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

4th MODE Workshop on Differentiable Programming for Experiment Design

Jan Kaiser on behalf of all contributors

Valencia, 23 September 2024

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

THE UNIVERSITY OF CHICAGO BERKELEY LAB SLAC KIT Karlsruher Institut für Technologie DESY.

What is *Cheetah*?

Cheetah

Linear Beam Dynamics Simulation Python Package

- **Python package for beam dynamics simulations based on PyTorch for use with machine learning applications.**

- Two main features in support of ML applications:
  - **Ultra-fast compute**: (at the cost of fidelity) Cheetah can run order of magnitude faster than some other codes.
  - **Differentiability**: Based on PyTorch, Cheetah supports automatic differentiation for all its computations.

- Incidentally, 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.)

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

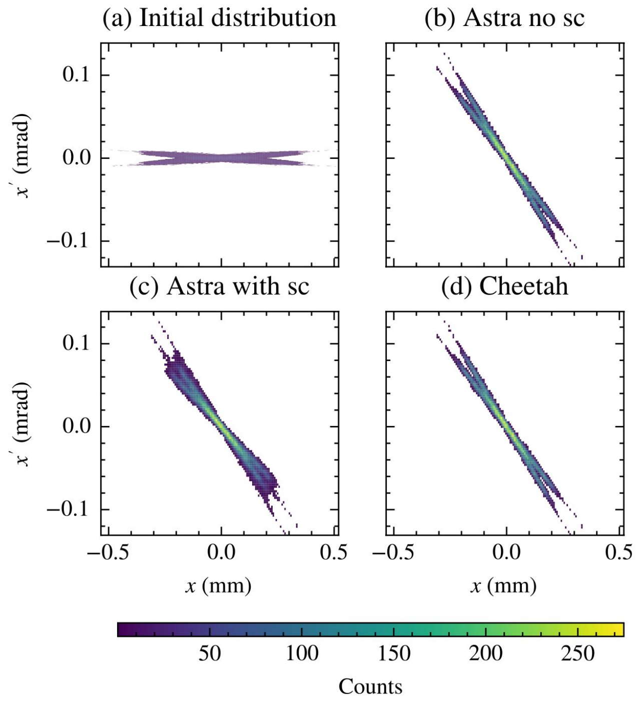
```
pip install cheetah-accelerator
```

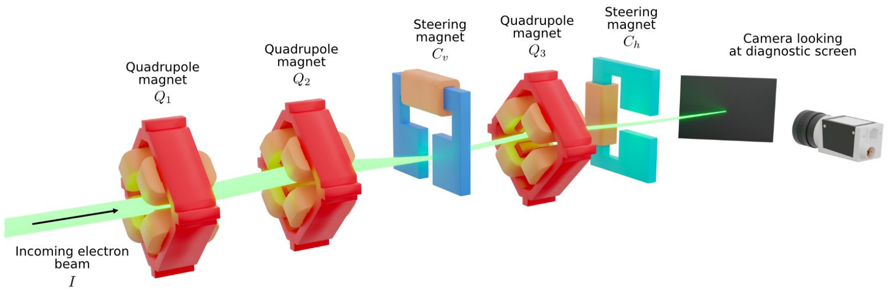
```python
# Load initial beam distribution from ASTRA tracking
beam_in = ParticleBeam.from_astra("beam_in.ini")

# 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)
```

Results

Does Cheetah work?

Phase space through the ARES Experimental Area





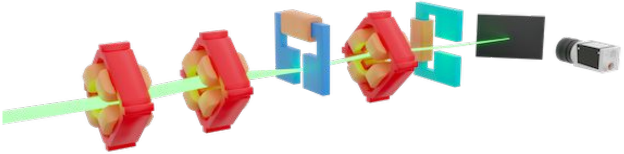
Step compute times through the ARES Experimental Area

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?

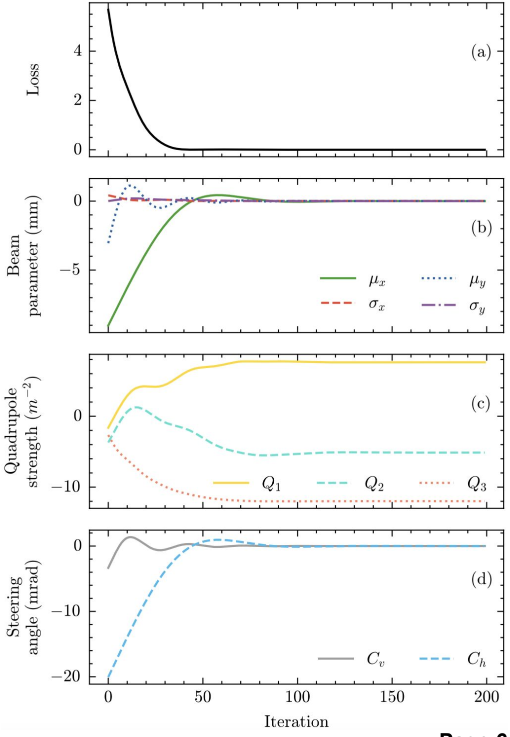
Making Use of Cheetah's Differentiability

Gradient-based tuning, system identification and BO prior

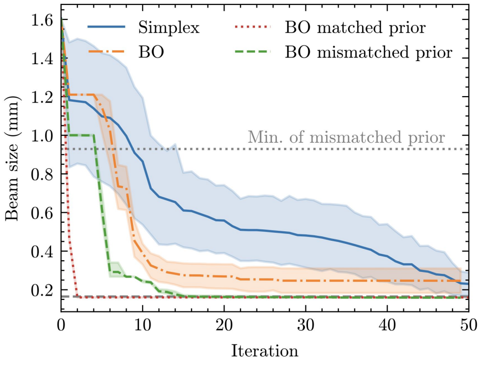


- Taking advantage of the **gradient of the beam dynamics model** computed through **automatic differentiation**, seamless **integration with PyTorch** tools.
- Becomes very useful for **high-dimensional optimisation tasks** (see neural network training).
- Use for example for accelerator tuning, system identification (e.g. misalignments) and even as a physics-informed prior mean for Bayesian optimisation.

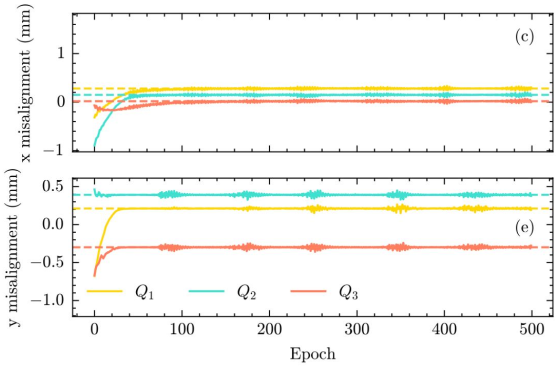
Transverse beam tuning



Bayesian optimisation prior



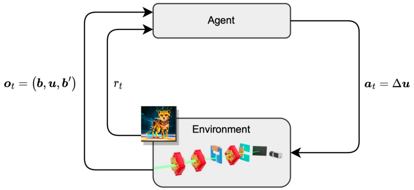
Misalignment identification

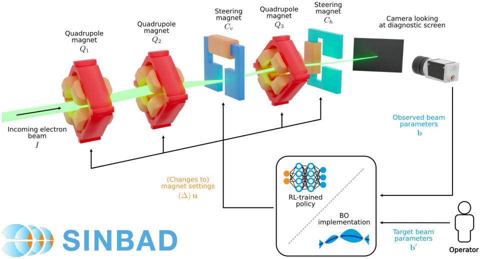


Making Use of Cheetah's Speed

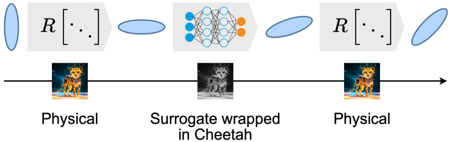
Fast reinforcement learning and collective effects

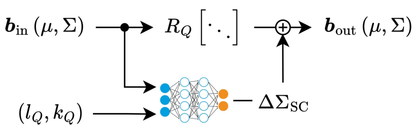
- Proof-of-concept for reinforcement learning (RL) at ARES.
- Would require **3 years of beam time** one the real machine, training would take **11 days with Ocelot**, takes ca. **1 hour with Cheetah**.
- Deploy a RL-trained optimisation algorithm to the **real-world** with **zero-shot learning** thanks to **domain randomisation**.
- The trained policy outperforms other optimisation algorithms and expert human operators.

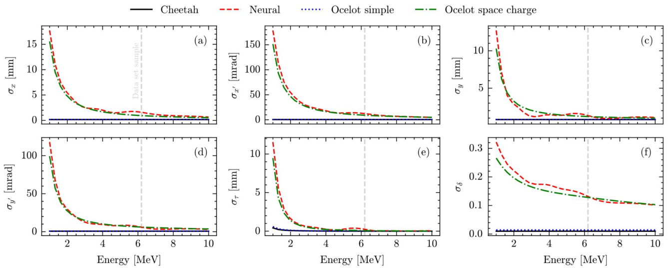




- **AI/ML coupling** with **modular neural network surrogates**.
- Neural networks implemented in PyTorch are effectively **native** to Cheetah. **Differentiability** is preserved. Integration is **easy**.
- Example: **Tracking with space charge** through quadrupole 3 orders of magnitude faster than Ocelot (**370 microseconds**).





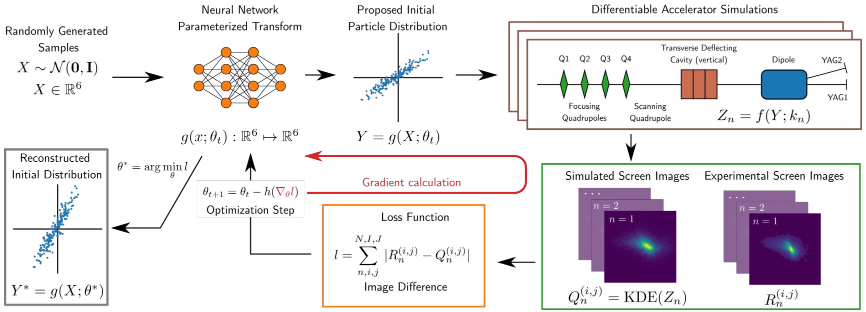


Ongoing Work

Ongoing Projects Involving Cheetah

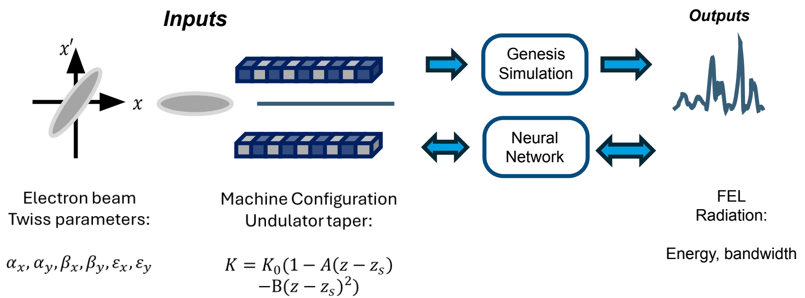
Generative Phase Space Reconstruction and Fast Coupled FEL Modelling

- **Generative machine learning** method for phase space reconstruction using differentiable simulations.
  - **Generative Phase Space Reconstruction (GPSR)**
- **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**.

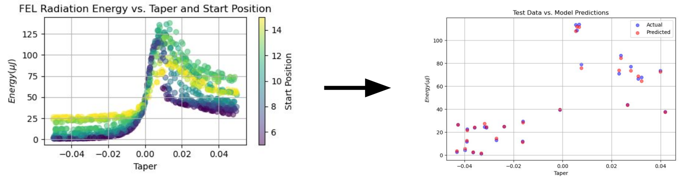


Courtesy of Ryan Roussel and J. P. Gonzalez-Aguilera
R. Roussel et. al. PRL 2023 + R. Roussel et al. PRAB 2024

- **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

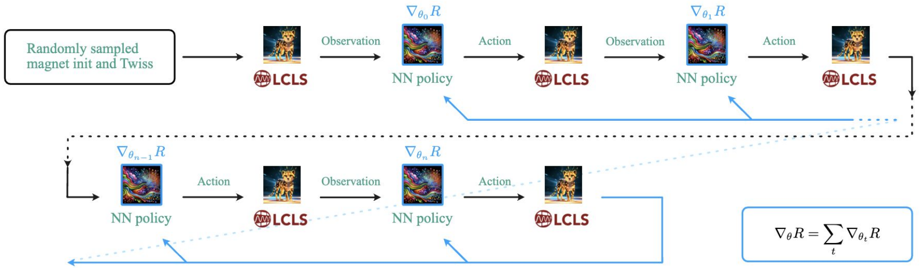


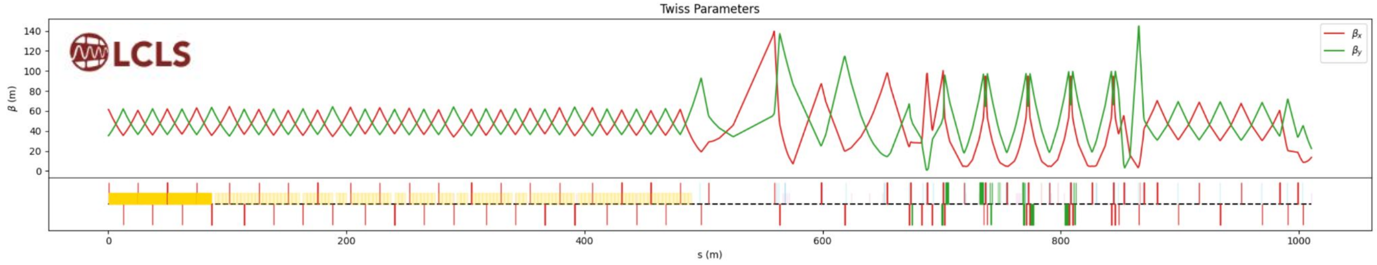
Courtesy of Jenny Morgan

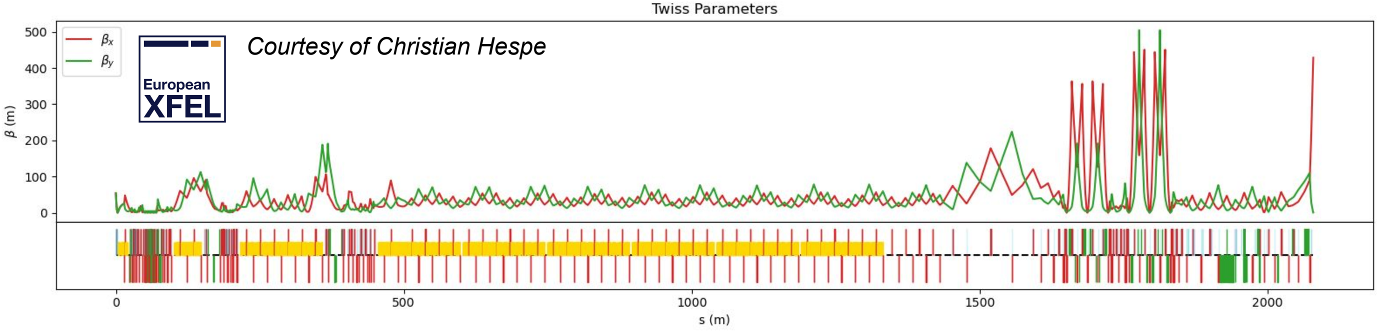
Reinforcement Learning for FELs

Projects at EuXFEL and LCLS

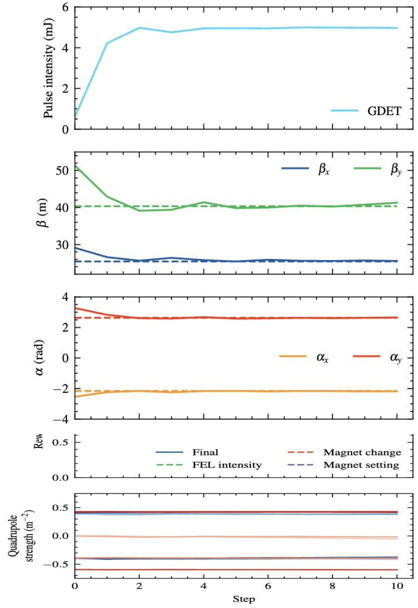
- **45x faster RL training** for **FEL intensity tuning** at LCLS
- TLD **dump line feedback** at EuXFEL with **RL / MPC-like controllers**







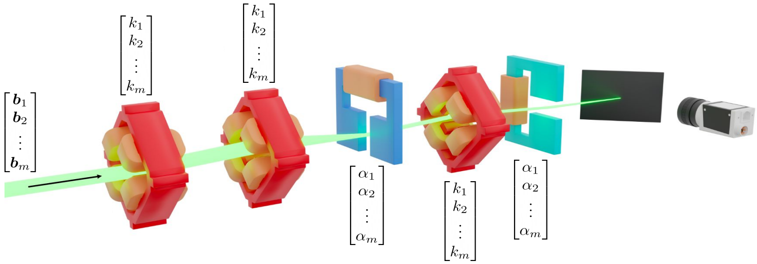
Courtesy of Christian Hespe



Cheetah Development

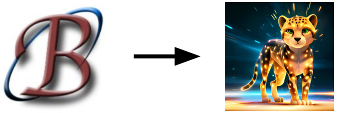
What we are currently working on!

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



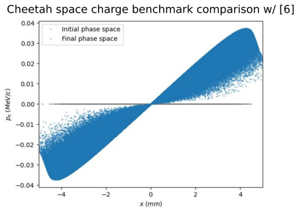
- **Merge features of Bmad-X** → on **`master`**, soon **v0.7**
  - Higher order effects in quadrupoles, dipoles, drift and transverse deflecting cavities

By J. P. Gonzalez-Aguilera



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

By Remi Lehe and Axel Huebl



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

People Involved

Very successful collaboration!

DESY.


Jan Kaiser


Christian Hespe


Annika Eichler

KIT
Karlsruher Institut für Technologie


Chenran Xu


A. Santamaría García

BERKELEY LAB


Grégoire Charleux


Axel Huebl


Remi Lehe

THE UNIVERSITY OF CHICAGO

SLAC


J. P. Gonzalez-Aguilera


Ryan Roussel


Daniel Ratner

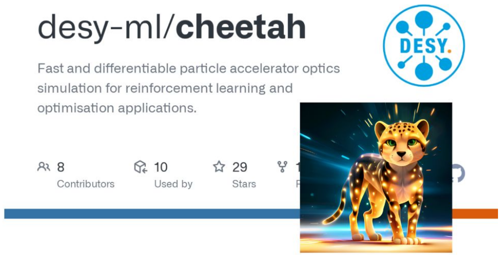

Auralee Edelen


Jenny Morgan

Use Cheetah!

Where to find it and how to get started

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



Scan for repository!

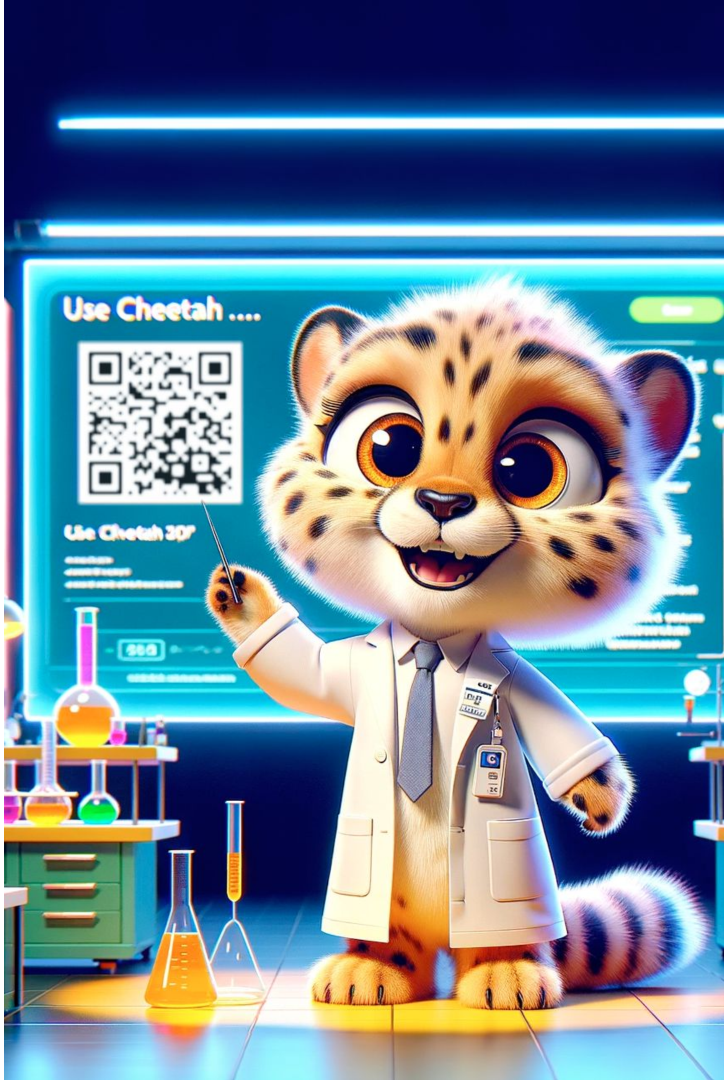
https://github.com/desy-ml/cheetah

- … or directly install the more **stable v0.6.3**

```
pip install cheetah-accelerator
```

- **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.



Contact

DESY. Deutsches
Elektronen-Synchrotron

www.desy.de

Jan Kaiser
Machine Beam Controls (MSK)
jan.kaiser@desy.de