FERMILAB-CONF-22-019-AD

MACHINE LEARNING FOR SLOW SPILL REGULATION IN THE FERMILAB DELIVERY RING FOR Mu2e

A. Narayanan*†[1], J. Arnold, M. Austin, J. R. Berlioz, P. Hanlet, K. J. Hazelwood, M. A. Ibrahim, V. P. Nagaslaev, D. J. Nicklaus, G. Pradhan, P. S. Prieto, B. A. Schupbach, K. Seiya, A. Saewert, R. M. Thurman-Keup, N. V. Tran, D. Ulusel
Fermi National Accelerator Laboratory, Batavia, IL, USA‡
J. Jiang*, H. Liu, S. Memik, R. Shi, M. Thieme*
Northwestern University, Evanston, IL, USA
[1]also at Northern Illinois University, DeKalb, IL, USA

Abstract

A third-integer resonant slow extraction system is being developed for the Fermilab's Delivery Ring to deliver protons to the Mu2e experiment. During a slow extraction process, the beam on target is liable to experience small intensity variations due to many factors. Owing to the experiment's strict requirements in the quality of the spill, a Spill Regulation System (SRS) is currently under design. The SRS primarily consists of three components - slow regulation, fast regulation, and harmonic content tracker. In this presentation, we shall present the investigations of using Machine Learning (ML) in the fast regulation system, including further optimizations of PID controller gains for the fast regulation, prospects of an ML agent completely replacing the PID controller using supervised learning schemes such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) ML models, the simulated impact and limitation of machine response characteristics on the effectiveness of both PID and ML regulation of the spill. We also present here nascent results of Reinforcement Learning efforts, including continuous-action actor-critic methods and soft actor-critic methods, to regulate the spill rate.

RESONANT EXTRACTION FOR Mu2e

Resonant extraction is a beam physics technique employed to extract a slice of beam turn by turn in a circular accelerator by exciting third integer resonance using dedicated sextupole magnets.

Slow extraction at Fermilab is to be done at the Delivery Ring (DR) using a circuit of 6 harmonic sextupoles and 3 fast quadrupoles, driving the horizontal tune from 9.650 to 9.666 to extract 10^{12} protons over the course of 43 ms ($\approx$ 25000 turns) to be sent to the muon production target. Any protons left in the DR after 43 ms would be aborted. The ideal spill quality would be to extract 3×10^7 protons every turn, but we expect the spill quality to be heavily affected with irregularities due to noises that could arise from various accelerator components and other factors.

The Mu2e experiment imposes strict requirements on the spill quality since the detectors are sensitive to the intensity variations of particles coming off of the muon production target. We thus need a robust regulation system in place to ensure high and steady spill quality.

REGULATION SYSTEM

The spill regulation system (SRS) comprises of slow regulation, fast regulation, and harmonic content canceller. The present design bandwidth of the regulation system is 10 KHz, i.e., 430 data points within one spill. A brief functional overview of the SRS can be found at [3].

Fast Regulation System

The fast regulation in the SRS concerns the control of spill quality *within* one spill to curtail any semi-random noise that might arise and also suppress any high frequency 60 Hz harmonic content that the harmonic content suppressor system is not able to suppress. The fast regulation would be supplemented on top of the slow regulation through a fast feedback loop. This loop will send a control signal update at every time step within one spill, and this signal will superpose to the (already preloaded) quad current ramp of the tune ramping quadrupoles to reduce the instantaneous noises in the spill rate.

One way to implement the fast regulation is through PID feedback control, which is a fairly robust and proven technique. But given the non-linearity of resonance process, low spill time, and the strict requirements from the Mu2e experiment on the spill quality, we also investigate possible use of machine learning to enhance the fast regulation.

If the ideal spill rate is normalized to 1, the quality of the spill is defined by the spill duty factor (SDF),

$$\mathrm{SDF} = \frac{1}{1+\sigma_{\mathrm{spill}}^2} \tag{1}$$

where σ_{spill} is the standard deviation of the spill rate. An ideal spill would bear a constant spill rate value of 1, giving us an SDF of 1. The goal of the SRS for Mu2e is to obtain an SDF of 0.6 or higher.

[*] Equal contribution

[†] anarayanan1@niu.edu

[‡] This work was supported by the United States Department of Energy under contract DE-AC02-07CH11359 and the READs project [1, 2], also performed in part at Northwestern University with support from the CS and ECE Departments.

ISBN: 978-3-95450-232-5 ISSN: 2673-7000 doi:10.18429/JACoW-NAPAC2022-MOPA75

ANALYTICAL MODELLING OF EXTRACTION WITH FAST REGULATION

Modelling of the extraction is necessary to validate various regulation schemes in simulations. Particle tracking could be very computationally expensive, we thus developed a simplified analytical model of the spill process to verify the efficiency of different regulation schemes (an overview of the algorithms can be found in [3]).

Physics Simulator

We built a physics simulator in Python (using the analytical model) to simulate the spill process and generate the required spill data to train various machine learning models. The details of the physics simulator can be found in our earlier work [3].

To train a machine learning model to regulate the extracted spill intensity, we need a way to simulate the effects of the models' regulation signals on the extracted spill. This, then, requires that the physics simulator itself must also be fully differentiable, as gradients must be able to flow back from the spill signal, through the physics simulator and into the ML model. To accomplish this, we leveraged differentiable operations in the widely used ML framework PyTorch [4] to produce a fully-differentiable clone of the existing simulator. Every operation in the original simulator, including the low-pass filter for limiting the effective interaction rate of the control system, has been replaced with its differentiable counterpart in PyTorch. This allows us to propagate gradients through the simulator and update the weights of machine learning model. In the following section, we motivate the use of ML in this context, and discuss why our approach is not only effective but theoretically sound.

ML FOR SPILL REGULATION

Our objective is to construct a machine learning system that maps measured spill values (deviations from the ideal spill) onto quadrupole currents that bring the spill intensity back to the target intensity level, thereby smoothing the noise inherent in resonant extraction. Historically, this problem was solved using proportional–integral–derivative controllers (PIDs) which are control loop mechanisms designed to maintain a system in a steady state in the presence of external perturbations. While simple, effective, and well understood, PIDs are a linear and symmetric heuristic control system with constant parameters, meaning they are designed to operate in domains in which the response of the system is invariant across all operating regions. And while approaches such as gain scheduling [5] exist to address this issue, any PID-based regulation system will still be heuristic. As we cannot presume the exact noise distribution and possible nonlinearities in the extraction system, a control system that is 1) nonlinear and 2) capable of adapting to the idiosyncratic response of the extraction system, is warranted. As modern neural networks represent a class of arbitrary nonlinear function approximators, they are a natural solution for extending resonant extraction control systems into the nonlinear regime.

Broadly, the machine learning problem can be formulated as follows. Given an input spill $x \in \mathbb{R}^{430}$, we are looking for a model $f(\cdot)$, parameterized by trainable parameters θ, such that:

$$f_\theta : x \in \mathbb{R}^{430} \rightarrow q \in \mathbb{R}^{430} \tag{2}$$

where q is a sequence of quadrupole correction currents. Note that we do not index the input spills because each spill is generated procedurally and is entirely random. As mentioned before, the effect of these quadrupole correction currents must be simulated using a differentiable simulator we denote $SIM(\cdot)$ which is a differentiable function mapping $q \in \mathbb{R}^{430} \rightarrow s \in \mathbb{R}^{430}$, where s is the final corrected spill. Note that while $SIM(\cdot)$ is differentiable, it does not contain any trainable parameters. The final spill profile is then $s = SIM(f_\theta(x), \lambda)$, where x is the input noise profile and λ is the low-pass filter value that effectively puts an upper limit on the interaction frequency of the regulation system. We include it here because it has a strong impact on regulation performance and is tunable via the use of different materials for the beampipe.

Supervised Learning

In supervised learning, our goal is to learn a model that minimizes the difference between the models' output and the supervision label. As our ultimate goal is to minimize the noise inherent in the spill, we use as our supervision signal the difference between the SDF of the corrected spill (generated by our model) and the SDF of an ideal spill (noiseless spill intensity with a dimensionless value of 1). The training loss ℓ then becomes:

$$\ell = \mathrm{MSE}(s, \vec{\mathbf{1}}) \tag{3}$$

Given the formulation in Eq. (2), a natural choice is a recurrent network, which ingests an input sequence and produces an output sequence of equal size. Towards this end, we implement a simple GRU [6] having two layers, a hidden state of size 128 that ingests, at each point in the spill, a window of the past 40 observations. To allow the model to begin regulation at the first step in the spill, we prepend the spill profile with a vector of length 40. Each element in this prepended vector is assigned the value of the first element in the random noise profile. Then, we walk forward in the usual manner to generate 430 quadrupole corrections. At the end of each spill, the loss is calculated using Equation 3 and the model parameters θ are updated.

Reinforcement Learning

Reinforcement Learning (RL) is a machine learning methodology based on maximizing rewards and minimizing penalties through trial and error. While our RL solution is still under development, we briefly describe and motivate it here. Of primary interest is the capacity for RL models to be trained on real-world observations, i.e. outside of a

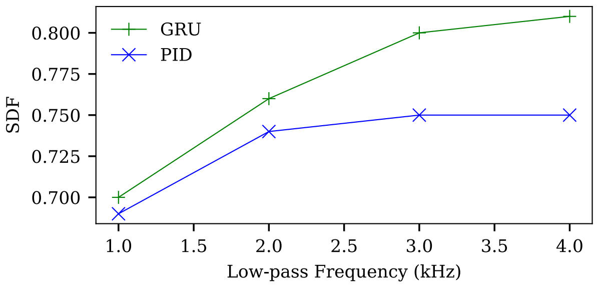

Figure 1: Regulation performance of the GRU vs. PID. Each point represents the average SDF over 1k independent spills with identical noise profiles. Input noise had an average SDF of 0.51 for all trials.

simulator, and, as our ultimate goal is to control a physical device, RL represents a compelling opportunity.

RL can be modeled as a Markov Decision Process (MDP) involving environmental states, actions and rewards. The magnitude of the reward is a function of the state-action pair, which is often dependent on stochastic transition probabilities describing the likelihood of some action ‘successfully’ transitioning the environment from one state to another.

Following the formulation in Eq. (2), our goal is to learn a policy mapping noised spills onto quadrupole corrections. At each point in the spill, our RL agent ingests a window of the past 10 observations, in RL called the ‘state space’, and the produces a single real-valued number corresponding to quadrupole correction, in RL called the ‘action space’. For each spill, the model produces 430 such corrections and our goal is to learn a policy which maximizes the reward (performance) over the entire spill.

To do this, we implement Soft Actor Critic (SAC), which is an off-policy, policy gradient based algorithm that we chose for two main reasons. First, it is designed to balance exploration and exploitation via entropy regularization, and second, it utilizes a method called the double-Q trick to reduce the bias. Both of these help to stabilize training in settings like ours having large or continuous action spaces.

To prevent the errors accumulating, we define a termination condition. If the SDF of the corrected spill drops below a predefined threshold, we immediately assign a large negative reward to discourage the agent from continuing down that path. If this condition is met more than 10 times in a spill, we stop the MDP trajectory and start a new one.

Finally, to improve the sample efficiency, we implement replay memory, which is a technique for extracting more information from each sample by storing it and learning from it multiple times.

RESULTS, CHALLENGES, AND FUTURE WORK

In Fig. 1, we show the average corrected SDF of using the GRU trained in a supervised manner and the PID. We note that this is as fair a comparison as can be made, as we optimize the PID gain values independently for each low-pass frequency using the differentiable simulator introduced in our previous work [3]. The SDF of the input noise for all trials had an average SDF of 0.51. The GRU, trained in a supervised setting, is able to robustly outperform the PID at all low-pass frequencies. Additionally, we see that the advantage increases with the effective interaction frequency, indicating that the value of capturing nonlinear characteristics of the regulation system scales with the interaction frequency. We put a particular emphasis on this point as it is relevant to physical design choices, particularly for those machine elements that limit the beam response time.

As PID controllers are used not just across Fermilab but many engineering disciplines, we hope these results serve as an impetus for continued exploration into learnable control systems in other environments.

RL Challenges and Future Work

While we have built an environment for training RL agents (Fig. 2), performance to date is not competitive with the PID. We believe this to be a consequence of two central features of our problem. First, is the MDP nature of RL. In an MDP, the next state depends on the previous state-action pair, meaning that once the model selects a sub-optimal action, the immediate reward of this state-action pair will be used to optimize the selection for the next policy update, causing errors to accumulate over time. This is particularly detrimental towards the latter end of the spill. In our case, the model relies too much on the exploitation of the present policy, which can leads to a dead-end in the action space. We are approaching the issue by setting termination conditions in our experiment and varying the exploration factor.

The second challenge arises from RL’s heavy reliance on initialization. In our case, as the RL agent can take an action value that is any real number within $[-200, 200]$, the action space is continuous, which makes finding an optimal path time consuming. We are currently exploring the use of a discrete action space as well as methods for carefully selecting larger learning rates.

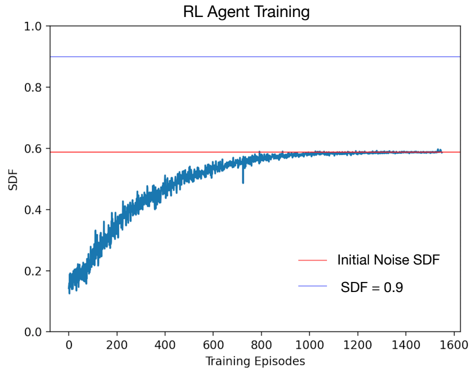

Figure 2: As RL training proceeds, regulation performance approaches the noise level, meaning the RL model is likely learning not to intervene.

REFERENCES

[1] K. Seiya *et al.*, "Accelerator real-time edge ai for distributed systems (READS) proposal," 2021. doi:10.48550/arXiv. 2103.03928

[2] K. J. Hazelwood *et al.*, "Real-Time Edge AI for Distributed Systems (READS): Progress on Beam Loss De-Blending for the Fermilab Main Injector and Recycler," in *Proc. IPAC'21*, Campinas, Brazil, May 2021, pp. 912–915. doi:10.18429/JACoW-IPAC2021-MOPAB288

[3] A. Narayanan *et al.*, "Optimizing Mu2e Spill Regulation System Algorithms," in *Proc. IPAC'21*, Campinas, Brazil, May 2021, pp. 4281–4284. doi:10.18429/JACoW-IPAC2021-THPAB243

[4] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 8024–8035.

[5] Z.-Y. Zhao, M. Tomizuka, and S. Isaka, "Fuzzy gain scheduling of pid controllers," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 5, pp. 1392–1398, 1993. doi: 10.1109/21.260670

[6] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-Decoder approaches," 2014. doi:10.48550/ARXIV.1409.1259