ISBN: 978-3-95450-247-9 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2024-WEPR59

DEVELOPING NESTED AUTO-DIFFERENTIATION TRACKING CODE FOR BEAM DYNAMICS OPTIMIZATION*

J. Wan, Y. Hao†, C. Ratcliff
Facility for Rare Isotope Beams, Michigan State University, East Lansing, MI, USA
J. Qiang, Y.-K. Kan
Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, CA, USA

Abstract

An innovative particle tracking code is in development using the Julia programming language, utilizing the power of auto-differentiation (AD). With the aid of Enzyme, a AD tool working at LLVM level, this code enables automatic calculation of derivatives for particle tracking simulation results and coefficients of truncated power series algebra (TPSA) with respect to selected parameters of interest. This tracking code provides a flexible and powerful solution for accelerator physicists applicable across various research topics, especially for beam dynamics optimization.

INTRODUCTION

The calculation of derivatives is important in accelerator physics simulations. Traditional approaches to calculate derivatives primarily involve numerical and symbolic methods. While numerical methods can approximate derivatives for complex systems, they often suffer from precision limitations due to truncated errors and round-off errors, especially for ill-conditioned problems. On the other hand, symbolic methods, though accurate, struggle with deriving analytical expressions for complicated systems. To address these challenges, automatic differentiation (AD) emerges as a robust alternative [1]. AD provides a mean to computationally derive exact derivatives efficiently, bypassing the difficulties associated with numerical instability of numerical methods and algebraic complexity of symbolic methods.

AD is a computational technique used to evaluate the derivatives of functions in computer programs. Unlike numerical differentiation and symbolic differentiation, AD calculate derivatives of functions by decomposing functions into elementary operations and applying the chain rule to these operations sequentially. This method not only ensures computational precision, but also enhances performance by avoiding the complexity and overhead associated with symbolic methods. So far, AD has been a fundamental tool in various fields such as machine learning, optimization, and numerical simulation [2].

The JuTrack code is a novel accelerator modeling package developed in the Julia language. This package is specifically designed for numerical simulation of particle tracking using symplectic integration [3]. In addition to standard particle tracking, JuTrack also supports the computation of Truncated Power Series Algebra (TPSA) [4]. A powerful LLVM-level AD tool, Enzyme [5], is implemented in this code for swift and precise derivative computation at the compiler level.

ACCELERATOR MODELING

Figure 1 shows the structure of the JuTrack code. It includes four main parts, lattice functions, TPSA functions, tracking functions and utility functions. As a particle tracking tool, JuTrack supports 6-D tracking of the particle coordinates $[x, p_x, y, p_y, z, \delta]$, where x and y are the transverse coordinates, p_x and p_y are the transverse momentum, $z = \beta c\tau$ is the path lengthening with respect to the reference particle, and $\delta = dE/E_0$ is the energy deviation with respect to the reference particle.

In addition to standard particle tracking, JuTrack also supports calculation of TPSA. The 6-D coordinates $[x, p_x, y, p_y, z, \delta]$ can be represented as six polynomials and be tracked as standard 6-D particle coordinates. The results of the TPSA tracking are also six polynomials representing the 6-D coordinates.

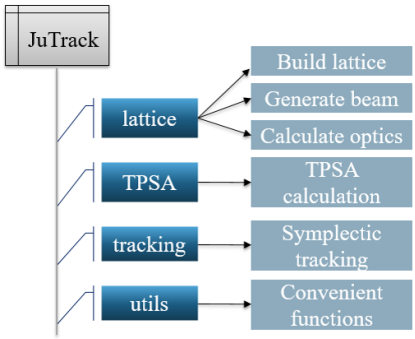

Figure 1: Structure of the JuTrack code.

Taking the Electron Storage Ring (ESR) of the Electron-Ion Collider (EIC) [6] as an example, in this section, we will demonstrate how to build such a complex accelerator lattice and how to calculate the accelerator parameters of the ring in JuTrack. The ring lattice consists of various types of elements, such as drift space, dipoles, quardupoles, sextupoles, thin multipoles, correctors, solenoids, radio frequency (RF) cavities and crab cavities. Each of these components has been developed and integrated into the package for comprehensive simulation capabilities.

Figure 2 shows the optics of the storage ring. The Twiss parameters are solved based on the transfer matrix, which is calculated from the first-order TPSA tracking. Compared to

* Work supported by DOE office of science, with award number DE-SC0024170.
† haoy@frib.msu.edu

ISBN: 978-3-95450-247-9 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2024-WEPR59

the MADX [7], a well-known accelerator modeling tool used for the EIC design, the maximum difference of β_x and β_y is around $5e^{-8}$ m. The tunes of the betatron oscillation are 50.0800/46.1400 for the JuTrack lattice, which is identical to the original MADX lattice.

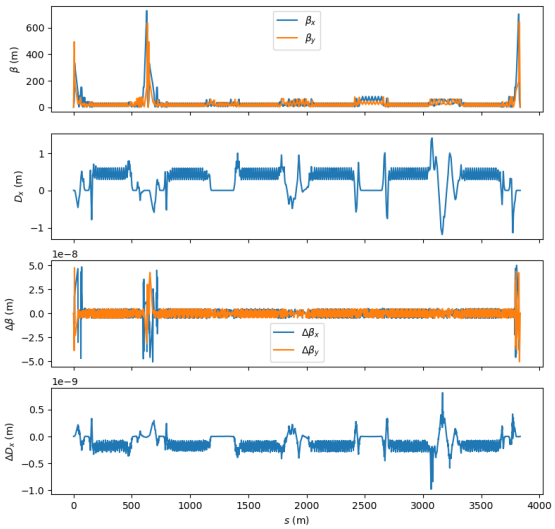

Figure 2: Beta functions and dispersion of the ESR lattice calculated with JuTrack and the difference compared with results of MADX.

AUTO DIFFERENTIATION

Derivatives of Particle Tracking Results

With the assistance of Enzyme, JuTrack allows us to automatically obtain derivatives of functions. The output of a differentiated function can be any differentiable lattice parameters and simulation solutions. The input of the function are usually design parameters of the accelerator, such as parameters of magnets and RF cavities.

Taking a simple transport line [Q1, D1, Q2, D2] as an example, we simulate the trajectory of a particle. Q1 and Q2 represent quadrupoles, and D1 and D2 represent drift space. The particle starts with initial coordinates [$1e^{-3}$, $1e^{-4}$, 0, 0, 0, 0] and is tracked through the sequence of the transport line's elements. The strengths of the two quadrupoles, k_1 and k_2, are used as input variables of the tracking function. The output variables are the final 6-D coordinates of the particle.

Table 1 shows the automatically obtained derivatives of the final 6-D coordinates with respect to k_1 and k_2 when $k_1 = 1.0\,\mathrm{m}^{-2}$ and $k_2 = -1.0\,\mathrm{m}^{-2}$. The results are proven to be true by comparing it with numerical differentiation. The derivatives of vertical coordinates are zero because the particle is initially positioned at the center of the quadrupole in the vertical plane.

Table 1: Derivative of Single Particle Tracking Results $k_1 = 1.0\,\mathrm{m}^{-2}$ and $k_2 = -1.0\,\mathrm{m}^{-2}$

	dx	dp_x	dy	dp_y	dz	$d\delta$
dk_1	$-3.1e^{-3}$	$-2.1e^{-3}$	0	0	$2.5e^{-6}$	0
dk_2	$8.4e^{-5}$	$1.5e^{-4}$	0	0	$-8.1e^{-6}$	0

Derivatives of Twiss Parameters

In this section, we demonstrate how to obtain derivatives of Twiss paramters for a storage ring and how to optimize the Twiss parameters using the derivatives.

A 3-rd generation storage ring light source is chosen as an example. The ring lattice consists of 20 identical double-bend-achromatics (DBAs) where the β_x at the center of long straight section is 10 m. The optimization task is to optimize the β_x at the center of long straight section to be 7 m by tuning k_1, which is the strength of the first quadrupole in the DBA cell. The derivative of β_x with respect to k_1 can be automatically obtained with AD. See in Fig. 3, the value of k_1 is adjusted based on a simple gradient descent optimization strategy, $k_1 = k_1 - \mathrm{step} \times d\beta_x/dk_1$, where step is set to be $1e^{-4}$. Within 10 steps, the β_x is optimized to the desired value.

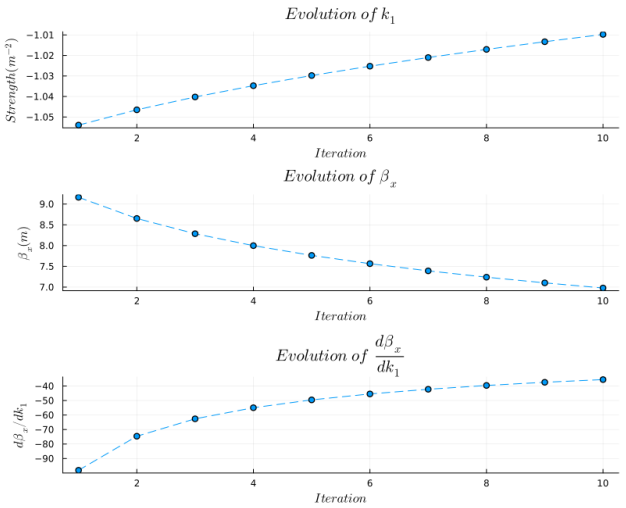

Figure 3: Optimization of β_x by tuning k_1 based on the derivative $d\beta_x/dk_1$.

CONCLUSION

An AD-enabled accelerator modeling code is in development based on Julia. The capabilities of this AD code are demonstrated through its application to a single-particle tracking scenario. Furthermore, we demonstrate how to optimize Twiss parameters of a storage ring accelerator using the automatically calculated derivatives.

The package is available at `https://github.com/MSU-Beam-Dynamics/JuTrack.jl.git`. This package is still being actively improved, with the aim of delivering a comprehensive AD-enabled tool for accelerator modeling.

ISBN: 978-3-95450-247-9 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2024-WEPR59

ACKNOWLEDGEMENTS

This work is supported by DOE office of science, with award number DE-SC0024170.

REFERENCES

[1] C. C. Margossian, "A review of automatic differentiation and its efficient implementation", in *Wiley interdisciplinary reviews: data mining and knowledge discovery*, vol. 9, p. e1305, 2019. http://jmlr.org/papers/v18/17-468.html

[2] 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, pp. 1–43, 2018. http://jmlr.org/papers/v18/17-468.html

[3] E. Forest and R. D. Ruth, "Fourth-order symplectic integration", *Physica D*, vol. 43, pp. 105–117, 1990. doi:10.1016/0167-2789(90)90019-L

[4] M. Berz, *Modern Map Methods in Particle Beam Physics*, Academic Press, 1999. http://bt.pa.msu.edu/pub

[5] W. Moses and V. Churavy, "Instead of rewriting foreign code for machine learning, automatically synthesize fast gradients", in *Advances in neural information processing systems*, vol. 33, pp. 12472–12485, 2020. https://proceedings.neurips.cc/paper_files/paper/2020/file/9332c513ef44b682e9347822c2e457ac-Paper.pdf

[6] A. Accardi *et al.*, "Electron-Ion Collider: The next QCD frontier: Understanding the glue that binds us all", *Eur. Phys. J. A*, vol. 52, pp. 1–100, 2016. doi:10.1140/epja/i2016-16268-9

[7] H. Grote and F. Schmidt, "MAD-X – An Upgrade from MAD8", in *Proc. PAC'03*, Portland, OR, USA, May 2003, paper FPAG014, pp. 3497–3499. doi:10.1109/PAC.2003.1289960