

Virtual to Physical

Reinforcement Learning to Optimize SNS
Particle Accelerator Controls

April 9th 2025

Armen Kasparian

 Jefferson Lab



U.S. DEPARTMENT
of ENERGY

 OAK RIDGE
National Laboratory



Acknowledgment

This research used resources at the Spallation Neutron Source and Jefferson Laboratory and is supported by the DOE Office of Science, United States under Grant No. DE-SC0009915

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

JLab is managed by Jefferson Sc. Assoc., LLC for the US Department of Energy

Mission Statement

Create an automated control system using Reinforcement Learning that can respond to the dynamics of a complex accelerator through continuous interaction

Outline

1. Background Knowledge

- a. What is Reinforcement Learning (RL)
- b. SNS accelerator
- c. Virtual Accelerator (VIRAC)

2. What is our challenge?

3. What do we have to work with?

4. Our approach

- a. Scientific Optimization and Controls Toolkit
- b. Environment

5. Initial Results

6. Current work

Background

What is Reinforcement Learning?

Learning from interaction with an environment to achieve/maximize a long-term goal related to the state

Agent

- Learns to make decisions and take actions to achieve a specific goal or maximize a reward signal

Environment

- Represents the world or system in which the agent operates and learns
- Encompasses everything external to the agent that the agent interacts with, observes, and receives feedback from

In summary: **Instead of learning how we've played the game historically, teach the rules and learn by experience**

Reinforcement Learning Practically: Chess

State: Position of all pieces on board, Whose turn it is, move count etc.

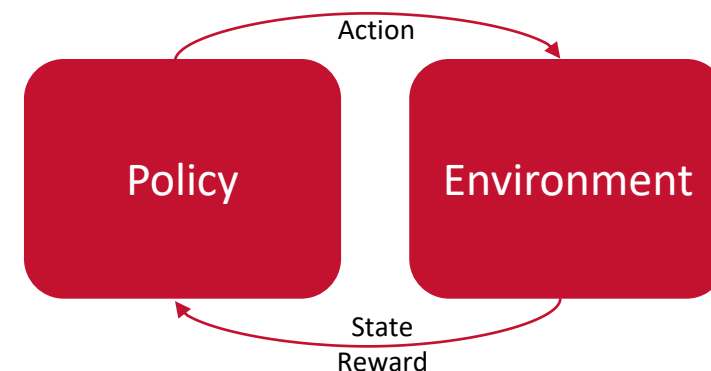
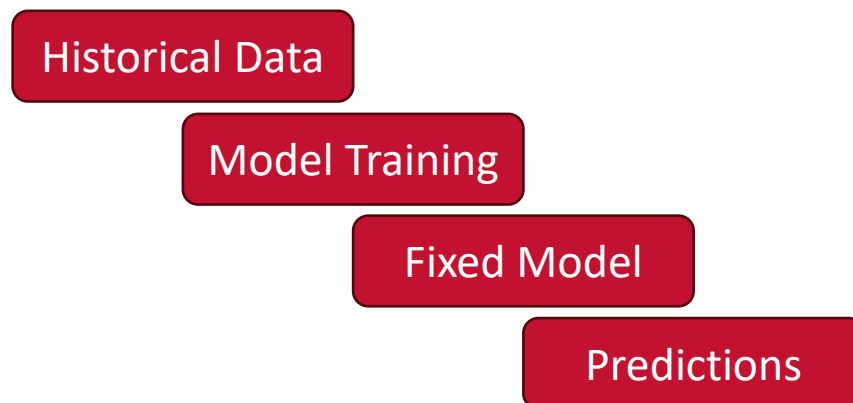
Action: A legal move of a piece

Traditional Machine Learning

- Analyze millions+ historical chess games to identify patterns between board positions (states) and moves (actions) that lead to successful outcomes
- Mimics skilled players of the past

Reinforcement Learning

- Learns by playing millions+ games against itself, receiving rewards for winning positions, gradually improving its strategy through trial and error rather than studying human games
- Unique style to humans



Spallation Neutron Source (SNS)

- DOE research facility operating at 1.7 Megawatts
- Propels bunches (pulsed) protons that collide with a target to create neutrons
- Our initial research focuses on the medium energy beam transport line (MEBT)



MEBT found in linac

Virtual Accelerator (VIRAC)

- Built and developed by the team at ORNL
- Developed in Python
 - Using PyORBIT
- Particle-In-Cell (PIC) models and simulates the dynamics of individual or groups of particles
- Interfaced via the Experimental Physics and Industrial Control System (EPICS)



Virac

Our Challenge

What is our challenge?

- Reinforcement learning agents begin with randomly instantiated models
- Real environments are expensive to train in
 - Starting with a randomly instantiated model may be too costly
 - RL is **not** sample efficient
- Real environments may not be available to train in
 - Real system may not be running
 - May be a long lead time until training runs could be available
- Virtual environments are not perfect
 - Great for exploration but may fall short in certain areas

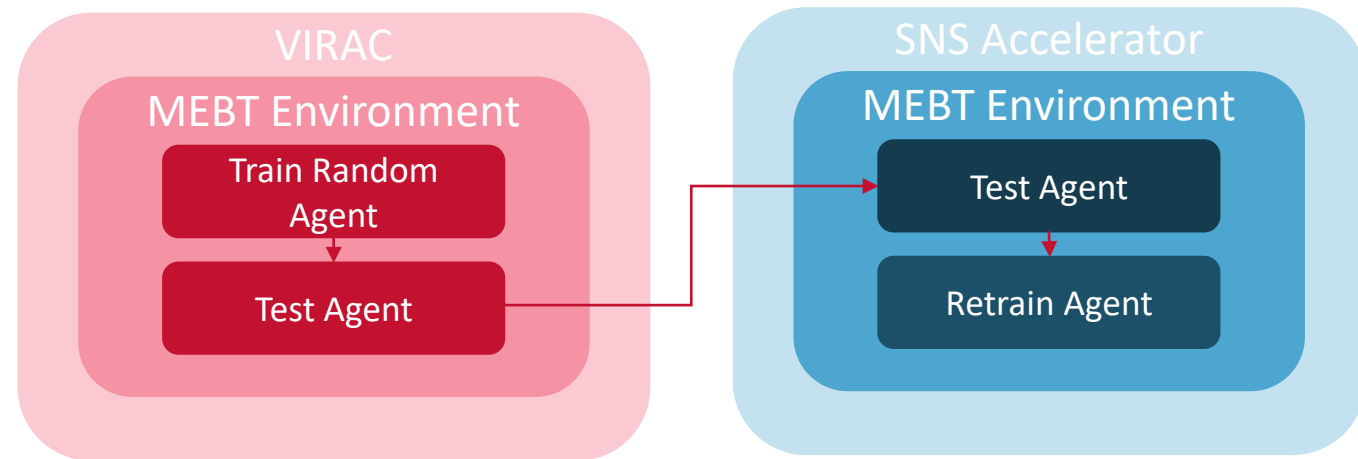
What do we have to work with?

- Historical run data may be available
 - How can we utilize this data to train our agent?
- Virtual Accelerator is available
 - Unknown differences between real and virtual
 - How close is the VIRAC to the real machine?
- What options do we have available?
 - Can we train our agent to mimic to current control solution?
 - Can we pretrain on the VIRAC?
 - Can we make a surrogate based of the historical data and train the agent there? (transfer learning)
 - Limiting factors with uncertainty quantification in state spaces out of distribution

Our Approach and Tools

Our Approach

1. Create an environment **mirrored** on both the VIRAC and the real accelerator
 - State, action, rewards are defined to be the equivalent between the virtual and real system
2. Train the RL agent on the virtual environment
 - Expose the agent to a similar state space it will see on the real accelerator
 - Pretraining models
3. Apply RL agent to the real accelerator
 - Inference first to see how well agent is pretrained
 - Model retraining as needed



Approach

*MEBT Environment does not change, only the system it integrates with changes

Scientific Optimization Control Toolkit (SOCT)

<https://github.com/JeffersonLab/SciOptControlToolkit>

Composable

- Easy to add new agents and environments
- Registration mechanism
 - Allows for easy development and use of new components

Standard Gymnasium API

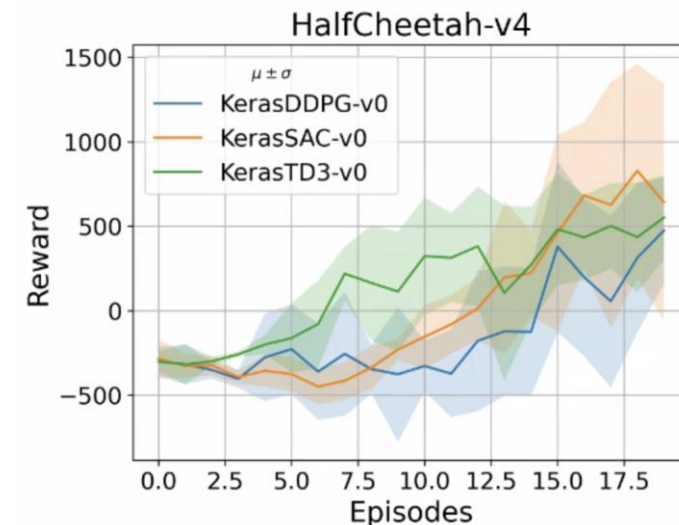
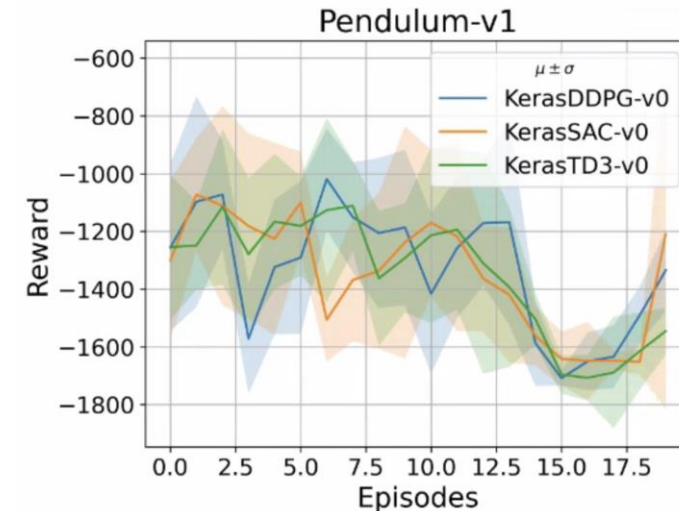
- Utilize industry standard environment specification

Visualizations

- Tensorboard integration allows for live monitoring of training

Reproducible

- Models and configurations are saved and stored for inference/retraining
- Easily pull in previous runs for continued research



Single Plane MEBT Orbit Correction Environment

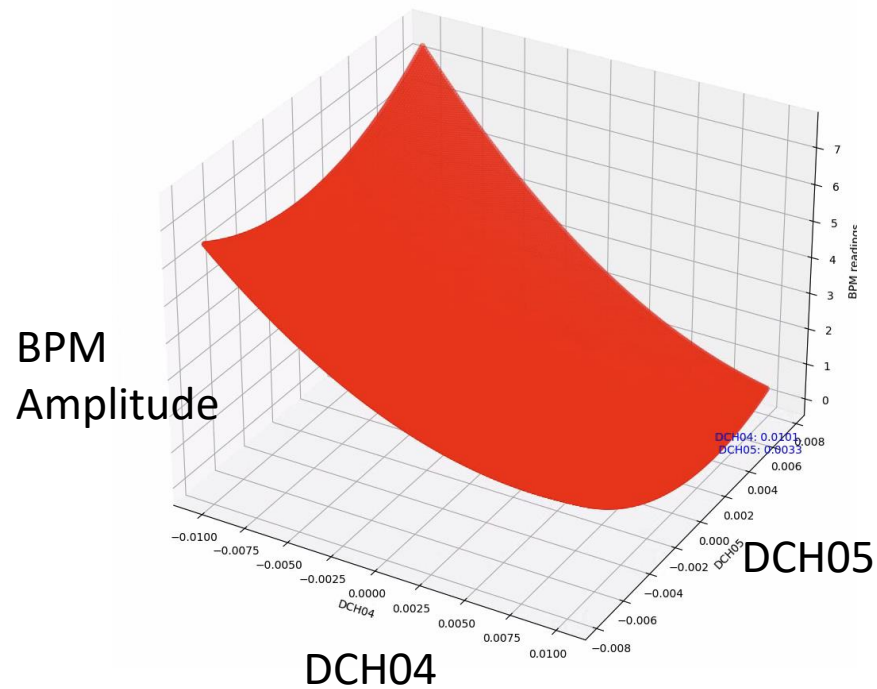
- Custom Gymnasium environment designed and built for communication between RL agent and SNS/VIRAC accelerators
- Goal: Flatten orbit
 - State (6): Correctors (2), BPM (3), BPM Amplitude (1)
 - Action (2): Correctors (DCH04, DCH05)
 - Reward: Maximize the final amplitude reading



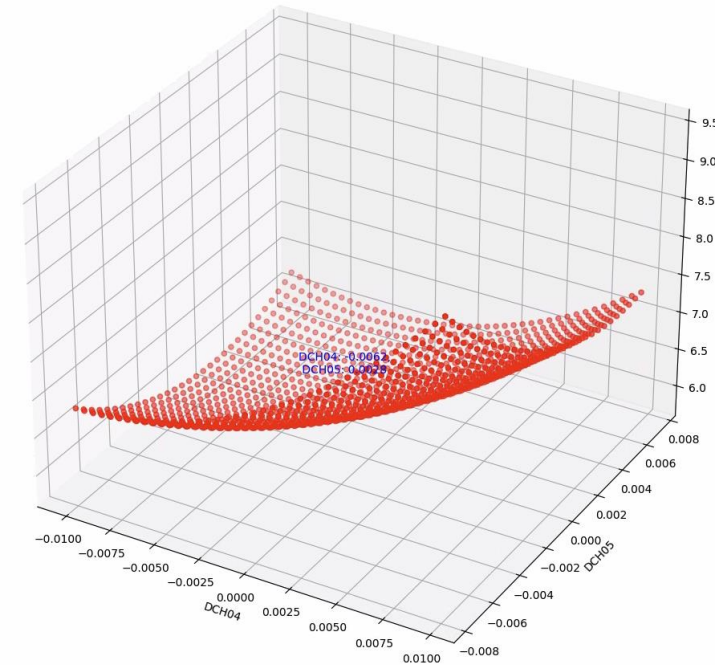
Initial Results

Parameter Sweep

- To understand the landscape of our reward a parameter sweep was done of the VIRAC and the real accelerator
 - Sweep DCH04/05 and log the reward value



VIRAC



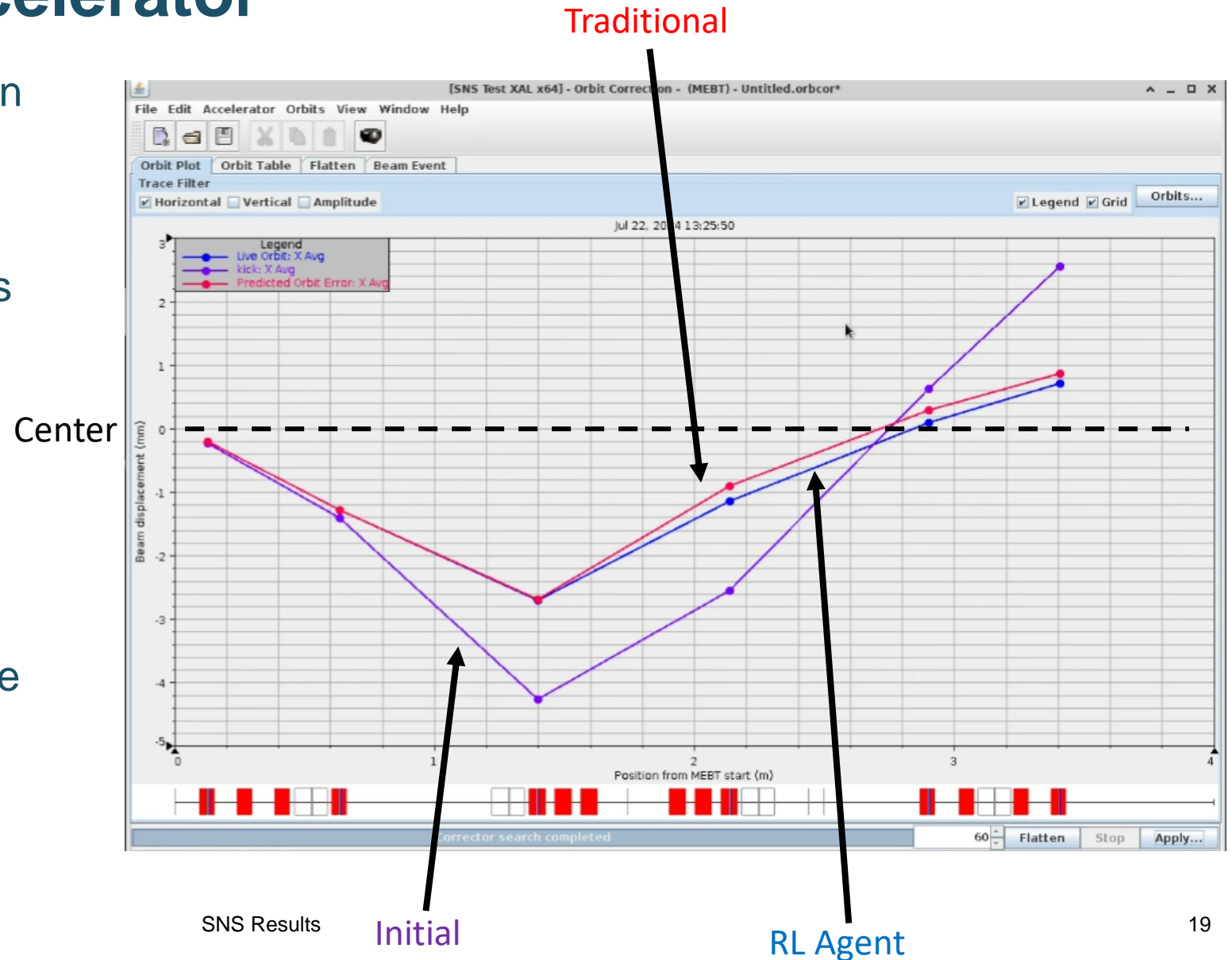
SNS

Inference on SNS Accelerator

Model pretrained on VIRAC applied in inference to the real accelerator

- Purple line shows initial conditions
- Red line shows traditional optimization values
- Blue line shows the RL agents solution

Our pretrained RL model matched the traditional strategies approach

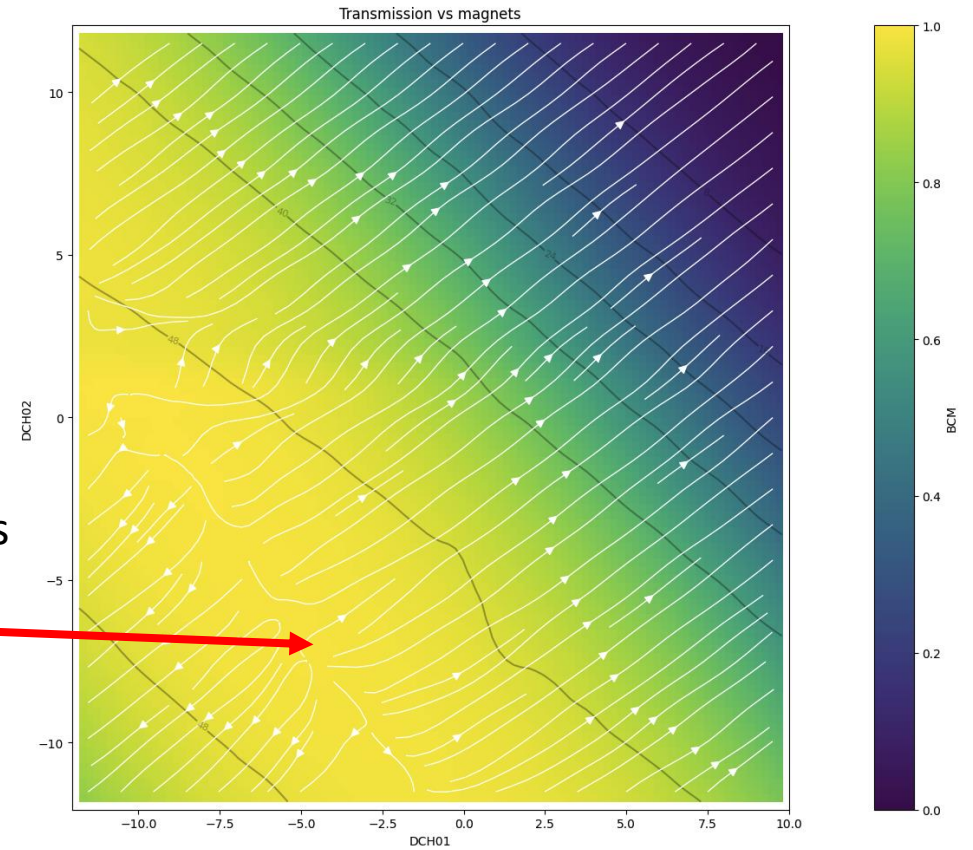


Future Work

Beam Test Facility (BTF)

- Smaller system designed to run many tests at lower energy level
- Linac (similar to MEBT)
- Currently manually tuned with known setpoints
- Can build similar environments to SNS
- VIRAC has a BTF model
- Control vs Optimization

Optimal Setpoints
(-4.5, -7.0)



Real parameter sweep of BTF in test environment

QUESTIONS?

Appendix

Imitation Learning

- Imitation learning is the act of training an agent to mimic the behaviors of an existing systems
- Existing control behaviors can be captured in historical data
- If our agent can learn to act like the existing machine, it can start at a decent initial point and continue exploration from a pre-existing decent solution
- How can we pretrain our model to act as the existing control system?