ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-WEPA065

TOWARDS FULLY DIFFERENTIABLE ACCELERATOR MODELING

J.P. Gonzalez-Aguilera*[1], Y.-K. Kim, University of Chicago, Chicago, IL, USA
R. Roussel, A. Edelen, C. Mayes, SLAC National Accelerator Laboratory, Menlo Park, CA, USA
[1]also at Enrico Fermi Institute, Chicago, IL, USA

Abstract

Optimization and design of particle accelerators is challenging due to the large number of free parameters and the corresponding lack of gradient information available to the optimizer. Thus, full optimization of large beamlines becomes infeasible due to the exponential growth of free parameter space the optimization algorithm must navigate. Providing exact or approximate gradient information to the optimizer can significantly improve convergence speed, enabling practical optimization of high-dimensional problems. To achieve this, we have leveraged state-of-the-art automatic differentiation techniques developed by the machine learning community to enable end-to-end differentiable particle tracking simulations. We demonstrate that even a simple tracking simulation with gradient information can be used to significantly improve beamline design optimization. Furthermore, we show the flexibility of our implementation with various applications that make use of different kinds of derivative information.

INTRODUCTION

Particle accelerators are complex instruments with a large number of free parameters. This aspect makes full optimization of large beamlines extremely challenging. Gradient-based optimization methods are powerful tools to solve high-dimensional problems, but both analytical and numerical differentiation scale poorly with the number of dimensions. Automatic differentiation (AD) allows the evaluation of high-dimensional derivatives efficiently and accurately, which is why it is widely used in the machine learning (ML) community [1]. Thus, *differentiable* accelerator models, *i.e.,* models that support AD, can enable high-dimensional gradient-based optimization in particle accelerators [2–4].

The idea of using AD in particle accelerator simulations is not new [5–7]. Nevertheless, the scope of these implementations is to calculate arbitrary order Taylor polynomials using *forward mode* AD. This sets a limitation on the flexibility of AD usage and its extensibility to high-dimensional optimization problems (>100D). In order to efficiently evaluate gradients in high-dimensional input spaces, the tracking code must support *backward mode* AD. Furthermore, efficient higher-order derivative evaluation needs both forward and backward mode AD. Thus, a more flexible implementation of AD in accelerator modeling is needed for high-dimensional gradient based optimization.

* jpga@uchicago.edu

METHODS

Automatic Differentiation

Automatic differentiation (AD) is a family of techniques used to evaluate partial derivatives of numerical computer functions *fast* and *accurately* [1]. AD makes use of the chain rule and the derivatives of the elementary operations and functions that compose a numerical computation. This procedure results in the computation of derivatives as *fast* as the evaluation of the function times a small constant, and as *accurate* as the working precision of the computation. There are two main AD 'modes': forward mode and backward mode. The choice depends on the dimensions n and m of the function $f : \mathbb{R}^n \to \mathbb{R}^m$ for which the Jacobian is evaluated.

Forward Mode AD In forward mode AD, the chain rule computation order goes from inner to outer arguments. In the limiting case where $n = 1$, the derivatives are computed using a single forward pass through the composition of the functions. In the case $m = 1$, it requires n different evaluations. Thus, forward mode AD runs efficiently when $n \ll m$.

Reverse Mode AD In reverse mode AD, the chain rule computation order goes from outer to inner arguments. In the limiting case where $m = 1$, the function is evaluated first via a forward pass and the derivatives are computed using a single backward pass through the composition of the functions. Thus, reverse mode AD runs efficiently when $m \ll n$. This is the reason why optimization problems with large number of input parameters and a single loss function use reverse mode AD instead of forward mode.

Library-Agnostic Particle Tracking in Python

Python is one of the most popular computer languages, and the community develops various libraries that support forward and reverse mode AD. Furthermore, some of these libraries support GPU acceleration, just-in-time compilation and many other ML tools. In order to make use of all of these tools in particle accelerator modeling, we have developed Bmad-X [8]: a Python package with library-agnostic tracking routines based on Bmad [9]. The code was tested with different examples using the PyTorch library [10].

APPLICATIONS

High-Dimensional Gradient-Based Optimization

Consider a toy-model composed by ten quadrupoles with tunable focusing strengths separated by drift segments of equal length. At the beginning of the lattice ($s = 0\,\mathrm{m}$), the beam has a non-round Gaussian distribution with r.m.s.

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-WEPA065

beamsizes of $\sigma_x = 1\,\text{mm}$ and $\sigma_y = 3\,\text{mm}$. Suppose that a round beam with $\sigma_x = \sigma_y = \sigma_{\text{target}} = 1\,\text{mm}$ is required at the end of the beamline ($s = 11\,\text{m}$). We want to find a set of quadrupole strengths $k_0, \ldots, k_9$ such that the beam profile is as close as the target beam. This can be achieved by minimizing the following loss function:

$$\mathscr{L} = \sqrt{\Delta_x^2 + \Delta_y^2} = \sqrt{(\sigma_x - \sigma_{\text{target}})^2 + (\sigma_y - \sigma_{\text{target}})^2}. \quad (1)$$

Since Bmad-X particle tracking supports reverse mode AD, a gradient-based optimizer can be used to solve this problem efficiently. Figure 1 shows the results using PyTorch's implementation of the Adam optimizer [11], which is an extension of stochastic gradient descent and is commonly used in the ML community. When initiallizing all quadrupole strengths at zero, the optimizer converges to a solution in less than a minute using a consumer-grade Intel Core i7-12700K CPU. By contrast, optimizing with gradient-based algorithms is infeasible when using numerical differentiation due to the large number of input variables.

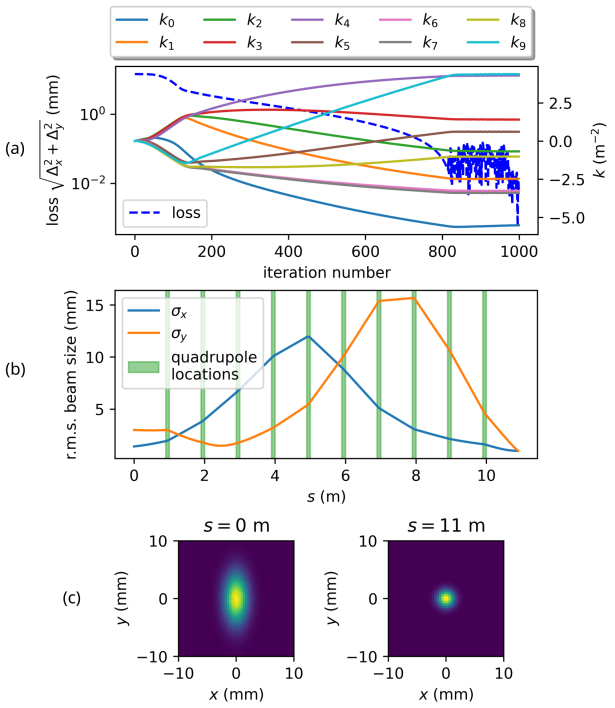

Figure 1: Gradient-based optimization of ten quadrupole strengths to obtain a round beam downstream with r.m.s. beam size of $\sigma_{\text{target}} = 3\,\text{mm}$. (a) Loss function and quadrupole strengths vs. iteration number. (b) σ_x and σ_y r.m.s. beam sizes vs. s position for the optimal quadrupole configuration. (c) Upstream and downstream beam profiles for the optimal quadrupole configuration.

Hessian Computation

It has been shown that Hessian information can speed up Gaussian process optimizers in online particle accelerator tuning [12]. A combination of forward and backward mode AD enables efficient computation of Hessians. Fortunately, most of AD Python libraries have methods to compute the Hessian efficiently since it is widely used in the ML community. Thus, Bmad-X can potentially be used for online machine optimization.

As an example, the Hessian of the ten-quadrupole lattice final beam size with respect to the quadrupole strengths was computed:

$$\text{Hess}(\sigma_x)_{ij} = \frac{\partial^2 \sigma_x}{\partial k_i \partial k_j}, \; i,j = 0, \ldots, 9. \quad (2)$$

The CPU times for this computation were 320 ms and 28 s when using PyTorch's Hessian and numerical differentiation methods, respectively. Figure 2 shows the Hessian computation time for numerical and automatic differentiation versus the number of quadrupoles in the lattice. The exponential grouth when using numerical differentiation makes it impractical for large number of dimensions.

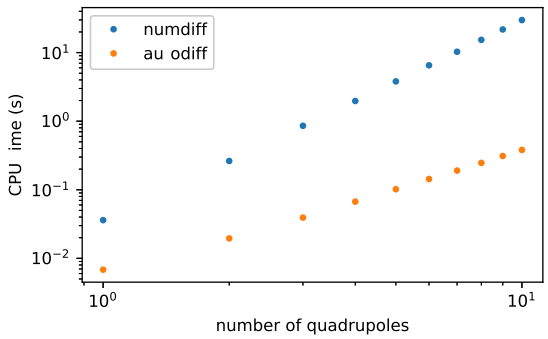

Figure 2: Hessian computation time vs. number of free parameters using numerical and automatic differentiation. 10,000 particles were tracked.

Model Calibration

It is possible to calibrate models by including misalignments in Bmad-X beamline elements. As an example, consider the beamline in Fig. 3 (a). The beamline is composed by four quadrupoles with unknown transverse offsets and tilts, and a screen that measures the transverse beam profile. The offsets kick the beam transversely, and the tilts introduce x-y correlations. This results in transverse displacements and rotations that are functions of the quadrupole strengths, offsets and tilts. Each quadrupole can be independently increased and decreased once, resulting in eight beam profiles observed downstream. Thus, it is possible to find the offsets and tilts by minimizing a loss function that encapsulates the divergence between simulated and observed beam profiles.

The loss function can be defined as the sum of the Kullback-Leibler divergences [13] of the simulated and observed images:

$$\mathscr{L} = \sum_{n=1}^{8} \sum_{i,j} O_n^{(i,j)} \log\left(\frac{O_n^{(i,j)}}{S_n^{(i,j)}}\right), \quad (3)$$

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-WEPA065

where $S_n^{(i,j)}$ and $O_n^{(i,j)}$ are the (i,j) pixel intensities of the n^{th} simulated and observed image respectively. In order to propagate gradients of the loss function with respect to the offsets and tilts, the simulated images must be differentiable. This is achieved by doing a non-parametric estimation of a probability density function that represents the two-dimensional histogram via kernel density estimation [14]. Figure 3 (b) shows an example of kernel density estimation applied to a beam histogram. The results after using the Adam gradient-based optimizer are summarized in Fig. 4.

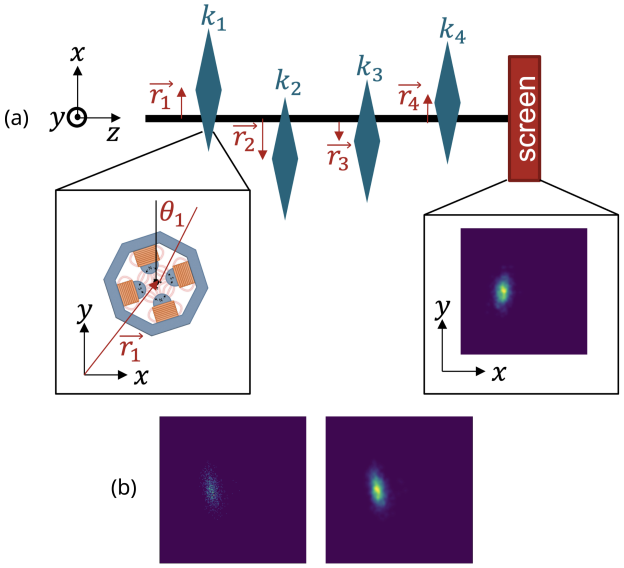

Figure 3: (a) Beamline cartoon for the model calibration example. The beamline consists of four quadrupoles with unknown transverse offsets $\vec{r}_1, \ldots, \vec{r}_4$ and transverse tilts $\theta_1, \ldots, \theta_4$. (b) Particle count x-y histogram (left), and differentiable version using kernel density estimation (right).

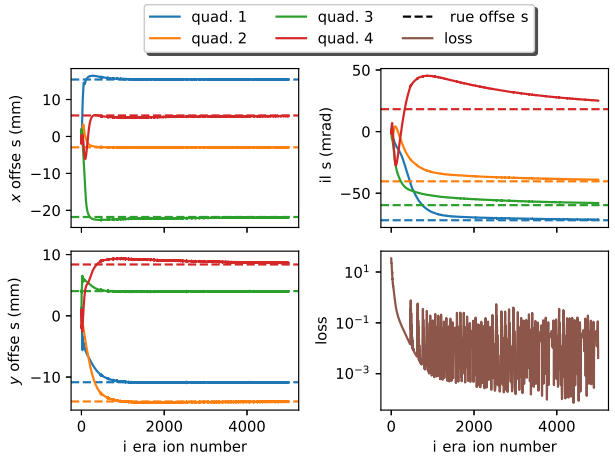

Figure 4: Model calibration results. Iteration number vs. x and y offsets (top/bottom left), tilts (top right), and loss function (bottom right). Unknown offsets and tilts converge to the ground truth values.

Phase Space Reconstruction

Bmad-X has already been used to accurately reconstruct the 4D phase space of a beam with simple diagnostics and few observations in both experiment and simulation [15]. This method makes use of a neural network parametrization of the initial beam phase space. Since neural networks have a large number of free parameters, Bmad-X AD support is essential in the reconstruction process. Additionally, the library-agnostic approach of Bmad-X allows seamless integration between the neural network and the differentiable simulations by using PyTorch, which also enables GPU support for hardware acceleration.

CONCLUSIONS

In this work, we have developed a library-agnostic implementation of Bmad tracking routines in Python. This allows the use of libraries developed by the machine learning community in accelerator simulations. These libraries provide forward and backward mode automatic differentiation support enabling high-dimensional gradient-based optimization of beamlines, efficient Hessian computation for Gaussian process optimizers, model misalignment calibration, and phase-space reconstruction. Additionally, these libraries provide seamless integration with neural networks, GPU acceleration and other machine learning modules.

ACKNOWLEDGEMENTS

The authors would like to thank William Lou for the development of the bending magnet and David Sagan for providing help with Bmad. This work was supported by the U.S. National Science Foundation under Award PHY-1549132, the Center for Bright Beams.

REFERENCES

[1] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey", *J. Mach. Learn. Res.*, vol. 18, no. 153, pp. 1-43, Apr. 2018. doi:10.48550/arXiv.1502.05767

[2] T. Dorigo *et al.*, "Toward the end-to-end optimization of particle physics instruments with differentiable programming: a white paper", *arXiv preprint*, Mar. 2022. doi:10.48550/arXiv.2203.13818

[3] R. Roussel and A. Edelen, "Applications of differentiable physics simulations in particle accelerator modeling", *arXiv preprint*, Nov. 2022. doi:10.48550/arXiv.2211.09077

[4] R. Roussel *et al.*, "Differentiable Preisach modeling for characterization and optimization of particle accelerator systems with hysteresis", *Phys. Rev. Lett.*, vol. 128, no. 20, p. 204801, May 2022. doi:10.1103/PhysRevLett.128.204801

[5] M. Berz, "Differential algebraic description of beam dynamics to very high orders", *Part. Accel.*, vol. 24, pp. 109-124, 1989.

[6] K. Makino and M. Berz, "COSY INFINITY version 8", *Nucl. Instrum. Methods Phys. Res. Sect. A*, vol. 427, pp. 338-343, May 1999. doi:10.1016/S0168-9002(98)01554-X

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-WEPA065

[7] J. Qiang, "Differentiable self-consistent space-charge simulation for accelerator design", *Phys. Rev. Accel. Beams*, vol. 26, p. 024601, Feb. 2023. doi:10.1103/PhysRevAccelBeams.26.024601

[8] https://github.com/bmad-sim/Bmad-X

[9] D. Sagan, "Bmad: a relativistic charged particle simulation library", *Nucl. Instrum. Methods Phys. Res. Sect. A*, vol. 558, pp. 356-359, Mar. 2006. doi:10.1016/j.nima.2005.11.001

[10] A. Paszke *et al.*, "PyTorch: an imperative style, high-performance deep learning library", in *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, Vancouver, BC, Canada, Dec. 2019, pp. 8024–8035.

[11] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization", *arXiv preprint*, Dec. 2014. doi:10.48550/arXiv.1412.6980

[12] A. Hanuka *et al.*, "Physics model-informed Gaussian process for online optimization of particle accelerators", *Phys. Rev. Accel. Beams*, vol. 24, no. 7, p. 072802, Jul. 2021. doi:10.1103/PhysRevAccelBeams.24.072802

[13] S. Kullback and R. A. Leibler, "On information and sufficiency", in *Ann. Math. Stat.*, vol. 22, no. 1, pp. 79-86, Mar. 1951.

[14] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function", in *Ann. Math. Stat.*, vol. 27, no. 3, pp. 832-837, Sep. 1956.

[15] 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. 145001, Apr. 2023. doi:10.1103/PhysRevLett.130.145001