Differentiable Beam Dynamics Simulation Codes

CHENRAN XU
APS, Argonne National Laboratory
2026-06-29, AI4MuC Virtual Mini-Workshop

Challenges of Modern Particle Accelerators

- Particle accelerators are among the most complex real-world scientific instruments
- Increasingly demanding performance requirements & narrower tolerated operation region

- Many tasks involve high-dimensional, coupled optimization and control problems:
  — Accelerator design with competing objectives and constraints
  — Online tuning of machine settings
  — Calibration of models against measurements

- Differentiable simulations provide gradient information to help navigating the high-dimensional parameter space

What is a differentiable beam dynamics simulation?

- Physics-based simulation that can also return derivatives of its outputs with respective to desired inputs using automatic differentiation (AD).
  - Inputs: machine settings, beam distribution, hidden model parameters
  - Outputs: beam size, emittance, orbit, etc.

- In practice, modern AD tools compute the gradient by applying chain rule through the computation graph, using forward or reverse accumulation depending on the scenario.

- Many libraries and programming languages now have AD supports

Enzyme-AD

Cheetah

A PyTorch-based high-speed beam simulation code to support ML applications [1]

- Full GPU supports and seamless integration of ML models built in PyTorch
- Open-source: https://github.com/desy-ml/cheetah
- Modular design, easy to use and extend

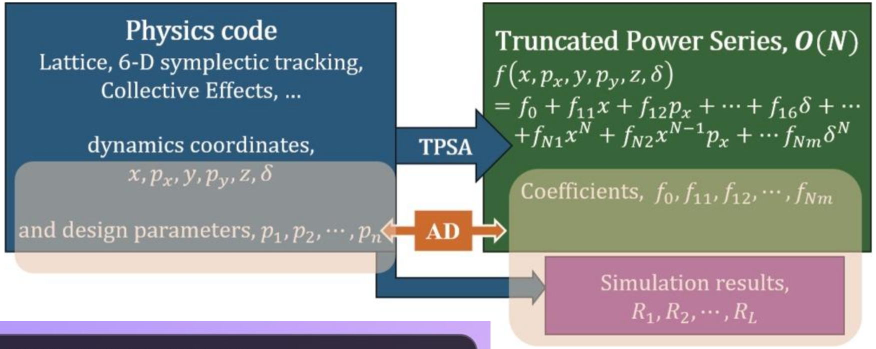
```
pip install cheetah-accelerator
```

```
# Load initial beam distribution from ASTRA tracking
beam_in = ParticleBeam.from_astra("beam_in.ini")

# Create a FODO lattice
segment = Segment(
    [
        Drift(length=torch.tensor(0.2)),
        Quadrupole(length=torch.tensor(0.2), name="Q1"),
        Drift(length=torch.tensor(0.4)),
        Quadrupole(length=torch.tensor(0.2), name="Q2"),
        Drift(length=torch.tensor(0.2)),
    ]
)

# Change the magnet strengths
segment.Q1.k1 = torch.tensor(10.0)
segment.Q2.k1 = torch.tensor(-9.0)

# Tracking through the segment
beam_out = segment.track(beam_in)
```

JuTrack.jl

A Julia package for auto-differentiable accelerator modeling and particle tracking

- 6D symplectic tracking
- Supports collective effects like space-charge, wakefield, and beam-beam interaction
- Open-source: https://github.com/MSU-Beam-Dynamics/JuTrack.jl



Get the derivative w.r.t. quadrupole k1 using TPSA

```julia
set_tps_dim(2) # 2 variables for fastTPSA
function tracking_wrt_k1(x1::DTPSAD{NVAR(), Float64}, x2::DTPSAD{NVAR(), Float64})
    D1 = DRIFT(len=DTPSAD(1.0))
    D2 = DRIFT(len=DTPSAD(1.0))
    Q1 = KQUAD(len=DTPSAD(1.0))
    Q2 = KQUAD(len=DTPSAD(1.0))
    Q1.k1 = x1
    Q2.k1 = x2
    beam = Beam(DTPSAD.([0.1 0.0 0.0 0.0 0.0 0.0]))
    LINE = [D1, Q1, D2, Q2]
    linepass!(LINE, beam)
    return beam.r
end
k1 = -0.9
k2 = 0.3
g, r = Jacobian(tracking_wrt_k1, [k1, k2], true)
```

J. Wan et al, J. CPC 2025

SciBmad.jl (Bmad-Julia)

A new high-performance, polymorphic, and differentiable accelerator physics simulation ecosystem in Julia.

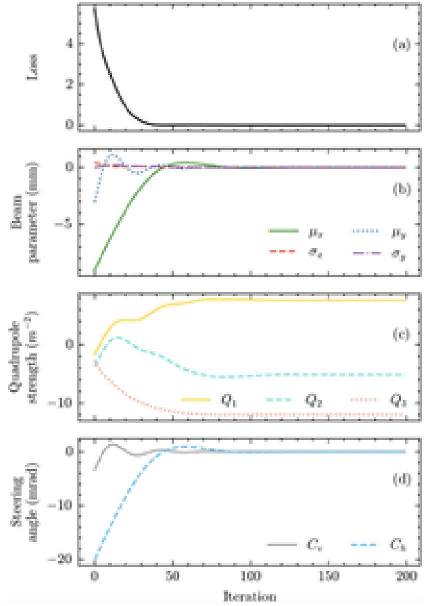
- Open source: https://github.com/bmad-sim/SciBmad.jl
- Inspired by the Bmad ecosystem, highly modularized packages
  - BeamTracking.jl : differentiable charged particle tracking
  - GTPSA.jl: high-order auto-diff using the GTPSA library
  - Beamlines.jl
  - AtomicAndPhysicalConstants.jl
  - NonlinearNormalForm.jl

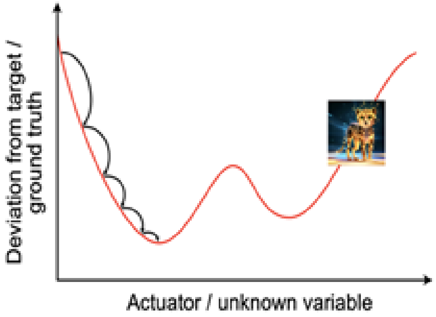
D. Sagan et al, IPAC 2024

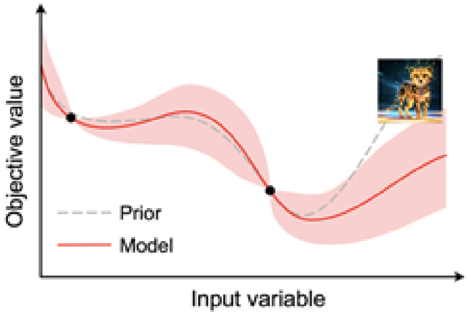
Application Examples

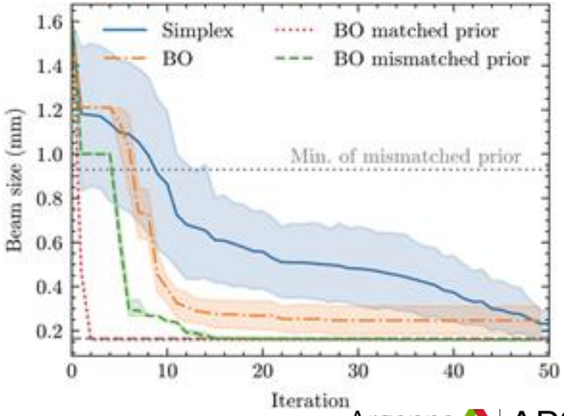
Example: Gradient-based Tuning and Optimization

- Tune magnet settings using the **gradient of the beam dynamics model** computed through **automatic differentiation**.
- Can be scaled up to **high-dimensional optimization**
- Act as physical-prior for sample-efficient optimization, such as Bayesian optimization





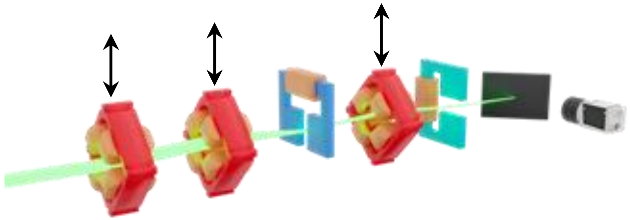




J. Kaiser, C. Xu et al. PRAB 2024

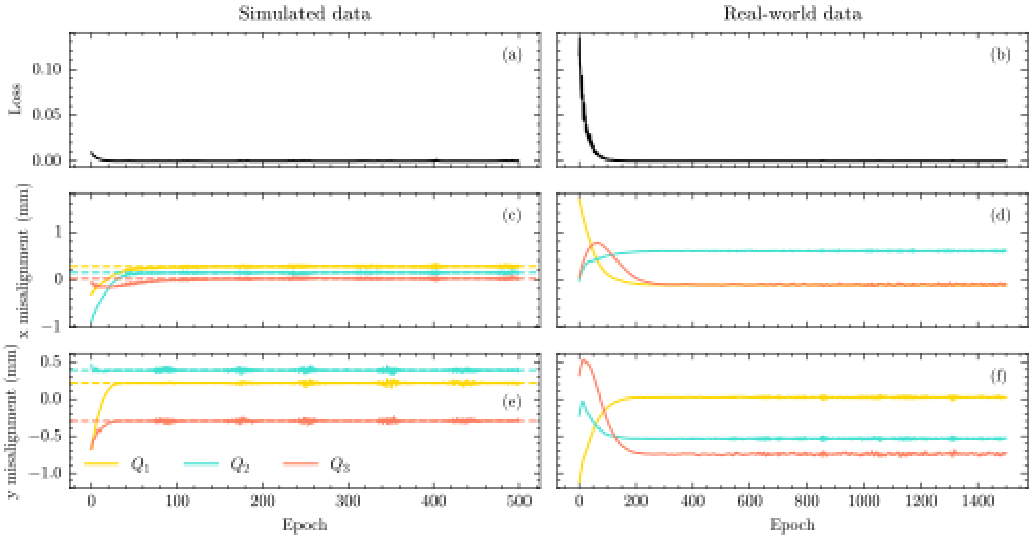
Example: Gradient-based System Identification

Quadrupole misalignments fitting



- Determine **hidden system properties** using the **gradient of the beam dynamics model** computed through **automatic differentiation.**
- Compared to BBA, this approach can also be applied **parasitically** on other data, e.g. from optimization runs or emittance measurements
- Possible integration with Bayesian Inference framework such as Pyro

```
ares_ea.AREAMQZM1.misalignment = nn.Parameter([0.0, 0.0])
ares_ea.AREAMQZM2.misalignment = nn.Parameter([0.0, 0.0])
ares_ea.AREAMQZM3.misalignment = nn.Parameter([0.0, 0.0])

optimizer = Adam(ares_ea.parameters())

for sample in dataset:
    set_magnets(ares_ea, sample.magnets)
    ougoing = ares_ea.track(incoming)
    loss = loss_fn(outgoing, sample.measurement)

    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
```

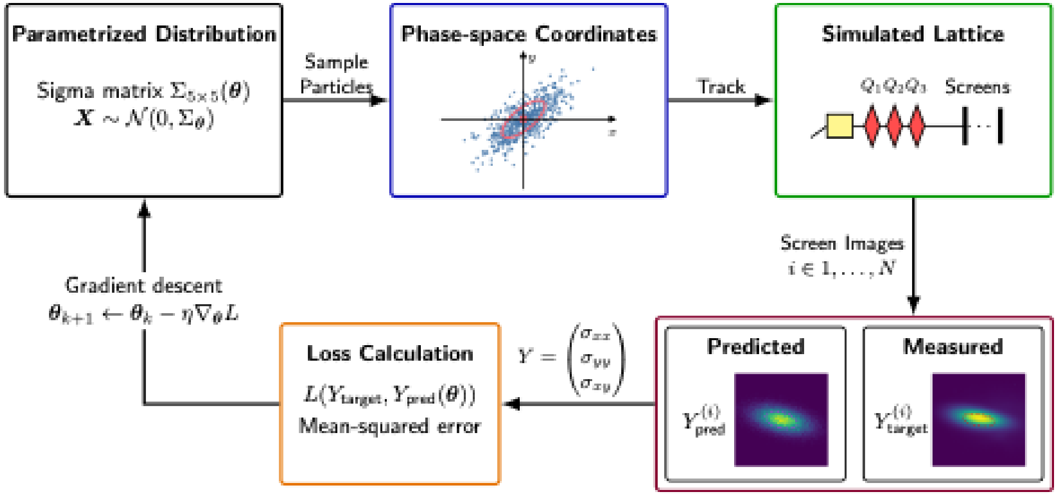


J. Kaiser, C. Xu et al. PRAB 2024

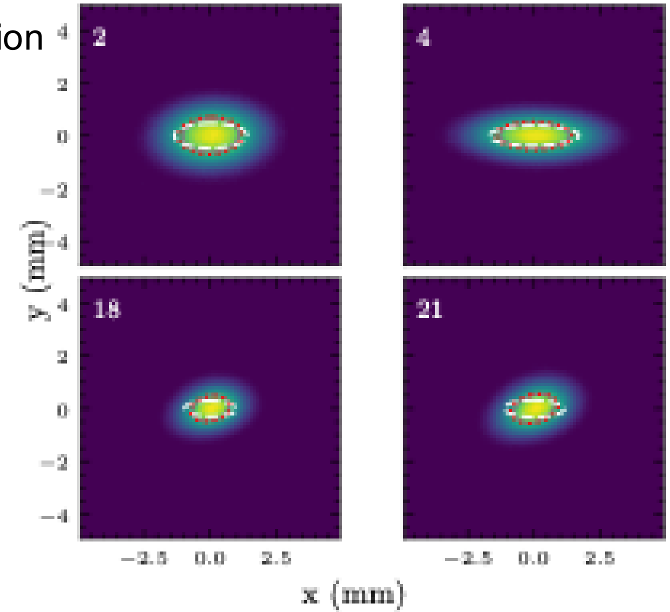
Example: Beam Characterization in Transport Lines

Measurement of 5x5 beam sigma matrix in Booster-to-Storage Ring (BTS) Line at APS

- Scan quadrupole magnets & measure transverse beam profiles on downstream screens
- Use differentiable-simulation to reconstruct the initial beam distribution

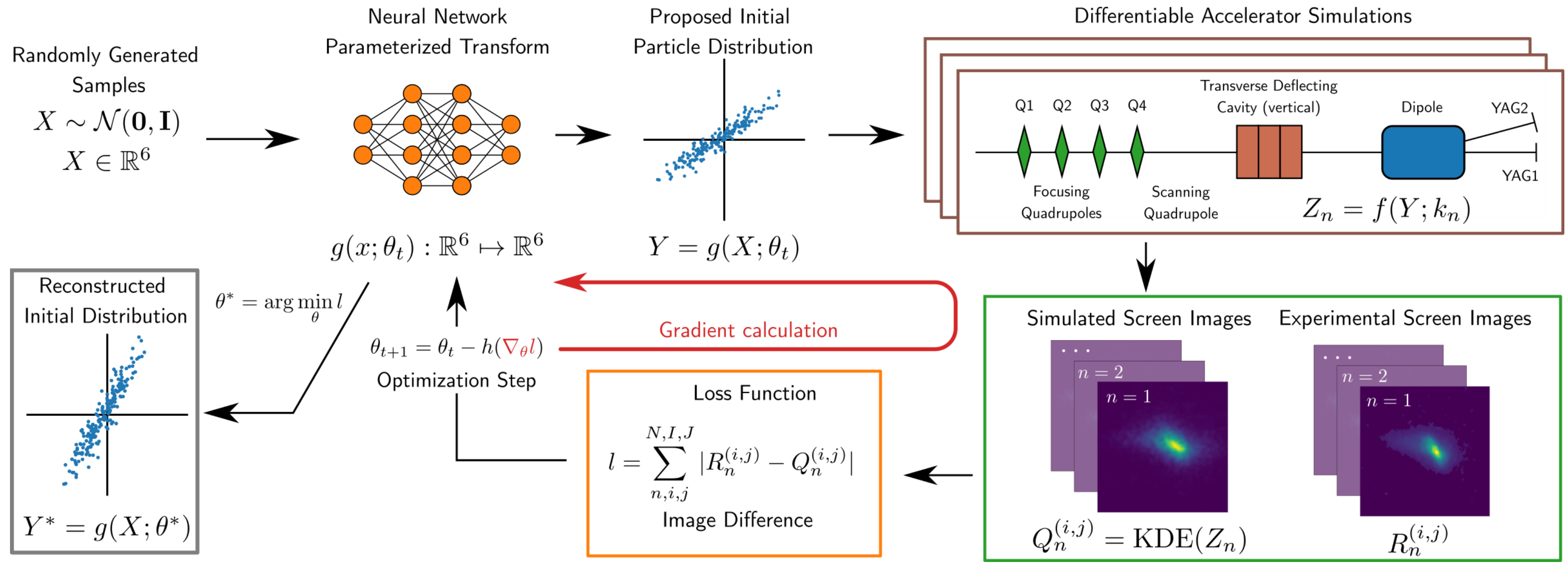


Predicted transverse beam profile before (white) and after (red) fitting



Example: Generative Beam Phase Space Reconstruction (GPSR)

Sample efficient reconstruction for up to 6D phase spaces



Package available at: https://github.com/roussel-ryan/gpsr

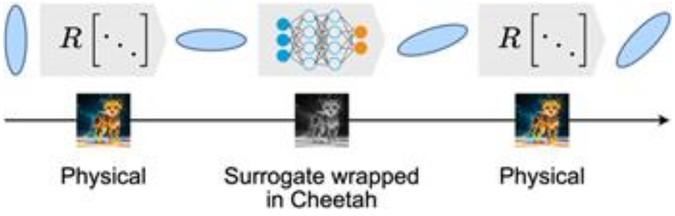
R. Roussel et al. 2023, PRL.130.145001
R. Roussel et al. 2025, PRAB.27.094601

Courtesy: Ryan Roussel

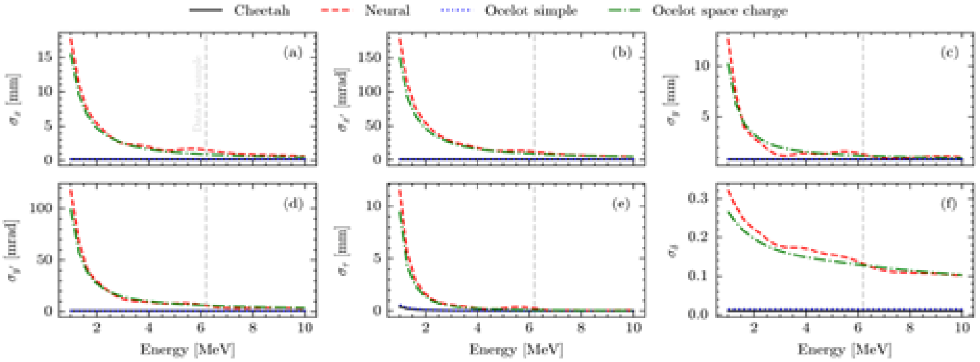
Example: Integrate Modular Neural Network Surrogate Models

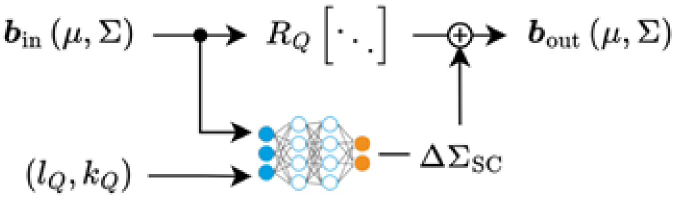
Modeling complex processes with surrogate models than can be reused

- Replace/**augment** Cheetah elements with neural network surrogates trained on high-fidelity simulations or real data.
- Neural networks implemented in PyTorch are effectively **native** to Cheetah. Easy integration & preserved end-to-end **differentiability.**



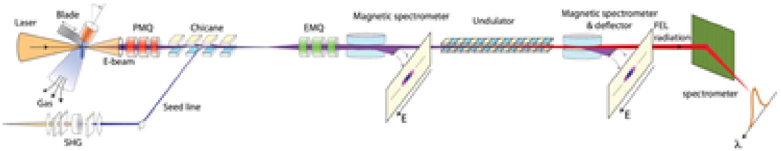

```python
class SCQuadrupole(Element):
    net = SCNet().load_state_dict(torch.load("weights.pth"))

    def track(self, incoming: Beam) -> Beam:
        return self.net(incoming)
```





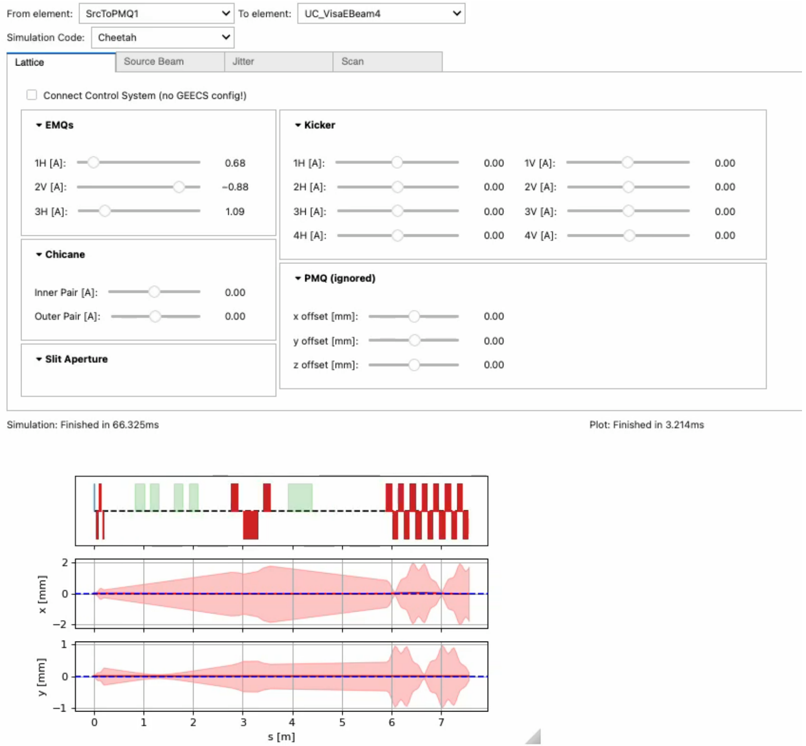
Efforts Towards Building a Digital Twin at LBNL (BELLA): Interactive Beam Dynamics Simulations

- Beam dynamics simulations of the BELLA HTU beamline using ImpactX or Cheetah **take < 1s** enabling **interactive use in the control room**



- LBNL team has built an **interactive interface** to explore accelerator behavior in simulations.

- The interface can connect to the control system and **read control parameters in real-time**, and update simulations accordingly.



Effort funded by an LBNL LDRD. The code is available at https://github.com/BLAST-AI-ML/bella_htu_digital_twin

Courtesy: Remi Lehe

Conclusion

Differentiable beam dynamics simulation codes enable:

- high-dimensional design optimization
- faster and safer online tuning
- model calibration and system identification
- beam characterization and phase-space reconstruction
- digital twins and virtual accelerators.

Joint development effort and community collaboration will be the key to enable new applications throughout the lifecycle of a particle accelerator

Acknowledgement

Cheetah Collaboration

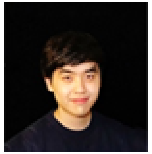
DESY.

Argonne NATIONAL LABORATORY | Advanced Photon Source

BERKELEY LAB


Jan Kaiser


Christian Hespe


Chenran Xu


Grégoire Charleux


Axel Huebl


Annika Eichler

UNIVERSITY OF LIVERPOOL

The Cockcroft Institute


A. Santamaría García


Amelia Pollard


Remi Lehe

THE UNIVERSITY OF CHICAGO

SLAC


J. P. Gonzalez-Aguilera


Ryan Roussel


Daniel Ratner


Auralee Edelen


Jenny Morgan


Zihan Zhu

The work to develop Cheetah 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). Further, we gratefully acknowledge funding by the EuXFEL R&D project "RP-513: Learning Based Methods". This work is also supported by the U.S. Department of Energy, Office of Science under Contract No. DE-AC02-76SF00515, the Center for Bright Beams, NSF Award No. PHY-1549132, and the U.S. DOE Office of Science-Basic Energy Sciences, under Contract No. DE-AC02-06CH11357. In addition, we acknowledge support from DESY (Hamburg, Germany) and KIT (Karlsruhe, Germany), members of the Helmholtz Association HGF as well as from the Hamburg Open Online University (HOOU) and the Science and Technology Facilities Council (UK).

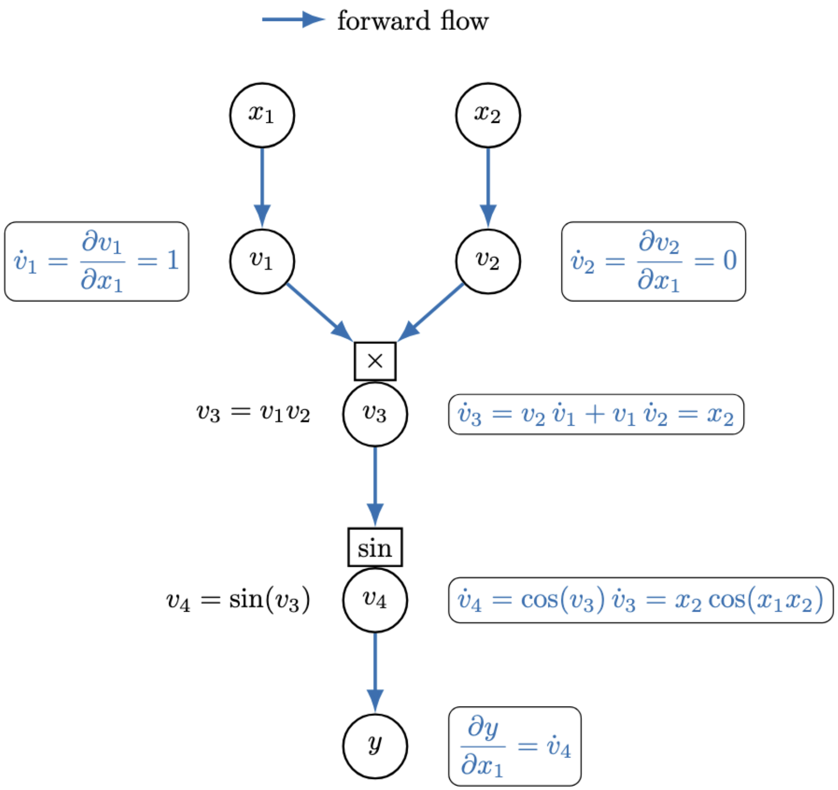
Backup Slides

Automatic Differentiation – Forward Mode

- Propagates the directional derivatives (tangents) alongside the intermediate values when moving forward through the computation graph
- Example:

calculate the derivative $\frac{\partial y}{\partial x_1}$ for

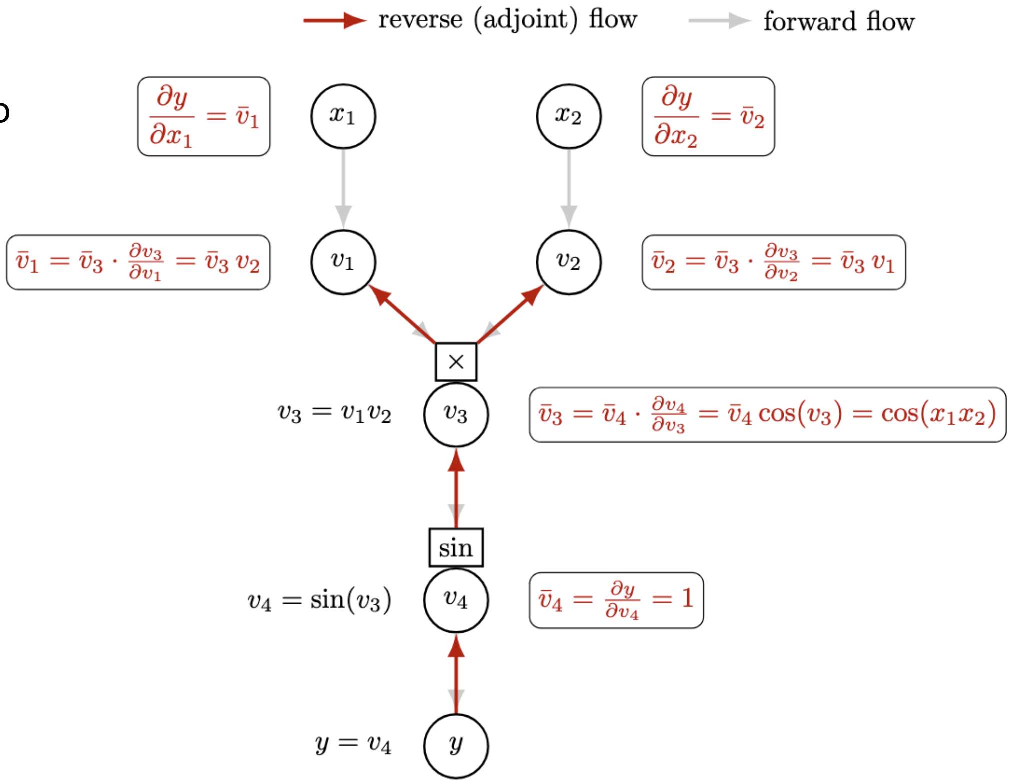
$$y = \sin(x_1 x_2)$$



Automatic Differentiation – Reverse Mode

- Propagates the adjoint values from the output to the input variables, after evaluating the function
- Example:

  calculate the derivative $\frac{\partial y}{\partial x_1}$ for $y = \sin(x_1 x_2)$



All the codes are under active development

	Cheetah	JuTrack	SciBmad
Programming Language	Python	Julia	Julia
JIT Compilation	✅	✅	✅
Reverse mode AD	✅	✅ Enzyme	✅ ReverseDiff.jl
Forward mode AD	✅ *experimental in Pytorch*	✅ TPSA + Enzyme	✅ ForwardDiff.jl, GTPSA.jl

- There are also other auto-differentiable beam simulation codes that do not consider AI/ML applications: COSY-Infinity, MAD-NG, …

How-To Achieve AD Beyond Transfer Maps

Example: Implementing Space Charge Effect in Cheetah

Parameter	Value
Beam charge Q	10 nC
Beam energy $\mathcal{E}$	250 MeV
Beam initial radius R_i	1 mm
Beam initial length L_i	1 mm
Length of the drift L	5.5 m
Number of macroparticles $N_{\mathrm{part.}}$	1,000
Number of grid cells N_{cells}	32^3
Numer of slices N_{slices}	3

- Differentiable SC kick using integrated Green's function (IGF) method
  - s-to-t transformation
  - **Cloud-in-cell charge-deposition** in grid points

$$\rho_{ijk} = \frac{1}{\Delta x \Delta y \Delta z} \sum_{p=1}^{N_{part.}} q_p S(\boldsymbol{x}_{ijk} - \boldsymbol{x}_p)$$

  - Field solver to obtain the Lorentz force

$$\partial_x^2 \phi + \partial_y \phi^2 + \frac{1}{\gamma^2} \partial_z^2 \phi = -\frac{\rho}{\epsilon_0}. \quad \boldsymbol{F} = -q \boldsymbol{\nabla} \phi / \gamma^2$$

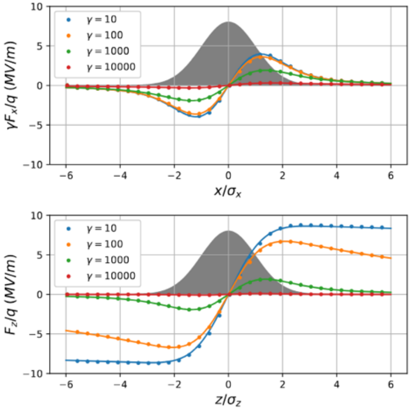
  - Field gather to interpolate the force to macroparticles

$$\boldsymbol{F}(\boldsymbol{x}_p) = \sum_{ijk} S(\boldsymbol{x}_p - \boldsymbol{x}_{ijk}) \boldsymbol{F}_{ijk}$$

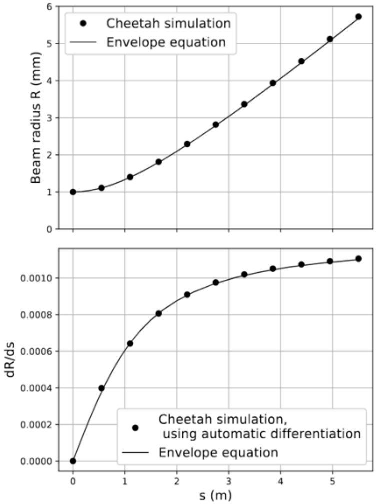
  - Apply kick
  - t-to-s transformation

Implementation Benchmarks

Gaussian beam at different energy



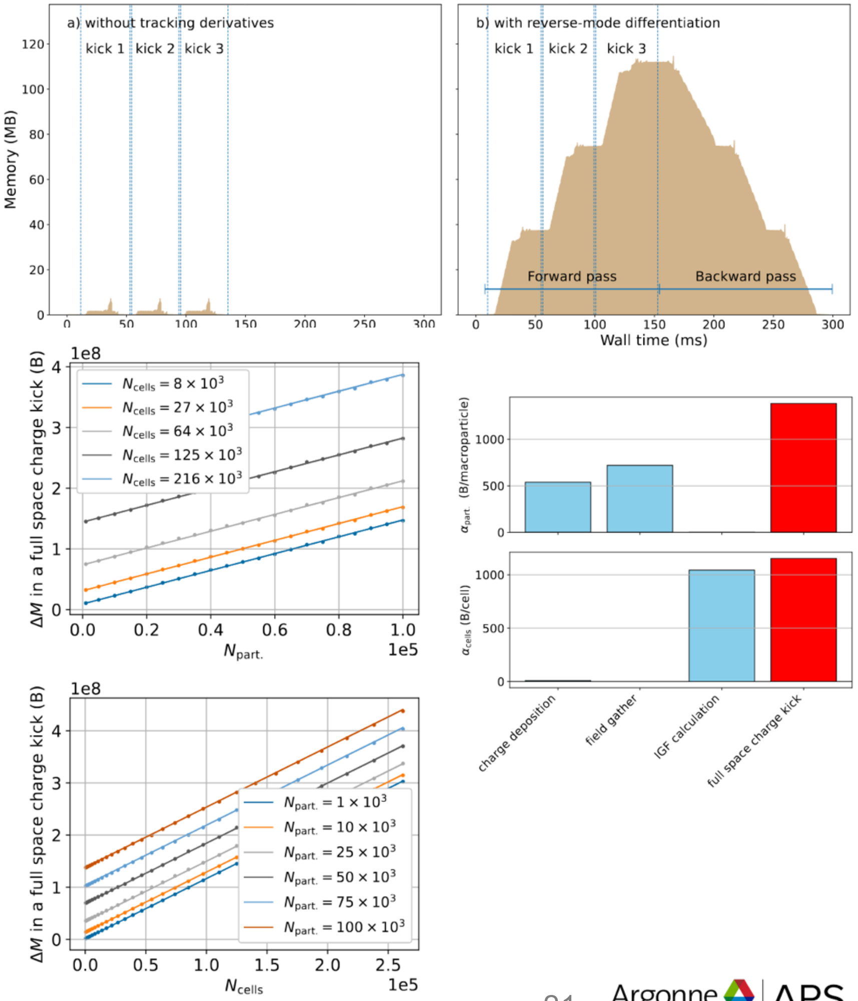
Cold beam expansion



Memory Scaling

Example: Space Charge Effect

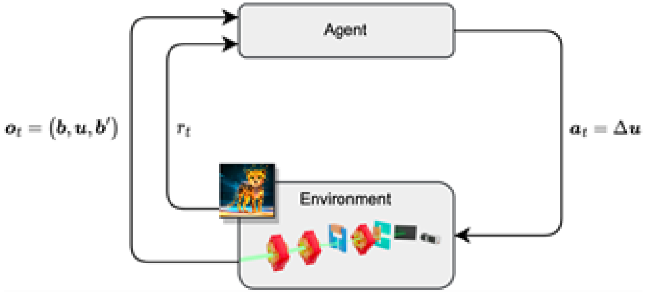
- Increased memory usage to keep track of gradient
- Linear scaling for both #particles and #grid cells
- For long term tracking with more computation steps, possible mitigation with gradient checkpointing

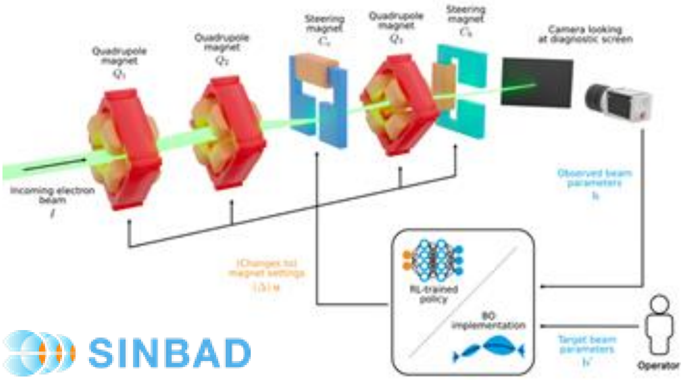


R. Lehe et al. PASC'26

Training Reinforcement Learning Controllers

Transverse beam tuning at ARES



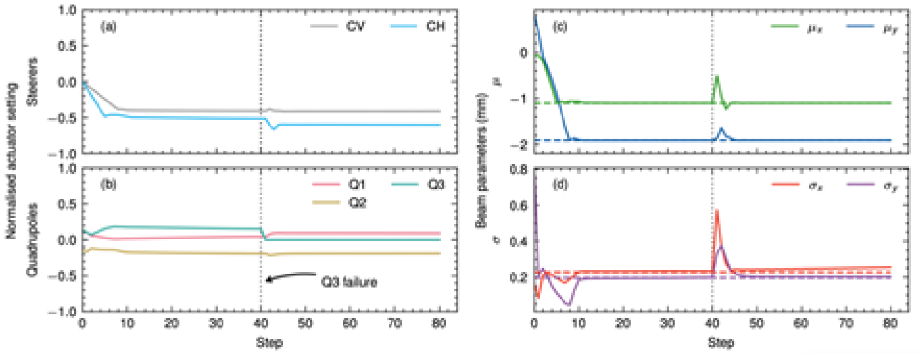


- RL is proven to be efficient in real-world control tasks, but training is extremely time-consuming (up to millions of interaction steps)

- Differentiable simulations can serve as low-cost training environment to (pre)train, which is more explainable than data-driven surrogate models

- The RL-trained optimization algorithm is deployed to the **real-world** with **zero-shot learning** thanks to **domain randomization**

RLO in ARES EA (including feedback)



J. Kaiser, C. Xu et al. Sci. Rep 2025

One Step Further: Gradient-based RL

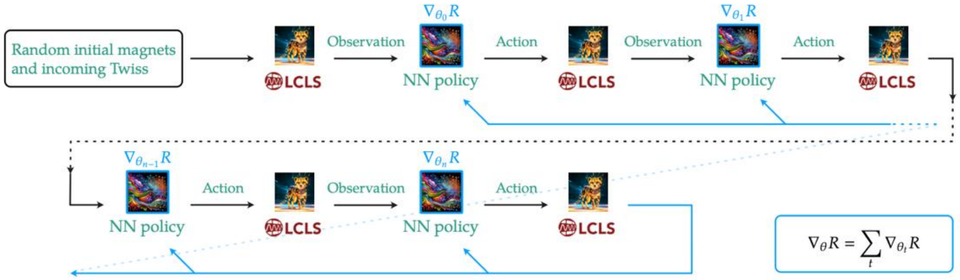
Example of LCLS FEL Tuning with 14 quadrupoles

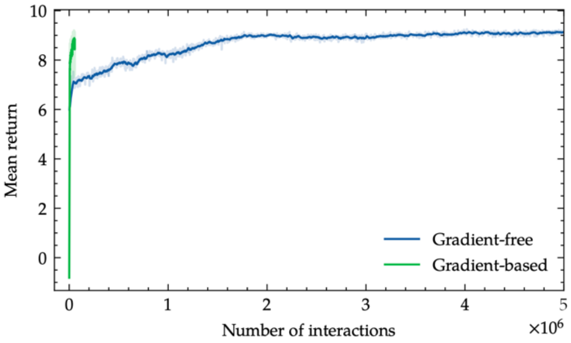
- The sample inefficiency of RL training is because it needs to **estimate** the expected return by applying certain action sequences

- Differentiable environment can provide direct gradient information for improving the RL policy

- Gradient-based RL can train **orders of magnitude** faster than the gradient-free variant

Roll-out the episode, accumulate rewards, and back-propogate to update the policy

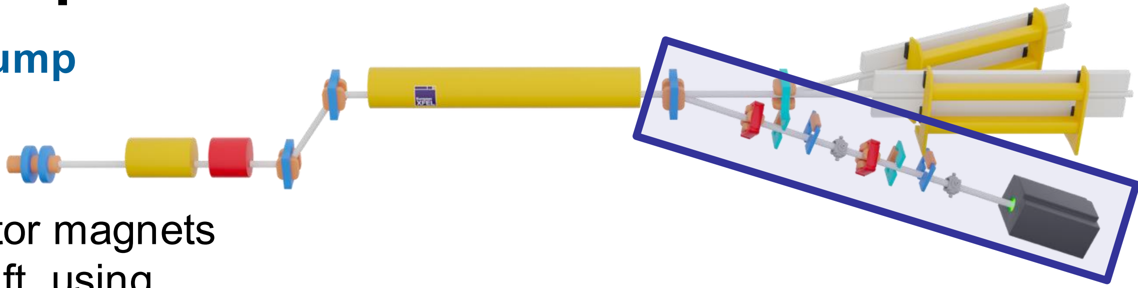




Courtesy: J. Kaiser, Dissertation 2025

Data-Driven Feedback Optimization

European-XFEL Electron Beam Dump



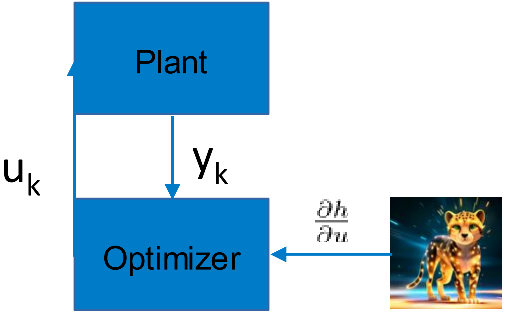
- Find optimal settings for 12 corrector magnets and compensate for continuous drift, using readings from 10 BPMS

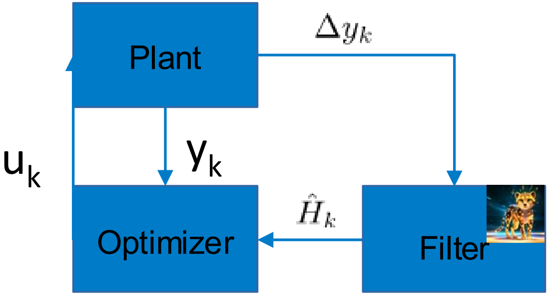
$$u_{k+1} = u_k - \alpha_k \left(\frac{\partial \Phi}{\partial u}(u_k, y_k) + \frac{\partial \Phi}{\partial y}(u_k, y_k) \boxed{\hat{H}_k} \right) + \boxed{\nu_k}$$

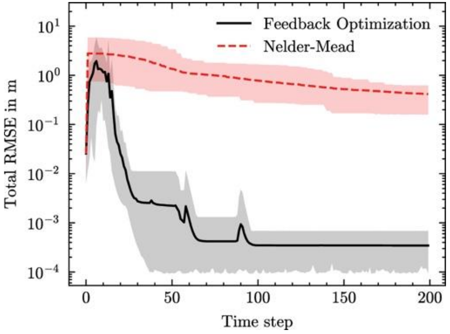
Initial sensitivity from Cheetah model, refine with Kalman filter

Random excitation signal

Model-based → **Data-driven**







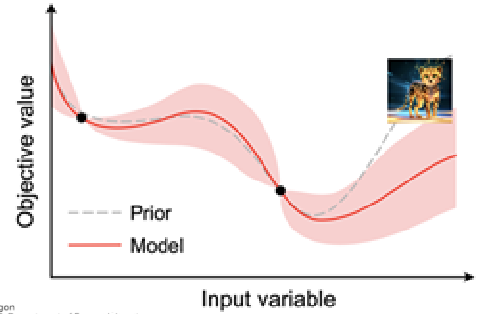
C. Hespe et al, Automatisierungstechnik 2025

Courtesy: Christian Hespe

Physics-based Prior Mean for Bayesian Optimisation

Combine Cheetah with BO

- A physics-informed prior can help **improve the performance of BO** by preventing over-exploitation.
- Differentiability allows **efficient acquisition function optimization** using gradient descent methods in modern BO packages like BoTorch.
- Can be used in **combination with gradient-based system identification** to overcome model inaccuracies.



FODO cell beam focusing example