OPTIMIZING MU2E SPILL REGULATION SYSTEM ALGORITHMS

A. Narayanan*[1,2†], K.J. Hazelwood[2], M.A. Ibrahim[2], H. Liu[3], S. Memik[3], V. P. Nagaslaev[2], D.J. Nicklaus[2], P.S. Prieto[2], K. Seiya[2],R. Shi[3], B.A. Schupbach[2], M. Thieme[3†], R.M. Thurman-Keup[2], N.V. Tran[2]
[1] Northern Illinois University, DeKalb, USA
[2] Fermi National Accelerator Laboratory, Batavia, USA ‡
[3] Northwestern University, Evanston, USA

Abstract

A slow extraction system is being developed for the Fermilab's Delivery Ring to deliver protons to the Mu2e experiment. During the extraction, the beam on target experiences small intensity variations owing to many factors.Various adaptive learning algorithms will be employed for beam regulation to achieve the required spill quality. We discuss here preliminary results of the slow and fast regulation algorithms validation through the computer simulations before their implementation in the FPGA. Particle tracking with sextupole resonance was used to determine the fine shape of the spill profile. Fast semi-analytical simulation scheme and Machine Learning models were used to optimize the fast regulation loop.

RESONANT EXTRACTION AT DELIVERY RING

Slow extraction is a well established technique to deliver continuous beams to the experiments. Nevertheless, the spill quality remains to be one of the biggest challenges as the beam intensity and complexity of the experiments are growing. Slow extraction in the Fermilab Delivery Ring (DR) [3] for the Mu2e experiment [4] is achieved by exciting the 3rd integer resonance and by driving (squeezing) the machine tune to the exact resonance value (2/3). The resonance strength is controlled by the two circuits of sextupole magnets and the squeeze is driven by the dedicated circuit of 3 tune ramping quadrupoles (QX). To satisfy those requirements, the Spill Regulation System (SRS) is being developed.

REGULATION SYSTEM FOR EXTRACTION

Spill Regulation System

The SRS will regulate extraction through 2 primary elements: the QX circuit and the RF Knock-Out (RFKO) system. Each regulation element is controlled by separate but concurrent control loops. The SRS has been designed to mitigate several sources of ripple in the spill profile. The SRS system architecture will consist of the System-On-Module (SoM) and a carrier board. The SoM is a FPGA mezzanine card that hosts the Intel Arria10 SoC. Arria10 SoC features a second-generation dual-core ARM Cortex-A9 MPCore processor-based hard processor system (HPS). One ARM Core will be designated for the front-end software application which provides an interface between the FPGA controller and the control system. The second ARM core of the HPS can be dedicated to calculating cycle-to-cycle feedforward corrections or to facilitate machine learning algorithms. Furthermore, the SoM uses bottom FMC connectors to mount onto a carrier board, which, in turn, is contained within a rack wide chassis. The FMC connectors provide two PCIe x8 Gen3 LVDS lanes, to interface with the components on the carrier board. The carrier board hosts the peripherals, which will receive clock signal, timing signals and spill monitoring signals. It will be critical to coordinate the SRS processes within the machine cycle and within each spill interval.

Functional overview

The overarching goal of this work is to develop and study algorithms that would facilitate signal processing in the SRS. There are three main components of the SRS: slow spill profile regulation, fast random ripple regulation and the harmonic ripple content tracker. Here we will discuss the first two of them.

The characteristics of the proton beam in the DR could slowly change with time due to slow drifts in the various accelerator components. The slow regulation controller will be tracking the slow changes in the spill profile producing corrections to the QX current ramp needed to achieve the uniform spill rate. This slow regulation is done adaptively over many spills.

The fast regulation system response, on the other hand, would be supplemented on top of the slow regulation in order to correct for instantaneous ripples in the spill intensity. We assume here that this fast 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. These fast fluctuations can be large. In the SRS, this is handled by the fast PID loop controller.

Slow Regulation Simulation

The algorithms for the slow regulation were tested on a simulation by tracking 132,000 particles in a lattice with

* anarayanan1@niu.edu
† Equal contribution
‡ 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 the Northwestern University with support from the CS and ECE Departments.

a single sextupole. The input for the simulation is the machine's horizontal tune value at every turn. The simulated beam's kinetic energy was 8 GeV (with no momentum dispersion) and a normalized rms emittance of 2.6 π mm-mrad. The time step for the simulation is one turn, and the total time for one iteration is one full spill. For our purposes, we start with a linear tune curve from $\nu_x = 9.650$ to $\nu_x = 9.666$ during the first spill. The simulation tracks the position and angle of each particle at the end of each turn, and any particle whose position is found beyond the electrostatic septum plane is counted as extracted.

For computational purposes, the total number of turns was divided into bins of 250 turns. The number of particles extracted (n_{ext}) is counted bin-wise and is compared with the ideal number of particles that ought to be extracted n_{ideal}. If n_{ext} is greater (or lesser) than n_{ideal}, then the tune distance from resonance for that respective bin is increased (or decreased) by k%. The new modified tune values for each of the bins are then stored and fed as feedforward input to the next spill. This is iteratively repeated until the ideal spill rate (with ±5% error tolerance) is achieved for all the bins. Various k% values were tested and the results are discussed in the poster. A typical result is presented in Fig. 1. The algorithm was successfully able to find the ideal tune curve within a few hundred iterations. In real life, one iteration amounts to a spill duration of 43 ms, thus the algorithm should correct the spill rate within few minutes.

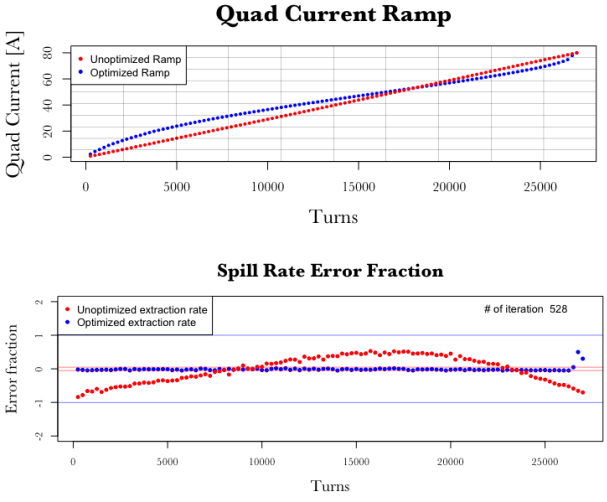

Figure 1: Slow regulation algorithm finding the ideal tune ramp for uniform extraction.

Fast Regulation with PID Controller

As noted earlier, the noise source that demands fast regulation is random (or semi-random) and induces variations in the spill rate within one spill. This is met with a PID loop that would send a control signal superimposed on top of the ideal ramp curve in order to curtail the ripples in extraction. The primary knobs for the PID controller are the three gain values, G_p, G_i, and G_d, where the control signal u is given by

$$u = G_p.e(t) + G_i.\int e(t)\,\mathrm{d}t + G_d.\frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

where e is the error in spill rate. If the PID controller is ideal, the control signal would be equal in magnitude but opposite in sign to the noise profile present within the spill. However, the tuning of the PID controller can sometimes be tricky as the complete loop includes many machine elements and the beam response. While manual optimization techniques exist, we construct an end-to-end machine learning (ML) differentiable simulator parameterized by the PID gains that allows us to optimize them directly from simulated data. In addition to full-spill optimization, the ML system is capable of finding optimal PID gain values in arbitrary subdomains of the spill.

In order to generate the training data for the ML to tune the PID gains, it would be computationally expensive to carry out particle tracking. Instead, the machine ripples and the regulation response can be very adequately reproduced with a simplified analytical model of extraction.

ANALYTICAL MODELLING OF EXTRACTION WITH FAST REGULATION

Simulator goal and approximations

To simulate the fast regulation, we assume that the ideal tune ramp to give the perfect extraction is already in place (because in real life, the slow regulation loop will have already provided that). The analytical model approximates the ideal quad current ramp to be a logarithmic function of time, on top of which the fast regulation performance is simulated. The goal of the physics simulator is to test the performance of the PID loop in terms of spill quality quantified by the spill duty factor (SDF), defined as $1/(1+\sigma^2_{\text{ext. rate}})$. The spill rate in the physics simulation is normalized to an expectation value of 1. The time step of the simulation is done at 10 kHz as the SRS has been designed to have a total Gain-Bandwidth of 10 kHz [5].

Analytical Modelling - Fast Regulation Simulation

The ripples in the extraction rate are generated in the physics simulator using a randomwalk log-normal distribution. The PID loop (loaded with the 3 gain values) reads the rippled extraction rate and computes the control signal to be sent to the quads in order to suppress the ripples. Since this PID response is superimposed with the quad current, the semi-analytical model is involved in calculation of the fast extraction rate changes. Also the fast quad modulation is passed through a Butterworth low-pass filter to simulate the steel beam pipe shielding quadrupole's magnetic field variations higher than 1 kHz. When a particle finds itself outside the stable region in phase-space, it will take a finite time to transit to the septum to get extracted. The nature and extent of the delay was studied in earlier simulations and an appropriate time delay transit function was constructed.

The low-pass filtered PID response is convolved with this transit time delay function. The delayed and low-pass filtered control signal is then superimposed with the idealized logarithmic current ramp, and the total number of particles extracted at every time step is computed along with the extraction rate, which is the output of the physics simulator. The ML simulator uses this output to find optimal gain values for the PID loop to maximize performance of the fast regulation loop.

MACHINE LEARNING VIA DIFFERENTIABLE SIMULATION

Hybrid Machine Learning Simulator

At the base of our ML simulator is a neural network which maps a constant scalar input $\in \mathbb{R}^1$ to a vector of PID gains $G \in \mathbb{R}^3$. With each training iteration, PID gains generated by the neural network are ingested by the PID simulator to produce a corrected spill $c \in \mathbb{R}^{430}$ (10 points per millisecond in the 43 ms spill). Each corrected spill has an associated SDF value, which is a volatility metric that varies in the range (0,1]. The loss function ℓ is defined as $\ell = (1 - \text{SDF})^2$, the squared error between the actual and target SDF. Gradients with respect to the PID gains $\partial\ell/\partial G$ are calculated directly from data, and are backpropagated through the simulator to update the neural network weights and minimize the loss (i.e. maximize the PID performance).

We refer to this approach as a *Hybrid* ML Simulator because only those functions which must be differentiable 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.

Training Procedure: PID Gains Tuning

An SDF of 1 corresponds to perfect regulation (zero volatility in the corrected spill). The SDF is thus used as the objective function with a target of 1 (perfect regulation). To optimize the PID gains, we minimize the loss function ℓ. The procedure for training the neural network to generate optimal PID gains is as follows: (i) the neural network is initialized randomly and generates a set of PID gains which are fed alongside a random noise signal into the PID simulator, (ii) the PID simulator uses these gains to produce a corrected spill c, (iii) the SDF value of this spill is compared to the target and the gradients of the loss function ℓ are calculated, (iii) gradients with respect to the PID gains $\partial\ell/\partial G$ are backpropagated [6] through the simulator and the neural network to minimize the error, (iv) the updated neural network generates a new set of PID gain values and the process repeats. Note that each iteration uses independent noise profiles and is minimizing the error over an *entire spill*. All differentiable functions are built with PyTorch [7], and we use the Adam Optimizer [8] with learning rate 0.01 for training.

Spill Segmentation

Since the ideal quad current ramp is non-linear in time, the extraction rate is sensitive to varied degrees in different parts of the spill. To characterize this variation, our system is capable of independently optimizing PID gains within arbitrary subdomains of the spill. Fig. 2(a) shows how the system evolves towards optimal PID gain values for 4 subdomains plotted along with the full spill's SDF value. Fig. 2(b) shows optimal PID gain values within four equally sized subdomains of the spill.

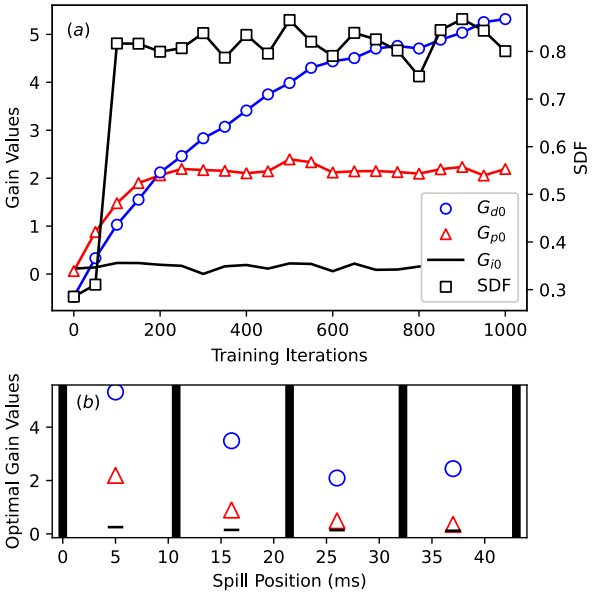

Figure 2: Top (a) Evolution of the PID gains in domain-0 (leftmost subdomain of bottom plot) over the full spill, as well as the SDF. Bottom (b) Four subdomains of the spill are segmented by vertical bars. Optimal gain values within each subdomain are shown on the vertical axis.

Regulation Performance

Our system is able to identify PID gains that realize high SDF values on representative input noise distributions (noise has an average SDF of 0.5). After 1000 training iterations, we can achieve corrected spills with median SDF values of 0.74, and by splitting the spill into four subdomains, median SDF values of 0.83, representing 48% and 66% reductions in the noise, respectively. If achieved in the reality this would well satisfy the experimental requirements of Mu2e. The regulation is especially effective on the high spikes, which are the main concerns for the detector performance.

REFERENCES

[1] K. Seiya *et al.*, *Accelerator real-time edge ai for distributed systems (reads) proposal*, 2021. arXiv: 2103.03928 [physics.acc-ph]. https://arxiv.org/abs/2103.03928

[2] K. 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 *Conference IPAC'21, Campinas, Brazil, paper MOPAB288, this conference*, 2021.

[3] V. Nagaslaev *et al.* (2010). "Third integer resonance slow extraction scheme for a mu2e experiment at fermilab," http://accelconf.web.cern.ch/HB2010/papers/mopd40.pdf

[4] (2015). "Mu2e Technical Design Report," https://arxiv.org/abs/1501.05241

[5] M. A. Ibrahim *et al.*, *Preliminary Design of Mu2e Spill Regulation System (SRS)*, 2019. https://accelconf.web.cern.ch/ibic2019/papers/mopp033.pdf

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," in *Neurocomputing: foundations of research*, Cambridge, MA, USA: MIT Press, Jan. 1988, pp. 696–699.

[7] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. http://arxiv.org/abs/1412.6980