FERMILAB-SLIDES-24-0012-AD

Machine Learning in Spill Regulation

Aakaash Narayanan

Slow Extraction Workshop, Wiener Neustadt, Austria

12 February 2024

Resonant Extraction for Mu2e

Beam Parameters for Resonant Extraction		
Maximum Intensity in DR	< 1 X 10^{12}	protons
Revolution time in DR	1.694	μs
Orbit length of DR	505.294	m
Horizontal tune in DR	9.650 to 9.666	
Protons per extracted pulse	< 4 X 10^{7}	protons
Spill Duty Factor	>60%	
Single spill duration	43	ms
Beam Power	8	kW

Main types of spill non-uniformities:

- Slow variations in the spill rate
- Fast random ripples (noise)
- Higher order harmonics from power supply

Spill ripples negatively impact the data:

- Detector pile-up
- Reconstruction inefficiency
- Dead time

We need to mitigate:

- Slow varying noise that could span across many spill.
- Fast variations that could arise within one spill.

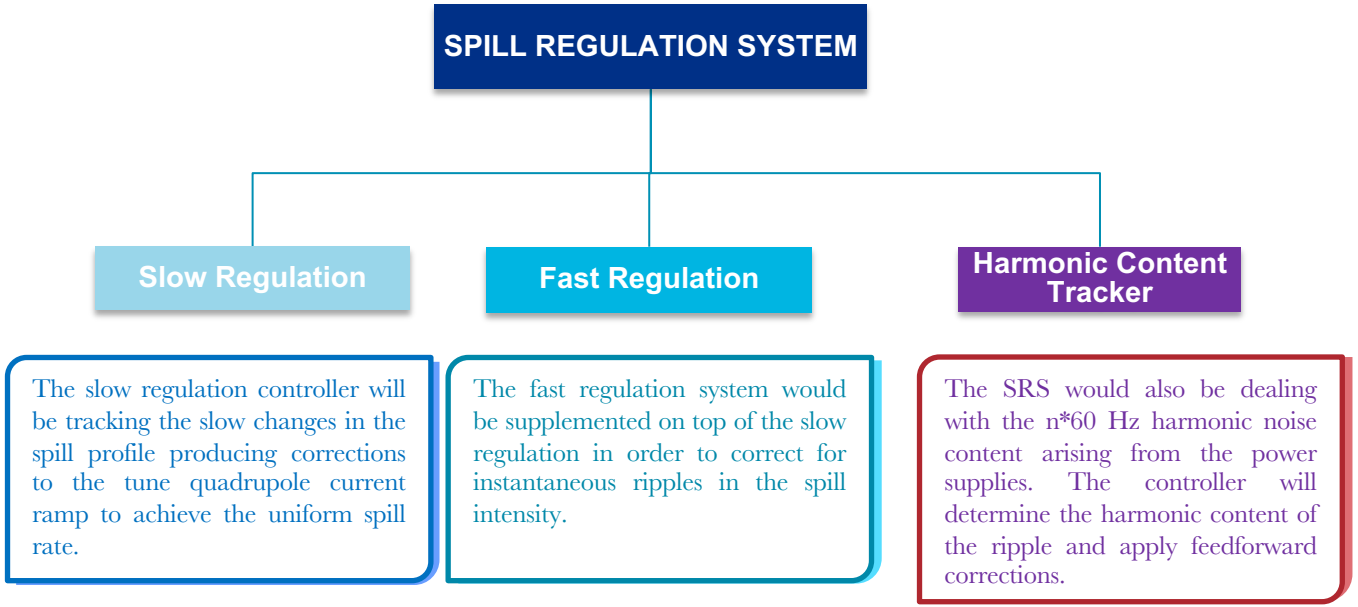
Goal:

$$\text{Spill Duty Factor} = \frac{1}{1+\sigma^2} > 0.6$$

Extraction scheme is to excite 3rd integer resonance using fixed strength harmonic sextupoles and move the beam tune closer to 29/3 using dedicated fast tune-ramping quadrupoles.

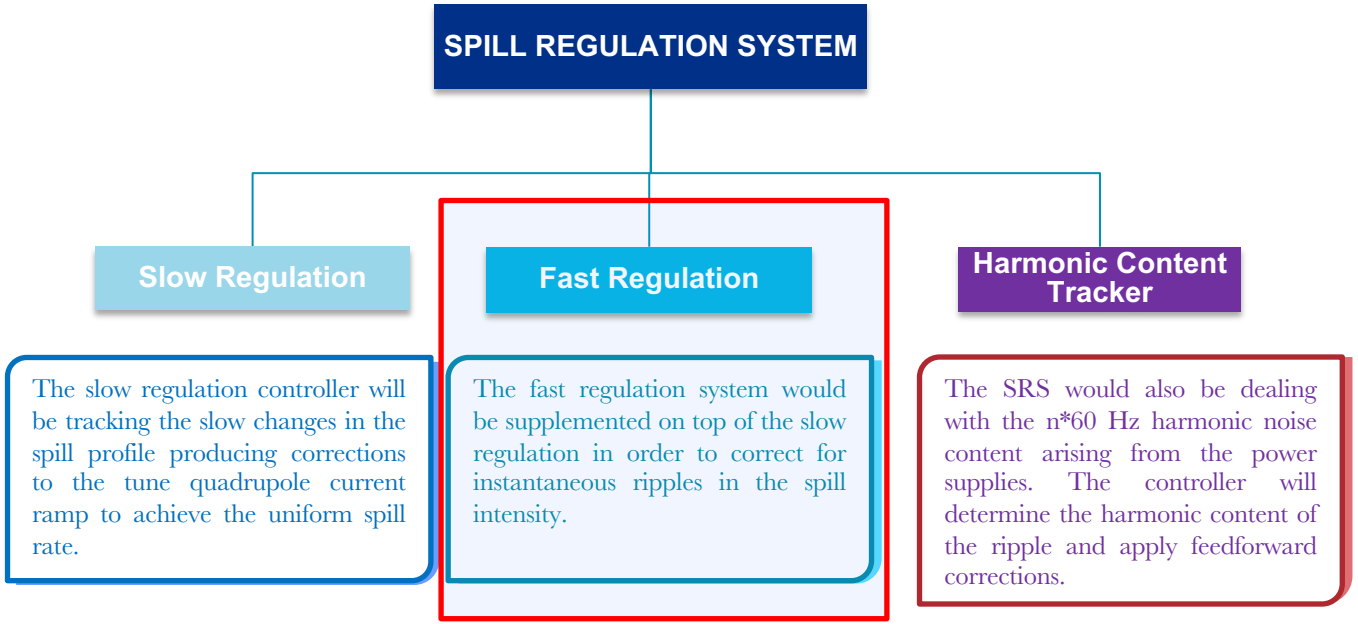
Fermilab

Spill Regulation System



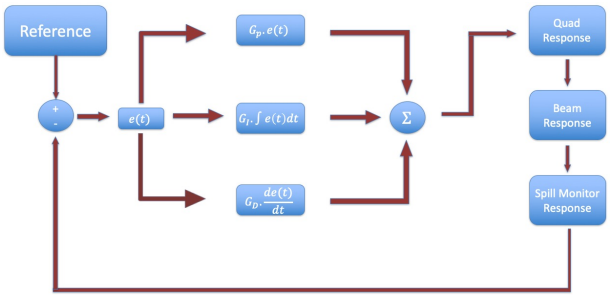
Fermilab

Spill Regulation System



Fast Regulation Loop

The fast regulation loop deals with instantaneous noises that affect the spill quality within one spill.



$$u(t) = G_P e(t) + G_I \int_0^t e(\tau)d\tau + G_D\left(\frac{de(t)}{dt}\right)$$

The instantaneous fluctuations within one spill due to random noises can be large. In the SRS, this is handled by the fast PID loop controller.

We assume here that this noise (ripples) have a random nature or otherwise are a semi-random component of regular harmonic noise that the harmonic controller is not able to suppress.

The correction signal to counter the noise will superimpose on top of the ideal tune ramp provided by the slow regulation.

The three main parameters to control the spill quality in the fast regulation loop are the three gain values of the PID controller: (G_P, G_I, G_D)

Fermilab

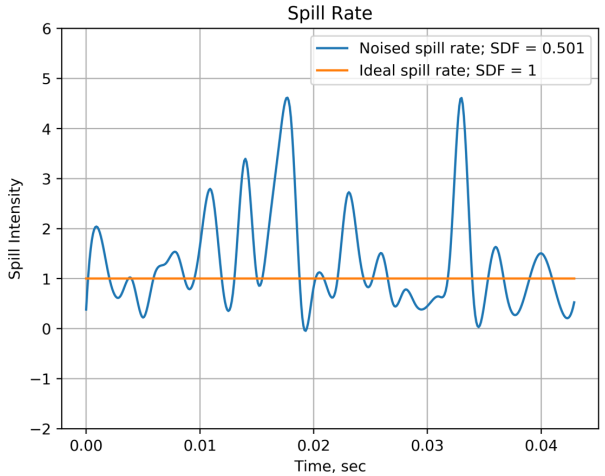
Using Machine Learning

We explore using machine learning (ML) algorithms to tune the PID gain values to achieve an improved spill quality.

The spill quality is quantified by 'Spill Duty Factor' (SDF), defined by:

$$SDF = \frac{1}{1 + \sigma^2_{\{ext.rate\}}}$$

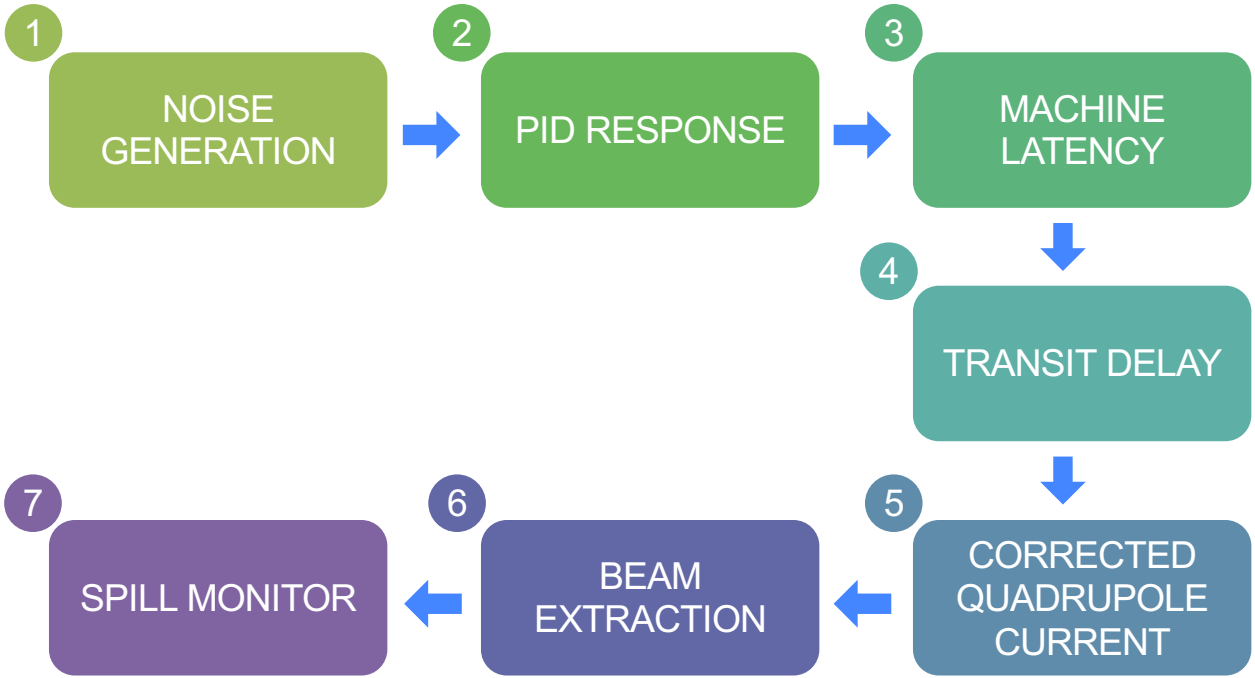
where $\sigma^2_{\{ext.rate\}}$ is the variance in the extraction rate computed for one full spill, assuming the average intensity is normalized to 1.



We use a simplified semi-analytical model to simulate the dynamics of resonant extraction in order to generate training data for the machine learning algorithm.

Fermilab

Physics Simulator Model



Fermilab

Physics Simulator Blocks

1 NOISE GENERATION

The noise for one full spill is generated in terms of extraction rate. We assume a log-normal distribution for the noise spectrum.

Since the source of noise could emanate from any of the elements in the ring, the noise is pre-generated before the spill and is added directly as fluctuations in the spill.

Ideal spill is normalized to an expectation value of 1.

2 PID RESPONSE

With the full extraction rate known, the PID calculates the error at every time step and computes the control signal.

At every time step, the difference between the ideal spill and the actual spill is computed, and the PID calculates the control signal to be given to counter the noise in the spill rate.

Physics Simulator Blocks

3 FIELD SHIELDING EFFECT (LATENCY)

The SS beam pipe would screen high frequency components of the quadrupole $\vec{B}$ field.

The PID response in the simulator is thus passed through a low-pass filter to simulate the steel beam pipe screening any magnetic field variations greater than 1 kHz.

4 TRANSIT DELAY

Once the particle is unstable, it does not get immediately extracted once it gets unstable. It takes some finite amount of turns to get to the septum location.

Transit time studies were done to determine the number of turns particles take to get extracted. This delay is modelled into the physics simulator.

Physics Simulator Blocks

5 CORRECTED QUADRUPOLE CURRENT

According to our analytical model, the ideal tune current ramp is taken to be a logarithmic function in time. The delayed and low-pass filtered PID response is superimposed with the idealized logarithmic quadrupole current curve.

6 BEAM EXTRACTION

With the corrected quad current ramp, the total extracted beam intensity is computed for the full spill duration. This would be the fast regulated spill.

7 SPILL MONITOR

The spill monitor block computes the spill rate for one full spill at a time step of 10 kHz (which is the total gain bandwidth of the SRS). We assume the spill monitor to be fast enough to not affect the loop.

PID Tuning Scheme Using ML Simulator

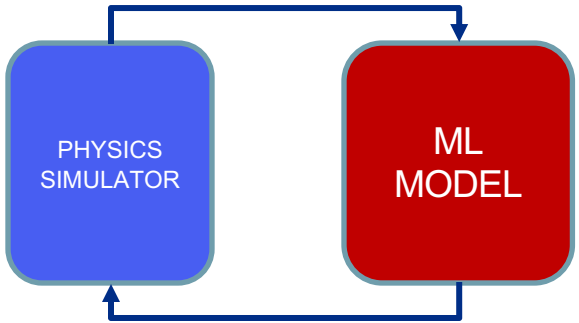
The ML simulator uses the physics simulator iteratively to compute loss functions and optimize the gain values.

In the very first iteration, the neural network assigns random gain values and calls the physics simulator. The physics simulator then outputs the PID loop regulated spill rate, from which the spill duty factor is calculated.

Once the SDF is calculated, a loss function l is defined to train the ML model:

$$l = (1 - SDF)^2$$

EVERY FULL SPILL…



The machine learning tool used in the optimization of the PID gains is PyTorch (version 1.8.1).

Differentiable Machine Learning Simulator

$$l = (1 - SDF)^2$$

The loss value, along with the previous three gain values, are fed into the neural network.

The neural network then calculates the gradient of the loss function with respect to the PID gain (i.e., $\delta l/\delta G$) and backpropogates to update the weights in the direction of minimization of the loss function, outputting new gain values.

Backpropagation is done through with Adaptive Momentum (Adam) optimizer:

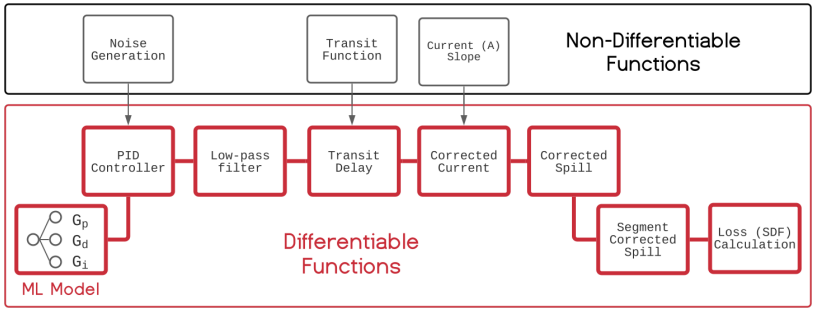


Figure: A. Narayanan, M. Thieme, et. al, "Optimizing Mu2e Spill Regulation Algorithms", IPAC 2021

$$g_t = \nabla_\theta J(\theta)$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1}$$

$$\hat{v} = \frac{v_t}{1 - \beta_2}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}} + \epsilon} \hat{m}_t$$

where:

- θ denotes the network parameters, i.e., weights and biases of the network,
- ∇_θ denotes gradient w.r.t the network parameters, $J(\theta)$ is the loss function,
- $\epsilon, \beta_1, \beta_2$ are constants
- α is the learning rate

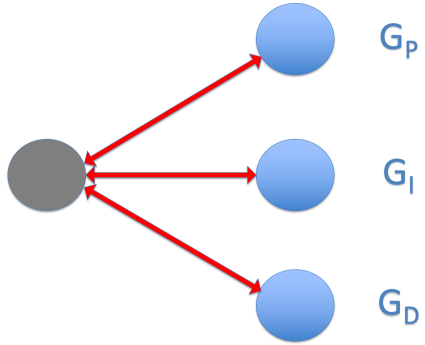
Fermilab

Backpropagation Scheme

With the updated gain values, the physics simulator is run again for a full spill, but this time with a completely new random noise profile.

After the 2nd full spill, the SDF and the loss function are again computed and fed into the neural network to compute the loss gradients.

The neural network again updates the weights and gives a new set of (G_P, G_I, G_D), and the physics simulator is called again.



This is done iteratively until the loss function becomes minimal (i.e., the PID loop's performance becomes maximal).

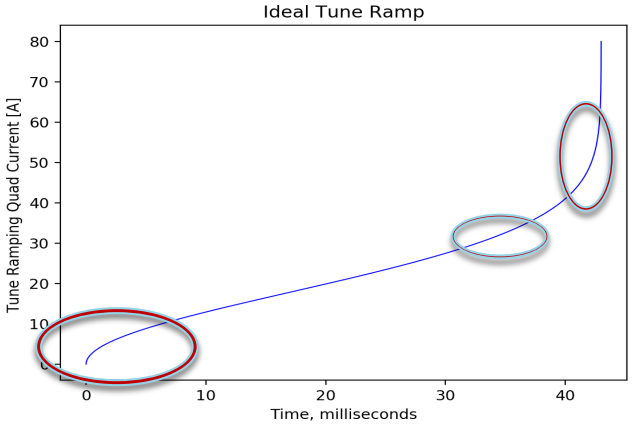
We refer to this approach as a Hybrid ML Simulator because only those functions which must be differentiable (i.e., $(\delta l/\delta G)$ computable) are made so. This allows functions such as noise generation and tune ramps to be pre-computed and excluded from the more computationally expensive gradient calculation and backpropagation steps.

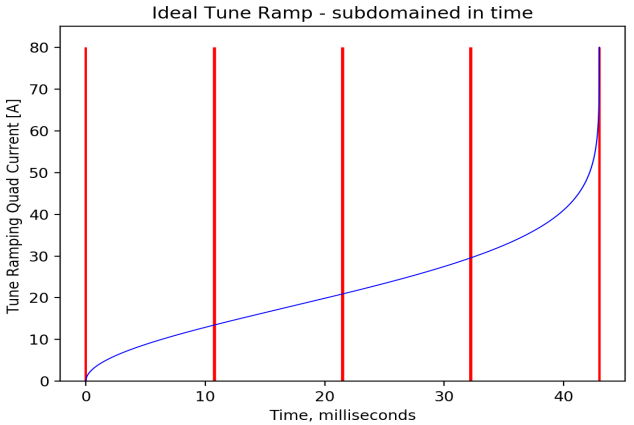
Differentiable Simulator - Results

The sensitivity of control varies across the spill duration.

To characterize this sensitivity, our system divides the full spill into subdomains.

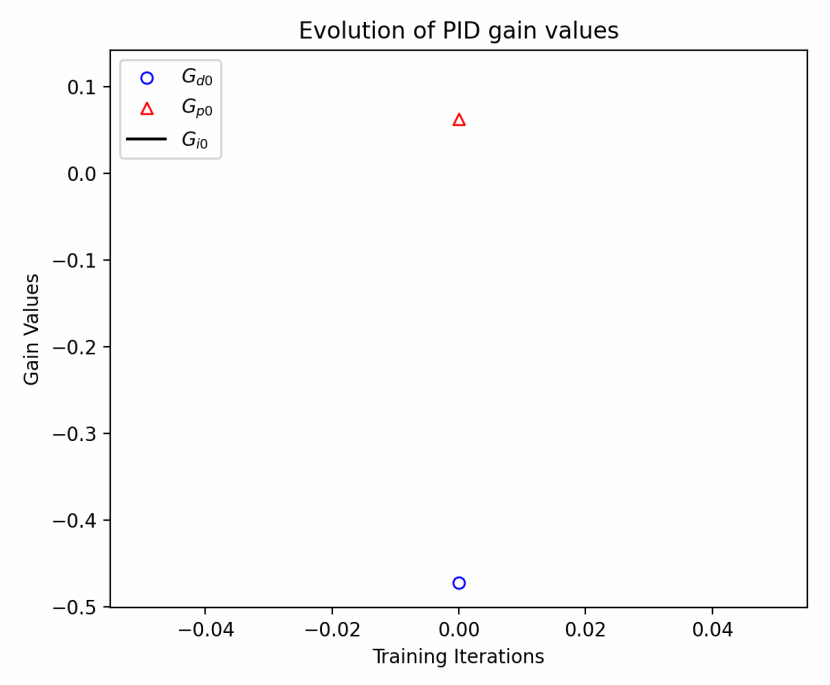
The ML simulator optimizes (G_P, G_I, G_D) within each of the subdomain with every spill.

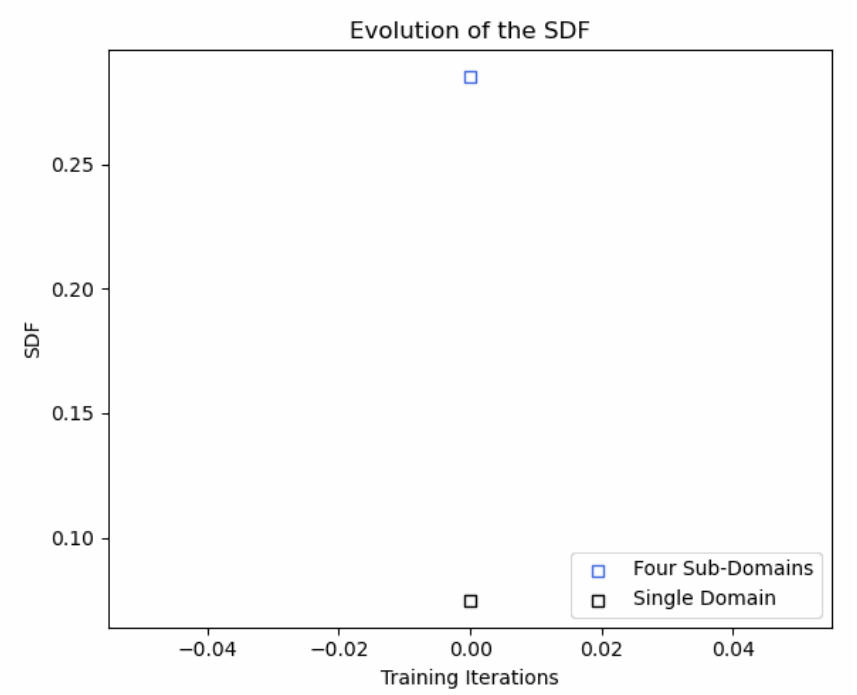




Fermilab

ML Optimization at work…

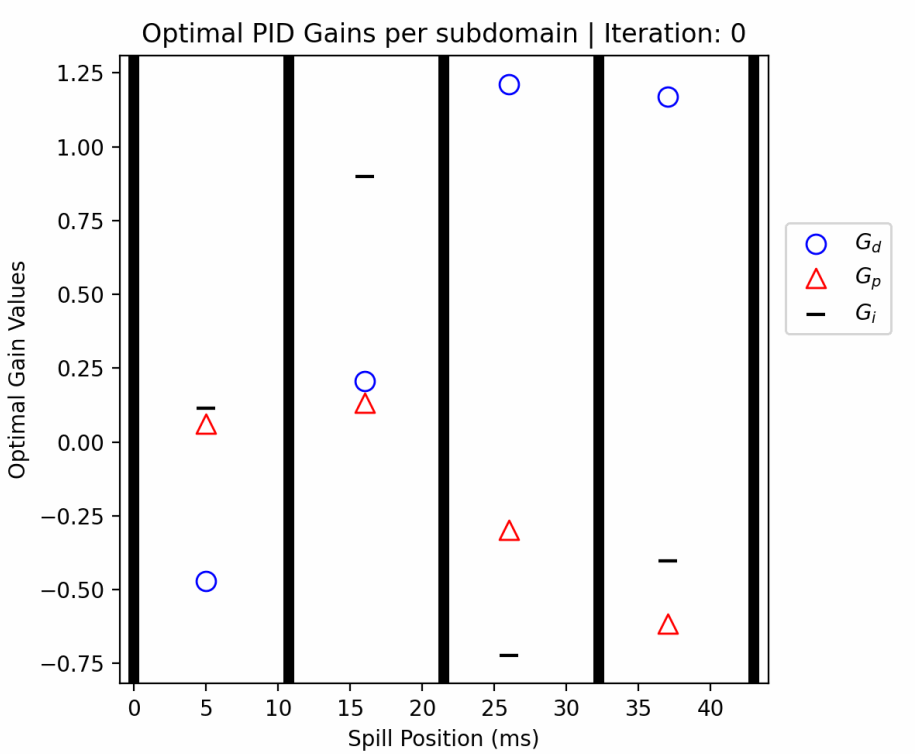




A. Narayanan, M. Thieme, et. al, "Optimizing Mu2e Spill Regulation Algorithms", IPAC 2021

Fermilab

Evolution of PID gains



A. Narayanan, M. Thieme, et. al, "Optimizing Mu2e Spill Regulation Algorithms", IPAC 2021

Fermilab

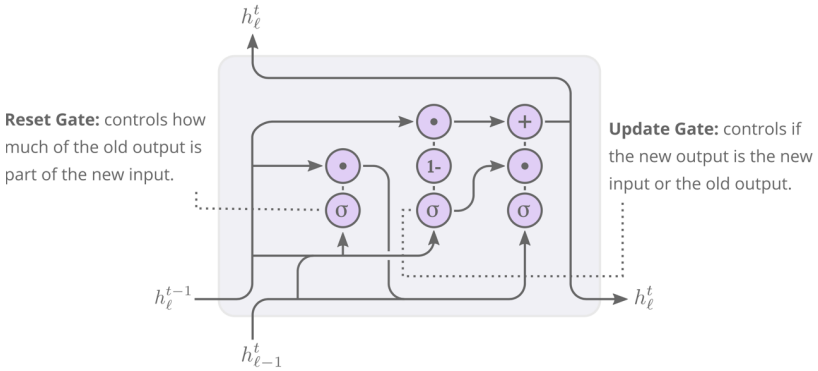
Spill Regulation ML Model

We next explore the possibility of a Machine Learning agent entirely replacing the PID controller, instead of simply tuning the gains of the PID controller.

Since the spill is temporally sensitive, a Recurrent Neural Network was chosen to train the model to emulate the PID controller.

RNNs typically suffer from 'short-term' memory (the 'vanishing gradient problem').

To over come that, LSTM and GRU are a type of neural network that have 'internal loops' that enables connecting temporally sensitive past information to perform present tasks.



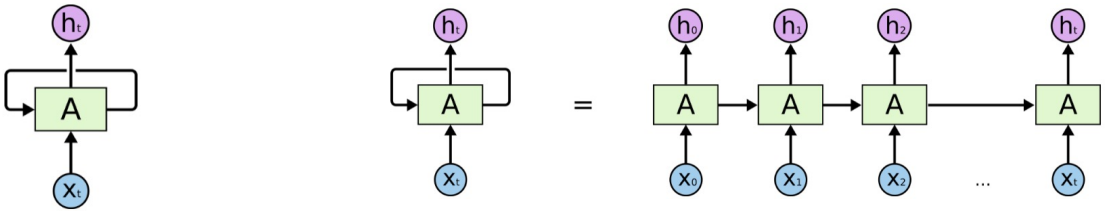
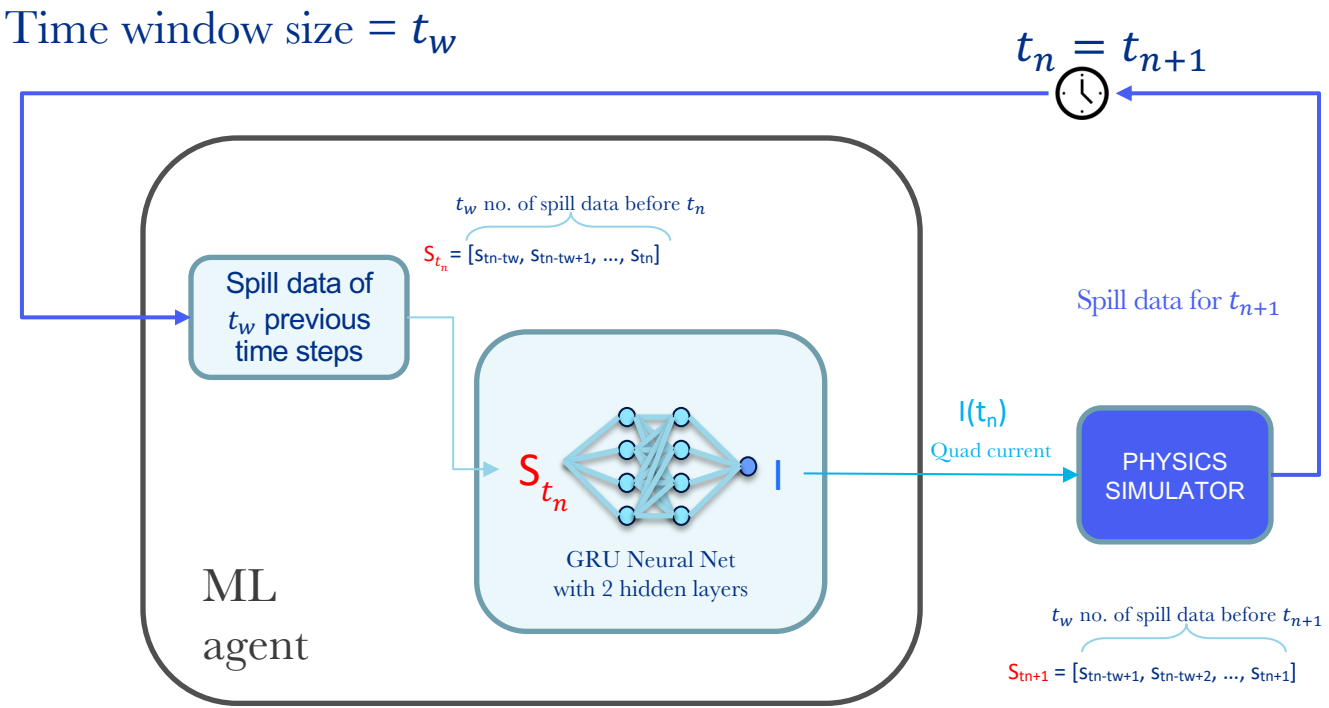


Image source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

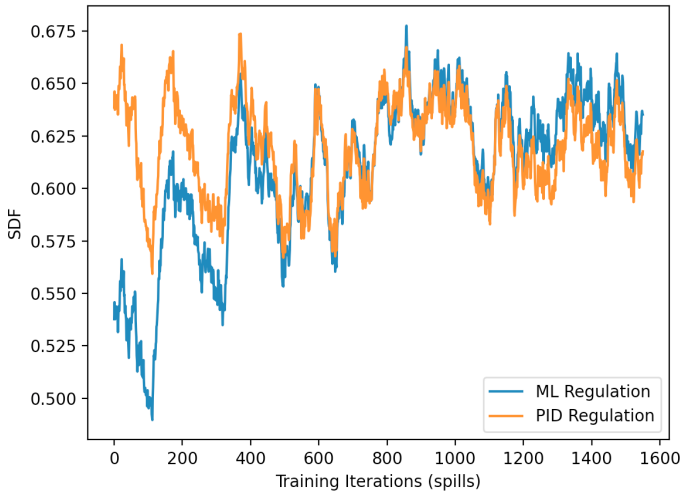
Fermilab

GRU ML Model Replacing PID



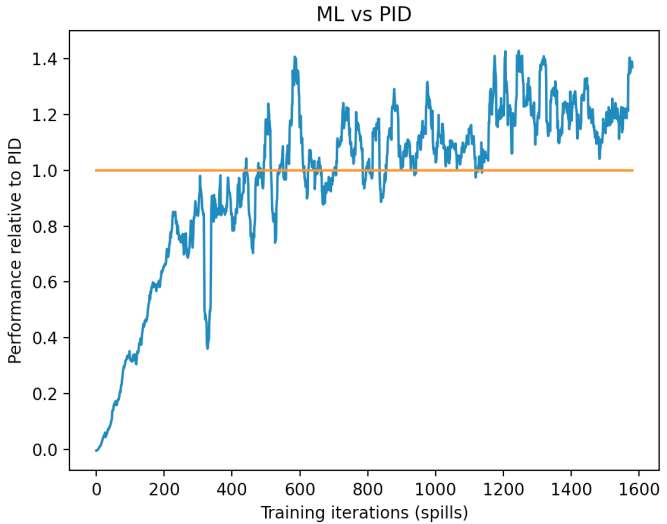
Fermilab

GRU Model Matching PID Performance



Raw SDF

$$\text{SDF} = \frac{1}{1+\sigma^2_{\{ext.\,rate\}}}$$



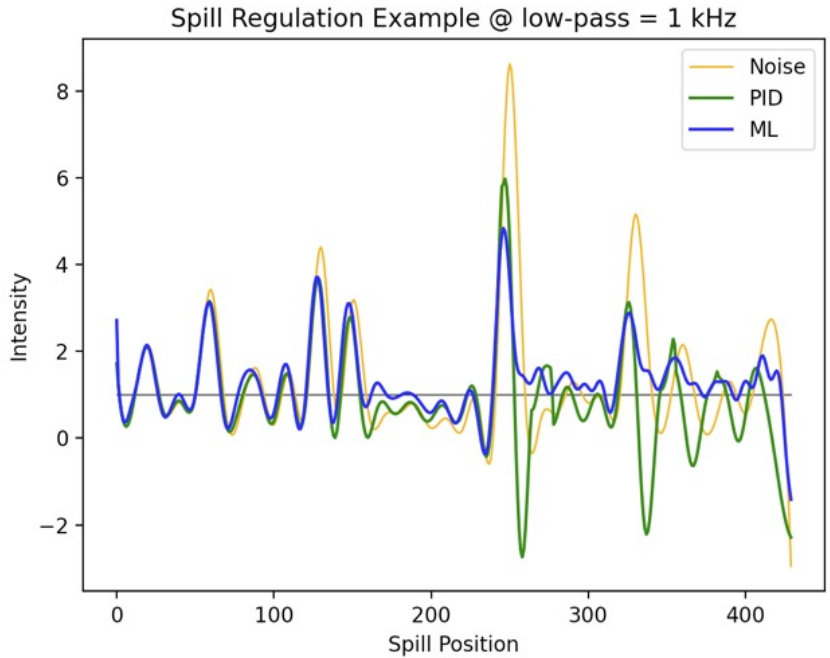
Relative performance to PID

$$\text{Blue line} = \frac{ML_{SDF} - Noise_{SDF}}{PID_{SDF} - Noise_{SDF}}$$

A. Narayanan, J. Jang, M. Thieme, et al., "ML Techniques in Slow Spill Regulation System for Mu2e", NAPAC 2022.

Fermilab

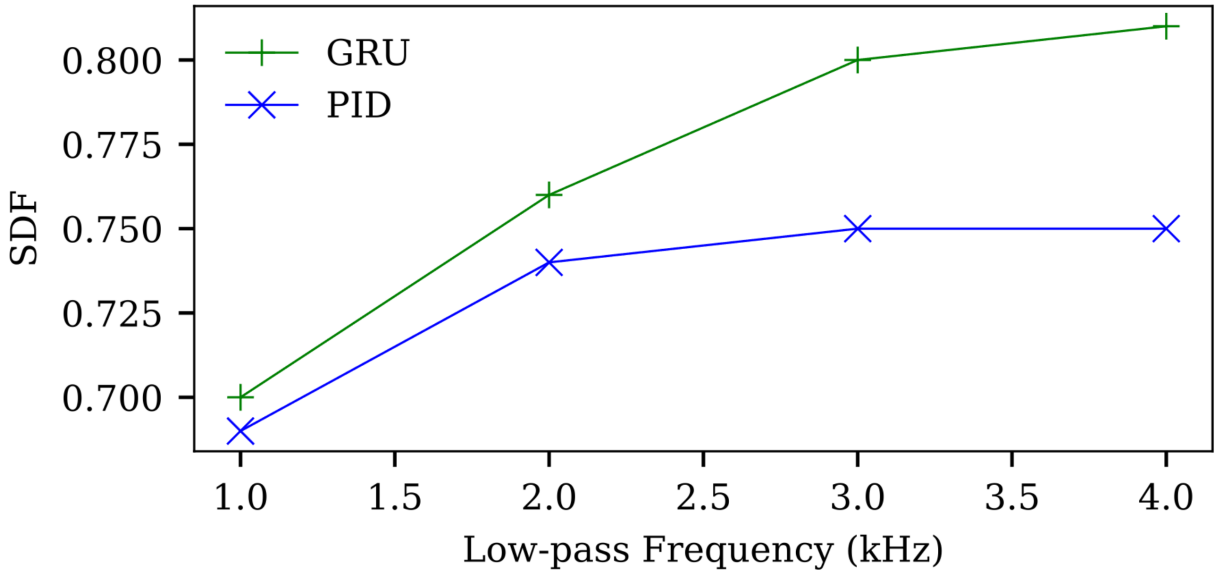
GRU Model Performance



A. Narayanan, J. Jang, M. Thieme, et al., "ML Techniques in Slow Spill Regulation System for Mu2e", NAPAC 2022.

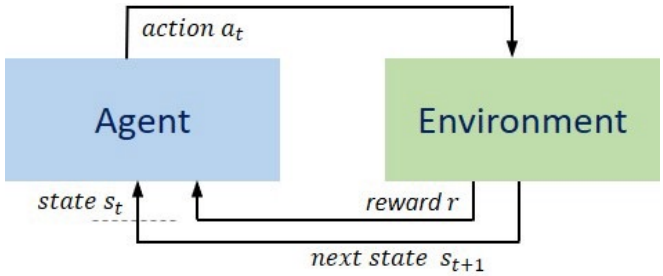
Fermilab

Performance vs Bandwidth



Reinforcement Learning

Reinforcement Learning (RL) is a type of ML technique that enables an agent to interact with the environment by trial and error using feedback from its own actions and experiences.



The action function a is defined as the set of actions taken by our agent in its environment.

The state function s is defined as the set of environment parameters that affect the course of our agent.

The reward function r is defined as a value we assign to a specific action a taken in a specific state s.

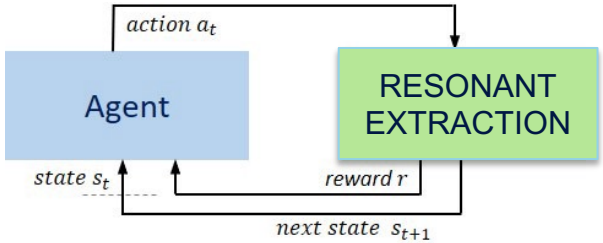
As the agent progresses in time, we accumulate a set of (a, s, r): $(s_0, a_0, r_1), (s_1, a_1, r_2), \dots, (s_i, a_i, r_{i+1}), \dots, (s_T, a_T, r_{T+1})$

Reinforcement Learning in Resonant Extraction Environment

Quadrupole current

The action variable a is defined as the set of actions taken by our agent in its environment.

The state variable s is defined as the set of environment parameters that affect the course of our agent.



The reward value r is defined as a value we assign to a specific action a taken in a specific state s.

In the case of slow spill, the action space is continuous as the control signal's magnitude could be any real number. To deal with continuous action space, we use policy-based actor-critic methods whereby two neural networks are trained.

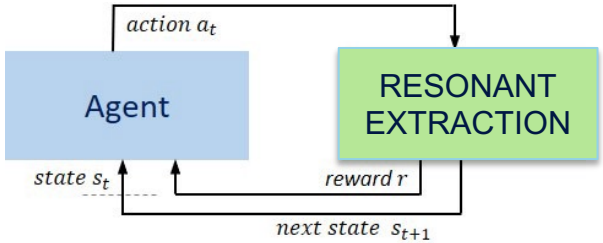
Reinforcement Learning in Resonant Extraction Environment

Quadrupole current

The action variable a is defined as the set of actions taken by our agent in its environment.

$e(t), e'(t), \int e(t)dt, t$, etc.,

The state variable s is defined as the set of environment parameters that affect the course of our agent.



The reward value r is defined as a value we assign to a specific action a taken in a specific state s.

In the case of slow spill, the action space is continuous as the control signal's magnitude could be any real number. To deal with continuous action space, we use policy-based actor-critic methods whereby two neural networks are trained.

Reinforcement Learning in Resonant Extraction Environment

Quadrupole current

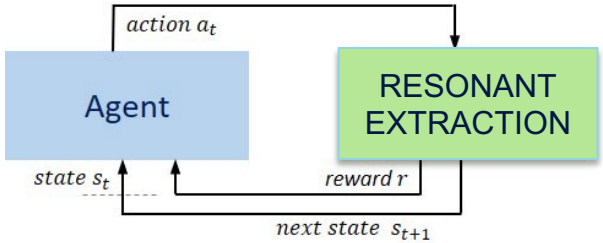
The action variable a is defined as the set of actions taken by our agent in its environment.

$e(t), e'(t), \int e(t)dt, t$, etc.,

The state variable s is defined as the set of environment parameters that affect the course of our agent.

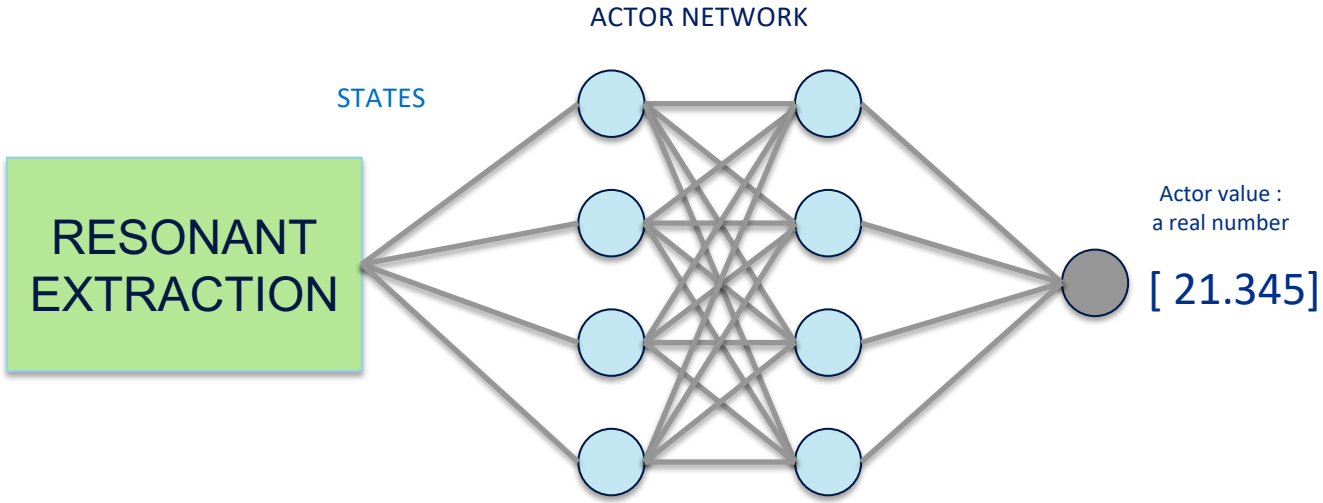
Scaled with regulation performance

The reward value r is defined as a value we assign to a specific action a taken in a specific state s.



In the case of slow spill, the action space is continuous as the control signal's magnitude could be any real number. To deal with continuous action space, we use policy-based actor-critic methods whereby two neural networks are trained.

Reinforcement Learning – Actor Model

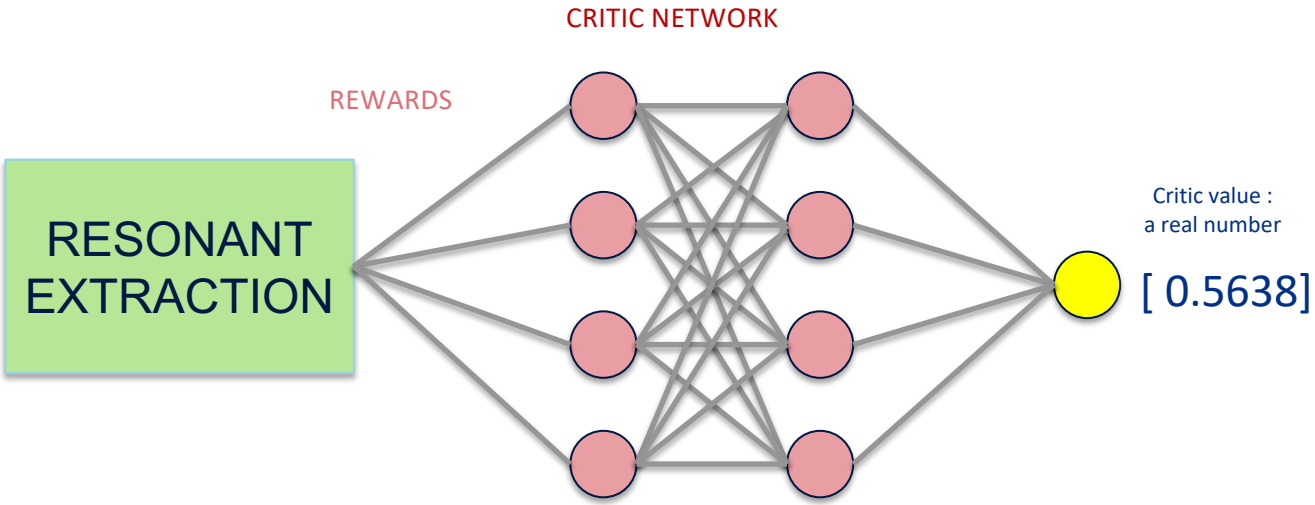


The actor network takes in the state space variables as the input and outputs the action (i.e., the control signal) to be superimposed to the tune ramp quad current.

This action is played for the next time step and the new spill rate is obtained.

Fermilab

Reinforcement Learning – Actor Model

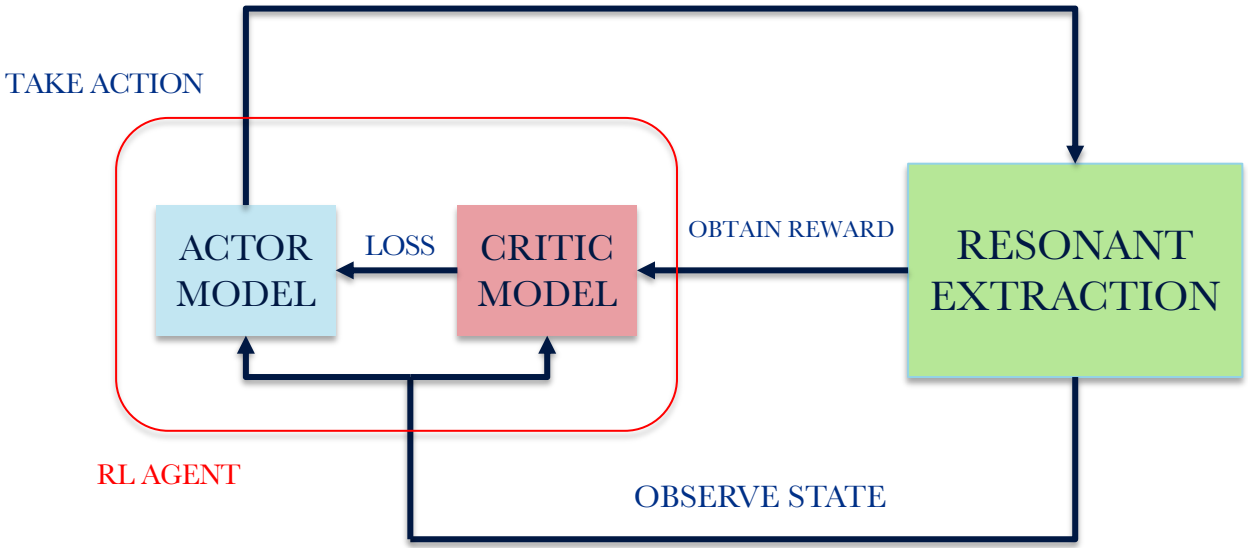


The critic network takes in the reward values for the episode and gives out a value 'criticizing' the how good the action taken was.

This is fed back into the actor network so it takes a better action the next time.

This way, the both the networks together directly learns the policy.

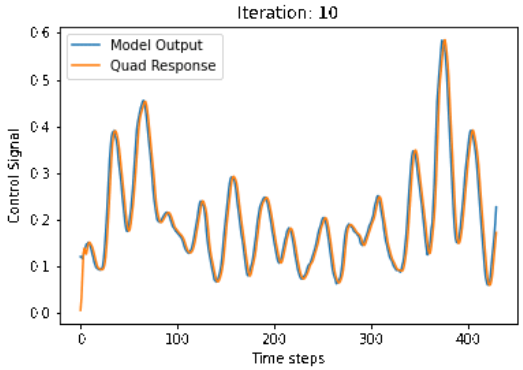
Reinforcement Learning – Actor Critic Method

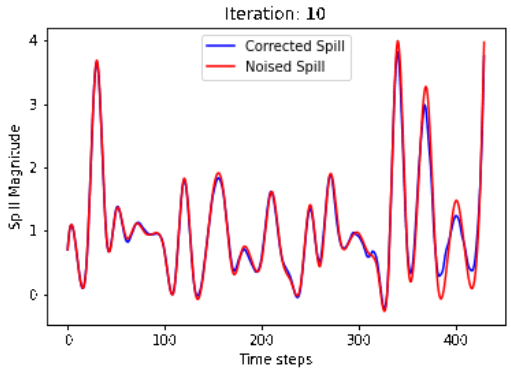


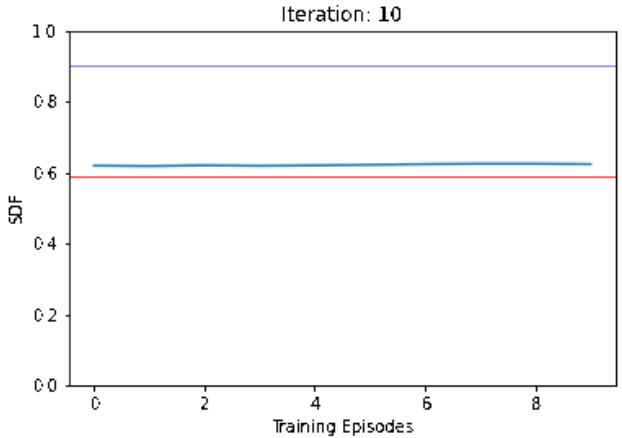
Actor learns to *act* better.

Critic learns to *criticize* better.

Reinforcement Learning – Initial Results

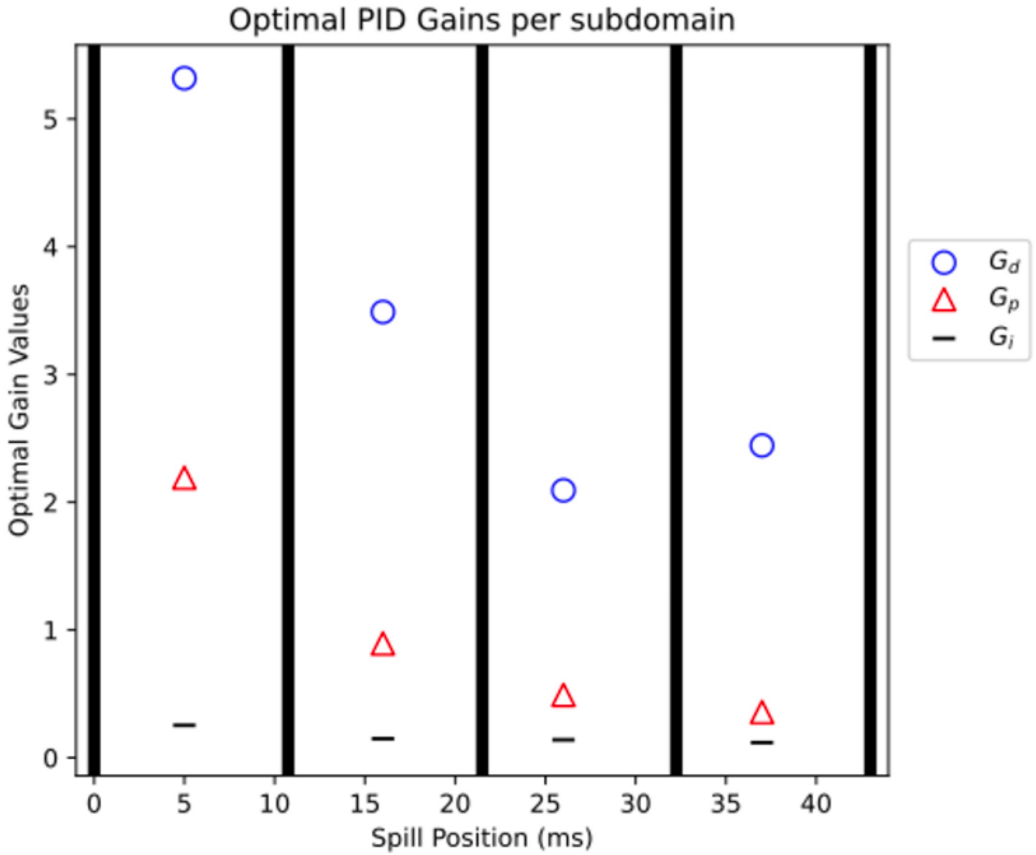




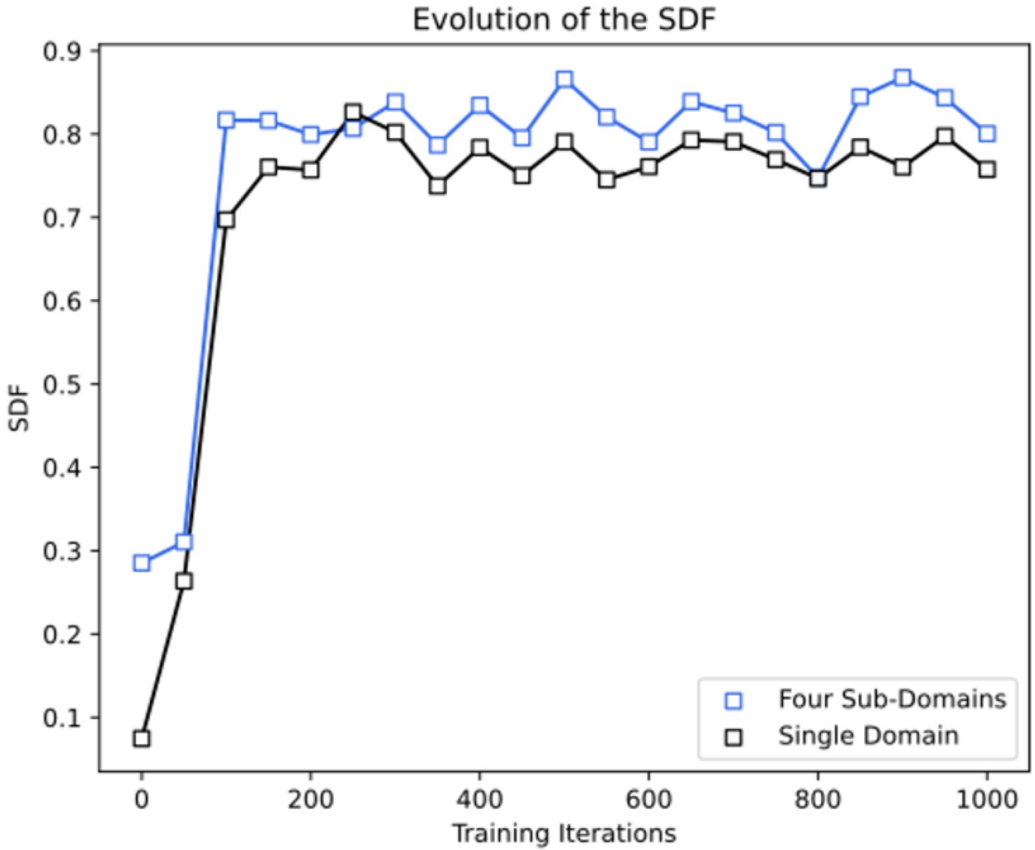


Fermilab

THANK YOU


Optimal PID Gains per subdomain
Optimal Gain Values
Spill Position (ms)
0
1
2
3
4
5
0
5
10
15
20
25
30
35
40
G_d
G_p
G_i

Fermilab

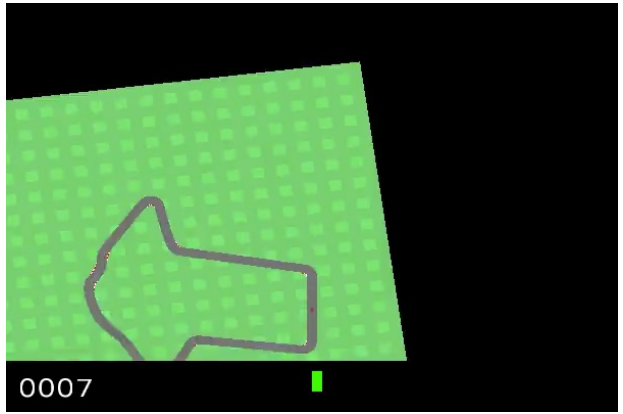

Evolution of the SDF
SDF
Training Iterations
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
200
400
600
800
1000
Four Sub-Domains
Single Domain

Fermilab

Reinforcement Learning – A fun example

Action space:

- Accelerate at constant rate
- Decelerate at constant rate
- Turn right at fixed angle
- Turn left at fixed angle



State space:

- The raw 96x96 pixels of every frame

Rewards:

- -0.1 for every passing frame
- $1000/N_{tiles}$ for every track tile visited
- Episode finished when all tiles are visited

Reinforcement Learning – A fun example