KIT Karlsruhe Institute of Technology

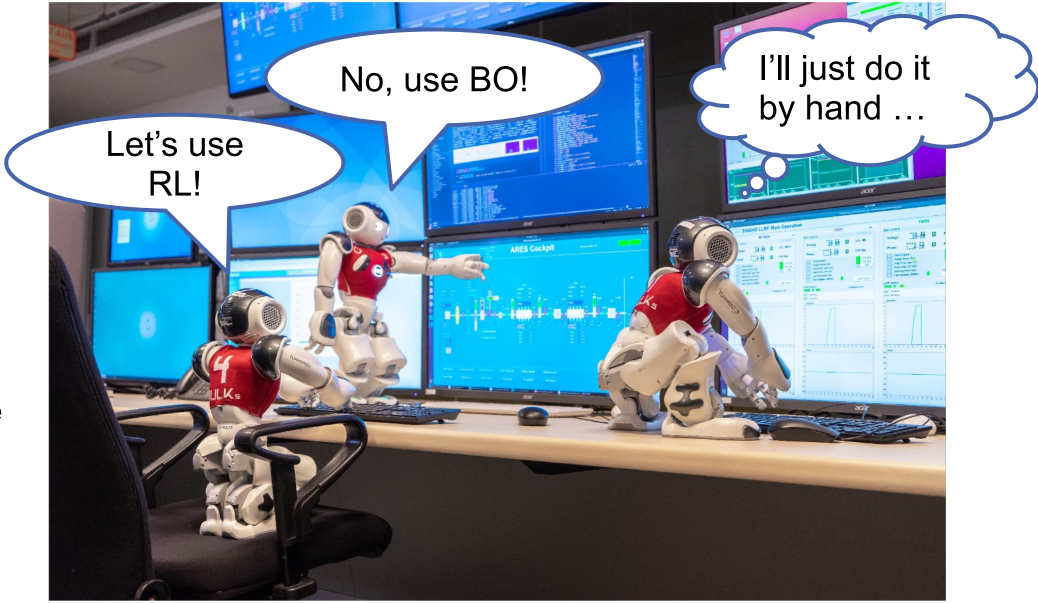
DESY.

HELMHOLTZ

Reinforcement Learning-trained Optimisers and Bayesian Optimisation for Online Continuous Tuning

Jan Kaiser, **Chenran Xu**, Annika Eichler, Andrea Santamaria Garcia, Oliver Stein, Erik Bründermann, Willi Kuropka, Hannes Dinter, Frank Mayet, Thomas Vinatier, Florian Burkart, Holger Schlarb

4th ICFA Beam Dynamics Mini-Workshop on Machine Learning Applications for Particle Accelerators

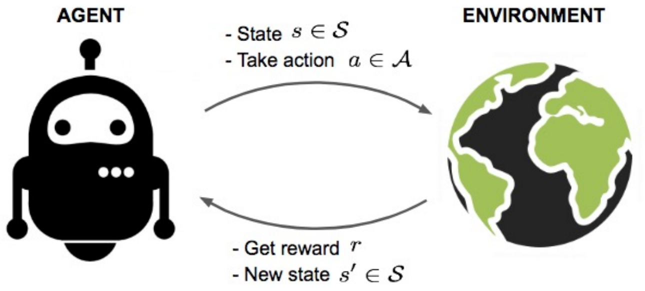


Reinforcement Learning-trained Optimiser (RLO)

RL is a powerful learning paradigm, where an **RL agent** learns through **trial-and-error** interactions with the environment to maximize the **cumulative reward**

The RL-loop:

- The environment is in state $\boldsymbol{s_t}$, agent gets observation $\boldsymbol{o_t}$
- The agent chooses the next action based on its policy $\boldsymbol{\pi}(\boldsymbol{s_t}) = \boldsymbol{a_t}$, which is a neural network in deep RL
- The environment transitions to $\boldsymbol{s_t} \rightarrow \boldsymbol{s_{t+1}}$, receives reward $\boldsymbol{r_t} = \boldsymbol{r}(\boldsymbol{s_t}, \boldsymbol{a_t})$



https://lilianweng.github.io/posts/2018-02-19-rl-overview/

RLO: use RL to (pre-)train the agent, and deploy the agent as an optimiser for the online-tuning problem

Bayesian Optimisation (BO)

BO is a sequential algorithm for **global** optimization of a **black-box** objective function

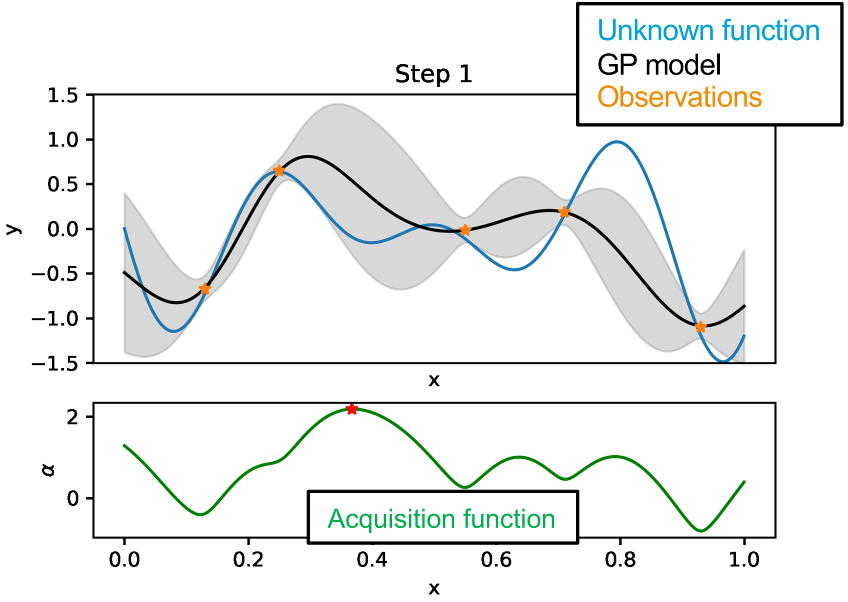
$$\max_{x \in N} f(x)$$

Unknown function

Set of observations

The BO-loop:

- Build a **Gaussian Processes (GP)** model using the observation dataset $N_t = \{(x_i, y_i), i = 1 \dots t\}$
- Construct an **acquisition function** $\boldsymbol{a}(\boldsymbol{x})$ using the GP model, to guide the exploration (experiments)
- Evaluate the new setting proposed by the acquisition function $\boldsymbol{x}_{t+1} = \mathbf{argmax}_x\, \boldsymbol{a}(\boldsymbol{x})$ and observe y_{t+1}



Choosing an optimiser is a trade-off

Apart from the performance metrics (convergence speed, results), one should also consider:

	RLO	BO
Engineering cost: resources needed before deployment	high	low
Inference cost: computational power needed at application time, inference speed	low / ~ ms	high / 0.1 ~ 1 s
Expertise at application time:	low (*nothing to be changed at runtime*)	low - medium (*small hyperparameter adjustments*)

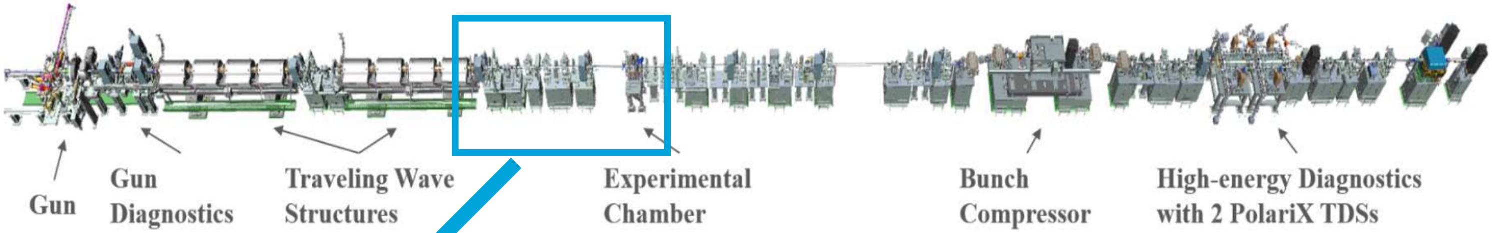
Here we consider the specific case

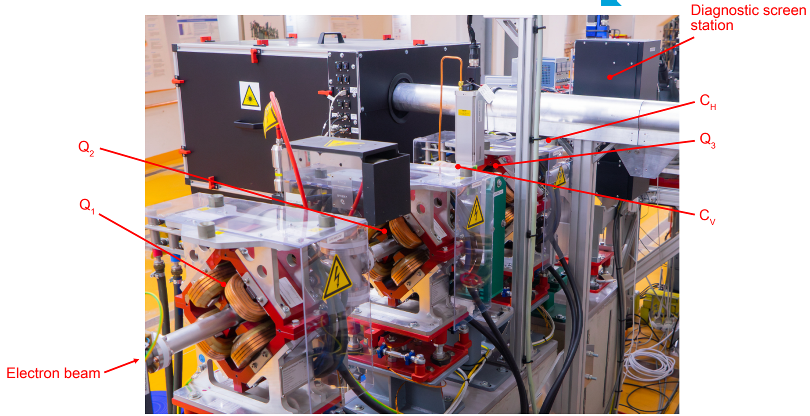
- RLO: model-free algorithm, with pre-training, NN policy
- BO: without informed prior, training from scratch, standard acquisition functions

Assumes stationary conditions

The ARES linear accelerator

Small research accelerator at DESY's SINBAD facility



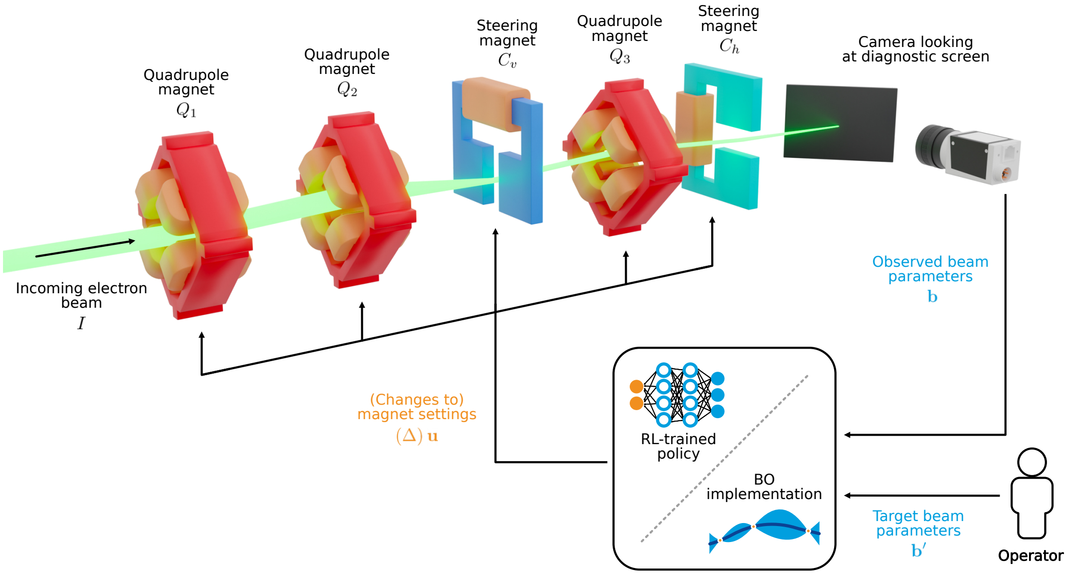


Properties	Target Values
Charge	0.01 - 200 pC
Momentum	50 -155 Mev/c
Momentum Spread	1,00E-04
Transverse emittance	< 0.8 π.mm.mrad
Duration	Sub-fs to ≈ 10 fs

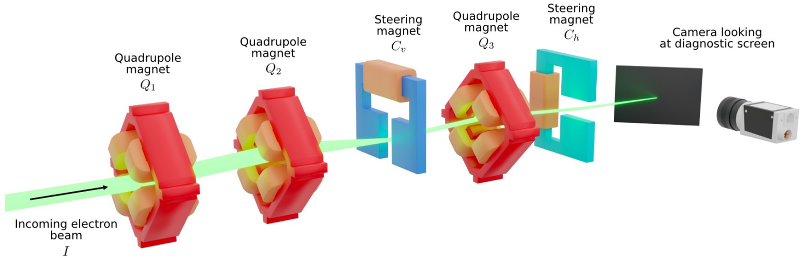
Courtesy: Annika Eichler

ARES Experimental Area (EA) beam tuning task

- **Task**: focus and position the electron beam
- **Actuators**: 3 quadrupole magnets + 2 corrector magnets
- **Observation**: beam image on the diagnostic screen



Formulating ARES-EA as an RL task



Observations

- Magnet Settings $u = [k_{\mathrm{Q1}}, k_{\mathrm{Q2}}, k_{\mathrm{Q3}}, \theta_{\mathrm{v}}, \theta_{\mathrm{h}}]$
- Current Beam $b^{(\mathrm{Current})} = (\mu_x, \sigma_x, \mu_y, \sigma_y)^{(\mathrm{C})}$
- Target Beam $b^{(\mathrm{Target})} = (\mu_x, \sigma_x, \mu_y, \sigma_y)^{(\mathrm{T})}$

Action

Changes to the current magnet setting
$a = \Delta u$ (max step size 10%)

Objective

MAE (mean absolute error)

$$O(u_t) = \frac{1}{4} \left| b_t^{(\mathrm{Current})} - b_t^{(\mathrm{Target})} \right|_1$$

Reward

- Differential mode $r(s_t) \propto \ln(O(u_t)) - \ln(O(u_{t-1}))$ earlier
- Feedback mode $r(s_t) \propto -O(u_t)$ current

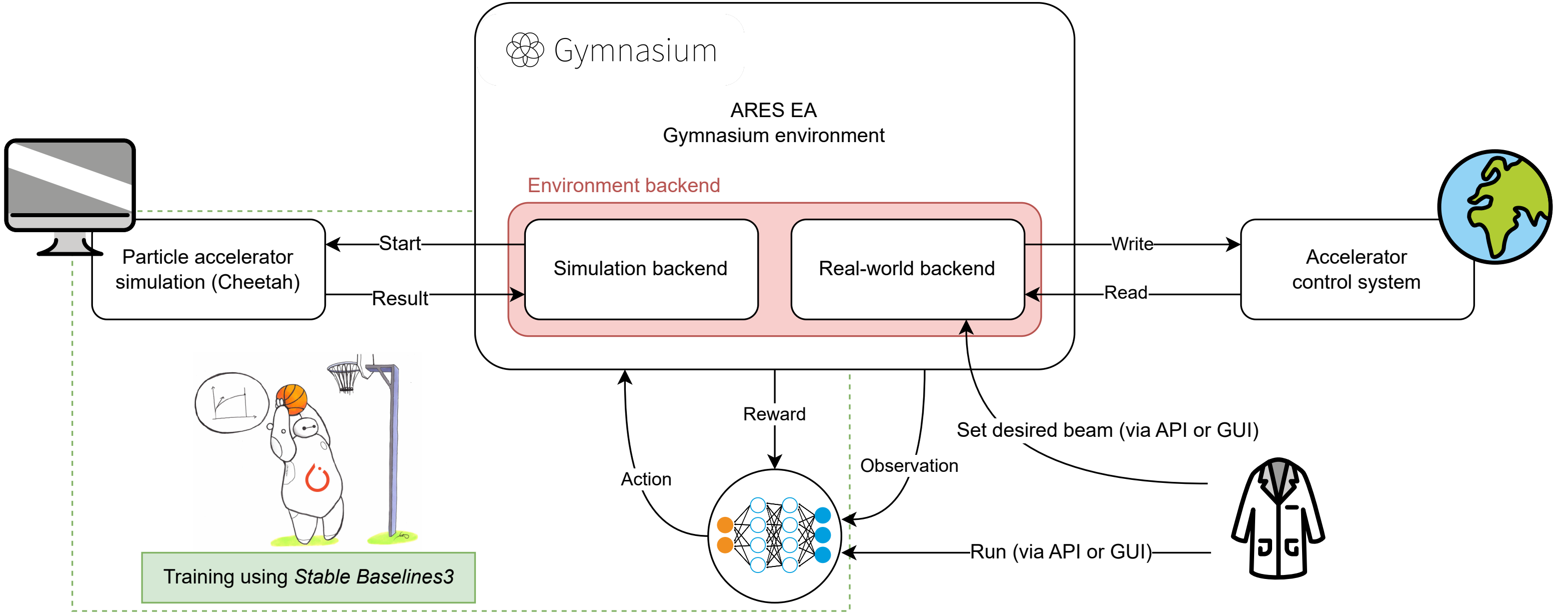
+ transformation (clipping, ...)
+ additional terms (on-screen, magnet changes, ...)

Partially observable Markov decision process (POMDP)

Note: see tutorial for more optional components in the reward definition

- State s = Observation + Hidden variables (incoming beam, magnet and screen misalignments)

Reinforcement learning implementation framework



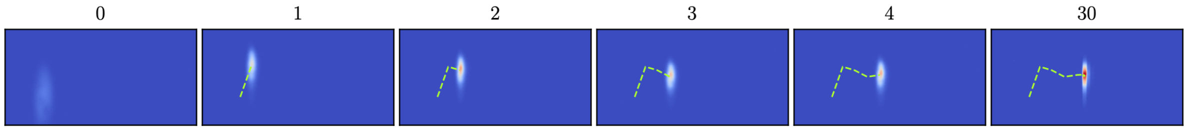
Check out Jan Kaiser's talk on Friday on Cheetah

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

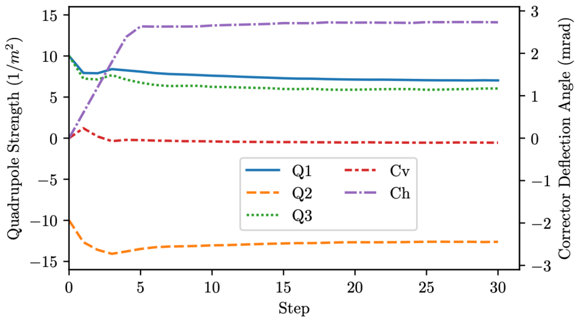
Lahan Select Gyeongju, South Korea 09:40 - 10:00

RL-trained optimiser successfully solves the task

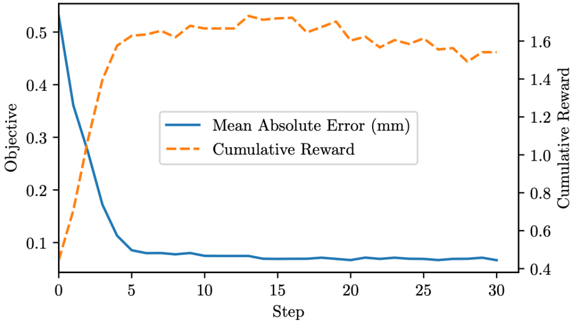
RLO trained with **domain randomisation** in simulation can be deployed to the **real-world** ARES accelerator with **zero-shot learning**



(a) Cropped diagnostic screen image at different steps with high beam intensity in red, low intensity in blue and medium intensity in white.

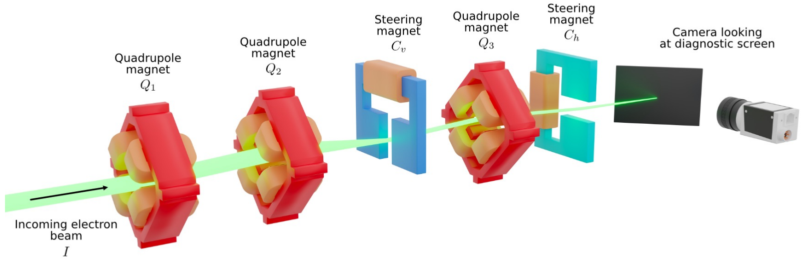


(b) Actuator values over the optimisation run.



(c) Objective and cumulative reward over the optimisation run.

ARES-EA as an optimisation task



Observations

- Magnet Settings $u = [k_{\mathrm{Q1}}, k_{\mathrm{Q2}}, k_{\mathrm{Q3}}, \theta_{\mathrm{v}}, \theta_{\mathrm{h}}]$
- Current Beam $b^{(\mathrm{Current})} = (\mu_x, \sigma_x, \mu_y, \sigma_y)^{(\mathrm{C})}$
- Target Beam $b^{(\mathrm{Target})} = (\mu_x, \sigma_x, \mu_y, \sigma_y)^{(\mathrm{T})}$

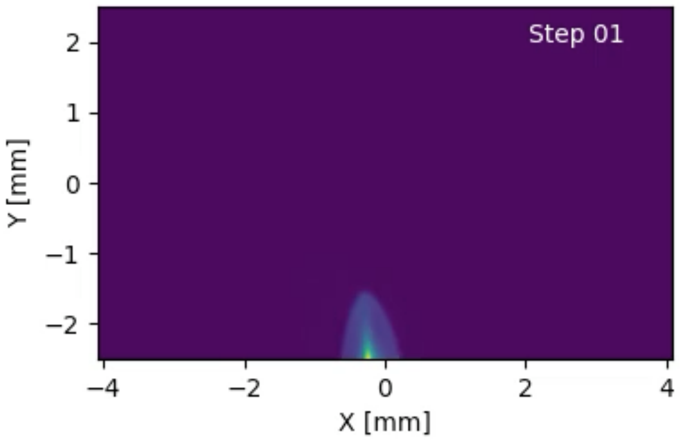
Action (GP input)

Direct magnet settings
$a = u$ (max step size 10% as in RL)

Objective (GP output)

Log-MAE (mean absolute error)
$O(u_t) = -1 * \log\left(\frac{1}{4}\left|b_t^{(\mathrm{Current})} - b_t^{(\mathrm{Target})}\right|_1\right)$

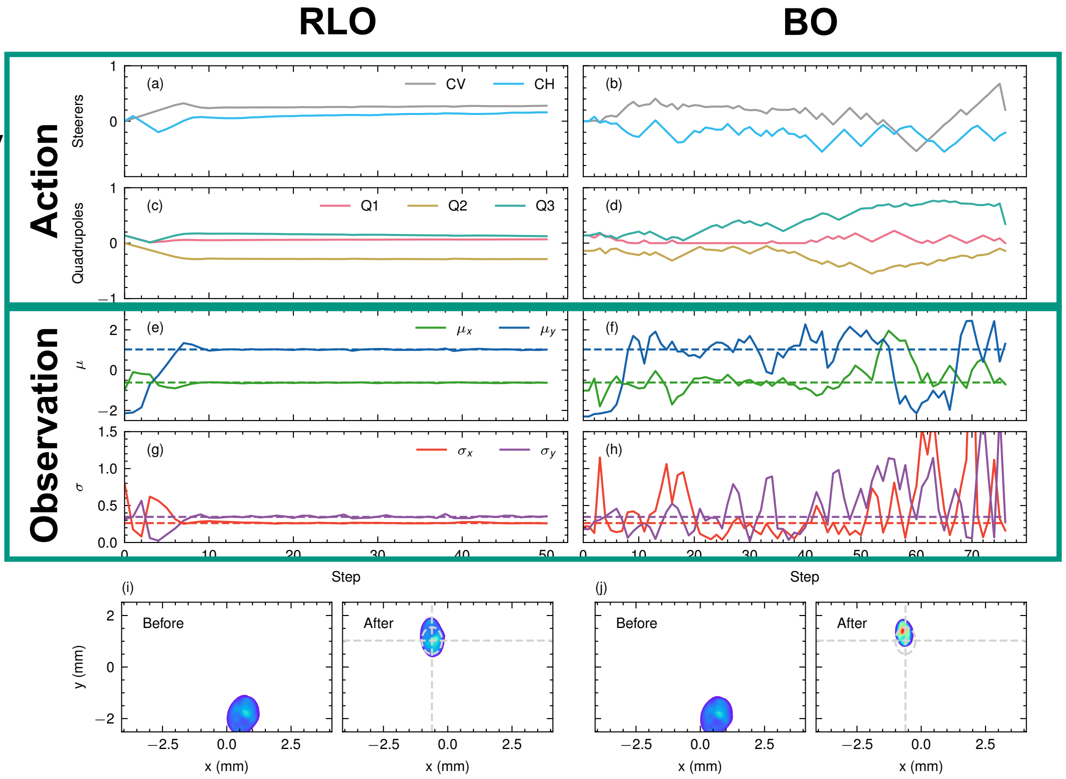
Applying BO to center and focus the beam



RLO and BO applied at ARES

- RLO smoothly converges to the target beam parameters, because it implicitly contains the model information

- BO explores the model on-the-fly and demonstrated more noisy behaviour during the tuning steps



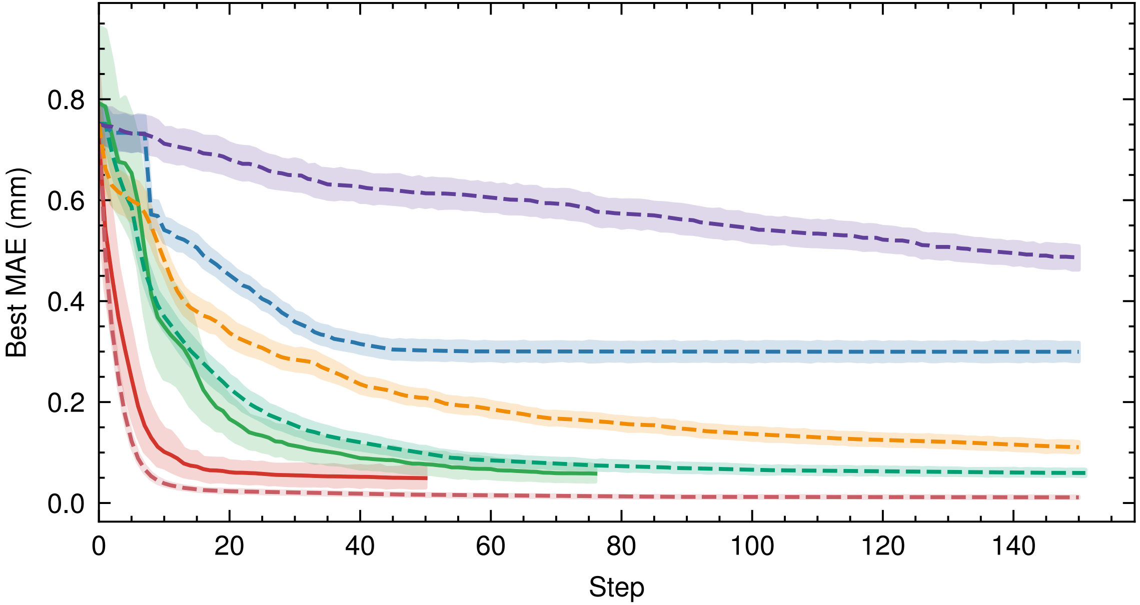
Benchmarking different optimisers' performance

Simulation *(dashed)* & real world *(solid)*

- **RLO**: best final beam parameters and fastest convergence overall
- BO: no performance degrade in real-world

Simulation only

- Extremum seeking: decay of amplitude needed for convergence
- Nelder-Mead simplex: often get stuck in the local optima
- Random-search



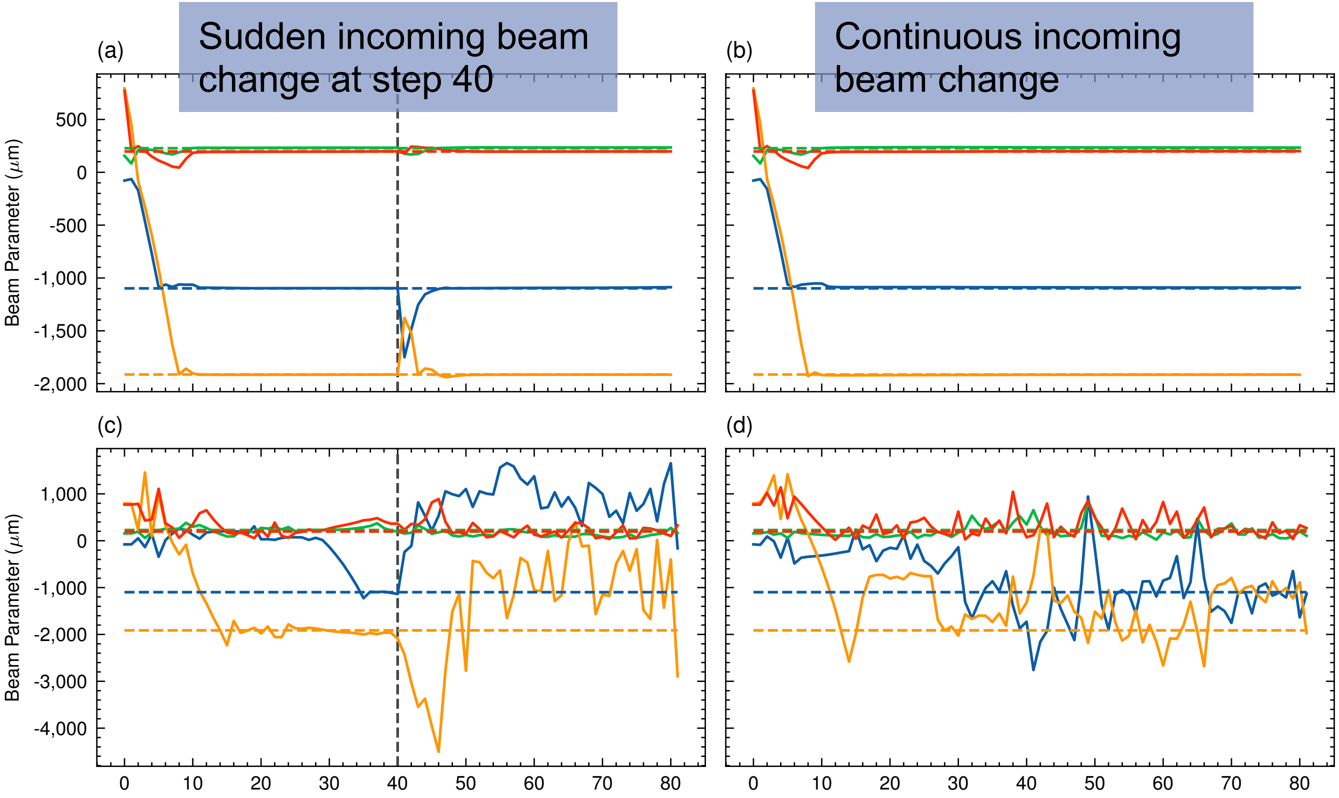
The envelopes show the 95 % CL over 300 simulation and 22 real-world trials

Behaviour in a non-stationary system

- **RLO** quickly adapts to changes of the env. hidden state, as a robust feedback controller

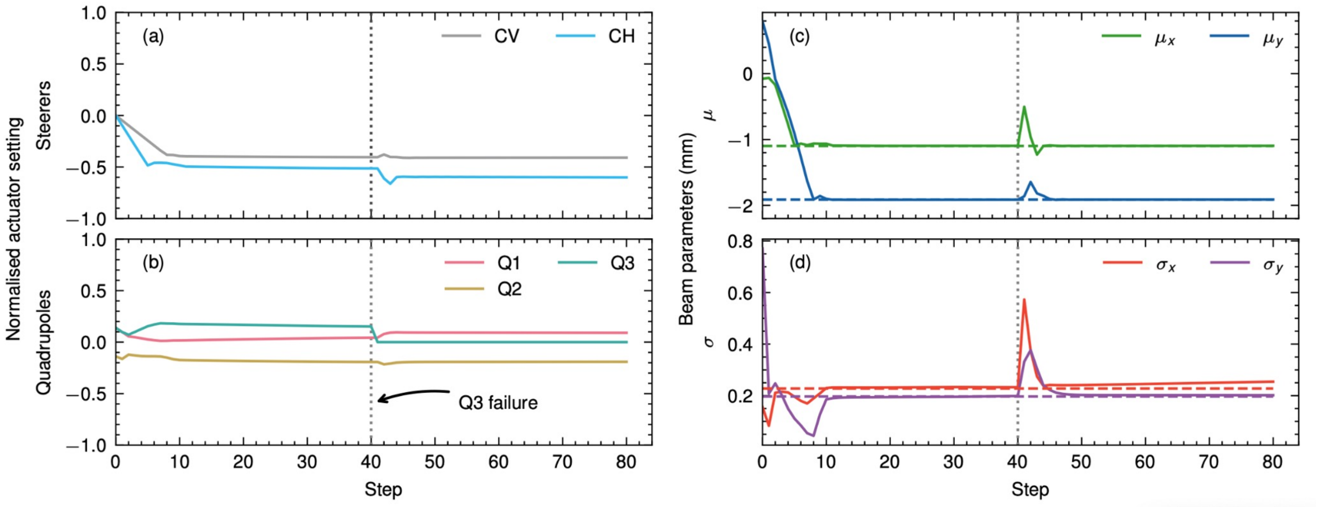
- **BO** struggles to deal with changes in the system (violating the GP assumption)

Note: BO can better adapt with slow drifts when **including time information** into the kernel



Running RLO as a feedback

RLO can also adapt to **changes of the underlying system** to some extent.
Example: 1 of the 3 quadrupoles fails (strength goes to zero)



Conclusion

Karlsruhe Institute of Technology

HELMHOLTZ

- Pre-trained RLO can be directly deployed at real machine with zero-shot transfer. It is faster and achieves best results among the compared methods.

- BO can be applied as a turn-key solution and works well on the common tuning task.

- Both methods have potential for better performance
  - RL: reducing the upfront-engineering effort & sample requirements by using model-based RL or meta RL.
  - BO: faster convergence and better tuning results using methods tailored to the task, e.g. NN-/physics-prior GP, adaptive kernel,...

Read the paper

https://arxiv.org/abs/2306.03739

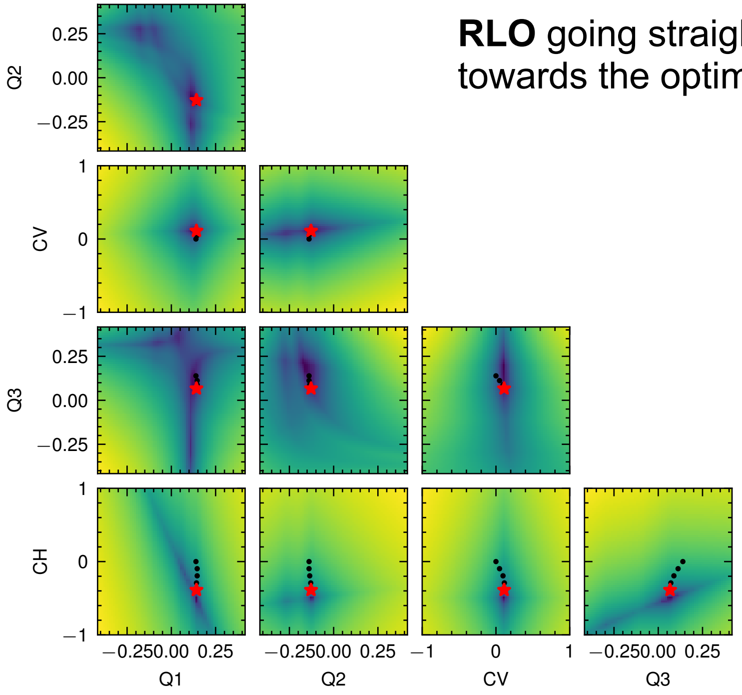
Next up: hands-on tutorial

- We will be looking at the **RL implementation details**, and the **design choices** we faced for the ARES-EA task

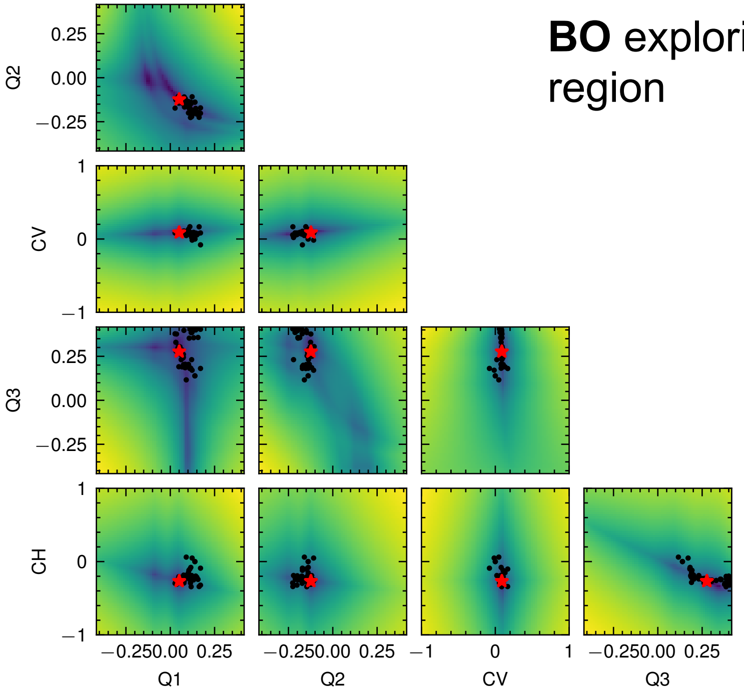
- GitHub link: https://github.com/RL4AA/rl-tutorial-ares-basic

Backup slides

Objective space exploration comparison

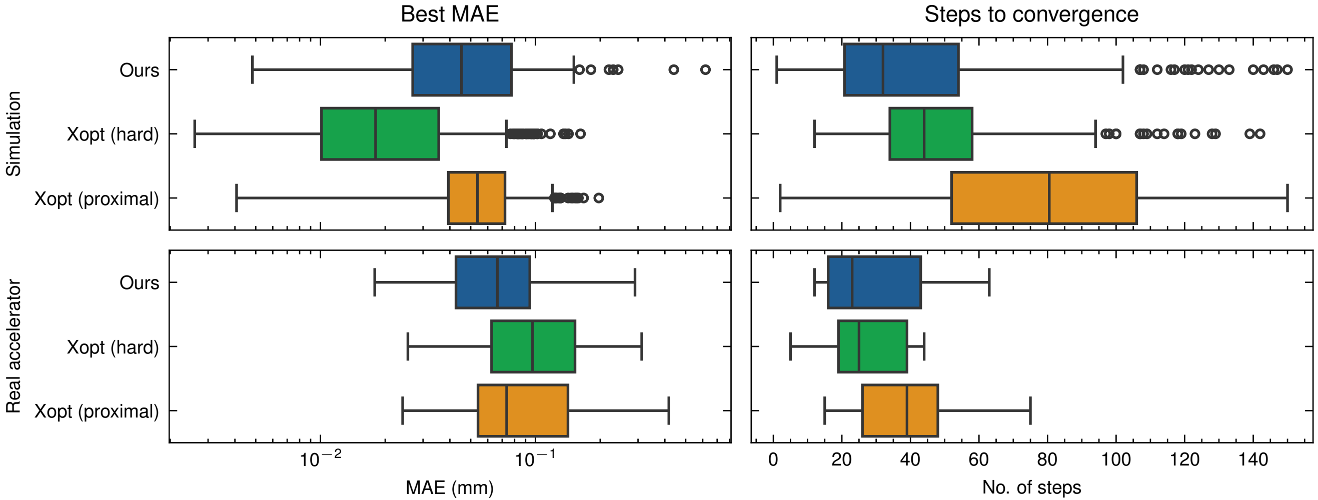


RLO going straight towards the optimum



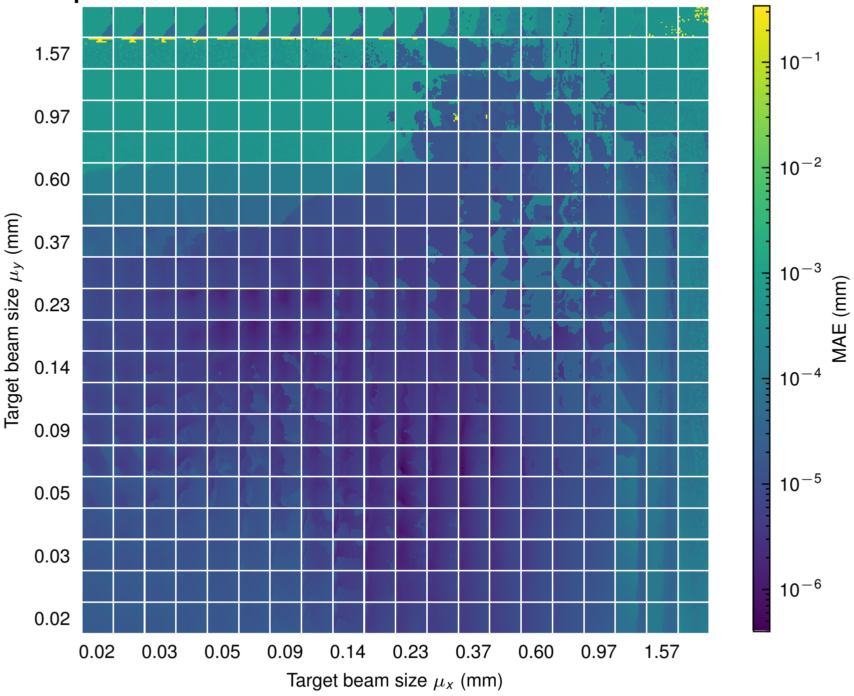
BO exploring the region

Our custom BO implementation demonstrates similar performance as the Xopt implementations

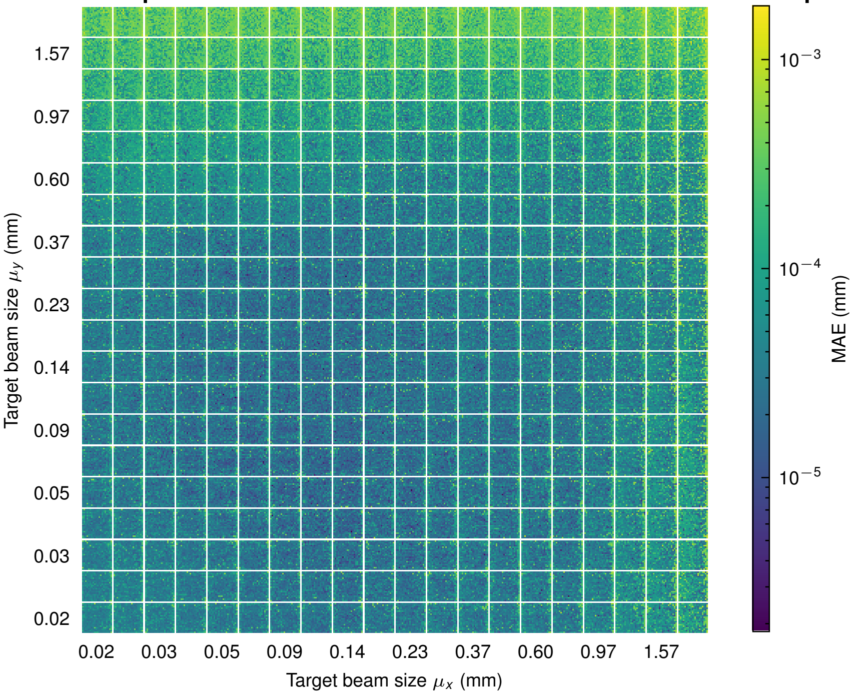


Performance for different target beam parameters

RLO prefers small-medium beam sizes



BO performance rather uniform across the space



Small boxes: beam positions on the screen