FERMILAB-THESIS-2022-34

2022

Third-integer Resonant Extraction Regulation System for Mu2e

Aakaash Narayanan
aakaashn@gmail.com

ABSTRACT

THIRD-INTEGER RESONANT EXTRACTION REGULATION SYSTEM FOR MU2E

Aakaash Narayanan, Ph.D.
Department of Physics
Northern Illinois University, 2022
Michael James Syphers, Director

A third-integer resonant slow extraction system is being developed for Fermilab's Delivery Ring to deliver protons to the upcoming Mu2e experiment. The timescale of the extraction (or spill) duration is 43 milliseconds, which is extremely short and unprecedented. Additionally, the experiment's strict and challenging requirements on the quality of the spill at this time scale has led to the development of a new Spill Regulation System (SRS) design. The SRS primarily consists of three components - slow regulation, fast regulation, and harmonic content suppressor. Contributions to the first two components of the SRS, i.e., Slow Regulation and Fast Regulation subsystems, will be presented in which new adaptive learning algorithm schemes for the slow regulation of the spill – validated using particle tracking simulations – shall be described. In addition to these novel methods for the enhancement of the spill regulation system, results of employing Machine Learning in enhancing the performance of the resonant extraction are also presented. At the forefront of applying ML techniques to solve non-linear accelerator control problems, this work includes optimizing the PID gains as well as the replacement of the traditional PID controller using Recurrent Neural Networks and Gated Recurrent Unit (GRU) ML models to achieve efficiencies greater than a PID controller. Cutting-edge on-going Reinforcement Learning efforts, including actor-critic family

of learning algorithms, to regulate the spill rate will be reviewed, as well as present analytical calculations pertaining the transit time of particles in a third-integer resonant extraction. Detailed numerical investigations and validations of such calculations, the model of which could be exported and reliably used in future analytical modeling and design of any resonant extraction, are discussed.

NORTHERN ILLINOIS UNIVERSITY
DE KALB, ILLINOIS

DECEMBER 2022

THIRD-INTEGER RESONANT EXTRACTION REGULATION SYSTEM FOR MU2E

BY

AAKAASH NARAYANAN

A DISSERTATION SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE

DOCTOR OF PHILOSOPHY

DEPARTMENT OF PHYSICS

Dissertation Director:
Michael James Syphers

ACKNOWLEDGEMENTS

Thanks to:

- Michael J. Syphers, for introducing me to (and teaching) accelerator physics, for having the kindness to let me ask any dumb question, and for being a generous human being in making me feel welcome to a land that is half the world away from home,

- Vladimir P. Nagaslaev, for giving his patient support in guiding me throughout the project of this dissertation, for showing me the strict rigorousness of science, and for offering very many valuable advices on both physics and life,

- the READS collaboration, including Mattson Thieme, Kiyomi Seya, Kyle Hazelwood, Jing Jiang, and everyone else, for giving me the opportunity to work on applying Machine Learning to real world accelerator physics problems,

- Andrew Fielder, to have introduced me to Mike Syphers, (ergo, to accelerator physics), to have given unlimited number of car rides to Fermilab, and for being a very good friend,

- Prudhvi Nikhil Bhattiprolu, for being the greatest roommate in the world, for sharing a childlike enthusiasm for physics with me, and for many scintillating blackboard discussions over countless biryani sessions,

- Prudhvi Raj Varma Chintalapati, for being a wonderful friend whom I could always count on, for lending his help selflessly on numerous occasions the past 5 years,

- my peers at the Physics Department of NIU, including Christina Sarosiek, Osama Mohsen, William Baker, Ramanpreet Singh, Sebastian Szustkowski, Wei Hou Tan,

Brendan Leung, and all others to have given me a warm, colorful, and enriching grad school experience,

- Prof. V. Balakrishnan (IIT-M) for showing the profound interconnectedness of mathematics and physics through his online NPTEL lectures, for pointing out so many hidden 'Ahaa!'s in physics I never knew existed, and for convincing me (albeit virtually through his lectures) that physics is worth leaving home,

- VIT University, for the limitless freedom it gave me during my undergrad days and for having an amazing library with an excellent stock of physics textbooks,

- Nirmal Raj, for fueling my physics desire alive at a point when no one was there to talk physics to, for being an inspiration to take up physics as a serious career, and for being one with whom I share a maximum plethora of interests amongst all of my coterie,

- Ravi Shankar, my high school physics teacher, for showing me the joy of physics through his inimitable Socratic way of teaching, for showing it is *okay* to *not know*, and for inspiring very many students to take up physics after high school (including yours truly),

- Vimal Raj, for being a constant companion from our 8th grade through college through now, for being my best friend with a unmeasurably big heart, and for being my guardian angel at innumerable times of my life,

- Padma Dwivedi, for showing me innumerable tricks in **R**, for continually conquering both my heart and mind, for being with me through darkness and light, mundane and colorful, thick and thin, lending her unconditional support and care, and for keeping me in one piece throughout my PhD journey and for giving context to life,

- Narayanan Srinivasan, for giving me a wonderful childhood filled with nothing but joy, laughter, and happiness, to have dropped and picked me up from school for 14 years, for fostering in me the courage and temerity to explore anything, for showing me the world and all things in it, and for making me who I now am,
- Chithra Sampath, for ... everything.

DEDICATION

To Amma...

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

INTRODUCTION

The work done in this thesis is primarily pertaining the regulation system of third-integer resonant slow extraction to be done in the Delivery Ring machine in Fermi National Accelerator Laboratory (Fermilab) for the *Mu2e* experiment. In this introduction, we present a brief overview of the thesis, state a few important results, and give the reader an overall picture of the scope of the work done.

In Chapter 1, we present an introduction to the basic of physics of accelerators.

In Chapter 2, we derive a few important results regarding the particle dynamics of the non-linear third integer extraction process. We then provide an overarching introduction to the *Mu2e* experiment, the need for pulsed protons, and the need for a resonant extraction scheme. We present the details of the real-world extraction, along with the accelerator systems at Fermilab that involve providing beam to make the resonant extraction happen.

Mu2e is an upcoming experiment at Fermilab that intends to look for physics beyond the Standard Model in the decay of muons. The fundamental experimental technique is to capture and 'store' the muons inside Aluminum atom in the Coulomb field of nucleus and wait for the bound muons to decay.

A muon typically decays to an electron and two neutrinos almost 100 % of the time [1],

$$\mu \rightarrow e^{-} + \bar{\nu}_e + \nu_\mu \tag{1}$$

The *Mu2e* experiment intends to look for a neutrinoless *direct* conversion of muons to electrons, and is designed to make the most sensitive search ever made for the coherent conversion of muons to electrons in the Coulomb field of nucleus.

To create the muons for the experiment, a highly energetic proton beam with a kinetic energy of 8 GeV is to be bombarded on to a stationary production target. By the time the resulting muon beam reached the Al stopping target, it could be polluted with charged pions π^-. When these pions get captured inside Al atom, approximately 2.1% [2] of the pions could produce high energy photons through a process called 'radiative pion capture' (RPC).

$$\pi^- N \rightarrow \gamma N^* \tag{2}$$

And these photons, in turn, could interact with other photons and pair produce an electron-positron pair,

$$\pi^- N \rightarrow \gamma\gamma N^* \rightarrow e^+ e^- N^* \tag{3}$$

The experiment intends to detect electrons that are *directly* converted from muons, but the electrons produced from these RPC photons could potentially mimic a 'muon-converted-electron'. This background from RPC has been one of the biggest systematic effects in the previous searches for CLFV interactions. But *Mu2e*, however, would heavily suppress this background by using a pulsed muon beam (which in turn would be produced by a pulsed proton beam). This way, the detectors live-detection window could delayed in the search for the muon-converted-electrons until all of the RPC electrons escape away, and the measurement of the muon-converted-electron could thus be clean.

To produce a pulsed beam of protons, Fermilab intends to use a sophisticated beam physics technique called 'third-integer resonant slow extraction' to extract pulses of protons from the Delivery Ring to the and send it to the production target. This process involves

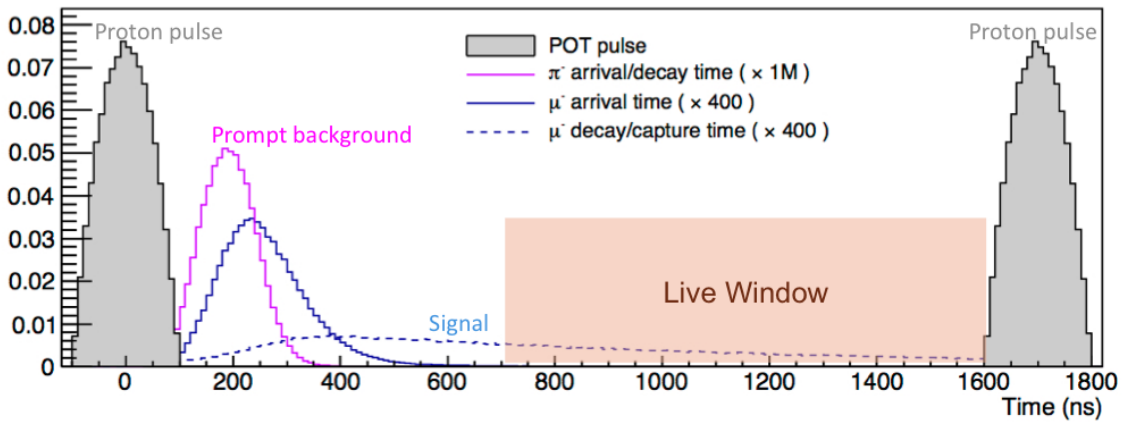

Figure 1: Pulsed time structure for the *Mu2e* experiment [3].

delicately driving the beam's horizontal tune very close to a fractional third integer (29/3 in our case), and the dynamics of the beam is thus made non-linear in a controlled fashion.

As the horizontal tune of the beam is driven closer to 29/3, a triangular stable region emerges in the normalized (X, X') phase space (defined in Chapter 1) of the particle (as shown in Fig. 2). The particles inside the stable region would continue to remain stable and not venture out the triangular separatrix. But the horizontal position of particles that are outside the stable region would start to increase with every turn. The goal is to drive the beam's horizontal tune close to resonance in a controlled way so as to *slice* away a uniform fraction of the circulating proton beam every turn (the slicing done by an electrostatic septa that gives an instantaneous horizontal kick to the slice) and transport it to the production target.

The third-integer resonant extraction is executed using six sextupoles and three fast-ramping quadrupoles. The sextupole introduces the third-integer resonance and the dedicated fast ramping quadrupoles control the rate at which the horizontal tune of the beam reaches the resonant tune of 29/3. Thus the ideal current ramp curve fed to the fast tune-shifting quadrupoles that would result in an ideal extraction is given in Figure 3. (In this

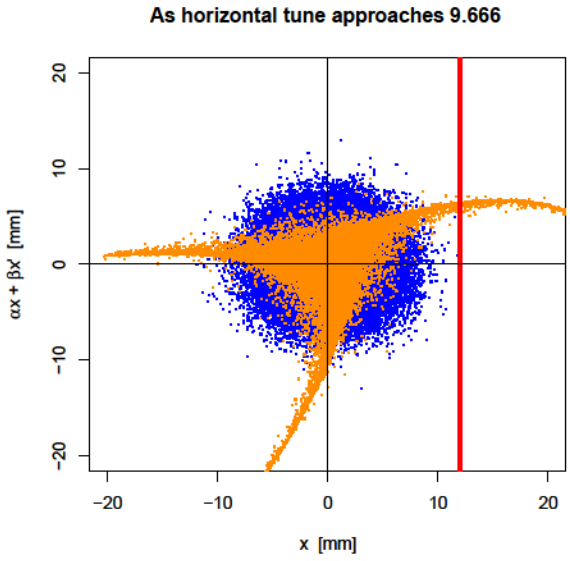

Figure 2: A triangular separatrix forms in the normalized phase space as the beam approaches a fraction of third integer.

dissertation, this power supply ramp curve for the tune-ramping quadrupole is both analytically calculated as well as obtained through adaptive learning algorithms developed by the author.)

However, in the real-life operation of the resonant extraction, we expect the spill to not be uniform but marred with noises in the extraction rate. A typical noised spill rate is expected to look like the one in Figure 4. We thus need a regulation system in place in order to curtail the noises in the spill rate. The spill quality is quantified by a factor called the spill duty factor (SDF), given by,

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

where σ_{spill} is the standard deviation of the variations in the extraction rate (the variations in the spill rate is normalized to an expectation value of 1). An ideal spill corresponds to

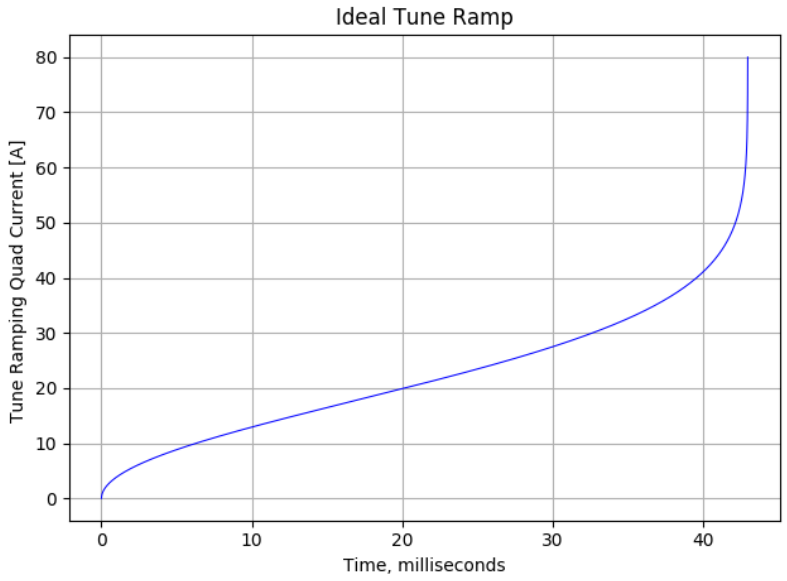

Figure 3: The ideal current ramp curve given to the fast tune-shifting quadrupoles in order to achieve ideal extraction.

zero standard deviation, giving an SDF value of 1 for a perfect spill. The goal of the Spill Regulation System (SRS) is to obtain an SDF value of 0.6 or higher.

The Spill Regulation System consists of three main components:

- Slow spill profile regulation
- Fast random ripple regulation
- Harmonic ripple content suppressor

In this work, we focus on the first two components of the SRS.

In Chapter 3, we present the investigations and results of the Slow Regulation algorithms which were validated using particle tracking. The main goal of the Slow Regulation is to curtail slow varying noise which could be caused by the slow drift of the accelerator components' parameters in the DR. We discuss in detail the workings of the particle tracking code developed by the author, which was used to both test the slow regulation schemes as well to perform transit time studies.

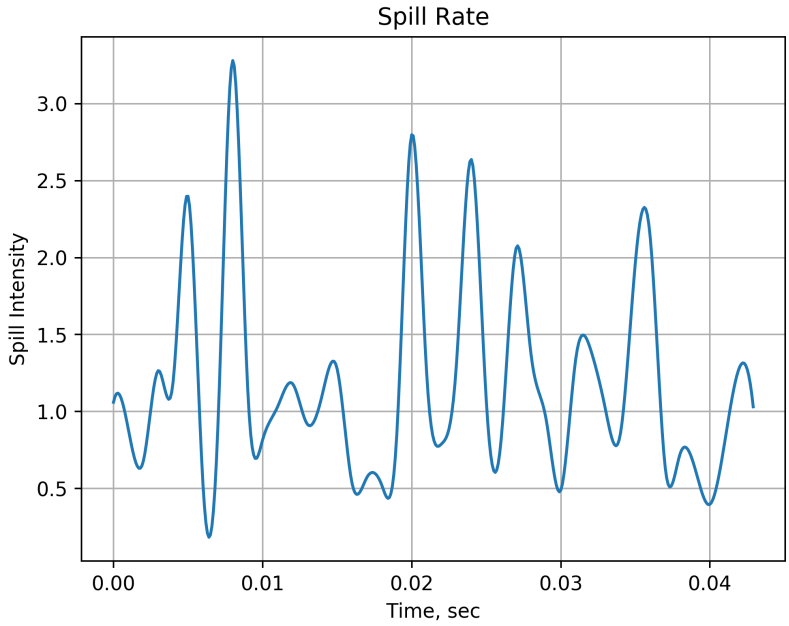

Figure 4: Possible variations in the spill rate within one spill duration. The spill rate here is normalized to 1.

The slow regulation algorithm would task itself in finding the ideal quad current ramp in the presence of slowly varying noises to make the average value of the spill as close to ideal as possible. We present an algorithm that successfully regulates the slow variation in noise, that was tested with about 132,000 particles that takes a just few hundred spills when starting with a non-ideal quad current ramp (as shown in Figure 5).

In Chapter 3, we go on to describe the Fast Regulation system, whose primary purpose is to mitigate instantaneous variations in the spill rate that could arise within one spill. We give an overview of the traditional Fast Regulation to be performed by PID controller, and we also present the effort behind the generation of VHDL code of the PID routine generated using MATLAB's HDL Coder toolbox.

In Chapter 4, we present the investigations of using machine learning (ML) to enhance the Fast Regulation system. We give a fairly basic introduction to machine learning concepts

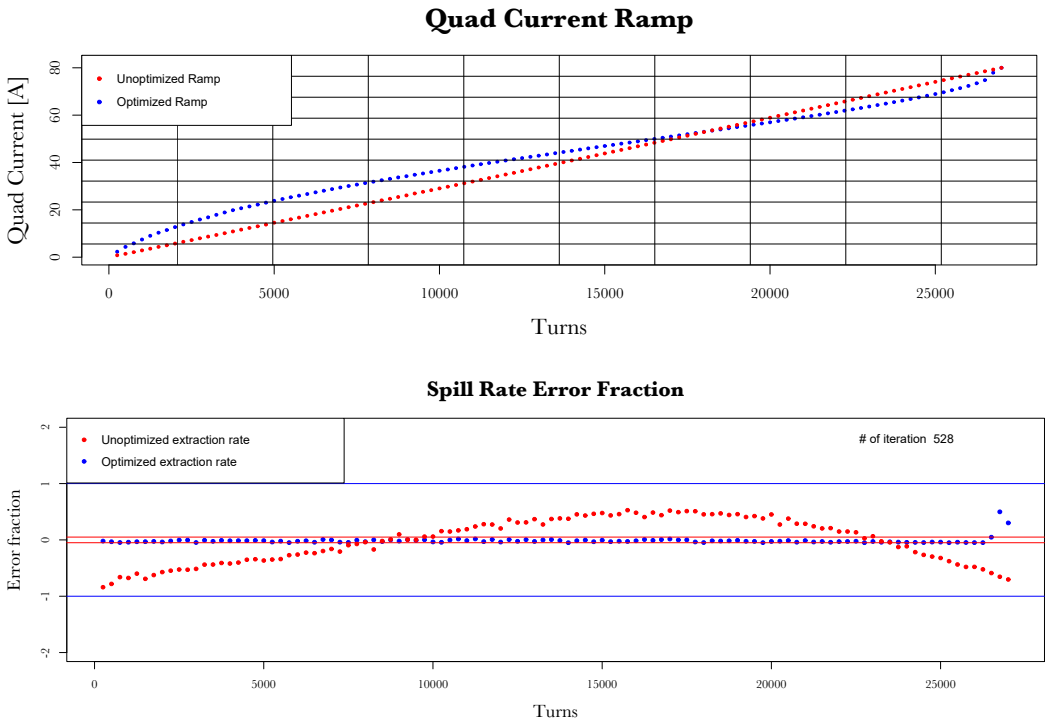

Figure 5: The result of Slow Regulation algorithm that successfully regulating the spill rate within $\pm 5\%$ accuracy. In both the graphs, the initial iteration and final iteration's plots are shown. We see that the algorithm was able to adaptively learn to make the extraction smooth in just over 500 spills.

that are pertinent to understanding the ML efforts in enhancing the Fast Regulation. We start the investigation by using tools from ML in optimizing the gains of the traditional PID controller (using an advanced optimizer), and we present the result of enhanced tuning of the PID controller. We see in Figure 6 that the SDF value is optimized with every training iteration. We also discuss dividing the spill into subdomains so that the PID can perform even more efficiently, as shown in Figure 7.

We next explore the possibility of replacing the PID controller entirely with a machine learning agent. We provide an introduction to a special type of neural network called the Recurrent Neural Network (RNN), and we also introduce advanced versions of RNN, such as Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) type of neural

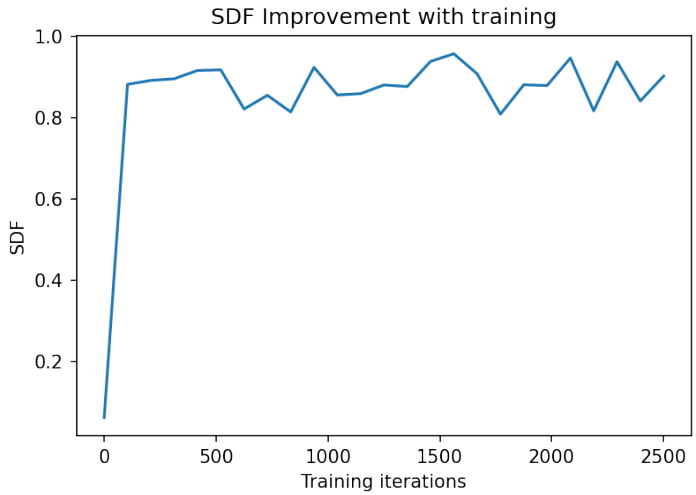

Figure 6: We see here that the SDF of the spill has significantly improved with the PID gain being optimized using ML techniques with every training iteration.

networks to over come the shortcomings of the vanilla RNN. We present the results of a GRU-based ML agent not only successfully matching the performance of a tuned PID controller but also outperforming the controller after sufficient training (shown in Figure 8.

We next talk about the on-going efforts (at the time of writing of this thesis in 2022) in exploring Reinforcement Learning (RL) techniques to outperform the traditional PID controller. We present an introduction to the general RL scheme, its difference from the Supervised Learning approach make our way to sophisticated RL technique called policy gradient methods, of which are the actor-critic class of algorithms.

In Chapter 5, we present some brief results of the transit time studies pertinent to the third-integer resonant extraction process. When a third-integer extraction is performed, the triangular separatrix is squeezed to almost an area of zero, and as the squeeze happens, particles that are outside the stable region would take a finite number of steps to reach the septum position in the horizontal plane in order to get extracted. We present here numerical validation of a few analytically derived transit time quantities, and we look for insights so

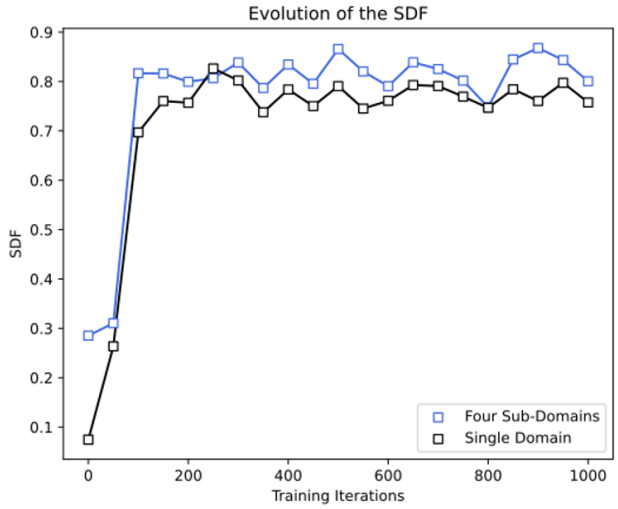

Figure 7: Evolution of SDF values when the spill is divided into subdomains over 1000 training iterations [38]. We see that the spill regulation is more efficient when the spill is divided into subdomains as against using just three gain values for the full spill.

one can better model and design a resonant extraction system using the results of transit time studies.

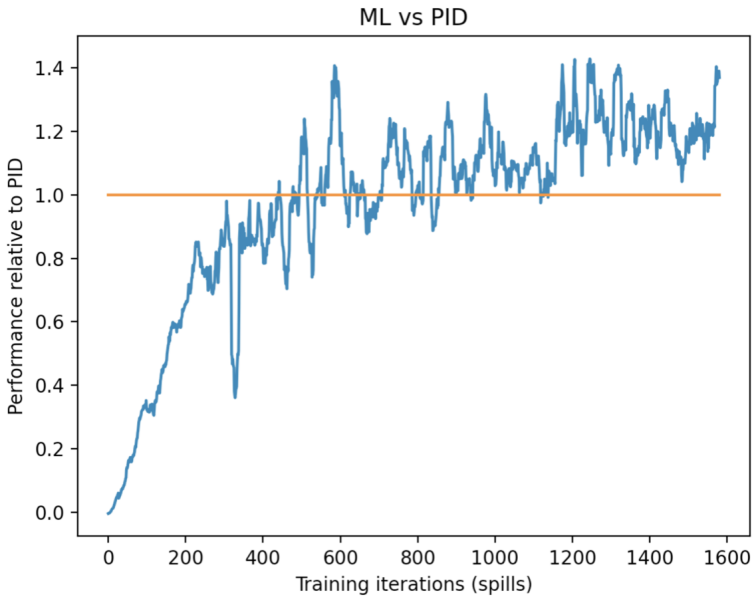

Figure 8: The relative performance of ML regulation vs PID regulation [39].

CHAPTER 1

PHYSICS OF PARTICLE ACCELERATORS

This chapter provides the necessary basic accelerator physics concepts needed for the physics of third-integer resonant extraction. The reader is advised that this chapter is not an exhaustive review of the basics of accelerator physics and only the concepts pertinent to resonant slow extraction, such as the transverse dynamics, is covered. For a more thorough introduction to the physics of particle accelerators, the reader can refer to [4] [1], [5], [6], [7], and [8].

1.1 Introduction

Accelerator physics is a branch of physics that primarily deals with the transportation and acceleration of particles (elementary or composite) from one physical location to another. Ever since its inception (and invention) in the early 20th century, particle accelerators have evolved and grown to be amazing tools resulting in applications varying from helping find the fundamentals laws of Nature to fighting cancer.

At the time of writing this thesis, there are more than 30,000 particle accelerators in the world [9], each varying in size, design, and purpose [2]. However, the principles mentioned in this chapter would govern the working of any generic particle accelerator.

A brief whirlwind tour of the history of particle accelerators can be found in [10].

[1] The introduction provided in this chapter closely follows *An Introduction to the Physics of High Energy Accelerators* by D. A. Edwards and M. J. Syphers.

[2] Surprisingly, only 1% of those 30,000 accelerator are used for fundamental research with particles' energies ranging above 1 GeV.

1.2 Coordinate System

Since one goal of accelerator physics is to transport particles, it is necessary to first choose a coordinates system that would help us analyze the dynamics of particles as they move from one place to another. Typically, particles traverse through beam lines that are practically bolted down to the ground. It would thus be convenient to have a coordinate system to describe the particles' positions and momenta relative to the ideal design trajectory going through all of the beam line elements. This ideal trajectory could be three-dimensional in the real-world, but we assume all the beam lines to lie on the same horizontal plane for the ease of introduction of accelerator physics concepts.

We could devise many new coordinate systems to suit our purpose of study, but one such coordinate system already exists [11] and is called the 'Frenet-Serret' coordinate system [3], named after the French mathematicians Joseph Alfred Serret and Jean Frederic Frenet [4] (both of whom had independently studied it).

The coordinate system consists of three independent coordinates with unit vectors: $(\hat{s}, \hat{x}, \hat{y})$, where the $|\hat{s}|$ denotes the distance along the design reference trajectory that a particle with ideal energy ought to follow. The $\hat{x}$ is in the plane coordinate, where by

$$\{\hat{s} \ . \ \hat{x}\} = 0$$

And $\hat{y}$ is the vertical coordinate, where by,

$$[\hat{x} \times \hat{y}] = \hat{s}$$

[3] This coordinate system is used in many situations that involve study of co-moving entities in a reference frame that itself is moving relative to the observer, such as study of disintegrating meteors, star/galaxy clusters in cosmology, particles in particle accelerators etc.,

[4] In fact, Frenet presented the crux of the transformation equations of this coordinate system in *his* doctoral thesis at Toulouse in 1847.

When we say, for example, the 'x-plane' or the 'y-plane', what we mean is the horizontal or the vertical plane of the beam profile, respectively.

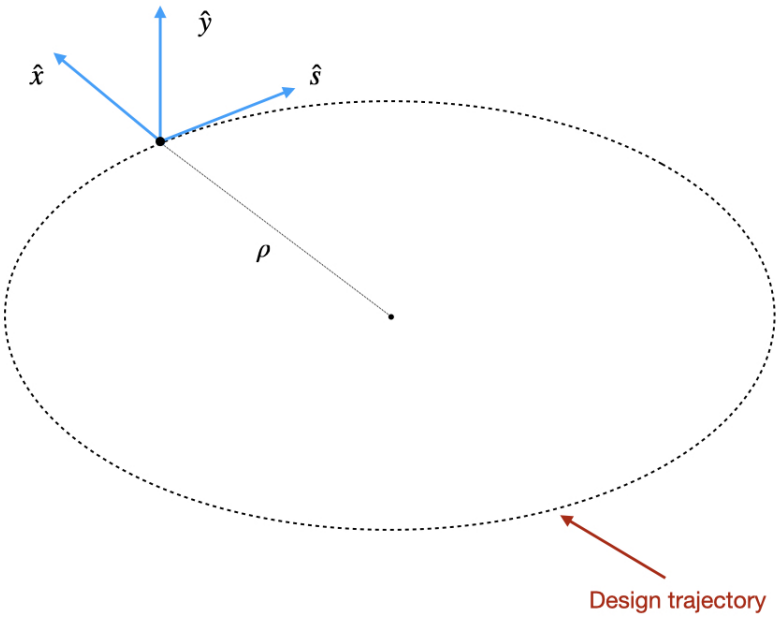

Figure 1.1: The $\hat{s}$ coordinate denotes the ideal design trajectory of the particle, the $\overrightarrow{x}$ denotes the radial coordinate, and the $\overrightarrow{y}$ denotes the vertical coordinate.

In this introductory chapter, unless otherwise stated, we are going to assume the kinetic energy of the particle to be a constant, i.e., the relativistic Lorentz factor γ to be a constant.

1.3 Equation of Motion

A charged particle responds to electric and magnetic fields in accordance to the Lorentz equation:

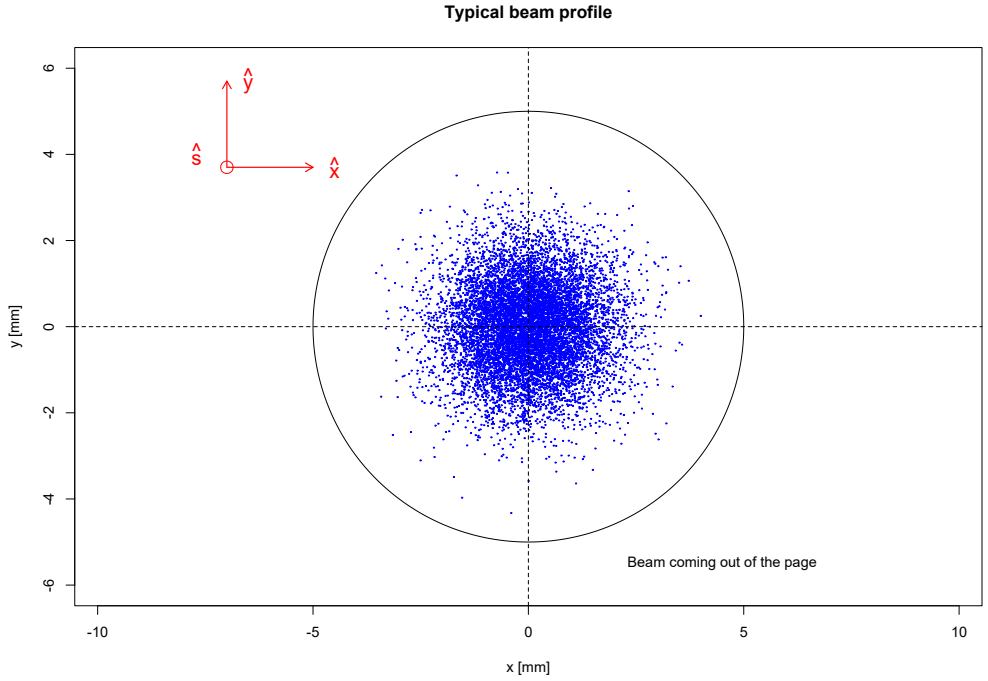

Figure 1.2: A typical distribution of particles (in the physical space) as seen inside a beam pipe, the $\hat{x}$ denotes the horizontal direction and the $\hat{y}$ denotes the vertical direction.

$$\overrightarrow{F} = \frac{\mathrm{d}\overrightarrow{p}}{\mathrm{d}t} = q(\overrightarrow{E} + \overrightarrow{v} \times \overrightarrow{B}) \tag{1.1}$$

For high energy accelerators, as is evident from the above equation, it is preferable to use $\overrightarrow{B}$ fields than $\overrightarrow{E}$ fields since the Lorentz force scales with v. For instance, to bend an 8 GeV kinetic energy proton about a radius of curvature of 80 meters [5], it would take 0.329 Tesla of $\overrightarrow{B}$-field but it would take about ≈100 MV/m of $\overrightarrow{E}$-field for an equivalent bend. The magnitude of electric field required is far more technologically challenging to achieve when compared to producing the magnetic field that would result in the same amount of Lorentz

[5]This would be the approximate scale of the Delivery Ring and the proton kinetic energy at Fermilab.

force on the particle. Thus, unless otherwise stated, the default type of field assumed in this work would be magnetic and not electric.

Our goal here is to devise a formalism that would help us write down the evolution of the particles' position and momenta as they travel through the beam line experiencing different kinds of fields.

In the Frenet-Srenet coordinate system, the position of a particle could be written as

$$\overrightarrow{R} = (x + \rho)\hat{x} + y\hat{y} \tag{1.2}$$

where ρ is the design radius of curvature.

Assuming that the particle's kinetic energy is constant, the LHS of equation (1.1) could be written as,

$$\begin{aligned}
\frac{\mathrm{d}\overrightarrow{p}}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\gamma m\overrightarrow{v} \\
&= \gamma m\frac{\mathrm{d}\overrightarrow{v}}{\mathrm{d}t} \\
&= \gamma m\frac{\mathrm{d}\dot{\overrightarrow{R}}}{\mathrm{d}t} \\
&= \gamma m\ddot{\overrightarrow{R}}
\end{aligned}$$

$$\implies q(\overrightarrow{E} + \overrightarrow{v} \times \overrightarrow{B}) = \gamma m\ddot{\overrightarrow{R}}$$

$$\boxed{\ddot{\overrightarrow{R}} = \frac{q(\overrightarrow{E} + \overrightarrow{v} \times \overrightarrow{B})}{\gamma m}}$$

Next we must evaluate $\ddot{R}$,

$$\begin{aligned}\vec{R} &= \underbrace{(x+\rho)}_{\text{say } = r}\hat{x} + y\hat{y} \\ \dot{\vec{R}} &= \frac{\mathrm{d}}{\mathrm{d}t} r\hat{x} + \frac{\mathrm{d}}{\mathrm{d}t} y\hat{y} \\ &= \dot{r}\hat{x} + r\dot{\hat{x}} + \dot{y}\hat{y}\end{aligned}$$

From the Fig.(1.3), we see that

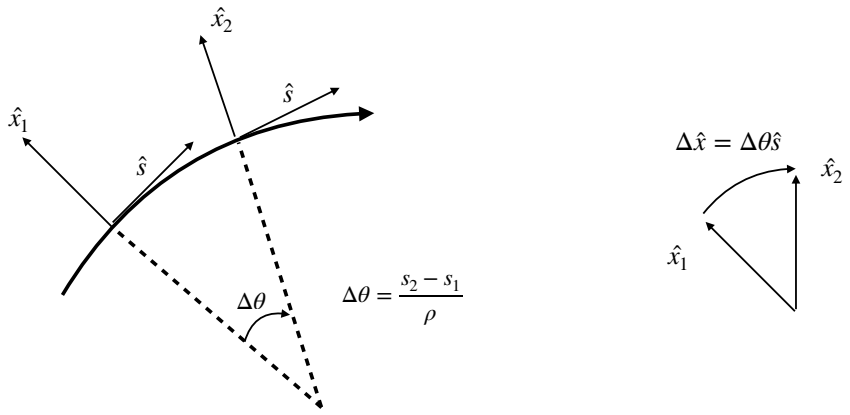

Figure 1.3: Time rate change of unit vector $\hat{x}$

$$\dot{\hat{x}} = \dot{\theta}\hat{s} \tag{1.3}$$

Thus,

$$\dot{\overrightarrow{R}} = \dot{r}\hat{x} + r\dot{\theta}\hat{s} + \dot{y}\hat{y} \tag{1.4}$$

Differentiating again,

$$\begin{aligned}\ddot{\overrightarrow{R}} &= \ddot{r}\hat{x} + \dot{r}\dot{\hat{x}} + \dot{r}\dot{\theta}\hat{s} + r\ddot{\theta}\hat{s} + r\dot{\theta}\dot{\hat{s}} + \ddot{y}\hat{y} \\ &= \ddot{r}\hat{x} + \dot{r}\dot{\theta}\hat{s} + \dot{r}\dot{\theta}\hat{s} + r\ddot{\theta}\hat{s} + r\dot{\theta}\dot{\hat{s}} + \ddot{y}\hat{y} \\ &= \ddot{r}\hat{x} + 2\dot{r}\dot{\theta}\hat{s} + r\ddot{\theta}\hat{s} + r\dot{\theta}\dot{\hat{s}} + \ddot{y}\hat{y}\end{aligned} \tag{1.5}$$

From Fig. 1.3, we equivalently see also that,

$$\dot{\hat{s}} = -\dot{\theta}\hat{x} \tag{1.6}$$

Plugging (1.6) into (1.5),

$$\begin{aligned}\ddot{\overrightarrow{R}} &= \ddot{r}\hat{x} + 2\dot{r}\dot{\theta}\hat{s} + r\ddot{\theta}\hat{s} + r\dot{\theta}(-\dot{\theta}\hat{x}) + \ddot{y}\hat{y} \\ &= \ddot{r}\hat{x} + 2\dot{r}\dot{\theta}\hat{s} + r\ddot{\theta}\hat{s} - r\dot{\theta}^2\hat{x} + \ddot{y}\hat{y} \\ &= (\ddot{r} - r\dot{\theta}^2)\hat{x} + (2\dot{r}\dot{\theta} + r\ddot{\theta})\hat{s} + \ddot{y}\hat{y}\end{aligned} \tag{1.7}$$

The magnetic field experienced by the beam, assuming no longitudinal fields, could be written as,

$$\overrightarrow{B} = B_x\hat{x} + B_y\hat{y}$$

$$\overrightarrow{B} = \left(B_x(0,0) + \frac{\partial B_x}{\partial x}x + \frac{\partial B_x}{\partial y}y \right)\hat{x} + \left(B_y(0,0) + \frac{\partial B_y}{\partial x}x + \frac{\partial B_y}{\partial y}y \right)\hat{y}$$

Now let us compute the RHS of equation (1.1),

$$\vec{v} \times \vec{B} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{s} \\ v_x & v_y & v_s \\ B_x & B_y & 0 \end{vmatrix}$$

$$\boxed{\vec{v} \times \vec{B} = -v_s B_y \hat{x} + v_s B_x \hat{y} + (v_x B_y - B_y v_x)\hat{s}} \tag{1.8}$$

Equating (1.8) with (1.7), we have

$$\ddot{R} = \frac{q(\overrightarrow{E} + \overrightarrow{v} \times \overrightarrow{B})}{\gamma m}$$

$$(\ddot{r} - r\dot{\theta}^2)\hat{x} + (2\dot{r}\dot{\theta} + r\ddot{\theta})\hat{s} + \ddot{y}\hat{y} = \frac{q\Big(- v_s B_y \hat{x} + v_s B_x \hat{y} + (v_x B_y - B_y v_x)\hat{s} \Big)}{\gamma m}$$

Equating the x-components, we have,

$$\ddot{r} - r\dot{\theta}^2 = \frac{q(-v_s B_y)}{\gamma m}$$

Multiplying and dividing by v_s on the RHS,

$$\ddot{r} - r\dot{\theta}^2 = \frac{q(-v_s^2 B_y)}{\underbrace{\gamma m v_s}_{p_s}}$$

Since the particle's momentum is overwhelmingly in the longitudinal $\vec{s}$ direction, we could assume $p_{tot} \approx p_s = p$ (say).

$$\ddot{r} - r\dot{\theta}^2 = \frac{-qv_s^2 B_y}{p} \tag{1.9}$$

Since the beam line is fixed, we are interested in the evolution of particle's local position and momenta not just with respect to time but with respect to *where* the particle is in the beam line, i.e., w.r.t. s.

$$\frac{\mathrm{d}}{\mathrm{d}t} = \frac{\mathrm{d}s}{\mathrm{d}t}\frac{\mathrm{d}}{\mathrm{d}s} \tag{1.10}$$

But from Fig. 1.3, we see that,

$$\mathrm{d}s = \rho\,\mathrm{d}\theta = v_s\,\mathrm{d}t\,\frac{\rho}{r} \tag{1.11}$$

Assuming the particle's kinetic energy does not change as it goes through a beam element, i.e., $\frac{\mathrm{d}^2 s}{\mathrm{d}t^2} = 0$, we have,

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2} = \left(\frac{\mathrm{d}s}{\mathrm{d}t}\right)^2 \frac{\mathrm{d}^2}{\mathrm{d}s^2} = \left(v_s \frac{\rho}{r}\right)^2 \frac{\mathrm{d}^2}{\mathrm{d}s^2} \tag{1.12}$$

Thus (1.9) would look,

$$\begin{aligned}
\frac{\mathrm{d}^2 r}{\mathrm{d}t^2} - r\left(\frac{\mathrm{d}\theta}{\mathrm{d}t}\right)^2 &= \frac{-qv_s^2 B_y}{p} \\
\left(v_s\frac{\rho}{r}\right)^2\frac{\mathrm{d}^2 r}{\mathrm{d}s^2} - r\left(\frac{\mathrm{d}s}{\mathrm{d}t}\frac{\mathrm{d}\theta}{\mathrm{d}s}\right)^2 &= \frac{-qv_s^2 B_y}{p} \\
\left(v_s\frac{\rho}{r}\right)^2\frac{\mathrm{d}^2 r}{\mathrm{d}s^2} - r\left(\frac{v_s\rho}{r}\frac{1}{\rho}\right)^2 &= \frac{-qv_s^2 B_y}{p} \\
\left(v_s\frac{\rho}{r}\right)^2\frac{\mathrm{d}^2 r}{\mathrm{d}s^2} - \frac{v_s^2}{r} &= \frac{-qv_s^2 B_y}{p} \\
\left(v_s\frac{\rho}{r}\right)^2\frac{\mathrm{d}^2 r}{\mathrm{d}s^2} - \frac{v_s^2}{r} &= \frac{-qv_s^2 B_y}{p}
\end{aligned}$$

Plugging in $r = \rho + x$,

$$\begin{aligned}
\left(v_s\frac{\rho}{(\rho+x)}\right)^2\frac{\mathrm{d}^2(\rho+x)}{\mathrm{d}s^2} - \frac{v_s^2}{(\rho+x)} &= \frac{-qv_s^2 B_y}{p} \\
\cancel{v_s^2}\frac{\rho^2}{(\rho+x)^2}\frac{\mathrm{d}^2 x}{\mathrm{d}s^2} - \frac{\cancel{v_s^2}}{(\rho+x)} &= \frac{-q\cancel{v_s^2}B_y}{p} \\
\frac{\mathrm{d}^2 x}{\mathrm{d}s^2} - \frac{(\rho+x)}{\rho^2} &= \frac{(\rho+x)^2}{\rho^2}\frac{-qB_y}{p} \\
\frac{\mathrm{d}^2 x}{\mathrm{d}s^2} - \frac{(\rho+x)}{\rho^2} &= -\underbrace{\frac{B_y}{p/q}}_{(B\rho)}\frac{(\rho+x)^2}{\rho^2}
\end{aligned}$$

Thus, the evolution of the x-position of the particle as it travels through the beam line is given by,

$$\boxed{\frac{\mathrm{d}^2 x}{\mathrm{d}s^2} - \frac{(\rho+x)}{\rho^2} = -\frac{B_y}{(B\rho)}\left(1+\frac{x}{\rho}\right)^2} \tag{1.13}$$

Similarly, equating the $\vec{y}$ components, we have,

$$\left(v_s\frac{\rho}{r}\right)^2\frac{\mathrm{d}^2y}{\mathrm{d}s^2} = \frac{qv_sB_x}{\gamma m}$$
$$\frac{\mathrm{d}^2y}{\mathrm{d}s^2} = qB_x\frac{r^2}{\underbrace{\gamma m v_s}_{\approx p}\rho^2}$$
$$\frac{\mathrm{d}^2y}{\mathrm{d}s^2} = B_x\frac{(x+\rho)^2}{\underbrace{(p/q)}_{=(B\rho)}\rho^2}$$
$$\frac{\mathrm{d}^2y}{\mathrm{d}s^2} = B_x\frac{(x+\rho)^2}{(B\rho)\rho^2}$$

The evolution of the vertical y-position of the particle as it travels through the beam line is given by,

$$\boxed{\frac{\mathrm{d}^2y}{\mathrm{d}s^2} = \frac{B_x}{(B\rho)}\left(1+\frac{x}{\rho}\right)^2} \tag{1.14}$$

1.3.1 Piece-wise solution

Next left for us is to solve the above equations in x and y for different types of fields. We note that equations (1.13) and (4.14) are of the form

$$\frac{\mathrm{d}^2x}{\mathrm{d}s^2} + xK(s) = 0$$

This typically can be assumed to the simple harmonic motion of a mass attached to a spring, except that now the spring constant is itself a function of position.

In real-life accelerators, the beam line elements are fairly 'thin' [6] and the K-value *within* each of these elements can reasonably be assumed to be a constant. (However, the K-value from one element to another could most certainly vary.)

If we limit our interest in finding how the particles' coordinates and momenta change upon entering and exiting a particular element, we could then 'integrate' the accumulative effects of the individual pieces and thus get the full evolution of particles' coordinates for any arbitrary distance through the beam line.

If $K = 0$, this would give us,

$$\begin{aligned} \frac{\mathrm{d}^2 x}{\mathrm{d}s^2} + 0 &= 0 \\ \implies \frac{\mathrm{d}x}{\mathrm{d}s} &= \text{constant} = x' \\ x(s) &= \int_0^L x' \,\mathrm{d}s = Lx' \end{aligned}$$

If we write the solution in terms of matrix,

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{\text{out}} = \begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix}_{\text{in}} \tag{1.15}$$

If $K > 0$, that would imply a simple harmonic solution,

$$x(s) = A\cos\left(\sqrt{K}s\right) + B\sin\left(\sqrt{K}s\right)$$

[6] A beam element is considered 'thin' when the proportion of the length of the element to the geometric length of interest it induces in the particle's dynamics (such as the focal length for a quadrupole or the bending radius curvature for a dipole) is very small.

To verify, plugging in and solving,

$$\begin{aligned}
\frac{\mathrm{d}x(s)}{\mathrm{d}s} &= -A\sqrt{K}\sin\left(\sqrt{K}s\right) + B\sqrt{K}\cos\left(\sqrt{K}s\right) \\
\frac{\mathrm{d}^2x(s)}{\mathrm{d}s^2} &= -AK\cos\left(\sqrt{K}s\right) + BK\sin\left(\sqrt{K}s\right) \\
&= -K\Big[A\cos\left(\sqrt{K}s\right) + B\sin\left(\sqrt{K}s\right)\Big] \\
&= -Kx(s)
\end{aligned}$$

To get the coefficients A and B, we input the boundary conditions the following way.

If the initial position of the particle before it enters the beam element $x(s=0)$ is x_i , then,

$$\begin{aligned}
x_i &= A\cos\left(\sqrt{K}\times 0\right) + B\sin\left(\sqrt{K}\times 0\right) \\
\implies A &= x_i
\end{aligned} \tag{1.16}$$

To get B, let us differentiate the above and set $s=0$,

$$\begin{aligned}
\frac{\mathrm{d}x(s=0)}{\mathrm{d}s} &= -x_i\sqrt{K}\sin\left(\sqrt{K}\times 0\right) + B\sqrt{K}\cos\left(\sqrt{K}\times 0\right) \\
\frac{\mathrm{d}x(s=0)}{\mathrm{d}s} &= x_i' = \sqrt{K}B \\
\implies B &= \frac{x_i'}{\sqrt{K}}
\end{aligned}$$

Plugging in the constants,

$$\begin{aligned}
x(s) &= x_i\cos\left(\sqrt{K}s\right) + \frac{x_i'}{\sqrt{K}}\sin\left(\sqrt{K}s\right) \\
\frac{\mathrm{d}x(s)}{\mathrm{d}s} &= -x_i\sqrt{K}\sin\left(\sqrt{K}s\right) + x_i'\cos\left(\sqrt{K}s\right)
\end{aligned}$$

Writing this solution in terms of matrices,

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{\text{out}} = \begin{pmatrix} \cos\left(\sqrt{K}l\right) & \frac{1}{\sqrt{K}}\sin\left(\sqrt{K}l\right) \\ -\sqrt{K}\sin\left(\sqrt{K}l\right) & \cos\left(\sqrt{K}l\right) \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix}_{\text{in}}$$

Similarly, if $K < 0$, we would get divergent hyperbolic cosine and sine function,

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{\text{out}} = \begin{pmatrix} \cosh\left(\sqrt{K}l\right) & \frac{1}{\sqrt{K}}\sinh\left(\sqrt{K}l\right) \\ -\sqrt{K}\sinh\left(\sqrt{K}l\right) & \cosh\left(\sqrt{K}l\right) \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix}_{\text{in}}$$

In the 'thin lens' approximation, whereby the focal length is much larger compared to the length of the beam element, i.e., if $l \to 0$ whereas Kl is finite, the matrix for positive K reduces to,

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{\text{out}} = \begin{pmatrix} 1 & 0 \\ -1/f & 1 \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix}_{\text{in}}$$

where $1/f$ is K. And, for negative K values, we would have,

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{\text{out}} = \begin{pmatrix} 1 & 0 \\ +1/f & 1 \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix}_{\text{in}}$$

Once we know the constant K_i for a particular i-th beam line element, we can get the evolution of particle's x and x' as it goes through that element. This could be extended to arbitrary number of such beam elements the particles traverse through whose K-value is

known. To get the net accumulated evolution of (x, x'), all one needs to do is multiply these 'propagation' matrices in the same order as the respective accelerator element is placed,

$$M_{\text{net}} = M_N \ . \ M_{N-1} \ . \ ... \ . \ M_2 \ . \ M_1$$

1.3.2 Analytical Solution

The piece-wise solution, though captures the essence of solving the equations, is not quite mathematically rigorous. The class of equation of the form $dx^2/ds^2 + xK(s) = 0$ is known in mathematical literature as the Hill's equation and was extensively studied through the second half of 18th century. Borrowing results from the mathematical literature, we just state here that the general solution of Hill's equation [7] is of the following general form,

$$x(s) = Aw(s)\cos\left[\phi(s) + \delta\right] \tag{1.17}$$

where A and δ are two constants of integration. For K being a positive constant, we infer the solution is of the form,

$$x(s) = A\cos\left[\sqrt{K}s + \delta\right]$$

where, again, A and δ are constants of integration.

If we have a beam line consisting of elements whose K values are different, and if the beam line is periodic, then A may no longer remain a constant and $\phi(s)$ too need not be a linear function of s.

[7] This result, in particular, was derived by Gaston Floquet and is known by Floquet theorem.

Assuming the solution of Hill's equation to be of the form (1.17),

$$
\begin{aligned}
x(s) &= Aw(s)\cos\left[\phi(s)+\delta\right] \qquad (1.18)\\
x'(s) &= Aw'(s)\cos\left[\phi(s)+\delta\right] - Aw(s)\sin\left[\phi(s)+\delta\right].\phi'(s)\\
x''(s) &= Aw''(s)\cos\left[\phi(s)+\delta\right] - Aw'(s)\phi'(s)\sin\left[\phi(s)+\delta\right]\\
&\quad - Aw'(s)\phi'(s)\sin\left[\phi(s)+\delta\right] - Aw(s)\phi''(s)\sin\left[\phi(s)+\delta\right] - Aw(s)\phi'^2(s)\cos\left[\phi(s)+\delta\right]\\
x''(s) &= A\cos\left[\phi(s)+\delta\right]\Big(w''(s) - w(s)\phi'^2(s)\Big) - A\sin\left[\phi(s)+\delta\right]\Big(2w'(s)\phi'(s) + w(s)\phi''(s)\Big)
\end{aligned}
$$

$$
\boxed{
\begin{aligned}
x''(s) + Kx(s) = 0 = \;& A\cos\left[\phi(s)+\delta\right]\Big(w''(s) - w(s)\phi'^2(s) + Kw\Big)\\
& - A\sin\left[\phi(s)+\delta\right]\Big(2w'(s)\phi'(s) + w(s)\phi''(s)\Big)
\end{aligned}
} \qquad (1.19)
$$

If the above equation has to hold independent of the constants of motion A and δ, then both the sine as well the cosine terms' coefficients must individually vanish.

$$
\begin{aligned}
w'' - w\phi'^2 + Kw = 0 \quad &\text{(cosine coefficients)}\\
2w'\phi' + w\phi'' = 0 \quad &\text{(sine coefficients)}
\end{aligned}
$$

Multiplying the sine coefficients by w,

$$\begin{aligned} 2ww'\phi' + w^2\phi'' &= 0 \\ \frac{\mathrm{d}}{\mathrm{d}s}w^2\phi' &= 2ww'\phi' + w^2\phi'' \\ \implies \frac{\mathrm{d}}{\mathrm{d}s}w^2\phi' &= 0 \\ w^2\phi' &= \text{constant} = k(\text{say}) \\ \boxed{\phi'(s) = \frac{k}{w^2(s)}} & \end{aligned} \tag{1.20}$$

(As a sanity check, we see from the above result that if $w(s)$ is a constant, then we retrieve the simple harmonic motion result where ϕ linearly grows with s: $\phi = \int \phi' \,\mathrm{d}s = \int k_0 \,\mathrm{d}s = k_0 s + s_0$.)

Using the relation in (1.20), we can solve for the cosine coefficients,

$$\begin{aligned} w'' - w\phi'^2 + Kw &= 0 \\ w'' - w\frac{k^2}{w^4} + Kw &= 0 \\ w'' = w\frac{k^2}{w^4} - Kw & \\ w'' = \frac{k^2}{w^3} - Kw & \\ w^3(w'' + Kw) &= k^2 \end{aligned} \tag{1.21}$$

In our analytical treatment so far, we have assumed K to not be a constant but a function of s. However, as mentioned earlier, it is reasonable to assume K to be a constant within an accelerator element. So the above differential equation is essentially for a *collection* of accelerator elements placed in the trajectory integrated with s.

In real-life, the accelerator elements are, in general [8], periodic in nature, for it could either contain repeating units of finite length C or perhaps an accelerator (with or without repeating elements) that closes in on itself with circumference C. So it would serve us well if we look for *periodic* solution to the above differential equation.

Rewriting equation (1.18) by subsuming the constants of motion (A and δ) into amplitude constants (A_1 and A_2), we have,

$$
\begin{aligned}
x(s) &= w(s)\Big[A_1 \cos\left[\phi(s)\right] + A_2 \sin\left[\phi(s)\right]\Big] \qquad (1.22)\\
x'(s) &= w'\Big[A_1 \cos\phi + A_2 \sin\phi\Big] + w\Big[- A_1\phi' \sin\phi + A_2\phi' \cos\phi\Big]\\
&= w'A_1 \cos\phi + w'A_2 \sin\phi + -wA_1\phi' \sin\phi + wA_2\phi' \cos\phi\\
&= [w'A_1 + wA_2\phi'] \cos\phi + [w'A_2 - wA_1\phi'] \sin\phi
\end{aligned}
$$

Substituting equation (1.20) and eliminating ϕ',

$$
\begin{aligned}
\phi' &= \frac{k}{w^2} \qquad (1.23)\\
x'(s) &= \left[w'A_1 + wA_2\frac{k}{w^2}\right] \cos\phi + \left[w'A_2 - wA_1\frac{k}{w^2}\right] \sin\phi\\
x'(s) &= \left[w'A_1 + \frac{kA_2}{w}\right] \cos\phi + \left[w'A_2 - \frac{kA_1}{w}\right] \sin\phi \qquad (1.24)
\end{aligned}
$$

To get the constants A_1 and A_2, we input the boundary condition that at the initial position $s = s_0$, we take ϕ to be zero.

$$
\begin{aligned}
x_0 &= w(s)\Big[A_1 \cos\left[0\right] + A_2 \sin\left[0\right]\Big]\\
x_0 &= wA_1\\
\implies &\boxed{A_1 = \frac{x_0}{w}} \qquad (1.25)
\end{aligned}
$$

[8] Transport lines need not necessarily be periodic.

And,

$$\begin{aligned} x_0' &= [w'A_1 + \frac{kA_2}{w}]\cos 0 + [w'A_2 - \frac{kA_1}{w}]\sin 0 \\ &= w'A_1 + \frac{kA_2}{w} \\ &= \frac{w'x_0}{w} + \frac{kA_2}{w} \\ &= \frac{w'x_0 + kA_2}{w} \\ x_0'w &= w'x_0 + kA_2 \\ \implies &\boxed{A_2 = \frac{x_0'w - x_0w'}{k}} \end{aligned} \tag{1.26}$$

Plugging in A_1 and A_2 into (1.22),

$$\begin{aligned} x(s) &= w\Big[\frac{x_0}{w}\cos[\phi] + \frac{x_0'w - x_0w'}{k}\sin[\phi]\Big] \\ &= x_0\cos\phi + \frac{x_0'w^2 - x_0ww'}{k}\sin\phi \\ &= x_0\cos\phi + \frac{x_0'w^2}{k}\sin\phi - \frac{x_0ww'}{k}\sin\phi \\ &\boxed{x(s) = x_0\left[\cos\phi - \frac{ww'}{k}\sin\phi\right] + x_0'\left[\frac{w^2}{k}\sin\phi\right]} \end{aligned} \tag{1.27}$$

Similarly, plugging A_1 and A_2 values into (1.24),

$$x'(s) = \Big[w'\frac{x_0}{w} + \frac{kx_0'w - x_0w'k}{wk}\Big]\cos\phi + \Big[\frac{x_0'ww' - x_0w'^2}{k} - \frac{kx_0}{w^2}\Big]\sin\phi$$

$$
\begin{aligned}
&= \frac{x_0 w' \cos\phi}{w} + \frac{k x_0' w \cos\phi}{wk} - \frac{x_0 w' k \cos\phi}{wk} + \frac{x_0' w w' \sin\phi}{k} - \frac{x_0 w'^2 \sin\phi}{k} - \frac{k x_0 \sin\phi}{w^2} \\
&= \frac{x_0 w' \cos\phi}{w} - \frac{x_0 w' k \cos\phi}{wk} - \frac{x_0 w'^2 \sin\phi}{k} - \frac{k x_0 \sin\phi}{w^2} + \frac{k x_0' w \cos\phi}{wk} + \frac{x_0' w w' \sin\phi}{k} \\
&= x_0 \left[\cancel{\frac{w' \cos\phi}{w}} \cancel{- \frac{w' \cos\phi}{w}} - \frac{w'^2 \sin\phi}{k} - \frac{k \sin\phi}{w^2} \right] + x_0' \left[\frac{k w \cos\phi}{wk} + \frac{w w' \sin\phi}{k} \right]' \\
&= x_0 \left[- \left[\frac{w'^2}{k} - \frac{k}{w^2} \right] \sin\phi \right] + x_0' \left[\cancel{\frac{k w}{w k}} \cos\phi + \frac{w w'}{k} \sin\phi \right]
\end{aligned}
$$

$$
\boxed{x'(s) = x_0 \left[- \frac{w^2 w'^2 - k^2}{k w^2} \sin\phi \right] + x_0' \left[\cos\phi + \frac{w w'}{k} \sin\phi \right]} \tag{1.28}
$$

If we assume the periodicity of the beam elements to be repeated after distance C, and that the particle starts at s_0 in the beam line, the phase of the particle is going to change by some amount $\Delta\phi$ and the (x, x') would evolve according to the above equations (1.27) and (1.28).

If we combine them both and write them in terms of a single matrix equation, we have:

$$
\begin{pmatrix} x \\ x' \end{pmatrix}_{s_0+C} = \begin{pmatrix} \cos\Delta\phi - \frac{w w'}{k} \sin\Delta\phi & \frac{w^2}{k} \sin\Delta\phi \\ -\frac{w^2 w'^2 - k^2}{k w^2} \sin\Delta\phi & \cos\Delta\phi + \frac{w w'}{k} \sin\Delta\phi \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix}_{s_0} \tag{1.29}
$$

1.4 Courant-Snyder Parameters

Having gone through solving the equations of motion and also through (a somewhat) formal treatment of the Hill's equation, next left for us is to interpret the above results.

By inspecting the 'simple harmonic motion'-like matrix, we infer that as the particle goes through the periodic distance C, its oscillation phase advances by an amount,

$$\phi(s_0 \to s_0 + C) = \Delta\phi = \int_{s_0}^{s_0+C} \frac{k}{w^2(s)}\,\mathrm{d}s \tag{1.30}$$

It does not matter from which particular s_0 you start from since $w(s)$ is periodic in C. This would be the phase advance of the particle.

We also realize that both $w(s)$ (as well as its derivatives) simply scale with k. Since the quantities of interest to us are $w(s)$ and its derivative, it is useful to define new variables,

$$\beta(s) \equiv \frac{w^2(s)}{k} \tag{1.31}$$

$$\alpha(s) \equiv -\frac{1}{2}\frac{\mathrm{d}\beta(s)}{\mathrm{d}s} = -\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}s}\frac{w^2(s)}{k} = -\frac{ww'}{k} \tag{1.32}$$

$$\gamma(s) \equiv \frac{1+\alpha(s)^2}{\beta(s)} \tag{1.33}$$

These new variables α, β, and γ are referred to as the 'Courant-Snyder' parameters named after Ernest Courant and Hartland Sweet Snyder for their work in their groundbreaking paper 'Theory of the Alternating-Gradient Synchrotron' in which these variables were first introduced in the context of accelerator physics [14]. They are also called Twiss parameters, named after the British astronomer Richard Q. Twiss (for reasons unknown even to Twiss himself). The equation (1.29) now becomes,

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{s_0+C} = \begin{pmatrix} \cos\Delta\phi + \alpha\sin\Delta\phi & \beta\sin\Delta\phi \\ -\gamma\sin\Delta\phi & \cos\Delta\phi - \alpha\sin\Delta\phi \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix}_{s_0} \tag{1.34}$$

Now the phase advance in equation (1.30) can be interpreted as,

$$\phi(s_0 \to s_0 + C) = \Delta\phi = \int_{s_0}^{s_0+C} \frac{k}{w^2(s)}\,\mathrm{d}s = \int_{s_0}^{s_0+C} \frac{1}{\beta(s)}\,\mathrm{d}s \tag{1.35}$$

We see that $\beta(s)$ can be interpreted as the wavelength in the period $s_0 \to s_0 + C$, and it gives us a sense of the amplitude of a particle's position. This is known as the 'betatron function' or simply the 'amplitude function'. And the oscillation exhibited by the particle as they go around in an accelerator is called 'betatron oscillation'. Thus the solution to the equation of motion (1.17) can be rewritten as,

$$\boxed{x(s) = A\sqrt{\beta(s)}\cos[\phi(s) + \delta]} \tag{1.36}$$

where $A = \sqrt{k}$.

1.4.1 Computing the Courant-Snyder Parameters

Now that we have solved the equation of motion in two slightly different ways in the last two sections, we can combine them both to get an insight into how to compute the Courant-Snyder parameters.

Let us suppose we know the strengths K of each element of the beam line and we compute the *net* propagation matrix (derived in section 1.3.1) through one repeat period,

$$M_{\text{net}} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{1.37}$$

This should, in principle, be equivalent to the matrix we had derived in the above section (equation (1.34)),

$$M_{\text{net}} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} \cos\Delta\phi + \alpha\sin\Delta\phi & \beta\sin\Delta\phi \\ -\gamma\sin\Delta\phi & \cos\Delta\phi - \alpha\sin\Delta\phi \end{pmatrix} \tag{1.38}$$

If we numerically have the values for (a, b, c, d), then we could simply compute the Courant-Snyder parameters at a given point in a lattice through,

$$\Delta\phi = \cos^{-1}\left(\frac{a+d}{2}\right) \tag{1.39}$$

$$\beta = \frac{b}{\sin\Delta\phi} \tag{1.40}$$

$$\alpha = \frac{a-d}{2\sin\Delta\phi} \tag{1.41}$$

$$\gamma = \frac{1+\alpha^2}{\beta} \tag{1.42}$$

The same scheme could be employed to find the Courant-Snyder parameters at *all* points in the lattice. Once we have $\alpha(s)$, $\beta(s)$, and $\gamma(s)$ for any s in the lattice, we can easily compute the evolution of a particle's (x, x') by computing the transfer matrix between two points in the lattice s_1 and s_2.

$$M_{\text{transfer}} = \begin{pmatrix} \sqrt{\frac{\beta_2}{\beta_1}}[\cos\Delta\phi + \alpha_1\sin\Delta\phi] & \sqrt{\beta_1\beta_2}\sin\Delta\phi \\ -\frac{1+\alpha_1\alpha_2}{\sqrt{\beta_1\beta_2}}\sin\Delta\phi + \frac{\alpha_1-\alpha_2}{\sqrt{\beta_1\beta_2}}\cos\Delta\phi & \sqrt{\frac{\beta_1}{\beta_2}}[\cos\Delta\phi + \alpha_2\sin\Delta\phi] \end{pmatrix} \tag{1.43}$$

where $\Delta\phi$ is the phase advance in the particle's position as it goes from s_1 to s_2, given by,

$$\Delta\phi(s_1 \to s_2) = \int_{s_1}^{s_2} \frac{\mathrm{d}s}{\beta(s)} \tag{1.44}$$

The total number of raw oscillations executed by the particle for one turn around a closed-loop accelerator is called the 'tune', and is given by,

$$\nu = \frac{1}{2\pi} \oint \frac{\mathrm{d}s}{\beta(s)} \tag{1.45}$$

Starting from solving the basic Lorentz equation, we now have developed a formalism to describe particles' motion as they go around in an accelerator. We next introduce the concept of phase-space and the evolution of particles' (x, x') in the phase-space.

1.5 Phase Space

A convenient way to study the evolution of particles is in their phase space, a space from which all the information regarding the dynamics of the system can be gotten. In a general dynamical system with N dimensions in physical space, the dimension of phase space would be $2N$.

In accelerator physics, we are usually interested in the evolution of particles in the horizontal and the vertical plane, and are also interested in the particle's kinetic energy and its 'arrival' time. Thus a typical particle tracking analyses would comprise of a $6N$-dimensional phase space.

However, in our case, we have for now assumed the particle's kinetic energy to be a constant, the particle arrives 'on-time', and also that its motion is decoupled in the horizontal and vertical planes. So it suffices for our phase space dimension to be just two: say (x, x')

and do not involve any dispersion term.

We know,

$$x(s) = A\sqrt{\beta(s)}\cos\left[\phi(s) + \delta\right]$$

Differentiating $x(s)$,

$$x'(s) = A\frac{\beta'(s)\cos\left[\phi(s) + \delta\right]}{2\sqrt{\beta(s)}} - A\frac{\beta(s)\phi'(s)\sin\left[\phi(s) + \delta\right]}{\sqrt{\beta(s)}}$$

Since $x'(s)$ is a bit clumsy, we devise a new phase space called the 'normalized' phase space, whose variables are x and $\alpha x + \beta x'$.

$$\beta x' = \beta\left(A\frac{\beta'\cos\left[\phi + \delta\right]}{2\sqrt{\beta}} - A\frac{\beta\phi'\sin\left[\phi + \delta\right]}{\sqrt{\beta}}\right)$$

Using the relation, $\phi'(s) = 1/\beta(s)$,

$$\begin{aligned}
\beta x'(s) &= A\frac{\beta\beta'\cos\left[\phi + \delta\right]}{2\sqrt{\beta}} - A\frac{1}{\beta}\frac{\beta^2\sin\left[\phi + \delta\right]}{\sqrt{\beta}} \\
&= A\frac{\sqrt{\beta}\beta'\cos\left[\phi + \delta\right]}{2} - A\sqrt{\beta}\sin\left[\phi + \delta\right] \\
\alpha x(s) &= -A\frac{\beta'}{2}\sqrt{\beta}\cos\left[\phi + \delta\right] \\
\alpha x(s) + \beta x'(s) &= \cancel{-A\frac{\beta'}{2}\sqrt{\beta}\cos\left[\phi + \delta\right]} + \cancel{A\frac{\sqrt{\beta}\beta'\cos\left[\phi + \delta\right]}{2}} - A\sqrt{\beta}\sin\left[\phi + \delta\right]
\end{aligned}$$

Thus,

$$\boxed{x(s) = A\sqrt{\beta(s)}\cos\left[\phi(s) + \delta\right]} \tag{1.46}$$

$$\boxed{\alpha(s)x(s) + \beta(s)x'(s) = -A\sqrt{\beta(s)}\sin\left[\phi(s) + \delta\right]} \tag{1.47}$$

Squaring and summing equations (1.46) and (1.47), we have

$$x^2 + (\alpha x + \beta x')^2 = A^2 \Big[\beta \cos^2 [\phi(s) + \delta] + \beta \sin^2 [\phi(s) + \delta] \Big]$$

$$A^2 \beta = x^2 + \alpha^2 x^2 + \beta^2 x'^2 + 2\alpha\beta x x'$$

$$A^2 = \frac{1}{\beta} \Big(x^2 + \alpha^2 x^2 + \beta^2 x'^2 + 2\alpha\beta x x' \Big)$$

$$A^2 = \Big(x^2 \underbrace{\Big(\frac{1 + \alpha^2}{\beta} \Big)}_{=\gamma} + \beta x'^2 + 2\alpha x x' \Big)$$

$$\boxed{A^2 = \gamma x^2 + \beta x'^2 + 2\alpha x x'} \tag{1.48}$$

Since A is a constant, we see that the RHS of above equation is an invariant. (This indirectly stems from the fact that the determinant of the matrix (1.34) is equal to one. And that, in turn, is a reflection of the fact the magnetic force is a conservative force.)

We note here that in the normalized phase space, the trajectory taken by a particle as it goes around the ring (with linear elements) would be a circle. And the transfer matrix for a quadrupole acting on the normalized phase space coordinates would simply be a rotation matrix. The evolution of a single particle in the normalized phase space is given in Fig. 1.4.

1.5.1 Emittance

The invariant we had derived in the previous section serves an important purpose in the analysis of beam dynamics. The beam can be thought of as points distributed in the phase space and we can, in many ways, leverage the invariance property to our advantage to

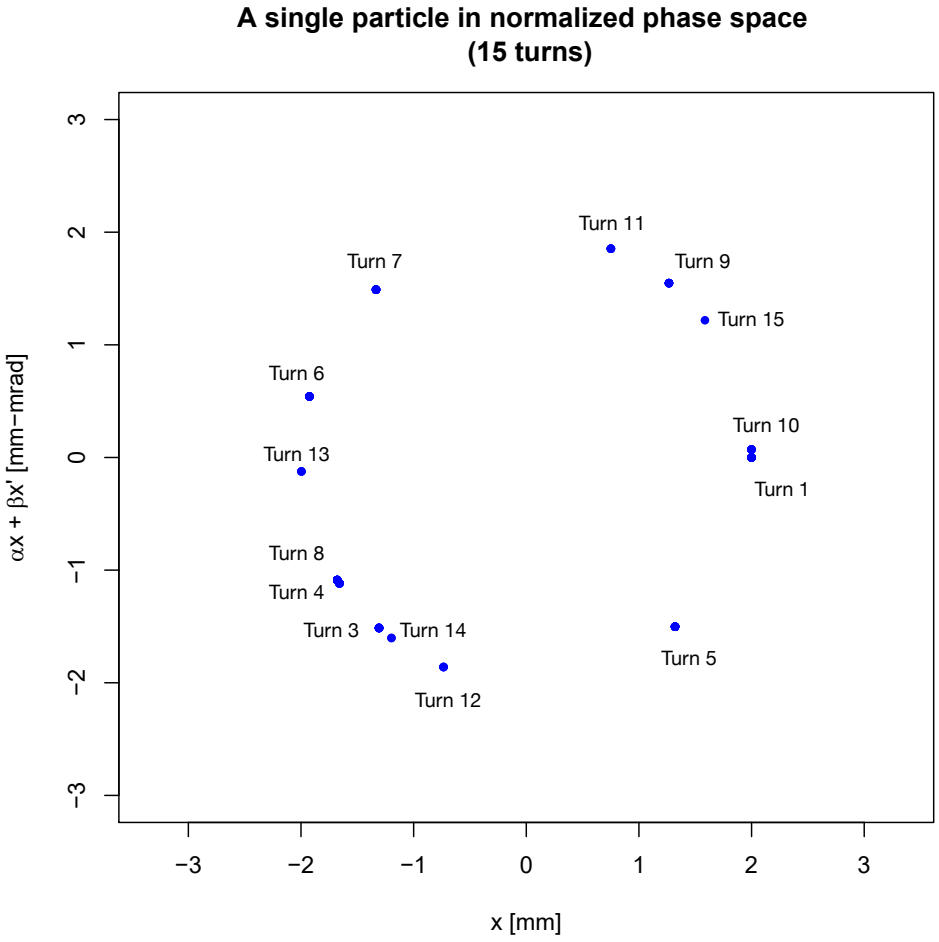

Figure 1.4: Turn-by-turn trajectory of a single particle in the normalized phase space in an accelerator with only linear elements (drifts elements, dipoles, and quadrupoles). We see that the particle traverses a circle in the normalized phase space.

physically measure different characteristics of the beam. One such important characteristic is the emittance of the beam (which shall be mathematically defined shortly).

The emittance of a beam is measure of its 'size' in the phase space. In an ideal world, we would like for this area to be zero, which would imply all the particles are *exactly* on the ideal design trajectory with also zero angular deviation from the ideal trajectory. However, it is an almost impossible task for many accelerators in real-life to have a zero emittance,

and this area in phase space is a good quantity to keep track of to measure the quality of the beam as the particles get transported.

We note that as we run the clock t (or equivalently as the particle travels through s), a single particle's state in the phase space of (x, x') would traverse an elliptical curve in accordance with the equation (1.48) which we had derived in the previous subsection. The characteristics of the ellipse turns out to be of the form presented in Fig. 1.5.

The general equation of an ellipse can be written as,

$$ax^2 + 2bxx' + cx'^2 = d \tag{1.49}$$

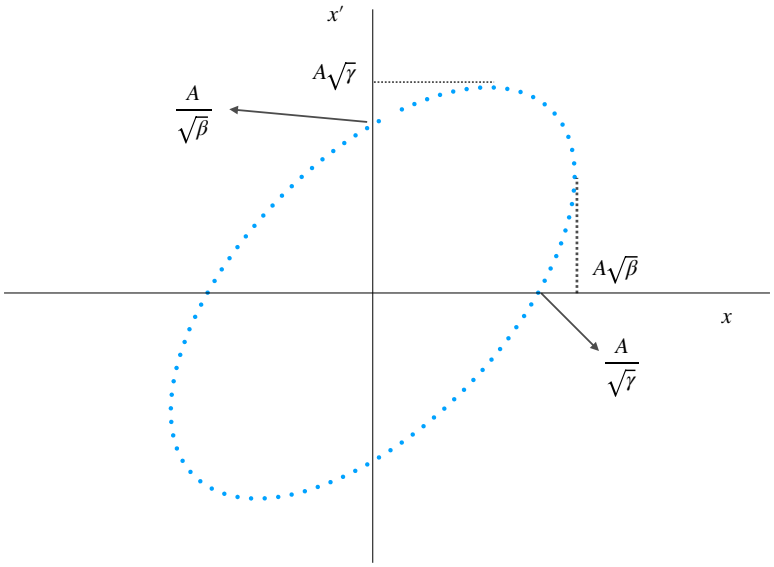

Figure 1.5: Turn-by-turn trajectory of a single particle in phase space in an accelerator with only linear elements (drifts elements, dipoles, and quadrupoles).

And the area of this ellipse would be,

$$\text{Area} = \frac{\pi d}{\sqrt{ac - b^2}} \tag{1.50}$$

Looking at equation (1.48), we identify that $d = A^2$, $a = \gamma$, $b = \alpha$, and $c = \beta$. So the area of our ellipse would be,

$$\text{Area} = \frac{\pi A^2}{\sqrt{\gamma\beta - \alpha^2}} \tag{1.51}$$

But from the definition of γ given in (1.33), we see

$$\sqrt{\gamma\beta - \alpha^2} = 1$$

Hence,

$$\text{Area} = \frac{\pi A^2}{\sqrt{\gamma\beta - \alpha^2}} = \pi A^2 \tag{1.52}$$

We thus define emittance ϵ as the area of this ellipse in phase space,

$$\boxed{\frac{\epsilon}{\pi} = \gamma x^2 + 2\alpha x x' + \beta x'^2} \tag{1.53}$$

We have in Fig. (1.5) phase space trajectory of a single particle. For a bunch of particles that travel around the accelerator, a snapshot of all the particles' phase space portrait we would see something like in Fig. (1.6).

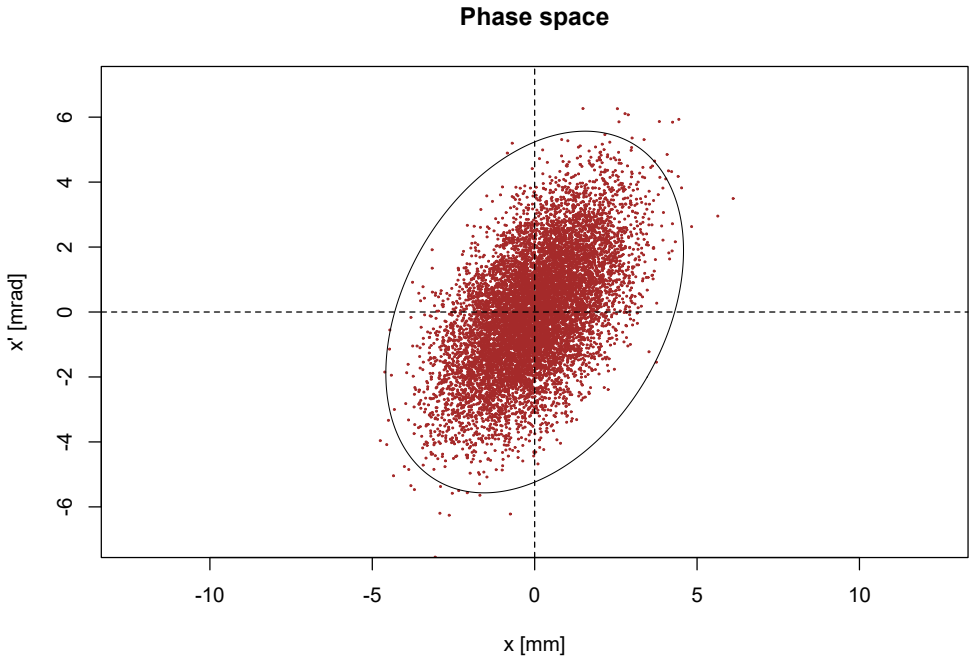

Figure 1.6: A typical phase space snapshot of circulating particles at a given location in an accelerator. The area of the ellipse would be ϵ, where ϵ denotes the emittance of this beam.

1.6 Beam Line Elements

A typical particle accelerator consists of three main beam elements: drift element, dipole, and quadrupole. We shall briefly review the functions of these elements and also represent them using the formalism we have developed thus far. We shall also look into sextupoles, which is crucial not only for curtailing something called chromaticity effects[9], but is the main driver of the third-integer resonance extraction.

[9]Chromaticity in the context of beam physics is defined as the change in the particle's tune that is proportional to its momentum deviation from the ideal momentum.

1.6.1 Drift Element

The drift element is the simplest of all elements. It is just an empty beam pipe with a finite aperture, and the particle simply coasts inside this drift element. The transfer matrix for the drift element would correspond to,

$$M_{\text{drift}} = \begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix} \tag{1.54}$$

where L is the length of the drift element. We see that the particle's angle remains unchanged (it is simply coasting after all), and its deviation in the horizontal/vertical position depends on how long it has coasted (i.e., on L).

1.6.2 Dipole

The main function of dipole is to bend the particle by a certain angle with a radius of curvature ρ. This corresponds to Hill's equation,

$$\frac{\mathrm{d}^2 x}{\mathrm{d}s^2} + \frac{1}{\rho^2} = 0 \tag{1.55}$$

The solution to this would be,

$$M_{\text{dipole}} = \begin{pmatrix} \cos\theta & \rho\sin\theta \\ (-1/\rho)\sin\theta & \cos\theta \end{pmatrix} \tag{1.56}$$

But in the limit of element being thin, i.e., for $\theta \to 0$, we would have,

$$M_{\text{dipole}} = \begin{pmatrix} 1 & \rho\theta \\ -\theta/\rho & 1 \end{pmatrix} \tag{1.57}$$

But $\rho\theta$ = arc length $\approx L$, and $-\theta/\rho \approx 0$ since ρ would be typically be very large. Thus, in this limit (which is not always true), the dipole element approximates a drift element.

1.6.3 Quadrupole

The primary purpose of a quadrupole is to 'focus' a particle that is going astray in a given plane. The price we pay for this, owing to the conservation of emittance, is that we would 'defocus' the same particle on the other plane. But we overcome this defocussing by placing focussing quadrupoles and defocussing quadrupoles successively so the beam, as a net effect, is contained on both planes.

The transfer matrix for a focussing quadrupole would be the one that we had derived earlier,

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{\text{out}} = \begin{pmatrix} \cos(\sqrt{K}l) & \frac{1}{\sqrt{K}}\sin(\sqrt{K}l) \\ -\sqrt{K}\sin(\sqrt{K}l) & \cos(\sqrt{K}l) \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix}_{\text{in}}$$

where K is the quadrupole strength.

Similarly, the transfer matrix for a defocussing quadrupole would contain hyperbolic sine and cosine functions in place of sine and cosine,

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{\text{out}} = \begin{pmatrix} \cosh(\sqrt{K}l) & \frac{1}{\sqrt{K}}\sinh(\sqrt{K}l) \\ -\sqrt{K}\sinh(\sqrt{K}l) & \cosh(\sqrt{K}l) \end{pmatrix} \begin{pmatrix} x \\ x' \end{pmatrix}_{\text{in}}$$

1.6.4 Sextupole

Sextupole is a non-linear element that, as a particle passes through it, changes a particle's x' proportional to its x^2. Sextupole is the most crucial element in a third-integer resonant extraction as it is the sextupole field that induces the third-integer resonance. Here we shall briefly discuss the equation of motion as the particle goes through a sextupole element.

The scalar potential of a magnet with $2m$ poles can be given by [15],

$$\Phi = A_m \mathrm{Re}(x+iy)^m + B_m \mathrm{Im}(x+iy)^m \tag{1.58}$$

The first term denotes the fields of a skewed sextupole magnet and the second term denotes the normal sextupole magnet.

The fields of the normal sextupole magnet can be obtained by taking the divergence of Φ,

$$B_x = -\frac{\partial \Phi}{\partial x} \tag{1.59}$$

$$B_y = -\frac{\partial \Phi}{\partial y} \tag{1.60}$$

For the normal sextupoles, $m = 3$, and this gives us,

$$\Phi = B_3 \mathrm{Im}[(x+iy)^3] \tag{1.61}$$

$$= B_3 \mathrm{Im}[x^3 - xy^2 + i2x^2y + ix^2y - iy^3 - 2xy^2] \tag{1.62}$$

$$= B_3[2x^2y + x^y - y^3] \tag{1.63}$$

So the B-field components are,

$$\begin{aligned} B_y &= -\frac{\partial \Phi}{\partial y} \\ &= -\frac{\partial}{\partial y} B_3[2x^2y + x^2y - y^3] \\ &= -B_3[-2x^2 - x^2 - 3y^2] \\ &= -B_3[x^2 + y^2] \end{aligned} \tag{1.64}$$

And,

$$\begin{aligned} B_x &= -\frac{\partial \Phi}{\partial x} \\ &= -\frac{\partial}{\partial x} B_3[2x^2y + x^2y - y^3] \\ &= -B_3[4xy + 2xy] \\ &= -6B_3xy \end{aligned}$$

The Taylor expansion of the magnetic field w.r.t $x = y = 0$ could be written as,

$$B_y = B_0 + \frac{1}{1!}\frac{\mathrm{d}B_y}{\mathrm{d}x}x + \frac{1}{2}\frac{\mathrm{d}^2B_y}{\mathrm{d}x^2} + ... \tag{1.65}$$

Comparing (1.65) with (1.64), we see that,

$$B_3 = -\frac{1}{6}\frac{\mathrm{d}^2B_y}{\mathrm{d}x^2} \tag{1.66}$$

Thus the horizontal and vertical fields in a sextupole can be written as,

$$B_y = \frac{1}{2}\frac{\mathrm{d}^2 B_y}{\mathrm{d}x^2}(x^2 - y^2) \tag{1.67}$$

$$B_x = \frac{\mathrm{d}^2 B_y}{\mathrm{d}x^2} xy \tag{1.68}$$

If we assume the length of the sextupole magnet to be small, the change in the particle's position after it exits the sextupole, δx, can be assumed to be zero. But the kick given by the sextupole would change the direction of the particle's trajectory. This simple fact of sextupole changing the angle of the particle proportional to its x^2 leads to extremely interesting non-linear phenomena in the phase space and this non-linearity is crucial for third-integer resonant slow extraction. This shall be described in detail in the second chapter.

If $(B\rho)$ is the magnetic rigidity, then,

$$\delta x' = \frac{B_y}{(B\rho)} \tag{1.69}$$

$$= \frac{1}{2}\frac{l_s}{(B\rho)}\frac{\mathrm{d}^2 B_y}{\mathrm{d}x^2}(x^2 - y^2) \tag{1.70}$$

$$= \frac{1}{2}k' l_s (x^2 - y^2) \tag{1.71}$$

where k' is the normalized sextupole gradient.

Thus, the transfer matrix for a particle going through a sextupole can be written as,

$$\begin{pmatrix} x \\ x' \end{pmatrix}_{\text{out}} = \begin{pmatrix} 0 \\ k x_{\text{in}}^2 \end{pmatrix} + \begin{pmatrix} x \\ x' \end{pmatrix}_{\text{in}} \tag{1.72}$$

Now that we are equipped with the arsenal to describe transverse dynamics of particles in an accelerator, we shall, in the next chapter, briefly overview the theory of third-integer slow extraction, the *Mu2e* experiment, Fermilab's accelerator system pertinent to *Mu2e*, and

the need (and possible schemes) for delivering muons through resonant slow extraction to the *Mu2e* experiment.

CHAPTER 2

THIRD INTEGER RESONANT EXTRACTION AT FERMILAB

This chapter contains a basic introduction to the physics of third integer resonant slow extraction, the *Mu2e* experiment, the need for pulsed proton pulses for *Mu2e* , an overview of Fermilab's accelerator system pertinent to *Mu2e* , and extraction design parameters for *Mu2e* .

2.1 Theory of Resonant Extraction

2.1.1 Resonant Extraction

Resonant extraction is a beam physics process by which a circulating beam's horizontal size is gently increased and a slice of the beam is extracted turn by turn and sent to a desired location through a transfer beam line. The horizontal beam size is made to increase by purposefully driving the beam close to a resonance condition.

For third-integer resonant extraction, we use sextupole elements to excite the third-integer resonance and dedicated tune ramping quadrupoles to drive the beam's horizontal tune closer to a third-integer resonant tune i.e., $\nu_x \rightarrow (n \pm 1/3)$ or $(n \pm 2/3)$, where n is an integer.

The extraction of the beam is typically achieved using an electrostatic septa (placed at the extraction location) which gives an instantaneous horizontal kick to the slice of beam that lies past the septum position in the horizontal plane i.e., when $x > x_{\text{septum}}$. This horizontal

deflection to the beam directs it to an extraction beam line whereby it is transported to its final location.

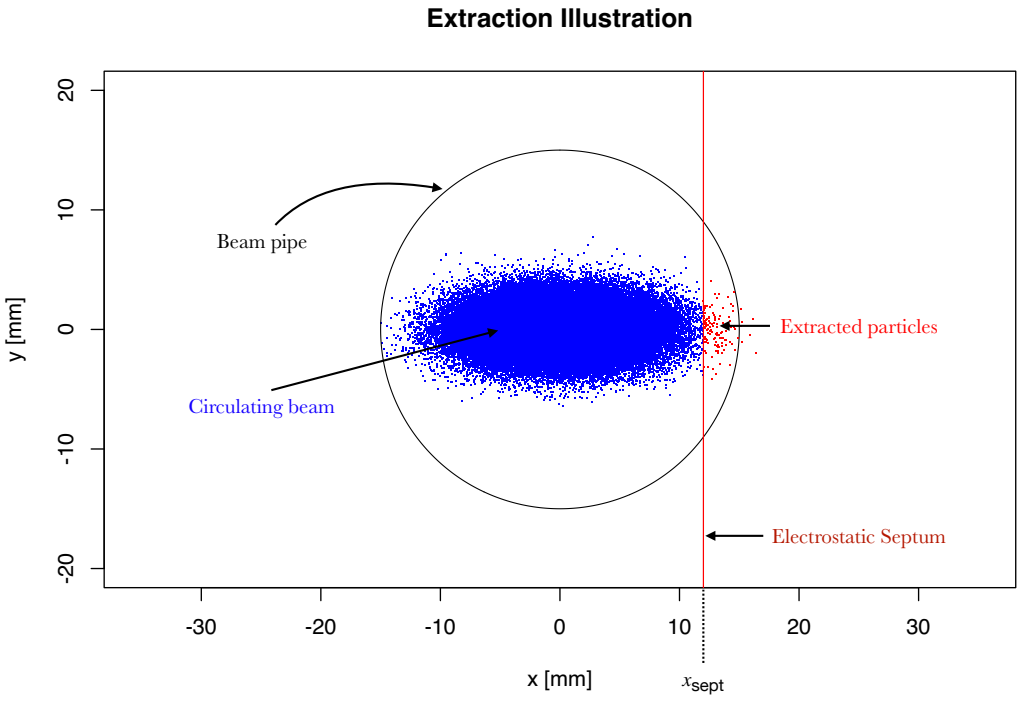

Figure 2.1: An illustration of snapshot of the beam in the physical space at extraction location. The resonant condition makes the horizontal beam size increase and a slice of circulating beam (that is past the position of the electrostatic septum) is extracted.

The third-integer extraction process is inherently nonlinear and we shall briefly overview in this subsection the formalism of the nonlinear dynamics.

2.1.2 Hamiltonian Formalism

We recall from chapter 1 that a particle's x' changes proportional to the square of its position x^2 when it passes through a sextupole magnet element. This very simple fact

introduces a rich non-linear behavior into particles' dynamics, whose effects are especially compounded when the horizontal tune of the beam approaches very close to $\nu = (n \pm 1/3)$ or $(n \pm 2/3)$, where n is an integer.

It is well served if we analyze the third-integer resonance in the normalized coordinates. When there are only linear elements in the beam line, we recall that the particle follows a circular trajectory in the normalized phase space. This would simply be a rotation in the normalized phase space.

The transfer matrix in the normalized coordinates for particles making N turns in an accelerator with only linear elements could be written as [1],

$$M = \begin{pmatrix} \cos(2\pi N Q_h) & \sin(2\pi N Q_h) \\ -\sin(2\pi N Q_h) & \cos(2\pi N Q_h) \end{pmatrix} \tag{2.1}$$

where Q_h is the horizontal tune. For reasons to be described shortly, we are interested in visualizing the particles' behavior not *every* turn but every three turns.

Let us compute the transfer matrix (with only linear elements) for $N = 1$, 2, and 3.

$$M_1 = \begin{pmatrix} \cos(2\pi Q_h) & \sin(2\pi Q_h) \\ -\sin(2\pi Q_h) & \cos(2\pi Q_h) \end{pmatrix} \tag{2.2}$$

$$M_2 = \begin{pmatrix} \cos(4\pi Q_h) & \sin(4\pi Q_h) \\ -\sin(4\pi Q_h) & \cos(4\pi Q_h) \end{pmatrix} \tag{2.3}$$

$$M_3 = \begin{pmatrix} \cos(6\pi Q_h) & \sin(6\pi Q_h) \\ -\sin(6\pi Q_h) & \cos(6\pi Q_h) \end{pmatrix} \tag{2.4}$$

[1] An extensive review of the third-integer resonant extraction could be found in the PhD thesis of Marco Pullia, titled '*Dynamics of slow extraction and its influence on transfer lines design*', 1999.

Since we expect the beam's tune to be driven very close to a third-integer resonance, let us write $Q_h = n + 1/3 + \delta Q$, where δQ is the tune distance to the third integer resonance.

This gives us,

$$M_1 = \begin{pmatrix} \cos\left(2\pi(n \pm 1/3 + \delta Q)\right) & \sin\left(2\pi(n \pm 1/3 + \delta Q)\right) \\ -\sin\left(2\pi(n \pm 1/3 + \delta Q)\right) & \cos\left(2\pi(n \pm 1/3 + \delta Q)\right) \end{pmatrix} \tag{2.5}$$

$$M_2 = \begin{pmatrix} \cos\left(4\pi(n \pm 1/3 + \delta Q)\right) & \sin\left(4\pi(n \pm 1/3 + \delta Q)\right) \\ -\sin\left(4\pi(n \pm 1/3 + \delta Q)\right) & \cos\left(4\pi(n \pm 1/3 + \delta Q)\right) \end{pmatrix} \tag{2.6}$$

$$M_3 = \begin{pmatrix} \cos\left(6\pi(n \pm 1/3 + \delta Q)\right) & \sin\left(6\pi(n \pm 1/3 + \delta Q)\right) \\ -\sin\left(6\pi(n \pm 1/3 + \delta Q)\right) & \cos\left(6\pi(n \pm 1/3 + \delta Q)\right) \end{pmatrix} \tag{2.7}$$

Simplifying the trigonometry,

$$\begin{aligned} \cos\left(2\pi(n \pm 1/3 + \delta Q)\right) &= \cos\left(2\pi n \pm 2\pi/3 + 2\pi\delta Q\right) \\ &= \underbrace{\cos\left(2\pi n\right)}_{=1} \cos\left(\pm 2\pi/3 + 2\pi\delta Q\right) + \underbrace{\cancel{\sin\left(2\pi n\right)}}_{=0} \sin\left(2\pi/3 + 2\pi\delta Q\right) \\ &= \cos\left(\pm 2\pi/3\right) \quad \underbrace{\cos\left(2\pi\delta Q\right)}_{=1 \text{ in the limit } \delta Q \to 0} + \sin\left(2\pi/3\right) \quad \underbrace{\sin\left(2\pi\delta Q\right)}_{=0 \text{ in the limit } \delta Q \to 0} \end{aligned}$$

$$\implies \cos\left(2\pi(n \pm 1/3 + \delta Q)\right) \approx -\frac{1}{2} \tag{2.8}$$

And similarly,

$$\begin{aligned}
\sin\left(2\pi(n+1/3+\delta Q)\right) &= \sin\left(2\pi n \pm 2\pi/3 + 2\pi\delta Q\right) \\
&= \underbrace{\sin\left(2\pi n\right)}_{=0}\cos\left(\pm 2\pi/3 + 2\pi\delta Q\right) \\
&\quad + \cos\left(2\pi n\right)\sin\left(\pm 2\pi/3 + 2\pi\delta Q\right) \\
&= \cos\left(2\pi n\right)\sin\left(\pm 2\pi/3 + 2\pi\delta Q\right) \\
&= \underbrace{\cos\left(2\pi n\right)}_{=1}\Big[\quad \underbrace{\sin\left(2\pi\delta Q\right)}_{=\,0 \text{ in the limit } \delta Q \to 0}\quad \cos\left(2\pi/3\right) \\
&\quad \pm \quad \underbrace{\cos\left(2\pi\delta Q\right)}_{=\,1 \text{ in the limit } \delta Q \to 0}\quad \sin\left(2\pi/3\right)\Big] \\
&= \pm\sin\left(2\pi/3\right)
\end{aligned}$$

$$\boxed{\sin\left(2\pi(n+1/3+\delta Q)\right) \approx \pm\frac{\sqrt{3}}{2}} \tag{2.9}$$

And similarly for a 2-turn matrix,

$$\cos\left(4\pi(n+1/3+\delta Q)\right) \approx \cos\left(4\pi/3\right) = -\frac{1}{2} \tag{2.10}$$

$$\sin\left(4\pi(n+1/3+\delta Q)\right) \approx \pm\sin\left(4\pi/3\right) = \mp\frac{\sqrt{3}}{2} \tag{2.11}$$

For the 3-turn matrix, we have,

$$\cos\left(6\pi(n\pm 1/3+\delta Q)\right) \approx \cos\left(6\pi/3\right) = 1 \tag{2.12}$$

And,

$$\begin{aligned}
\sin\left(6\pi(n+1/3+\delta Q)\right) &= \sin\left(6\pi n \pm 6\pi/3 + 6\pi\delta Q\right) \\
&= \underbrace{\sin\left(6\pi n\right)}_{=0}\cos\left(\pm 6\pi/3 + 6\pi\delta Q\right) \\
&\quad + \cos\left(6\pi n\right)\sin\left(\pm 6\pi/3 + 6\pi\delta Q\right) \\
&= \cos\left(6\pi n\right)\sin\left(\pm 6\pi/3 + 6\pi\delta Q\right) \\
&= \underbrace{\cos\left(6\pi n\right)}_{=1}\Big[\quad \underbrace{\sin\left(6\pi\delta Q\right)}_{=\,6\pi\delta Q \text{ in the limit } \delta Q\to 0} \qquad \cos\left(6\pi/3\right) \\
&\quad \pm \quad \underbrace{\cos\left(6\pi\delta Q\right)}_{=\,1 \text{ in the limit } \delta Q\to 0} \quad \underbrace{\sin\left(6\pi/3\right)}_{=\,0}\Big] \\
&= 6\pi\delta Q
\end{aligned}$$

Thus, the transfer matrices for 1, 2, and 3-turns in the normalized phase space become,

$$M_1 = \begin{pmatrix} -1/2 & \pm\sqrt{3}/2 \\ \mp\sqrt{3}/2 & -1/2 \end{pmatrix} \tag{2.13}$$

$$M_2 = \begin{pmatrix} 1/2 & \mp\sqrt{3}/2 \\ \pm\sqrt{3}/2 & 1/2 \end{pmatrix} \tag{2.14}$$

$$M_3 = \begin{pmatrix} 1 & 6\pi\delta Q \\ -6\pi\delta Q & 1 \end{pmatrix} \tag{2.15}$$

For a particle exactly on a third-integer tune, i.e., $\delta Q = 0$, we see that M_3 simply becomes an identity matrix, indicating that a particle on third integer tune will arrive at the same location in the phase space every three turns.

In order to get a three-turn mapping of the resonant extraction process, let us treat the motion through three turns as the superposition of three separate effects [13]:

- One turn + Sextupole + Two turns
- Two turns + Sextupole + One turn
- Three turns + Sextupole

We saw in the previous chapter that the transfer matrix of a sextupole simply induces a change in a particle's x' proportional to its x^2. This, along with the linear transfer matrices derived above, gives us what we need to calculate the net effect of a sextupole on a particle taking three turns around an accelerator.

For 'Three turns + Sextupole' (with the sextupole strength being S), the particle's normalized coordinates (X,X') after three turns would be,

$$\begin{aligned}
\begin{pmatrix} X_3 \\ X_3' \end{pmatrix}_{3+S} &= SM_3 \begin{pmatrix} X_0 \\ X_0' \end{pmatrix} \\
&= S \begin{pmatrix} 1 & 6\pi\delta Q \\ -6\pi\delta Q & 1 \end{pmatrix} \begin{pmatrix} X_0 \\ X_0' \end{pmatrix} \\
&= S \begin{pmatrix} X_0 + 6\pi\delta Q X_0' \\ -6\pi\delta Q X_0 + X_0' \end{pmatrix} \qquad (2.16) \\
&= \begin{pmatrix} X_0 + 6\pi\delta Q X_0' \\ -6\pi\delta Q X_0 + X_0' + S(X_0 + 6\pi\delta Q X_0')^2 \end{pmatrix} \qquad (2.17)
\end{aligned}$$

So 'Three turn + Sextupole', we have,

$$X_3 = X_0 + 6\pi\delta Q X_0' \tag{2.18}$$

$$X_3' = -6\pi\delta Q X_0 + X_0' + S(X_0 + 6\pi\delta Q X_0')^2 \tag{2.19}$$

For 'One turn + Sextupole + Two turns', we have,

$$\begin{aligned}
\begin{pmatrix} X_3 \\ X_3' \end{pmatrix}_{1+S+2} &= M_1 S M_2 \begin{pmatrix} X_0 \\ X_0' \end{pmatrix} \\
&= \begin{pmatrix} -1/2 & \pm\sqrt{3}/2 \\ \mp\sqrt{3}/2 & -1/2 \end{pmatrix} S \begin{pmatrix} 1/2 & \mp\sqrt{3}/2 \\ \pm\sqrt{3}/2 & 1/2 \end{pmatrix} \begin{pmatrix} X_0 \\ X_0' \end{pmatrix} \\
&= \begin{pmatrix} -1/2 & \pm\sqrt{3}/2 \\ \mp\sqrt{3}/2 & -1/2 \end{pmatrix} S \begin{pmatrix} X_0/2 \mp \sqrt{3}X_0'/2 \\ \pm\sqrt{3}X_0/2 + X_0'/2 \end{pmatrix} \\
&= \begin{pmatrix} -1/2 & \pm\sqrt{3}/2 \\ \mp\sqrt{3}/2 & -1/2 \end{pmatrix} \begin{pmatrix} X_0/2 \mp \sqrt{3}X_0'/2 \\ \pm\sqrt{3}X_0/2 + X_0'/2 + S\Big(X_0/2 \mp \sqrt{3}X_0'/2\Big)^2 \end{pmatrix} \qquad (2.20) \\
&= \begin{pmatrix} -X_0/4 \pm \sqrt{3}X_0'/4 + \sqrt{3}/2\Big[\pm \sqrt{3}X_0X_0'/4 \\ \qquad +S\Big(X_0/2 \mp \sqrt{3}X_0'/2\Big)^2\Big] \\ \mp\sqrt{3}/2\Big[X_0/2 \mp \sqrt{3}X_0'/2\Big] - 1/2\Big[\pm \sqrt{3}X_0/2 + X_0'/2 \\ \qquad +S\Big(X_0/2 \mp \sqrt{3}X_0'/2\Big)^2\Big] \end{pmatrix} \qquad (2.21)
\end{aligned}$$

Thus, for 'One turn + Sextupole + Two turns', we have,

$$X_3 = -\frac{1}{2}\Big(-\frac{1}{2}X_0 \pm \frac{\sqrt{3}}{2}X' \Big) \mp \frac{\sqrt{3}}{2}\Big(\mp \frac{\sqrt{3}}{2}X_0 - \frac{1}{2}X'_0 + S\Big(-\frac{1}{2}X_0 \pm \frac{\sqrt{3}}{2}X'_0 \Big)^2 \Big) \quad (2.22)$$

$$X'_3 = \pm\frac{\sqrt{3}}{2}\Big(-\frac{1}{2}X_0 \pm \frac{\sqrt{3}}{2}X' \Big) - \frac{1}{2}\Big(\mp \frac{\sqrt{3}}{2}X_0 - \frac{1}{2}X'_0 + S\Big(-\frac{1}{2}X_0 \pm \frac{\sqrt{3}}{2}X'_0 \Big)^2 \Big) \quad (2.23)$$

And similarly, for 'Two turns + Sextupole + One turn', we have,

$$\begin{aligned}
\begin{pmatrix} X_3 \\ X'_3 \end{pmatrix}_{2+S+1} &= M_2 S M_1 \begin{pmatrix} X_0 \\ X'_0 \end{pmatrix} \\
&= \begin{pmatrix} 1/2 & \mp\sqrt{3}/2 \\ \pm\sqrt{3}/2 & 1/2 \end{pmatrix} S \begin{pmatrix} -1/2 & \pm\sqrt{3}/2 \\ \mp\sqrt{3}/2 & -1/2 \end{pmatrix} \begin{pmatrix} X_0 \\ X'_0 \end{pmatrix} \\
&= \begin{pmatrix} -1/2 & \pm\sqrt{3}/2 \\ \mp\sqrt{3}/2 & -1/2 \end{pmatrix} S \begin{pmatrix} -X_0/2 \pm \sqrt{3}X'_0/2 \\ \mp\sqrt{3}X_0X'_0/4 \end{pmatrix} \\
&= \begin{pmatrix} -1/2 & \pm\sqrt{3}/2 \\ \mp\sqrt{3}/2 & -1/2 \end{pmatrix} \begin{pmatrix} X_0/2 \mp \sqrt{3}X'_0/2 \\ \pm\sqrt{3}X_0X'_0/4 + \Big(X_0/2 \mp \sqrt{3}X'_0/2\Big)^2 \end{pmatrix}
\end{aligned} \quad (2.24)$$

$$\begin{pmatrix} X_3 \\ X'_3 \end{pmatrix}_{2+S+1} = \begin{pmatrix} -X_0/4 \pm \sqrt{3}X'_0/4 \\ +\sqrt{3}/2\Big[\pm \sqrt{3}X_0X'_0/4 + \Big(X_0/2 \mp \sqrt{3}X'_0/2\Big)^2 \Big] \\ \mp\sqrt{3}/2\Big[X_0/2 \mp \sqrt{3}X'_0/2 \Big] \\ -1/2\Big[\pm \sqrt{3}X_0X'_0/4 + \Big(X_0/2 \mp \sqrt{3}X'_0/2\Big)^2 \Big] \end{pmatrix} \quad (2.25)$$

Thus, for 'Two turns + Sextupole + One turn', we have,

$$X_3 = -\frac{1}{2}\Big(-\frac{1}{2}X_0 \mp \frac{\sqrt{3}}{2}X' \Big) \pm \frac{\sqrt{3}}{2}\Big(\pm \frac{\sqrt{3}}{2}X_0 - \frac{1}{2}X'_0 + S\Big(-\frac{1}{2}X_0 \mp \frac{\sqrt{3}}{2}X'_0 \Big)^2 \Big) \quad (2.26)$$

$$X'_3 = \mp\frac{\sqrt{3}}{2}\Big(-\frac{1}{2}X_0 \mp \frac{\sqrt{3}}{2}X' \Big) - \frac{1}{2}\Big(\pm \frac{\sqrt{3}}{2}X_0 - \frac{1}{2}X'_0 + S\Big(-\frac{1}{2}X_0 \mp \frac{\sqrt{3}}{2}X'_0 \Big)^2 \Big) \quad (2.27)$$

Adding all three perturbative effects together, subtracting the initial coordinates (X_0, X'_0) from the equation and keeping only first-order in δQ, we have,

$$\Delta X_3 = 6\pi\delta Q X'_0 \pm \frac{\sqrt{3}}{2}S\Big(-\frac{1}{2}X_0 \mp \frac{\sqrt{3}}{2}X'_0 \Big)^2 \mp \frac{\sqrt{3}}{2}S\Big(-\frac{1}{2}X_0 \pm \frac{\sqrt{3}}{2}X'_0 \Big)^2 \quad (2.28)$$

$$\Delta X'_3 = -6\pi\delta Q X'_0 + SX_0^2 - \frac{1}{2}S\Big(-\frac{1}{2}X_0 \mp \frac{\sqrt{3}}{2}X'_0 \Big)^2 \mp \frac{\sqrt{3}}{2}S\Big(-\frac{1}{2}X_0 \pm \frac{\sqrt{3}}{2}X'_0 \Big)^2 \quad (2.29)$$

Simplifying, we get the following equation of motion:

$$\Delta X_3 = \quad 6\pi\delta Q X'_0 + \frac{3}{2}SX_0X'_0 \quad (2.30)$$

$$\Delta X'_3 = -6\pi\delta Q X'_0 + \frac{3}{4}S(X_0^2 - X_0'^2) \quad (2.31)$$

2.1.3 Kobayashi Hamiltonian

We had computed the three-turn evolution of particle dynamics in presence of sextupole field. Resonant extractions typically happen over a fairly long time relative to the revolution time period, thus the 3-turn duration can be considered as a single step time unit.

Since we have already obtained the equation of motion (albeit through transfer matrices), we can reconstruct the Hamiltonian H from those equations. (We note here that 'time' $\mathrm{d}n$ here is dimensionless since $\mathrm{d}n = \mathrm{d}t\,(f_0/3)$ where f_0 is the revolution frequency.)

$$\Delta X_3 = \frac{\Delta X}{\Delta t} = \frac{\partial H}{\partial X'} = 6\pi\delta Q X' + \frac{3}{2} S X X' \tag{2.32}$$

$$\Delta X_3' = \frac{\Delta X'}{\Delta t} = -\frac{\partial H}{\partial X} = -6\pi\delta Q X' + \frac{3}{4} S (X^2 - X'^2) \tag{2.33}$$

Integrating the above equations of motion, we get the Hamiltonian H,

$$H = \underbrace{3\pi\delta Q\Big(X^2 + X'^2\Big)}_{\text{First term}} + \underbrace{\frac{S}{4}\Big(3XX'^2 - X^3\Big)}_{\text{Second term}} \tag{2.34}$$

This particular Hamiltonian is known as the Kobayashi Hamiltonian. Like all Hamiltonians in a conservative system, the Kobayashi Hamiltonian is also a constant of motion. Even though the Kobayashi Hamiltonian is a first-order approximation (since we had neglected higher-orders in δQ), it is nonetheless a very useful tool to learn a lot about particle dynamics in a third-integer resonant extraction.

The first term, for instance, describes the motion in the phase space caused just by the linear elements. We can verify that by setting the sextupole strength to zero, i.e., $S = 0$, and we see that the phase space portraits in the normalized phase space can only be circles since $(X^2 + X'^2) = H/(3\pi\delta Q) = \text{constant}$. If we change δQ, we see that the phase space portraits would still remain circles, albeit with different radii.

But as we turn on the sextupole strength S, i.e., the second term introduces non-linearity into the system.

We note that when $H = (4\pi\delta Q)^3/S^2$, equation (2.34) factorizes into three straight lines,

$$\left(\frac{S}{4}X + \pi\delta Q\right)\left(\sqrt{3}X' + X - \frac{8\pi\delta Q}{S}\right)\left(\sqrt{3}X' - X + \frac{8\pi\delta Q}{S}\right) = 0 \tag{2.35}$$

These three lines form an equilateral triangle in the normalized phase space. The phase space region on or inside this triangle constitutes the 'stable region' and the region outside is the 'unstable region'. Each side of the triangle is called the 'separatrix'.

We can find the coordinates of the fixed points by setting

$$\frac{\partial H}{\partial X'} = \frac{\partial H}{\partial X} = 0 \tag{2.36}$$

Solving equation (2.36), we find the three fixed points (P_1, P_2, P_3) to be:

$$P_1 = \left(\frac{8\pi\delta Q}{S}, 0 \right) \tag{2.37}$$

$$P_2 = \left(-\frac{4\pi\delta Q}{S}, \frac{12\pi\delta Q}{\sqrt{3}S} \right) \tag{2.38}$$

$$P_3 = \left(-\frac{4\pi\delta Q}{S}, -\frac{12\pi\delta Q}{\sqrt{3}S} \right) \tag{2.39}$$

The fourth (trivial) fixed point is (0,0), but this corresponds to the case where a particle is on the ideal trajectory. This suggests that it is important to have a sufficiently non-zero emittance to facilitate third-integer resonant extraction.

From the geometric analysis, we see that the area of the stable triangle region is given by,

$$\text{Stable region area} = 48\sqrt{3}\pi\frac{(\delta Q)^2}{S^2} \tag{2.40}$$

When a particle is inside the stable region, its x amplitude does not monotonically increase but it instead executes a stable phase space motion whose trajectory takes a profile similar to that of a triangle. The 'triangle-ness' of a particle's phase space trajectory is directly related to how close its initial (X, X') was to the separatrix.

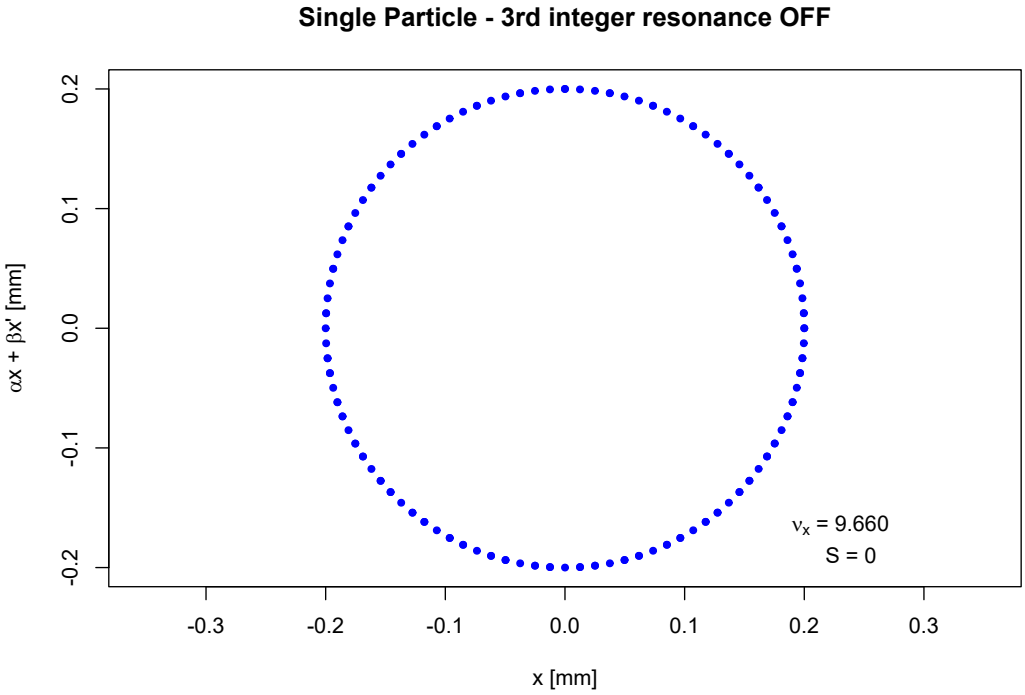

Figure 2.2: Trajectory of a single particle in the normalized phase space before exciting the third-integer resonance (i.e., $S = 0$).

As an illustration of the particle behavior, we see in Fig. 2.2 that when the sextupole is off, a particle's phase space trajectory traverses a circle.

The same particle (with the same initial condition) traverses a near triangular trajectory after switching on the sextupole field and keeping the quad strengths constant (Fig. 2.3). Even though the phase space trajectory is triangular, the particle is still stable, and we see the trajectory to close in on itself.

In Fig. 2.4, the horizontal tune of the particle is not constant but is driven closer to third-integer turn by turn. The changing tune makes the stable region shrink with every turn. After a few turns, the particle that was erstwhile in the stable region ends up outside the stable region, and its x starts to increase with every turn. After a finite number of turns,

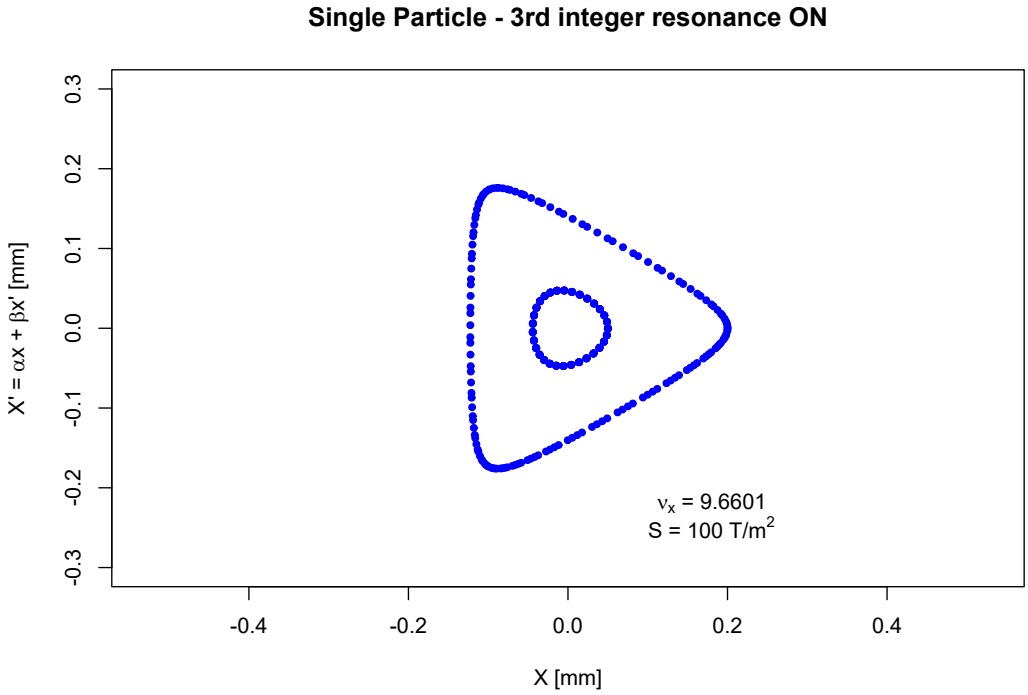

Figure 2.3: Trajectory of two particles in the normalized phase space after exciting the third-integer resonance (i.e., $S \neq 0$). The particle that takes the triangular trajectory has the same initial coordinates as the one in Fig. 2.2. We also see that a particle that is deep inside the stable region traverses almost a circular trajectory.

the particle will end up past the septum position $x > x_{\text{sept}}$, and the particle immediately gets extracted. This is the essence behind third-integer resonance extraction.

The three previous figures illustrated the extraction of a single particle, but the same conditions apply for a circulating bunch consisting of millions of particles. The phase space evolution of such a case is illustrated in Fig. 2.5

We note from equation (2.40) that since the area of the stable region depends directly on the tune distance to third-integer resonance and also depends inversely on the sextupole strength, we could introduce third-integer resonance to the beam through three possible methods:

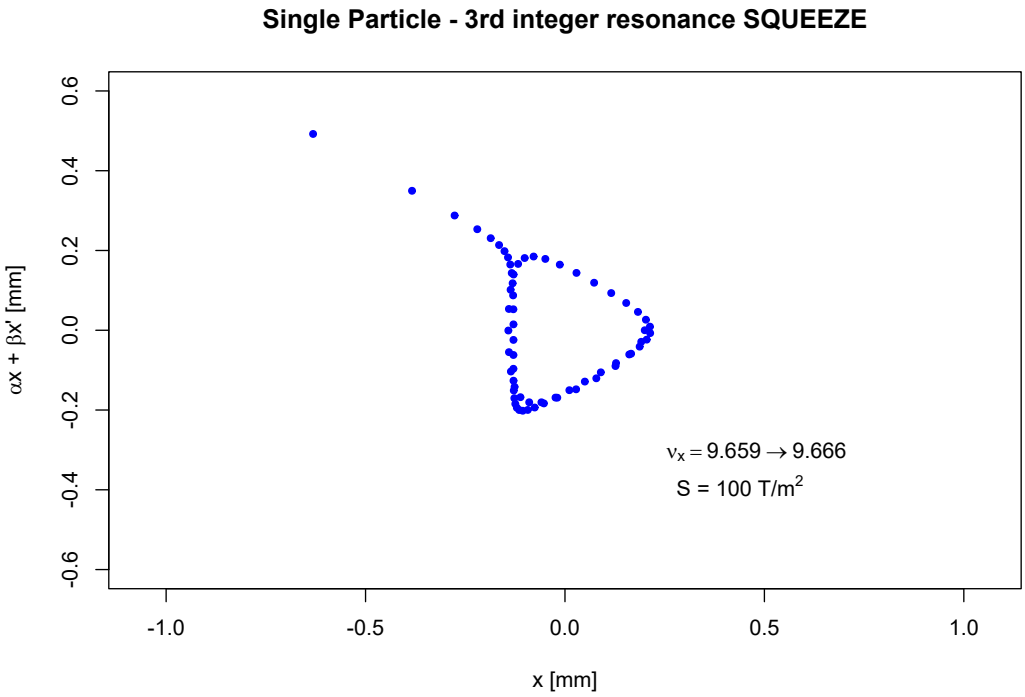

Figure 2.4: Evolution of phase space while approaching the third-integer resonance. We see the particle becoming unstable and its horizontal position increase turn by turn.

- Keeping the sextupole strength S constant and driving the horizontal tune ν_x to third-integer resonance using tune-ramping quadrupoles.
- Keeping the horizontal tune constant and ramping up the sextupole strength S.
- Ramping up both the sextupole strength (to increase the resonant strength) as well as ramping the fast quadrupoles' strength (to get ν_x close to the resonant tune).

For delivering muons to the *Mu2e* experiment, a third-integer resonance extraction scheme is to be employed using the first of the aforementioned methods, i.e., keeping the sextupole strength constant and ramping up the tune-shifting quadrupoles' strengths.

A dedicated set of 6 harmonic sextupoles would provide the sextupole field required to excite the third-integer resonance.

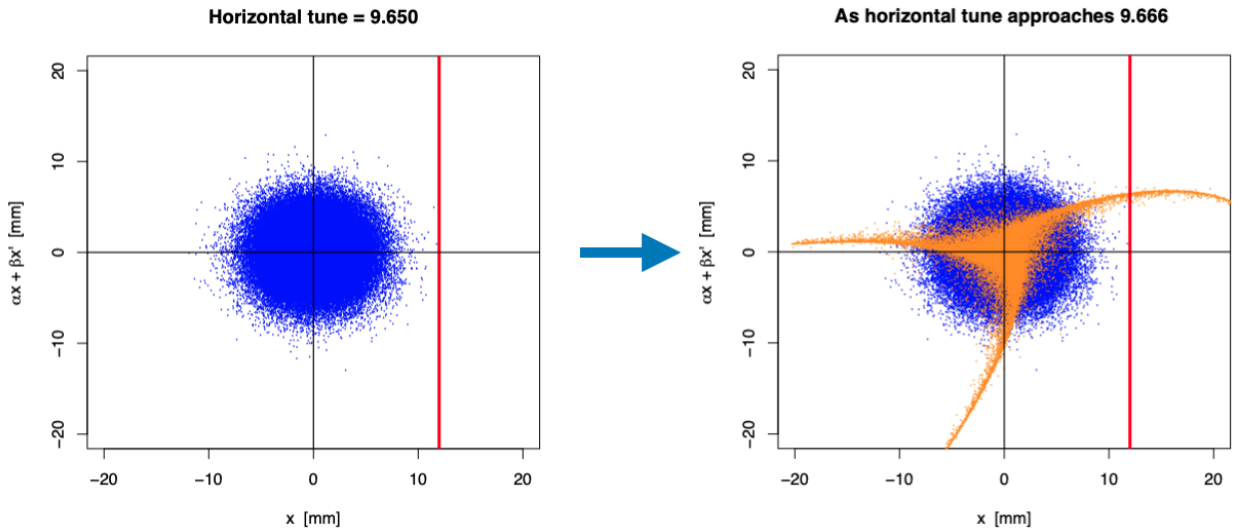

Figure 2.5: Evolution of phase space while approaching the third-integer resonance for multiple particles.

Three fast ramping quadrupoles would help drive the horizontal tune from 9.650 to a resonant tune of 9.666 (which is 29/3).

Further details of the resonant extraction scheme, and the motivation for using resonant extraction, for Mu2e is discussed in the next subsection.

2.2 The Mu2e Experiment

Mu2e is an upcoming experiment at Fermilab that intends to look for charged lepton flavor violation (CLFV) through the process $\mu^- N \to e^- N$, where N is a nucleus, by measuring the following ratio of events,

$$\mathrm{R}_{\mu\to e} = \frac{\mu^- N \to e^- N}{\mu^- N \to \text{all muon captures}} \tag{2.41}$$

The methodology behind the *Mu2e* experiment is to capture negatively charged muons in the Coulomb field of the Aluminum nucleus and measure the energy of the electron it

decays into. If there is an electron produced through a direct CLFV process, we expect its energy to be slightly less than the rest mass of the muon (which is almost equal to ≈ 104.97 MeV).

2.2.1 Physics Motivation

Mu2e , as the name suggests, is designed to look for *direct* conversion of a muon to an electron. The probability of this decay happening, in accordance with the Standard Model, is not non-zero but very small ($\approx 10^{-50}$).

However, there are other BSM physics models, such as Minimal Super Symmetric Models with right-handed neutrinos, SUSY with R-parity violation, as well as models with lepto-quarks, large extra dimensions, new gauge bosons, and a non-minimal Higgs sector, that suggest a higher probability of such CLFV interactions. If a non-zero value of $R_{\mu\to e}$ (in 2.41) is measured by the *Mu2e* experiment, it would be a direct hint of new physics. A near exhaustive list of physics motivations behind the search for CLFV can be found at [3].

2.2.1.1 Best Lower Bound Before *Mu2e*

The best bound there is (in 2022, at the time of writing of this thesis) on direct CLFV muon to electron conversion comes from the SINDRUM-II experiment at PSI [17], which used a magnetic spectrometer built especially for detecting direct muon to electron conversion [2].

The SINDRUM-II's experimental technique is similar to *Mu2e* in that muons were directed into an atom to displace an electron with a muon, capturing the muon in the Coulomb

[2] A collaboration was formed in 1987 between ETHZ, UZH, PSI, RWTH Aachen, and Univ. of Tbilisi [18] to measure $R_{\mu\to e}$ but the experiment could not commence due to technical difficulties until 1998 and the data was taken at PSI in 2000.

field of the atom's nucleus to create a muonic atom (the stopping target were gold atoms in the case of SINDRUM-II). Once the muon bound inside the atom decays, the detectors search for an electron that has directly converted from muon.

The muon to electron conversion in a typical muonic atom would result in an electron of energy,

$$E_e = m_\mu c^2 - B_\mu(Z) - R_N$$

where $B_\mu(Z)$ is the atomic binding energy in the nucleus and R_N is the nucleus recoil energy.

For the case of $\mu^- \to e^-$ with gold atoms, calculated to the first approximation ($B_\mu \propto Z^2$ and $R_N \propto A^{-1}$), the values of $B_\mu(Z) = 10.08$ MeV and $R_N = 0.025$ MeV, leaving the electron energy to be 95.55 MeV [19]. Using the data collected in 2000, the SINDRUM-II improved its own previous bound of muon to electron conversion $R_{\mu\to e}$ from $\approx 10^{-11}$ to a new bound of 7×10^{-13} (with 90 % confidence level).

In the case of Aluminum atom, the E_e is 104.97 MeV and the muon lifetime in the nucleus bound state is 864 ns [21] [22].

The goal of *Mu2e* experiment is to attain a single event sensitivity on the order of,

$$\mathrm{R}_{\mu^-\to e^-} = \frac{\mu^- N \to e^- N}{\mu^- N \to \text{all muon captures}} \approx 10^{-17}$$

which is a four order-of-magnitude improvement from SINDRUM-II's best lower bound.

2.2.2 Background and Sensitivity

In order to achieve such a high level of sensitivity of $R_{\mu^-\to e^-}$, various background effects that could mimic a muonic muon-to-electron signal needs to be taken into account. Also, (as is going to be explained shortly) reducing the background effects is one of the principal

motivations behind choosing resonant slow extraction scheme to deliver protons to the muon production target.

2.2.2.1 Radiative Pion Capture

One of the main background effects that motivates resonant slow extraction scheme for muon beam delivery is the radiative pion capture (RPC). When the high energy protons hit a muon production target, we have a flurry of secondaries, amongst which are pions. If these secondary pions reach the Aluminum stopping target and get into a bound state with the nucleus, then the following process could be possible ,

$$\pi^- N \rightarrow \gamma N'$$

where N' is an excited nuclear state and γ is a photon. This process is the RPC and has a probability of about 2.1% [25].

The average energy of a photon resulting from RPC could be around 110 to 120 MeV. This photon could have interactions that in turn could pair-produce an electron-positron pair, and the electron resulting from this pair-production might mimic a directly converted electron from muon.

There is another mode by which an electron-positron pair is produced, and that happens instantaneously through the annihilation of a virtual photon,

$$\pi^- N \rightarrow e^+ e^- N'$$

Coincidentally, the rate of pair production of both these channels are about equal (thanks to the geometry of the Al target and interaction length of the photon within this material).

Both these channels thus contribute almost equally to the background. The RPC background was one of the most significant in diminishing the signal in the SINDRUM-II experiment (that has so far set the best limit on direct muon-to-electron conversion).

The *Mu2e* experiment more or less cleverly avoids this background by using a pulsed beam structure with the aid of resonant extraction.

2.2.2.2 Motivation for Pulsed Beam Structure

For the *Mu2e* experiment, muons are produced by making high energy protons bombard a muon production target (made of tungsten). The time structure of these protons is not continuous but pulsed in time. This way, the muons and the secondary particles produced from the production target are also pulsed in time. The time interval between the pulses is approximately 1.695 μs.

When the first of proton pulses hits the muon production target, they produce the first pulse of pions and muons which are transported to the *stopping* target, i.e., Aluminum foil. The muons (and pions) then get bound in the Coulomb field of Al nucleus. If the pions from the secondaries happen to get captured in the nucleus through RPC mechanism and produce a photon (real or virtual), it would then promptly result in the pair production of e^-e^+.

The key in using this pulsed time structure is to avoid detecting the background from the RPC electrons by letting them first escape. Once all (or most) of the RPC electrons escape, we start the live data collection window for the detectors to search for the *Mu2e* electrons of particular energy 104.96 MeV. After the first proton micropulse arrives to the Al target, we wait for approximately 700 ns for most of the RPC electrons to escape before turning on the live search window. After 700 ns of wait time, we switch on the live search for a further 995 ns. By now, $\approx 1.6\mu$s has elapsed, and the next micropulse arrives to the

stopping target, and we switch off the live search window again to let the RPC electrons escape. An approximate time structure of this scheme can be seen in Fig. 2.6.

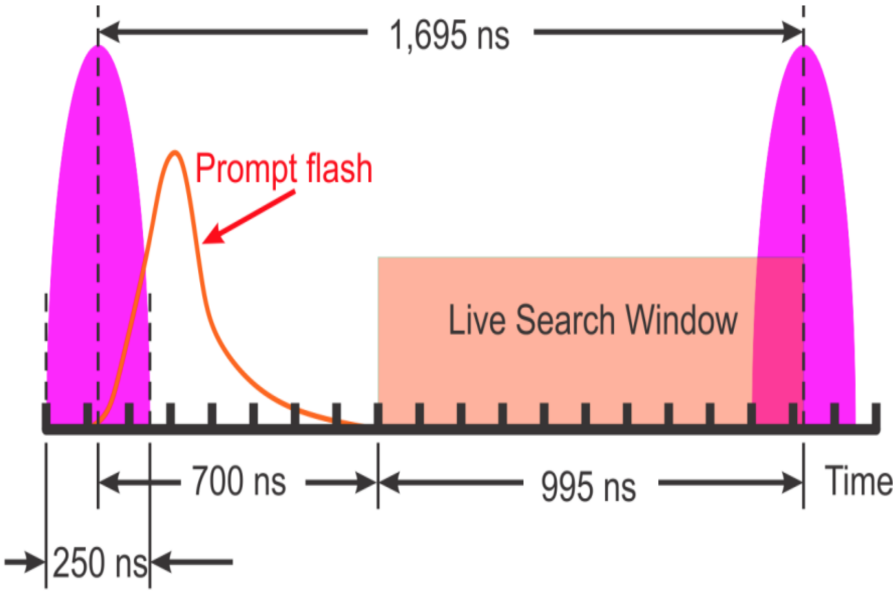

Figure 2.6: The muon delivery cycle to the stopping target is such that the live data taking window starts when the electrons produced from background effects have significantly escaped [3].

The quality of the pulsed beam structure is thus very important to the systematics of the *Mu2e* experiment. And this pulsed beam is provided by Fermilab's sophisticated accelerator system.

2.3 Accelerator System at Fermilab

The accelerator system at Fermilab is large and complex; here we discuss only the components that are part of the beam production for the 8 GeV kinetic energy protons at the Delivery Ring that makes the resonant slow extraction possible. For a more detailed overview of the accelerator system, the reader can refer to [26].

2.3.1 Pre-accelerator System

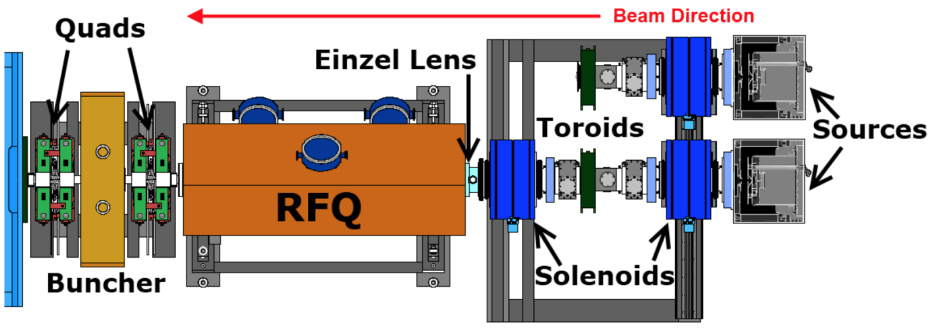

Figure 2.7: A schematic of the pre-accelerator system. [26]

Beam production for *Mu2e* starts at Fermilab's pre-accelerator system (Fig. 2.7) that produces and accelerates H^- ions to 35 KeV, which are then fed to the Linear Accelerator (linac). The pre-accelerator system is comprised of the following:

- Two H^- ion sources (one active and the other a spare)
- Low Energy Beam Transport (LEBT)
- Radio-frequency Quadrupole (RFQ) Cavity
- Medium Energy Beam Transport (MEBT)

The ion source gives out a pulse of 35 KeV H^- ions that is approximately 100 μs long at the rate of 15 Hz. This pulse train is then directed towards the Low Energy Beam Transport which contains solenoidal magnetic fields for transverse focussing of the H^- ions.

At the end of the source section, there is an Einzel lens that acts as a beam chopper to give a time structure to the rather lengthy pulse arriving from the source. The lens achieves this through an electrostatic potential of 38 KV and pushes the beam back at specific intervals

(shown in Fig.2.8). This is done because we do not want the entire pulse from the H^- source to be accelerated by the RFQ and the linac.

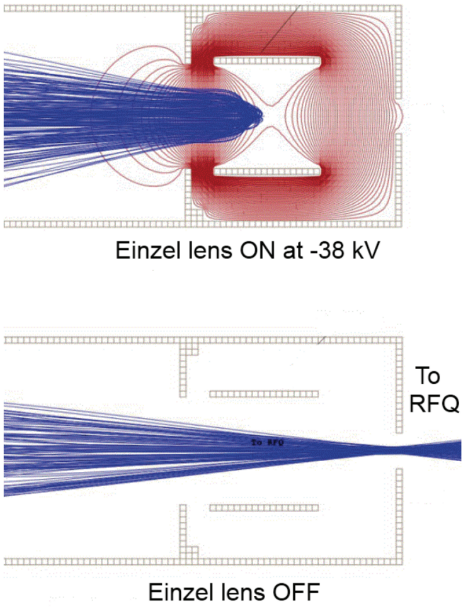

Figure 2.8: When on, the Einzel lens pushes back the H^- ions from the source by achieving an electrostatic potential of -38 KV. When off, it lets all of the beam through [26].

Once the beam goes past the Einzel lens, it is then fed into the radio-frequency quadrupole (RFQ) cavity that operates at 201.24 MHz. The RFQ then accelerates the H^- ions from 35 KeV to 750 KeV. One of the reasons we accelerate the H^- ions to 750 KeV even before sending it to linac is because the sooner we accelerate the particles, lesser will be the effects of repulsive forces amongst them (owing to special relativistic effects), preserving the emittance of the beam. As the RFQ accelerates the ions, it also transversely focusses the beam (using electric fields) at the same time.

After the ions get accelerated to 750 KeV, they are sent to the Medium Energy Beam Transport that consists of a quadrupole doublet that helps focus the beam on both the planes and an RF buncher that helps reinforce the bunch structure of the beam, i.e., it keeps the

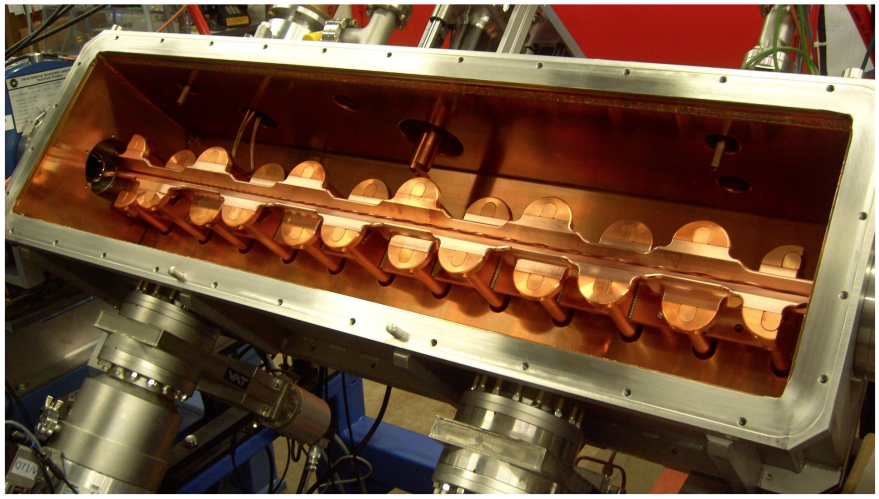

Figure 2.9: An inside view of the Radio Frequency Quadrupole Cavity which accelerates and also transversely focusses the beam at the same time.

spread in 'arrival time' of the particles less. (This is important because having the particles bunched together helps increase the efficiency of the linac.)

Once the particle exits the MEBT, it then enters the linac to get accelerated to about 400 MeV.

2.3.2 Fermilab Linac

Fermilab's linear accelerator consists of two portions: low-energy linac and high-energy linac.

The low-energy linac accelerates the H^- ions from 750 KeV to 116.5 MeV using five drift tube linac units. The low energy linac produces an oscillating electric field at the same RF frequency as the Enzel lens (at 201.24 MHz) to facilitate synchronization and maximize the efficiency of accelerating the ions. Let us suppose the $\vec{E}$-field oscillates from -E_0 to +E_0, with +E_0 pointing in the direction of intended acceleration.

Ideally, we would like for the particles to experience the electric field only when $\vec{E}(t)$ is positive. When the electric field starts pointing opposite to the beam direction, the particles would enter one of the many cleverly placed drift tubes (Fig. 2.10) that would shield them from experiencing the negative electric field outside the tube. And as the electric field starts pointing in the beam direction, the ions exit the drift tube to experience the electric field and get accelerated in the forward direction [3]. Once the ions traverse through the low-energy linac, it reaches an energy of 116.5 MeV, after which it enters the high-energy linac.

The high-energy linac consists of 7 side-coupled cavity RF stations and it operates at a higher frequency of 804.96 MHz (4 times 201.24 MHz), accelerating the ions to 400 MeV kinetic energy, after which they would be sent to the Booster.

2.3.3 Booster Ring

The Fermilab Booster Ring is a circular accelerator that takes in 400 MeV H^- ions, converts them to 400 MeV protons, and accelerates the protons to 8 GeV kinetic energy. The Booster is $2\pi \times 74.47$ m $= 467.9$ m in circumference and has 24 equal length repeating 'periods'.

At 400 MeV energy, the time period of revolution around the Booster Ring would take approximately 2.2 μs. If we let the H^- ions pass through for 2.2 μs, we would fill the booster with one-turn worth of protons.

The length of the H^- pulse that comes out of the linac is approximately 100 μs, and is made to pass through a chopper that selects n times 2.2 μs of the ion pulse and sends to the Booster. As the H^- ions are injected into the Booster, they are immediately made to pass through an electron stripping foil (made of Beryllium), converting them into protons (which

[3] The longitudinal dynamics of the particles are a little more involved than this, but this would suffice for our purpose of understanding.

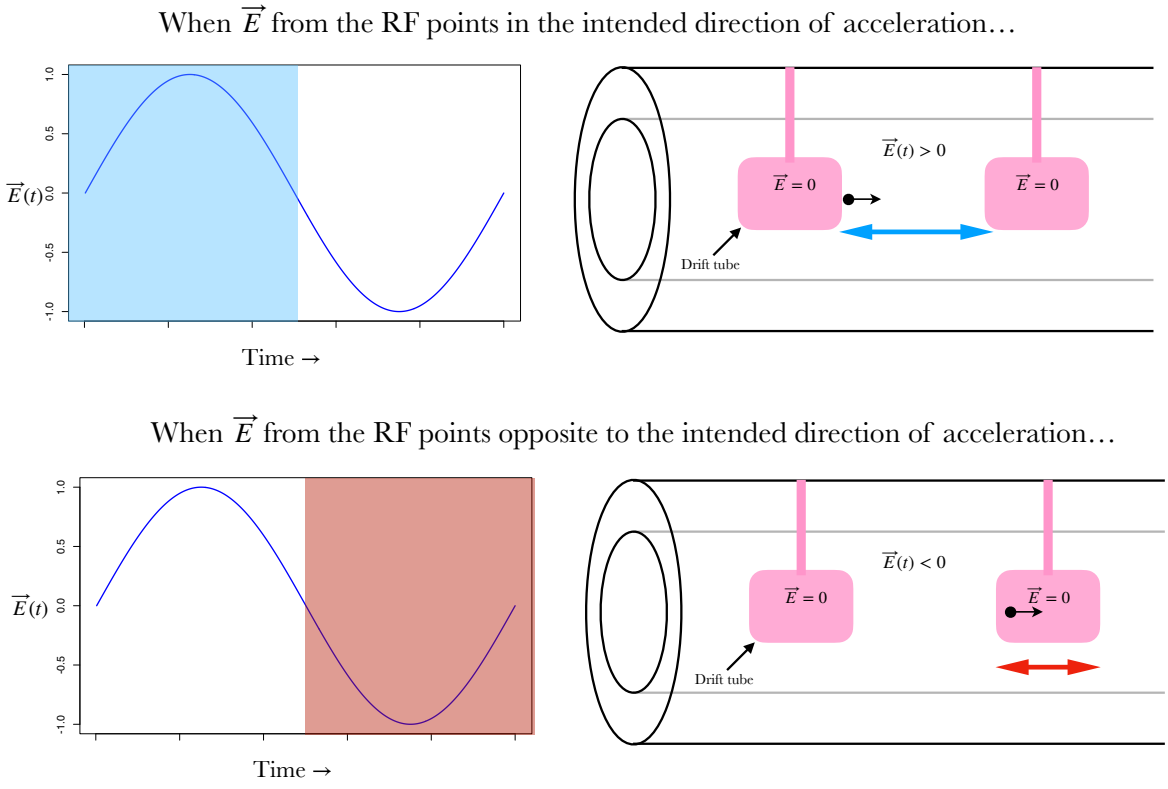

Figure 2.10: When $\vec{E}$ points in the direction of acceleration, we let the particle experience the field. But when the $\vec{E}$ flips the sign, the particle enters the drift tube which act as a Faraday shield, not letting any of the $\vec{E}$-field inside the tube. This way, the particle experiences only acceleration.

continue to circulate in the Booster). The Booster typically gets filled up with multiple-turn worth of protons since the pulse length from linac is more than the revolution period of the Booster at 400 MeV.

The RF system of the Booster is comprised of two groups, A and B stations. In order to not accelerate the protons during the injection process, the 2 RF systems are made to remain out of phase. Once all the H^- ions are injected, filling the Booster with protons, then the phases of both RF systems are brought in-phase, capturing the proton beam in RF buckets before acceleration. Since there are no RF buckets to keep intact during the

injection process, debunching occurs naturally in the Booster since the injected proton beam has a natural (albeit little) momentum spread.

Once the injection is completed, the protons are accelerated from 400 MeV to 8 GeV kinetic energy with the aid of 19 RF cavity stations. The Booster is a synchroton machine and frequency of the RF cavities is choreographically increased from 37.8 MHz to 52.8 MHz. This is so because as the velocity of the protons increases, their travel time between RF stations decreases and thus the RF cavity's frequency must also appropriately increase to accelerate the protons efficiently. As the proton beam's energy increases from 400 MeV to 8 GeV, its revolution period around the Booster decreases from 2.2 μs to 1.6 μs.

The reader might note that the RF cavity frequencies are faster than the corresponding revolution frequencies of the particles by a factor of 84. This factor is called the *harmonic number*, and this creates 84 bunches of protons inside the booster that are temporally separated from each other. At the end of acceleration when all the protons are at 8 GeV, all of the 84 bunches (called one *batch*) are extracted into the MI-8 line [4].

For *Mu2e* beam delivery, MI-8 beam line transports the 8 GeV proton beam to the the Recycler Ring (RR) or to the BNB beamline.

2.3.4 Recycler Ring/Main Injector

The Recycler Ring (RR) is the largest operational machine in Fermilab and is presently used, amongst other things, to deliver protons to the Muon Campus.

The Recycler Ring is an accelerator designed for 8 GeV protons with mostly permanent magnets. The RR receives one (or more) proton batches from the Booster, has the capability

[4]The actual process is a little more involved since Booster performs something called a bunch rotation, which we are ignoring for our purpose since we are interested only in the beam delivery to the Delivery Ring for resonant extraction.

Table 2.1: Booster Ring Parameters

Injection Kinetic Energy	400 MeV
Extraction Kinetic Energy	8 GeV
Repeating lattice period	24
Lattice structure	FOFDOOD
RF Stations	19
RF Injection Frequency	37.77 MHz
RF Extraction Frequency	52.8 MHz
Harmonic Number	84

to perform slip-stacking [5] and transfers the protons to MI (though there would be no slip-stacking for Mu2e beam operation).

The protons from the Booster could be bucketed into 82 bunches and sequentially injected all at once into the RR. Once the injection is complete, the RF system of RR is switched on, operating at a frequency of 2.5 MHz. This longitudinally restructures the 82 beam bunches grouping them into 4 big bunches of 10^{12} protons per bunch. After the rebunching is done (which takes about 90 ms), the 4 bunches are then sequentially sent to the Delivery Ring in the Muon Campus.

2.4 Muon Campus

The Muon Campus at Fermilab houses two experiments: the Muon *g-2* experiment and the *Mu2e* experiment. The Muon Campus primarily consists of target station for *g-2* , the Delivery Ring, *g-2* experiment hall, *Mu2e* experiment hall, and associated beam lines.

Both *g-2* and *Mu2e* use the Delivery Ring to deliver beam for the respective experiments (muon beam for *g-2* and proton beam for *Mu2e*). The time structure of the experiments

[5] Slip-stacking is a process by which two batches of protons are injected into RR to to double the beam intensity. Slip-stacking at RR is capable of stacking up to 12 batches of protons.

is similar but both the *g-2* and the *Mu2e* cannot operate concurrently since the beam requirements are different for both the experiments. However, the accelerator system can be operationally modified to enable beam production to either of the experiments.

The *g-2* experiment at Fermilab intends to measure the Magnetic Dipole Moment (MDM) of muons. This is done by storing muons of 3.1 GeV/c momentum (called the 'magic momentum') in the *g-2* storage ring and measuring the energy of the positrons decayed from the positive muons. The MDM makes the muon's spin precess about the almost pure vertical magnetic field in the storage ring, and the precession frequency is reflected in the energy spectrum of the positrons that muons decay into. By simply counting the number of muons above a particular energy threshold, the oscillation in the count gives us the precession frequency [6].

The muon production for the *g-2* experiment is achieved by 8 GeV proton beam hitting a production target to produce muons and pions (which also eventually primarily decay into muons). The decay products consists mostly of π^+, μ^+, and protons, and are sent to the Delivery Ring. After 4 turns, almost all the pions decay into muons and we are left only with protons and muons that get spatially separated [7]. The spatially separated protons are dumped by firing a kicker magnet and all the muons in the DR are sent to the *g-2* storage ring for the MDM spin precession measurement to be taken.

2.4.1 Beam Delivery for *Mu2e*

The beam delivery for *Mu2e* is different because the muon production target for *Mu2e* lies downstream of DR and not before. In *Mu2e* mode of operation, the 8 GeV proton beam

[6] The actual fitting function for the MDM spin precession frequency has many more parameters than this.

[7] This is so because the difference in the masses of protons and muons make them accelerate at different rates to the same magnitude of RF fields.

from the RR is made to bypass the target to produce for the Muon *g-2* experiment and is sent directly to the Delivery Ring.

The time structure for *Mu2e* beam delivery is as follows:

- The Recycler Ring (RR) is injected with 2 batches of protons from the Booster, with each batch consisting of 84 bunches (though only 82 are sent in reality).
- Once the injection from the Booster into RR is complete, the 2.5 MHz RF system in the RR turns on, rebunching and coalescing the circulating beam from 84×2 bunches to 4×2 bunches, resulting in 8 circulating bunches with 10^{12} protons per bunch.
- At $t = 90$ ms, the rebunching is completed and the first of the eight bunches (i.e., 1×10^{12} protons) is sent to the Delivery Ring.
- Once the 1×10^{12} protons are in the DR, they are resonantly extracted turn by turn over the course of 43.1 ms.
- The resonantly extracted bunch of protons is transported to the muon production target, and the muons off of the target are transported to the stopping target via a solenoidal magnet.
- In the DR, after the extraction time is complete, the system is reset for the next cycle of extraction during the 5 ms reset time. Any unextracted protons that may be left in the ring is aborted by firing a kicker magnet.

2.5 Proposed Third Integer Resonant Extraction in the DR

The resonant extraction at Delivery Ring is to be achieved by exciting the third integer resonance using dedicated harmonic sextupoles and by driving the horizontal tune closer to

third integer resonance using fast ramping quadrupoles. The spill duration is 43.1 ms and the number of turns over which the extraction occurs is approximately 25,430, extracting about $\approx 4 \times 10^7$ protons per spill. The extraction is to be achieved by keeping the sextupole strength constant but delicately squeezing the horizontal tune ν_x from 9.650 to 9.666.

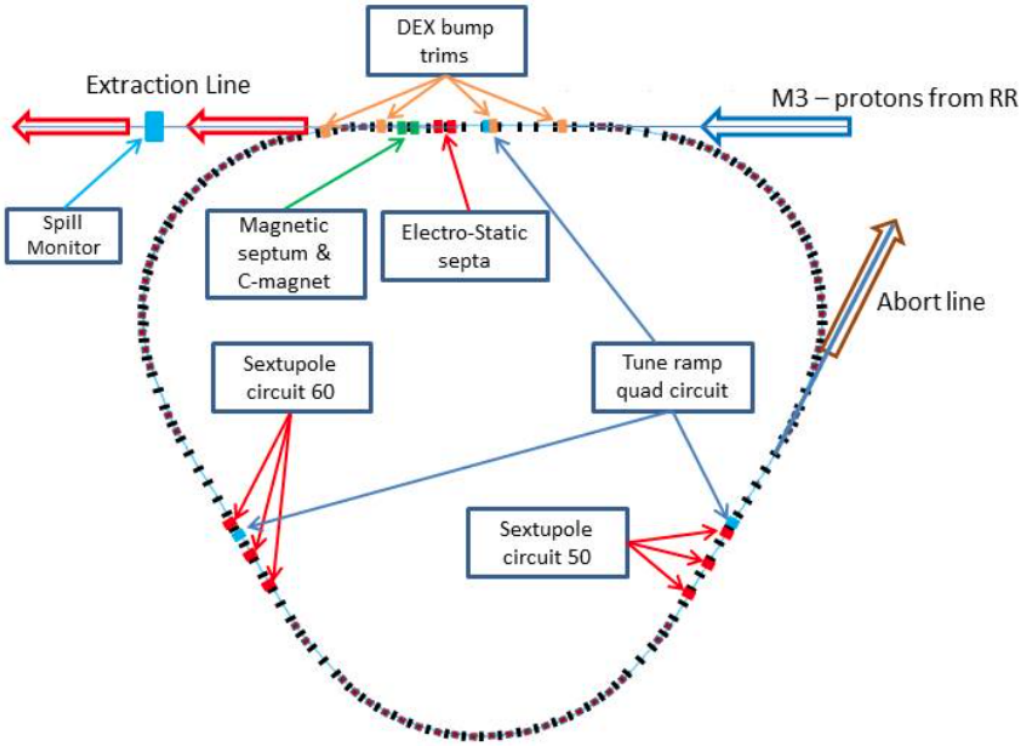

Figure 2.11: Elements in the DR that facilitate resonant extraction. [3]

There are two independently powered families of ISA-type harmonic sextupole magnets in the DR located in two adjacent straight sections. The sextupole magnet's operational gradient strength is to be 80 T/m^2. As shown in Fig. 2.11, the two families of sextupole magnets are located at the main focussing quadrupoles, so the 60° phase FODO cell makes a convenient summation for the third harmonic strength over these locations. The phase contribution of each circuit to the coupling constant differs by about 90°, also convenient for phase adjustments.

There are three fast CQA quadrupole magnets to squeeze the horizontal tune ν_x from 9.650 to 9.666. The fast quadrupoles are capable of a maximum dI/dt of 16,000 A/s with a regulation accuracy of 0.5 %.

There are also 4 Dynamic Extraction (DEX) bump magnets, two of them located before and two after the electrostatic septa. During the extraction process, the angle of the particles would keep varying as the size of the separatrix becomes smaller (shown in Fig. 2.12). This change of the angle of particles at the septum location can give rise to inefficiency in the extraction, potentially leading to elevated levels of radioctivity. In order to control the horizontal extraction angle and maintain high extraction efficiency, the DEX bump magnets are dynamically powered (Fig. 2.13). In this thesis work, we assume that the dynamic bump curve is given to us.

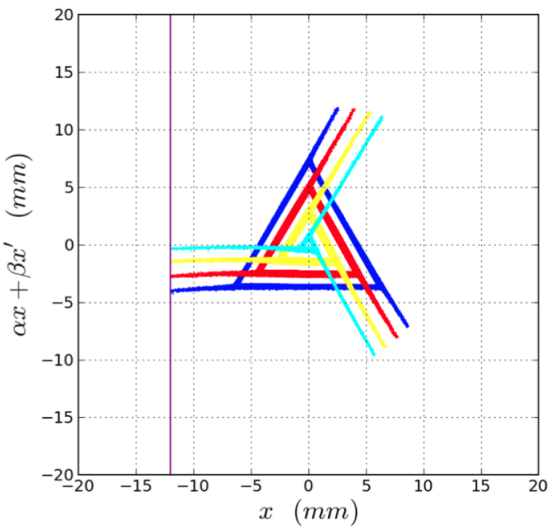

Figure 2.12: Evolution of separatrix with time at the extraction location. We see that the angle of the extracted particles are different at different parts of the spill.

An electrostatic septum (that consists of two parts) is placed at the extraction location. The septum provides high electrostatic field to give an instantaneous kick (in the horizontal direction) to the portion of the circulating beam that lies past the septum position in order to direct it to the extraction beam line.

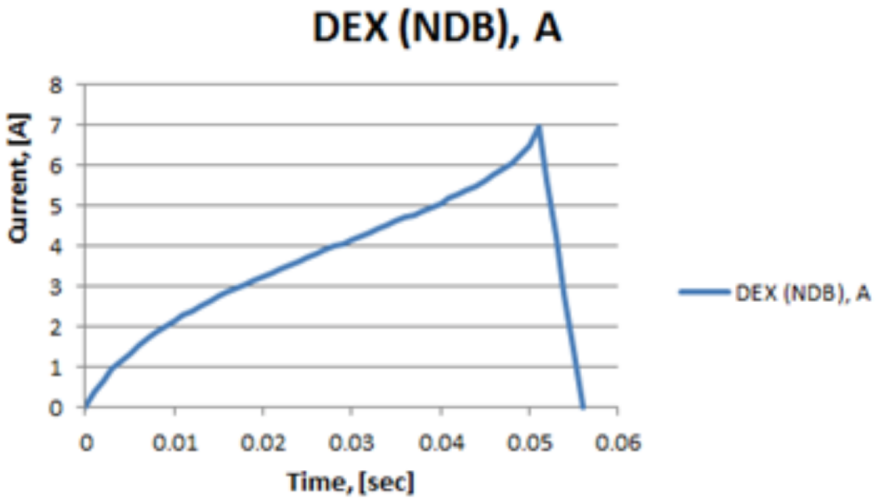

Figure 2.13: The dynamic extraction bump (DEX) magnets are powered dynamically during the spill to control the extraction angle at the electrostatic septa location in order to increase the extraction efficiency [3].

The septum is a complex device that was built from prior experience of operating electrostatic septa at Fermilab [27]. The high electrostatic field used to deflect the portion of circulating beam is provided by the high voltage between a foil plane and a titanium cathode. The foils are kept at zero voltage and the cathode is charged to give a stable voltage of -150 KV with a 15 mm gap between the foil plane and the cathode [3]. The foil thickness determines the *efficiency* of extraction, defined as the ratio of number of particles lost in the septa foil to number of extracted particles.

In our work, the septum location is crucial because that determines the extraction condition in the numerical tracking code. We assume the septum location, i.e., the placing of the foil plane, to be 12 mm from the beam centre. And the foil thickness is taken to be 0.1 mm.

Having now gone through the introduction to basics of accelerator physics, theory of resonant extraction, *Mu2e* and motivations for resonant extraction, and an overview of the Fermilab accelerator system, we are now well suited to delve into the design, model building,

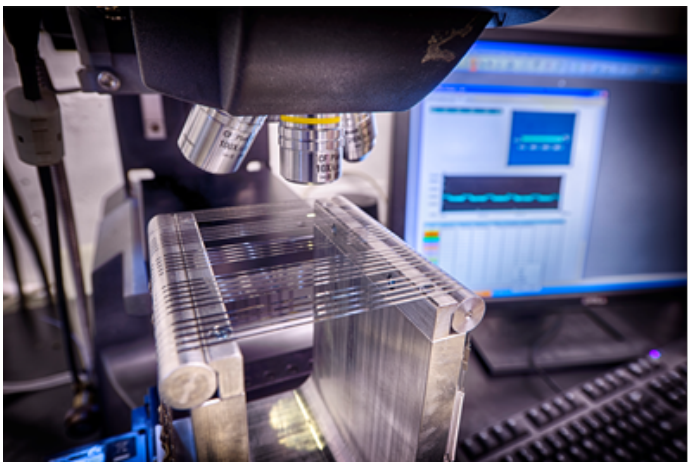

Figure 2.14: Picture of a prototype of the electrostatic septum foil plane [3].

and numerical validation of the slow regulation system and the fast regulation system in the following third and fourth chapters, respectively.

CHAPTER 3

SPILL REGULATION SYSTEM

This chapter contains an introduction to the Spill Regulation System for the resonant extraction in the Delivery Ring for the *Mu2e* experiment. The details of the numerical tracking code to validate various Slow Regulation algorithms are presented. The results of the investigations of various Slow Regulation Loop schemes are also presented. A functional overview of the Fast Regulation Loop (FRL), a brief introduction to RF-Knockout, as well as to the traditional approach to Fast Regulation using PID controller are discussed.

3.1 Spill Regulation System

3.1.1 Requirements

The *Mu2e* experiment imposes strict requirements on the uniformity of the pulsed muon beam delivery to the Aluminum stopping target. Since the muons are produced from the proton pulse hitting the muon production target, the intensity of the muons produced from the target depends directly on the intensity of the protons extracted turn-by-turn through resonant extraction to be performed at the DR.

This is primarily because drastic variations in the spill rate could adversely affect the signal of the *Mu2e* experiment. The Cosmic Ray Veto (CRV) system, for example, tasks itself in excluding events in which electrons from the cosmic ray could potentially mimic *Mu2e* electrons. The dead-time in detectors can be defined as the time the detectors does not take

any data between two consecutive events in order to minimize the background. In Figure. 3.1, we see that part of the (simulated) *Mu2e* Cosmic Ray Veto (CRV) detector system's dead-time increases non-linearly with the beam intensity [28]. This is primarily because the background from the target resulting from a higher than design intensity of protons-on-target could swamp the CRV system, resulting in more dead-time fraction, leading to a higher wastage of potentially interesting events. This, amongst other things, thus imposes a strict requirement on the quality of resonant extraction rate of protons from the Delivery Ring to the muon production target.

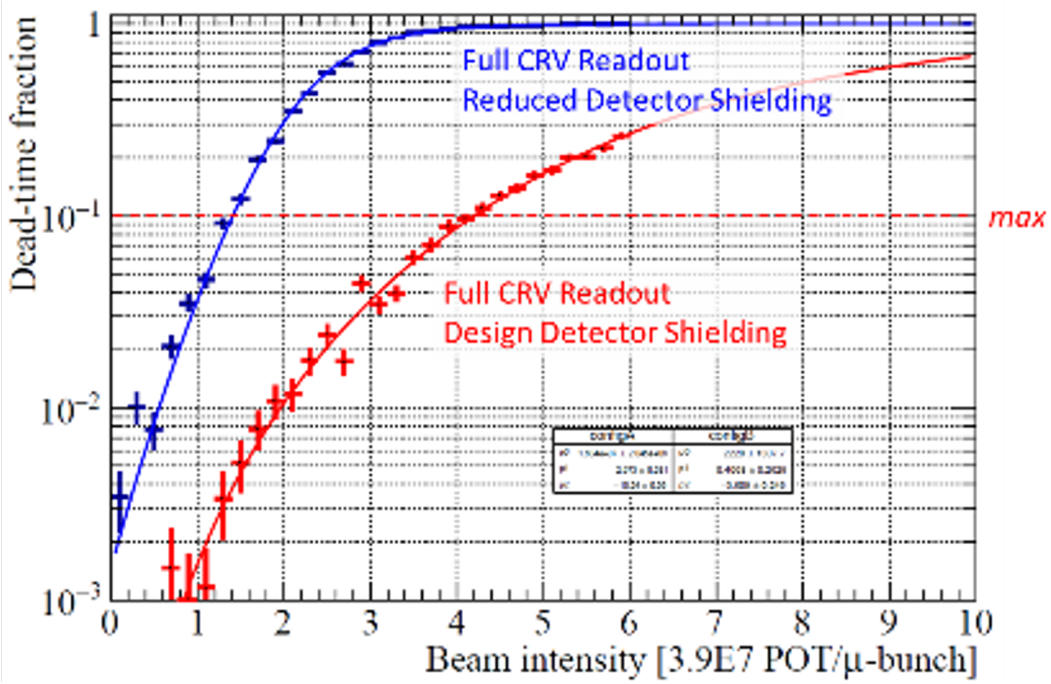

Figure 3.1: The dead-time of *Mu2e* detectors as a function of proton pulse density [28].

In this work, we primarily study the regulation of the spill rate resulting from either slow drift in the accelerator components and the random (or semi-random) fluctuations in the spill rate that could arise within one spill time. Some of the main beam parameters required for the resonant extraction at DR are given in Table 3.1. We also assume here that the

initial number of particles is the same for every spill, and this would qualitatively not be a problem for the purpose of validating Slow Regulation algorithms [1].

Table 3.1: Some main parameters pertinent to resonant extraction.

Parameter	Value
Beam kinetic energy	8 GeV
Spill Duration	43 ms
Number of spills per super cycle	8
Number of bunches per spill	1
Initial proton intensity	10^{12} protons
# of protons extracted per turn	$< 4 \times 10^7$ protons
Time between proton micropulses	1.695 μs
Normalized Emittance (95 % expectation) ϵ_x	16 π mm-mrad
Spill Duty Factor (SDF)	> 60 %
Reset time between spills	5 ms

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

$$\text{SDF} = \frac{1}{1 + \sigma^2_{\text{spill}}} \tag{3.1}$$

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 requirement of the Spill Regulation System (SRS) for *Mu2e* is to achieve an SDF value of 0.6 or higher.

[1] If the beam profile (especially the tail), of the initial injected beam into the DR varies from spill to spill, we may have to offset the starting point of the quadrupole current appropriately.

3.1.2 Functional Overview

The Spill Regulation System primarily consists of three components:

- Slow spill profile regulation
- Fast random ripple regulation
- Harmonic ripple content suppressor

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 three dedicated fast tune-ramping quadrupoles' 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.

A significant effort has also been put into investigating Machine Learning (ML) techniques, including supervised learning as well as reinforcement learning, to help improve the spill regulation system (to be discussed in Chapter 4).

The harmonic content suppressor tasks itself in suppressing and snuffing out the 60 hertz harmonics that may arise from the magnet power supplies and leak into the spill rate fluctuation.

The SRS has been designed to mitigate several sources of ripple in the spill profile. The SRS will regulate extraction through 2 primary elements: the Quad Regulation Circuit and the RF Knock-Out (RFKO) system. Each regulation element is controlled by separate but concurrent control loops.

3.1.3 Quadrupole Regulation Circuit

In our scheme of third integer resonant extraction, we remind the reader that we fix the sextupole field strength and drive the horizontal tune of the beam close to a third integer resonance. The way the horizontal tune is driven is by appropriately changing the three dedicated fast tune-ramping quadrupoles' strengths (the three of which are wired together).

Both the Slow Regulation Loop as well as the Fast Regulation Loop consists of regulating the fast quadrupole's power supply current ramp to regulate the variations in the spill rate. The way this is effective is because we know the horizontal tune of the beam changes proportional to the change in the strength of the quadrupole fields in the lattice:

$$\delta\nu = \frac{29}{3} - \delta Q = \frac{1}{4\pi} \oint \beta(s) K(s)\, \mathrm{d}s \tag{3.2}$$

where K is the quadrupole strength.

The reader may recall from equation (2.40) that the area of the stable region in phase space in a third-integer extraction is directly proportional to square of the tune distance of the beam to a third integer tune:

$$\text{Stable region area} = 48\sqrt{3}\pi \frac{(\delta Q)^2}{S^2} \tag{3.3}$$

If a uniform number of particles must be extracted every turn, then the area of the stable region, too, must be squeezed to zero at a particularly delicate rate every turn. Since the horizontal tune distance δQ is directly proportional to the quadrupole strength through equation (3.2), the rate at which the fast, dedicated tune-shifting quadrupoles are ramped would determine the rate of extraction of the particles.

Thus, for a given initial distribution of particles in the phase space, in the absence of any noises in the lattice elements, there exists *one* particular ramp profile for the power supply of the fast quadrupoles that would result in a more or less ideal extraction rate. However, *in the presence of noise* in the lattice elements, the extraction could vary in an unpredictable fashion. We thus use the power supply that is fed to the fast tune-shifting quadrupoles as a knob to control and regulate the variations in the spill rate.

The Quadrupole Regulation Circuit is used in both the Slow Regulation Loop as well as the Fast Regulation Loop, but in a qualitatively different fashion in each of the respective loops.

3.2 Slow Regulation Loop

Some of the main characteristics of the beam, such as its horizontal size, injection emittance, beam steering error, etc., could vary slowly over time due to many factors. Apart from this, a slow drift in the accelerator components could reflect in the spill intensity profile as a low frequency deviation from the ideal spill rate. This variation in the spill rate could be very slow and possibly span across multiple spill durations. A regulation loop is thus needed to keep the overall profile of the spill intensity as close to the ideal spill profile as possible, and this would be done using the Slow Regulation Loop (SRL).

The Slow Regulation Loop would not concern itself about any random fluctuations in the spill rate that could occur just within one spill, nor does it care about the high frequency noise components, either arising from the magnet power supplies or other wise, that could fluctuate the spill intensity within one spill.

In order to avoid correcting the random fluctuations in the spill rate that may occur just within one spill (but not repeat in the subsequent spills), the Slow Regulation Loop would work over *many* spills.

The exact number of spills to average the spill rate data over could vary depending on the actual operation of the spill. But the main goal of the Slow Regulation Loop is to 'zero-in' on the ideal power supply ramp of the tune-ramping quadrupoles that would result in an *ideal* spill rate (in the absence of any instantaneous fast noises in the accelerator system).

3.2.1 Numerical Simulation

Since the physical resonant extraction is yet to be commissioned in the Delivery Ring at Fermilab for *Mu2e* (at the time of writing this thesis), a numerical particle tracking simulation code was constructed by the author in order to simulate the physics of resonant extraction process and validate various schemes of the Slow Regulation Loop.

Even though the real beam lattice in the DR comprises of six harmonic dedicated sextupoles, the effects of only one sextupole is employed in the tracking simulation since a single virtual sextupole (of appropriately scaled strength) suffices to emulate the effect of 6 sextupoles, simulate the resonant extraction, and validate the Slow Regulation Loop algorithms.

The tracking code was written in **R** [29] using the aforementioned (Table 3.1) design beam parameters, and the way the numerical simulation code validates an SRL scheme is slightly involved.

The code primarily comprises of two components:

- Turn-by-turn particle tracking for one full spill
- Spill analyses and quad ramp curve modification after one full spill

In the first component, an initial sample distribution of particles is ingested by the code and the resonant extraction is simulated for one full spill time, turn-by-turn. All the particles' positions and angles are tracked every turn and the number of extracted particles on each turn are kept count of.

Once the full spill is simulated, the extraction data for every turn is exported and analyzed. Using the extraction data, the quadrupole current ramp curve for the next spill is computed using a SRL logic scheme. The simulation then proceeds to the next spill using the new quad current ramp curve. (It is to be noted that in real life, as mentioned earlier, the spill intensity data would be averaged over many spills. But since we do not introduce any random instantaneous noise in this particular numerical simulation for slow regulation, there is no need to take an average of spill rates in this simulation.)

3.2.2 Simulation Parameters and Details

The inputs to the tracking simulation are the number of particles: N_{part}, the total number of turns over which resonant extraction happens: N_{turns}, the total number of bins in one spill duration: N_{bin}, the initial tune-ramp curve $\nu_x(n)$, and Slow Regulation Loop algorithm parameters (to be discussed shortly).

The tracking code ingests an initial phase space distribution (X, X') for any number of particles and perform resonant extraction for any number of turns. For the purposes of this

study, an initial distribution was prepared with a mean value of 0 and a standard deviation (which would be determined with respect to specific design emittance values), given by:

$$\sigma_{\text{distribution}} = \sqrt{\beta_s \epsilon} \tag{3.4}$$

where $\sigma_{\text{disttribution}}$ is the standard deviation of the distribution, β_s is the Courant-Snyder beta function, and ϵ is the rms emittance value. Since the distribution is in the normalized phase space, the respective standard deviations of the distributions in X and X' are the same.

When the beam is injected into the Deliver Ring, it is expected to have a reasonable number of particles in the tail of the distribution. These particles could get extracted in the first few turns with a large splurge, and we would like to avoid this splurge in the simulations as it could take us into situations that could not be realistic in validating the slow regulation algorithms.

And so, N_{part} number of positions and (normalized) phase-space angles are generated, and we perform resonant extraction at a constant horizontal tune of $\nu_x = 9.650$ so that all the tail of the beam (also called the 'halo') is first extracted in the first few hundred turns (typically less than 300 turns). We then take this prepared input distribution and use it to validate various regulation algorithm schemes. A sample of such a prepared input can be seen in Figures 3.2 and 3.3.

The kinetic energy of the beam in the simulation is assumed to be 8 GeV, and the total energy is

$$E_{\text{tot}} = \text{KE} + m_{\text{proton}} = 8.938 \text{ GeV} \tag{3.5}$$

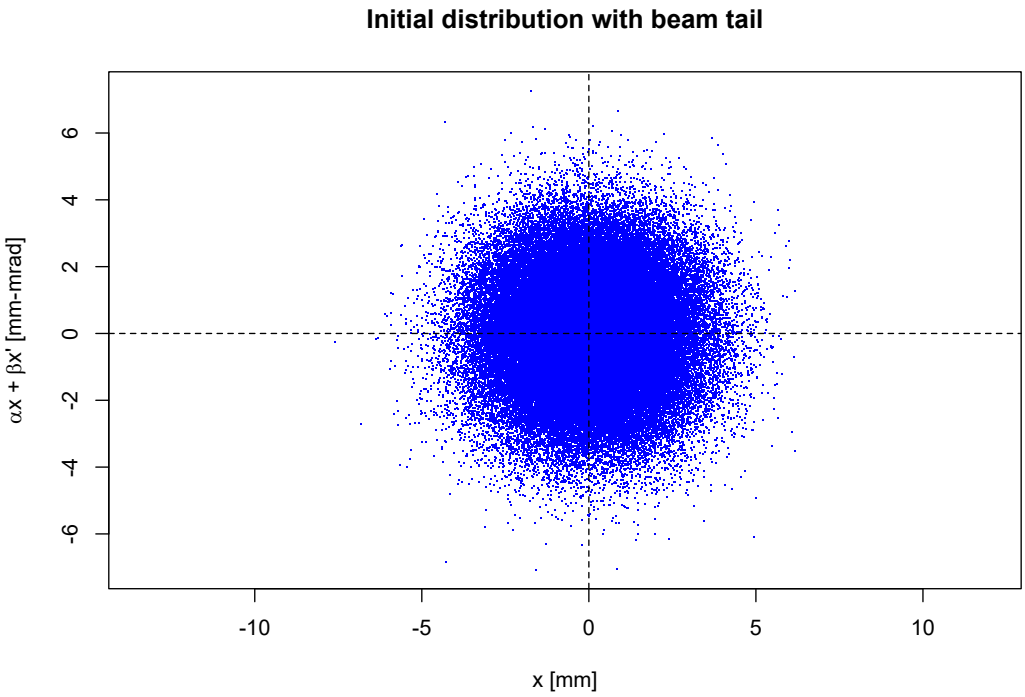

Figure 3.2: An initial Gaussian distribution with zero mean, $\sigma_X = \sigma_{X'} \approx 1.58$ mm, and $N_{\text{part}} = 10^5$ particles with beam tail.

The velocity of an on-momentum particle, v, is given by,

$$\gamma_{\text{rel.}} = \frac{E_{\text{tot}}}{m_{\text{proton}}} \tag{3.6}$$

$$\beta_{\text{rel.}} = \sqrt{1 - \frac{1}{\gamma_{\text{rel.}}^2}} \tag{3.7}$$

$$v = \beta_{\text{rel.}} c \tag{3.8}$$

The total number of turns N_{turns} is given by,

$$N_{\text{turns}} = \frac{T_{\text{spill}}}{T_{\text{revolution}}} \tag{3.9}$$

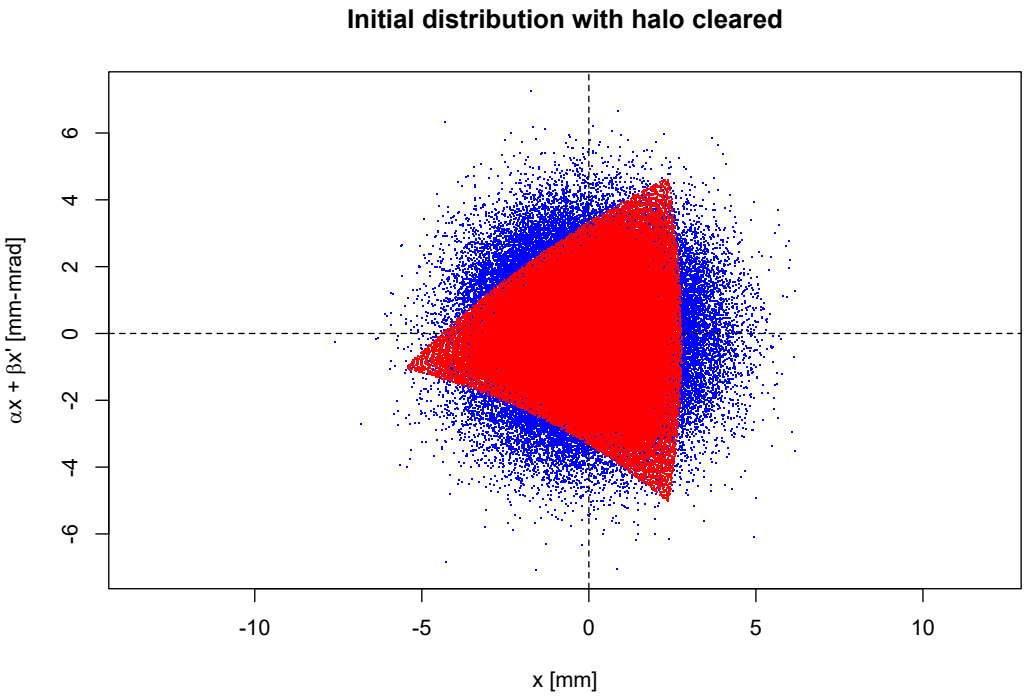

Figure 3.3: The red colored distribution is the same distribution in Figure 3.2 after clearing all the halo by performing resonant extraction at a constant tune of $\nu_x = 9.650$. (The 'halo' of the beam is defined here as any particle that gets extracted at a constant initial horizontal tune of 9.650).

where $T_{\text{revolution}}$ is given by,

$$T_{\text{revolution}} = \frac{\text{DR}_{\text{path length}}}{v} \tag{3.10}$$

The average path length around the DR was assumed to be approximately 505.2 meters, the total spill time T_{spill} to be 43 ms, and with a kinetic energy of 8 GeV for the protons, this gives us the total number of turns per spill time as $\approx$ 25,371 turns.

Since the maximum bandwidth of the Spill Regulation System is 10 KHz, we have 430 data points per spill time. In the simulation, this fact is reflected through the variable n_{bin},

whereby the total spill time is divided into n_{bin} number of bins. Even though the ideal n_{bin} is 430, simulations were tested with different n_{bin} values for computational purposes.

The most crucial input parameter to the simulation is the $\nu_x(n)$, the tune curve for one full spill. The goal for the Slow Regulation Loop, in essence, is to find the ideal tune curve $\nu(n)$ such that a uniform number of particles are extracted in every bin of the spill. But the initial and final values of the $\nu_x(n)$ are fixed to always be $\nu_{x,\text{ini.}} = 9.650$ and $\nu_{x,\text{final}} = 9.666$ since they are design parameters of the extraction system.

For the purpose of validating various Slow Regulation adaptive learning algorithms, we input a tune curve that is purely linear, running from $\nu_x = 9.650$ to 9.666 from the start to the end of one spill duration.

A linear tune ramp would not result in a uniform spill profile. This is because a linear decrease in tune would result in a linear decrease in the perimeter of the stable triangle region, and a non-linear decrease in the area of the stable triangle region. If we add to this the fact that the initial phase space distribution in (X, X') of particles will not be uniform but radially more dense towards the origin (typically Gaussian), the total number of particles that get into the unstable region (i.e., the particles ti be extracted) during the tune squeeze will not be the same with every time step if the tune ramp is linear.

To validate Slow Regulation algorithms, we thus start with a linear tune curve and see if the algorithm is able to zero-in on the ideal tune ramping curve from the linear one. We remind the reader that the tune-ramping curve is synonymous with the quad current ramp given the relation in equation (3.2).

(We note here that in the actual operation of the resonant extraction, we would most likely not be loading the quad power supply ramp with a linear ramp. We would rather be starting the extraction with a 'best-known-guess' power supply ramp curve which would not result in an extraction rate that is as poor as a linear ramp.)

To keep track of the particles' coordinates in the phase space, the simulation creates two arrays, one for X and another for X'. In addition to these, the simulation creates a 2D array, one dimension to keep track of the particle's identity and the other to keep track of the number of turns traversed by the particle. The count of this 2D array for the whole of the simulation would thus be $N_{\text{part}} \times N_{\text{turns}}$.

The tracking simulation assumes a single sextupole at the proper location with respect to the septum. We use a single 'virtual' sextupole strength 500 T/m^2 to simulate the effect of the six dedicated harmonic sextupoles in the real lattice of the Delivery Ring to excite the third integer resonance.

The electrostatic septum foil plane is assumed to be located 12 mm from the beam center. Any particle whose X value goes higher than this value will not circulate further in the beam lattice. The electrostatic septum foil at the extraction location is assumed to have a thickness of 100 microns. The width of the septum foil could prove crucial because the simulation is designed to keep track of not just how many particles are extracted but also how many are lost in the septum foil. This is important because the safety regulations require that the efficiency of extraction (defined as the ratio of extracted particles to particles lost in the septum) needs to be 2% or less; this is to keep the radioactivity levels in check in the septa foils. Having the ability to compute the extraction efficiency by the tracking code could prove useful in the future if the efficiency happens to be a parameter the regulation system needs to keep an eye on.

The simulation also assumes an admittance value of 40π mm-mrad for the lattice; any particle that strays outside this circle of radius 40π is counted as lost and not extracted.

The code assumes a simplified model of the Delivery Ring, where by it is divided into two parts. The first part would be consist the segment between the injection point of the particles and the sextupole. The second segment consists of propagating the particle from

the sextupole to the injection point. In this model, we have the extraction point at the injection location.

At a given turn, the initial coordinates of the particle is made into a 2×1 column matrix and is then multiplied with a rotation matrix. This transfer matrix would be the approximate net effect of all the transfer matrices to simulate the effect of the particle going through the linear beam line elements (dipoles and quadrupoles).

After the beam traverses through these linear elements, we then apply the non-linear effect of the sextupole with strength S_0. (This assumes a thin lens approximation of the sextupole whereby the position of the particle does not change after it goes through the sextupole.)

After the first turn, the i-th particle then proceeds to the second turn, whereby the coordinates of the particle from the end of the previous turn is stored and fed as the initial position for the next turn.

After every turn, the simulation code checks for the following conditions:

- If the particle's position X is greater than or equal to the septum location 12 mm.
- If the particle's position X is in between the septum location X_{sept} 12 mm and the edge of the septum thickness position $X_{\text{sept}} + \text{sept.}_{\text{thickness}} = 12.1$ mm.
- If the particle's $\sqrt{X^2 + X'^2}$ is greater than the admittance value of 40π mm-mrad.

If the particle satisfies the first condition, it is counted as 'extracted'. If either the second or the third condition is satisfied, then the particle is counted as 'lost'. Once the particle is either extracted or lost, they are flagged and no longer participate in the resonant extraction process.

3.3 High Performance Computing for Particle Tracking

The numerical simulation of resonant extraction through particle tracking can be very computationally expensive. The author thus had to take recourse to using the High Performance Computing (HPC) resources at Northern Illinois University (NIU).

The main chunk of the simulation efforts for the Slow Regulation investigations were carried out at NIU's Center for Research Computing and Data with their main high performance computing cluster Gaea, which is a hybrid CPU/GPU cluster with a peak computing performance of more than 30 teraflops.

Gaea has 60 nodes, with each of the node containing 12 cores. Each computing node is an HP SL380s G7 equipped with [30]:

- 2 × Intel X5650 2.66 GHz 6-core processors
- 72 GB RAM, 4 x 500 GB 2.5" SATA disk drives in RAID10 configuration (i.e., 1 TB each node)
- 1 x NVIDIA TESLA P4 GPU, Pascal architecture, 8GB RAM
- All 60 nodes are connected via Full 1:1 non-blocking Infiniband and Ethernet switch connectors.

A PBS (Portable Bash Script) was written by the author in order to run the numerical tracking simulation in the Gaea the cluster. The script typically would call for two separate **R** scripts.

The first script performs the aforementioned particle tracking of all the particles for one full spill and exports the spill intensity data. The second **R** script ingests the spill intensity data from the numerical tracking code to perform the Slow Regulation and output a new

quad ramp curve for the next spill. This is done iteratively to check the efficiency of a given Slow Regulation algorithm.

The numerical simulation for one spill for about 132,000 particles typically takes about 20 minutes when the workload is distributed over 6 nodes (which is equivalent to 60 cores). For the Slow Regulation validation, it typically takes about 500 spills to fully verify its validity. The author performed and tested out about fifty algorithms with different schemes and parameter values.

The reader might recognize that the total computation time for trying ≈ 50 such algorithms would amount to approximately 20 minutes $\times$ 500 spills $\times$ 50 algorithms, which is approximately 347 days. The reason why it did not take a year (but a few months) for this effort is because the PBS script was written in a fashion that the spill intensity is outputted and made accessible while the code is still live and running at the cluster. This way, the validity of a Slow Regulation algorithm would be known in a hundred spills or so.

During the extraction process, the particles near the separatrix experience the nonlinearity induced by the sextupole fields more than the particles near the core of the distribution. This makes the simulation of *all* of the particles unnecessary since particle tracking is already very computationally expensive.

Thus the core of the beam, especially near the center that is far away from the separatrix, is removed from the initial distribution for a given tune squeeze (shown in Figure 3.4). And as the separatrix shrinks, the phase space inside the triangle, approximately 60% of the circumcircle of the separatrix triangle, is populated with the beam so they can start feeling the effects of non-linearity so the simulation is accurate. However, once an algorithm proves efficient, we validate it by using the full distribution without any core removed to make sure all the particles experience all the non-linear effects throughout the extraction process.

We note that for comparisons of different Slow Regulation algorithms, we use the same input sample distribution to stay consistent. Even though, in reality, the *exact* phase space

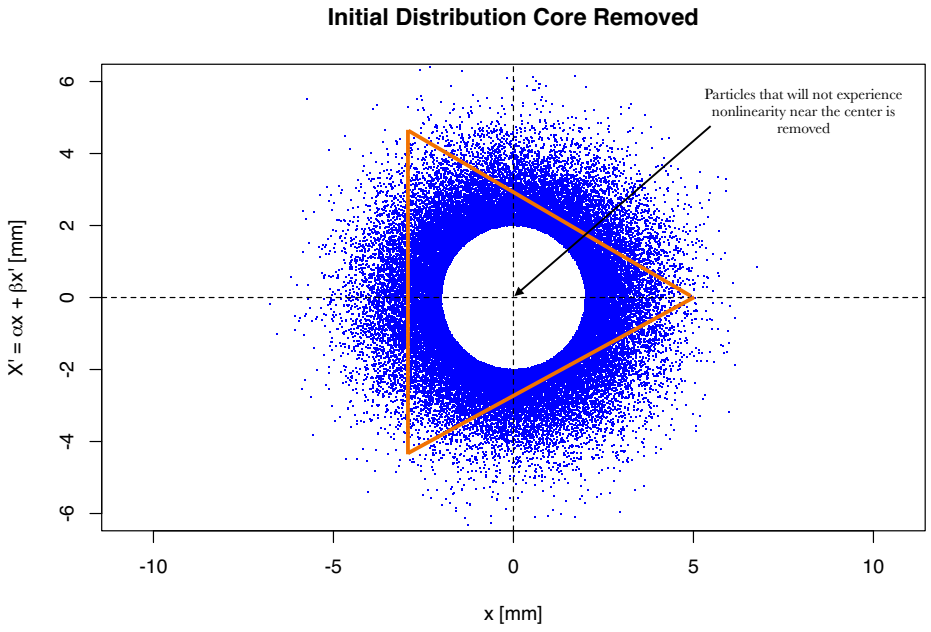

Figure 3.4: To be computationally efficient, the core of the beam near the center is removed as they don't participate in the extraction. But as the triangular separatrix area closes to zero, the core is dynamically populated in advance so that the resonant extraction is realistic.

distribution of the particles injected into DR could vary spill-to-spill, the statistics of the initial distribution to validate the Slow Regulation algorithm can be reasonably expected to wash out any significant differences between different possible distributions.

Since we do not simulate the effects of space charge, the particles can be tracked independent of each other. The way this is done at the Gaea cluster is the following (Figure 3.5):

- Two initial samples consisting a total of N_{part} number of particles is prepared, one for X and another for X'.
- The PBS script makes a request from the Gaea cluster for N_{node} number of nodes, with each node consisting of 10 active cores (one core can be though of as one CPU).

- Once the nodes (and cores) are designated for the job, the particle tracking **R** script is called by the PBS script.
- The particle tracking script divides the grand total number of particles N_{part} into N_{cores} number of cores, this way the particle tracking simulation can be done in each of the cores in parallel.
- The code runs the tracking particle-by-particle, with each particle traversing either the full N_{turn} number of turns or until it gets extracted (or gets lost in the septum or lost past the admittance value).
- Once one full spill is completed by each of the cores, they export the spill intensity data, i.e., the number of turns each particle took to get extracted.
- The second **R** script is now called and the spill data exported by all the cores is integrated into total spill data.
- Using the total spill data, the second **R** script executes the learning algorithm and gives out a new tune curve to be ingested by the particle tracking code for the next following spill.

This is done iteratively until the performance of the slow regulation scheme in question is validated. The details of the slow regulation algorithm are given in the following section.

3.4 Slow Regulation Algorithms

With the tracking code set up, we are now in a position to validate various slow regulation algorithms. This would done by the second **R** script where the algorithm parameters are specified.

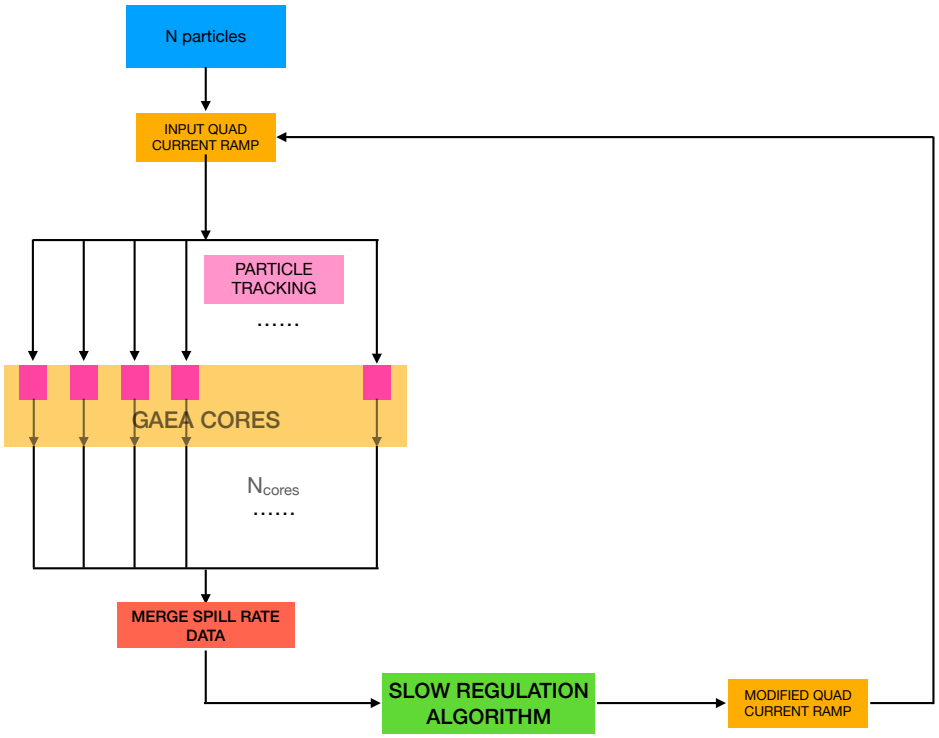

Figure 3.5: Data flow of numerical simulation of particle tracking distributed over Gaea cluster cores.

3.4.1 Slow Regulation Validation Schemes

The main idea behind the Slow Regulation algorithms is to find the ideal quad current ramping profile so that the extraction is as close to the ideal extraction rate in the absence of any noises in the beam lattice elements.

The regulation algorithm ingests the spill data and looks at how many particles are extracted at every bin. Since the algorithm knows the total number of particles in the initial

distribution (that is cleared of the halo), it computes the ideal number of particle that ought to be extracted every turn:

$$N_{\text{ideal}} = \frac{N_{\text{part. tot.}}}{N_{\text{turns tot.}}} \tag{3.11}$$

Since the design bandwidth of the system is not going to be at the revolution frequency around the DR, the spill intensity has to be integrated for a finite number of turns in order to make any evaluation or changes to the tune-ramping quadrupole's power supply.

In the initial efforts of the simulations, the data rate was taken to be at every 500 turns for a total of 27,000 turns. This implies that there would be a total of 54 bins (i.e., 27,000 turns/ 500 turns), which is about a tenth of the maximum design bandwidth of 10 KHz. It is important to note we assume here that the bandwidth of the Spill Regulation System and the bandwidth of the fast tune ramping quadrupoles are high enough that the regulation process is feasible.

3.4.2 Adaptive Learning

A convenient structure for the practical functionality of the Slow Regulation would be to simply load the algorithm into the system and let it adaptively learn the quad current power supply by itself over many number of spills.

Once the spill intensity data is obtained from the tracking simulation after one spill, the slow regulation algorithm analyzes each of the bins to see if *higher* number of particles have been extracted or *fewer*.

If the number of extracted particles per bin is higher, then it implies that the separatrix has been shrunk at a *faster* rate that it ought to have been for that bin, or vice-versa. This suggests that the quad current value in the next spill for that particular bin value must be

lower, or vice-versa. The slow regulation algorithms thus modifies the quad ramp profile from spill-to-spill in order to make the spill rate converge towards the ideal spill rate.

3.4.2.1 Algorithm 1

Let j denote the bin number of the spill, let the horizontal tune of the particle be $\nu[j]$ for a given bin j, and let $n[j]$ denote the number of particles extracted at j-th bin. Since it is almost impossible for to achieve a *perfect* extraction rate, we set an error tolerance in the extraction rate of $\pm 5\%$ of N_{ideal}.

$$N_{\text{ideal}} = \frac{N_{\text{tot}}}{N_{\text{tot. turns}}} \times N_{\text{turns per bin}} \tag{3.12}$$

If $n[j] > N_{\text{ideal}} \times (1 + 5\%)$, then we *decrease* the tune-ramping quad current value $I[j]$ for the j-th turn to,

$$I[j] = I[j] - kI[j] \tag{3.13}$$

$$k = \text{constant} \tag{3.14}$$

If $n[j] < N_{\text{ideal}} \times (1 - 5\%)$, then we *increase* the tune-ramping quad current value $I[j]$ for the j-th turn to,

$$I[j] = I[j] + kI[j] \tag{3.15}$$

$$k = \text{constant} \tag{3.16}$$

And if $n[j] > N_{\text{ideal}} \times (1-5\%)$ and $n[j] < N_{\text{ideal}} \times (1+5\%)$, we then leave the tune-ramping quad current value $I[j]$ for the j-th turn untouched,

$$I[j] = I[j] \tag{3.17}$$

Here, k is a proportionality factor and is a parameter of the algorithm that must be set before hand. The initial simulation was tested out with a fixed k value of 3% and with an input quad current ramp profile being linear, running from 0 to 80 A in 43 ms (or, equivalently, the horizontal tune value ν_x running from 9.650 to 9.666 in 43 ms).

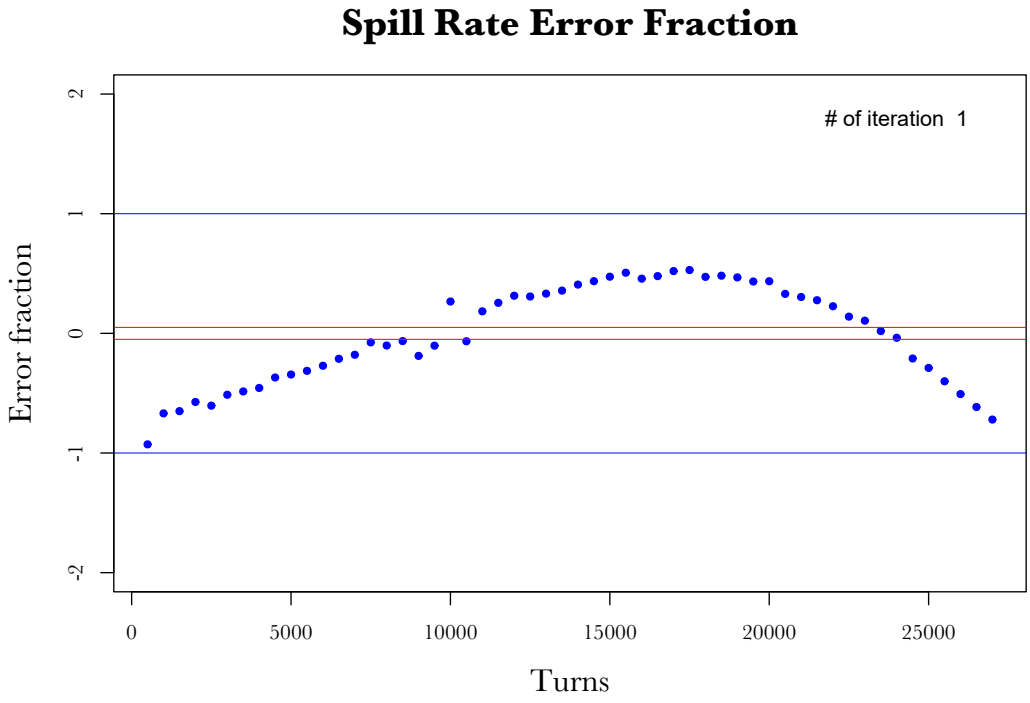

Figure 3.6: The spill rate error fraction for the first iteration of the slow regulation algorithm for a linear tune-shifting quad current ramp profile give in Figure 3.7. The spill rate error fraction is defined as $(N_{\text{spill}} - N_{\text{ideal}})/N_{\text{ideal}}$. Thus an error fraction of -1 corresponds to no particle getting extracted.

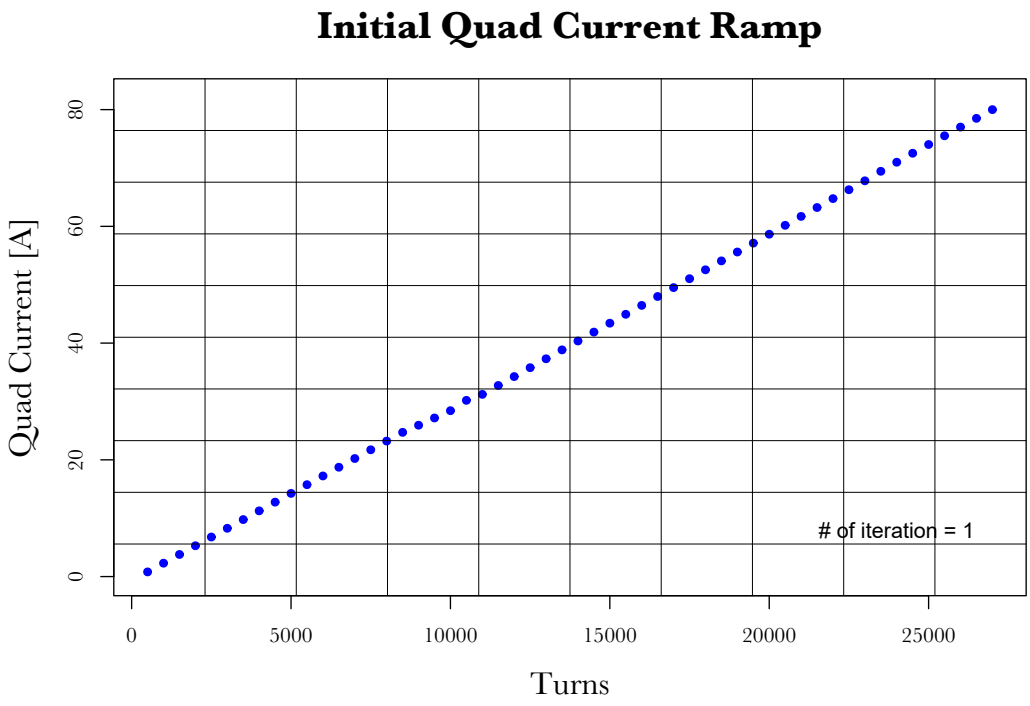

Figure 3.7: This would be the initial quad current ramp we provide the slow regulation algorithm with, we see the corresponding extraction rate for this current ramp profile in Figure 3.6.

In the first spill, we see in Figure 3.6 that the spill rate is not uniform at all because the quad current ramp curve is linear.

As the algorithm proceeds further down the iterations (i.e., spills), we see in Figure 3.8, for instance, in iteration #13 that the very first bin's error rate has fallen in place of the expected spill error fraction of $\pm 5\%$.

However, as the iteration proceeds, we see that this algorithm does not converge on the extraction rate but exhibit an oscillatory behavior in the extraction rate.

By oscillatory behavior, what we mean is that the amplitude of the spill error fraction swaps between +(Error fraction) and -(Error fraction) as shown in Figure 3.10. (A sample oscillation in error fraction illustrated in the Figure 3.10 is between the 100th and 101th iterations.)

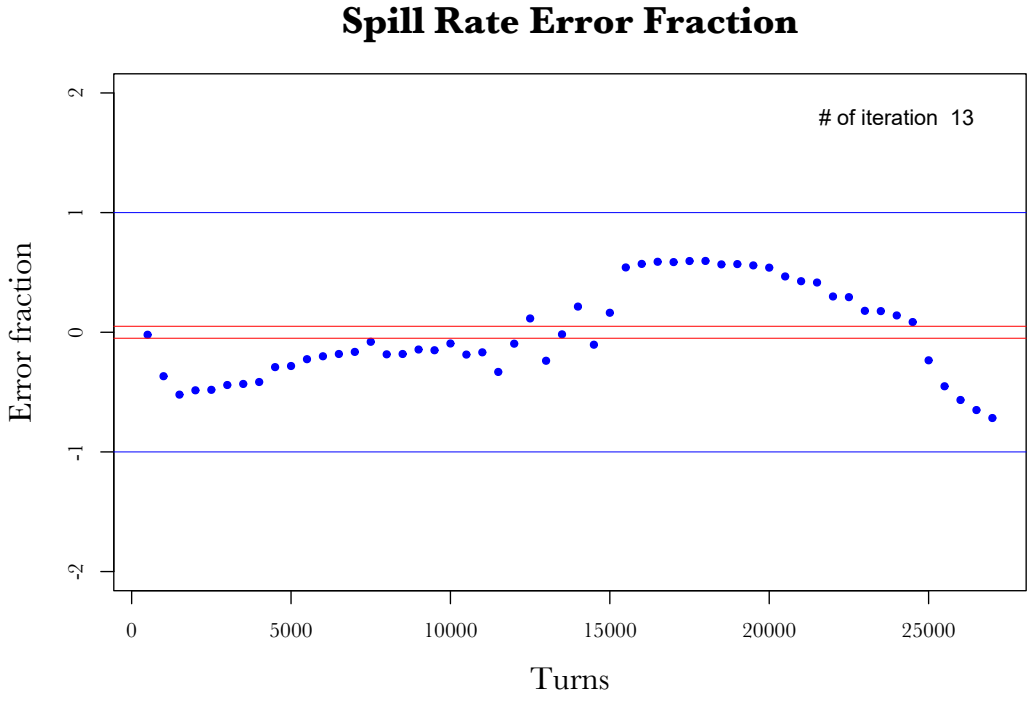

Figure 3.8: At iteration number 13, we see that the algorithm has made the error in extraction rate fall into the expected tolerance of $\pm 5\%$.

The reason why the algorithm exhibits this oscillatory behavior is because the amplitude of change in the quad current value is more than it ought to be. One way to imagine this would be the following: let us suppose at a given j-th bin of turns, more than the ideal number of particles were extracted. This implies that the *squeezing* rate of the stable region in the phase space has been more than it ought to have been. Thus, for the next spill, this rate of squeeze (for this particular bin) is modified such that in the next spill the rate of squeeze is not as high. But, if the rate of squeeze is decreased drastically, this would result in a big *negative* extraction rate, i.e., very fewer number of particles would get extracted. And because of the negative extraction rate, the quad current $I[j+1]$ would again be made higher than $I_{\text{ideal}}[n]$, resulting in a big +(Error fraction). And this goes on to produce the oscillatory behavior.

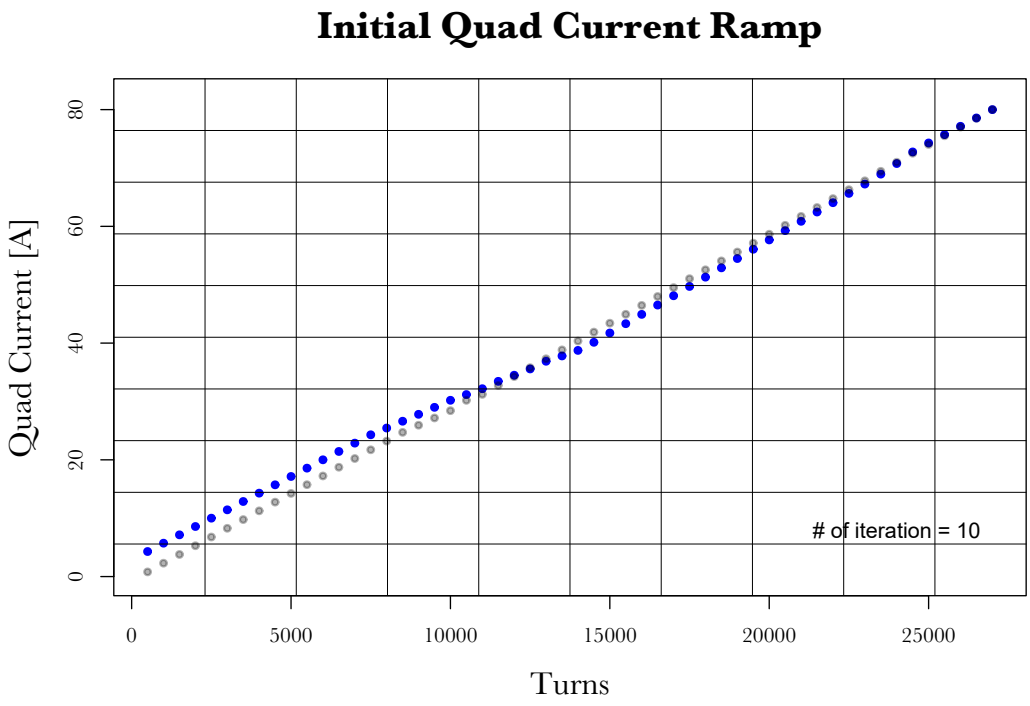

Figure 3.9: We see the quad current ramp has started to deviate from the initial (inefficient) quad current ramp.

This tells us that the proportionality coefficient k that determines this change in the rate of squeeze has to be low enough that the algorithm stands a chance to converge (faster) towards the ideal extraction rate.

3.4.2.2 Algorithm 2

Since having a high k value does not result in convergence, various values of k from 3% to 0.05% were investigated. We also increase the number of bins from 54 bins to 108 bins (this corresponds to integrating the extracted particles every 250 turns instead of 500 turns).

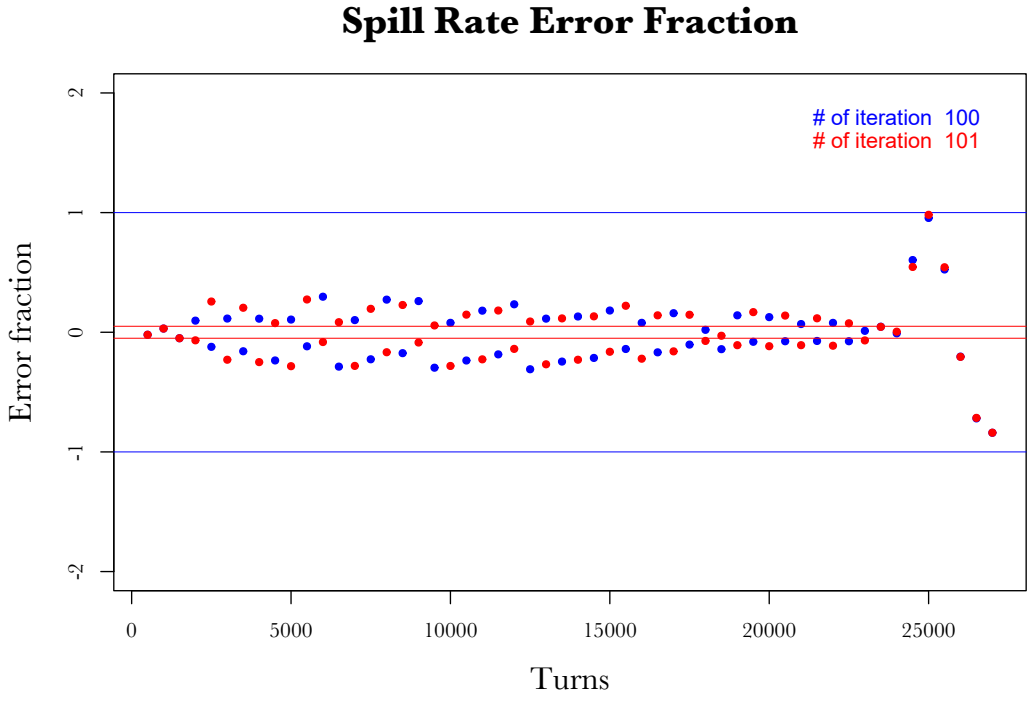

Figure 3.10: After hitting a certain number of iterations, say 100, Algorithm 1 starts exhibiting oscillatory behavior.

And upon decreasing the k value, we observe that the same initial distribution of particles (and the same initial current ramp) is now regulated well, giving us a convergence towards the ideal extraction rate.

Even though this algorithm does successfully regulate, there is a whiplash effect in the extraction rate that makes it go very high in the later parts of the spill when the initial part of the spill is getting closer to the ideal spill rate. It is to be noted that this could be a facet of the linear ramp curve inputted and it may be the case that if an analytically computed curve is what we begin with there would perhaps not be such an effect. However, to be on the safer side, we consider another algorithm that could help attenuate this whiplash effect.

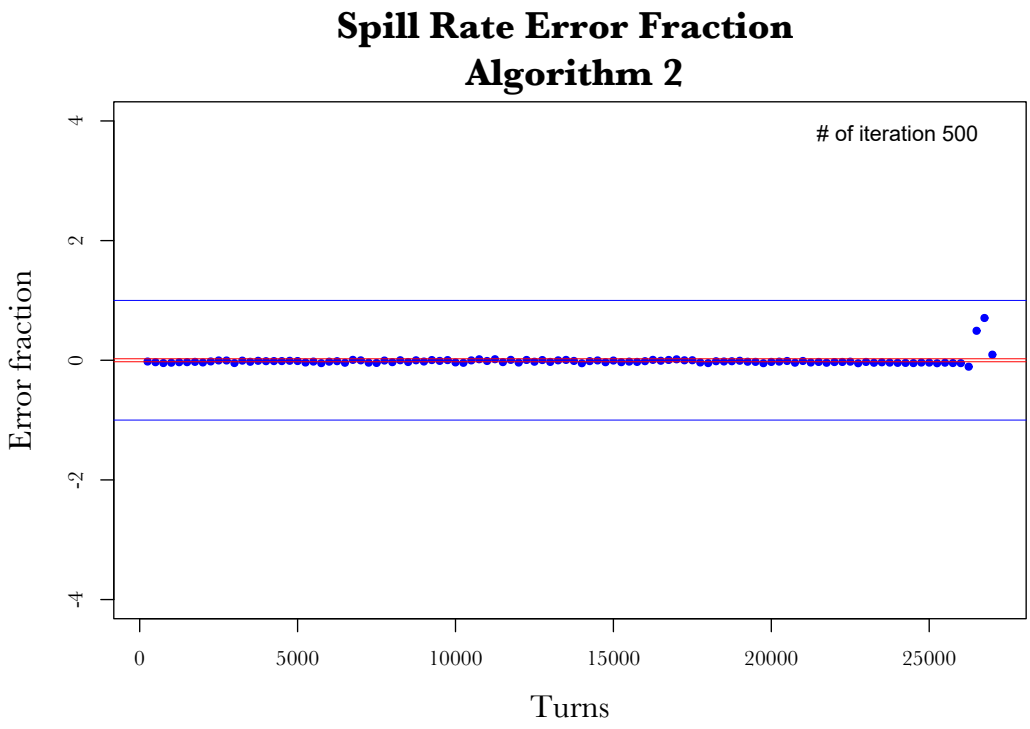

Figure 3.11: We see that Algorithm 2 is able to converge towards the ideal extraction rate. However, in the process of convergence, this algorithm is plagued by the whiplash effect shown in Figure 3.12.

3.4.2.3 Algorithm 3

Since we see the whiplash effect in Algorithm 2, we now consider making the gain coefficient not a constant but a function of the bin number. This could prove effective because while the particles in the earlier bins are getting 'locked' into the ideal extraction zone, the particles at later bins take the brunt of a large change in the $I[j]$. If instead we made the gain coefficient a decreasing function of the bin number, we can avoid changing the $I[j]$ values by too much for larger bin values whilst the $I[j]$ for earlier bins could still converge on the extraction rate.

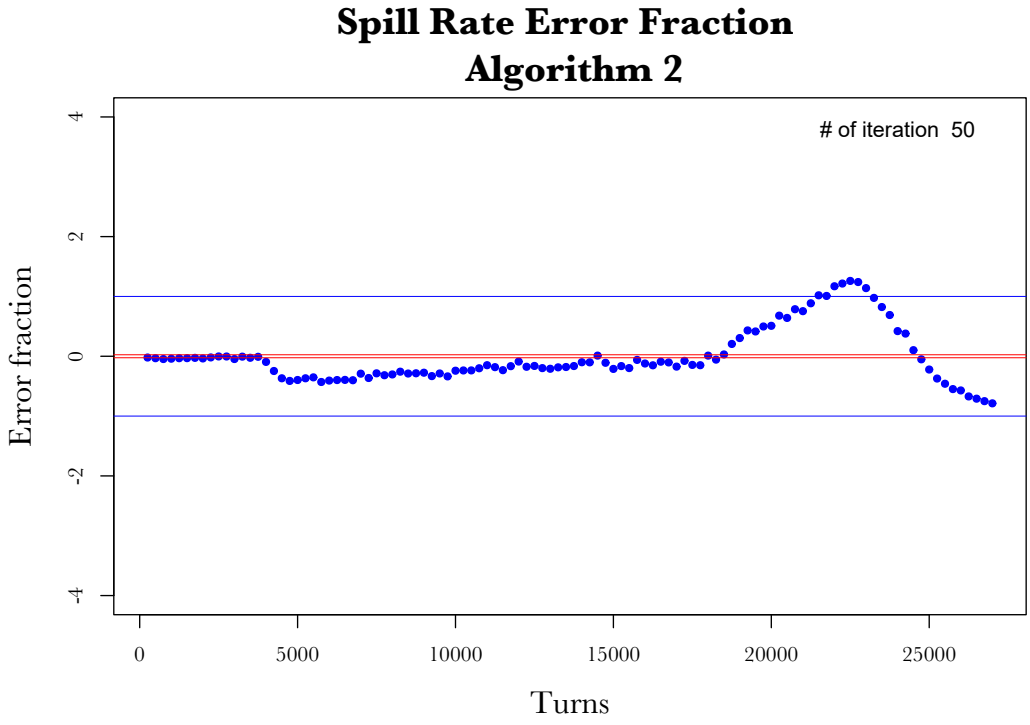

Figure 3.12: In the process of convergence, we see here the surge in the extraction rate in the later part of the spill while the initial spill rate is regulated.

In this algorithm, we change the quad current of a specific bin not by a constant amount but one that is proportional to the error (multiplied with a coefficient G that itself could be a function of the bin number):

- If $n[j] > N_{\text{ideal}} \times (1 + 5\%)$, then we *decrease* the tune-ramping quad current value $I[j]$ for the j-th turn to,

$$I[j] = I[j] + kI[j] \tag{3.18}$$

$$k = 0.05\% + G \times (n[j] - N_{\text{ideal}}) \tag{3.19}$$

- If $n[j] < N_{\text{ideal}} \times (1 - 5\%)$, then we *increase* the tune-ramping quad current value $I[j]$ for the j-th turn to,

$$I[j] = I[j] - kI[j] \tag{3.20}$$

$$k = 0.05\% + G \times (n[j] - N_{\text{ideal}}) \tag{3.21}$$

- And if $n[j] > N_{\text{ideal}} \times (1 - 5\%)$ `and` $n[j] < N_{\text{ideal}} \times (1 + 5\%)$, we then leave the tune-ramping quad current value $I[j]$ for the j-th turn untouched,

$$I[j] = I[j] \tag{3.22}$$

where G is a function of the bin number that linearly decreases from 0.002 to 0.001.

3.5 Slow Regulation Algorithms Results

Of the three algorithms mentioned[2], we report here that Algorithm 3 demonstrates the best performance in Slow Regulation.

We see in Figure 3.13 that the error in extraction rate has converged almost perfectly to the required spill error fraction with ±5% tolerance.

The algorithm was able to adaptively learn and successfully converge to the ideal quad current power supply ramp all by itself. This is despite the algorithm starting with a relatively worse initial quad current ramp. And the algorithm was able to achieve this in a little over 500 spills. If the duration for one spill is 43 ms, then this spills corresponds to at most a few minutes time.

[2]Various other algorithms have also been tried, but here we have selected three that has some of the interesting and useful results.

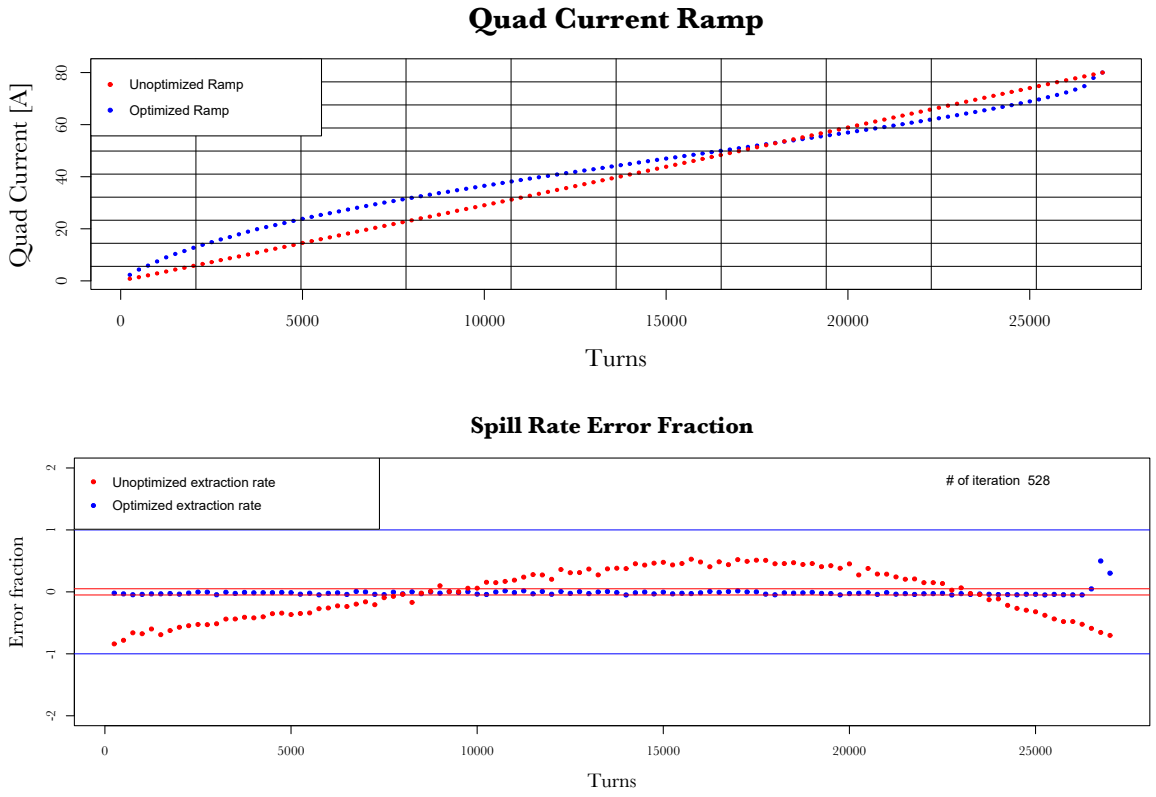

Figure 3.13: Algorithm 3 successfully regulating the spill rate within ±5% accuracy. In both the graphs, the initial iteration and final iteration's plots are shown. We see that the algorithm was able to adaptively learn to make the extraction smooth in just over 500 spills.

When the resonant extraction would be commissioned, we would ideally not start with any random quad current ramp but with a best-guess ramp which is expected to result in a reasonable spill rate (one way of calculating the quad current curve analytically is to be derived in Chapter IV). We remind the reader that this algorithm pre-supposes a steady state condition and is supposed to regulate any *slow* drift in the beam lattice parameters. But since this algorithm is able to find the optimal quad current ramp from even a very sub-optimal extraction rate, makes it all the more efficient.

We note here that the reader might notice that the last two or three data points in the spill rate does not quite reach the level of the expected error tolerance. The reason for this

very slow convergence in the end is because the area of the separatrix is becoming orders of magnitude smaller, and it becomes too tricky for the algorithm to find the precise rate of squeeze of the stable region. Another *possible* reason for the non-convergence could be because of the inherent error tolerance that we provided in the extraction rate. There could be an asymmetry between positive and negative extraction rate across all the bins, and the algorithm may not be left with enough number of particles at the end of the spill to make adjustments in the quad current ramp.

3.5.1 Robustness of the Slow Regulation Algorithm

We saw that the Slow Regulation algorithm mentioned in the previous section works for a beam that is prepared and is on-centre and on-momentum, and so on. In this section we slightly let go of the ideal beam conditions and check if the algorithms is able to still find the ideal quad current ramp.

3.5.1.1 Steering error

If there is a steering error in the beam due to magnet misfiring or any other factor, the beam could be displaced by a finite amount from the beam center.

In order to check the robustness of the Slow Regulation algorithm, we now give an explicit horizontal offset of 5 mm to all the particles and perform resonant extraction to see if the algorithm is able to find the ideal quad current ramp. As the extraction proceeds, the particles in the phase space are expected to 'dilute' and the mean X position of the beam would move towards $X = 0$. However, the size of the beam in the phase space may would be larger.

Both the dilution as well as the enlargement of the beam in the phase space was observed in the tracking simulation results, and in Figure 3.14, we see that the algorithm is able to adaptively learn and perform slow regulation. It must be noted that the gain coefficient in the algorithm for a beam that is offset had to *slightly* modified. But in real life operation, if the beam offset is detected, the algorithm's coefficients can, in principle, be reparameterized to perform the slow regulation.

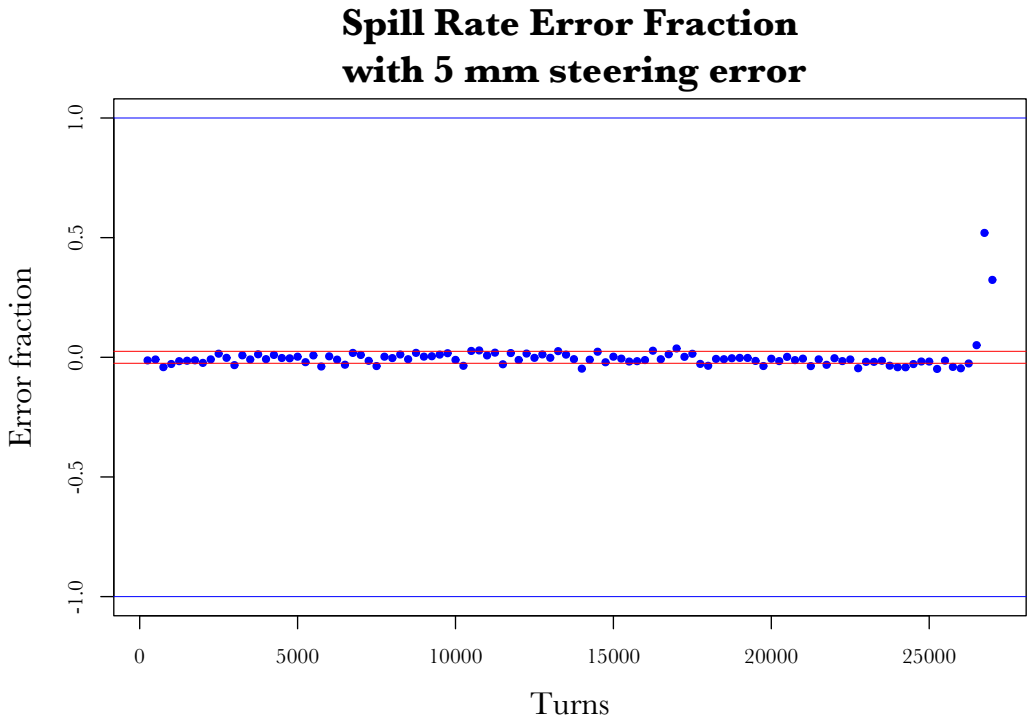

Figure 3.14: We see that the Slow Regulation Algorithm successfully regulates a beam that is offset by 5 mm which could happen due to steering errors.

3.5.1.2 Larger Emittance

The Slow Regulation algorithm was also validated with a distribution with an emittance that is explicitly twice as big as the design emittance of 16π mm-mrad. We see in Figure

3.15 that the algorithm is again able to successfully regulate towards the ideal quad current ramp profile. Here the effect is somewhat similar to what would happen if the beam is offset, but the mean of the distribution remains more or less about $X = 0$ throughout the spill.

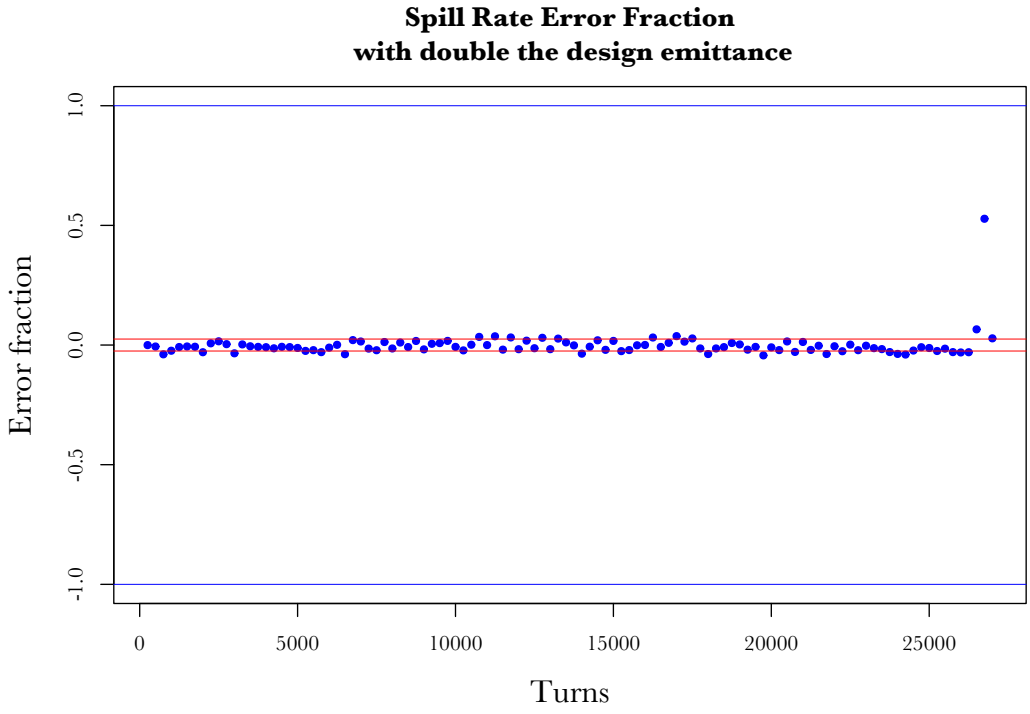

Figure 3.15: The Slow Regulation Algorithm is able to regulate a beam that has twice the emittance of the design emittance value (with the design emittance being 16π mm-mrad).

The combination of both larger emittance as well as a beam offset was also tested, and the algorithm, surprisingly, worked even without having to adjust the gain values by much. (Perhaps it is because a larger emittance value simply implies a change in the standard deviation of the initial distribution, which is not as dramatic as the whole beam being displaced from the ideal beam center.)

3.5.2 Effects of Space Charge

Space charge is a collective beam effect whereby the co-moving charged particles in a high intensity particle beam could exert Coulomb forces on each other resulting in measurable changes in the beam parameters. This effect could result in a change in the betatron tune of the beam, lowering it by a certain amount ΔQ. This can intuitively be thought as a defocussing effect due to Coulomb repulsions experienced by the traveling beam within itself [3].

The maximum tune shift can be estimated using the Laslett formula [31]:

$$\Delta Q_x = -\frac{3 r_{\mathrm{p}} N_{\mathrm{tot}} L_R}{2\pi\gamma^2 \epsilon_N L_b} \tag{3.23}$$

where r_{p} is the classic proton radius, N_{tot} is the total number of particles in the bunch, L_R is the total orbit length, ϵ_N is the 95% normalized emittance, and L_b is the effective bunch length.

Plugging in the values pertinent to *Mu2e* resonant extraction, neglecting the dispersion in the bend-sections, we find that the maximum possible tune shift could be only as large as $\Delta\nu_x = 0.0097$. If one includes the three bend-sections, this goes down by approximately a factor of 2. Since the space charge effect depends on the intensity of the particle, and since the beam bunch's intensity would be decreasing within the spill time, there would be different shifts in the horizontal tune at different times during a spill, but far lesser.

The space charge effects are certainly expected to modify the convergence rate of the Slow Regulation algorithm, but since the collective effects of the beam is expected to be present with regularity in all of the spills, we expect the algorithm to still perform but ending with a

[3] This would be from the particle's rest frame. But that Coulomb repulsion would be partly transformed as also magnetic field in the lab frame.

different ramp curve than the one it would have landed on without the space charge effects. An extensive space-charge included particle tracking was out of scope in the time frame of the writing of the thesis, but it could be taken up on for a later time.

We next look at the next component of the Quad Regulation Circuit, which is the Fast Regulation System.

3.6 Fast Regulation System

As mentioned in the previous section's introduction, the Fast Regulation System is one of the main components of the Spill Regulation System without which we cannot aim to achieve the required spill duty factor (SDF) of greater than 60%.

If the accelerator components do not have any instantaneous (or semi-random) noises whatsoever, we would have an *ideal* extraction from the DR turn-by-turn without any fluctuation in the extraction rate (or spill rate). However, in real life, we expect the spill rate to be heavily affected with irregularities due to noises that could arise from various accelerator components in the DR. Part of the noise could also come from insufficient compensation of the slow variations that may appear as the non-coherent ripples.

The main goal of the Fast Regulation System (FRS) is to mitigate these fast fluctuations in the spill rate and the regulate the rate of extraction (a.k.a spill rate) keeping it as close to the ideal extraction rate as possible. The FRS needs to have a low latency and response time compared to the Slow Regulation System since FRS needs to regulate the spill within one spill duration. The Fast Regulation system would be supplemented on top of the Slow Regulation system, with the latter providing a coarse ideal extraction rate and the former providing finer corrections to the extraction rate. For the purpose of this work, we assume

that the Slow Regulation loop and the Fast Regulation loop are sufficiently separated in the frequency domain and do no impinge on each other.

3.6.1 Need for Fast Regulation System

The fast variations in the spill rate even within one spill (a possible sample of which is shown in Fig. 3.16), can be caused by instantaneous noises arising in any of the components in the Delivery Ring. These could include random fluctuations in the accelerator components' operational parameters and/or superpositions of high frequency components from the 60 hertz harmonics from the magnet power supplies which the harmonic content suppressor system cannot fully mitigate.

3.6.2 Fast Regulation Approaches

There are two main techniques used in regulating the instantaneous noises in the spill rate arising within one spill, both of which are part of the SRS:

- Radio Frequency Knock-out (RFKO) system
- Quadrupole Regulation Circuit (QRC)

3.6.3 Radio Frequency Knock Out (RFKO)

RFKO is a technique by which the circulating beam is subjected to a transverse kick every time the beam arrives at the RFKO station (which typically would comprise of a stripline kicker). When the beam gets the transverse RF kicks over multiple turns, the beam

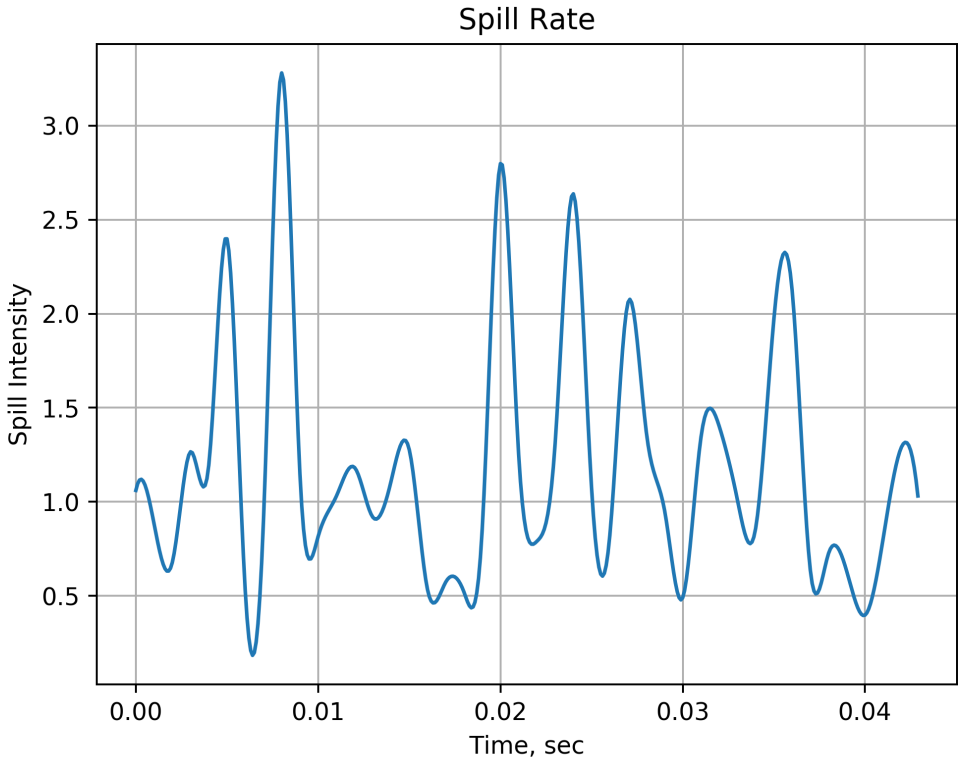

Figure 3.16: Possible variations in the spill rate within one spill duration. The spill rate here is normalized to 1. Further details behind the generation of noise is discussed in Chapter IV.

distribution in the phase space gets heated up. The RFKO frequency should be carefully chosen in order to ensure all the particles experience the kick. This frequency should thus sweep the betatron frequency band of the circulating beam. The delicately chosen RFKO frequency band, along with the intensity of its kick, could help regulate the spill rate of resonant extraction.

Assuming there is no momentum dispersion in the beam, and that the particles would be executing betatron oscillation, the RF kick should be of the following frequency in order to ensure synchronicity and successfully heat up the beam [32]:

$$f_{\mathrm{RFKO}} = (n \pm \nu_x) f_0 \tag{3.24}$$

where n is an integer, ν_x is the horizontal tune of the beam, and f_0 is the revolution frequency of the beam.

If there is momentum dispersion in the beam, then not all particles would arrive at the same time. This would thus reflect in the revolution frequency of the beam,

$$f_{\mathrm{rev}} = f_0 + f_0 \eta \frac{\mathrm{d}p}{p} \tag{3.25}$$

where η is the slip factor.

In addition to that, there would also be a shift in the betatron tune owing to chromaticity effects, with the tune shift given by,

$$\nu_x = \nu_{x0} + \xi_x \frac{\mathrm{d}p}{p} \tag{3.26}$$

where ξ is the chromaticity value.

Thus the RFKO frequency would actually span a range of frequencies proportional to the one spanned by the momentum dispersion,

$$f_{RFKO} = f_0 \left[(n \pm \nu_0) + \frac{\mathrm{d}p}{p} (\xi + \eta(n + \nu_0)) \right] \tag{3.27}$$

By thus choosing an appropriate RFKO frequency (and amplitude), we have a knob to heat the beam as much as we desire. In accelerators that provide beam for therapy in medical physics, RFKO can be employed as a tool to not just *regulate* the rate of extraction but also *drive* the rate of extraction, possibly keeping both the horizontal tune as well as the sextupole strength a constant. But the spill times in such machines is very large (in the order of seconds), unlike the spill duration for resonant extraction for *Mu2e* (43 milliseconds).

Since RFKO is inherently very fast, it could be used as a component in the Fast Regulation Loop and as a knob for finer corrections in the variations in the spill rate by appropriately modulating the frequency and amplitude of the RFKO system. (It is to be noted that one of the main disadvantages of RFKO is that the beam heating is an irreversible process.)

In this work, for the Fast Regulation system, we primarily deal with the Quadrupole Regulation Circuit (QRC), whereby the variations in the spill intensity would be mitigated by regulating the current powering the fast tune-ramping quadrupoles.

To remind the reader, as mentioned in chapter 2, our scheme of resonant extraction is to fix the sextupole strength of the 6 harmonic sextupoles and use the three dedicated fast ramping quadrupoles in the Delivery Ring to drive the horizontal tune of the circulating beam close to a third-integer resonance ($\nu_x = 29/3$). The maximum dI/dt of these quadrupoles is 16,000 A/s, with a peak current ceiling of 80 A.

The total spill duration of one spill is 43 ms and the maximum bandwidth (i.e., data collection rate) of the Spill Regulation System is 10 KHz. Thus the fast ramping quadrupoles would be regulated every 100 μs (with a maximum current change of 1.6 A/100 μs).

The Fast Regulation System would be implemented on top of the pre-loaded current ramp provided by the Slow Regulation System, with the QRC giving out a control signal that would be superposed on top of the pre-loaded current ramp. This provides for finer corrections in the spill rate to mitigate the instantaneous noises that arise within one spill.

3.7 Traditional Fast Regulation

3.7.1 PID Controller

If we take the traditional approach, the fast regulation could be done through a feedback control loop with the aid of a proportional-integral-derivative (PID) controller. PID controllers have proven to be robust in many industrial applications that require a feedback loop control on a *process variable* (PV) that needs to be regulated and kept close to a pre-decided *set point* (SP) value. PID controllers are ubiquitous in control system design and are also used in the control of resonant extraction in various accelerator facilities around the world.

A typical PID controller ingests an error value $e(t)$, and computes the control signal $u(t)$, given by,

$$\text{error} = e(t) = \text{PV} - \text{SP} \tag{3.28}$$

$$\text{control signal} = u(t) = G_p e(t) + G_i \int_0^t e(\tau)\,\mathrm{d}\tau + G_d \frac{\mathrm{d}}{\mathrm{d}t} e(t) \tag{3.29}$$

where:

- SP is the set point value, which is the ideal value we would like for the system's output to be
- PV is the process variable, the variable of interest to be measured and controlled
- $e(t)$ is the error in the system, i.e., deviation of the output of the system from the ideal output, at a given time t
- G_p is the proportional gain of the controller

- G_i is the integral gain of the controller
- G_d is the derivative gain of the controller
- $u(t)$ is the control signal given out by the PID controller which would be fed back into the system to make $e(t+1) \to 0$.

In our case, the process variable PV would be the instantaneous extraction rate, the set point SP would be normalized the ideal extraction rate of 1, the error e is the difference between ideal extraction rate and instantaneous extraction rate, and the control signal u would be the current superposed to the fast tune-ramping quadrupoles.

The PID regulation for the third-integer extraction for *Mu2e* in the Delivery Ring in the SRS would be done for every spill. Since the total spill time is 43 milliseconds and the maximum bandwidth of the system is 10 KHz, the spill data could be taken every 100 μs, giving us 430 data points within one spill. At every of the 430 time steps, the PID controller will ingest the error value, i.e., the deviation in the spill rate from the ideal spill rate, and would compute a control signal to be implemented in the next time step.

3.7.2 Spill Monitoring

Currently, there are three choices to monitor the spill data, and all three could be used to gather the spill data:

- Wall Current Monitor (WCM)
- Extinction Monitor (EM)
- Beam line Extinction Monitor

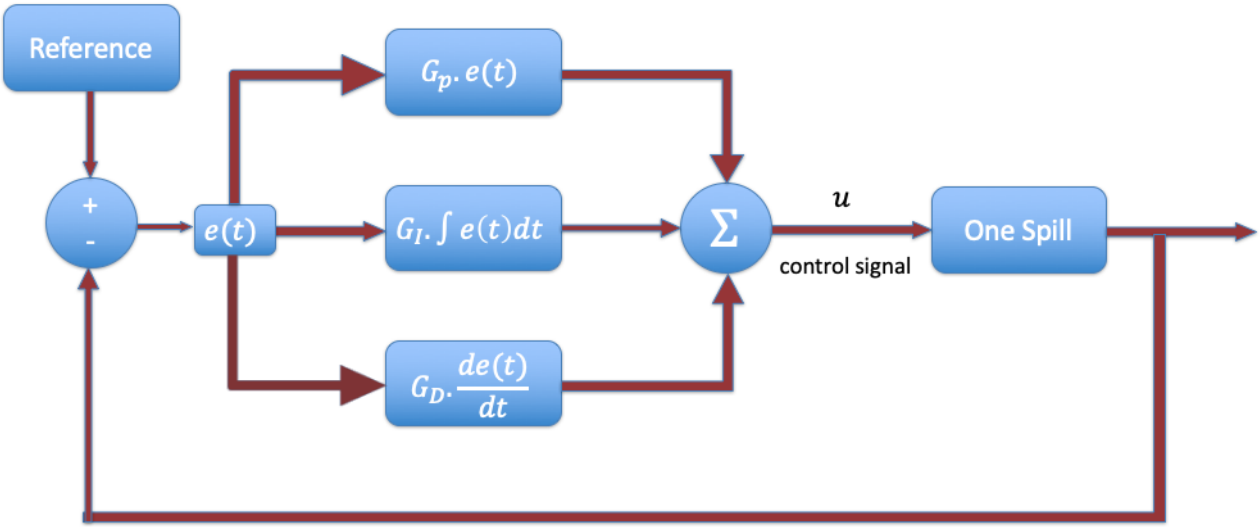

Figure 3.17: PID regulation fLow chart. The PID controller regulates the variations in the spill rate within one spill duration.

The Wall Current Monitor (WCM) [34] is a non-invasive monitoring system that is designed to measure the circulating beam's intensity at the revolution frequency of about 590 KHz. The WCM primarily consists of a beam pipe that involved 12 magnetic cores and a beam pipe-ceramic gap that measures the beam intensity through the mirror charge induced on the device by the passing beam. The WCM could be placed either inside the Delivery Ring or in the extraction beam line (or both).

The most accurate measurement we could hope to get of the spill data is that from the Extinction Monitor (EM). The EM is primarily designed to detect protons (and other particles) both in the extinction window between the two proton pulses in order to reduce background effects, and also to detect protons during the spill process. The EM also provides the feedback signal to the SRS; this signal will be provided by digitizing the analog output of one of the Extinction Monitor scintillator detectors and sending it to the Regulation System via an optical fiber [36].

As an additional source of the spill monitoring, which would especially important during the early beam commissioning to the Diagnostic Absorber, is the scintillator based monitor

that is planned to be installed in the M4 beam line, downstream of the Halo Collimator 907. This monitor uses the same principles as the Extinction Spill Monitor.

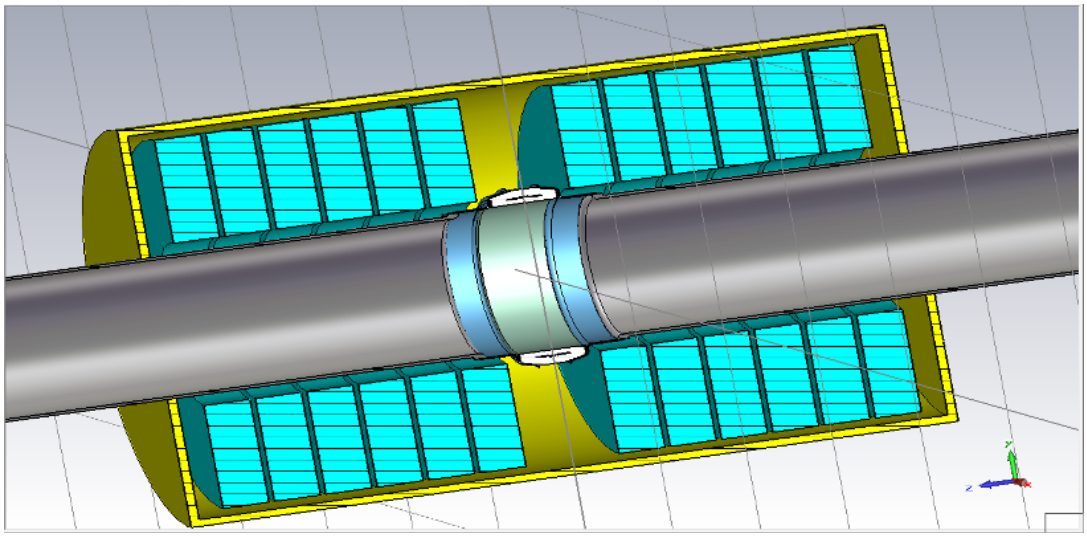

Figure 3.18: A cut-away view of the Wall Current Monitor that measures the beam intensity non-invasively [34].

3.8 Hardware requirement for the SRS

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. The 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. The Slow Regulation system shall be housed in the HPS since it can support spill-to-spill feedforward corrections [38].

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 [38].

The spill data fed to the SRS could be from one (or all) sources listed in the previous subsection, i.e., the Wall Current Monitor, and the Extinction Monitor and the beam line Extinction Monitor.

3.8.1 MATLAB Simulation

Since the Fast Regulation Loop has to regulate the spill intensity variations within one spill duration, it has to be implemented in a hardware that has very low latency time. The FRL shall thus be employed on an Intel Arria 10 FPGA contained in the SoC.

In order to perform the PID control operation in the FPGA, one needs to program the FPGA with a VHDL (VHSIC Hardware Description Language) code that prescribes the PID control logic on to the firmware. The author was involved in the initial efforts of generating a VHDL code using one of MATLAB's toolboxes called the 'HDL Coder' that could perform PID control logic.

The HDL Coder toolbox is not only capable of generating a hardware description language code but is also interlinked in MATLAB's Simulink environment where a specific process for which control system is to be designed, can be modeled and simulated.

The fast regulation of the SRS was first modeled by the author in the MATLAB Simulink in order to build a 'plant' environment in which the VHDL code could be generated. An

analytical model of the resonant extraction was constructed (discussed in Chapter IV) and it was created as a built-in custom function block inside Simulink.

A separately written code in the MATLAB environment generates the required input files into the MATLAB workspace. Once the files and the variables are loaded into the workspace, the Simulink software automatically detects them from the workspace.

Figure 3.19: Simulink model layout to simulate single spill using analytical model.

The Simulink model starts with a block from the workspace that contains the time-series data of the input quadrupole ramp. This signal is then sent into a custom written function block called '`beam_ext`' where the beam extraction is modelled. Since the tune-

ramping quadrupole current curve for an ideal extraction is logarithmic, the custom defined '`beam_ext`' function models the extracted number of particles as

$$N_{\text{ext.}} = N_{\text{part}} \exp\{-cI(t)/(2\sigma^2)\} \tag{3.30}$$

where $I(t)$ is the quad current ramp, c is a factor that relates the beam size to a given quad current (see equation (4.42) for a detailed derivation), and the σ is the horizontal rms beam size (which is computed from the design emittance of the beam and the Courant-Snyder beta function value at the observation point), and the total number of particles at the start of the extraction (which is 10^{12}).

The output of this function block is fed to a temporary memory block to store the circulating beam's intensity. The function of the temporary memory is to remember the circulating particle's intensity in the previous time step so that the model can compute the number of particles extracted.

The output from the '`beam_ext`' function and the temporary memory block 'B' is then fed into another custom written function called '`spill_rate`'. This function computes the difference between the intensity of the circulating beam and the intensity of the circulating beam from the previous time step to compute the spill rate for the present time step.

The output of the spill monitor is then sent to a difference block to compute the difference between the computed spill rate and the ideal spill rate. This would be our error value e.

This error e is then fed into a block that simulates the PID control action. The PID controller is a custom built-in function that computes the control signal based on what settings we set the controller to (the derivative action of the controller includes the low-pass filter component mentioned in the previous subsection).

The PID block was set to a discrete controller (since the process is discrete), and the sampling rate of the controller was taken to be 100 μs. The sampling rate mentioned here is not the rate at which the controller operates but the rate at which the data is fed into the controller. The internal clock of the controller must run at a higher rate than the maximum design bandwidth for the reason that the controller would take some finite number of steps to calculate the control signal (this was taken to be 125 MHz), and must also be carefully synchronized with the SRS decision rate.

Once the PID controller gets the input signal, it computes the control signal u, which is then sent to an adder block which adds the control signal to the pre-defined quad current ramp (the first block that was mentioned here). The new quad current for the next time step is now in the direction of reduction in error, and the feedback loop continues for 430 points.

Once the simulation was tested to give the desired output, the HDL Coder tool was used to generate the VHDL code for the PID controller.

3.8.2 HDL Coder

The HDL Coder is a toolbox in MATLAB that generates portable, synthesizable Verilog and VHDL code from MATLAB functions, Simulink models, and Stateflow charts [40]. The generated HDL code can be used for FPGA programming or ASIC prototyping and design.

In the typical toolbox-less Simulink environment of MATLAB , there exists hundreds of ready-to-use blocks which serve as a building block to build your own custom plant environment. The HDL Coder toolbox not only has the capability to generate VHDL code out of custom written functions but it also encompasses a few hundred of the pre-existing Simulink block that is compatible with hardware description language generator environment.

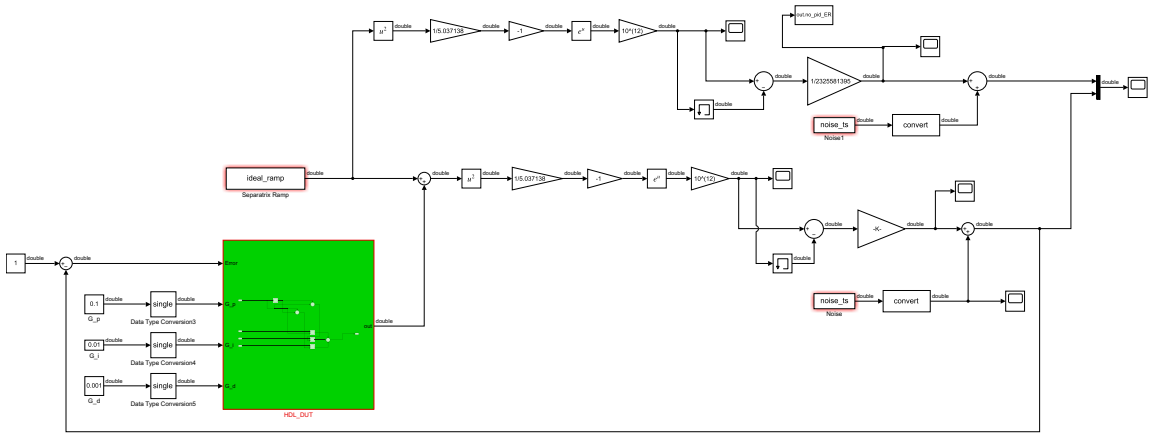

Figure 3.20: The Simulink layout in the HDL Coder environment with the PID controller constructed as a Device Under Test (DUT).

To generate the VHDL code, the spill simulation using the analytical model tested out in the normal Simulink mode was adopted in the HDL Coder. A separate 'Device Under Test' was created to emulate the PID control, and to generate a VHDL code to enable performing the same operation in an Arria 10 FPGA. The layout of the Simulink model employing the HDL Coder and DUT is shown in Figure 3.20. The layout of the explicit (HDL Coder compatible) blocks emulating a PID controller is shown in Figure 3.21.

We use the HDL Workflow Advisor tool to help generate the VHDL code for our particular control system. The HDL Coder is compatible with a few select FPGA devices, of which Intel Arria 10 is one (which is the intended hardware to be used for the Fast Regulation).

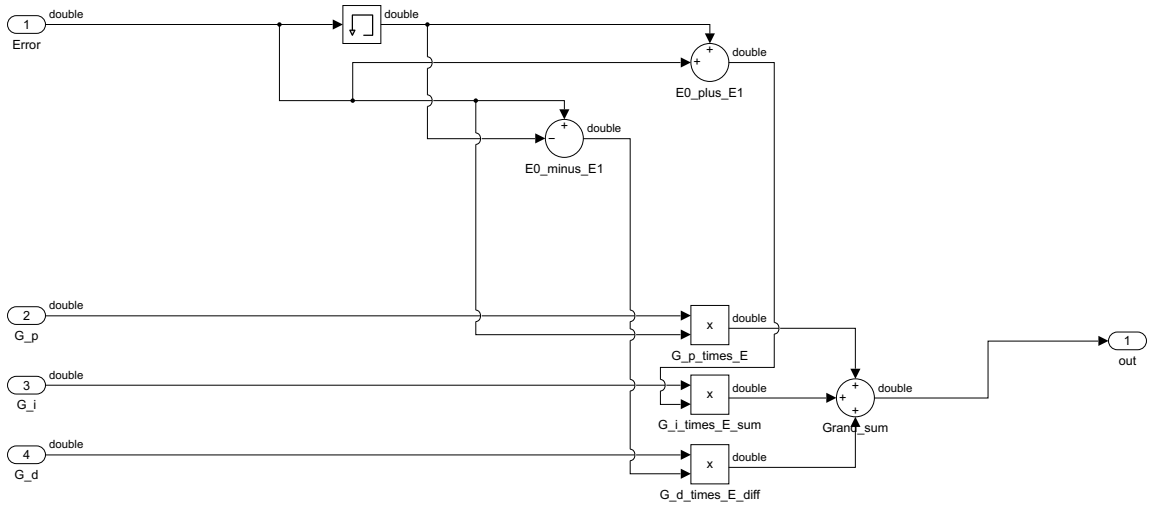

Figure 3.21: The inner contents of the Device Under Test (the green box in Figure 3.20).

The device selected in the Simulink was 10AS016C3U19E2LG, and the synthesis tool was chosen to be Altera Quartus II. The clock rate frequency of the FPGA operation was chosen to be 125 MHz.

Before generating the HDL code, we first verify the working of the Simulink model. After successfully verifying the performance of the model in Simulink, we prepare the model for HDL Code generation using the Workflow Advisor which checks the model if it is compatible to be able to generate the VHDL code. If any non-'HDL Coder' blocks was used in the Simulink model, an error will show up. Since the custom environment function were proving to be challenging, the explicit mathematical functions involved inside the custom function

(for beam extraction, for example) was modeled using the HDL Coder compatible Simulink block.

Once the Workflow Advisor checks the model and 'okays' it, we then go to 'Code Generation' and we select 'VHDL' (the HDL Coder can also generate Verilog, which is another hardware description language). We then also enable the generation of RTL Code and Testbench code.

We then validate the model with a proposed fixed-point data type and generate a fixed-point design. This is done so because the data types typically used in an FPGA would be fixed point and not floating point.

After validating the data types, the custom built control model for the extraction environment is now ready for the generation of the VHDL code.

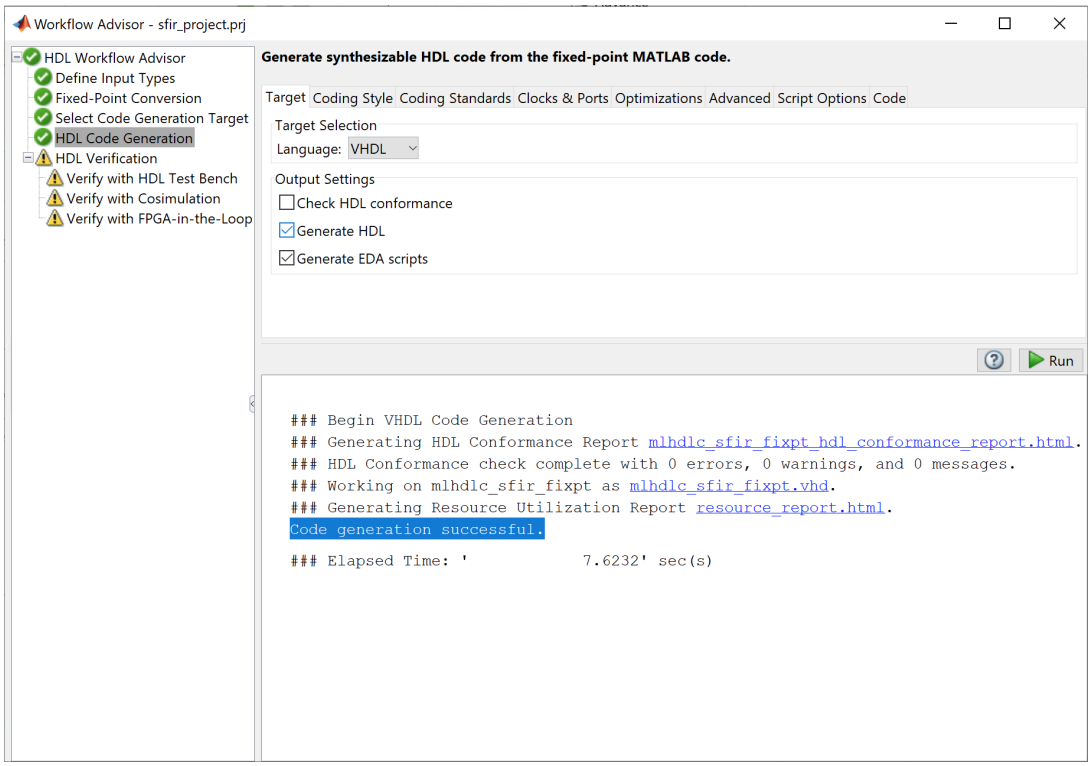

Figure 3.22: The HDL Coder successfully generates the VHDL code.

The VHDL code for the PID operation was successfully generated (Figure 3.22) after validating the working of the resonant extraction simulation environment on Simulink. A sample of the generated VHDL code is shown in Figure 3.23.

A corresponding test bench code was also generated, which could be used to test out the generated VHDL code to perform the PID action. The test bench code can be thought of as a platform on which the DUT can be tested (PID in our case). Since the physical hardware was not ready at the time of writing the thesis, the generated VHDL code is yet to be physically programmed into the FPGA. But the VHDL code is ready to be used (or edited) to suit the needs of the control system in real life operation.

In this chapter, we looked at an overview of the Spill Regulation System, and its components, primarily the Slow Regulation and the Fast Regulation loops. We presented the details of the particle tracking simulation code and an HPC scheme to implement millions of particles. We looked at the result of the Slow Regulation algorithms and its robustness under slow varying conditions of the beam, such as steering error and higher emittance value. We also looked into the Fast Regulation loop and its traditional mode of operation through PID control, and generation of the VDHL code for the traditional Fast Regulation loop.

In the next chapter, we investigate using Machine Learning to enhance the Fast Regulation loop, along with an introduction to Machine Learning and the methods that were studied.

```
-- -------------------------------------------------------------
--
-- File Name: hdlsrc\Standalone_PID\HDL_DUT_tb.vhd
-- Author: Aakaash Narayanan
-- Created: 2021-11-17 13:44:41
--
-- Generated by MATLAB 9.10 and HDL Coder 3.18
--
--
-- -------------------------------------------------------------
-- Rate and Clocking Details
-- -------------------------------------------------------------
-- Model base rate: 8e-07
-- Target subsystem base rate: 0.0001
-- Explicit user oversample request: 125x
--
--
-- Clock Enable  Sample Time
-- -------------------------------------------------------------
-- ce_out        0.0001
-- -------------------------------------------------------------
--
--
-- Output Signal                 Clock Enable  Sample Time
-- -------------------------------------------------------------
-- out_rsvd                      ce_out        0.0001
-- -------------------------------------------------------------
--
-- -------------------------------------------------------------


-- -------------------------------------------------------------
--
-- Module: HDL_DUT_tb
-- Source Path:
-- Hierarchy Level: 0
--
-- -------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.std_logic_textio.ALL;
USE IEEE.float_pkg.ALL;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
LIBRARY STD;
USE STD.textio.ALL;
USE work.HDL_DUT_tb_pkg.ALL;

ENTITY HDL_DUT_tb IS
END HDL_DUT_tb;
```

Figure 3.23: A sample of the generated VHDL code.

CHAPTER 4

MACHINE LEARNING TECHNIQUES IN FAST REGULATION SYSTEM

In this chapter, we look into enhancing the Fast Regulation using Machine Learning (ML) techniques. We shall cover a brief introduction to ML, neural networks, supervised learning, and reinforcement learning. We shall look into using gradient descent approach to optimize a PID controller, recurrent neural network schemes (such as Long-Short Term Memory and Gated Recurrent Units) to possibly *replace* the PID controller, and also actor-critic family of policy gradient methods to regulate the spill using reinforcement learning. [1]

4.1 Introduction to Machine Learning

Machine Learning (ML) can broadly be defined as an approach to make an agent achieve a specific task by training the agent. The computing agent could be as simple as a neural network or it could involve a more complex computational architecture.

The goal of Machine Learning is to train an agent by feeding it with an appropriate amount of training data and help the agent achieve a task without *explicitly* coding for it to achieve the task. All we do is to *tell* the agent *how* to learn through a learning algorithm, and the agent would (ideally) learn it from a set of training data through *inference*, provided the learning algorithm is efficient.

[1] The work done in this chapter is in part with the Real Time Edge AI Distributed System (READS) collaboration at Fermilab along with Northwestern University, and the author was an active participant in the ML efforts alongside other members, amongst whom the author would like to especially thank Mattson Thieme and Jing Jiang for their collaborative work that is interspersed along with the author's.

The object of machine learning is to infer (or predict) the values of some unknown function at points which have not yet been observed. To do so, we begin by fitting a model to the function using a subset of the observed values. Fitting the model, also called optimization, is generally conducted via an iterative process by which the parameters of the model are gradually updated to minimize the difference between the model predictions and the ground truth. The quality of a learned model is then assessed by its performance on "unseen" data, or data which were not used in the previous optimization step. (This is also known as the "generalization performance".)

There are primarily three broad types of ML:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

In this work, we shall primarily be looking at efforts in using Supervised Learning and Reinforcement Learning to help enhance the Fast Regulation system for the resonant extraction.

Supervised learning is a field of ML whereby, as the name suggests, we feed *labelled* training data and make the ML agent learn to achieve a specific task. By 'labelled' data, what we mean is that the data is tagged with a meaningful parameter that could be useful for the agent to parse out and learn to perform the task.

For example, if we are training an ML agent try to learn and process a language through text, we would typically feed the agent texts of sentences of the language. The data in the textual sentences could be labelled into nouns, pronouns, verbs, etc., to help the agent identify parts of the language. If we are training an ML agent to recognize voice, the training data could be labelled as male, female, inanimate noises such a whistle or a bark of a dog or a car honk, and so on, to help the agent classify different kinds of noises.

For the Fast Regulation loop, the reader might recall from Chapter 3 that the traditional method used for Fast Regulation is to use a PID control logic. As a first step to enhance the regulation system, we investigate here the optimization of the gains of the PID controller using the techniques of Machine Learning. We also investigate the possibility of replacing the entire PID controller with an ML agent using supervised learning, and we report the initial results of using reinforcement learning to replace the PID controller. In order to help understand the training process and the results, we introduce the reader to some basic notions of ML that is pertinent to the efforts being discussed here. For a more exhaustive review of supervised learning, the reader can refer to [41], [42], [43], and [44].

4.1.1 Artificial Neural Network

An artificial neural network (or simply 'neural network' from now on) is a computational architecture consisting of a circuit or network of neurons. This is inspired from biological systems in Nature where various neurons (or nodes) are interlinked to each other with a certain weight ascribed to each of the connections that help in processing and learning information. Even though the *actual* functionings of a mammalian (or any other) neurological system is extremely complex, we borrow just the basic structural idea from them to help us learn to perform specific tasks through computational means.

The basic building block of a neural network is called a node. A node can take in multiple inputs but typically gives out only one output. The inputs to a node are weighted with a certain weight values, after the operation of which a single bias value is added. The weights can be thought as the 'strength' of the link between nodes, and the weights and biases of a network determine the output of the network.

A neural network starts with a set of nodes called the 'input nodes', the layer of which is called the 'input layer'. The input layer is connected to a set of nodes in the next layer. The final set of nodes on which the network ends is called the 'output nodes', and the layer called the 'output layer'. There could be any number of layers containing any number of nodes in-between the input layer and the output layer. These layers in-between are called the 'hidden layers'. The overall arrangement of the input layer, the output layer, the number of hidden layers, and the number of nodes per hidden layer, would constitute the 'topology' of a neural network.

Let us suppose we have a single node i that is connected with N number of nodes on the input side (as shown in Figure 4.2) and gives out one output y_i. Let us suppose each of the input nodes have values x_j, running from $j = 1 \rightarrow N_j$. Let us denote the weights of each connection between the j-th node and our node of interest i to be w_{ij}.

The output from the node i is simply the summation of all the products of the weights and the respective input node value with one bias value added for each connection. Mathematically, this is simply a dot product between the input node values and the weights of the network that is then summed with a bias value,

$$y_i = W_{i1}x_{i1} + W_{i2}x_{i2} + W_{i3}x_{i3} + ... + W_{iN}x_{iN} + B_i \tag{4.1}$$

$$= (\sum_{j=1}^{N} W_{ij}X_{ij}) + B_i \tag{4.2}$$

$$= W_i \cdot X_i + B_i \tag{4.3}$$

The node of our interest, the i-th node, could in-turn be one of many nodes in the network, and the output of this node could be fed as the input to all the subsequent nodes it is connected to in the next layer.

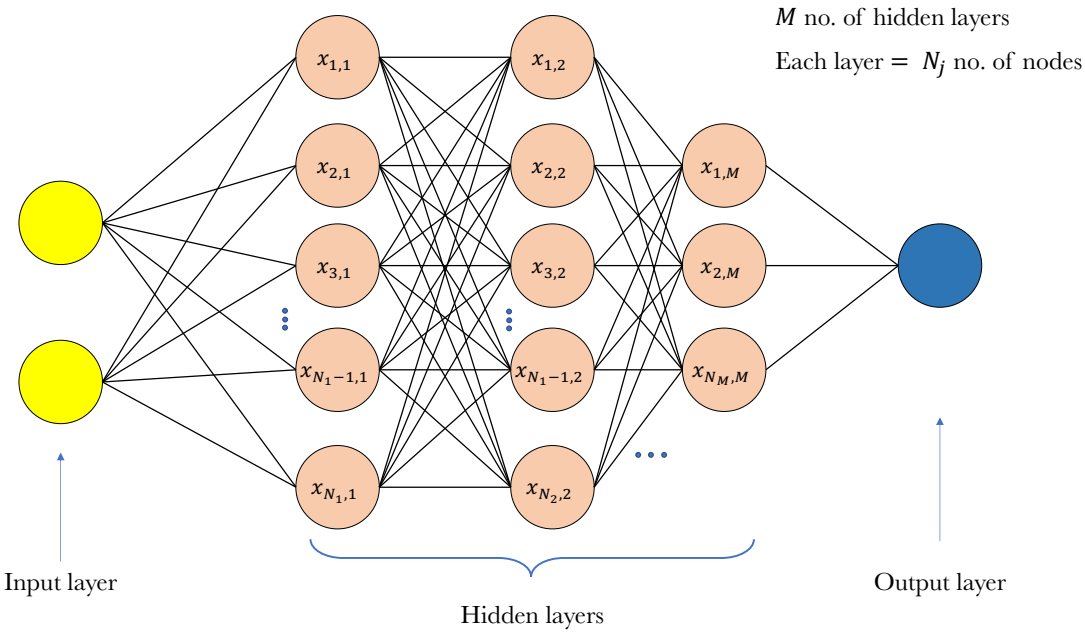

Figure 4.1: An illustration of a neural network with 2 input nodes and one output node, with M hidden layers in-between. A neural network can typically consist of any number of input/output nodes.

This output is then typically inputted into another function, called the activation function, in order to normalize the output value and also make sure that the output does not diverge to $\pm\infty$. There exists many activation functions, but some of the typically used activation functions are:

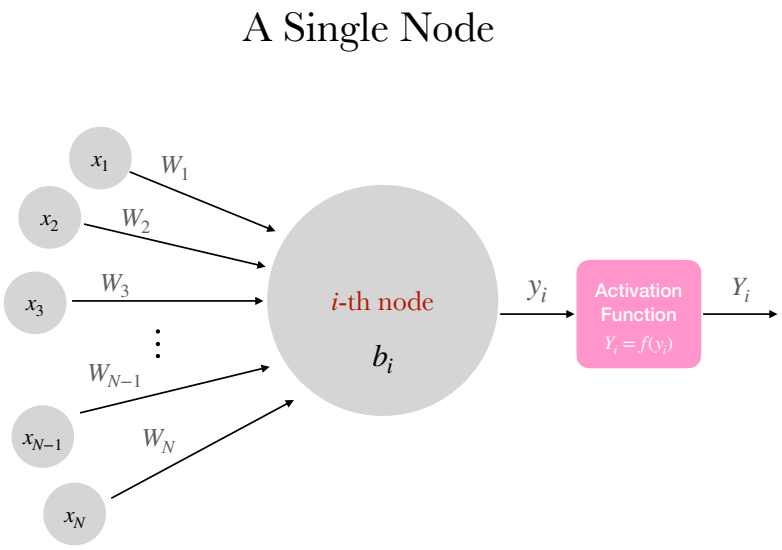

Figure 4.2: Illustration of a single node that is connected to N nodes from the previous layer.

- 'ReLU' activation (a.k.a rectilinear unit), which inputs a value y and outputs:

$$Y(y) = \begin{cases} ay, & \text{if } y > 0 \\ 0, & y \leq 0 \end{cases} \tag{4.4}$$

  where a is a constant.

- The sigmoid activation function, given by,

$$Y(y) = \frac{1}{1 + e^{-y}} \tag{4.5}$$

- The `tanh` activation function, given by,

$$Y(y) = \frac{e^{y} + e^{-y}}{e^{y} - e^{y}} \tag{4.6}$$

And in some rare scenarios that involve only linear functions, one could get away with having no activation function at all, provided the behavior of the network throughout the training is reasonable and the output of the network does not diverge. The role of the activation, thus, is to not only make sure the output is normalized but to also add nonlinearities into the model such that it can fit nonlinear functions.

As illustrated in Figure 4.1, the final output of a neural network would be a function of all the dot products between all the node values with its respective weights, added with their respective bias, being passed through an activation function, and across all the layers to the terminal node(s).

The reason we care about neural networks is because neural networks make up the backbone of what is called 'deep learning', whereby appropriate training data is inputted into the network, and the weights and biases of the networks are updated with every training iteration to make the output of the network as close to the desired output value as possible. In some sense, neural network can be thought of as a glorified fitting function, but a very powerful one when used with the right learning algorithm. Neural networks can also be used as a means to generate an output which can act as an input into an external system in order to control or optimize it. In this work, we leverage the power of neural network to train it and eventually let it control the resonant extraction process so it can regulate the uniformity of extraction rate.

In order to understand *how* a neural network really *learns*, i.e., updates its weights, we have to understand backpropagation [45] [46] of error and what gradient based optimization does.

4.1.2 Error Backpropagation

Let us suppose we have a neural network with M number of layers with each layer consisting of N number of nodes, with the input number of nodes being N_{input}, and let us suppose we want the output of the network to ideally be a certain y_{ideal}.

Our goal is to 'train' the neural network to achieve the ideal output for a given set of input node values. The training could take place over many number of iterations. In the first iteration, the weights in the network between all of the nodes typically would be randomly generated values (since the network has to start *somewhere*), so there is no guarantee that the output of the network would be even remotely close to the ideal desired output.

Our goal is to find a method to optimize the weights of the network in such a way that the output value of the network comes as close to the ideal output value as possible (or at least converges towards it with every subsequent iteration). And the way we achieve it is through a technique called the *backpropagation of error*.

4.1.3 A simple example

In order to elucidate the reader with the method of backpropagation (since it is very crucial), let us work through a very quick example of a simple neural network and how the weights are updated. Just for illustrative purposes, let us suppose we have a neural network having just 1 hidden layer, with the input layer containing 2 nodes, the hidden layer containing 2 nodes, and the output layer containing 1 node. In order to make the example simple, let us not have any activation function. The network is illustrated in Figure 4.3.

Let the weights connecting the first two layers be denoted as w_1, w_2, w_3, and w_4. Let the weights connecting the hidden layer and the output layer be w_5 and w_6. Let the input node

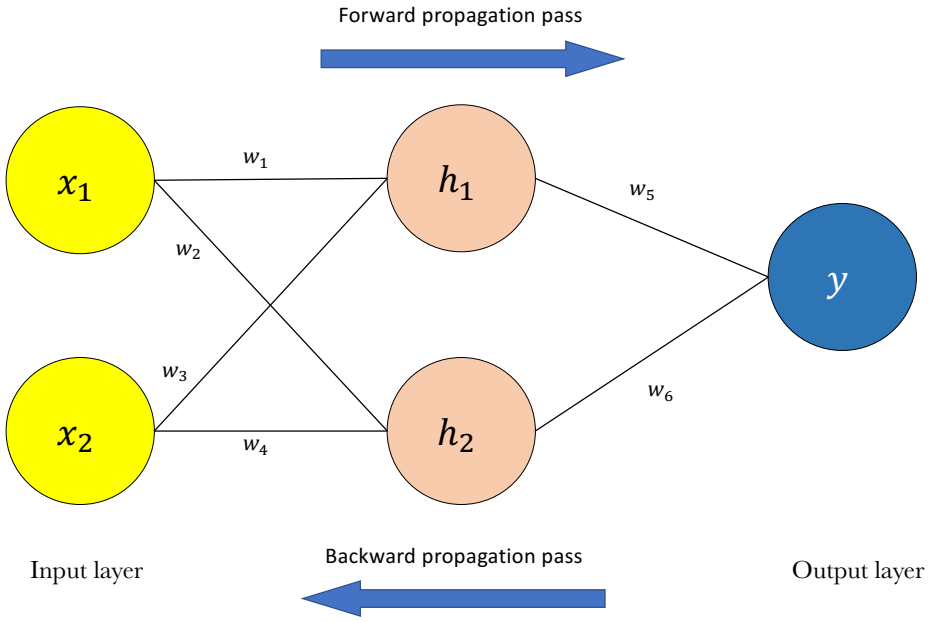

Figure 4.3: An example neural network for which backpropagation and gradient descent is illustrated [47] in section 4.1.3.

values be denoted as x_1 and x_2, let the hidden layer be denoted as h_1 and h_2, and let the output layer be denoted as y. (Here we assume the bias value is zero for all the nodes.)

Let us suppose that our initial node values are fixed to be $x_1 = 2$ and $x_2 = 3$, and we would like for the ideal output of the network to be of value $y = 1$ given those input values to the network.

In the very first iteration, the weights are randomly generated, so let us ascribe the following values:

$$w_1 = 0.11 \tag{4.7}$$

$$w_2 = 0.21 \tag{4.8}$$

$$w_3 = 0.12 \tag{4.9}$$

$$w_4 = 0.08 \tag{4.10}$$

$$w_5 = 0.14 \tag{4.11}$$

$$w_6 = 0.15 \tag{4.12}$$

with $x_1 = 2$ and $x_2 = 3$.

If we perform a feedforward pass with these weight values, we get the output y (also known as the 'prediction') to be,

$$y = \left(\begin{pmatrix} w_1 & w_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \begin{pmatrix} w_3 & w_4 \end{pmatrix} \begin{pmatrix} x_3 \\ x_4 \end{pmatrix} \right) \begin{pmatrix} w_5 \\ w_6 \end{pmatrix} \tag{4.13}$$

$$= \Big[\underbrace{(x_1 w_1 + x_2 w_2)}_{h_1} \Big] \times w_5 + \Big[\underbrace{(x_3 w_3 + x_4 w_4)}_{h_2} \Big] \times w_6 \tag{4.14}$$

$$= 0.191 \tag{4.15}$$

We have now successfully performed a single forward pass of the network. Since our initial weights were randomly generated and ascribed, it is no surprise that the output is no where close to the desired ideal output of 1.

The error e in the output is defined as,

$$\delta y = y - y_{\text{ideal}} \tag{4.16}$$

$$e = \frac{1}{2}(\delta y)^2 \tag{4.17}$$

Our goal is to minimize the error by appropriately changing the weights before we perform the next forward pass. The way the weights are updated in the most basic backpropagation method is through what is called the 'gradient descent' approach.

The new weight w_i^* is obtained from the old weight w_i through,

$$w_i^* = w_i - \alpha \frac{\partial e}{\partial w_i} \tag{4.18}$$

where the parameter α is called the 'learning rate', and is one of the crucial hyperparameters in any of the deep learning algorithms that involve neural networks.

Since we are *back*propagating, we have to sequentially go through the weights in the reverse order of the network from the output. The derivative of the error function with respect to weights is taken by the aid of chain rule.

We can start with optimizing the weight w_6 since it is one of the first weights that connects the output node to the hidden layer. The new weight w_6^* would be given by, δy

$$w_6^* = w_6 - \alpha \frac{\partial e}{\partial w_6} \tag{4.19}$$

$$= w_6 - \alpha \frac{\partial e}{\partial y} \frac{\partial y}{\partial w_6} \tag{4.20}$$

$$= w_6 - \alpha \frac{1}{2} \frac{\partial (\delta y)^2}{\partial y} \frac{\partial y}{\partial w_6} \tag{4.21}$$

$$= w_6 - \alpha (y - y_{\text{ideal}}) \frac{\partial (y - y_{\text{ideal}})}{\partial y} \frac{\partial y}{\partial w_6} \tag{4.22}$$

$$= w_6 - \alpha (y - y_{\text{ideal}}) \frac{\partial y}{\partial w_6} \tag{4.23}$$

But we know from the forward pass equation (4.14) that y is simply,

$$y = \Big[(x_1 w_1 + x_2 w_2)\Big] \times w_5 + \Big[(x_3 w_3 + x_4 w_4)\Big] \times w_6 \tag{4.24}$$

$$\implies \frac{\partial y}{\partial w_6} = (x_3 w_3 + x_4 w_4) \tag{4.25}$$

$$= h_2 \tag{4.26}$$

Thus the new weight is simply,

$$w_6^* = w_6 - \alpha(\delta y) h_2 \tag{4.27}$$

where all the involved variables on the RHS are known to us apriori after the first forwardpass and we can thus readily compute the updated weight w_6^*.

In the same fashion, we find that the updated weight for w_5 is,

$$w_5^* = w_5 - \alpha(\delta y) h_1 \tag{4.28}$$

The reader can apply the same chain rule to backpropagate through the weights w_1, w_2, w_3 and w_4 and find that the newly updated weights are,

$$w_6^* = w_6 - \alpha(h_2(\delta y)) \tag{4.29}$$

$$w_5^* = w_5 - \alpha(h_1(\delta y)) \tag{4.30}$$

$$w_4^* = w_4 - \alpha(x_2(\delta y) w_6) \tag{4.31}$$

$$w_3^* = w_3 - \alpha(x_1(\delta y) w_6) \tag{4.32}$$

$$w_2^* = w_2 - \alpha(x_2(\delta y) w_5) \tag{4.33}$$

$$w_1^* = w_1 - \alpha(x_1(\delta y) w_5) \tag{4.34}$$

Assuming a learning rate of $\alpha = 0.05$, plugging in the new weights and performing a forward pass for the second iteration, we find the new output of the network to be,

$$y = \Big[(x_1 w_1^* + x_2 w_2^*)\Big] \times w_5^* + \Big[(x_3 w_3^* + x_4 w_4^*)\Big] \times w_6^* \tag{4.35}$$

$$= 0.26 \tag{4.36}$$

We see that the new output 0.26 is closer to our ideal desired output of 1 than the output of the previous network (which was $y = 0.191$), and this is because the weights are updated *in the direction* of minimizing the error between the ideal output and the actual output. If we perform this feedforward and backward propagation over 10 times, we get the output to be 0.95, and we see a clear convergence of the output towards the ideal desired output of 1. (The rate of this convergence depends on the initial weight values and also on the learning rate. If the learning rate is made too big, then the network could keep oscillating around the minima but never converge. One must thus be careful in choosing appropriate hyperparameters before training a network.)

In our little example above, the input layer had 2 nodes and the output layer had only one node. But the power of deep learning and the backprogation algorithm is that it can work through arbitrary number of input and output nodes with arbitrary number of hidden layers that could have arbitrary number of hidden nodes.

Let us suppose we want to train a neural network to identify cat in a given image. In this case, the input to the network could be a collection of pixels of a given image (with or without cat), and we can set the network's output to be either 1 (there is a cat) or 0 (there is no cat). In order to train the network (i.e., update its weights in the right direction), we feed the network thousands of known images for which we already have a label saying if it has a cat or not. Based on the error in the predicted output of the network with each image fed, we perform a backpropagation of the error after every iteration to update the network's

weights. Once we are confident that the error value is consistently close to nil, the network can be considered 'trained' and can successfully classify (with a very high probability) an unknown new image to either have a cat in it or not.

We next look to leverage this optimization algorithm in order to tune the gain values of the PID controller in the Fast Regulation system. But in order to do that, we would first need to create a physics simulator that could generate the required data to train any ML algorithm.

4.2 Resonant Extraction Physics Simulator

In order to facilitate machine *learning*, we need an environment to generate the required training data to train various ML algorithms.

One option would be to generate the training data for the resonant extraction would be to use particle tracking (as was done for validating Slow Regulation algorithms). However, as mentioned in Chapter 3, numerical particle tracking can be very computationally expensive to simulate. To remind the reader, to simulate one full resonant extraction for ≈130,000 particles in the initial distribution takes about 20 minutes when the load is distributed over 40 cores. Particle tracking can thus be very impractical if we have to simulate hundreds of thousands of spills (as would be the case, for instance, in reinforcement learning).

Another handicap in using particle tracking to generate training data for the supervised learning is that the particle tracking tool would be an external 'black box' that is not part of the chain of numerical operations in minimizing the loss function and there would be a break in the computation graph, making it a non-differentiable simulator. This would lead to the neural network's weights simply not getting updated.

We thus built an *analytical* model of the resonant extraction process that, although makes some approximations, is expected to be reasonably close to the actual physics process of the extraction. The advantage with the analytical model is that we can simulate the resonant extraction without having to perform particle tracking, and the computation speed is orders of magnitude faster than the case of using particle tracking to generate training data.

4.2.1 Analytical Modeling of Third Integer Resonant Extraction

The essence of the analytical model is to calculate the analytical form of the ideal quad current ramp for which the spill rate would be uniform. In this subsection, we derive and justify a logarithmic quad current curve for an ideal extraction in a third-integer resonant extraction process.

4.2.1.1 Shrinking separatrix

In the third-integer resonant extraction scheme, we know from equation (2.40) that as the horizontal tune of the beam approaches a third-integer resonant tune, the area of the separatrix shrinks, and thus particles become unstable and their positions start to increase non-linearly with every turn. The reader may recall from Chapter II that the shape of the separatrix separating the stable from the unstable region is triangular. We assume the particle density distribution (for both X and X') in the normalized phase space to be Gaussian when the extraction begins.

In the analytical approximation, for the ease of derivation, we could assume the separatrix to be circular instead of triangular. Let us try to derive an expression for the rate of squeezing of the separtrix circle (as shown in Figure 4.4) that would give us a uniform rate of extraction.

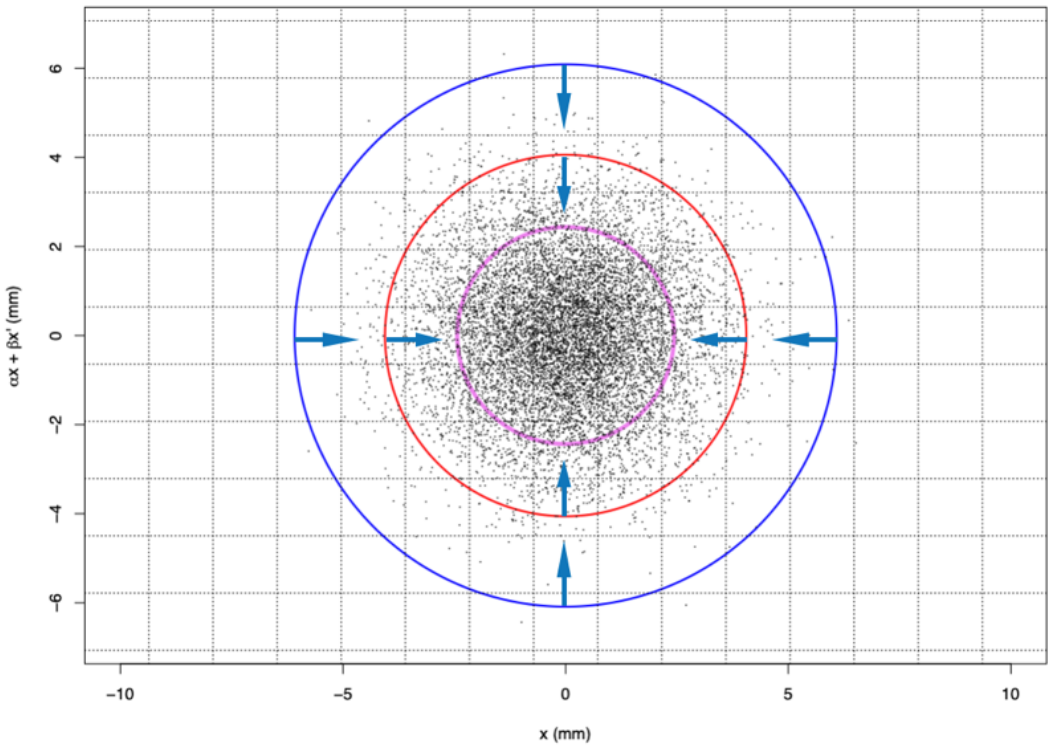

Figure 4.4: To analytically calculate the quad ramp function, we assume the separatrix of the third-integer resonant extraction to be circular instead of triangular.

4.2.1.2 Particle Density Function

If we want to find the total number of particles N within a circle of radius r_0 with the particle density function being $f(r)$, then,

$$N(r_0) = N_{\text{total}} \int_0^{2\pi} \int_0^{r_0} f(r) r \, \mathrm{d}r \, \mathrm{d}\phi \tag{4.37}$$

Let us define our Gaussian distribution density function, with a standard deviation σ, as

$$\boxed{f(r) = \frac{1}{2\pi\sigma^2} e^{-r^2/2\sigma^2}} \tag{4.38}$$

The total number of particles N within a given radius r_0 thus would be,

$$N(r_0) = N_{\text{tot}} \int_0^{r_0} \int_0^{2\pi} f(r) r \, \mathrm{d}r \, \mathrm{d}\phi \tag{4.39}$$

$$= N_{\text{tot}} \int_0^{r_0} \int_0^{2\pi} \frac{1}{2\pi\sigma^2} e^{-r^2/2\sigma^2} r \, \mathrm{d}r \, \mathrm{d}\phi \tag{4.40}$$

$$= 2\pi N_{\text{tot}} \int_0^{r_0} \frac{1}{2\pi\sigma^2} e^{-r^2/2\sigma^2} r \, \mathrm{d}r \tag{4.41}$$

$$\boxed{\frac{N(r_0)}{N_{\text{tot}}} = \left(1 - e^{-r_0^2/2\sigma^2}\right)} \tag{4.42}$$

Even though the Gaussian tail technically extends till infinity, let us have a cut-off radius of $r_{\text{max}} = 6\sigma$ to account for *almost* all of the particles. If we plugged in a value of $\sigma_{\text{beam}} = 1.58$ mm in x-coordinate, the plot below gives us the fraction of particles under a certain radius r, computed from $r = 0$ to $r = 6\sigma$.

We see in Figure 4.5 that initially the fraction of particles covered within the radius of circle increases since the chunk of the Gaussian lies within 1σ of 1.58 mm. But as we keep increasing the radius r, the number of particles covered within that circle does not increase as much because the particle density wanes off exponentially (as a Gaussian naturally does), and thus the fraction of particles contained within that radius starts to saturate asymptotically towards 1.

4.2.1.3 Computing Separatrix Radii for a Uniform Spill

In our spill process, we need a uniform spill quality for over N_{turn} number of turns. In other words, we need a fraction of $N_{\text{particles}}/N_{\text{turn}}$ to get extracted every turn to achieve a

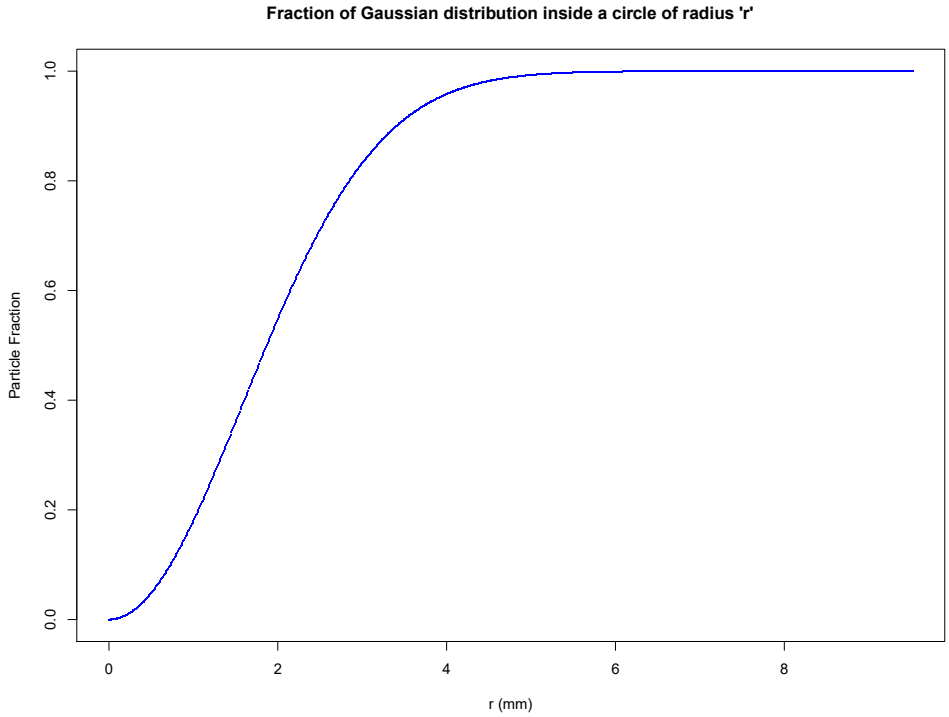

Figure 4.5: The fraction of particles inside the circular separatrix computed as a function of the separatrix's radius.

uniform extraction rate. This mathematically translates to the condition that the area under the $f(r)$ vs r curve in the above plot for r_1 to r_2 must be the same as the area under curve for r_2 to r_3.

$$\int_0^{2\pi} \int_{r_1}^{r_2} f(r) r \, \mathrm{d}r \, \mathrm{d}\phi = \int_0^{2\pi} \int_{r_2}^{r_3} f(r) r \, \mathrm{d}r \, \mathrm{d}\phi \tag{4.43}$$

For a Gaussian particle density in the phase space, this means,

$$\int_{r_1}^{r_2} 2\pi f(r) r \, \mathrm{d}r = [e^{-r_1^2/2\sigma^2} - e^{-r_2^2/2\sigma^2}] \tag{4.44}$$

$$\int_{r_2}^{r_3} 2\pi f(r) r \, \mathrm{d}r = [e^{-r_2^2/2\sigma^2} - e^{-r_3^2/2\sigma^2}] \tag{4.45}$$

$$\implies [e^{-r_1^2/2\sigma^2} - e^{-r_2^2/2\sigma^2}] = [e^{-r_2^2/2\sigma^2} - e^{-r_3^2/2\sigma^2}] \tag{4.46}$$

We can now solve for the *next* radius r_3 which would give us the same value of area under the curve between r_2 and r_3 as the area under the curve between r_1 and r_2. We get,

$$\boxed{r_3 = \sqrt{-2\sigma^2 \log\left(2e^{-r_2^2/2\sigma^2} - e^{-r_1^2/2\sigma^2}\right)}} \tag{4.47}$$

And we can thus compute all of the subsequent radii $r_4, r_5, ..., r_{\mathrm{N_{turn}}}$ iteratively.

To kick start this process from $r = 0$ to $r = r_1$ and find the first radius r_1, we need to know the exact fraction that we want to extract. If the total spill time is T_{spill} and if our revolution time period is $T_{\mathrm{rev.}}$, the total number of turns would be $T_{\mathrm{spill}}/T_{\mathrm{rev.}} = N_{\mathrm{turn}}$. If we normalize the total initial number of particles to be 1, we would then need $1/N_{\mathrm{turn}}$ particles

to be extracted at every turn, including the first turn. And this condition gives us our first r_1,

$$\int_0^{r_1} f(r) r \, \mathrm{d}r = 1/N_{\text{turn}} \tag{4.48}$$

$$= \frac{1}{2\pi\sigma^2} \int_0^{r_1} 2\pi e^{-r^2/2\sigma^2} r \, \mathrm{d}r \tag{4.49}$$

$$1/N_{\text{turn}} = \frac{1}{2\pi\sigma^2} 2\pi\sigma^2 (1 - e^{-r_1^2/2\sigma^2}) \tag{4.50}$$

$$e^{-r_1^2/2\sigma^2} = 1 - \frac{1}{N_{\text{turn}}} \tag{4.51}$$

$$\boxed{r_1 = \sqrt{-2\sigma^2 \log\left(1 - \frac{1}{N_{\text{turn}}}\right)}} \tag{4.52}$$

With a beam rms size of $\sigma_{\text{beam}} = 1.58$ mm, T_{spill} of 43 ms, T_{cycl} of 1.694μs, and an N_{turn} of 25371, we get r_1 to be about 0.0140 mm. And we can thus iteratively compute the next '$N_{\text{turn}} - 1$' number of radii values that would give us uniform fraction of particles contained in the area enclosed between any pair of r_n and r_{n+1}.

The next question is to find the tune value from knowing the radius of the separtrix. We know that for third-integer resonance, the x-coordinate of the stationary points is directly proportional to the tune distance from resonant tune δQ. Since we are only qualitatively interested in finding the tune curve, we can approximate that $r = k\delta Q$, where k is a constant that depends on the inverse of sextupole strength. In other words, how δQ should change with every turn (to get a uniform spill) is directly proportional to how r should change with every turn (which we have already computed above).

4.2.1.4 Relation to tune distance

In a regular sextupole driven third-integer resonance extraction, the length of the side of triangular separatrix is approximately given by [4],

$$x_0 = \frac{16\pi\delta Q}{A} \tag{4.53}$$

where δQ is the tune distance of machine from the resonant tune, given by $\delta Q = \nu_{\text{resonance}} - \nu_{\text{machine}}$, and the parameter A is proportional to the sextupole strength, given by [4],

$$A = \frac{\beta_0}{B\rho} \oint \left(\frac{\beta}{\beta_0}\right)^{3/2} \left(\frac{B''}{2}\right) \cos\left(3\nu_0\phi\right) \mathrm{d}s \tag{4.54}$$

Since we have approximated the sepratrix to be circular, let us see how the evolution of radius with time relates to the evolution of the horizontal tune.

We can approximate the radius of our separatrix circle to be about half of the side of the triangle, $r = x_0/2 = 8\pi\delta Q/A$. If the sextupole strength is constant throughout extraction, as in our extraction scheme, the tune distance is directly proportional to the radius.

$$\delta(r) \propto \frac{A}{8\pi} r \tag{4.55}$$

If we assume to have only one sextuple in our lattice with a strength of 500 T/m^2, a sextupole length of 0.46 m, magnetic rigidity of 29.15 T-m, and assume that the Courant-Snyder β value at the observation point to be 12 m and at the sextupole to be 9 m, we get A to be 0.0596 (A is also dependent on the tune itself, which can also be computed iteratively for every turn). Since we already have computed the radii r-s on the RHS, we can now

compute the respective tune values, and thus, the tune-ramp function required for uniform spill.

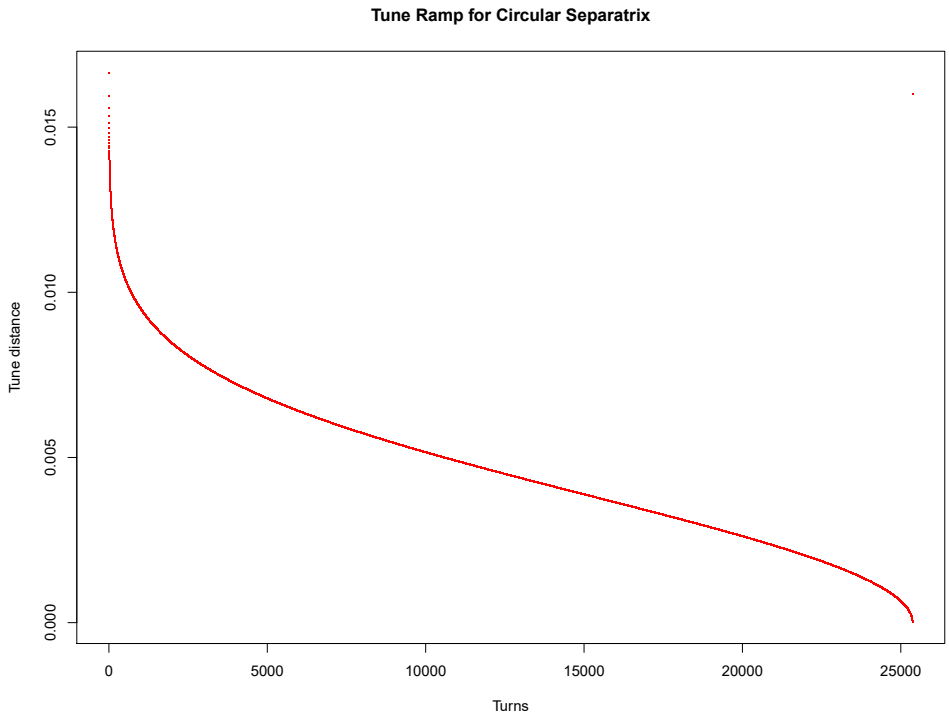

Figure 4.6: Analytically calculated tune distance $\delta\nu = 29/3 - \nu_x$ plotted as a function of number of turns.

From section 3.1.3, we know that the tune of the beam is directly proportional to the quadrupole strength (the whole premise upon which the quad regulation circuit hinges on). Thus the tune ramp curve can be converted to quadrupole strength, the equivalent quad power supply ramp to the above calculated tune distance ramp would be the one shown in Figure 4.7. And the corresponding density function (in congruence, the fraction of extracted particles) when the above tune-ramp is applied is now a linear function, as shown in Figure 4.8.

We see that the quad current ramp profile calculated analytically using the above model approximates a logarithmic curve. We can thus use a logarithmic function for the quad current ramp profile to get an ideal extraction rate.

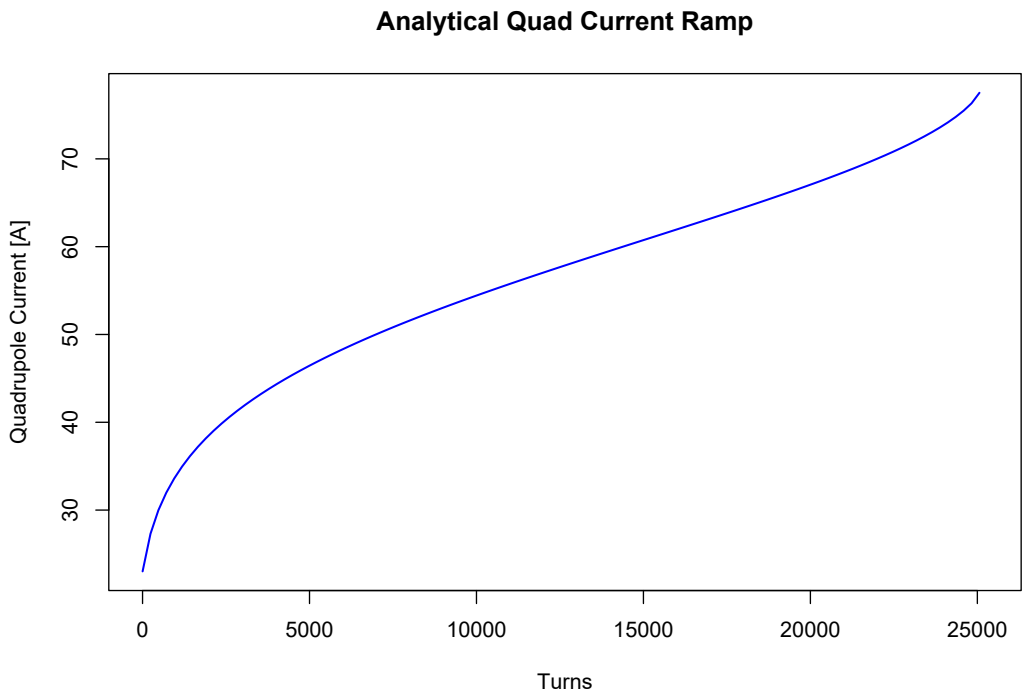

Figure 4.7: Analytically calculated quadrupole current ramp for one full spill duration.

4.2.1.5 Assumptions in the Analytical Model

The above scheme employed is under the assumption that the separatrix is circular and the beam distribution is Gaussian. However, in the actual resonant extraction process:

- the separatrix is triangular and not circular,
- the beam distribution in the stable region may not remain Gaussian throughout the extraction,

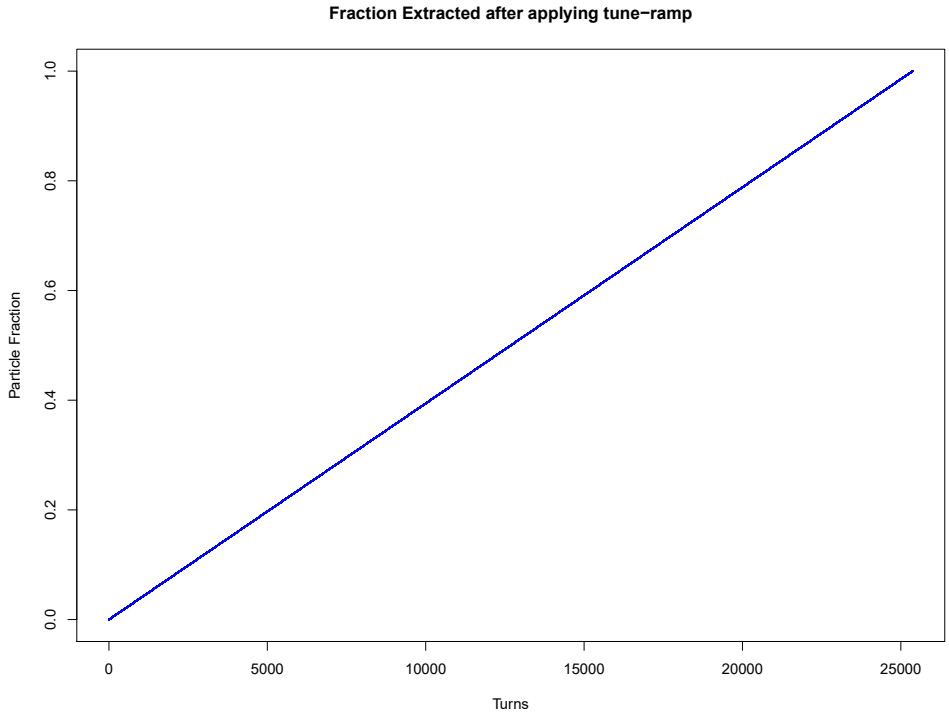

Figure 4.8: Particle fraction outside the circular separatrix in for the analytically calculated quad current ramp. Since we assume the particle to be extracted once it is outside the separatrix, this also corresponds to the total number of particles extracted.

- the initial injected beam may not always be perfectly centered,
- the particles near the stationary points (the vertices of the triangle) are going to move slower than the ones away from it,
- the particle does not get 'immediately' extracted once it goes out of the separatrix, as the particle takes a finite transit time (and more than one turn) to get past the septum (the actual effects of transit time is discussed in Chapter V),
- the transit time may also vary depending on where the particle radially is in the phase-space,
- the intensity dependent effects, such as space charge effects, may be present.

Even with these assumptions, it is very reasonable to expect the tune ramp curve for the actual resonant extraction process to be similar in quality as the one we derived above [37].

4.2.2 Python Modelling

With the analytical modeling of the resonant extraction in mind, we can build a simplified physics simulator in order to simulate the resonant process and generate the training data required to validate various ML algorithms that would help enhance the Fast Regulation loop. A physics simulator code was thus written in Python and we mention the details of the physics simulator code in this subsection.

4.2.2.1 Noise in spill rate

The modeling of the spill, for the purposes of Fast Regulation, assumes a *perfect* extraction in the absence of any instantaneous noises in the system. Since the goal of the simulator is to enhance fast regulation, we separate the slow regulation from the fast regulation so that we can understand its effects independently (it is also convenient that both are separated reasonably far apart in the frequency domain since the slow regulation does not regulate fast noises). We make the assumption that the Slow Regulation system (mentioned in Chapter III) provides us with an ideal quad current ramp already, which we assume with our analytical modeling to be a logarithmic curve.

Since we do not know the exact nature of the random fluctuations in the spill rate from the fast variation ripples, we model the noise using a log-normal distribution as an educated guess. For *every* spill, a new noise is generated with a mean value of 0.

The reader might recall from Chapter 2 that the SDF is a measure of the quality of the spill, and is defined as,

$$\text{SDF} = \frac{1}{1 + \sigma^2_{\text{spill}}} \tag{4.56}$$

The standard deviation of the noise is chosen such that the average spill duty factor (SDF) of a full spill is ≈0.50.

The physics simulator generates random walk log-normal points at 1 KHz rate and interpolates the generated points to provide a smooth noise data for over 430 time steps (since the maximum SRS design bandwidth is 10 KHz), with each time step constituting 100 μs (or equivalently ≈60 turns).

The expectation value of the spill rate is normalized to 1, and the log-normal generated noise is added to 1 to give us a noise profile for every spill. A sample variation is shown in Figure 4.9.

4.2.2.2 PID Control Loop

Once the noise in the spill rate is generated, we now have the error in the spill rate for every of the time steps. The noise data for the whole spill is generated and is sequentially fed to the PID control loop, along with the three gain values of the PID controller passed as an argument.

The PID function inputs the gain values, the spill rate data, and computes the control signal for each of the time steps, given by,

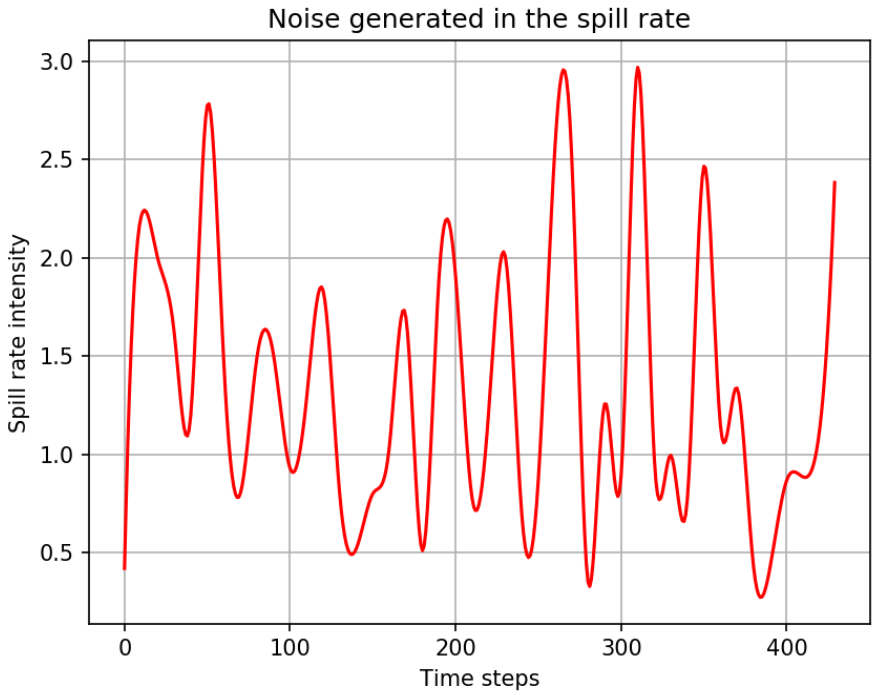

Figure 4.9: Variations in spill intensity generated by the physics simulator using log-normal distribution. We normalize the expectation value of the variations to 1.

$$u(t) = G_p e(t) + G_I \int_0^t e(\tau)\,\mathrm{d}\tau + \frac{\mathrm{d}}{\mathrm{d}t} e(t) \tag{4.57}$$

where G_p, G_i, G_d are the proportional, integral, and derivative gain. In this modelling of the PID controller, we do not incorporate a low-pass filter within the derivative action of the controller since the generated noise is guaranteed to not have any divergence [2] in $de(t)/dt$.

[2]Even though we do not yet know the nature of the random fast noise that may arise in the real beam operation, this is a reasonable assumption to have since we could, in principle, perform low-pass filtering of the error before feeding it to the PID controller.

4.2.2.3 B-Field Shielding Effect

Since we are using Quadrupole Regulation Circuit for the Fast Regulation, the control signal output from the PID control block will eventually be superposed to the ideal quad current ramp, which will in turn reflect in the magnetic field produced by the quadrupoles.

In reality, at the location of the fast tune-shifting quadrupoles, the beam circulates inside a stainless steel pipe and the magnet poles lie outside the beam pipe with the magnetic fields penetrating through the beam pipe material. Due to magnetic permeability of the material and electromagnetic skin effects, stainless steel has the property of shielding and attenuating high frequency variations in the magnetic field given out by the magnet. If the noise in the spill rate fluctuates very fast for some reason, the control signal, too, could follow and fluctuate very fast in order to mitigate the spill error. This could result in high frequency components in the quadruple's magnetic field. But these high frequency corrections would never reach the beam owing to the shielding effect by the stainless steel beam pipe [48].

In order to simulate the magnetic field screening of the stainless pipe, the output of the control signal is passed through a Butterworth low-pass filter with a low-pass frequency of 1 KHz.

4.2.2.4 Transit Time Delay

Once the horizontal tune of the beam is changed by the fast tune-shifting quadrupoles, the separatrix in the phase-space would shrink by some amount. This shrinking would leave a slice of particles in the unstable region. These particles will not get immediately get extracted but would take some finite time to reach the electrostatic septa. Thus there would be a finite delay in the beam response from any change in the tune-shifting quadrupoles.

In order to account for this delay in the beam response, we have a transit time delay function. Particle tracking simulations were done in order to find the transit time delay (to be discussed in detail in Chapter V), and a conservative 100 turn delay is added to the extraction rate throughout the spill. (Even though the transit time could vary as the spill progress, a constant transit time for the whole spill is not too unreasonable to assume given its smallness compared to the total spill duration.)

4.2.2.5 Quadrupole Response

After the control signal is low-pass filtered and the transit time delay is accounted for, the control signal is passed through a function that superposes it to the predefined logarithmic ideal quad current ramp value. The ideal quad current was taken to be logarithmic function of the spill time step with a maximum current value of 100 A (even though the maximum current value is not relevant as far as this simulation is concerned). In Figure 4.10, we see the quad ramp profile after the control signal is superposed.

4.2.2.6 Beam Extraction

After we obtain the actual quadrupole current ramp, which now includes the control signal superposed to it, we pass the current ramp to the `beam_extraction` function to compute the extracted beam intensity.

Since we have assumed a logarithmic curve in the analytical model of the spill for ideal extraction, we need a mathematical function that would map the logarithmic curve to a linear curve (which would be the total intensity of extracted particles with every time step). And according to the analytical model,

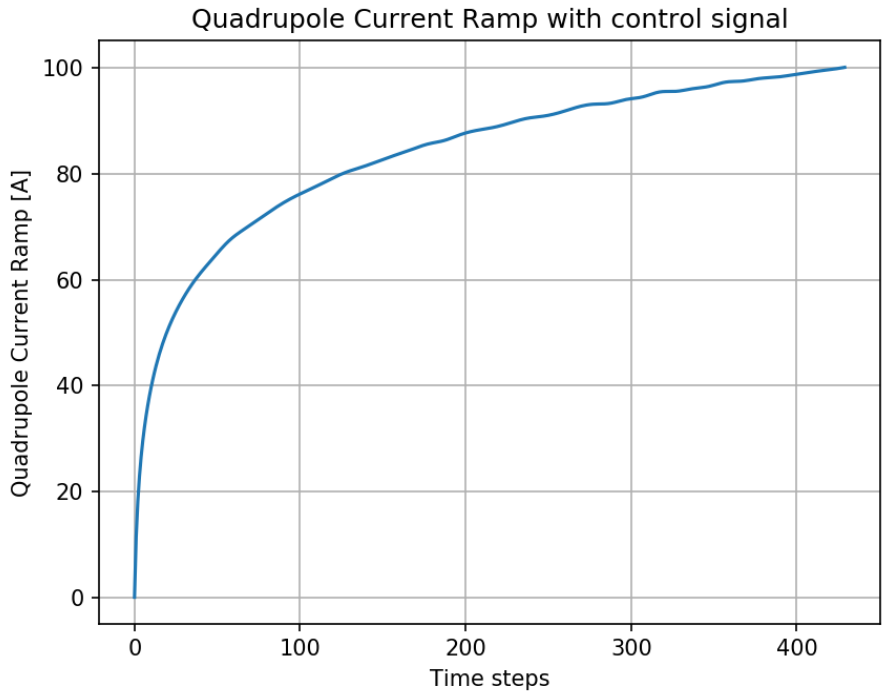

Figure 4.10: The quadrupole current ramp after the control signal from the PID loop is superposed in every time step.

$$\mathrm{N_{ext.}} = N_{\text{tot. step}}^{\frac{I(t)}{I_{\max}}-1} \tag{4.58}$$

where $N_{\text{tot. step}}$ is the total number of time steps (which is 430) and $I_{\max} = 100$ A.

The extracted beam intensity is shown in Figure 4.11. (If $I(t)$ is purely logarithmic without any control signal added to it, $\mathrm{N_{ext.}}$ would be purely a linear function, indicating that the rate of extraction is a constant. We see in Figure 4.11 that the extracted particle intensity is not strictly linear. This is because, even though the PID control has given the control signal, the noise has not been fully mitigated. But the control signal has done some regulation, as will be shown in Figure 4.12 below.)

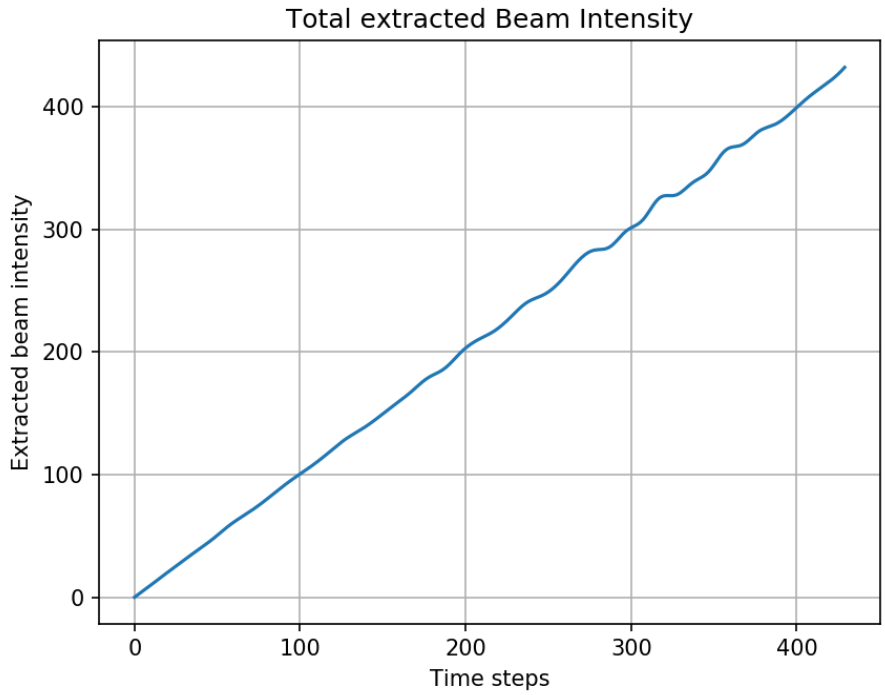

Figure 4.11: The count of total extracted beam intensity after applying the control signal from the PID controller to the ideal quad current ramp. We see that the intensity is not strictly linear, and that is because of the noise in the spill rate that we put in by hand through the noise generator function.

4.2.2.7 Spill Monitor

Once we have the total number of extracted beam intensity for every time step, we can next calculate the rate of spill intensity. This function computes the difference in the total spill intensity from j-th time step from $(j+1)$-th time step. The spill monitor thus outputs the *regulated* extraction rate, which is the final output of the physics simulator. Here we do not assume any bandwidth limitations for the spill monitor, and we also assume no additional noise coming from the spill monitor.

In Figure 4.12, we see both the regulated spill as well the unregulated spill (i.e., how the spill would have looked like if there was no fast regulation in place). Since the expectation

value of the noise is normalized to 1, we would expect an *ideal* extraction (without any noise) to bear a constant value of 1. And so, we see in Figure 4.12 that the regulated spill is a little more attenuated (and closer to 1) than the unregulated spill rate. We also see that the SDF of the regulated spill is much closer to 1 than the unregulated spill rate.

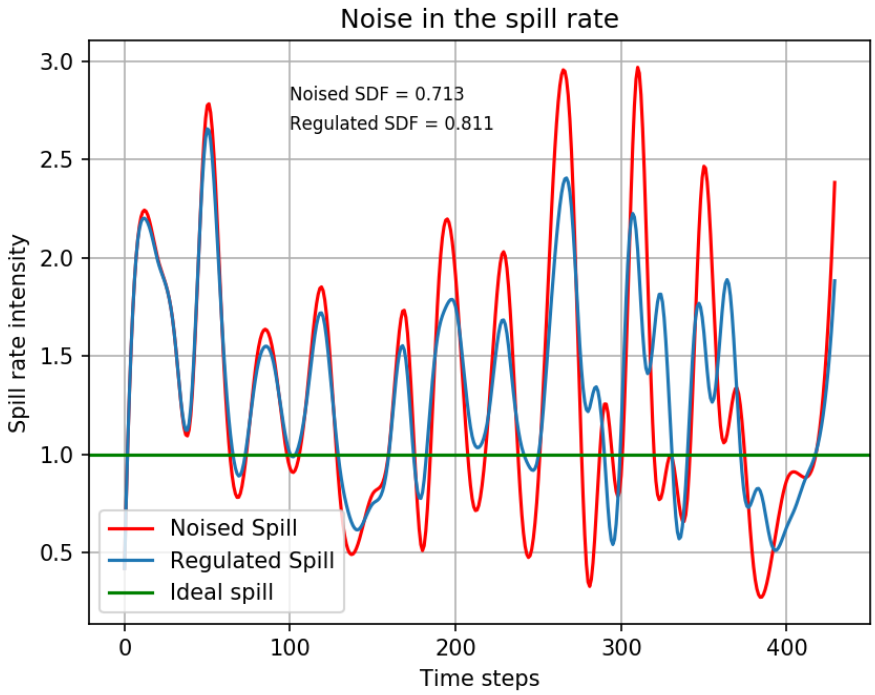

Figure 4.12: The noised extraction rate is the variations in the spill rate if there was no control system to curtail it. The regulated spill rate is after the spill is regulated by a PID controller. The ideal spill rate would bear a constant value of 1. We see that the regulated spill is a little more 'flatter' and attenuated towards 1 than the unregulated spill rate.

The goal of this study is to use the above constructed physics simulator in order to investigate ML techniques that would help regulate the spill rate such that the SDF is improved close to 1 as much as possible.

4.3 Optimization of PID Gain Values

As a first step in employing ML technique to enhance the Fast Regulation, we try to optimize the three gain values of the PID controller using back propagation and gradient descent methods discussed in the previous section.

The way we optimize the PID gains is to construct a very simple neural network consisting of 3 input nodes and one output node. Beforehand, random values are ascribed to the three input nodes, which stand for the three PID gain values. Once the gain values are generated, they are then passed as an argument and is fed into the physics simulator which inputs the three gains and has a noise function that is capable of generating a log-normal random noise for every spill.

The physics simulator then goes through all the blocks mentioned in the previous section, to give us a PID regulated spill rate profile in the end. Now that we have the regulated spill, we compute the SDF value for a given spill to measure the quality of the PID regulation for those gain values. Our target SDF value is 1, thus we can construct an error function (or 'loss' function), defined as,

$$l = (\mathrm{SDF}_{\mathrm{spill}} - 1)^2 \tag{4.59}$$

Now that we have an error value l, we backpropagate the error and take the gradient descent approach in order to optimize the weights of our neural network. The optimization code was written in Python using the PyTorch [49] framework. In this model, while back-propagating the loss, we differentiate through all the physics simulator functions that involve the gain values in them.

Since the noise generation function, for instance, does not involve any of the gain values, we simply by-pass backpropagating through it. Thus our physics simulator can be classified as what is called a 'hybrid' differentiable simulator in supervised learning, which helps saving computation time by avoiding differentiating through functions that do not involve optimizable parameters (as those arithmetic operations would result in zero values anyway).

4.3.1 Optimizer

In our earlier toy example, we had looked at the normal gradient descent approach, whereby the weights are updated by backpropagating the loss function and taking the gradient with respect to the weights multiplied by a factor called the learning rate. Let us rewrite equation (4.18) using a more succinct notation:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta J(\theta) \tag{4.60}$$

where $\theta \in \mathbb{R}^D$ would constitute the parameters of the model (such as the weights of our neural network with dimension D), α is the learning rate that determines the step size of the gradient to be subtracted, and $J(\theta)$ is the objective function (or the loss function) that we wish to minimize.

Even though the normal gradient descent approach could be effective, it runs into the risk of being stuck in a ravine where the parameters are optimized well in one dimension but are optimized badly in other dimensions. This approach could also result in oscillations about the slopes of a local minima but never converging on to the minima.

In order to work around this, several other optimization algorithms exists, one amongst which is a relatively new optimization technique called Adam optimization [50], which stands for 'Adaptive Momentum' optimization (introduced in 2015). In this optimization scheme,

we update the model parameters not just with respect to the gradient of the objective function but we also take into account the 'velocity' of the descent at every step.

The Adam optimization can be written as,

$$g_t = \nabla_\theta J(\theta_t) \tag{4.61}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{4.62}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{4.63}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{4.64}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{4.65}$$

$$\boxed{\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t} \tag{4.66}$$

where m_t and v_t are the mean and the variance of the gradients respectively. The authors who introduced the Adam optimization propose the β values to be set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and the ϵ value to be set to 10^{-8}.

Adam optimization has the advantage of often not getting stuck in a local ravine while descending in the gradient space. It also has an empirically proven record to be robust in a lot of learning applications since its inception in 2015, giving us enough motivation to consider using it as the optimizer.

4.3.2 Learning Flow

In order to optimize the PID gain values, we employ Adam optimization to update the gains with a learning rate value of 0.01. The workflow of the ML training for the PID gain optimization is as follows:

- Before the training begins, three random weights (i.e., gain values) are initialized.
- The gain values are fed into the physics simulator.
- The physics simulator inputs the gain values and performs one full spill.
- After one full spill is simulated, we get the regulated spill rate as the output from the physics simulator.
- Once the regulated spill profile is gotten, the loss function L is computed as mentioned in equation (4.59).
- The loss value is now backpropagated through the hybrid differentiable simulator and the weights (gains) are updated with respect to equation (4.66).
- The newly updates weights would now be ready to be fed into the physics simulator to regulate the next spill.
- The physics simulator generates a *new* noise profile altogether, upon with the PID controller regulates with the updated gain values.
- This iteratively continues until the loss function l is minimized to zero.

In Figure 4.13, we plot the evolution of the SDF over 2500 iterations. We see that the SDF converging towards 1, the loss function is converging towards 0. We see in Figure 4.14 the evolution of the PID gains over 2500 iterations.

4.3.3 Multiple Domains

In the above scheme of gain optimization, we had only three gain values that were constant throughout the entirety of a single spill. However, resonant extraction is inherently a

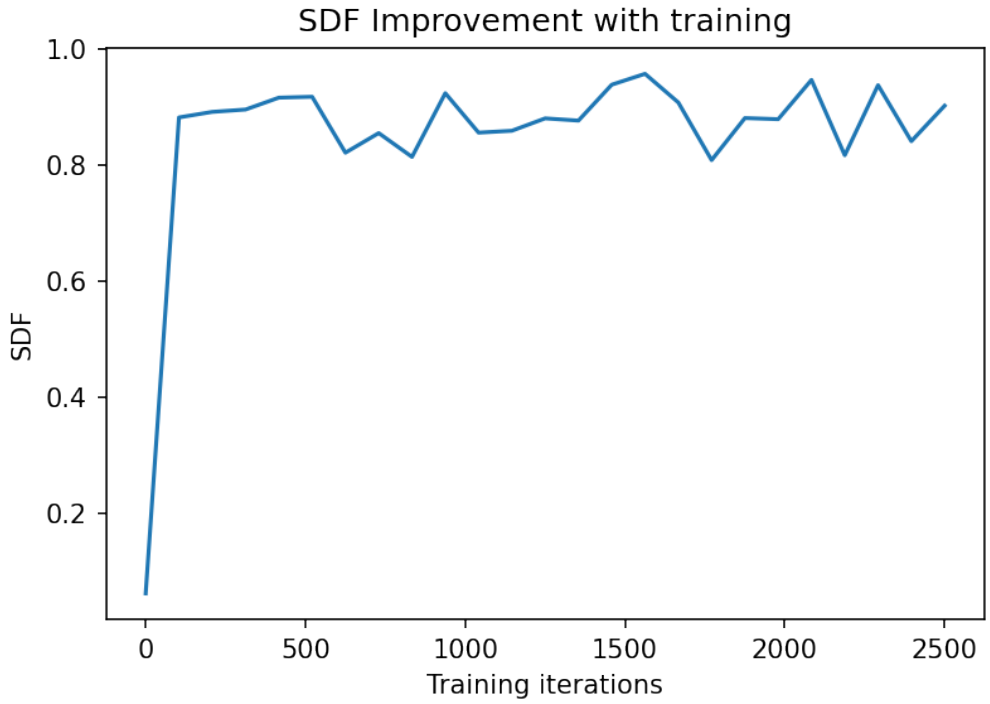

Figure 4.13: We see here that the SDF of the spill has significantly improved with the PID gain being optimized with every training iteration.

non-linear process, and the quad ramp curve is non-linear in different parts within one spill (as shown in Figure 4.15).

We thus divide the spill into four subdomains, with each of the subdomains have three paid gain values. Even though it is atypical for a PID controller to change its gain values in the middle of the process, it is not completely unwarranted given the non-linearity of the process. And this was validated with the training simulations, where we see an improved performance in the SDF when the spill is temporally split into subdomains. The gain value at the end of 1000 training iterations for a sample training is shown in Figure 4.16.

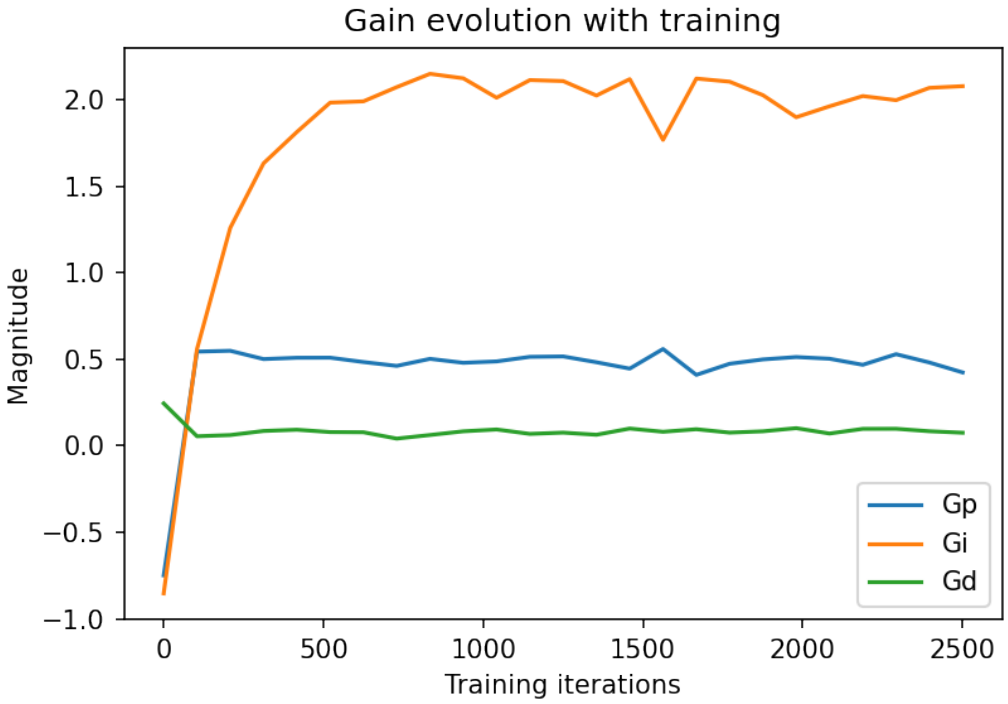

Figure 4.14: The PID gain values are optimized with every training iteration to give the SDF performance improvement shown in Figure 4.13.

4.4 ML Agent Replacing PID Controller

So far in our discussions, we have borrowed and leveraged ML techniques only to optimize the PID gain values. But here we investigate the possibility of a machine learning agent completely replacing the PID controller with the aid of supervised learning.

4.4.1 Recurrent Neural Network

In order to consider replacing PID controller with an ML agent, it is useful to consider a special class of neural networks called the Recurrent neural networks (RNN). RNNs are,

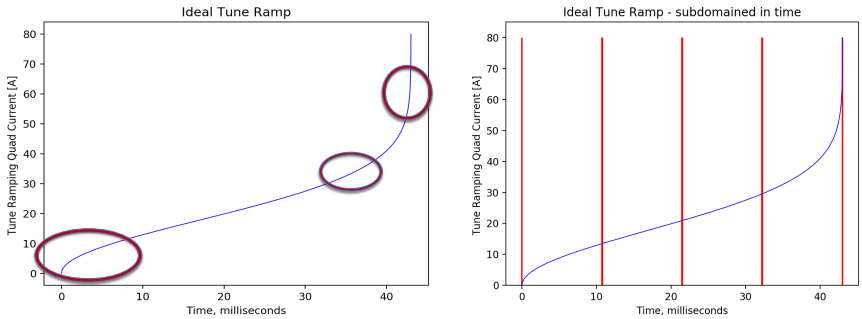

Figure 4.15: Since resonant extraction is especially more non-linear in some parts even within one spill, we divide the spill into 4 subdomains and ascribe three gain values for each of those subdomains of the spill.

in general, exceptionally capable of processing training data that contains time-series information or sequential data. If the training data is labeled in such a fashion that the input at time step t is sensitive to the input give in the previous time-step $t-1$, then we cannot use a normal neural network since it is usually capable of only feedforwarding with no memory retention capability to remember any of training data that was passed through the network in the previous time step(s).

In order to alleviate this handicap, RNNs have a special capability of not just feedforwarding but also *retaining* some of the information from the training data. This helps the network to have a more well informed output than otherwise. This is one of the reasons why RNNs have become popular in natural language processing in very recent times as it would be crucial for the neural network to remember patterns of sequences of words (gender con-

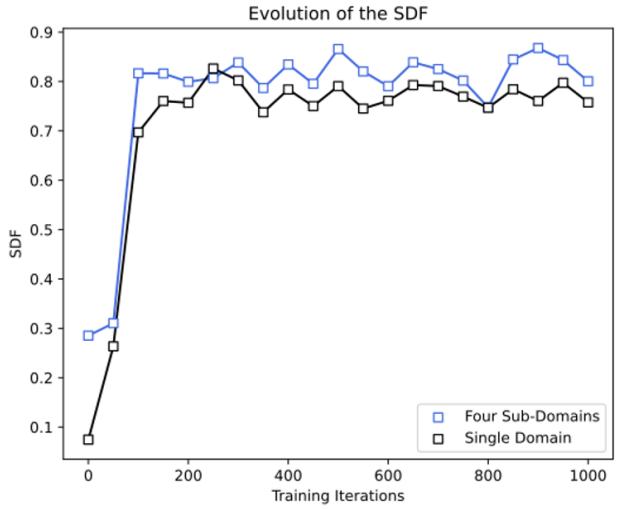

Figure 4.16: Evolution of SDF values when the spill is divided into subdomains over 1000 training iterations [38]. We see that the spill regulation is more efficient when the spill is divided into subdomains as against using just three gain values for the full spill.

sistency, tense consistency, etc.,) to make an ML agent possible to understand the grammar and process language,.

Recurrent neural networks are a natural choice to be considered to replace a PID controller in our case because the physics process of resonant extraction is also sequential and time-series in nature. Let us suppose that there is a rising spike in the spill rate as the spill proceeds. This rise would be embedded in the time-series data and we would like the ML agent to immediately recognize this noise signature and send out an appropriate control signal to curtail the noise. The neural network must thus have the capability to *remember* and process the spill data in the previous time-steps to first of all understand there is a rise in noise, and make an informed decision about what control signal to generate.

The way a vanilla RNN remembers parts of the training data is through having a feedback loop within the hidden layers of the neural network. In addition to the weights, biases, and input node values, there is another layer called a 'recurrent layer', and this layer would recurrently be repeated k number of times, where k is the number of time steps we would

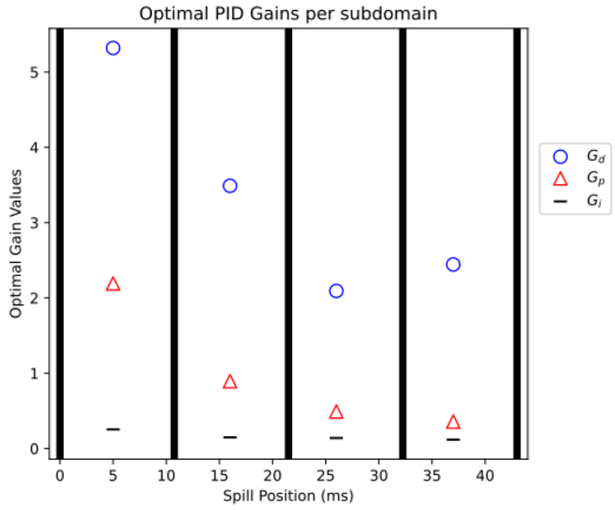

Figure 4.17: Evolution of gain values when the spill is divided into subdomains over 1000 training iterations [38], with each subdomain consisting of three gain values.

want to remember the past training data for. The memory is stored in another kind of node called a 'hidden state'.

Even though vanilla RNN can be powerful, they can typically suffer from what is called a 'vanishing gradient' problem or 'exploding gradient' problem. The way this happens is that, since the recurrent layers are repeated k number of times, as we backpropagate through the network to update the weights, we recall from the backpropagation calculation from the previous section that the new updated weights depends on the gradient of the loss function with respect to the weight value. However, if the hidden layer is rolled on to itself k number of times, there would now be factors in the chain of differentiation where they are exponentiated to the power of k. If the initialized weights $W > 1$, then this could result in an explosion of the gradient value since it is proportional to W^k, leading to an enormous increase in the weights that would result in non-sensical training. And if $W < 1$, then this would result in the gradient value almost *vanishing* to zero, in which case the weights would barely change at all, leading to effectively zero training.

In order to overcome the vanishing/exploding gradient problem in RNN, we have special kinds of RNN called Long-Short Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks.

4.4.2 Long-Short Term Memory (LSTM) Neural Network

In LSTM, we have the feedback loop that a RNN would, but in a more evolved version than the one in a vanilla RNN. The way LSTM overcomes the problem of vanishing or exploding gradient problem that a vanilla RNN faces is by cleverly having internal mechanism that *chooses* which information to keep and which ones to throw away. This way, we do not burden the network with having to remember *every* part of the training data but only ones that are relevant to improving the training.

A schematic of a single LSTM unit is shown in Figure 4.18. The core concept of the LSTM are the cell states and the gates within the unit. The cell state acts as a pathway to transfer information all the way down the sequence chain. This can be thought of as the memory of the network. The cell states can carry the information from earlier time steps to even the present time step. This helps in reducing the effects of a short term memory which a vanilla RNN might face.

In order to decide what is added (or not added) to the cell state, we let the gates to learn the relevance of a particular information. The gates in a typical LSTM would be made of sigmoid activation function (equation (4.5) which, when multiplied, has the property of classifying what is passed through either close to 1 (which would imply 'remembering' or 'letting through' the information) or close to 0 (which would imply throwing away the information, or 'forgetting').

A single LSTM unit

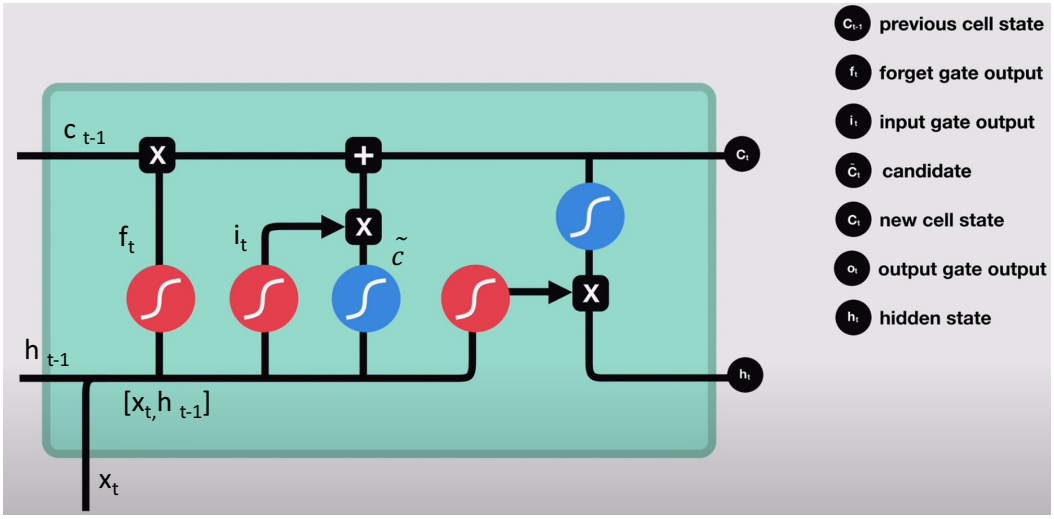

Figure 4.18: An illustration of a single LSTM unit with its internal gates that help with the vanishing/exploding gradient problem [51].

In a typical LSTM cell, we would have a cell state, a 'forget' gate, an 'input' gate, and an 'output' gate.

- The 'forget' gate inputs two quantities: the input vector x_t and the information remembered from the previous time step called the 'hidden state' vector, h_{t-1}. (The initial h_0 is set to 0 by default.) Both are passed through a sigmoid function to get a value that is either closer 1 or 0. Let us call this output f_t.

- The previous hidden state h_{t-1} and the present input x_t are also, in the same breath, passed through the 'input' gate, which is a `tanh` activation function that outputs a value between -1 and 1. This would constitute a 'candidate' vector $\tilde{c}_t$. This candidate

value is then multiplied with output of the sigmoid from the 'forget' gate f_t to decide which information to keep from the candidate vector.

- Now that we have an 'appropriately forgotten' (or 'appropriately remembered') information vector from the candidate vector, we can compute the cell state for the present time step c_t,

$$c_t = (f_t \times c_{t-1}) + (x_t \times \tilde{c}_t) \tag{4.67}$$

- We note that the newly calculated cell state for the present time step c_t also contains information from the previous time step through the h_{t-1} that was used to calculate, so we use this c_t to calculate the next time step's hidden state h_t by passing the cell state through a sigmoid function and multiplying it with the present input vector x_t and the previous hidden state h_{t-1},

$$h_t = \text{sigmoid}([x_t + h_{t-1}]) \times \texttt{tanh}(c_t) \tag{4.68}$$

- Now we have both the new cell state as well a new hidden state, both of which would be fed to the next LSTM unit in the next time step along with the new input vector x_{t+1}.

4.4.3 Toy control system with LSTM

Both as a learning exercise and also as a means to validate LSTM for control systems application, the author devised an LSTM learning scheme to control an Arduino heat reservoir.

The 'plant' system consists of a reservoir, a heater, and a thermometer to measure the reservoir's temperature, all of which can be interfaced with live using an Arduino board. The heater can take an almost continuous value from 0 to 100 % (with a resolution of 0.1 %), and the power ramp for the heater can be controlled at every time step. The set point value for the reservoir temperature as a function of time can be predefined, and we could write a logic to give the appropriate power to the heater in order to achieve the predefined set point values of the reservoir.

Since the Arduino board could be interfaced with Python, a PID routine was written in order to regulate the temperature of the reservoir. The Python script can read the temperature value from the reservoir at every time step, calculate the error value, process the error into a PID logic and give out a control signal to the heater.

The Arduino board was run for 6 hours with the set point of the reservoir changing every 15 minutes. The time series data of the reservoir temperature, control signal from the PID logic, and the heater value were stored so that an LSTM machine learning model could be trained to emulate the PID controller.

After successfully training the ML model using LSTM network, we see in Figure 4.19 that the machine learning agent is performing the regulation of the reservoir temperature very similar to the PID controller's performance.

Our goal is to train a machine learning model very similar to the above that would not just emulate a PID controller but also perform better than a PID controller.

4.4.4 Gated Recurrent Unit

In the family recurrent neural networks, there exists another class of network very similar to LSTMs called the 'Gated Recurrent Unit' (GRU).

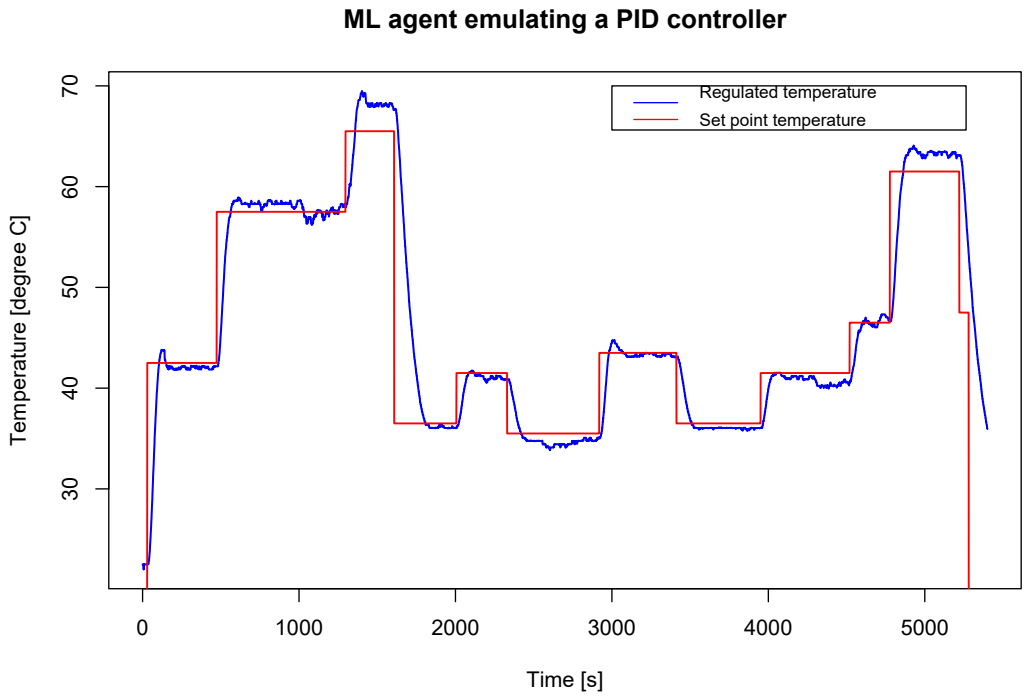

Figure 4.19: We see that the ML agent trained with LSTM network emulates a PID controller and controls the heater magnitude by itself to set the reservoir temperature to the predefined set point value.

The skeleton of GRU is similar to that of LSTM, but here we do not use a cell state but just the hidden state where the necessary information is encoded to. A GRU unit contains an 'update' gate and a reset gate. The 'update' gate acts similar to the combination of 'input' gate and 'forget' in an LSTM. This gate decides which information to throw away and which information to retain. The reset gate decides how much of the past information to remember. GRUs typically have relatively less tensor operations (since it does not have as many gates as an LSTM unit does), and thus it can be relatively faster to train compared to LSTM. The schematic of a GRU is shown in Figure 4.20.

In this work, we use a GRU network to train the ML agent to regulate the noise in the spill rate.

The workflow of the training regime is as follows:

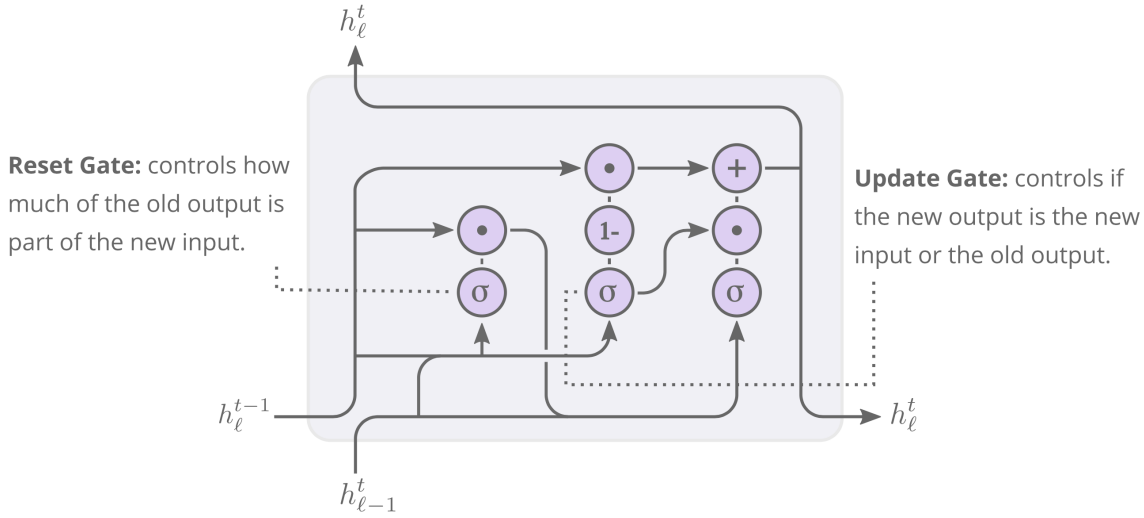

Figure 4.20: Schematic of a Gated Recurrent Unit (GRU). Here, unlike the LSTM unit, we do not use a cell state but we code the necessary information to be retained using just the hidden unit.

- The learning parameters, including the k number of time steps the model looks back to retain information of the spill, is defined. All the gradient values are cleared to simulate a new spill.
- In the first training iteration, a random noise is generated by the physics simulator.
- As the spill proceeds, the ML model ingests the previous k time steps' spill data.
- After processing the k time steps' spill data, the GRU model outputs the quadrupole current value to be set for the next time step. The physics simulator inputs the quadrupole current and produces the spill for the next time step.
- This happens iteratively until one full spill gets over, after which the SDF of the regulated spill is computed and the loss function is calculated (which is (1-SDF)^2).

- With the loss function, we backpropagate the error through the network to update its weights.
- The next spill with a completely new noise in the spill rate commences, and the whole procedure is iterated until the loss function goes to zero.

4.4.5 Supervised Learning Results

With the investigation of using GRU models to train an ML agent to regulate the spill, we report here that the ML model not only performs comparable to PID controller's regulation but also is capable of outperforming the PID controller after sufficient training. Here we use a window size of 40, whereby the ML model, at time step t_n, learns and generates the current to be supplied to the quadrupole on the t_{n+1}-th time step. (Since the window size is 40, the regulation starts to happen only after the 40th time step. However, this could be operationally mitigated by using the PID controller for the first 40 time steps and switch on the ML agent's control from the 40th time step till the end of the spill.)

In Figure 4.21, we compare the performance of the PID controller with ML controller by plotting the ratio of the improvement in SDF by the ML agent to that of the PID controller. If the ML agent perform to the same level of regulation as the PID controller, that would correspond to the value of 1. And we see in the figure that the ML agent starts outperforming the PID controller after just 400 training iterations. (The gains of the PID were tuned using the optimization techniques mentioned in section 4.3.)

In Figure 4.22, we see a sample of the spill where the regulation of the ML is clearly better than that of the PID controller.

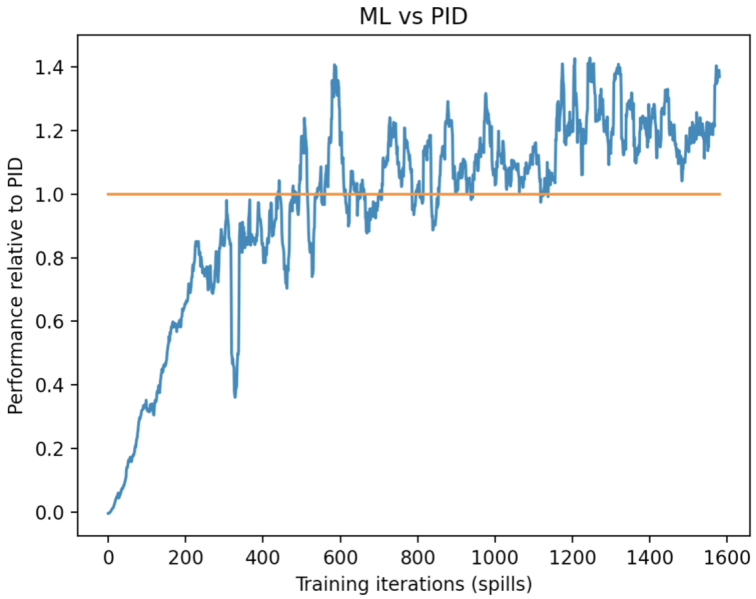

Figure 4.21: The relative performance of ML regulation vs PID regulation [39]. The blue line denotes (SDF_{ML} - SDF_{noise})/(SDF_{PID} - SDF_{noise}).

4.5 Reinforcement Learning - Future Direction

So far we have examined using supervised learning to both improve the PID controller's performance and also entirely replace the PID controller with a machine learning agent. We next look into exploring another field of machine learning called the 'reinforcement learning' to investigate if it can be used to further better a PID controller. Since the reinforcement learning efforts in spill regulation is fairly nascent at the time of writing of this thesis, we report here broadly the main RL approaches one could take for regulation, along with some initial results.

Reinforcement learning is typically performed by making an ML agent repeatedly play out a certain task (or game), either in simulation or in real life, and let the agent learn from those experiences repetitively. Even though RL can be applied to train an ML model using a real-life agent, it is typical to build a simulator to simulate the environment and the

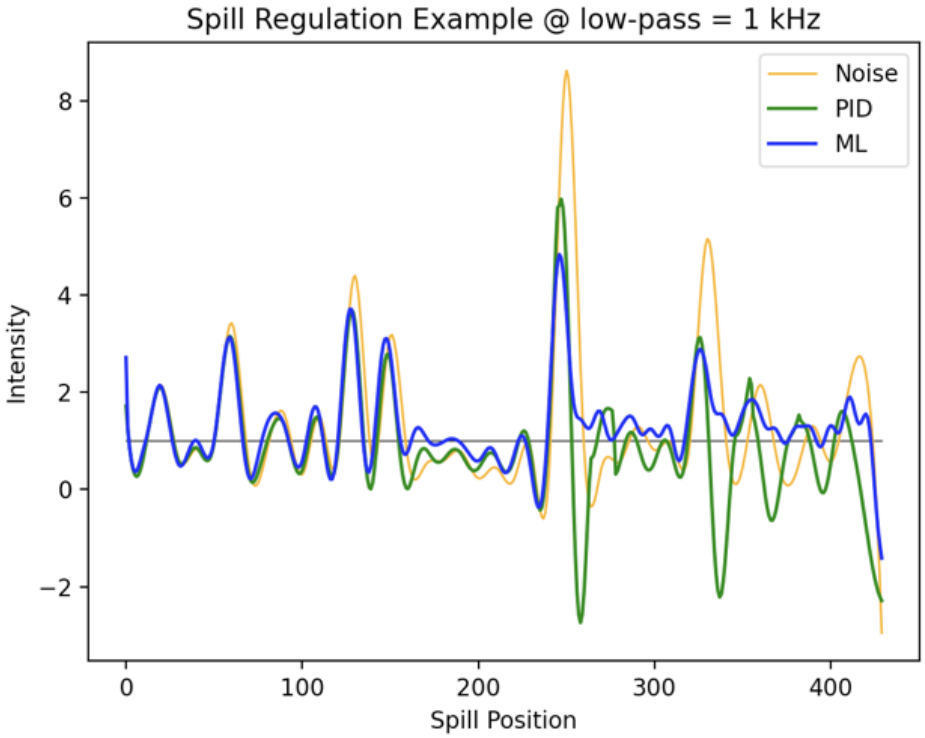

Figure 4.22: A single spill sample - performance of ML regulation vs PID regulation [39].

agent's effects on the environment since the training iterations can take an enormous amount of computation time, which would prove impractical. There is also an added advantage in using a simulator because the agent is going to get the task wrong more number of times than it gets right in the initial phase of learning, which could practically prove disastrous when used in real-life scenarios.

Unlike in supervised learning, there is no 'supervisor' in reinforcement learning. The learning process is centered around reward signal, which we ascribe to the actions taken by the agent. In reinforcement learning, unlike the classic supervised learning, the training data is, in some sense, generated by the agent itself through interacting with the environment. Thus the training data is heavily dependent on the action taken by the agent, irrespective of training algorithms used in RL.

4.5.1 Markov Decision Process

The basic RL process can be modeled a Markov Decision Process (MDP). An MDP is a discrete-time stochastic control process where the current state of the system does not depend upon the history of *how* the agent finds itself in that state. An MDP is mathematically defined as a 4-tuple entity, consisting of:

- The state of the environment in which the agent is functioning: S
- The set of action taken by the agent of interest: A
- The probability that a certain action a will take the system from state s to s': $P_a(s|s')$
- The immediate reward (or penalty) received by the agent for having taken a certain action a that takes the state from s to s': $R_a(s, s')$.

Before the training begins, we define all the parameters and rules of the environment in which we are going to train the agent to achieve a specific task. When an episode of training begins, the agent plays out the game. For each action taken by the agent at every time step during the game, we observe the change in the environment. If the change is in the correct direction, we assign a positive reward to the agent to have taken that action. If the environment changes in an incorrect direction, then we ascribe a negative reward to the agent for that time step. If the agent's action are continually wrong, leading to a constant accumulation of negative rewards, we could assign a threshold of negative value beyond which we can terminate the present training episode to start the next training episode.

In order to train an RL agent to regulate the spill, we must first formulate the resonant extraction process as a Markov Decision Process. In this case, the state space would constitute a set of variables that collects the spill data till that point in the spill. The action

space would constitute the control signal given out by the ML agent. The reward function would be shaped in such a fashion that it is maximized when the performance of the spill regulation is maximized.

4.5.1.1 Bellman Equation

Let us suppose we have an ML agent to train in a given environment. If we assume that we know the expected reward value for each action available to the agent, we would quite naturally opt to choose those set of actions for which the reward is maximum. This accumulated reward value for a given action-state pair is called the Q-value.

If the agent is in state s and selects an action a, the Bellman equation states that the Q-value gotten from having chosen that action is the immediate reward for that time step $r(s, a)$ plus the highest possible Q-value that the agent could get from transitioning to a new state s'.

$$Q(s, a) = r(s, a) + \gamma\left(\max\Big[Q(s', a)\Big]\right) \tag{4.69}$$

The factor γ here is called the discount factor and helps to control the value of rewards accumulated from actions that are from distant future so that we prioritize actions that leads to rewards in the *near* future.

4.5.2 Q-Learning

The basic approach behind Q-learning is to train an RL agent to choose a set of actions such that the Q-value mentioned in equation (4.69) is maximized. A typical way to do this

is to store all the action values taken at different states of the environment and tabulate how much reward has been scored for the respective sets of action-state pair. Having the state and action values stored, we can then use the Bellman equation to update the Q-value.

However, if the agent keeps choosing only those actions for which the Q-value is the highest, then it runs the risk of getting stuck only to those actions, without ever knowing if a better action exists outside that realm. In order to overcome this bump, we use what is called 'exploration', whereby we force the agent to also look at random actions that could result in an even improved performance.

4.5.2.1 'Deep' Q-learning

If we use Q-learning to simply update the Q-value table at every time step, there is a risk of a tremendous increase in the number of states. For a game as simple as, say, Pacman, the agent has only 4 possible actions (move left, right, up, or down), but we can imagine that the number of possible states that could be achieved just with those 4 possible actions already becomes too large. And this, in turn, would make the Q-value table too big to be meaningfully used to train an RL agent.

The way we overcome this is by replacing the Q-value table with a *neural network* which would act as an approximator to the Q-value table.

$$Q(s, a) \rightarrow Q(s, a; \theta)$$

, where θ denotes the parameters of the neural network (such as its weights and biases).

We now hope to train the parameters of this neural network (also called the 'Deep Q-Network' or DQN). The goal of this approach is to tune the parameters of DQN such that it approximates the Q-table as close as possible. This gives us a natural objective function,

$$\text{Cost} = \left[Q(s,a;\theta) - \left(r(s,a) + \gamma \left(\max \Big[Q(s',a) \Big] \right) \right) \right]^2 \tag{4.70}$$

This would serve as the error function with respect to which we can update the DQN's parameters. The key difference between this objective function and the one in the supervised learning case is that in the supervised learning the training data is a constant at every iteration. But here, both first term as well the second term is predicted by the neural network.

4.5.2.2 DQN Training

As the RL agent plays out a game, the DQN would suggest a best action value to take. This action would lead to a state, and we store the state, action, and the reward obtained for that particular action. This continues until the termination condition is achieved. We decide on a certain batch size n_b, and after every n_b number of records are recorded, we then select *randomly* the stored action, state, and reward value to update the neural network parameter. These are called memory buffers, and also are referred to as memory replay (or experience replay).

If we hope to use the DQN to approximate the Q-table, we then expect the network to be inputted with the state and action information and return one value as the output, the Q-value. However, this could be impractical because we would like to know the Q-value of not just one action but for various different possible actions. So the output of the DQN could be more than one possible Q-value, which helps in subsequent training of the network.

However, there could be some drawbacks in using DQN.

- If the number of possible actions available to the agent is high, then this leads to a combinatorial explosion in the possible actions to choose from, and the DQN becomes very challenging to train.
- If the action space is continuous instead of discrete, this approach becomes essentially hopeless since the action value has to be drawn from a pool of infinite choices.
- The exploration could be poor even if the two Q-values are similar in value.

Of these factors, the one that is most pertinent to the resonant extraction process is that the action space is continuous. This is because the nature of the noise in the spill rate is continuous, and thus the control signal to mitigate this noise would also take on a continuous value (i.e., any real number).

In order to tackle this, we look into what is called 'policy gradient' algorithms.

4.5.3 Policy Gradient Methods

In the Q-learning algorithm, we train the agent by making it find the goodness of action for a given set of action and state (also called *action-value function*) and we maximize that goodness by updating the network parameters using the Bellman equation.

Training $\rightarrow$ Get new weights and biases $\rightarrow$ compute Q-value $\rightarrow$ Get new policy π

where the policy π is defined as the probability of a certain action to be selected. In the DQN approach, the algorithm would always prefer to choose that policy for which the Q-value is the highest.

This could be cumbersome since it involves calculating the Q-value at every training iteration. However, we can bypass this by simply ignoring explicitly computing the Q-value and directly get the new policy π:

$$\text{Training} \rightarrow \text{Get new weights and biases} \rightarrow \text{Get new policy } \pi$$

This way, the neural network directly learns the policy, i.e., it outputs the probability of which action to make the agent do. Even though the above is schematic, it remains unclear how the weights of the neural network would be updated if the Q-value is not explicitly calculated.

The reader might recall from equation (4.60) that a neural network's parameters θ are updated in the following fashion:

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \tag{4.71}$$

where θ includes all the weights and biases of the network. The function $J(\theta)$ is typically an objective function that measures the performance of the forward propagation of the network. (In the PID optimization scheme, for instance, our objective function was the SDF of the spill.) If we define $J(\theta)$, then we have a solid prescription to update the neural network's parameters.

Since our ultimate objective is to maximize the reward values for the actions taken by the agent, they make up for an excellent candidate to be the objective function. In fact, what we like is to maximize the reward for not just *one* time step but maximize an accumulation of reward up until that point in time, i.e., the rewards summed over.

The reward at a specific time step t can be written as the product of the Q-value at t times the probability $p(a)$ to select an action a_t. Cumulatively, the total reward would be

the sum of the product of the Q-value at each time step multiplied by the probability $p(a_t)$ [52].

The policy gradient class of algorithms define the objective function $J(\theta)$ to be this total reward,

$$J = \sum_{a} \Big[Q(s_0, a).p(a) \Big]$$

The multiplication of the summation of all Q-values at a given state with the action probability is also called as the 'value function' $V(s)$, which is a measure of the expected reward from state s till the end of the game. The objective function in the policy gradient method is simply the value function at a given state s_0.

Next left is to compute the gradient for this objective function. The mathematics behind this derivation is quite involved, so we simply state the result here [3]:

$$\nabla J(\theta) = G_t.\nabla_\theta \ln \pi(a_t|s_t, \theta_t) \tag{4.72}$$

where G_t is the total accumulated reward from time step t till the end of the episode.

Thus, the neural network's parameters can be updated using the above calculated gradient value,

$$\theta_{t+1} = \theta_t + \alpha \Big(G_t.\nabla_\theta \ln \pi(a_t|s_t, \theta_t) \Big) \tag{4.73}$$

[3]The reader can refer to [53] for a detailed derivation.

4.5.4 Actor-Critic Algorithm

In the policy gradient method, we saw that the RL agent learns the policy directly without having to learn the state action pair value. It uses the aforementioned update policy (equation (4.73)) with every training iteration. The main metric with respect to which the policy gradient method is updated is the total reward accumulated by the agent because of its action for that episode.

However, this metric could prove a little misleading because, say that an agent performs very well until it does a majorly wrong action resulting in a big negative reward value. The total accumulated reward in this case will not tell the story of how well the agent had done until it hit the big negative reward because all that the network uses to update its weights is the *total* reward for the episode. This goes back to the fact that the policy gradient method does not keep tab of all the Q-values (which would exactly delineate the worthiness of each action to a set of states).

The way we overcome this handicap in the policy gradient method is by having two networks: one to learn the policy in order to output an action value, and another network to keep tab of the Q-value for the action taken. The former network is called the 'actor' network, because it gives out the action value, and the latter network is called the 'critic' network, because it measures the Q-value for that action, i.e., the goodness of the action taken by the actor network.

Thus the update rule for our network parameters would be,

$$\theta_{t+1} = \theta_t + \alpha\Big(Q(s_t, a_t).\nabla_\theta \ln \pi(a_t|s_t, \theta_t)\Big) \tag{4.74}$$

where we have replaced the earlier G_t with $Q(s_t, a_t)$, which is also a learned parameter (by the critic network).

Now that we have the critic network to look not just at the accumulated reward but also at action-specific rewards, there is still no straightforward way to determine the relative goodness of that action compared to other actions. For instance, the critic network could output that for a certain action a_1 but the network does not know if there exists another action a_2 for which the Q-value could be higher. In order to quantify the goodness of the Q-value, we define what is called the 'advantage' of action and feed it to the critic network. The advantage is given by,

$$A(s, a) = Q(s, a) - V(s) \tag{4.75}$$

$$= r(s, a) + \gamma V(s') - V(s) \tag{4.76}$$

where $V(s)$ is the state value function discussed before. The goodness of the action here would simply be the sign of the advantage value.

One of the primary advantages with the policy gradient approach, especially with actor-critic algorithms, is that they are capable of handling continuous action space.

4.5.5 Resonant Extraction as MDP

For the training process, a resonant extraction simulator environment (separate from the one used in the supervised learning) was built. This environment is compatible with Open AI Gym, which is a standardized environment structure and Python library that could be used to test different learning algorithms. The Open AI gym compatible physics environment has the following attributes:

- The environment is inherited from `gym.Env`.
- The `__init__` function has the action space and the state space declared, with the specification of whether they are going to continuous valued or discrete values.
- The environment consists of a `reset` function, which would reset the parameters of the environment so that a new spill can be started.
- The code contains a `step` function, which would propagate the spill from one time step to the next. The `step` function would return a 4-tuple object: the state of spill after one time step, the reward obtained for the action taken by the RL agent, a 'done' flag to indicate if the spill has terminated or not, and an 'info' variable that can be used to debug.

4.5.5.1 Actor-Critic Network Parameters

Since the regulation demands a continuous real number value of control signal, we try employing an actor-critic learning scheme to train the RL agent to regulate. The actor network consists of three hidden layers, containing 64 and 32 nodes respectively with a `ReLU` activation function. And the critic network consists of two hidden layers with 64 nodes each with a `tanh` activation function. The learning rate of actor was chosen to be 0.0004 with a standard deviation of 0.001. And the learning rate of the critic network was chosen to be 0.005.

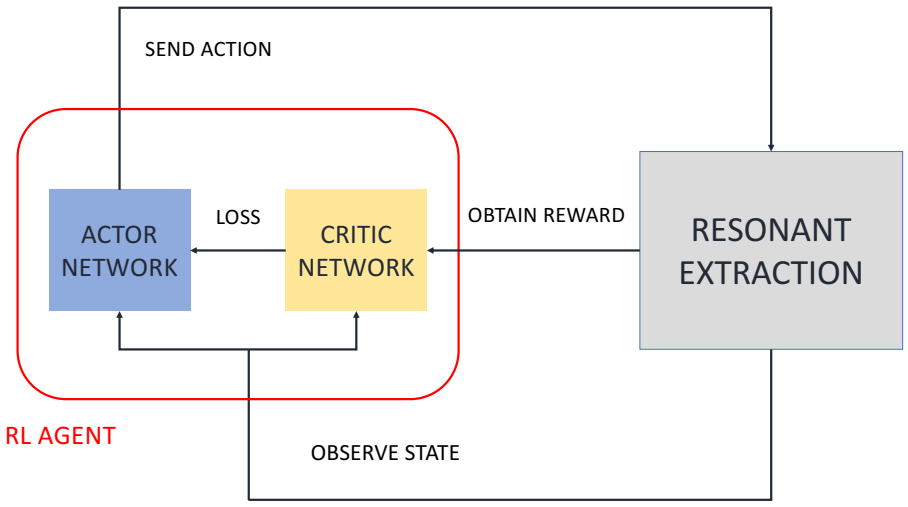

Figure 4.23: A schematic of the workflow of training procedure for an actor-critic RL agent to learn to regulate the spill intensity of resonant extraction.

4.5.5.2 Actor-Critic Training Procedure

The workflow for training an actor-critic RL agent to regulate the resonant extraction is as follows (schematic is given in Figure 4.23).

- Once the training starts, a `run` function is called upon in the environment class, whereby the `step` function is called.
- The `step` function calls the physics simulator module, and the noise for the first spill is generated. The simulator then goes through the first time step and gives out the 'bare' spill monitor output after going through all the functions mentioned in section 4.2.2.

(We say 'bare' spill monitor because the output of the first time step is unregulated and bare.)

- Looking at the spill data, depending how we have shaped the reward function, a certain reward value is awarded to the action taken in the first time step.
- The 4-tuple object consisting of rewards, 'done's, actions, and states is returned by the `step` function; this makes for one loop.
- This loop is repeated until one full spill is over, and the reward value for each time step and the state space of the spill (i.e., the spill rate intensity) is stored for every time step.
- The rewards, 'done's, actions, and states for one full spill are then passed into a `learner` class.
- The `learner` class ingests the state values, passes it to the critic network, and gets the critic value outputted by the critic network.
- The critic value is then used to calculate the advantage value from the rewards and the predefined discount factor γ for each of the action taken.
- After the advantage value of action is calculated, the state values for the full spill is now fed into the actor network, which outputs a mean value. The action is chosen from a Gaussian probability distribution that is generated with the mean suggested by the actor network (with a predefined standard deviation of 0.001). The standard deviation of this distribution decides how much the model can explore outside the suggested action value.

- With the updated new weights in the actor and the critic network, the next spill commences, with the action for each time step given by the actor network, after which the critic network provides the goodness of each action taken.
- With every training iteration, the actor network learns to provide better action, and the critic network also learns how to better criticize the actor network's action.

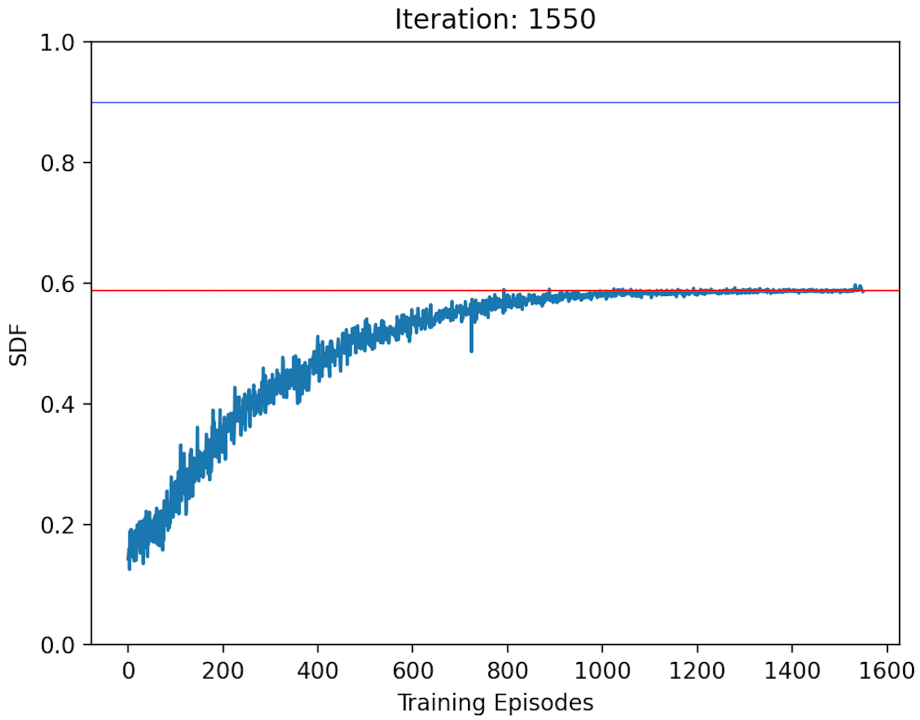

Figure 4.24: Initial result of actor-critic reinforcement learning. We see that the agent is improving its learning and regulates the spill better with every iteration.

The efforts of reinforcement learning is on-going at the time of writing of the thesis, but a preliminary result is show in Figure 4.24. Here we plot the spill duty factor of the spill for every RL training episode. The reward shaping for this scheme was chosen to be:

$$\text{Reward} = \sum_{N} \frac{1}{\text{mean}\{(\text{actual spill rate}) - (\text{ideal spill rate})\}} \tag{4.77}$$

In Figure 4.24, we see that the agent is trying to learn to increase the SDF. The red line plotted is the SDF of the noise, thus any time the blue line crosses the red, it implies a positive performance by the RL agent. In this case, we do not change the seed of the noise, since our eventual goal is to 'handhold' the RL algorithm to train to regulate by first making it to register the signatures of noise (such as the rising peak of the noise, magnitude of the noise peaks, etc.,). Once the agent learns to regulate one profile of noise, the goal is to train it with another set of noise, and it would ultimately be able to handle any kind of noise. The RL approach shown here is very promising.

Using ML could prove to be extremely powerful since control systems are ubiquitous, and the ML model can be exported to attack not just resonant extraction but possibly any accelerator control problem which could be modeled and simulated.

CHAPTER 5

TRANSIT TIME STUDIES

In this brief chapter, we investigate the transit time of particles in a third-integer resonance process. Transit time is defined as the number of turns a particle takes to get extracted once it is in the unstable region, i.e., outside the triangular separatrix that makes up the stable region. In Chapter 2, we provided an introduction to the theory of third-integer extraction. Here we review and borrow a few important results that would aid us in the transit time studies of particles during the extraction process.

The study of transit time is important because transit time determines the beam response time during resonant extraction. Knowing the transit time aprori could be very useful in designing any resonant extraction system as one has to take into account the finite time it would take for the beam to get extracted from the moment a change is induced in the knob that drives the extraction (e.g., the change in the sextupole strength or the change in tune-ramping quadrupole strength, or both, depending on what extraction scheme is employed).

In this chapter, we derive and review the analytical expression for the transit time given the initial conditions, for both static and dynamic case. The static case is when the stable region's area does not change until the particles get extracted. The dynamic case is when the separatrix is shrunk while the particle is still in transition to get extracted.

After reviewing the analytical expression, we then go on to compare the analytical transit time against the transit time gotten through extensive particle tracking (of over 4 millions particles).

5.1 Hamitonian Formalism

In section 2.1.2, we derived the transfer matrices of 1-turn, 2-turn, and 3-turn respectively in an accelerator with only linear elements, and we introduced the effect of having a single sextupole in the lattice.

We went on to derive the equation of a particle after every three turns in the presence of sextupole, and assuming that the horizontal tune goes very close to third-integer tune (29/3 in the case of *Mu2e* resonant extraction), we get the equations of motion to be,

$$\Delta X_3 = \quad 6\pi\delta Q X_0' + \frac{3}{2} S X_0 X_0' \tag{5.1}$$

$$\Delta X_3' = -6\pi\delta Q X_0' + \frac{3}{4} S (X_0^2 - X_0'^2) \tag{5.2}$$

where X_3 and X_3' are normalized phase space coordinates, δQ is the tune distance to a third-integer tune, and (X_0, X_0') are the initial normalized phase space coordinates of the particle. Since the spill duration is typically much larger than the one-turn revolution time of the particles, it is reasonable to consider three-turns as the basic unit of time instead of one-turn.

Since we have the equations of motion, we can derive the Hamiltonian by integrating the above equations of motion, and we get,

$$H = \underbrace{3\pi\delta Q\Big(X^2 + X'^2\Big)}_{\text{First term}} + \underbrace{\frac{S}{4}\Big(3XX'^2 - X^3\Big)}_{\text{Second term}} \tag{5.3}$$

The first term of the Hamiltonian represents the linear motion in the accelerator, and the second term, caused by the presence of the sextupole element, introduces non-linearity in the particle dynamics.

This non-linearity, the reader may recall, introduces a triangular stable region in the phase space. Particles inside the stable region execute stable motion and are bound inside it, never venturing out. But the particles outside the stable region start to increase their amplitude with every turn, with their horizontal position eventually surpassing and jumping past the septum position. Once the particle is past the septum, it experiences an instantaneous horizontal kick to be sent to the extraction beam line and to the desired final location.

The goal of the transit time studies is to determine the time it would take for a particle in the unstable region to reach the septum and get extracted. This study could be important because it not only gives us insights into the third-integer extraction process in addition to aiding the design of the extraction system, but it also could aid us in building better analytical models where the transit time function can be incorporated. One other reason to numerically verify the results of the transit time is that the Kobayashi Hamiltonian mentioned above only includes first-order in δQ. It could thus prove important to verify the goodness of this Hamiltonian so it can reliably be used for future modeling of resonant extraction.

5.2 Finding the Stable Region

In order to find the stable region in the phase space, we first have to find the vertices of the stable region's triangle. Since we know that the vertices are stationary points, the particle dynamics at those three points shall remain unchanged. We can thus use this fact and set equations (5.1) and (5.2) to zero.

$$\Delta X_3 = 0 = \quad 6\pi\delta Q X' + \frac{3}{2} S X X' \tag{5.4}$$

$$\implies 6\pi\delta Q X' = -\frac{3}{2} S X X' \tag{5.5}$$

$$\implies \boxed{X = -\frac{126\pi\delta Q}{3S}} \tag{5.6}$$

And,

$$\Delta X_3' = 0 = -6\pi\delta Q X_0' + \frac{3}{4} S (X^2 - X'^2) \tag{5.7}$$

$$\implies 6\pi\delta Q X = -\frac{3}{4} S (X'^2 - X^2) \tag{5.8}$$

$$6\pi\delta Q \left(\frac{-12\pi\delta q}{3S} \right) = -\frac{3}{4} S \left(X'^2 - \frac{24\pi\delta Q^2}{9S^2} \right) \tag{5.9}$$

$$\frac{288\delta Q^2}{9S^2} = X'^2 - \frac{144\delta Q^2}{9S^2} \tag{5.10}$$

$$X'^2 = \frac{288\delta Q^2}{9S^2} + \frac{144\delta Q^2}{9S^2} \tag{5.11}$$

$$X'^2 = \frac{432\delta Q^2}{9S^2} \tag{5.12}$$

$$\implies \boxed{X' = \pm \frac{12\pi}{\sqrt{3}} \frac{\delta Q}{S}} \tag{5.13}$$

Plugging these into the Hamiltonian itself, we find the Hamiltonian's to be,

$$H = 3\pi\delta Q\Big(X^2 + X'^2\Big) + \frac{S}{4}\Big(3XX'^2 - X^3\Big) \tag{5.14}$$

$$= 3\pi\delta Q\Big(\Big(-\frac{126\pi\delta Q}{3S}\Big)^2 + \Big(\pm\frac{12\pi}{\sqrt{3}}\frac{\delta Q}{S}\Big)^2\Big) \tag{5.15}$$

$$+ \frac{S}{4}\Big(3\Big(-\frac{126\pi\delta Q}{3S}\Big)\Big(\pm\frac{12\pi}{\sqrt{3}}\frac{\delta Q}{S}\Big)^2 - \Big(-\frac{126\pi\delta Q}{3S}\Big)^3\Big) \tag{5.16}$$

$$= \frac{3456\delta Q^3}{18S^2} - \frac{13824\delta Q^3}{108S^2} \tag{5.17}$$

$$H = \frac{8(6\pi\delta Q)^3}{27S^2} \tag{5.18}$$

Plugging this into the RHS of the Hamiltonian (equation (5.3)), we find that the Hamiltonian factorizes neatly into,

$$\left(\frac{S}{4}X + \pi\delta Q\right)\left(\sqrt{3}X' + X - \frac{8\pi\delta Q}{S}\right)\left(\sqrt{3}X' - X + \frac{8\pi\delta Q}{S}\right) = 0 \tag{5.19}$$

Solving this, we get the coordinates of the vertices (P_1, P_2, P_3) of the triangular stable region to be,

$$P_1 = \left(\frac{8\pi\delta Q}{S}, 0\right) \tag{5.20}$$

$$P_2 = \left(-\frac{4\pi\delta Q}{S}, \frac{12\pi\delta Q}{\sqrt{3}S}\right) \tag{5.21}$$

$$P_3 = \left(-\frac{4\pi\delta Q}{S}, -\frac{12\pi\delta Q}{\sqrt{3}S}\right) \tag{5.22}$$

The coordinates of the stable region are shown in Figure 5.1.

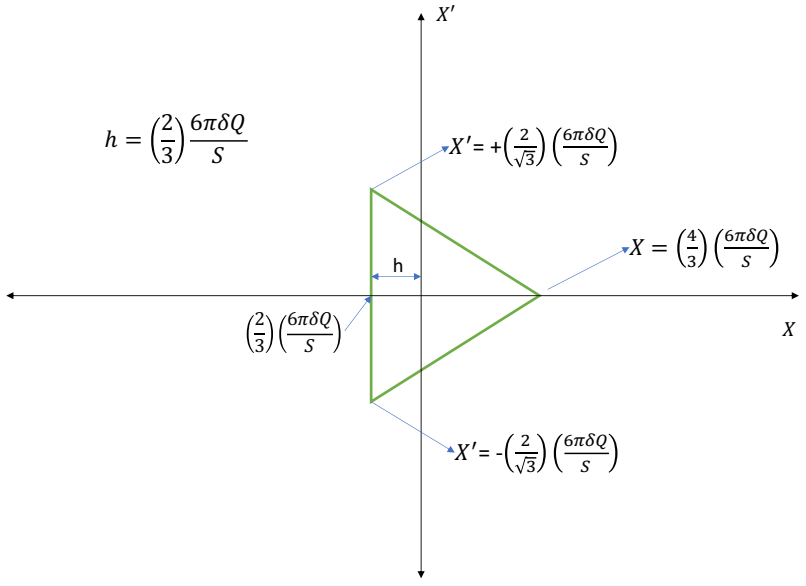

Figure 5.1: The coordinates of the stable region in the phase space.

5.3 Shifted Hamiltonian

Since we wish to calculate the transit time of particles, the calculations are slightly efficient if we shift the origin of our coordinate system to one of the vertices of the stable region [12] [1].

Let us translate the separatrix to the right in X by an amount $(2/3)(6\pi\delta Q)/S$ and translate the separatrix down in X' by an amount $(2/\sqrt{3})(6\pi\delta Q/S)$.

We can now plug in the new shifted coordinates and find the new translated Hamiltonian,

$$H_{\text{translated}} = 3\pi\delta Q\Big(X^2_{\text{translated}} + X'^2_{\text{translated}}\Big) + \frac{S}{4}\Big(3X_{\text{translated}}X'^2_{\text{translated}} - X^3_{\text{translated}}\Big) \tag{5.23}$$

$$= \frac{6\pi\delta Q}{2}\left(\Big(X-h\Big)^2 + \Big(X' + \frac{a}{2}\Big)^2\right) + \frac{S}{4}\left(3\Big(X-h)(X'+\frac{a}{2}\Big)^2 - \Big(X-h\Big)^3\right)^3 \tag{5.24}$$

$$= \frac{6\pi\delta Q}{2}\Big(X^2 + h^2 - 2Xh + X'^2 + \frac{a^2}{4} + X'a\Big) + \frac{S}{4}\left(3\Big(X-h\Big)\Big(X'^2 + \frac{a^2}{4} + X'a\Big) - X^3 + 3X^2h - 3h^2X + h^3\right) \tag{5.25}$$

where

$$h = \frac{2}{3}\frac{6\pi\delta Q}{S} \tag{5.26}$$

$$a = \text{side of the equilateral triangular stable region} = 2\sqrt{3}h \tag{5.27}$$

[1] For a more detailed derivation, the reader can refer to the PhD thesis of Marco Pullia, titled '*Dynamics of slow extraction and its influence on transfer lines design*', 1999.

Plugging these in, we get the translated Hamiltonian to be,

$$H_{\text{translated}} = \frac{S}{4}\Bigg[3hX^2 + 3h^3 - \cancel{6Xh^2} + \cancel{3hX'^2} + \cancel{9h^3} + \cancel{2\sqrt{3}hX'(3h)} \tag{5.28}$$

$$+ 3XX' + \cancel{9Xh^2} + 6\sqrt{3}XX'h - \cancel{3hX'^2} - \cancel{9h^3} - \cancel{6\sqrt{3}h^2X'} \tag{5.29}$$

$$- X^3 + 3X^2h - \cancel{3h^2X} + h^3\Bigg] \tag{5.30}$$

$$\boxed{H_{\text{translated}} = \frac{S}{4}\Bigg[3hX^2 + 3h^3 + 3XX'^2 + 6\sqrt{3}XX'h - X^3 + 3X^2h + h^3\Bigg]} \tag{5.31}$$

5.4 Equation of motion for the shifted Hamiltonian

We are now interested in the particle dynamics with the newly translated coordinates. We obtain them from the usual Hamilton's equations of motion,

$$\frac{\mathrm{d}X}{\mathrm{d}n} = \frac{\partial H}{\partial X'} \tag{5.32}$$

$$\frac{\mathrm{d}X'}{\mathrm{d}n} = -\frac{\partial H}{\partial X} \tag{5.33}$$

Plugging in,

$$\frac{\mathrm{d}X}{\mathrm{d}n} = \frac{\partial H}{\partial X'} \tag{5.34}$$

$$= \frac{S}{4}\frac{\partial}{\partial X'}\Big(6\sqrt{3}XX'h + 3XX'^2\Big) \tag{5.35}$$

$$= \frac{S}{4}\Big(6\sqrt{3}Xh + 6XX'\Big) \tag{5.36}$$

$$\boxed{\frac{\mathrm{d}X}{\mathrm{d}n} = \frac{6SX}{4}\Big(\sqrt{3}h + X'\Big)} \tag{5.37}$$

$$\tag{5.38}$$

Similarly, for the evolution of X', we have,

$$\frac{\mathrm{d}X'}{\mathrm{d}n} = -\frac{\partial H}{\partial X} \tag{5.39}$$

$$= -\frac{S}{4}\frac{\partial}{\partial X}\left(6hX^2 + 6\sqrt{3}XX'h + 3XX'^2 - X^3\right) \tag{5.40}$$

$$= -\frac{S}{4}\Big(12hX + 6\sqrt{3}X'h + 3X'^2 - 3X^2\Big) \tag{5.41}$$

$$\boxed{\frac{\mathrm{d}X'}{\mathrm{d}n} = -\frac{3S}{4}\Big(4hX + 2\sqrt{3}X'h + X'^2 - X^2\Big)} \tag{5.42}$$

Because the sextupole field is magnetic in nature, and since magnetic forces are conservative, the Hamiltonian here is a constant of motion. This implies that the Hamiltonian value for $H(X_0, X_0'; n)$ at turn n must numerically be the same as the one at a later turn $H(X, X'; n + \Delta n)$,

Using this fact, we can thus eliminate X',

$$H(X_0, X_0'; n) = H(X, X', n + \Delta n) \tag{5.43}$$

$$\begin{aligned} \implies & \frac{S}{4}\Big[3hX_0^2 + 3h^3 + 3X_0X_0'^2 + 6\sqrt{3}X_0X_0'h - X_0^3 + 3X_0^2h + h^3\Big] \\ & = \frac{S}{4}\Big[3hX^2 + 3h^3 + 3XX'^2 + 6\sqrt{3}XX'h - X^3 + 3X^2h + h^3\Big] \end{aligned} \tag{5.44}$$

$$\implies \boxed{X' = \frac{X_0^2 + \sqrt{3}X_0X_0' - X^2}{\sqrt{3}X}} \tag{5.45}$$

Substituting equation (5.45) into the equation of motion for the X coorindate (5.37), we have,

$$\frac{\mathrm{d}X}{\mathrm{d}n} = \frac{6SX}{4}\Big(\sqrt{3}h + X'\Big) \tag{5.46}$$

$$= \frac{S}{4}\left(6\sqrt{3}hX + \frac{6}{\sqrt{3}}X_0^2 + 6X_0X_0' - \frac{6}{\sqrt{3}}X^2\right) \tag{5.47}$$

$$= f(X) \qquad \text{(say)} \tag{5.48}$$

5.5 Transit Time Expression

In order to get the expression for transit time, we can simply invert equation (5.47) and find the number of turns taken by the particle to reach a certain X value.

We define transit time as the time taken by the particle to reach the horizontal position of where the septum is, X_{septum}. We can thus integrate equation (5.47) to get the transit time value,

$$T_{tt} = \int \mathrm{d}n \tag{5.49}$$

$$= \int_{X_0}^{X_{\text{septum}}} \frac{1}{f(X)}\,\mathrm{d}X \tag{5.50}$$

$$= \frac{2}{\sqrt{3}S}\int_{X_0}^{X_{\text{septum}}} \frac{1}{-X^2 + 3hX + X_0^2 + \sqrt{3}X_0X_0'}\,\mathrm{d}X \tag{5.51}$$

This integral is solvable using,

$$\int \frac{1}{ax^2+bx+c}\,\mathrm{d}x = \frac{1}{\sqrt{b^2-4ac}}\ln\left|\frac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}}\right|$$

Thus, the transit time function for a particle that is outside the stable region is given by,

$$\boxed{T_{tt} = \frac{2}{\sqrt{3}S}\frac{1}{\sqrt{9h^2+4(X_0^2+\sqrt{3}X_0X_0')}}\left[\ln\left|\frac{-2X+3h-\sqrt{9h^2+4(X_0^2+\sqrt{3}X_0X_0')}}{-2X+3h+\sqrt{9h^2+4(X_0^2+\sqrt{3}X_0X_0')}}\right|\right]_{X_0}^{X_{\text{sept}}}} \tag{5.52}$$

where we remind the reader that (X_0, X_0') is the initial coordinate of the particle, h is $(2/3)(6\pi\delta Q/S)$, and δQ is the horizontal tune distance of the beam from the resonant tune, and S is the sextupole strength. All these quantities are known to us, and thus the transit time of the particle can be computed for any particle whose initial condition is given.

As a sanity check, we see that when the sextupole strength S is zero, the transit time becomes infinity. This condition simply implies that there is no non-linearity in the dynamics and the particles are stable forever.

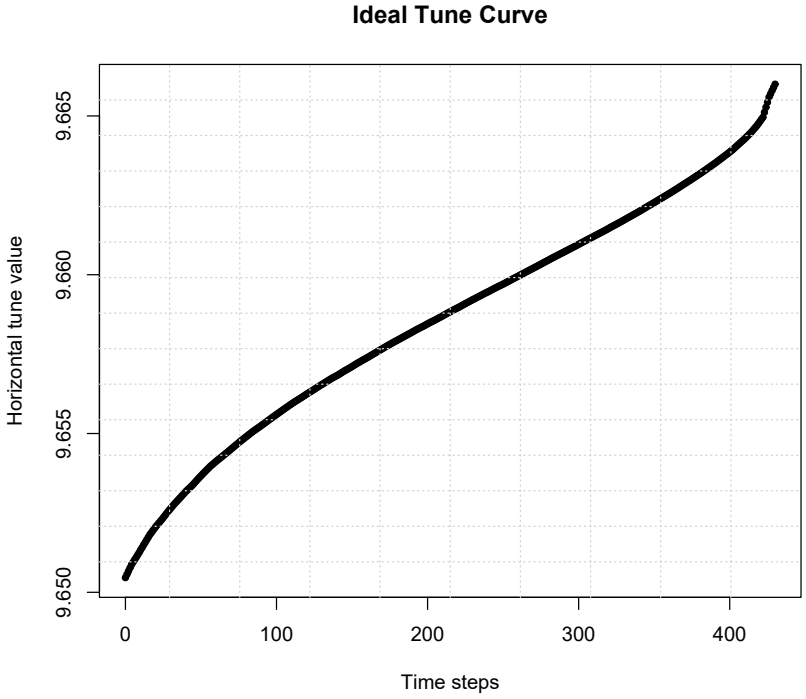

Figure 5.2: The ideal tune curve outputted by Slow Regulation algorithm, i.e., (turns vs. δQ). The initial tune ($\nu_x = 9.650$) and final tune ($\nu_x = 9.666$) values are fixed since they are design parameters.

In Figure 5.3, we have plotted the above analytically calculated transit time function for the case of resonant extraction for *Mu2e* in Fermilab's Delivery Ring. The input tune value is plugged in using the tune curve gotten by the slow regulation algorithm (plotted in Figure 5.2). The centroid of the stable triangle h value is given by $2a\sqrt{3}$, and the initial a, the side of the separatrix triangle, was taken to approximately 8 mm through visual inspection of the initial distribution (Figure 5.4). The septum location was taken to be 12 mm from the beam centre.

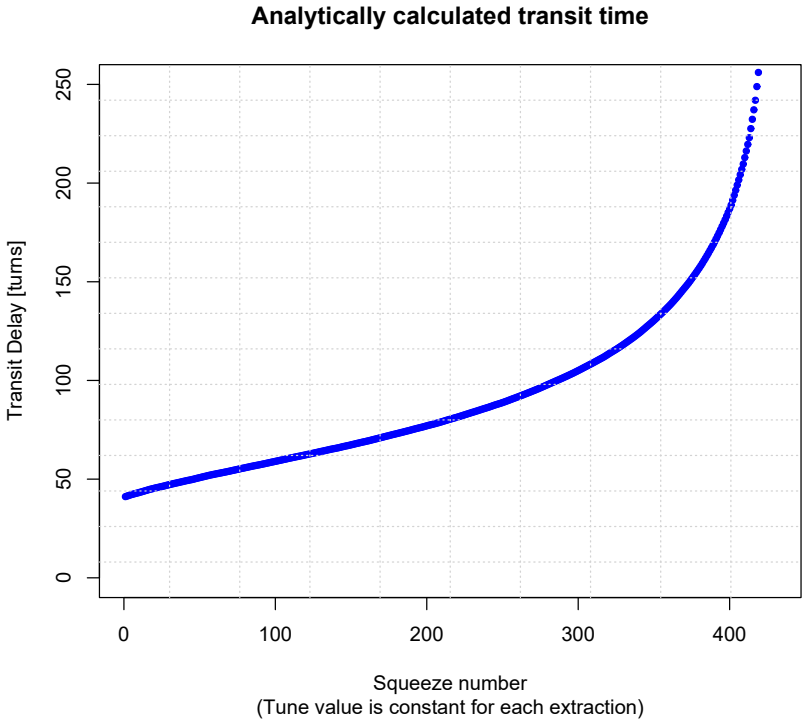

Figure 5.3: The analytically calculated transit time for the case of resonant extraction for *Mu2e* . The x-axis here denotes the static tune squeeze iteration number, and the y-axis denotes the number of turns a particle takes to get extracted from the moment the tune value is changed. Here we emphasize that the tune value is assumed to not be changed *during* the extraction.

We are next interested in validating the above expression through numerical particle tracking.

5.6 Numerical Validation of Analytically Derived Transit Time

In order to check and validate the above derived expression for transit time, extensive numerical particle tracking was done with a huge number of particle sample size of about 4 million particles. The skeleton of the numerical tracking code that was built by the author

to validate the slow regulation scheme (mentioned in section 3.3) was used and modified in order to do the transit time study.

To prepare the initial distribution of particles, we generate a Gaussian distribution of particles in both X as well as X', and the standard deviation of the distribution ($\sigma_X = \sqrt{\beta_0 \epsilon_0}$) was calculated from the design emittance for *Mu2e* resonant extraction scheme, with a beta function value of 12 m and a normalized 95% emittance value ϵ_0 of 16π mm-mrad. After we generate the initial distribution, we run the tracking simulation at a constant horizontal tune of $\nu_x = 9.650$ in order to extract the tail of the beam. This is done so that the transit time study is not affected by the surge of halo that would get extracted in the first few hundred turns. The prepared input sample is shown in Figure 5.4.

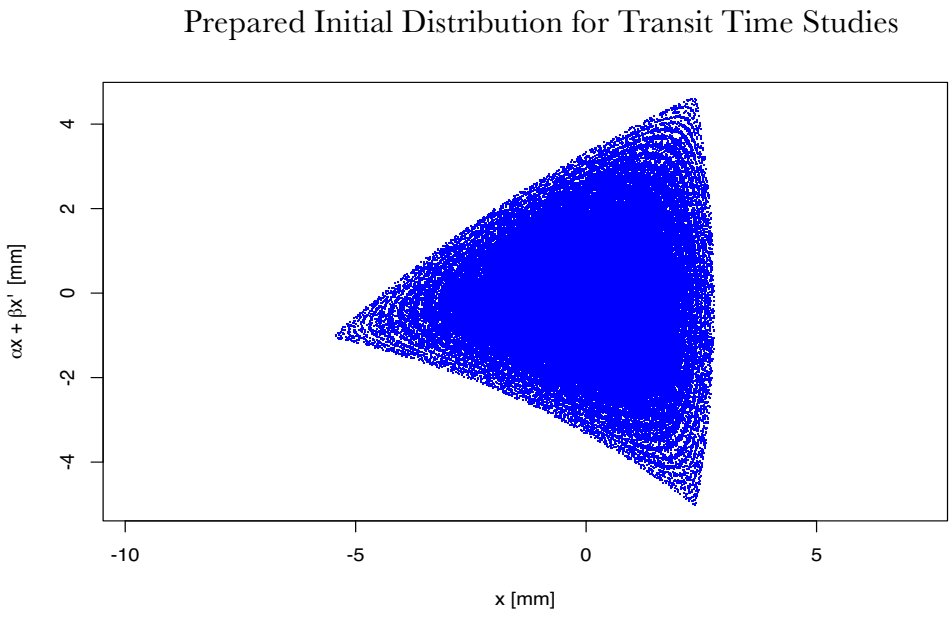

Figure 5.4: A sample of initial distribution prepared that is clear of the halo of the beam.

5.6.1 Validation Scheme

In order to validate the expression of transit time, we have to find a way to extract just a *slice* of the distribution so that the initial phase space coordinates of the *extracted* particles are reasonably known with little uncertainty (we would ideally prefer an *infinitesimal* slice of the beam, but a reasonably thin slice would be close enough). We thus perform a very slight tune squeeze, shrinking the size of the stable region by very little amount, in order to leave a thin band of particles outside the separatrix region. (An illustration of this is shown in Figure 5.5.)

Since the maximum bandwidth of the Spill Regulation System for the *Mu2e* resonant extraction is 10 KHz with a total spill duration of 43 ms, this gives us 430 data points for one spill. For the transit time study, we thus use 430 tune values (i.e., δQ values), and the extraction is performed for 1000 turns for *each* of these 430 tune values. Since the transit time expression that we had derived assumes that the tune is constant, we do not change the tune value within one iteration of extraction. (In the subsequent section, we derive the expression for when δQ itself is not a constant but *changes* with the number of turns.)

5.6.2 Simulation workflow

The simulation flow used for validating the transit time analysis is similar in structure to the one used for validating the slow regulation algorithms. However, for the case of transit time studies, we do not simulate the full spill but only a slice of the tune for each iteration. What we mean by that is, once the initial distribution is prepared, we squeeze the tune by the amount prescribed by the ideal tune curve, and at this static tune value, we perform the

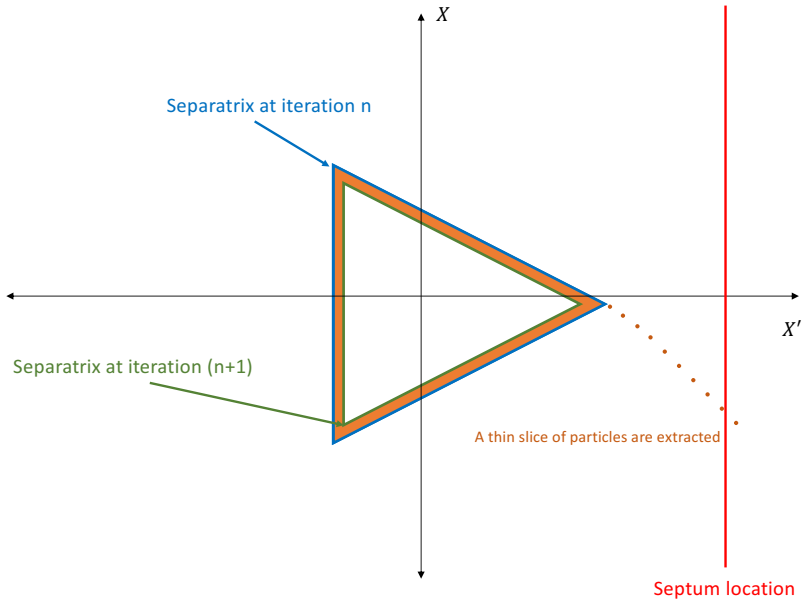

Figure 5.5: Here we illustrate the squeezing of the tune so that a small slice of particles are left in the unstable region. We can use the extraction data of these particles to validate the transit time study.

extraction for 1000 turns. These many number of turns are sufficient for of all the particles in the unstable region to get extracted (this was also verified in the simulation).

Since we are using approximately 4 million particles in the simulation, we took recourse to using the Gaea cluster Higher Performance Computing (HPC) facility at Northern Illinois University [30]. The simulation workflow is as follows:

- Two initial distribution consisting a total of $N_{\text{part}} \approx 4 \times 10^6$ number of particles is prepared, one for X and another for X'.

- The PBS script makes a request from the Gaea cluster for N_{node} number of nodes, with each node consisting of 10 active cores (one core can be though of as one CPU).

- Once the nodes (and cores) are designated for the job, the particle tracking **R** script is called by the PBS script.

- The particle tracking script divides the grand total number of particles N_{part} into N_{cores} number of cores, this way the particle tracking simulation can be done in each of the cores in parallel.

- The code runs the tracking particle-by-particle, with each particle traversing a 1000 turns at an initial horizontal tune of $\nu_x = 9.650$.

- The particles that get extracted are flagged and no longer participate in the extraction. The extracted particles' data are stored in a separate array, including the final coordinates of each particle.

- A second **R** script is called where by the tune value for the next iteration is selected. This tune value will be *slightly* closer to the resonant tune, resulting in effectively shrinking the stable region by a thin amount. This will leave a thin slice of particles outside the stable region which will be extracted (as shown in Figure 5.5).

- The PBS script calls the tracking code again, whereby the final coordinates of the particles from the previous iteration are fed as the initial coordinates, and particle tracking is done at the new tune value for another 1000 turns.

- We continue this whole iteration 430 times until the horizontal tune hits the resonant tune of 29/3 (i.e., when $\delta Q = 0$).

5.7 Results for static transit time

We report here a very good agreement of the results from the tracking simulations with the analytically computed value of the minimum transit time (shown in Figure 5.3).

We see in Figure 5.6 that there is a slight deviation from the analytically calculated value in the minimum transit times of the initial few time steps. This could be attributed to the approximations made in the analytical calculation, with the Kobayashi Hamiltonian containing only first order in δQ. We see that as we perform the studies at subsequent tune values, the agreement with the analytical calculation increases. In the last ten steps of the simulation there is a discrepancy between the theoretical and simulation. One possible reason for this could be that at the end of an iteration, few of the particles are still on their way to the septum and they get to be extracted only in the present iteration, ending up in a drastically reduced transit time.

5.8 Dynamic Transit Time

As mentioned before, the expression we derived in equation (5.52) is for how many turns a particle would take in order to reach the septum when it is outside the stable region, assuming that the stable region does not change while the particle is transitioning.

However, in real life resonant extraction, the area of the stable region would typically be changing dynamically throughout the extraction process. This implies that the particles that are outside the stable region would experience changing conditions in δQ (and possibly also in the sextupole strength S, depending on the scheme of extraction employed). This could result in a different transit time profile than one derived above.

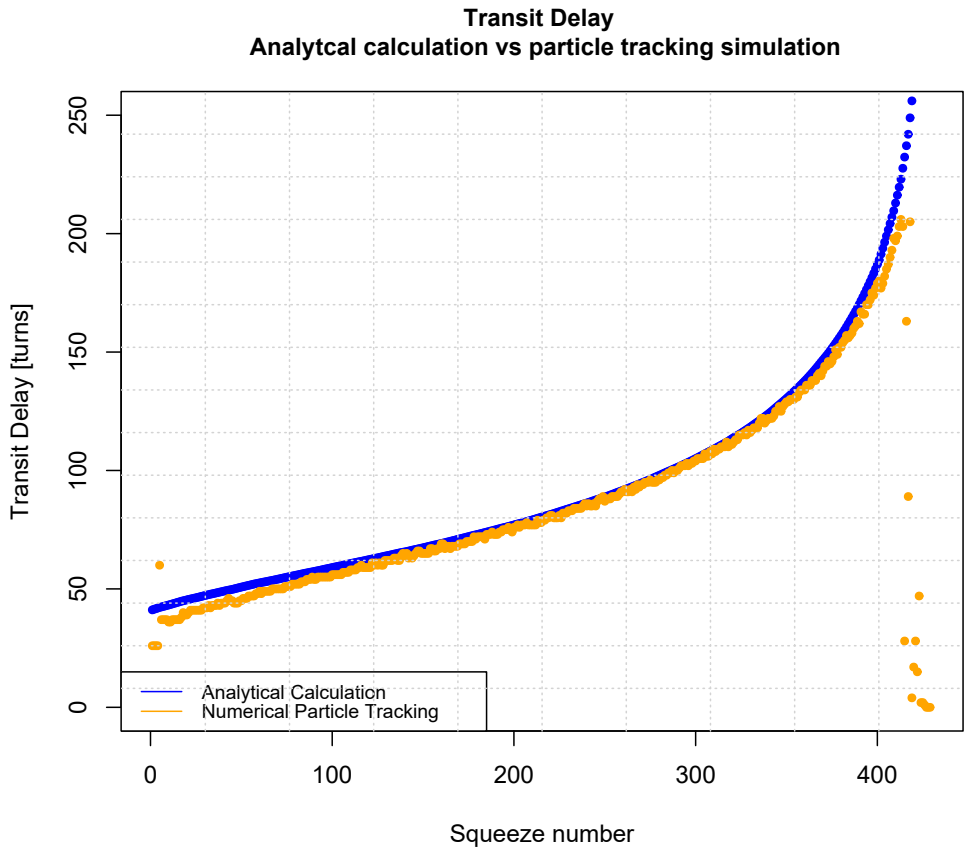

Figure 5.6: The comparison of analytically calculated for the case of resonant extraction for *Mu2e* with the minimum transit time value gotten from using the actual particle tracking.

5.8.1 Expression for Dynamic Transit Time

We borrow from the previous section's expression of evolution equation of X,

$$\frac{\mathrm{d}X}{\mathrm{d}n} = \frac{\partial H}{\partial X'} = \frac{S}{4}\Big(6\sqrt{3}hX + 6XX'\Big) \tag{5.53}$$

Since we are interested in the transit time of the band of particles that are very close to the separatrix, let us assume that the initial coordinates of the particle of interest to be $X_0 = 0$ and $X_0' = X/\sqrt{3}$ [12]. Thus the above equation becomes,

$$\frac{\mathrm{d}X}{\mathrm{d}n} = \frac{\partial H}{\partial X'} = \frac{S}{4}\Big(6\sqrt{3}hX + 6XX'\Big) \tag{5.54}$$

$$= \frac{S}{4}\left(6\sqrt{3}hX - \frac{6X^2}{\sqrt{3}}\right) \tag{5.55}$$

$$= \frac{3S}{2\sqrt{3}}(3hX - X^2) \tag{5.56}$$

If the tune value is ramped throughout the spill, then the separatrix is going to shrink at a particular rate, which implies that the centroid h of the equilateral triangle of the stable region would also shrink at the same rate. (Like in the previous static case, here too we assume the sextupole strength to be constant throughout extraction.)

The separatrix velocity, $\dot{h}$, would be given by,

$$\text{Separatrix velocity} = \dot{h} = -\frac{4\pi}{S}\frac{\mathrm{d}Q}{\mathrm{d}t} \tag{5.57}$$

where Q is the horizontal tune of the particle.

Since the separatrix would be moving *away* from a particle in the unstable region (the particle, too, would be moving away from the separatrix because of its ever increasing amplitude), we *add* the separatrix velocity to dX/dn (assuming this is the dominant effect),

$$\frac{\mathrm{d}X}{\mathrm{d}t} = \frac{\sqrt{3}S}{2}(3hX - X^2) + \frac{4\pi}{S}\frac{\mathrm{d}Q}{\mathrm{d}t} \tag{5.58}$$

Now we integrate the above equation (5.58) to get the transit time of particle in a dynamic condition:

$$T_{\text{dyn. TT}} = \int \mathrm{d}t = \int_{-X_0}^{-X_{\text{septum}}} \frac{1}{\frac{\sqrt{3}S}{2}(3hX - X^2) + \frac{4\pi}{s}\frac{\mathrm{d}Q}{\mathrm{d}t}}\,\mathrm{d}x \tag{5.59}$$

We note here that dQ/dt is independent of X.

Integrating the above result, we have the transit time function for the extraction in which conditions change dynamically,

$$T_{\text{dyn. TT}} = \frac{1}{3\sqrt{3}\pi\delta Q}\frac{1}{\sqrt{9h^2 + 4(X_0^2 + \sqrt{3}X_0X_0')}}\log\left|\frac{\left(\frac{2}{\sqrt{3}}\frac{X_{\text{sept}}}{h} - \frac{2}{3\delta Q}\frac{\mathrm{d}Q}{\mathrm{d}n}\right)\left(\frac{2}{\sqrt{3}}\frac{X_0}{h} + 2\sqrt{3} - \frac{1}{9\pi\delta Q^2}\frac{\mathrm{d}Q}{\mathrm{d}n}\right)}{\left(\frac{2}{\sqrt{3}}\frac{X_0}{h} - \frac{2}{3\delta Q}\frac{\mathrm{d}Q}{\mathrm{d}n}\right)\left(\frac{2}{\sqrt{3}}\frac{X_{\text{sept}}}{h} + 2\sqrt{3} - \frac{1}{9\pi\delta Q^2}\frac{\mathrm{d}Q}{\mathrm{d}n}\right)}\right| \tag{5.60}$$

In Figure 5.7, we plot the analytically calculated transit time for the dynamic case along with the static case. Since we know the ideal tune ramp, we calculate dQ/dt by simply taking the difference in the tune values, which would give us the rate of change of tune.

We see in Figure 5.7 that the transit times for the dynamic case is lesser than the static case. This could be explained because in the dynamic case, the separatrix is further shrinking away from the particles in the unstable region, and the velocity of the particle increases as it recedes further and further away from the separatrix.

5.9 Result for Dynamic Transit Time

In Figure 5.8, we compare the theoretically calculated transit time values for the dynamics case with the numerical particle tracking results. Here too we see that the transit time data

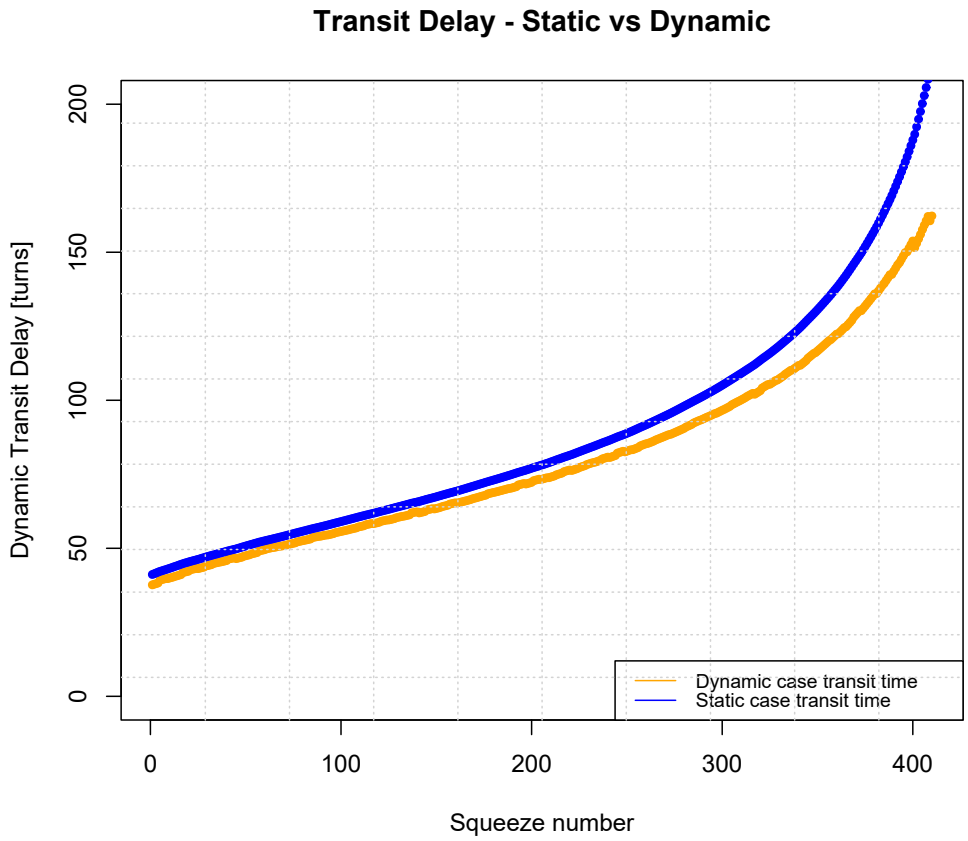

Figure 5.7: Comparison between analytically calculated static transit time vs. dynamic transit time. We see that the particles take less transit times in the case of dynamic squeeze, and this could be attributed to them experiencing more non-linearity since the separatrix is further moving away from them.

in the last ten time steps or so is misbehaving because some particles are still on their way to the septum when the previous iteration finished. But other than that factor, we see that there is a good agreement between the theoretical calculation and the particle tracking, thus validating our calculation of the transit time for the dynamic case.

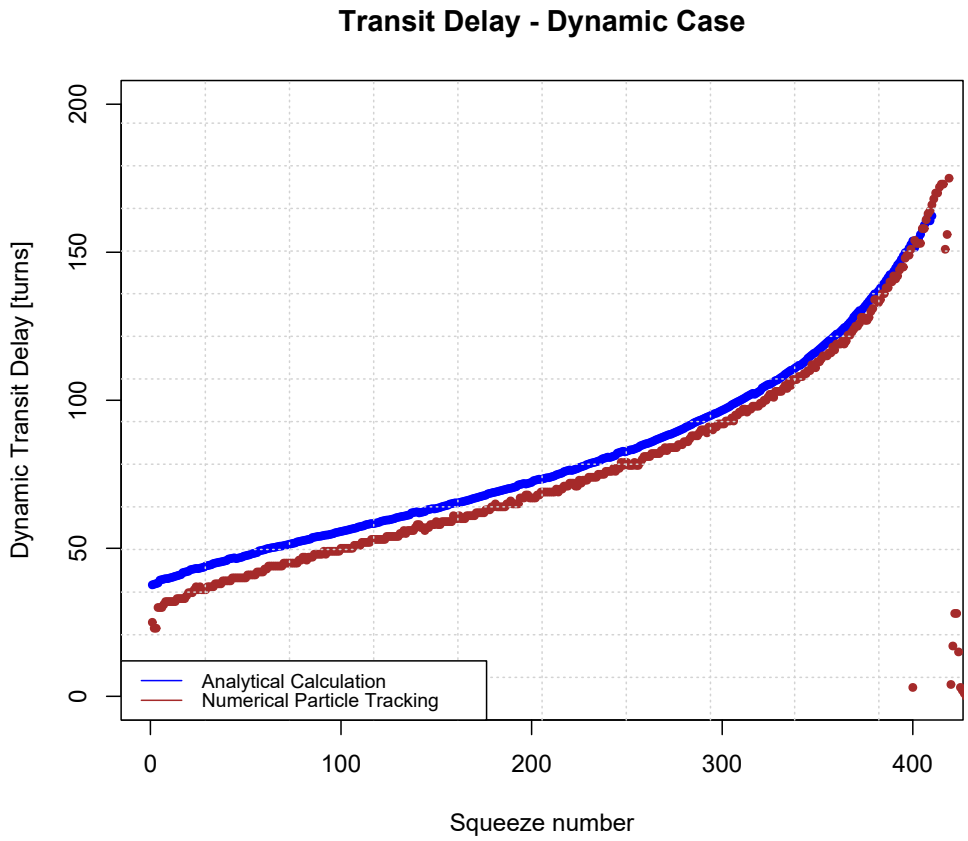

Figure 5.8: Comparison between analytically calculated dynamic transit time vs. numerical particle tracking.

5.10 Conclusion

We thus have validated the analytically derived transit time functions for both the static as well as the dynamic case, with a mean deviation of about 4.4% for the static case and about 7.3% for the dynamic case. The quantitative difference between the simulation and theoretical results could mainly stem from the fact that the Hamiltonian used do not involve higher orders of δQ and is only an approximation. But we see that the approximation is still relatively good and not too far off.

We conclude that the transit time functions validated above can be reliably used in analytical modeling the resonant extraction.

CHAPTER 6

CONCLUSION

6.1 Resonant Extraction

Mu2e is an upcoming experiment at Fermilab that intends to look for new physics in the decay of muons to electrons by capturing and letting them decay in the Coulomb field of Aluminum nucleus. In order to decrease the background effects, the experiment requires a pulsed muon beam to arrive at the Aluminum stopping target. This, in turn, is achieved by producing a pulsed proton structure hit a production target. To create the pulsed proton structure, we perform a third-integer resonant extraction in the Fermilab's Delivery Ring to extract protons and send them to the production target.(Chapter 1.)

Resonant extraction is a process by which we introduce non-linearity in the system by driving the beam delicately and purposefully into a resonance condition. Upon doing this, the horizontal size of the beam is driven into an unstable and non-linear growth, and the portion of the beam that lies past a certain horizontal position in the beam pipe is extracted at the septum location, whereby the septum gives an instantaneous kick to that portion of the beam to direct it to an extraction beam line. The extracted beam is then sent to the desired location (to the production target, in our case).

About 10^{12} protons of 8 GeV kinetic energy are extracted from the Delivery Ring over the course of 43 ms (which would be ≈25,000 turns), which is extremely short and unprecedented. We perform a third-integer resonant extraction at the Delivery Ring by fixing the strengths of the 6 harmonic sextupoles to excite the third-integer resonance and by ramping up the

strengths of the three dedicated fast ramping quadrupoles in order to drive the horizontal tune of the beam close to a third-integer resonance ($\nu_x = 9.650 \rightarrow 9.666$). (Chapter 2.)

The *Mu2e* experiment demands strict quality of extraction, we thus need a Spill Regulation System to achieve a uniform extraction rate of the protons from the Deliver Ring to the production target. The Spill Regulation System comprises of three main components:

- Slow Regulation System
- Fast Regulation System
- Harmonic Content Suppressor

In this work, we presented the contributions and results pertaining the first two components of the Spill Regulation System, i.e,. the Slow Regulation and the Fast Regulation loops.

6.2 Slow Regulation Loop - (Chapter 3)

The characteristics of the beam in the Delivery Ring could slowly change over time as the accelerator components' parameters in the ring could slowly drift over time. This could result in a low frequency noise in the spill rate that could span over many spills. In this work, we present different Slow Regulation loop algorithms that would prevent the slow drift in the rate of extraction, keeping the coarse rate of extraction as close to the ideal rate as possible.

The Slow Regulation loop works spill-to-spill by modifying the power supply ramp fed to the fast ramping quadrupoles. In this work, we presented a class of novel algorithms that would adaptively learn the ideal quad current ramp in order to achieve a coarse ideal extraction rate. Since the resonant extraction is yet to commission at the time of writing

this thesis, a numerical particle tracking code was constructed in order to simulate the resonant extraction process with the beam parameters pertinent to extraction for *Mu2e* in the Delivery Ring. The particle tracking simulator then was used to validate various Slow Regulation algorithms that were successfully able to learn, modify, and find the ideal ramp curve starting from even a linear quad current ramp (a linear ramp would result in a very poor extraction rate). The Slow Regulation algorithms were also tested successfully for various non-ideal beam conditions, such as the deviated injection emittance value, beam off set that could occur due to steering errors, and also for a beam with momentum dispersion and chromatic effects. This proves the robustness of the Slow Regulation algorithms.

Also, in all of the cases mentioned above, we conclude that the algorithms were able to achieve an extraction rate within $\pm 5\%$ of error tolerance from the ideal extraction rate, and were proven to achieve the coarse regulation even within a few hundred spills. (We remind the reader that when the actual resonant extraction process commences, we would not start with a quad ramp as bad as a linear ramp but with an analytically calculated ramp that would result in a good extraction rate; this is to emphasize that the starting point of the Slow Regulation algorithms will already be good. Thus, in a steady-state condition, and the Slow Regulation algorithms are expected to work *even more* efficiently, since they have proven to work and able to find the ideal quad current ramp starting from even a very sub-optimal quad current ramp such as the linear ramp.)

6.3 Fast Regulation Loop - (Chapter 4)

The Slow Regulation Loop would furnish us with a quad current ramp that would be pre-loaded for one full spill, which would achieve an ideal extraction rate (with $\pm 5\%$ error tolerance) in the absence of any noise in the beam elements. But in reality, we expect the

spill rate to be marred with instantaneous nosies that could arise from any of the accelerator components in the Delivery Ring. This would introduce instantaneous fluctuations in the extraction rate that could change drastically even within one spill.

In this work, we present the overview of the traditional approach of using a PID controller to mitigate the fast variations. We also presented the investigations of using of Machine Learning to enhance the Fast Regulation System. In order to validate Machine Learning models, we need to produce the necessary training data. Since particle tracking could be computationally expensive for millions of particles, we instead built a physics simulator that analytically models the resonant extraction process and gives the spill data which was used as the training data. The details of the physics simulator containing various modules and the reasonings behind them were discussed. The justification behind the analytical model approximating the resonant extraction was also presented through a geometric derivation, giving us a logarithmic curve as an approximation to the ideal quad current ramp that would result in a uniform extraction. The physics simulator was written in a way that it is able to output the spill data at every time step during a single spill, enabling it to provide efficient training data to validate ML models.

As a first step, we presented an approach to tune the gains of the PID controller using Machine Learning optimizer techniques. The gains were made to be the input nodes of a neural network and were optimized using the Adaptive Momentum (Adam) optimizer, which is a robust optimization technique that performs a backpropagation of the error value in the spill rate through the physics simulator in order to update the weights of the neural network. This is done adaptively over hundreds of training iterations, and we report a reasonably good increase in the Spill Duty Factor (SDF) of the extraction (SDF is a measure of the quality of the spill). An SDF of 1 corresponds to an absolutely ideal spill, and the target SDF for *Mu2e* resonant extraction is above 0.6. And we see in simulations that, on average, the ML optimizer is able to tune the PID controller resulting in a regulation that gives much higher

SDF value than 0.6 (we note that the average SDF of the unregulated spill is 0.5). We then present the result of the spill being divided into subdomains, and we report an even higher increase in the SDF.

We next present the prospects of an ML agent completely replacing the PID controller and taking over the regulation. We constructed a special type of neural network architecture called 'Recurrent Neural Network' (RNNs) that specialized in processing time series training data. This bodes well with our case since resonant extraction is a time-series process and the spill rate is a sequential data within one spill. RNNs have specialized loops inside them that would retain a portion of information from the training data, enabling them to remember noise signatures and send out appropriate control signal. In this work, we presented the results of a Gated Recurrent Unit, a special type of RNN network, regulating the spill all by itself. We see that the performance of the GRU ML agent not only matches the performance of PID controller but also outperforms (on average) the PID controller after a few hundred iterations. We thus conclude that Machine Learning is a very good candidate for performing spill regulation. We also present an on-going work on using Reinforcement Learning (RL) to regulate the spill, whereby the RL agent would learn to regulate the spill by playing the game (i.e., spill) over and over. Since the control signal to mitigate the fast variations could take a continuous value (i.e., any value in the real number line), we present a policy gradient approach of RL whereby the agent can give out a continuous action value. We also present the initial results of the same.

6.4 Transit Time Studies - (Chapter 5)

In the third-integer resonant extraction process, because of the excitation of the third-integer resonance, a triangular stable region is formed in the phase space. Particles inside the

stable region would remain bound and keep oscillating, but the amplitude of the particles outside the stable region would start to grow non-linearly. The area of the stable region depends directly on the square of the tune distance to the resonant tune ($\delta Q = \nu_x - \nu_{\text{res.}}$) and inversely on the strength of the sextupole. The equations of motions and the dynamics of the particles were derived through transfer matrix approach and the evolution equations of the normalized phase space variables (X, X') were integrated in order to obtain an approximate Hamiltonian for the third-integer extraction dynamics, also called the Kobayashi Hamiltonian [13].

A quantity of interest to us in the modeling of any resonant extraction process is the transit time of a particle during the extraction process. The transit time is defined as the time a particle takes to get extracted once it is outside the triangular stable region. We presented the derivation of the transit time of particles using the Kobayashi Hamiltonian as a function of the stable triangle's size, the particles initial phase space coordinates X_0, X_0'. In the first calculation, we assume the resonance conditions to remain static as the particle transits. We calculate the analytically predicted transit time for a sample *Mu2e* extraction numbers. We then performed extensive particle tracking to verify the analytical calculation, and we find an excellent agreement between the analytical modeling and the numerical particle tracking results, with an average deviation of just $\approx 4.3\%$.

We then derive the transit time function for a particle that is transiting in a *dynamic* resonant condition. Here, the separatrix keeps shrinking as the particle is still transiting. This would be a more realistic modeling because the rate of change of separatrix size would result dynamically increase the velocity of the particle as it travels towards the septum position. After deriving the transit time for dynamic resonant condition, we then calculate a sample analytical transit time values for *Mu2e* extraction. We then modify the conditions of our particle tracking code to reflect the dynamic resonance conditions and perform particle tracking to verify it with the analytical expression. We find an excellent agreement with

the analytically derived transit time function for the dynamic case as well, with an average deviation of $\approx$ 7.3%. We thus conclude that the derived transit time functions can be reliably used to analytically model any resonant extraction process, provided the extraction conditions and the parameters of the machine are known.

REFERENCES

[1] P. A. Zyla *et al* (Particle Data Group), Prog. Theor. Exp. Phys. 2020, 083C01

[2] J. A. Bistirlich *et al* Phys. Rev. C5, 1867 (1972).

[3] Bartoszek, L., & et al. (2015). Mu2e Technical Design Report. Lawrence Berkeley National Laboratory. http://dx.doi.org/10.2172/1172555 Retrieved from https://escholarship.org/uc/item/0fg2p3g5

[4] D. A. Edwards and M. J. Syphers, *An Introduction to the Physics of High Energy Accelerators*, Second Edition, 2004.

[5] S. Y. Lee, *Accelerator Physics*, Third Edition, 2011.

[6] M. Conte and W. W. MacKay, *An Introduction to the Physics of Particle Accelerators*, 2008. https://doi.org/10.1142/6683

[7] H. Wiedemann, *Particle Accelerator Physics*, Fourth Edition, 2015. DOI: 10.1007/978-3-319-18317-6

[8] K. Wille, *The Physics of Particle Accelerators*, Oxford University Press, Oxford, 2000.

[9] Department of Energy, How Particle Accelerator Work. *https://www.energy.gov/articles/how-particle-accelerators-work*

[10] P. J.Bryant, A Brief History and Review of Particle Accelerators, CAS-CERN Accelerator School: 5th general accelerator physics course, Jyvaskyla, Finland, 7-18 Sep 1992: Proceedings. 2 vol., 1-16.

[11] O. Struik and J. Dirk. Lectures on Classical Differential Geometry, Addison-Wesley, 1961

[12] M. Pullia, *Dynamics of slow extraction and its influence on transfer lines design, 1999.*

[13] Y. Kobayashi, H. Takahashi, Improvement of the emittance in the resonant ejection, Proceedings VIth Int. Conference on High Energy Accelerators, Massachusetts, (1967), p.347-51.

[14] Ernest Courant and Hartland Sweet Snyder, "Theory of the Alternating-Gradient Synchrotron", Annals of Physics, Volume 3, Issue 1, 1958, Pages 1-48, ISSN 0003-4916, https://doi.org/10.1016/0003-4916(58)90012-5. (URL: https://www.sciencedirect.com/science/article/pii/0003491658900125)

[15] W. H. Lamb and R. J. Lari, *Proceedings of international symposium on magnet technology, editors H. Brechna and H.S. Gordon, SLAC, p.487-96*, 1965.

[16] W. J. Marciano and A. I. Sanda, Phys. Lett. B67, 303 (1977), A. Rossi, Phys. Rev. D66, 075003 (2002) [arXiv:hep-ph/0207006].

[17] W. Bertl et al., Eur. Phys. J. C47, 337 (2006).

[18] A. van der Schaaf, doi:10.21468/SciPostPhysProc.2, *https://scipost.org/preprints/scipost_202103_00026v2.*

[19] R. Watanabe et al., Asymmetry and energy spectrum of electrons in bound118 muon decay, Atomic Data and Nuclear Data Tables 54(1), 165 (1993), 119 doi:https://doi.org/10.1006/adnd.1993.1012.

[20] S. Ahmed, et. al, Phys. Rev. D. Volume 38, Number 7, *Search for muon-electron and muon-positron conversion* https://journals.aps.org/prd/pdf/10.1103/PhysRevD.38.2102

[21] T. Suzuki et al., Phys. Rev. C35, 2212 (1987).

[22] Czarnecki A, Garcia i Tormo X, Marciano WJ. Muon decay in orbit: spectrum of high-energy electrons. Phys Rev D (2011) 84:013006 doi: 10.1103/PhysRevD.84.013006 (1987).

[23] A. Czarnecki, W.J. Marciano, and Xavier Garcia i Tormo, Phys. Rev. D84 (2011) 013006, arXiv:1106.4756 [hep-ph]

[24] Michel L. *Interaction between four half-spin particles and the decay of the -meson.* Proc Phys Soc A (1950) 63:514

[25] J. A. Bistirlich, et al., Phys. Rev. C5, 1867 (1972).

[26] *Fermilab Concepts - Rookie Book*, Accelerator Division - Operations Department, Fermilab, 2020. URL: `https://operations.fnal.gov/rookie_books/concepts.pdf`

[27] D. Trbojevic et al., "Improvement of the High Voltage Properties of the Fermilab Electrostatic Septa," IEEE Trans.Nucl.Sci. 32 (1985) 3074-3076. (Fermilab TM- 1318).

[28] S. Werkema. "Proton Pulse Intensity Variation", Mu2e-doc-32620-v2, April 2020.

[29] R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL: https://www.R-project.org/.

[30] https://www.niu.edu/crcd/prospective-user/index.shtml

[31] L.J. Laslett, BNL Report 7534 (1963) p. 324

[32] Michel L. *Implementation of RF-KO Extraction at CNAO* International Particle Accelerator Conference Proceedings, doi:10.18429/JACoW-IPAC2019-THPMP010

[33] Karl Johan Astrom *Control System Design*

[34] P. Prieto, "RFKO regulation for the Mu2e Resonant Extraction," Mu2e-doc-3625, December 2013.

[35] S. Zorzetti, "DIGITAL SIGNAL PROCESSING AND GENERATION FOR A DC CURRENT TRANSFORMER FOR PARTICLE ACCELERATORS," Masters Thesis, University of Pisa, 2013.

[36] P. Kasper, A. Gaponenko, E. Prebys, "WBS-475.02.08.03 - EXTINCTION MONITORING INTERFACES", Mu2e-doc-6508, 2016.

[37] T. Furukawa, et. al, "Global spill control in RF-knockout slow-extraction",

[38] A. Narayanan, M. Thieme, et. al., "Optimizing Mu2e Spill Regulation System Algorithms"

[39] A. Narayanan, M. Thieme, J. Jang, et. al., "Machine Learning for Slow Spill Regulation in the Fermilab Delivery Ring for Mu2e"

[40] MATLAB, *https://www.mathworks.com/products/hdl-coder.html#hdl-code-generation*

[41] Andriy Burkov, "The Hundred Page Machine Learning Book", 2019, ISBN: 9781999579548, 1999579542.

[42] Tom M. Mitchell, "Machine Learning: A Multistrategy Approach", 2011, ISBN: 0072299142.

[43] Nishant Shukla, "Machine Learning with Tensorflow", 2018, ISBN: 978-1617293870.

[44] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). The MIT Press.

[45] Linnainmaa, S. "The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding error," Master's Thesis (in Finnish), Univ. Helsinki, pp. 67, 1970.

[46] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[47] *https://hmkcode.com/ai/backpropagation-step-by-step/*

[48] Boris Podobedov, Lynne Ecker, David Harder, George Rakowsky, "Eddy Current Shielding by Electrically Thick Vacuum Chambers", TH5PFP083, Proceedings of PAC09, Vancouver, BC, Canada,

[49] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems, 32:8026 8037, 2019.

[50] 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

[51] Michael Nguyen, "Illustrated Guide to LSTM's and GRU's: A step by step explanation", *https://youtu.be/8HyCNIVRbSU*

[52] Shaked Zychlinski, 2019, URL: *https://towardsdatascience.com/qrash-course-deep-q-networks-from-the-ground-up-1bbda41d3677*

[53] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.), pages 325 to 327. The MIT Press. .