UNIVERSITY OF LIVERPOOL

Department of Physics

RL IN PARTICLE ACCELERATORS

Dr. Andrea Santamaria Garcia
Assistant professor, AI for particle accelerators

Dedicated AI/ML Accelerator Test Facility
17/07/2026 virtual presentation

Why is accelerator operation difficult?

Imperfect models

Real accelerator never matches model

Physics models provide useful operating points, not exact solutions

RF errors, magnet misalignments, field errors, and diagnostic uncertainties introduce discrepancies

Machine conditions evolve continuously

High-dimensional optimisation

Explore large parameter spaces

Hundreds to thousands of controllable parameters and strongly coupled subsystems

Multiple competing objectives (stability, intensity, efficiency, availability)

Large parameter spaces are difficult to explore systematically

Dependence on expert knowledge

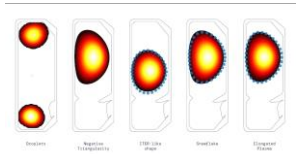
Performance depends on operator expertise

Many tuning procedures remain empirical

Operators iteratively adjust machine parameters based on experience

Knowledge is often tacit and difficult to formalise

Why RL for accelerators?

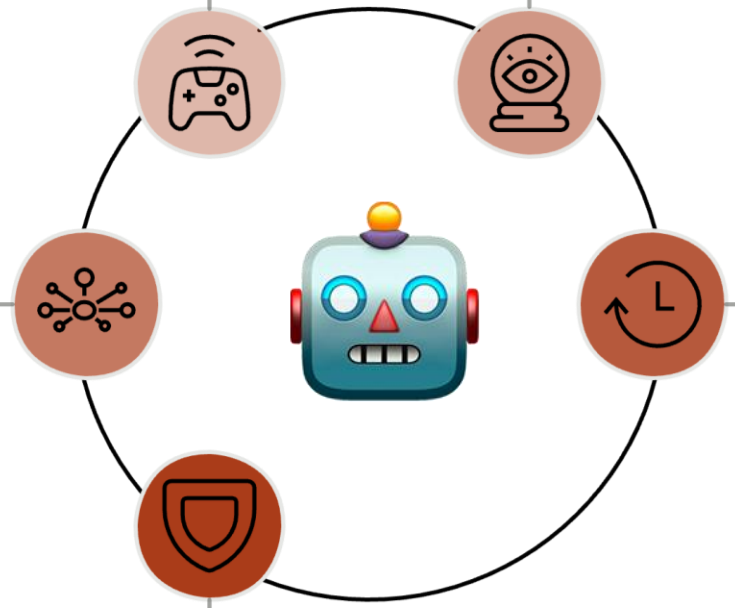


Control Policies

Learns strategies for managing accelerator operations

Future Evolution

Considers long-term system changes in decision-making



Simultaneous Control

Can coordinate many control variables at once for complex systems

Millisecond Inference

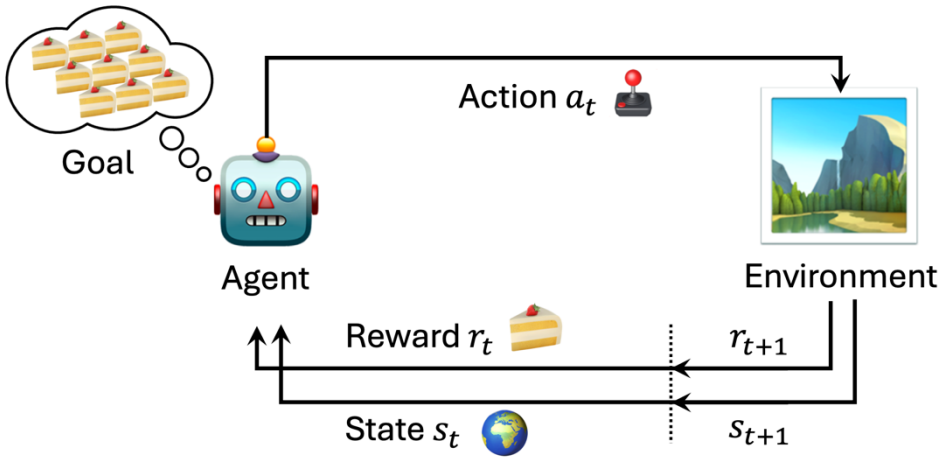
Achieves rapid control responses after training at constant time inference

Safe Simulation

Can be trained safely in simulation and safety can be embedded in the controller

RL in a nutshell

Actions are evaluated not only by their immediate outcome, but also by their consequences for future states and future rewards



An agent (algorithm) learns through trial-and-error by interacting with a dynamic environment

Agent

1. Executes **action** a_t ***free-will***
2. Receives **observation** s_t ***perception***
3. Receives scalar **reward** r_t ***motivation***

Reward

Scalar feedback signal r_t that indicates how well the agent is doing at step t

Cumulative reward (return) $G_t(\tau) = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \quad \gamma \in [0, 1)$

Goal

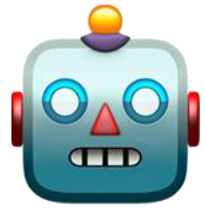
Maximisation of cumulative reward G_t through selected actions

Simple concept from which intelligent behaviour emerges

"Reward is enough" by Silver et al. (2021)
"Scalar reward is not enough": a response to Silver et al. (2021)

The RL policy π: a DNN controller

The map from observation to action that is ultimately deployed on the accelerator



Parametrized control policy π

Observations

Beam images

BPM readings

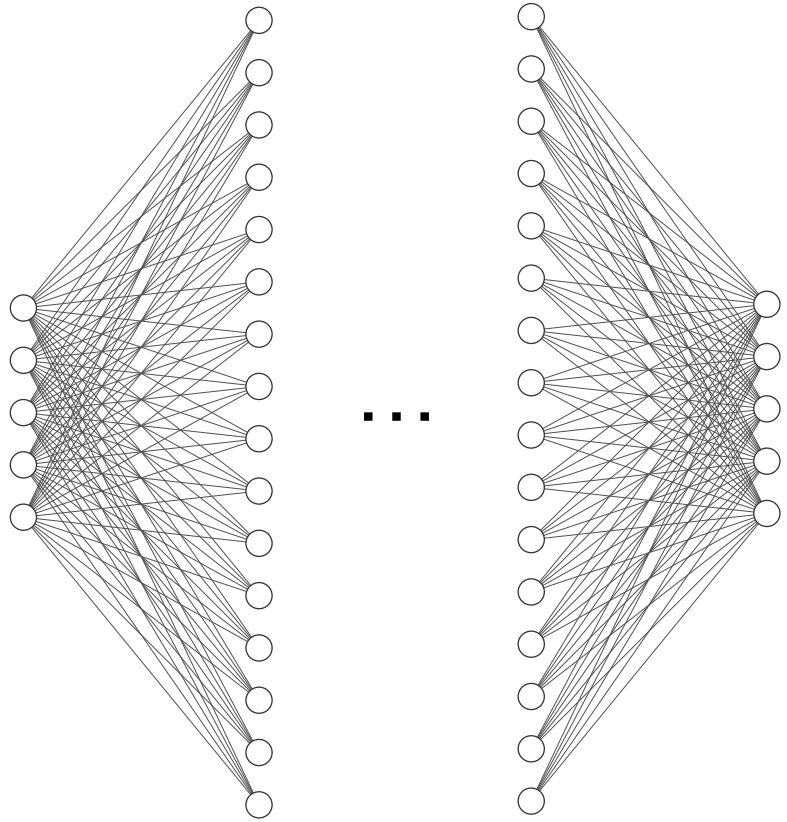
Charge

Beam losses

RF measurements

Magnet readbacks

Previous control actions



Actions

Steering corrections

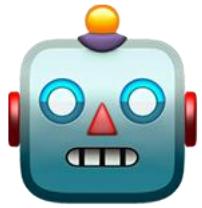
Quadrupole strengths

RF settings

Laser parameters

The RL policy π: a DNN controller

The map from observation to action that is ultimately deployed on the accelerator

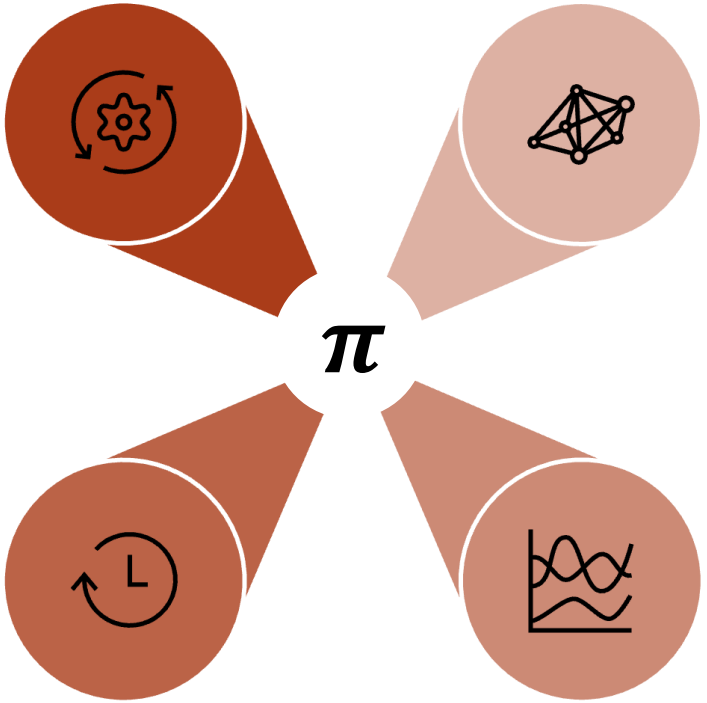


Closed-loop operation

Every new observation produces a new control action

High-dimensional input

Scales to high-dimensional input



Fast execution

Training is computationally expensive; deployment is only a forward pass through the network

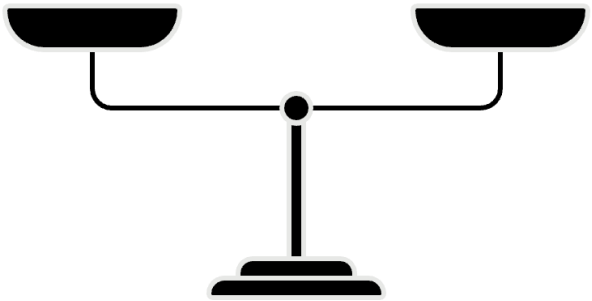
Nonlinear controller

Learns relationships that are difficult to derive analytically

The RL policy π $\neq$ a surrogate model

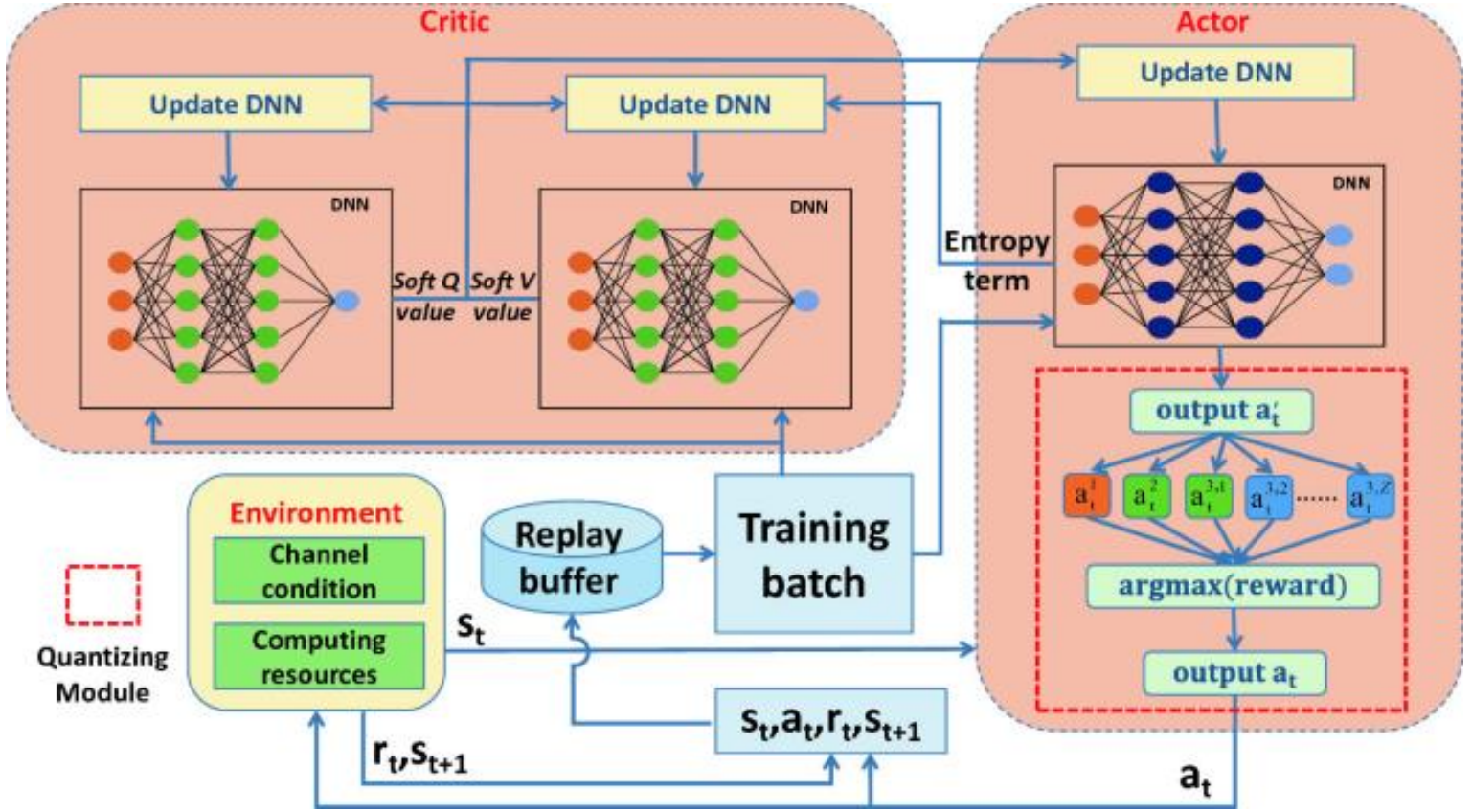
As we understand it

Surrogate model	RL policy
Predicts machine behaviour	Produces control actions
Learns the physics	Learns a control strategy
Output is a prediction	Output is an actuator command



The RL policy π $\neq$ a surrogate model

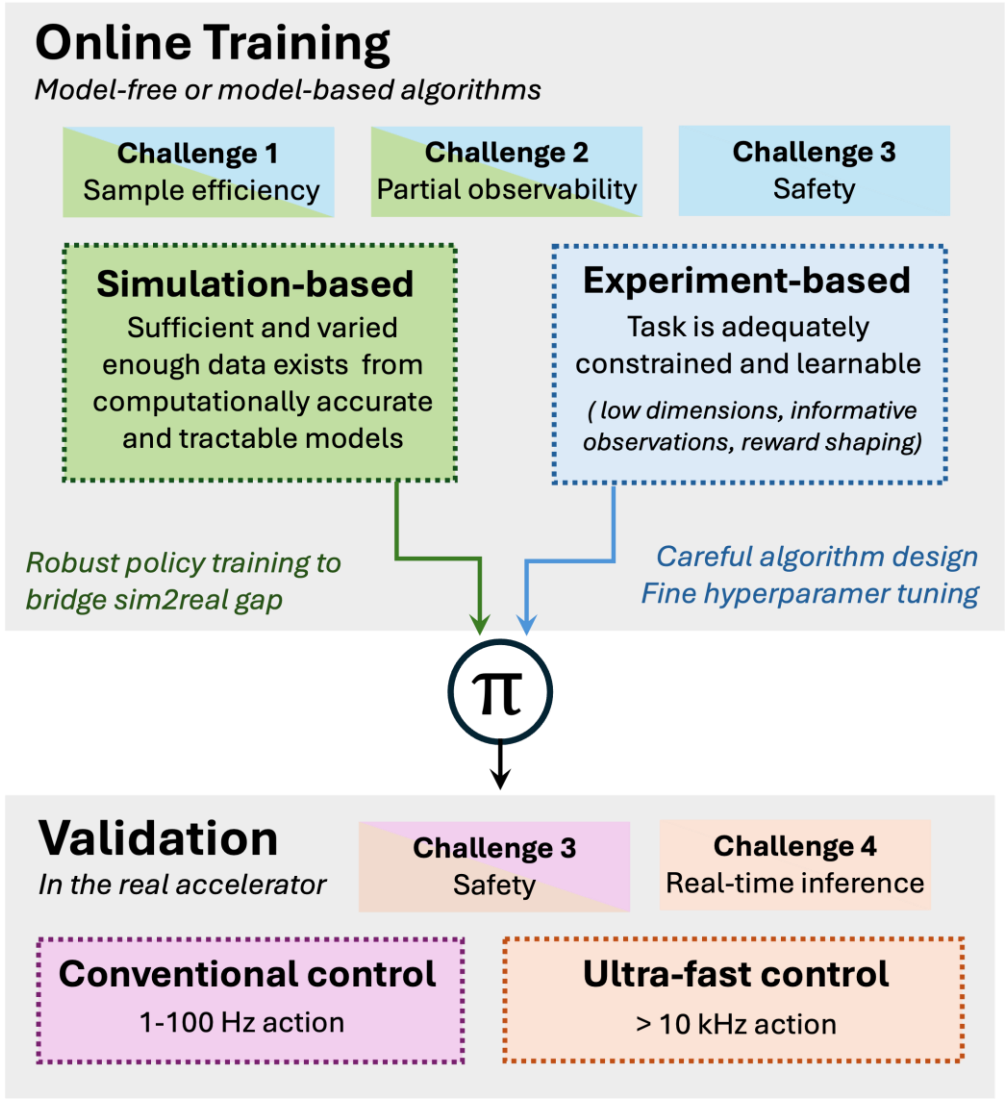
As we understand it



Online: data is actively collected during training (sim & experiment)

Offline: learns from a fixed dataset (supervised learning)

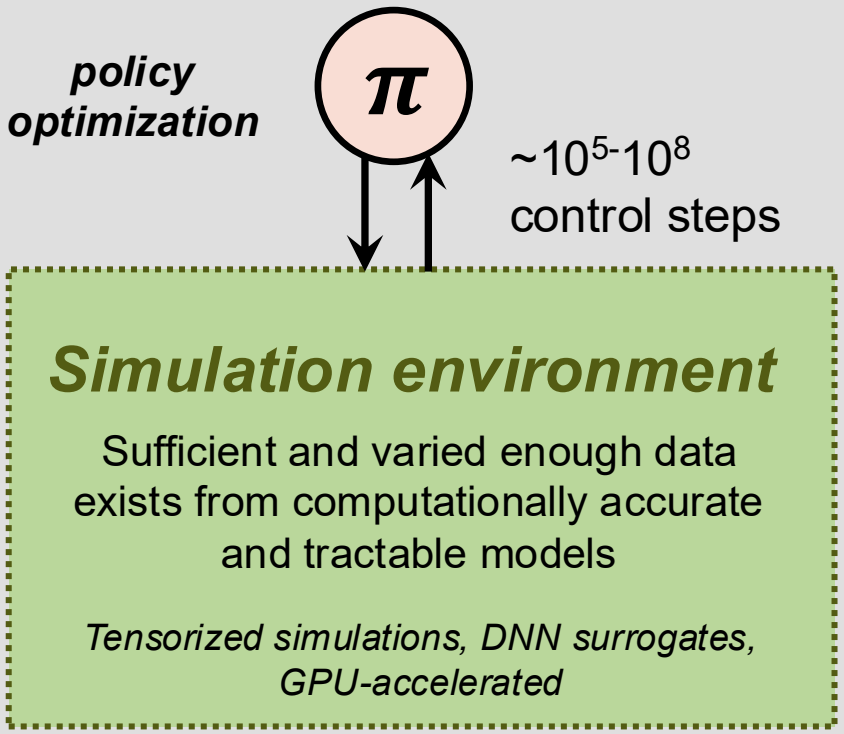
Main challenges of RL deployment



These same challenges apply generally to ML deployment in operational facilities

Santamaria Garcia, A. *et al*. Reinforcement learning in particle accelerators. in (2025).

RL trained in simulation, then deployed



Objective: learn a control policy that maximizes long-term performance through repeated interaction with the simulation

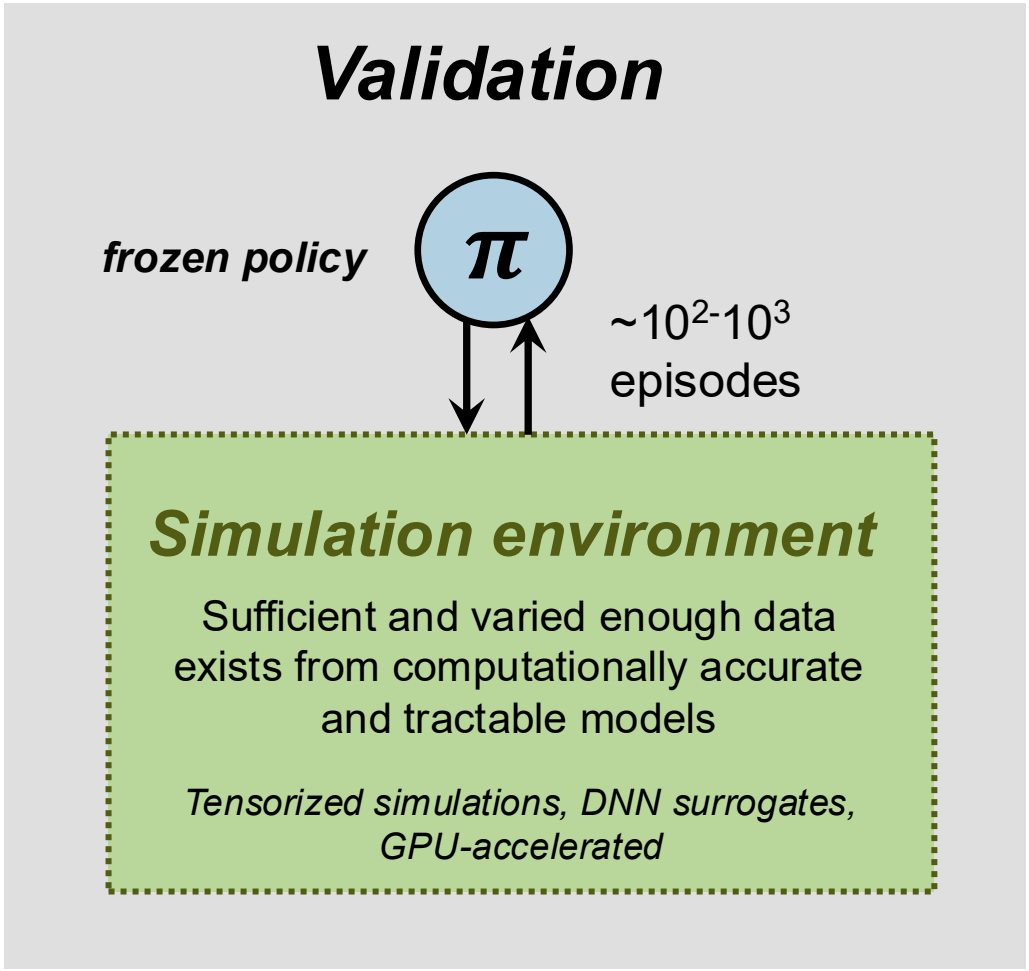
No training dataset, rather an interaction process

- Learn through trial-and-error interaction
- Explore and improve the control strategy
- Optimize over millions of simulated control steps
- Update the policy until convergence

Output: candidate controller

Training produces a **candidate controller**, not necessarily a deployable solution

RL trained in simulation, then deployed



Unbiased estimate of performance

Objective: Evaluate whether the learned policy is sufficiently robust, safe, and reliable for deployment

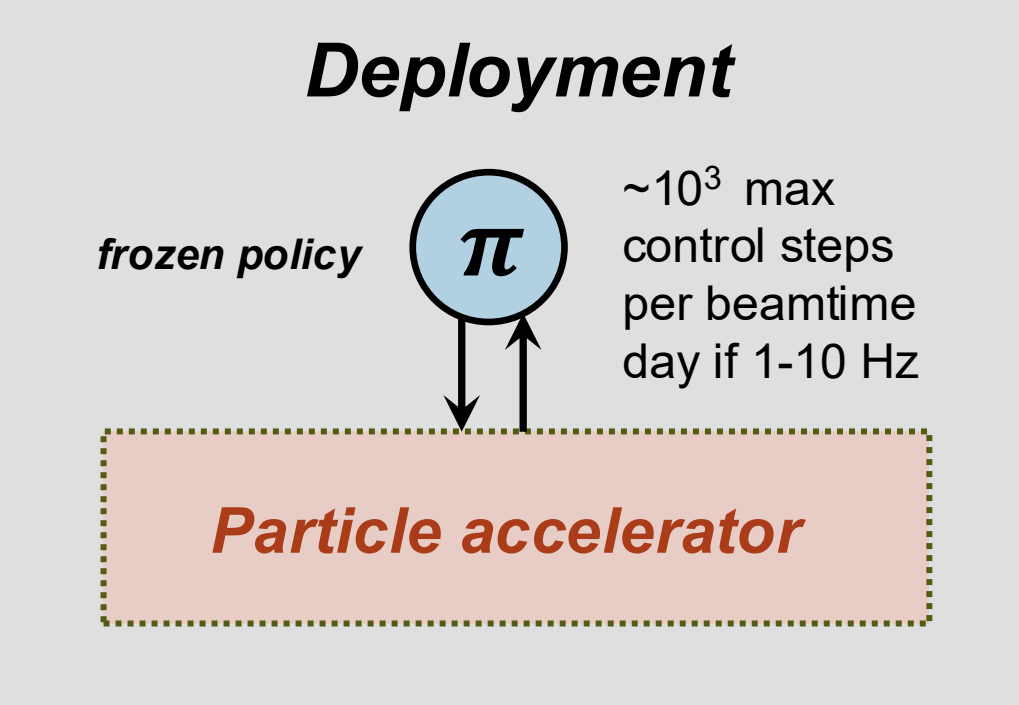
No validation dataset, rather a validation distribution

- Evaluate on unseen initial conditions
- Test robustness and generalization
- Assess operational constraints
- Collect statistics over many evaluation episodes

Output: decision on whether the controller is ready for real-machine testing

Simulation validation **does not** guarantee successful deployment

RL trained in simulation, then deployed



A correct policy **can fail** if the **deployment pipeline** differs from the training assumptions

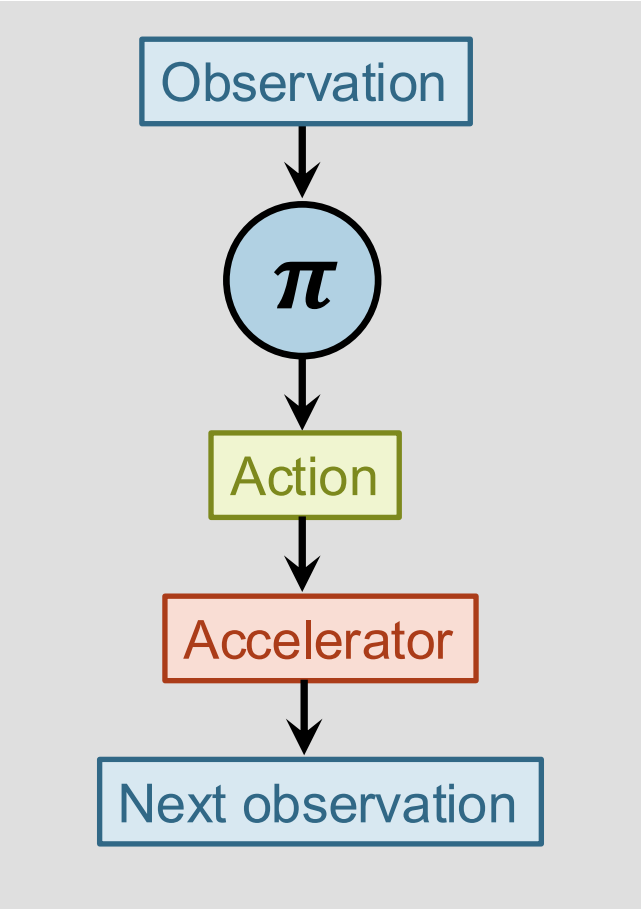
Objective: integrate the learned policy into the accelerator control system while preserving the assumptions made during training

Deployment is an engineering integration problem

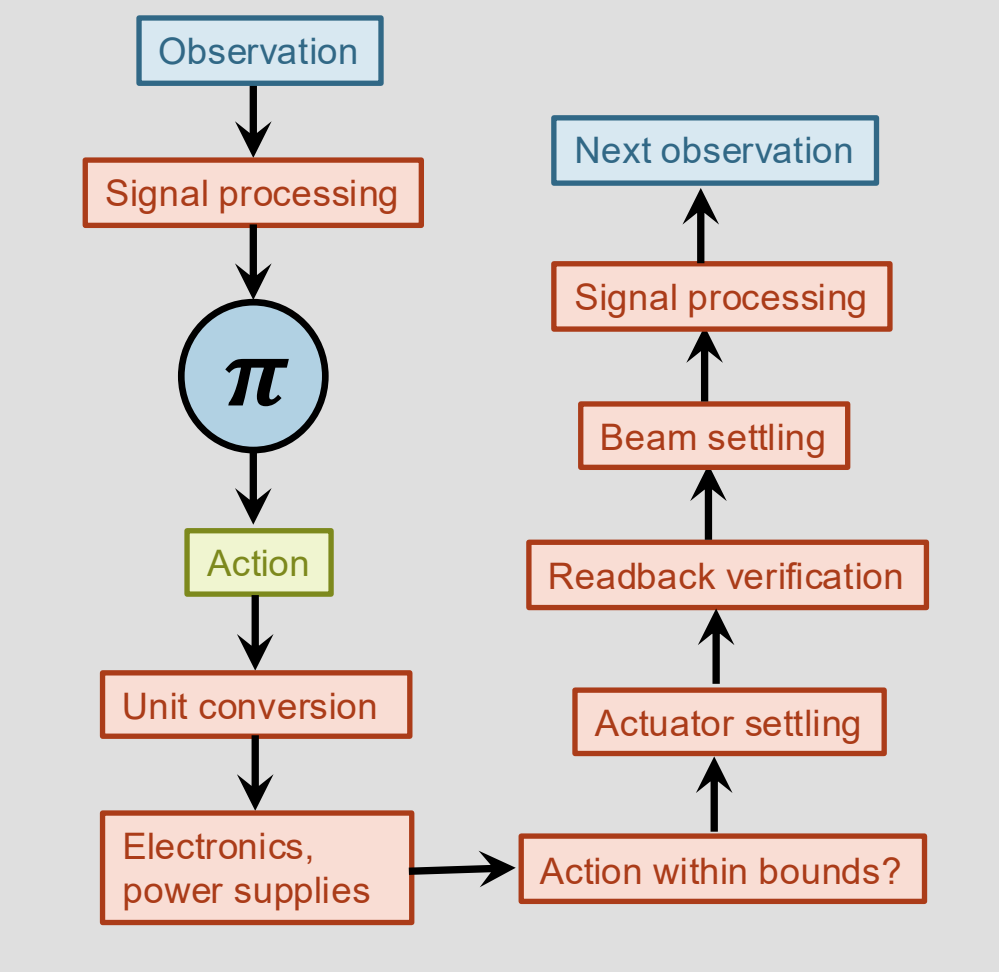
- Map simulation actions to physical actuators
  - Convert sim units to hardware commands
  - Respect hardware limits and interlocks
- Ensure observation consistency
  - Extract the same observables used during training
  - Match signal scaling, calibration, and preprocessing
- Handle operational uncertainties
  - Missing diagnostics or delayed measurements
  - Aleatoric and epistemic uncertainty
  - Out-of-distribution operating conditions
- Synchronize with the control system
  - Wait for actuator settling and verify readbacks
  - Ensure the policy observes the machine after the requested action has actually been applied

RL trained in simulation, then deployed

The policy assumes



Reality

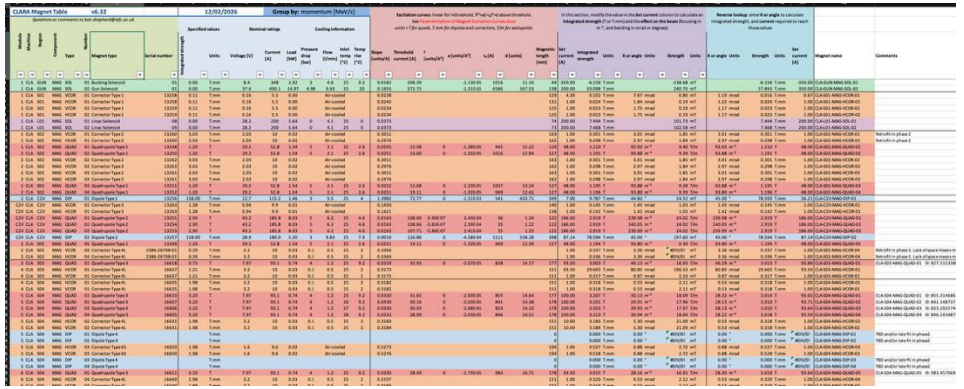


AI accelerator test facility wishlist

AI-native software stack

1. Standardised control system interface

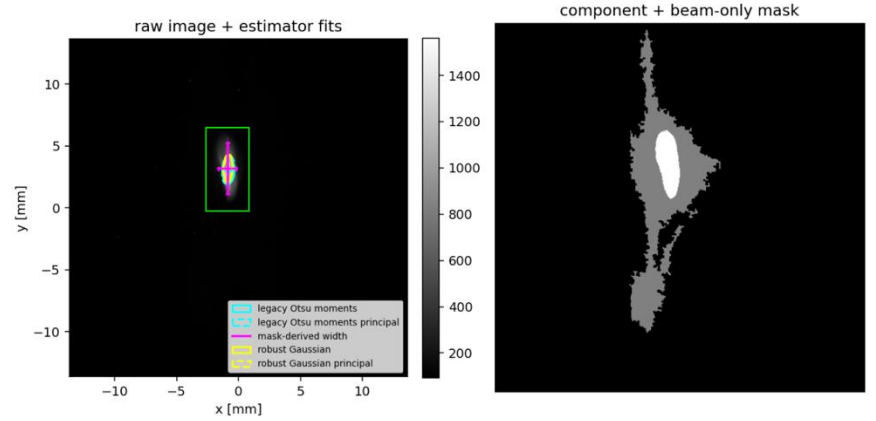
- Well-maintained Python middle layer
- Automatic conversion between physical quantities and hardware units
- Centralised management of actuator limits and interlocks
- Facility-independent software interfaces where possible



Not me dealing with a massive unit conversion excel sheet
But thank you codex for converting it for me

2. Trusted diagnostics layer

- Standardised beam analysis pipelines
- Calibrated diagnostic processing
- Automatic quality metrics and uncertainty estimation
- Consistent observation scaling between simulation and experiment



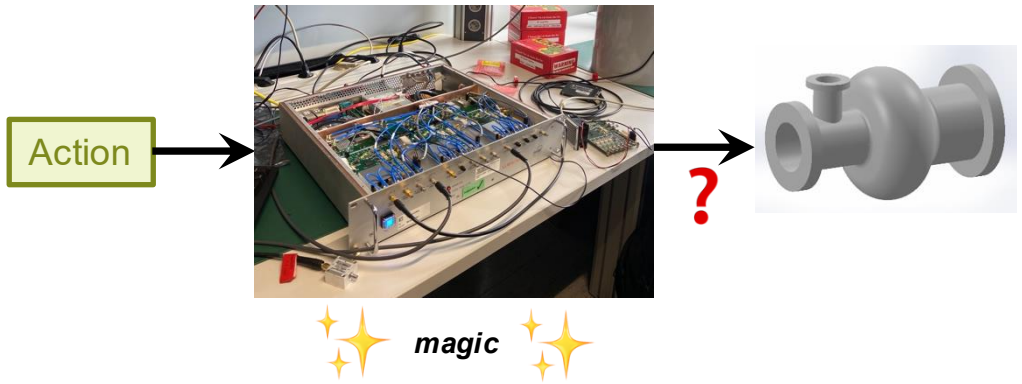
Beams are a lot of things, but nicely Gaussian, they are not

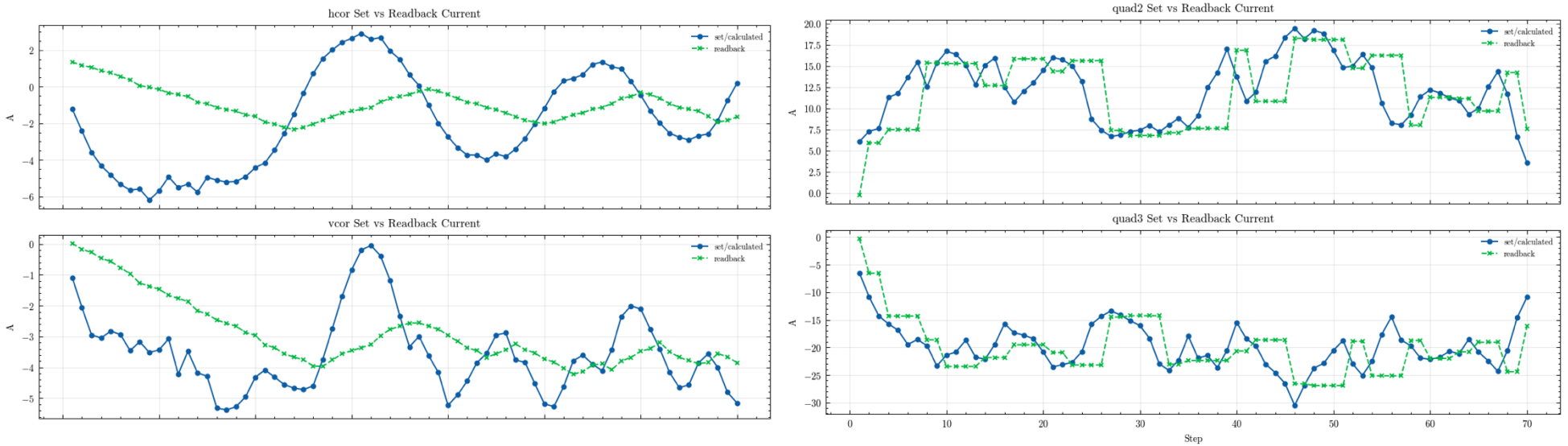
AI accelerator test facility wishlist

AI-native software stack

3. End-to-end system observability

- Verification that requested actions = applied actions
- Readback from the true control point where possible
- Timestamp synchronisation across diagnostics and controls
- Complete logging of commands, readbacks and observations





Set vs readback if you go too fast (no comment)

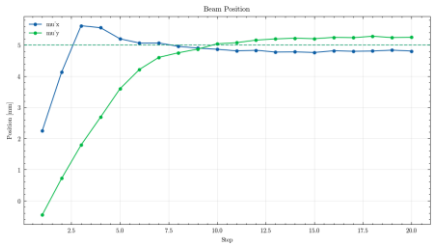
AI accelerator test facility wishlist

AI-native software stack

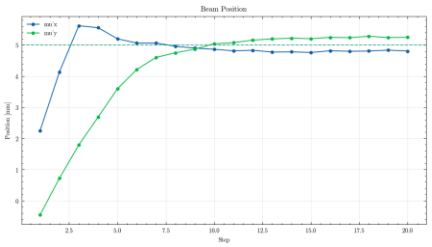
4. Digital twin and continuous model calibration

- Online calibration using experimental data
- Quantification of the sim-to-real gap
- Replay of beamtime experiments in simulation
- Clear on aleatoric and epistemic uncertainties

my agent in real world



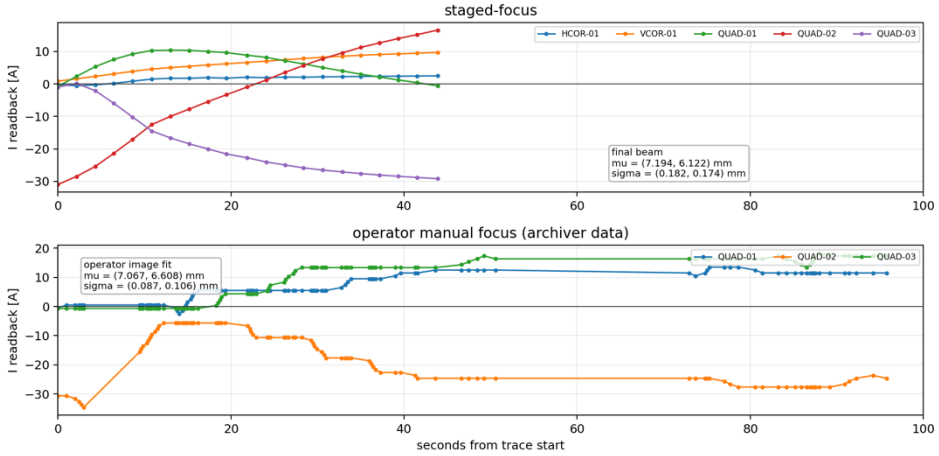
optics calculations



Me looking at model mismatch in real time

5. Benchmarking infrastructure

- Standard benchmark tasks (that are useful)
- Fixed evaluation protocols
- Reproducible initial machine states
- Automated comparison against baseline controllers



AI accelerator test facility wishlist

AI-native software stack

Long story short:

An **AI Test Facility** should provide an **AI-native software stack** that abstracts accelerator-specific infrastructure through standardized, well-maintained interfaces, enabling researchers to concentrate on **algorithm development** rather than rebuilding control, diagnostics, safety, and benchmarking tools for every project

Really important for RL 🤖

Even with a well-trained policy, deployment **could fail** if the assumptions made during training are violated

Observation mismatch

- Observations are processed differently than in simulation (scaling, calibration, feature extraction)
- Diagnostics become uninformative, delayed, or unavailable
- The policy receives observations outside its training distribution

Action mismatch

- Simulation actions are not equivalent to the physical actions applied to the machine
- Unit conversions, actuator dynamics or downstream electronics modify the requested action
- Hardware limits clip or reject actions without the policy accounting for it

Uncertainty mismatch

- Measurement noise (aleatoric uncertainty) exceeds training assumptions
- Model mismatch or unseen machine conditions (epistemic uncertainty)
- Operating points outside the training distribution

Really important for RL 🤖

Even with a well-trained policy, deployment **could fail** if the assumptions made during training are violated

Closed-loop mismatch

- The observation is acquired before the machine has settled
- Readbacks do not correspond to the commanded state
- Delays change the effective system dynamics experienced by the policy

Safety mismatch

- Safety interlocks modify or reject actions
- Emergency procedures interrupt the control sequence
- Missing observations require fallback behavior

Hidden transition mismatch

- The machine responds differently than assumed during training (e.g. actuator dynamics, delays, hysteresis, coupling, drift)

Can be tackled if the policy is fine-tuned in the machine or if considered during training

Edge cases I did not tackle

Light sources that operate at MHz rev. frequencies

- How to handle high throughput data processing
- How to handle microsecond inference
- Might need turn-by-turn diagnostics for observations

Training RL directly in the accelerator

- Experience collection database (complete records, timestamps, replay buffers)
- Suitable local compute for gradient updates
- Versioned model registry of policies, critics, normalization, checkpoints, full provenance
- Rollback and recovery mechanisms
- Safety supervision (emergency termination, fallback behavior, restricted exploration, runtime filter)
- Gated policy promotions

Large-scale facilities

High-dimensional data processing

Managing millions of signals and heterogeneous data ranges

Significant domain expertise

Reliance on expensive physics simulations and expert knowledge

Complex functioning and diagnostics

Maintaining stability across diverse and intricate subsystems

Expensive physics simulations

AI deployment challenges

System reliability and safety

Ensuring machine protection and trust in automated decisions

Sim-to-real transfer

Bridging the gap between simulated environments and physical reality

Nonstationarity and drift

Adapting to evolving conditions and performance degradation

Data limitations

Partial observability, limited diagnostics (narrow, scarce), uninformative, biased

Trust and interpretability

Key for acceptance and adoption of methods

Pilot deployments

↓

AI-driven

↓

AI-native

?

UNIVERSITY OF LIVERPOOL

THANK YOU FOR YOUR ATTENTION!

What questions do you have for me?

Dr. Andrea Santamaria Garcia
Lecturer at University of Liverpool
Researcher at Cockcroft Institute

ansantam@liverpol.ac.uk
https://www.linkedin.com/in/ansantam/
https://github.com/ansantam
https://instagram.com/ansantam
https://www.liverpool.ac.uk/people/andrea-santamaria-garcia