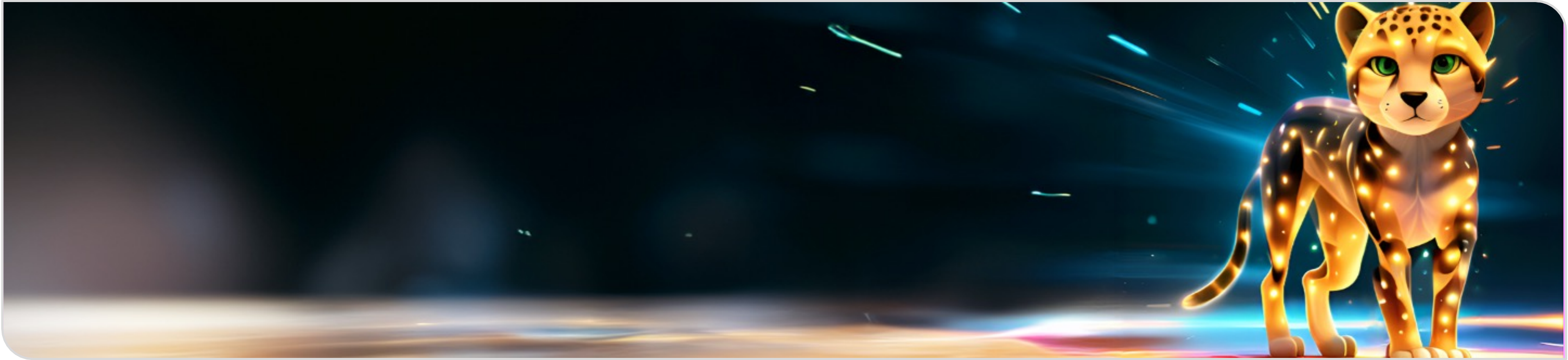
Cheetah: A High-speed Differentiable Beam Dynamics Simulation for Machine Learning Applications

Chenran Xu on behalf of all contributors

2024, 14th International Computational Accelerator Physics Conference



What is *Cheetah*?

A Beam Dynamics Simulation Python Package

- Two main features in support of ML applications
  - **Ultra-fast compute**: Cheetah can run order of magnitude faster than some other codes (at the cost of fidelity)
  - **Differentiability**: Based on **PyTorch**, Cheetah supports automatic differentiation for all its computations
- Cheetah provides full **GPU support** and **integrates seamlessly with ML** models built in PyTorch
- Designed to be **easy to use** and **easy to extend**.
  - We generally aim for high **code quality**!
  - **Black** / **isort** code formatting + **flake8** conformity enforced.
  - Encourage proper procedures in GitHub repository (automatic tests / PR templates, good **documentation** etc.)

```
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)
```

Beam Tracking Implementation

Karlsruhe Institute of Technology

- Cheetah uses **`ParameterBeam`** (Gaussian beam envelope) and **`ParticleBeam`**
- **`ParticleBeam`** contains an array of macro-particles. Each particle is represented using the 6d canonical coordinates

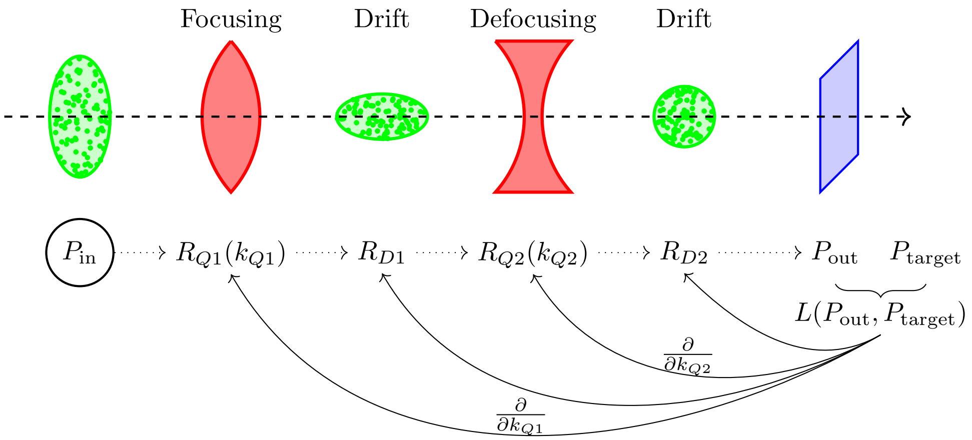
$$\boldsymbol{x} = (x, p_x, y, p_y, z, \delta)^\top$$

- For each lattice element in Cheetah, the linear transfer matrix is implemented

$$P_{\text{out}} = P_{\text{in}} R^\top .$$

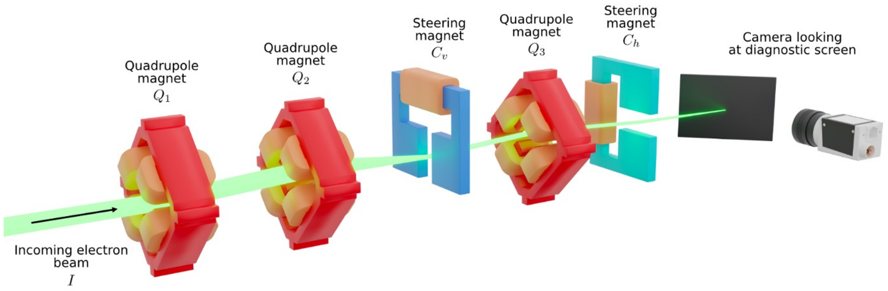
- Tracking method of the elements can be easily overwritten, allowing customizable tracking fidelity

Example: beam tracking through a FODO lattice in linear order

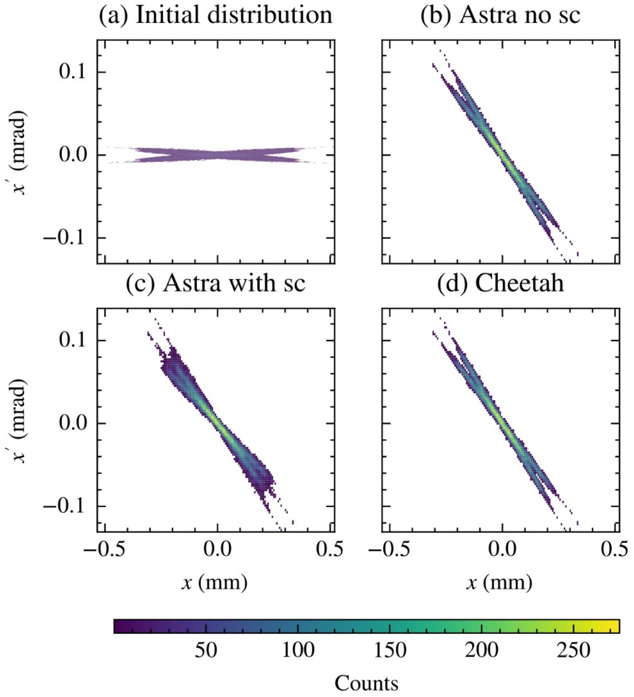


Automatic-differentiation

Cheetah Tracking Examples



Test bunch phase space through the ARES Experimental Area



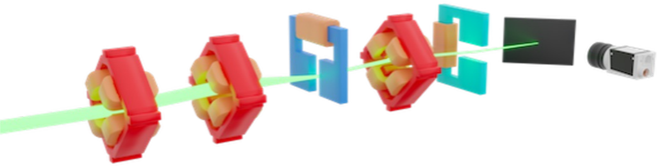
Step compute times through the ARES Experimental Area

Code	Comment	Laptop	HPC node
ASTRA	space charge	264 000.00	3 605 000.00
	no space charge	109 000.00	183 000.00
Parallel ASTRA	space charge	39 000.00	17 300.00
	no space charge	16 900.00	12 600.00
Ocelot	space charge	22 100.00	21 700.00
	no space charge	182.00	119.00
Bmad-X		40.50	74.30
Xsuite	CPU	0.81	2.82
	GPU	-	0.57
Cheetah	`ParticleBeam`	1.60	2.95
	`ParticleBeam` + optimisation	0.79	0.72
	`ParticleBeam` + GPU	-	4.63
	`ParticleBeam` + optimisation + GPU	-	0.09
	`ParameterBeam`	0.76	1.29
	`ParameterBeam` + optimisation	0.02	0.04

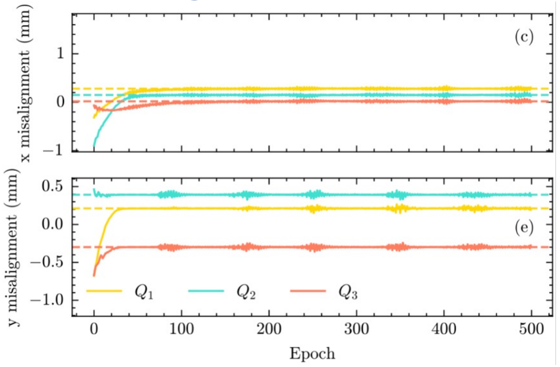
What can you do with it?

Utilizing Cheetah's Differentiability

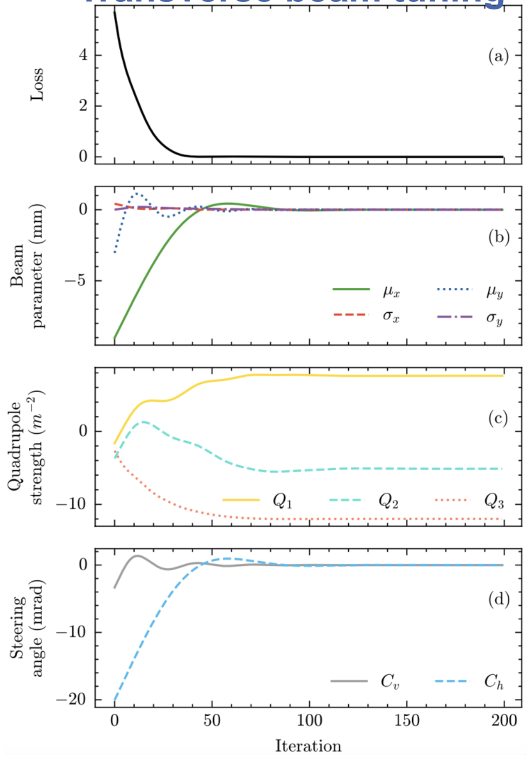
- Applying **gradient-based** algorithms with the gradients computed from the automatic-differentiation
- Efficient optimization with a large number of input parameters
- Example use cases:
  - Simulated accelerator optimization
  - System identification



Misalignment identification



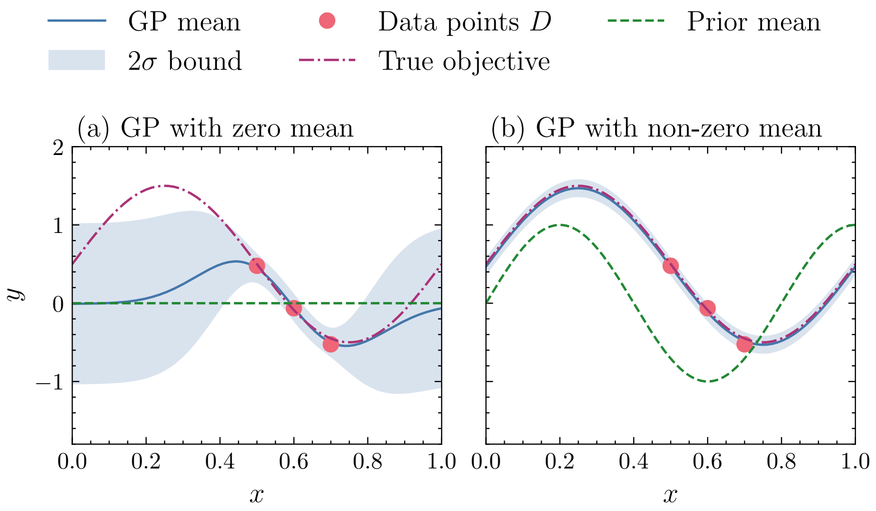
Transverse beam tuning



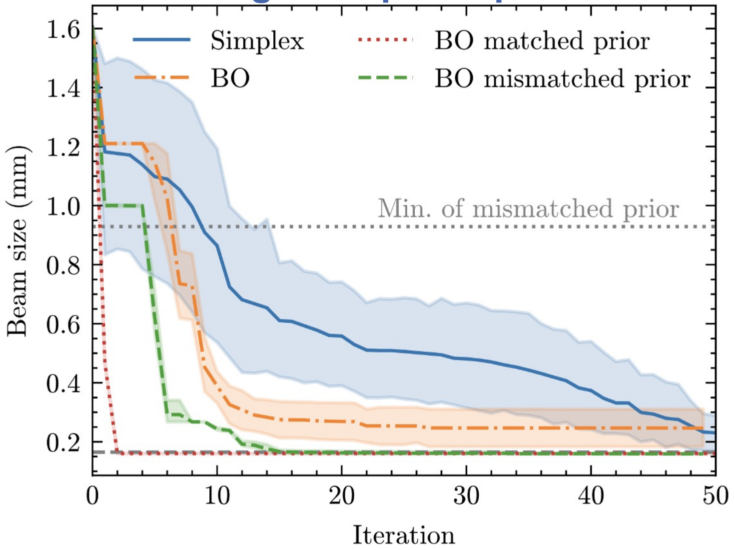
[J. Kaiser, C. Xu et. al. PRAB 2024]

Providing Prior Information for Efficient Optimization

- Cheetah currently only contains limited physical effect (low-fidelity)
- Using Chetah as **physics-informed prior mean** for Bayesian optimization (BO) allows more efficient parameter optimization



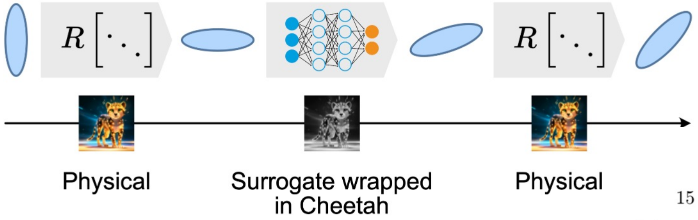
Simulated optimization of beam focusing with quadrupoles



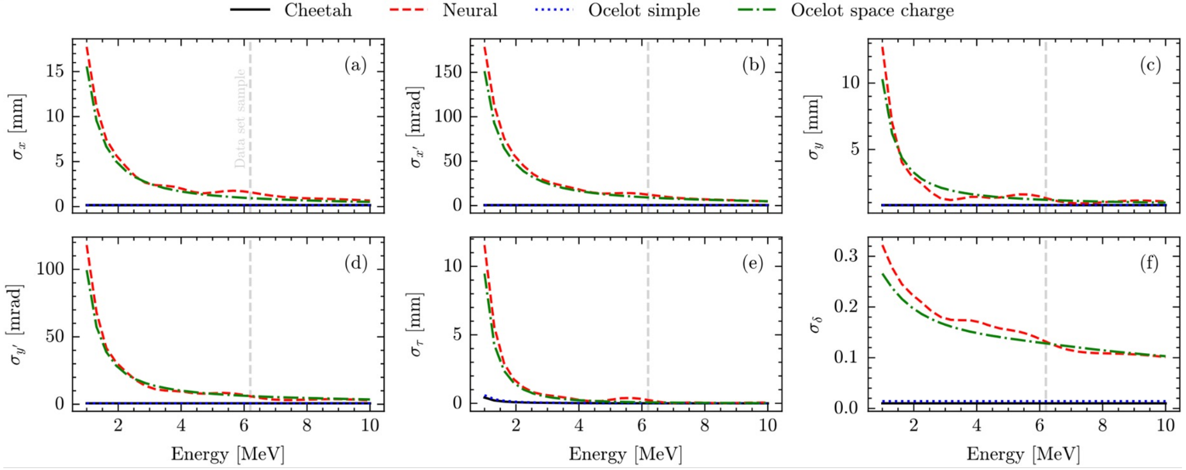
J. Kaiser, C. Xu et. al. PRAB 2024

Seamless Integration with Modular Surrogate Models

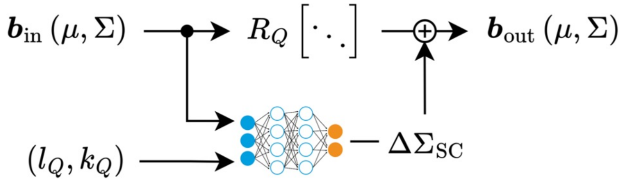
- Neural networks implemented in PyTorch are effectively **native** to Cheetah
- **Differentiability** is preserved and integration is **easy**.



Example: **Tracking with space charge** through quadrupole
3 orders of magnitude faster than Ocelot



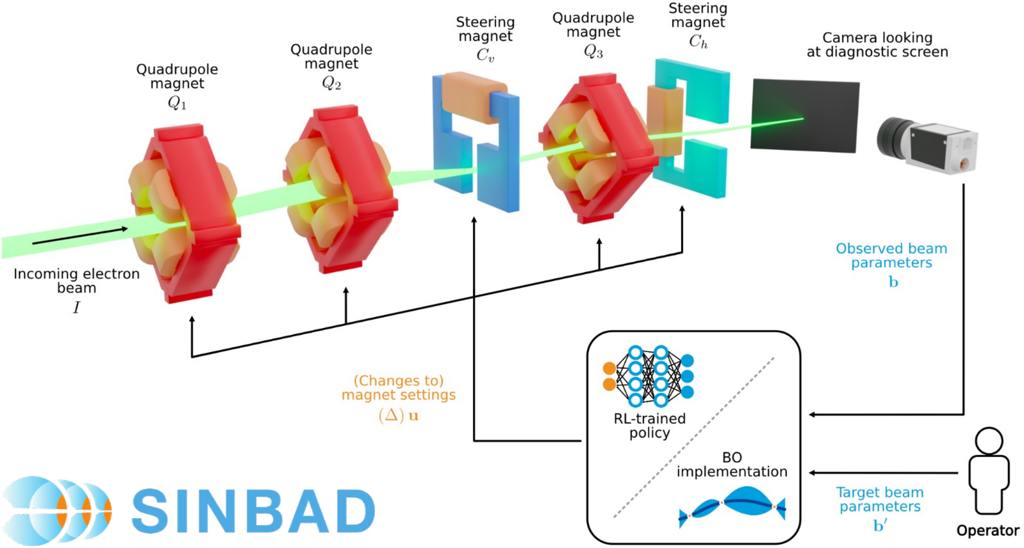
Train a NN to predict the residual in beam parameters due to SC

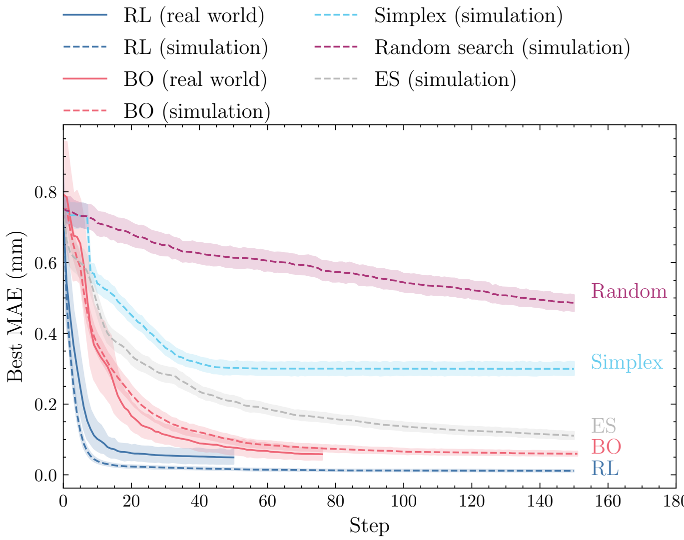


J. Kaiser, C. Xu et. al. PRAB 2024

Fast Training Environment for Reinforcement Learning

- Training a proof-of-principle RL controller at **ARES**
- Deploy a RL-trained optimizer to the **real-world** with **zero-shot** learning thanks to **domain randomization**.





[J. Kaiser et. al. ICML 2022](#)
[J. Kaiser, C, Xu et. al. Sci. Rep 2024](#)

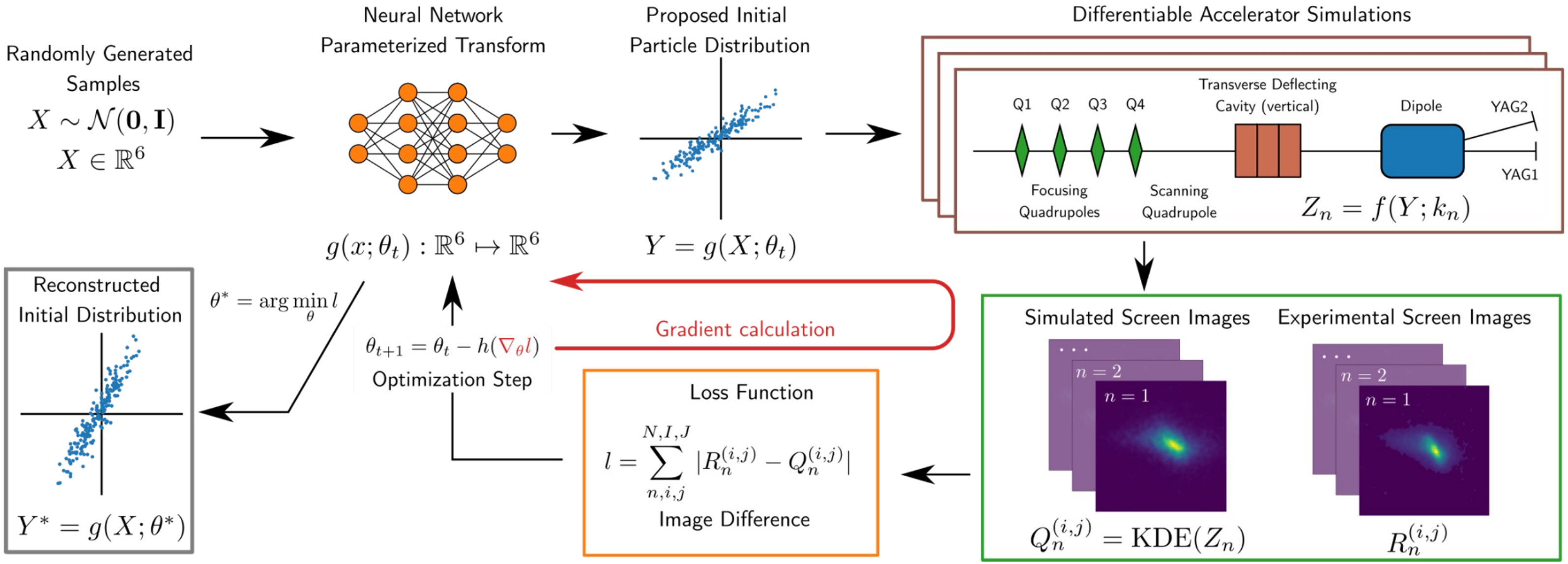
Ongoing Work

Ongoing Projects involving Cheetah

Generative Phase Space Reconstruction (GPSR)

Courtesy of Ryan Roussel and J. P. Gonzalez-Aguilera

- Previously developed at SLAC using **Bmad-X** differentiable simulator
- Ported to Cheetah for faster reconstruction using **vectorised computations** and **GPU acceleration**
- Bmad-X features have been **integrated into Cheetah**

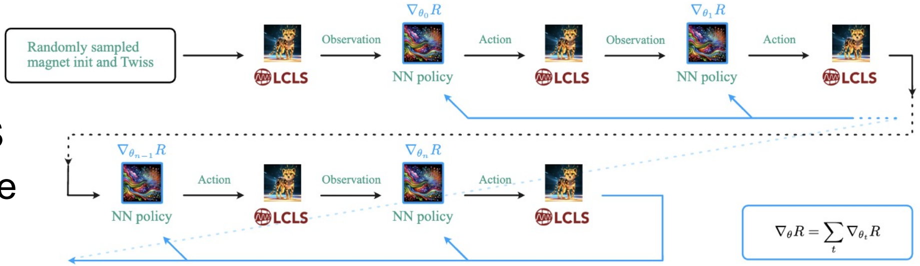


R. Roussel et. al. PRL 2023, R. Roussel et al. PRAB 2024

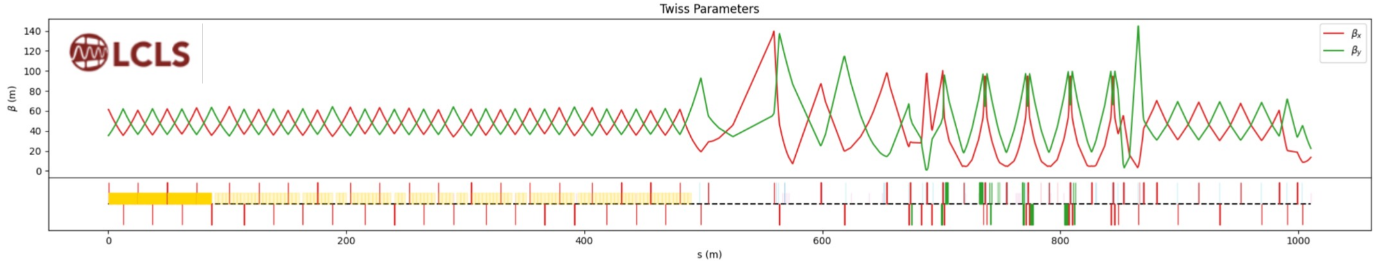
Ongoing Projects involving Cheetah

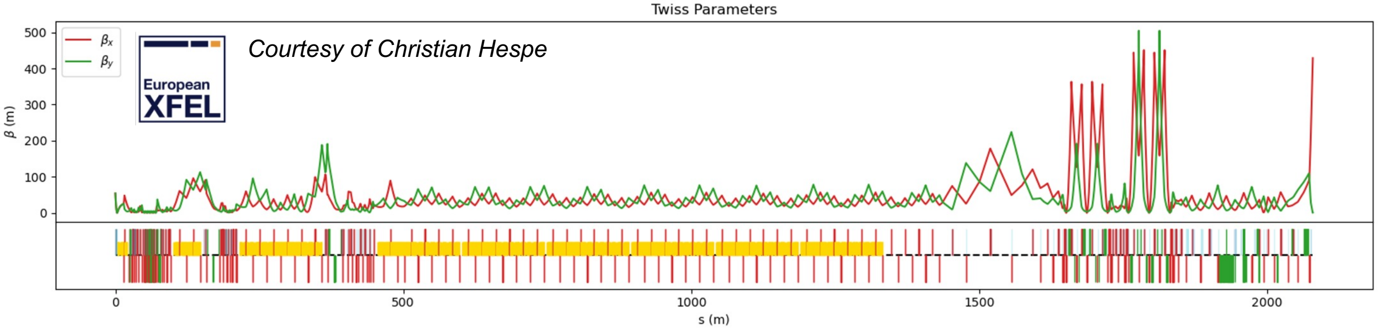
Reinforcement Learning for FELs

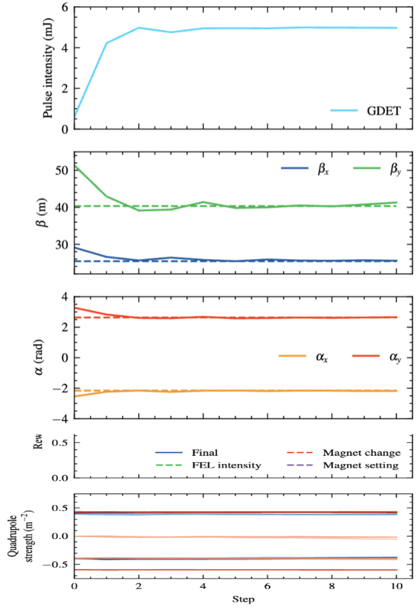
Courtesy of Jan Kaiser



- 45x faster RL training for FEL intensity tuning at LCLS
- TLD dump line feedback at EuXFEL with RL/ MPC-like controllers



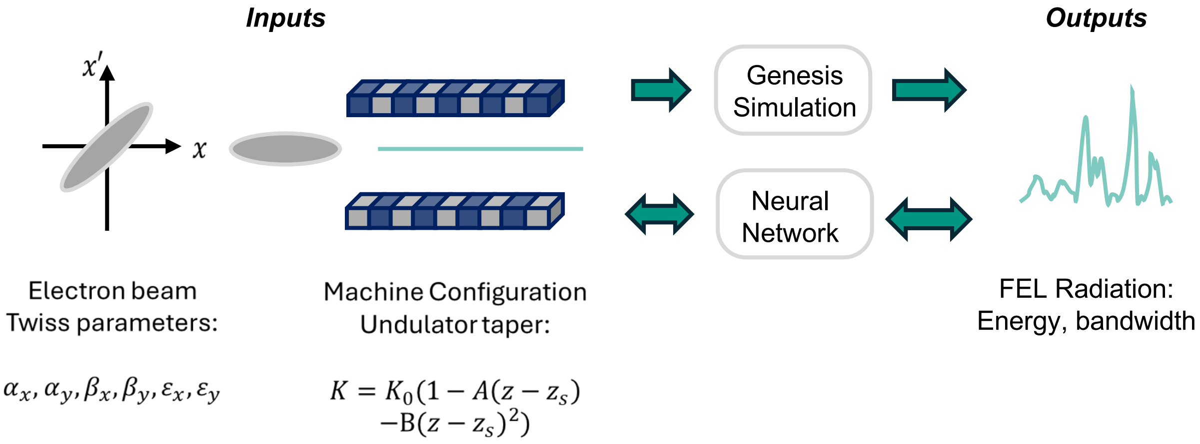




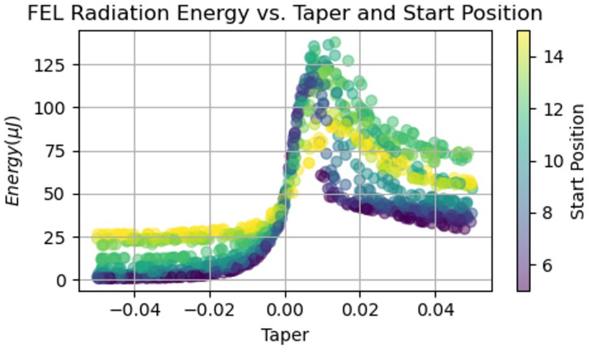
Ongoing Projects involving Cheetah
Fast FEL Modeling

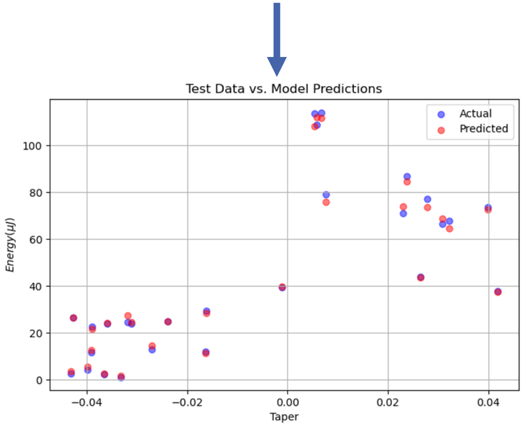
Courtesy of Jenny Morgan

- **Coupled neural network surrogate model for predicting FEL output** trained from GENESIS simulations Much faster FEL simulation
- Prototype predicting FEL output from Twiss and taper



Test case: linear taper

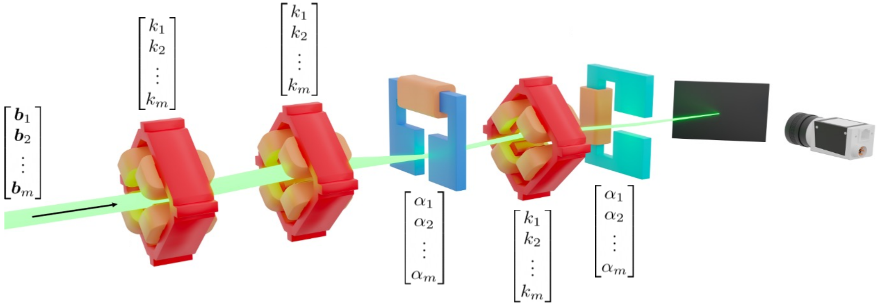




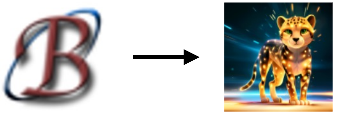
Cheetah Development

Karlsruhe Institute of Technology

- **Vectorized Cheetah** → near-final version available on **`master`** branch, soon **v0.7**
  - **Concurrent simulation** of different actuator settings and beams
  - About **50x speed-up** on CPU, expected to be **even faster on GPU**
  - Automatic PyTorch **broadcasting**



- **Merge features of Bmad-X** → on **`master`**, soon **v0.7**
  - Higher order effects (from Taylor maps) in quadrupoles, dipoles, drift and transverse deflecting cavities

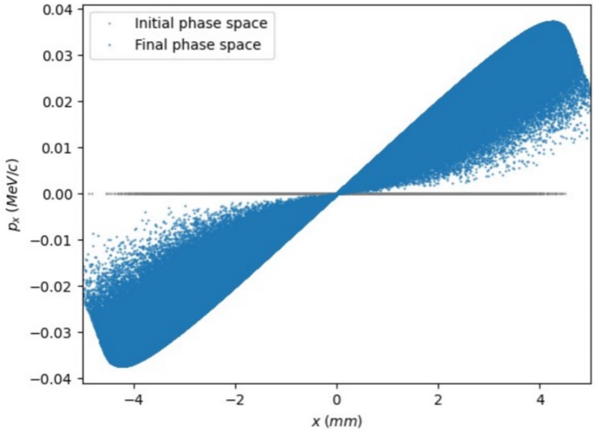


By J. P. Gonzalez-Aguilera

[R. Roussel et. al. LINAC2024](#)

Cheetah Development

- **Space charge** → available on **`master`** branch, soon **v0.7**
  - First backwards differentiable space charge implementation
  - Investigate **memory requirements** and scalability of automatic differentiation



By Remi Lehe, Axel Huebl, and Grégoire Charleux

- **Lynx** → Cheetah with a JAX backend
  - Expected to be faster thanks to **JIT** compilation
  - Support for **forward mode** auto-diff
  - Possibly suited for scientific computing

By Jan Kaiser

[*R. Roussel et. al. LINAC2024*]

People Involved

A successful collaboration!

DESY.

KIT Karlsruher Institut für Technologie

BERKELEY LAB


Jan Kaiser


Christian Hespe


Chenran Xu


Grégoire Charleux


Axel Huebl


Annika Eichler


A. Santamaría García


Remi Lehe

THE UNIVERSITY OF CHICAGO

SLAC


J. P. Gonzalez-Aguilera


Ryan Roussel


Daniel Ratner


Auralee Edelen


Jenny Morgan

Get Started with Cheetah

- Checkout the **GitHub repository** and try the latest version **v0.7 pre-release**

  https://github.com/desy-ml/cheetah

- Or directly install the stable version **v0.6.3**

```
pip install cheetah-accelerator
```

- Or Read the paper:
  - Jan Kaiser, Chenran Xu, Annika Eichler and Andrea Santamaria Garcia. **Bridging the Gap Between Machine Learning and Particle Accelerator Physics with High-Speed, Differentiable Simulations**. In Physical Review Accelerators and Beams, 2024

Scan for repository!