SCIBMAD: A DIFFERENTIABLE, GPU-PARALLELIZED SOFTWARE LIBRARY FOR PARTICLE ACCELERATOR DESIGN, NONLINEAR ANALYSIS, AND MACHINE LEARNING*

M. G. Signorelli†[1], J. P. Devlin[2], G. H. Hoffstaetter[1,2], D. Sagan[2]
[1]Brookhaven National Laboratory, Upton, NY, USA
[2]CLASSE, Cornell University, Ithaca, NY, USA

Abstract

We present SciBmad, a new accelerator physics software library developed to meet the needs of all accelerator design, analysis, and virtual modeling. SciBmad consists of a set of modular packages featuring fully differentiable and CPU/GPU-parallelized symplectic integrators, spin tracking and radiation, flexible and differentiable lattice definitions, nonlinear normal form analysis tools with Lie algebraic methods, batch parameter parallelization, and more. It is fully usable in Julia and easily callable from Python, making it easy to integrate it with external optimizers and machine learning frameworks. All releases of SciBmad undergo rigorous, automated testing with maximal code coverage, and all integrators are validated against PTC. With a growing list of features and contributors, SciBmad aims to be a powerful tool for any accelerator physics application.

INTRODUCTION

Since the dawn of the computer, the world has had no shortage of accelerator physics codes: from way back to SYNCH and RACETRACK [1], to the early differentiable codes COSY-INFINITY and the Polymorphic Tracking Code (PTC) [2–4], up to more recent frameworks such as MAD-NG, Xsuite, and Cheetah [5–7]. Many of them, at least implicitly, put some claim to being the "best". In this paper, we will follow this time-honored tradition, and provide the world with yet another accelerator physics code, which we also feel, in our clearly unbiased opinion, is the "best". We leave it up to you, dear reader, to decide.

It goes without saying that all of these softwares have pioneered new and useful features. Many machines have been designed and commissioned using all combinations of them. With 50+ years of codes, accumulated experience, and theory, the needed features of an accelerator physics code are, to a degree, understood now; standing on the shoulders of giants, a wishlist for a "dream" code might be:

- ❏ Natural syntax and easy to use in high-level programming languages (e.g. Python or Julia)
- ❏ Flexible accelerator lattice definitions including arbitrary parameter interdependencies
- ❏ Super fast (CPU/GPU parallelized) 6D symplectic, nonparaxial particle tracking including spin and radiation
- ❏ CPU/GPU parallelization over different lattices
- ❏ Fully forwards-/backwards-/Taylor differentiable to extract gradients w.r.t. anything for optimization/analysis
- ❏ Nonlinear parametric normal form analysis tools
- ❏ Modular and easy to integrate with other programs, optimizers, machine learning frameworks, etc

While many existing accelerator physics codes excel at various subsets of these, to our knowledge none simultaneously satisfy all. With the Electron-Ion Collider (EIC) design at Brookhaven National Laboratory progressing rapidly, the need to do so has only grown. To that end, inspired by all codes past and present, we introduce SciBmad.

SCIBMAD

SciBmad is a new accelerator physics software library that, in the spirit of classical Bmad [8], consists of a set of modular packages currently including:

1. **BeamTracking.jl:** Universally polymorphic/differentiable, parallelized integrators for simulating charged particles on the CPU and various GPUs (NVIDIA CUDA, Apple Metal, Intel oneAPI, and AMD ROCm)
2. **Beamlines.jl:** Fast, differentiable, and fully-featured accelerator lattice definitions and deferred expressions
3. **NonlinearNormalForm.jl:** Perturbation theory of differential-algebraic maps, including spin and damping, using Lie algebraic methods [9, 10]
4. **GTPSA.jl** Fast high-order automatic differentiation using the Generalised Truncated Power Series Algebra (GTPSA) library in MAD-NG [5, 11, 12]
5. **FundamentalFrequencies.jl:** GPU-batchable numerical analysis of fundamental frequencies [13]
6. **AtomicAndPhysicalConstants.jl:** Physical constants and properties for any atomic or subatomic particle

* Work supported by the U.S. Department of Energy under Contracts No. DE-SC0012704 (Brookhaven Science Associates, LLC) and No. 89233218CNA000001 (Office of Science, ARDAP), and by resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility, under award HEP-ERCAP0034811.
† msignorel@bnl.gov

SciBmad is written in Julia, which is a just-in-time (JIT) compiled, high-level, high performance computing language leveraging multiple dispatch and a powerful type system; these features enable SciBmad to be differentiable without sacrificing performance in other areas, and allow its integrators to be compiled, as-is, to a wide variety of hardwares, without separate CPU and GPU implementations [14]. Julia's plotting packages [15], optimizers, automatic-differentiation packages [16, 17], and machine learning tools [18] are immediately usable with SciBmad. Julia is easily callable from Python [19], and a dedicated SciBmad Python frontend with PyTorch bindings is also in development [20].

Flexible Lattice Definitions

`Beamlines.jl` aims to provide the fully-featured, expressive accelerator lattice definitions of both the Particle Accelerator Language Standard (PALS) [21] and classical Bmad. At the time of this paper writing, it supports arbitrary element placements/orientations (misalignments, patches), arbitrary-order magnetic multipoles, arbitrary bend geometries, standing/traveling wave RF cavities, reference energy/species changes, mechanical apertures, arbitrary s- and t-dependent electromagnetic fields as functions, and custom transport maps (e.g. a neural network surrogate).

SciBmad does not have any "bookkeeper". Instead, all dependent variables are computed *lazily* - on the fly, only when you need them. This, inspired by MAD-NG, minimizes overhead from storing/computing quantities you don't need, and removes the need for a bookkeeper to update all dependent variables, easing long-term maintainence. One can use lazily-evaluated deferred expressions via SciBmad's `DefExpr` type, which acts like a lazily-evaluated number, to define arbitrarily complex and infinitely-nested parameter dependencies. This is best shown with an example:

```
using SciBmad

kquad = 0.36
kqf = DefExpr(() -> kquad) # lambda function
kqd = -kqf # kqf acts like a number

qf = Quadrupole(Kn1 = kqf, L = 0.5)
d = Drift(L = 1.2)
qd = Quadrupole(Kn1 = kqd, L = 0.5)

fodo = Beamline([qf, d, qd, d], E_ref = 18e9,
        species_ref = Species("electron"))

# Updating kquad will "update" qf and qd:
kquad = 0.25
qf.Kn1 == -qd.Kn1 == 0.25 # true
```

Under the hood, all element "kinds" (drift, quadrupole, etc.) are actually *one single type*, to provide maximal flexibility. That means that there is nothing stopping you from doing:

```
d = Drift(L = 1.2)
d.Ks21 = -200 # Set 21st order skew multipole
```

This flexibility makes it easy to adjust the design on the fly. For example, in the EIC Electron Storage Ring (ESR), we need to add multipoles to the drifts in the interaction region to simulate field crosstalk from the Hadron Storage Ring. With SciBmad, one does *not* need to edit the lattice and change these "drifts" to "multipoles"; just set the multipole!

CPU/GPU Parallelized Tracking

`BeamTracking.jl` was designed from the ground up with CPU/GPU parallelization and differentiability as fundamental requirements. The resulting framework is structured so that contributors who follow a small set of conventions automatically get tracking kernels that satisfy both. To guarantee performance, every tracking kernel must pass rigorous testing under explicit CPU single instruction multiple data (SIMD), which directly manipulates the CPU's vector registers [22]. Because explicit SIMD does not permit branching - a performance killer on GPUs - we can guarantee that all of SciBmad's tracking is branch-free.

Symplecticity, spin tracking, and full coverage and correctness tests against PTC are also required for all tracking in SciBmad. Currently, SciBmad implements Yoshida's 2nd, 4th, 6th, and 8th order explicit and implicit integrators for various splits [23]. By default, each element's parameters are unpacked, a most appropriate split is decided, and then a lower-level compiled kernel is called. Benchmarks are included in the Examples section, and for more see [24].

Batch/Time-Dependent Parameters

In the same way that SciBmad parallelizes particle tracking via SIMD, it can also SIMD-parallelize over the accelerator parameters, such as magnet strengths. E.g., one can set a sextupole strength to a vector of 100,000 random values:

```
using CUDA
s=Sextupole(Kn2L=BatchParam(CUDA.rand(100000)))
```

For a bunch of 100,000 particles, each sees a different sextupole strength, all on a single CPU or GPU. Any number of elements and parameters can be set as `BatchParams`, enabling, for example, parallel dynamic aperture optimizations over sextupole settings, or parallel simulations of different magnet errors. Batch parameters were inspired by Cheetah.

SciBmad also allows one to set any accelerator parameter to be any function of time, using `Time`. E.g., an AC kicker:

```
v = VKicker(L=0.5, Ks0=1e-4*sin(1e6*Time()) )
```

The reference energy can also be set as time-dependent (for ramping). This is GPU compatible and SIMD-parallelizable.

Nonlinear Parametric Normal Forms

While `GTPSA.jl` provides the ability to compute high order parametric Taylor maps, `NonlinearNormalForm.jl` provides the Lie algebraic methods for analyzing these maps. Maps including radiation can be handled, and spin is handled

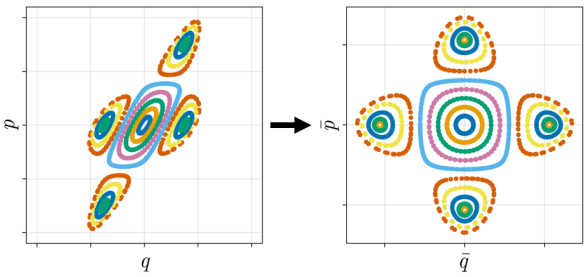

Figure 1: Poincaré section of a map near a 1/4 resonance, before (left) and after (right) a single resonance normal form transformation computed with SciBmad.

naturally by an extension of the Lie algebra [25, 26]. This is how SciBmad computes the fully-coupled lattice functions (Twiss parameters) and their nonlinear generalizations, amplitude dependent (spin) tunes, the invariant spin field (ISF), etc., as well as the gradients these quantities w.r.t. parameters (e.g. $\frac{\partial^2 Q_1}{\partial J_3 \partial(\text{sext. strength})}$). Figure 1 shows an example of a single resonance normal form computed with SciBmad. The package was developed in close collaboration with E. Forest, the developer of PTC and the Fully Polymorphic Package [27]. One of the primary goals of SciBmad was to make such powerful methods more accessible; as such, these quantities are all available via SciBmad's `twiss`.

EXAMPLES

Dynamic Aperture in the EIC-ESR

Here we present SciBmad's `dynamic_aperture` program with an 18 GeV EIC-ESR lattice, which has 6271 elements. All elements are integrated through using a 4th order Yoshida scheme with 1 step, including radiation damping. A bunch of 20,746 particles were initialized, and tracked for 2,000 turns (approximately 4 damping times); Fig. 2 took **8.2 min** to produce on a single NVIDIA A100 40GB GPU.

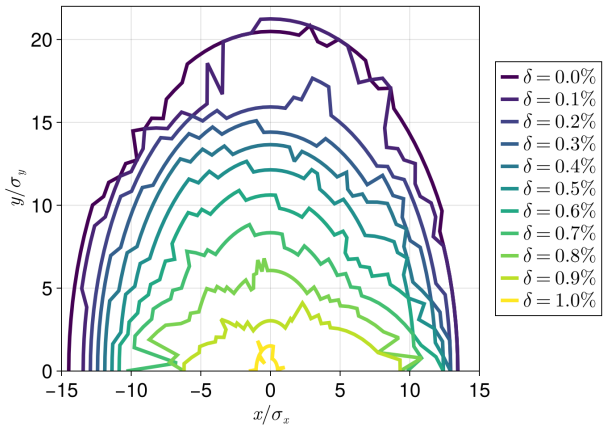

Figure 2: Acceptance of an 18 GeV EIC-ESR lattice, computed in 8.2 min on the GPU using SciBmad.

Radiative Spin Depolarization in the EIC-ESR

The spin-orbit coupling function $\frac{\partial \hat{n}}{\partial \delta}$, which quantifies radiative depolarization, is just one coefficient of the ISF

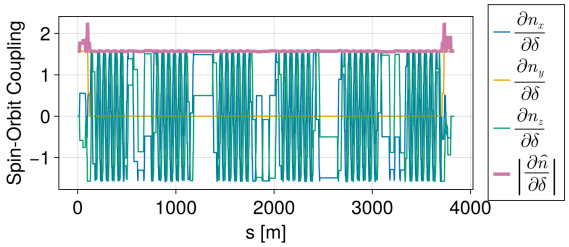

Figure 3: Spin-orbit coupling function in an 18 GeV EIC-ESR lattice computed using SciBmad's normal form.

Taylor series returned by `twiss`, and is shown in Fig. 3 for the same EIC-ESR lattice in the previous section. While this roughly approximates the depolarization time τ_{dep}, a better estimate is obtained from spin tracking with radiation; a polarized bunch at radiative equilibrium is initialized, tracked, and the slope of P vs. t approximates τ_{dep} [28, 29]. We spin-tracked 25,000 particles for 5,000 turns on the GPU through this EIC-ESR lattice, with radiation damping and fluctuations, and a 4th order 1-step Yoshida for all elements. SciBmad tracks quaternions for each particle instead of spin 3-vectors; this allows one to evaluate the polarization for any initial spin distribution with only one tracking. Figure 4 shows the polarization with all initial spins vertical and all along the 1st order ISF returned from `twiss`, and took **36.8 min** on a single NVIDIA A100 40GB GPU to produce.

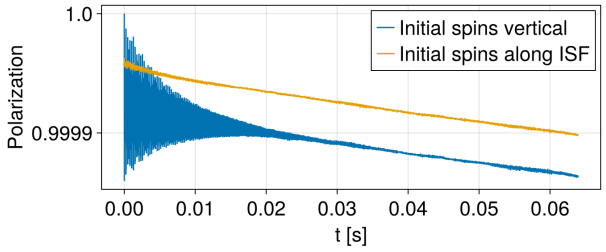

Figure 4: Polarization vs. t in an 18 GeV EIC-ESR, 25,000 particles × 5,000 turns, computed in 36.8 min on the GPU.

CONCLUSIONS AND OUTLOOK

SciBmad is a fully differentiable, CPU/GPU-parallelized accelerator physics library already being used for the EIC design. Its modular packages provide expressive lattice definitions, high-performance symplectic tracking (including batch and time-dependent parameters), and powerful nonlinear normal form analysis tools, making it suitable for a broad range of accelerator applications. Future work includes adding more collective effects modeling (currently only Gaussian intrabeam scattering is supported [30]), a high level optimization interface, and digital twin integration.

ACKNOWLEDGEMENTS

We thank all of the contributors to SciBmad. Especially, we thank Etienne Forest for his significant help with the normal form library, Laurent Deniau for his GTPSA library and for suggesting lazily-evaluated deferred expressions, and Ryan Roussel for suggesting batch parameters.

REFERENCES

[1] A. Wrulich, "RACETRACK: a computer code for the simulation of particle motion", DESY, Hamburg, Germany, Rep. DESY 84-026, 1984.

[2] M. Berz, *Modern map methods in particle beam physics*. San Diego: Academic Press, 1999.

[3] E. Forest, F. Schmidt, and E. McIntosh, "Introduction to the polymorphic tracking code: Fibre bundles, polymorphic Taylor types and "Exact tracking"", CERN, Geneva, Rep. CERN-SL-2002-044-AP, KEK-REPORT-2002-3, 2002. http://cds.cern.ch/record/573082

[4] E. Forest, "Geometric integration for particle accelerators", *Journal of Physics A: Mathematical and General*, vol. 39, no. 19, p. 5321, Apr. 2006. doi:10.1088/0305-4470/39/19/S03

[5] L. Deniau, "MAD-NG, a standalone multiplatform tool for linear and non-linear optics design and optimisation", 2025, doi:10.48550/arXiv.2412.16006,

[6] G. Iadarola *et al.*, "Xsuite: An Integrated Beam Physics Simulation Framework", *JACoW*, vol. HB2023, TUA2I1, 2024. doi:10.18429/JACoW-HB2023-TUA2I1

[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. 054601, May 2024. doi:10.1103/PhysRevAccelBeams.27.054601

[8] D. Sagan, "Bmad: a relativistic charged particle simulation library", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 558, no. 1, pp. 356–359, 2006. doi:10.1016/j.nima.2005.11.001

[9] E. Forest, *Beam dynamics*. CRC Press, 1998. doi:10.1201/9781315138176

[10] E. Forest, *From tracking code to analysis: generalised courant-snyder theory for any accelerator model*. Springer Japan: Springer Tokyo, 2016. doi:10.1007/978-4-431-55803-3

[11] M. G. Signorelli *et al.*, "GTPSA.jl: A SciBmad interface to the Generalised Truncated Power Series Algebra library", in *Proc. NAPAC2025*, Sacramento, CA, USA, pp. 145–148, Aug. 2025. doi:10.18429/JACoW-NAPAC2025-MOP044

[12] L. Deniau and C. Tomoiagă, "Generalised Truncated Power Series Algebra for Fast Particle Accelerator Transport Maps", in *6th International Particle Accelerator Conference*, MOPJE039, 2015. doi:10.18429/JACoW-IPAC2015-MOPJE039

[13] J. Laskar, "The chaotic motion of the solar system: a numerical estimate of the size of the chaotic zones", *Icarus*, vol. 88, no. 2, pp. 266–291, 1990. doi:10.1016/0019-1035(90)90084-M

[14] V. Churavy, KernelAbstractions.jl. doi:10.5281/zenodo.4021259

[15] S. Danisch and J. Krumbiegel, "Makie.jl: flexible high-performance data visualization for Julia", *Journal of Open Source Software*, vol. 6, no. 65, p. 3349, 2021. doi:10.21105/joss.03349

[16] F. Schäfer, M. Tarek, L. White, and C. Rackauckas, "Abstract-Differentiation.jl: Backend-Agnostic Differentiable Programming in Julia", 2022, doi:10.48550/arXiv.2109.12449,

[17] G. Dalle and A. Hill, "A Common Interface for Automatic Differentiation", 2025, doi:10.48550/arXiv.2505.05542,

[18] C. Rackauckas *et al.*, "Universal differential equations for scientific machine learning", *arXiv preprint arXiv:2001.04385*, 2020.

[19] C. Rowley, "PythonCall.jl: Python and Julia in harmony", 2022, https://github.com/JuliaPy/PythonCall.jl,

[20] C. Ung, M. G. Signorelli, G. H. Hoffstaetter, and D. Sagan, "Accelerator Optimizations Using SciBmad with PyTorch", presented at IPAC'26, Deauville, Normandy, France, May 2026, paper THP5364, this conference.

[21] A. H. et al., "A community effort toward a particle accelerator lattice standard (pals)", in *Proc. North American Particle Accelerator Conference (NAPAC2025)*, Sacramento, CA, USA, pp. 350–353, Aug. 2025. doi:10.18429/JACoW-NAPAC2025-TUP004

[22] E. Schnetter and contributors, "SIMD.jl: explicit SIMD vectorization in Julia", https://github.com/eschnett/SIMD.jl, 2016,

[23] H. Yoshida, "Construction of higher order symplectic integrators", *Physics letters A*, vol. 150, no. 5-7, pp. 262–268, 1990.

[24] J. P. Devlin, M. G. Signorelli, G. H. Hoffstaetter, and D. Sagan, "Differentiable, CPU-/GPU-Parallelized, Symplectic Spin-Orbit Tracking in SciBmad", presented at IPAC'26, Deauville, Normandy, France, May 2026, paper THP5302, this conference.

[25] E. Forest, "Private extension of *From Tracking Code to Analysis*", unpublished.

[26] M. G. Signorelli, "Electron spin polarization preservation in the Electron-Ion Collider", Ph.D. thesis, Cornell University, Dec. 2025. doi:10.7298/676d-xv17

[27] E. Forest and F. Schmidt, "The "full polymorphic package" (fpp)", *SIGPLAN Fortran Forum*, vol. 20, no. 3, pp. 12–17, Dec. 2001. doi:10.1145/570822.570824

[28] D. P. Barber and G. Ripken, "Computer Algorithms and Spin Matching", in *Handbook of Accelerator Physics and Engineering*. 2nd Edition, World Scientific Publishing Co. Pte. Ltd., 2013.

[29] D. Sagan, *The Bmad Manual*. New York, NY, 2022. https://www.classe.cornell.edu/bmad/manual.html

[30] J. P. Devlin, B. Nash, D. Abell, and G. H. Hoffstaetter, "Simulation of Intrabeam Scattering with Nonlinear Damping and Diffusion in SciBmad", presented at IPAC'26, Deauville, Normandy, France, May 2026, paper THP5301, this conference.