHY-1549132. This work has in part been funded by the IVF project InternLabs-0011 (HIR3X) and the Initiative and Networking Fund by the Helmholtz Association (Autonomous Accelerator, ZT-I-PF-5-6). The authors acknowledge support from DESY (Hamburg, Germany) and KIT (Karlsruhe, Germany), members of the Helmholtz Association HGF. This work was 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.

REFERENCES

[1] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 1999.

[2] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2008. doi:10.1137/1.9780898717761

[3] K. Makino and M. Berz, "COSY INFINITY Version 9", *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 558, no. 1, pp. 346–350, 2006. doi:10.1016/j.nima.2005.11.109

[4] E. Forest, F. Schmidt, and E. McIntosh, "Introduction to the polymorphic tracking code", KEK, Tsukuba, Japan, 2002.

[5] D. Sagan, "Bmad: A relativistic charged particle simulation library", *Nucl. Instrum. Meth. Phys. Res., Sect. A*, vol. 558, no. 1, pp. 356–359, 2006. doi:10.1016/j.nima.2005.11.001

ISBN: 978-3-95450-219-6 ISSN: 2226-0366 doi: 10.18429/JACoW-LINAC2024-THPB068

[6] J. Qiang, "Differentiable self-consistent space-charge simulation for accelerator design", *Phys. Rev. Accel. Beams*, vol. 26, no. 2, p. 024 601, 2023. doi:10.1103/PhysRevAccelBeams.26.024601

[7] J. Kaiser, C. Xu, A. Eichler, and A. Santamaria Garcia, "Bridging the gap between machine learning and particle accelerator physics with high-speed, differentiable simulations", *Phys. Rev. Accel. Beams*, vol. 27, no. 5, p. 054 601, 2024. doi:10.1103/PhysRevAccelBeams.27.054601

[8] J. Gonzalez-Aguilera, Y. Kim, R. Roussel, A. Edelen, and C. Mayes, "Towards fully differentiable accelerator modeling", in *Proc. IPAC'23*, Venice, Italy, pp. 2797–2800, 2023. doi:10.18429/JACoW-IPAC2023-WEPA065

[9] K. Hwang *et al.*, "Prior-mean-assisted Bayesian optimization application on FRIB Front-End tunning", *arXiv preprint*, 2022. doi:10.48550/arXiv.2211.06400

[10] T. Boltz *et al.*, "More Sample-Efficient Tuning of Particle Accelerators with Bayesian Optimization and Prior Mean Models", *arXiv preprint*, 2024. doi:10.48550/arXiv.2403.03225

[11] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", in *Advances in Neural Information Processing Systems 32 (NeurIPS'19)*, pp. 8024–8035, 2019. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[12] J. Qiang, S. Lidia, R. D. Ryne, and C. Limborg-Deprey, "Three-dimensional quasistatic model for high brightness beam dynamics simulation", *Phys. Rev. Spec. Top. Accel. Beams*, vol. 9, no. 4, p. 044 204, 2006. doi:10.1103/physrevstab.9.044204

[13] J. Qiang, S. Lidia, R. D. Ryne, and C. Limborg-Deprey, "Erratum: Three-dimensional quasistatic model for high brightness beam dynamics simulation [phys. rev. spec. top. accel. beams, vol. 9, p. 044204, 2006]", *Phys. Rev. Spec. Top. Accel. Beams*, vol. 10, no. 12, 2007. doi:10.1103/physrevstab.10.129901

[14] R. Roussel *et al.*, "Phase space reconstruction from accelerator beam measurements using neural networks and differentiable simulations", *Phys. Rev. Lett.*, vol. 130, no. 14, p. 145 001, 2023. doi:10.1103/PhysRevLett.130.145001

[15] R. Roussel *et al.*, "Efficient 6-dimensional phase space reconstruction from experimental measurements using generative machine learning", *arXiv preprint*, 2024. doi:10.48550/arXiv.2404.10853