DISSERTATION

NEURAL NETWORKS FOR MODELING AND CONTROL OF PARTICLE ACCELERATORS

Submitted by

Auralee Linscott Edelen

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Doctoral Committee:

Advisor: Dr. Stephen Milton
Co-Advisor: Dr. Sandra Biedron

Dr. Edwin Chong
Dr. Thomas Johnson

ABSTRACT

NEURAL NETWORKS FOR MODELING AND CONTROL OF PARTICLE ACCELERATORS

Charged particle accelerators support a wide variety of scientific, industrial, and medical applications. They range in scale and complexity from systems with just a few components for beam acceleration and manipulation, to large scientific user facilities that span many kilometers and have hundreds-to-thousands of individually-controllable components. Specific operational requirements must be met by adjusting the many controllable variables of the accelerator. Meeting these requirements can be challenging, both in terms of the ability to achieve specific beam quality metrics in a reliable fashion and in terms of the time needed to set up and maintain the optimal operating conditions. One avenue toward addressing this challenge is to incorporate techniques from the fields of machine learning (ML) and artificial intelligence (AI) into the way particle accelerators are modeled and controlled. While many promising approaches within AI/ML could be used for particle accelerators, this dissertation focuses on approaches based on neural networks. Neural networks are particularly well-suited to modeling, control, and diagnostic analysis of nonlinear systems, as well as systems with large parameter spaces. They are also very appealing for their ability to process high-dimensional data types, such as images and time series (both of which are ubiquitous in particle accelerators). In this work, key studies that demonstrated the potential utility of modern neural network-based approaches to modeling and control of particle accelerators are presented. The context for this work is important: at the start of this work in 2012, there was little interest in AI/ML in the particle accelerator community, and many of the advances in neural networks and deep learning that enabled its present success *had not yet been made* at that time. As such, this work was both an exploration of possible application areas and a generator of initial demonstrations in these areas, including some of the first applications of modern deep neural networks in particle accelerators.

ACKNOWLEDGEMENTS

In seeing this dissertation to fruition, I am very grateful for the support, collaboration, and friendship of many people. First, I would like to thank Sandra Biedron and Stephen Milton for giving me the opportunity to work on this exciting topic. I could not have asked for a better topic to study as a Ph.D. student, and this area combines my interest in complex system dynamics, wonderment at the ingenious problem-solving observed in the natural world, and joy in experimenting with new ideas. Thank you for setting me on this path. I am very grateful for the opportunity. Along with Sandra and Stephen, I would also like to thank my other committee members, Thomas Johnson and Edwin Chong, for their support and feedback. I am also thankful to the Office of Naval Research, which funded the work with the FAST RF Gun; Fermilab, which funded the work with the PIP-II RFQ; and the Directed Energy Professional Society, which supported some of this work with a scholarship award.

At Fermilab, Brian Chase took me under his wing when few others were interested in AI/ML. He taught me a lot about low-level RF systems, control systems, project design, and life in general. I am grateful for his mentorship.

I am also thankful for the guidance from Daniel Bowring during work with the PIP-II RFQ. In addition, Jonathan Edelen, Jim Steimel, Brian Chase, Denise Finstrom, Dennis Nicklaus, and Maurice Ball all played critical roles in that work, and it would not have happened without their support and collaboration.

I am also extremely grateful to Alexander Radovic, Brian Nord, and Gabriel Perdue for their support and encouragement, along with the community around AI/ML that they helped to build at Fermilab. Being a part of that community (even as the orphan from accelerator physics!) kept me going. I cherish the technical discussions, enthusiasm for new ideas, and camaraderie we had in trying to push forward ML in our respective fields, at a time when AI/ML was very much not yet a “hot topic” in physics.

Within the accelerator division at Fermilab, I am also grateful for the early encouragement from Philippe Piot, Giulio Stancari, Sergei Nagaitsev, Vladimir Shiltsev, Jinhao Ruan, Jamie Santucci, Denise Finstrom, Dennis Nicklaus, Alex Halavanau, Elvin Harms, Alexander Romanov, Cindy Joe, and Jeffrey Eldred. Your interest and support helped keep me excited about what I was doing. Early support from FAST was also extremely beneficial to me and was my first introduction to accelerator operations and commissioning. The lessons learned there have been incredibly valuable.

I also would like to thank Daniel Ratner, Xiaobiao Huang, Remi Lehe, Stephen Webb, Kathy Harkay, Christopher Mayes, Jeremiah Holzbauer, Jean Luc Vay, and Ji Qiang. Although we did not work together while I was at Fermilab, the encouragement you gave me (along with your enthusiasm about AI/ML for accelerators in general) was very helpful to me when I was deciding whether to continue working on AI/ML for particle accelerators or leave the field to work on AI/ML in other areas. You are part of the reason I am still in this field, so thank you for encouraging me to stick with it.

I also owe much gratitude to Christopher Mayes, whose encouragement helped keep me motivated while I was finishing up the writing of this dissertation. His comments ranged from wry (*"how many grand-kids will you have before we can call you Dr.?"*) to practical (*"at some point you're going to realize how much more money you'll be making when it's done"*) to wise (*"get it done so you can move on"*). All were helpful and motivating.

I feel extremely fortunate to have crossed paths with Dean "Chip" Edstrom, Jr. when I first started working with Fermilab in 2013. He was instrumental to many aspects of the work shown in this dissertation. Chip also let me stay at his house during early visits to Fermilab when funding was tight, and he was extremely patient in the control room when the phrase "let's just take another few measurements!" was uttered a little too often at FAST. Chip's dedication to accelerator science, his commitment to getting things done on the machine, and his skill as an operator are all great assets. I am grateful to him both as a colleague and a dear friend. The many excursions to Limestone, the Gammon Coach House, and Chateau Chip were a cornerstone of my time at Fermilab, filled with laughs, technical discussion, dreaming up new studies, some very bad puns,

and, yes, some much-needed kvetching from all parties involved. Life in Batavia would not have been the same without Chip.

Perhaps most importantly, Jonathan Edelen was in the trenches with me from the beginning to the end of this work. We were graduate students together at CSU, we decompressed from work and built up some of our early mountaineering acumen in the mountains of Colorado together, and we later worked closely together at Fermilab. Graduate school was a difficult time in my life for a host of reasons, but Jon believed in me and encouraged me. He listened to me talk about ML and accelerators probably more than anyone would have ever cared to at the time, and when there were few others interested, he put in the time and effort to collaborate. Jon knows more than anyone else what I went through during graduate school, and he was there through it all. Words cannot express my gratitude for your friendship, collaboration, and companionship through these years.

Along with Jonathan, I would also like to thank former fellow graduate student Christopher Hall, for the many fun memories the three of us built together in the early years of graduate school.

Finally, I owe a lot to my Mom and Dad. You both encouraged the kind of wonder about the world and intellectual curiosity that led me here. To Dad, I am sorry that you never got the chance to see this work finished; I know it would have meant a lot to you.

TABLE OF CONTENTS

ABSTRACT ii
ACKNOWLEDGEMENTS iii
LIST OF TABLES viii
LIST OF FIGURES ix

Chapter 1 Introduction 1
1.1 Notes on Terminology 5
1.2 Challenges for Modeling and Control in Particle Accelerators 6
1.3 Neural Networks and Machine Learning 11
1.3.1 Basic Background 11
1.3.2 Basic Neural Network Structure and Learning 17
1.3.3 Convolutional Neural Networks 20
1.3.4 Recurrent Neural Networks 23
1.3.5 Activation Functions 24
1.3.6 Common Training Algorithms 26
1.3.7 Model Validation, Overfitting, and Regularization 29
1.4 Relationship Between AI/ML and Optimization Methods Commonly Used in Particle Accelerators 35
1.5 Historical Context: Why Are neural network-based Approaches Now Relevant 39
1.5.1 Advances in Neural Networks 39
1.5.2 Examples of Developments in Related Science and Engineering Fields . 41
1.5.3 Advances in Computing and Operations Infrastructure for Particle Accelerators 43
1.6 Neural Networks and Particle Accelerators in the Context of this Dissertation 43

Chapter 2 Accelerator System Modeling: Faster Simulations, Image-based Diagnostics, and Bridging the Gap Between Simulation and Empirical System Behavior . 46
2.1 Introduction and Motivation 46
2.2 The FAST Low-Energy Beamline 49
2.3 Incorporating Image-based Diagnostics as Model Inputs 49
2.3.1 Simulations and Data Sets 51
2.3.2 Neural Network Modeling 54
2.3.3 Conclusion and Future Work 57
2.4 Virtual Diagnostics 58
2.4.1 Motivation 58
2.4.2 Transverse Phase Space Measurements at FAST 61
2.4.3 Initial Measurements and Simulations 64
2.4.4 Scalar Prediction on Simulation Data 71
2.4.5 Updating with Measured Data (Transfer Learning) 77
2.4.6 Direct Prediction of Simulated Image Diagnostic Output 80

2.4.7 Conclusions and Future Work . . . 92
2.5 Notes on Simulation Speedup . . . 93

Chapter 3 System Tuning and Rapid Switching Between Setups . . . 94
3.1 Neural Network Controller for Fast Energy Switching in a Compact FEL . . . 94
3.1.1 The TEU-FEL System Description and First Principles Simulation . . . 96
3.1.2 Study Setup . . . 97
3.1.3 Neural Network Model . . . 102
3.1.4 Neural Network Controller . . . 104
3.1.5 Conclusions and Future Work . . . 105

Chapter 4 Systems with Long-term Time Dependencies: Resonant Frequency Control in RF Cavities . . . 108
4.1 Background on Resonant Frequency Control . . . 109
4.2 Background on Model Predictive Control . . . 110
4.3 FAST RF Gun . . . 116
4.3.1 Background on the RF Gun and Cooling System at FAST . . . 116
4.3.2 Assessment of Existing Feed-forward/PI Loop and Motivation for Improved Control . . . 120
4.3.3 System Characterization . . . 123
4.3.4 Neural Network Modeling . . . 132
4.3.5 Model Predictive Control over the RF Gun and Cooling System . . . 135
4.3.6 Conclusion and Future Outlook . . . 142
4.4 Resonant Frequency Modeling and Control for a High-Power RFQ . . . 143
4.4.1 Description of the RFQ and Implications for Control . . . 148
4.4.2 RFQ Cooling System . . . 151
4.4.3 Analytic Modeling . . . 157
4.4.4 RFQ Water System Control Assessment with Analytic Model . . . 183
4.4.5 RFQ and Water System Characterization Data . . . 187
4.4.6 Neural Network Modeling . . . 192
4.4.7 Resonant Frequency Control System . . . 199
4.4.8 Control Studies and Commissioning of Control Framework . . . 206
4.4.9 Conclusion and Future Outlook for PIP-II RFQ . . . 217
4.5 Conclusion and Future Outlook for Neural Network-based Modeling and Control of Resonant Frequency Responses of RF Cavities . . . 219

Chapter 5 Conclusions . . . 221

Bibliography . . . 226

LIST OF TABLES

2.1 Ranges of data for FAST VCC image study. 53
2.2 Model performance at photoinjector exit. 56
2.3 Model performance at CC2 exit. 56
2.4 Simulated data ranges for FAST virtual diagnostic. 70
2.5 FAST virtual diagnostic data ranges for held-out ranges of settings within test set. . . . 73
2.6 Virtual diagnostic data ranges for test sets. 90
2.7 Simulated data ranges for FAST virtual diagnostic image predictions. 91

3.1 TEU-FEL Data Ranges. 99
3.2 TEU-FEL model performance. 103
3.3 TEU-FEL controller performance. 105

4.1 Transport and thermal delays for the FAST RF gun and cooling system. 119
4.2 Average performance for FAST RF gun model designs. 135
4.3 Benchmark MPC parameters. 138
4.4 *A priori* model parameters for the PIP-II RFQ. 164
4.5 RMS error, max error, temperature range during the test, and the ratio of the RMS error to the temperature range during the first case study. 173
4.6 RMS error, max error, temperature range during the test, and the ratio of the RMS error to the temperature range during the second case study. 174
4.7 RMS error, max error, temperature range during the test, and the ratio of the RMS error to the temperature range during the third case study. 174
4.8 Standard deviation of the error, for each temperature sensor across all three tests. . . . 174
4.9 Ratio of the standard deviation of the error to the temperature range, for each temperature sensor across all three tests. 174
4.10 RMS error, max error, frequency range during the test, and the ratio of the RMS error to the frequency range. 174
4.11 RMS error, max error, frequency range during the test, and the ratio of the RMS error to the frequency range for CW testing. 180
4.12 Variable ranges for measured RFQ data. 193

LIST OF FIGURES

1.1 Examples of various particle accelerator systems. 2
1.2 Tasks conducted by human operators. 10
1.3 Intuition and human inspiration for the basic learning paradigms. 13
1.4 Example of supervised learning. 13
1.5 AI/ML and optimization methods commonly used in particle accelerators. 15
1.6 Basic neural network structure. 17
1.7 Basic convolutional neural network structure. 21
1.8 Convolution operation. 22
1.9 Recurrent neural networks. 23
1.10 Common activation functions. 25
1.11 Examples of overfitting and underfitting. 30
1.12 Example of learning curves and overfitting. 31
1.13 Intuition for L1 and L2 regularization. 33
1.14 Regularization with dropout. 34
1.15 Broad categorization of terms relevant to AI/ML. 38
1.16 Historical improvement in convolutional neural network performance. 41

2.1 Layout of the Fermilab Accelerator Science and Technology Facility (FAST). 49
2.2 Example of changes in laser spot, as seen on the VCC, for nominally the same setup. . 50
2.3 FAST low-energy beamline for laser image studies. 51
2.4 Examples of the transverse emittances as a function of the normalized solenoid strength for three different photoinjector phases. 52
2.5 Transverse initial beam distributions used in the study. 53
2.6 Neural network structure for laser image studies. 54
2.7 Examples of predictive performance for two beam parameters. 56
2.8 Machine learning-based virtual diagnostic concept. 60
2.9 Multi-slit emittance measurement scheme at FAST. 62
2.10 Virtual diagnostic for multi-slit emittance measurement. 63
2.11 Example multi-slit measurement images from FAST for different combinations of gun solenoid strength and phase settings. 64
2.12 Image of the transverse laser profile measured with the virtual cathode camera at FAST. 65
2.13 Example of measured emittance scan data from FAST for various gun phase and solenoid current settings. 65
2.14 Comparison between measured data and OPAL simulations. 66
2.15 Comparison of measurements and simulation for FAST multi-slit emittance diagnostic. 67
2.16 Plots of the main input variable scans for the FAST OPAL simulations. 68
2.17 Plot of select output beam parameters for the FAST OPAL simulations to show rough ranges. Thirteen other output parameters are not shown for the sake of brevity. 69
2.18 Inputs for simulation scan data 72
2.19 Scalar predictions on the test set from the neural network trained on simulation data for a subset of the output parameters as a function of gun phase. 74

2.20 Scalar predictions on the test set from neural network trained on simulation data for a subset of the output parameters as a function of solenoid current. 75
2.21 Closer example of prediction performance of neural network scalar model in interpolation for one of the ranges not seen during training. 76
2.22 Example of updating the pre-trained neural network model with measured data. 79
2.23 OPAL simulation scans conducted to include the multi-slit measurement directly (e.g. image predictions). 81
2.24 OPAL simulation scans showing regions of input scans left out of training and used in the test set to examine interpolation of the neural network. 82
2.25 Comparison between the simulated multi-slit diagnostic images and neural network predictions for some representative cases in the test set. 84
2.26 Comparison between the simulated multi-slit measurement images and profiles and neural network predictions for the same test set cases as shown in Figure 2.25. 85
2.27 Comparisons between ground truth and neural network predictions on the test set for the multi-slit diagnostic images (continued on next page). 86
2.28 Comparisons between ground truth and neural network predictions on the test set for the multi-slit diagnostic images, specifically for the held out scan ranges. 88

3.1 General scheme for the neural network control policy. 95
3.2 Layout of the TEU-FEL accelerator system. 96
3.3 Schematic of generating training data and conducting initial pre-training of the control policy. 100
3.4 Schematic for neural network control policy updates. 101
3.5 Training data example for initial training set generated according to Figure 3.3, showing variation in the five quadrupole settings. 101
3.6 Neural network model predictions and simulated values for TEU-FEL. 103
3.7 Neural network model predictions and simulated values for wider variation in TEU-FEL input settings. 106

4.1 Basic concept of model predictive control. 111
4.2 Basic elements of Model Predictive Control (MPC). 112
4.3 FAST RF gun and nominal operating parameters (as of 2014). 116
4.4 Layout of the FAST gun water system and relevant instrumentation. 118
4.5 Oscillatory open-loop (uncontrolled) response in the water temperature. 120
4.6 1-°C step change under the existing PI controller. 121
4.7 Cooling system data (set 1). 125
4.8 Cooling system data (set 2). 126
4.9 Cooling system data (set 3). 127
4.10 Cooling system data (set 5). 128
4.11 Variation in ambient air temperature over several days. 129
4.12 Temperature differences between system components. 130
4.13 Difference between TIN and TCAV, as well as TOUT and TIN, as a function of RF power. 131
4.14 Illustration of neural network model inputs and outputs for the FAST RF gun cooling and rf system. 134

4.15 Structure of MPC for temperature control of the RF gun. 137
4.16 MPC performance on the FAST RF gun cooling system. 139
4.17 Measured and requested flow control valve actions. 140
4.18 Solid model drawing of the RFQ (courtesy LBNL) 144
4.19 The RFQ during installation. 144
4.20 Profile cutaway of RFQ and drawing showing the locations of the vane and wall cooling channels. 145
4.21 The nominal operating parameters (as of 2015) of the RFQ. 145
4.22 Simulated response in ANSYS of the RFQ to a 2 percent step increase in cavity field, provided by LBNL (courtesy Andrew Lambert, LBNL). 149
4.23 RFQ heating from the ANSYS model. 150
4.24 Simplified PIP-II RFQ cooling system diagram. 152
4.25 Flow valves for the wall and vane cooling circuits (yellow). 153
4.26 Pumps on vane and wall circuits for circulating water through the RFQ. 153
4.27 Distribution manifold for sending chilled water through the RFQ at multiple entry points (shown prior to final hosing installation.) 154
4.28 Block diagram of the water system model. 163
4.29 Absolute sensitivity of the model to 20% perturbation in the parameter during pulsed operation 166
4.30 Absolute sensitivity of the model to 20% perturbation in the parameter during CW operation 167
4.31 Impact on temperature prediction when different components of the model are removed. 167
4.32 Inputs to the model for the third case study in pulsed mode; both the RF amplitude and the flow valves were scanned. 170
4.33 Comparison of measured and simulated frequency shift versus time for a case where both the RF amplitude and the flow valves were scanned. 171
4.34 Comparison between measured and predicted water temperatures at low power; both the RF amplitude and the flow valves were scanned. 172
4.35 Changes in input variables during CW testing. 177
4.36 Comparison of measured and predicted temperature readings during high-power CW operation. 178
4.37 Comparison of measurements to simulations for the frequency shift in the RFQ versus time during high-power CW testing. 179
4.38 Comparison of resonant frequency measurements to analytic model predictions for a temperature transient. 182
4.39 Simulated RFQ response to a 10s RF trip. The uncontrolled response of just the RFQ, the RFQ in the water system model, and the RFQ with PI control over the vane valves is shown. 184
4.40 Simulated RFQ step response. The uncontrolled response of just the RFQ, the RFQ in the water system model, and the RFQ with PI control over the vane valves is shown. . . 185
4.41 Characterization data for RFQ. 186
4.42 Comparison of PI control and MPC control over the temperature of the water entering the vanes, computed with the analytic model. 186
4.43 RFQ resonant frequency response to changes in water temperature in the vane. 188

4.44 Measured, uncontrolled change in resonant frequency after a roughly 5-°C reduction in the cold supply temperature from the cooling skid. 189
4.45 Response times for changes in skid temperature. 189
4.46 Response times for changes in vane temperature. 190
4.47 Flow rate and pressure relationships obtained from the cooling system during initial characterization studies. 191
4.48 An example scan from the measured data. 196
4.49 Measured and predicted resonant frequency shift values versus time. 197
4.50 Measured and predicted resonant frequency values versus time for a wider data set. . . 197
4.51 Resonant frequency measurements and neural network predictions. 198
4.52 Overview of the program flow for the developed resonant frequency control system. . 201
4.53 Simplified, conceptual diagram of the resonance control system architecture and interfaces built in this work to support RFQ operation. 204
4.54 Modular control framework structure for the RFQ resonant frequency control. 205
4.55 Example of uncontrolled detuning in CW mode under a small change in cavity field (55 kV to 58 kV). 207
4.56 PI resonant frequency control under CW operation under a small change in cavity field. 208
4.57 PI frequency control during pulsed RF operation for a 2 ms increase in pulse duration and a cavity field of 65 kV. 209
4.58 PI frequency control during pulsed RF operation for a decrease in pulse duration. . . . 210
4.59 Vane supply temperature variation. 210
4.60 Frequency recovery from a 10-second RF trip at 60 kV using PI control over the vane flow valve. 211
4.61 Frequency recovery from a compound RF trip. 212
4.62 PI loop under CW operation with changing RF cavity fields. 212
4.63 RF for recovery from a 10-second trip in pulsed mode under PI resonant frequency control. 214
4.64 RF startup under PI resonant frequency control on the vanes. Note that a manual correction was made to an incorrect skid water temperature set point around the 14.5 minute mark. 214
4.65 PI loop performance compared to specifications for long trips. 215
4.66 PI loop performance compared to specifications for medium-length trips. 215

Chapter 1

Introduction

Charged particle accelerators support a wide variety of scientific, industrial, defense, environmental, and medical applications (e.g. see [1]). They range in scale and complexity from systems with just a few components for beam acceleration and manipulation, to large scientific user facilities that span many kilometers and have hundreds-to-thousands of individually-controllable components (e.g. see Figure 1.1). Specific operational requirements must be met by adjusting the many controllable variables of the accelerator. Meeting these requirements can be challenging, both in terms of the ability to achieve specific beam quality metrics in a reliable fashion and in terms of the time needed to set up and maintain the optimal operating conditions.

The difficulty in setting up and controlling accelerators arises in part because they have large parameter spaces, their behavior is often non-stationary and nonlinear, not every important variable is measurable, and anomalies or failures in system can cause disruptions (for example, due to equipment failures, or unexpected transients in incoming signals like power sources). Even once suitable operating conditions are met, the system responses often drift over time, which necessitates active feedback control and tuning. Adding to this inherent complexity, accelerator operation often involves interaction of a variety of sub-systems (e.g. steering control, rf cavity control, cooling systems). They are also often subject to tight tolerances on beam parameters and other performance metrics, and it is often desirable for them to run for extended periods of time with minimal interruption. At present, particle accelerators often rely heavily on expert human operators for tuning of machine settings to achieve the desired beam characteristics.

Models of accelerator systems can aid operation. However, there will also inevitably be deviations between physics-based simulations and the empirical system behavior. This is both due to fundamental physics effects that are difficult to model accurately (for example, coherent synchrotron radiation) and due to the many compounding errors in the description of beamline elements (e.g. position offsets, asymmetries in electromagnetic fields, etc.). Finally, these challenges

become more acute as increasingly high-intensity, high-energy, high-gradient accelerators that fundamentally rely on increasingly complex phenomena are built (e.g. plasma-based accelerating stages, superconducting radio frequency accelerating cavities).

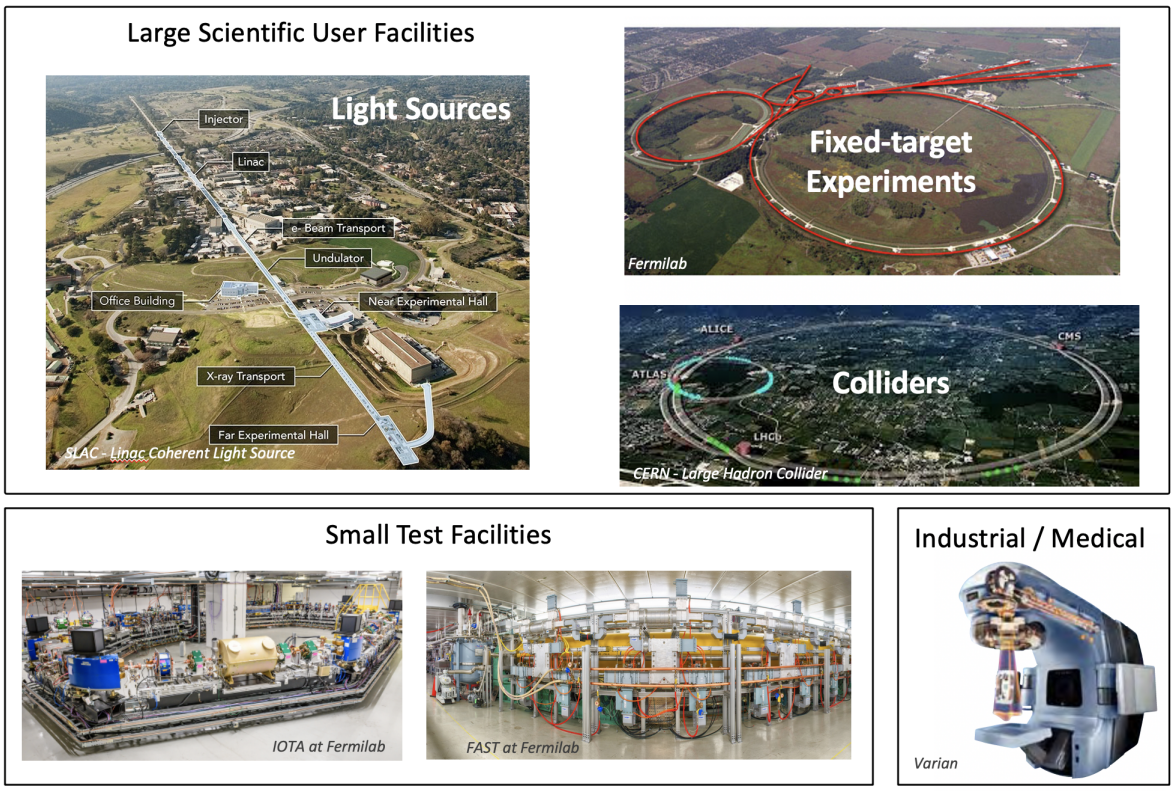


Figure 1.1: Examples of various particle accelerator systems, ranging from large scientific user facilities, to small R&D test facilities, to accelerators for medical applications (e.g. radiation therapy). Images courtesy SLAC National Accelerator Laboratory [2], Fermi National Accelerator Laboratory [3], CERN [4], and the Department of Energy Report "Accelerators for America's Future" [1].

Some types of accelerators operate in one primary configuration, and thus the main goal is to optimize one setup and maintain it over extended periods of time. For example, the main accelerator complex at Fermilab aims to reliably provide the maximum achievable number of protons-per-hour to their scientific users (e.g. see [5]). However, many particle accelerators are request-driven in that they need to quickly supply custom beam parameters for different use cases. For example, free electron laser (FEL) facilities like the Linac Coherent Light Source (LCLS) [6] must accommodate specific requests for beam properties that need to be met in a limited window of time to

ensure the success of scientific user experiments. Fulfilling these requests can be challenging both in terms of tuning time and the final achievable beam quality, especially for novel or non-standard setups. In both of these operational paradigms, time lost during tuning and unplanned interruptions tends to be expensive, both with regard to operational costs incurred by running the machine and with regard to scientific output. At many scientific user facilities, the machine is in high demand by facility users, and each experiment is generally allocated only a limited amount of machine time. Thus, maximizing the scientific output per dollar spent in operation is a major priority, and reducing tuning time can play a large role in this.

Aside from tuning efficiency, fundamental improvements to achievable beam quality can help open up new application areas. For example, the ability to stably produce short electron bunches [7] has enabled fundamentally new photon science experiments to be conducted. In many cases, achieving high beam quality for specific applications increasingly relies on advanced beam manipulations. For example, phase space manipulation techniques like emittance exchange [8] will likely be critical for creating the high emittance ratios needed for future particle collider experiments. At present, accelerators are often optimized in a piecemeal fashion (i.e. adjusting a few settings at a time at different points along the machine or in different machine subsystems). In light of this, more comprehensive use of the wide variety of available controllable settings in the accelerator could lead to fundamental improvements in achievable beam quality. Subtle statistical effects and nonlinearities across the accelerator could then be more directly leveraged.

Improving the quality and speed of accelerator optimization and control is also important for advancing medical, industrial, and defense applications of particle accelerators. For example, proton therapy has shown promise for treating cancerous tumors because it can deposit energy more precisely in tissue than can x-rays, thus enabling one to more closely target the tumor and spare more of the non-cancerous tissue [9]. Fine control over the proton beam energy (corresponding to penetration depth into tissue) and steering (to add greater flexibility in moving the beam around the tumor) could further enhance this. In addition, outside of large accelerator facilities, day-to-day reliance on highly-skilled human operators and technicians is often undesirable and may even

be an impediment to technological transfer (e.g. FELs for defense). Overall, these applications range from relatively well-established use cases where increased automation and better performance could be of significant benefit, to as-yet unrealized applications that require substantial improvements in control of particle beams.

One avenue toward addressing some of these challenges is to incorporate techniques from the fields of machine learning (ML) and artificial intelligence (AI) into the way particle accelerators are modeled and controlled. ML models and controllers can be used in conjunction with actual machine data, thereby accounting for noise, variable delays, subtle statistical correlations, and complex effects that may not be addressed *a priori*. In addition, particle accelerators serve as compelling use-cases for AI/ML with their own technical challenges that could help drive algorithmic development. For example, dealing with data distribution shifts in a physical system is an area of active research in ML, and accelerators are a relatively contained real-world system in which to study this. The existence of reasonable simulation environments, automatic archiving of historical data from many machine signals, and the large nonlinear parameter spaces make them appealing real-world test beds for AI/ML algorithms. For example, in deep reinforcement learning in continuous state and action spaces, having a relatively contained nonlinear environment that can be simulated and, in the real world, controlled is appealing for prototyping algorithms in a challenging real-world setting. Especially for accelerator systems where the risk of damage to machine components due to operational errors is low (e.g. low-power beams), the overall safety risk is lower than many other AI/ML applications (e.g. healthcare, automated driving, etc.). At the same time, operational safety is a concern in some accelerator systems (e.g. high power beams can easily damage accelerator components, and control of beam halo becomes a challenging problem). Finally, despite their limitations, rich physics models for accelerator systems do exist, which makes them a compelling application through which to explore physics-informed machine learning.

While many promising approaches within AI/ML could be used for particle accelerators, this dissertation focuses on approaches based on neural networks. Neural networks are particularly well-suited to modeling, control, and diagnostic analysis of nonlinear systems, as well as sys-

tems with large parameter spaces. They are also very appealing for their ability to process high-dimensional data types, such as images and time series (both of which are ubiquitous in particle accelerators). Neural networks can also be useful in cases where accurate data from simulations or some other computationally intensive procedure is available, but the input-output relationship needs to be computed more rapidly for real-time deployment. There are many such cases in accelerators, especially with regard to speeding up physics simulations and processing of complicated beam diagnostics output. Processing some kinds of beam diagnostics at present relies on simplified assumptions from beam dynamics, often throwing away substantial amounts of information in the image. Because of their functional flexibility, neural networks are also able to operate effectively for many different kinds of tasks.

In this dissertation, we provide background on relevant AI/ML concepts, describe some of the challenges of modeling and control for particle accelerator control, highlight the historical context and motivation for investigating neural network-based approaches to meet these challenges, discuss avenues for incorporating neural networks into particle accelerator modeling and control, and report on several original studies in this area. These innovative studies have played a significant role in influencing the particle accelerator community's adoption of AI/ML.

1.1 Notes on Terminology

Here, the term "optimization" will mean an iterative search process through which better combinations of accelerator settings are found such that specific performance goals are better met (e.g. this could be done manually by a human or automatically using a mathematical optimization routine). The term "control" is most often used here to denote a dedicated process through which a set point, series of set points, or other performance goals for one or more parameters are achieved and maintained despite the presence of disturbances (e.g. having a feedback controller constantly running to keep a particular beam parameter, such as beam energy, at a certain target value). The term "tuning" will be used as shorthand for one kind of task that an operator does: adjusting settings when needed such that desired performance is achieved, both for initial set up of a particular

operating condition and maintaining it over an operating session. "The machine" will be used as shorthand for a given particle accelerator system. The term "online" will be used to mean that a given computational procedure is running and interacting with the machine concurrent to operation. "Offline" will mean any process that does not run in this manner (e.g. using a simulation to find optimal parameter settings prior to running the accelerator, analysis of data gathered from a diagnostic after an operating run has been finished).

1.2 Challenges for Modeling and Control in Particle Accelerators

Some of the motivation and challenges for modeling and control of particle accelerators were described earlier in the introduction. Here we will delineate these into a few related categories. The first general challenge concerns **modeling and characterizing** accelerator systems. Often, physics models for particle accelerators exists, but these are not put to full use in operation because of their computational complexity; the execution speed is often simply too slow to be useful in accelerator operations. In addition, discrepancies between the physics model and the observed machine behavior can be significant. The sources of these discrepancies are often difficult to identify due to the many components and often limited number of beam diagnostics present in accelerator systems. It often requires much effort to calibrate the physics model to the machine, and even once a good solution is found, in practice the agreement can drift over time. During down-times, extensive maintenance is often done that replaces components, meaning the machine itself is typically not a static entity (and the degree to which this is the case varies from machine-to-machine). Simplified, fast-executing physics models are used for some online prediction and control tasks (e.g. see [10, 11]), as well as experiment planning, but these typically do not take into account all important effects and thus provide only a rough prediction of the machine response.

The second challenge concerns the process of actually **optimizing, or tuning, the accelerator** settings for a given setup. This is challenging in large part because it involves searching a large, nonlinear parameter space. In addition, even once good solutions are found for a particular setup,

the machine response can sometimes noticeably drift in short periods of time or change because of hidden variables that are not accounted for. At facilities where different beam requests must be accommodated to support user experiments, the task of tuning also becomes time-sensitive. Accelerator models can help in tuning, both by enabling offline setup planning (e.g. optimizing settings with the model ahead of time), and doing online model-based corrections. However, as described earlier, inaccuracies in the model and execution speed are a significant limitation; effects that are not taken into account by the correction from a model often need to be compensated for by hand or by an online optimization algorithm.

A related challenge is control of accelerator systems where the system dynamics or order of setting changes needs to be taken into account, necessitating planning of action sequences rather than simply finding a static set of optimal settings. For example, magnets and motors can show hysteresis, and control over RF cavity and cooling systems requires careful balancing of heating and cooling effects on different timescales to keep the beam accelerated at the proper energy. Systems that involve significant time delays relative to the timescale on which adjustments must occur can be challenging to control.

In addition to online optimization of existing machines, the **design of new particle accelerators** is yet another complex optimization problem. Often, settings as well as layouts of components are adjusted to achieve specific ranges of beam operation. This is typically cast as a multi-objective optimization problem, where trade-offs between competing beam parameters are examined (e.g. minimizing beam size while maximizing beam energy).

Another set of challenges concerns **beam diagnostics** in accelerators. First, in many cases, only limited beam diagnostics are available. Second, not all beam diagnostics are continuously available. For example, destructive diagnostics (such as those relying on screens that intercept the beam) cannot be used when the beam must be transported further along in the accelerator (e.g. for a downstream user experiment). It also means that while tuning, one must switch between destructive diagnostics, leading to incomplete information at any given time. Analysis of beam diagnostic output also sometimes relies on computations that are slow, or they may rely on assumptions about

the beam dynamics that do not perfectly hold (e.g. calculations requiring beams that are Gaussian in phase space, when in fact this is difficult to achieve). The full information in the image is often not used.

While a comprehensive survey of accelerator physics is outside the scope of this dissertation, pedagogical treatment of accelerator physics can be found in [12–17]. In addition, [18] discusses many of the practical considerations surrounding photoinjector systems, which are the subject of several of the studies examined in this dissertation. Also of relevance to some of the work presented in this dissertation are FELs; discussion on the physics of FELs can be found in [19–21]. Examining these resources may give readers greater appreciation for some of the physics involved in the challenges mentioned above.

Inspiration from Human Operators in Online Modeling and Control

Some inspiration can be gained by examining how accelerator operators interact with these machines. Operators will often conduct extensive tuning each time a machine is put into a new operating condition or starts up after a shut-down. This can work well for dynamics on human-compatible timescales (i.e. ones that are not too long or too, e.g. short—hundreds of milliseconds to tens of minutes), or sub-systems that involve just a few parameters and/or are not substantially nonlinear in their responses. The task can become unwieldy as the dynamics become more nonlinear or the number of parameters and interrelations increases. The presence of many variables and multiple timescales of behavior can also make isolation of relevant parameters extremely difficult. Furthermore, machines that require frequent changes in beam parameters or operating conditions vastly increase the number of learned control strategies and specific procedures that operators must employ. Ultimately, human operation is limited in the following ways: 1) humans can only process a handful of input parameters at once; 2) humans can only act on a few parameters at once; 3) humans can only operate on a relatively narrow set of timescales, and separating multiple timescales can become rapidly infeasible; 4) humans skill levels vary dramatically.

However, humans are remarkably good at collecting disparate kinds of information and putting it to judicious collective use. Experienced accelerator operators can become adept at tuning ma-

chines. Related to this, loss of institutional knowledge and skill when an expert operator leaves can have a profound impact on a facility's ability to run effectively. Significant advances can be made by critically examining the real-world human problem-solving process and breaking it into constituent parts for a given task. Take, for example, an advance in neural network-based recognition of written characters that significantly outperformed competing methods even with far fewer training examples: they key insight was to incorporate a process through which the neural network learns to mimic the way a human learns to produce letters in the first place, i.e. "learning to learn," rather than simply training extensively on a large data set of examples [22]. Thus, a human-inspired process that at first might seem tangential to a typical character recognition task was in fact of substantial importance for achieving improved performance.

For accelerator systems, operators have an understanding of the dynamics of the machine (i.e. a system model) through both a theoretical understanding of the ideal behavior of the machine and the observed behavior of the machine. This model is adapted through experience and can be compared with an understanding of similar machines. The operator also has a record in memory of previous states visited, actions taken, and resultant outcomes. An operator with many years of experience or deep knowledge of a given system does not typically need to solve partial differential equations or run a physics-based simulation to have a good idea of what will happen when they take certain actions when in certain machine states, even when the outcome of a specific combination of states and actions has not previously been observed. In other words, they have learned a fast, heuristic representation of the relevant processes that allows for generalization beyond direct experience and memory. Operators also process a tremendous amount of sophisticated information on-the-fly. The operator is in essence doing fast data reduction, image processing (e.g. visual input from a beam diagnostic), and pattern recognition (e.g. recognizing when an instability is starting to develop). Finally, operators also will often do extensive local tweaking of settings to iterate toward better performance, similar to how an online stochastic optimization procedure applied directly to controllable parameters would.

Thus, operators are making predictions based on mental models of the system, checking observed behavior against the models to improve them over time, planning series of future control actions, interpreting a substantial amount of diagnostic information, and using present and past performance to adjust the rules by which control decisions are made for various sets of observed or inferred machine states. For each of the above capabilities of an experienced operator, there are analogous techniques in optimization and AI/ML, and these could potentially be used to augment human operation.

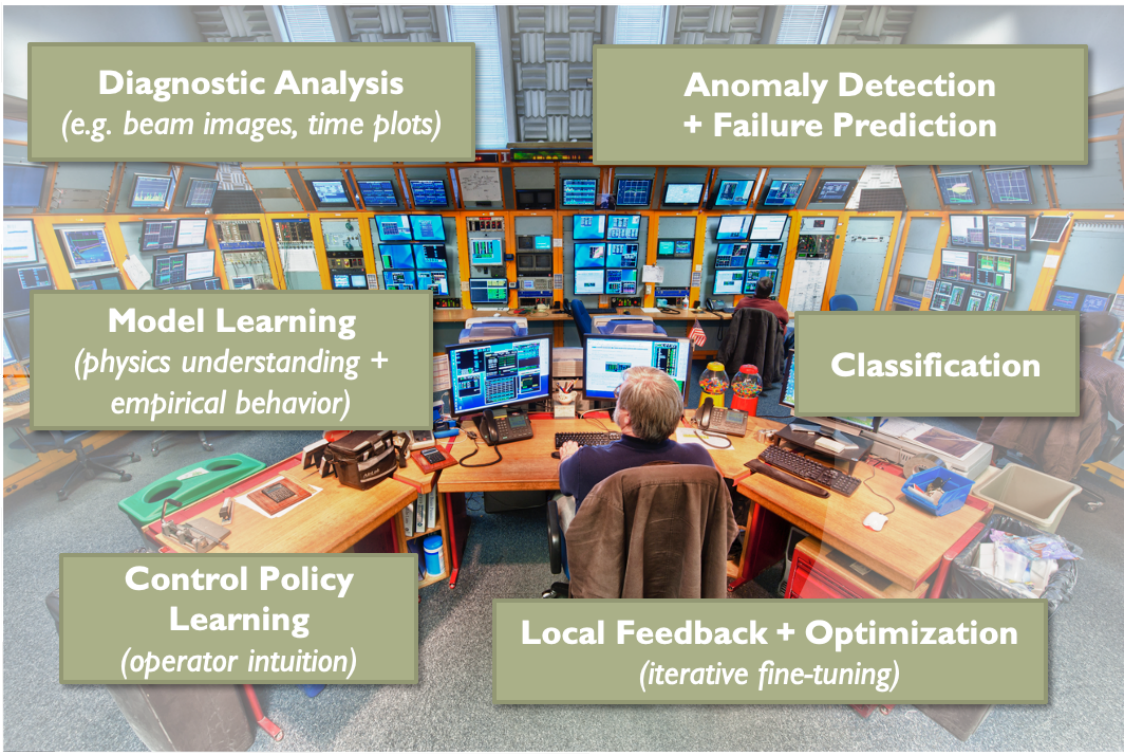


Figure 1.2: Various tasks implicitly conducted by expert accelerator operators. Background photo of Fermilab control room courtesy Reidar Hahn.

1.3 Neural Networks and Machine Learning

1.3.1 Basic Background

Here we cover basic background on neural networks and machine learning, focusing on aspects that are needed to understand the content of this dissertation. Broadly speaking, **machine learning** (ML) is concerned with improving an algorithm's performance some task over time by learning through interaction with data. For example, tasks could include making predictions about the behavior of a physical system, identifying particular handwritten characters, and detecting aberrant credit card activity. ML is a sub-field of **artificial intelligence** (AI), which is concerned more generally with creating systems that are capable of behaving "intelligently." The broadness and vagueness of this term contributes to much confusion. What is usually meant by "AI" is that some aspects of human intelligence are exhibited by the AI system or algorithm. For example, the ability to plan future actions based on predictions, the ability to interpret complicated environmental input (e.g. interpreting image data, viewing and mentally mapping the surrounding physical world), the ability to perform self-assessments of performance, the ability to adapt one's behavior in response to interactions with an environment, and the ability to engage in rational decision making (or some approximation of it) are all characteristics of intelligent behavior. Most modern ML is focused on "narrow" AI, which aims at creating systems that are capable of learning to complete just one or a few tasks with suitable proficiency. In contrast, the study of "general" AI is focused on creating general autonomous systems (or AI "agents") that are capable of learning a variety of tasks and expanding to new tasks. General AI is thus more closely aimed at trying to mimic or achieve similar performance to human or animal intelligence. There historically has been heavy interplay between neurobiology, sociology, psychology, and AI/ML, both with regard to using artificial systems to help study biological/natural systems and with regard to using biological/natural systems to inspire new AI/ML approaches. ML also relies heavily on techniques from computational statistics and stochastic optimization; this interplay occurs both in ML model development and in the use of ML to solve optimization problems. For more discussion, and a general overview of AI, see [23]. Within AI/ML, neural networks (NNs) are just one particular kind of learning structure or tool. For

a broad overview of neural networks and related topics, see [24–26]. The general terminology can be approximately summarized as follows:

1. **Artificial Intelligence** (AI) is concerned with enabling machines to exhibit aspects of human intelligence: knowledge, learning, planning, reasoning, self-assessment, and perception.

2. **Machine Learning** (ML) is a field within AI that is concerned with enabling machines to complete tasks without being explicitly programmed. Some typical tasks in machine learning include classification (categorizing instances of data), clustering (collecting similar kinds of data together), dimensionality reduction (reducing the number of random variables by finding and exploiting relationships between them or mapping them to a new set of variables), and regression (estimating relationships between variables).

3. **Artificial Neural Networks** (NNs) are, in general, parameterized structures that use many connected processing units to form complicated maps between input and output data.

4. **Deep Learning** (DL) in general is concerned with learning hierarchical representations of data. At the moment, the term is largely synonymous with the class of techniques that use many-layered (i.e. "deep") neural networks to learn such representations.

The learning (or "training") process is an essential component of AI/ML. Some typical learning paradigms within ML include **supervised learning**—in which examples demonstrating correct input-output relationships are given (i.e. "labeled data"), **unsupervised learning**—in which no correct example pairs are given and underlying structures in the data must be found on their own (e.g. unlabeled data that must be grouped into similar categories without specifying what those categories are), and **reinforcement learning**—in which an agent interacts with the environment and alters its behavior based on the "reward" (or penalty) it receives from the environment over time.

For example, in a regression problem using supervised learning (see Figure 1.4), a data set consisting of inputs x and outputs y is generated with process $y = g(x)$. The y data is often called

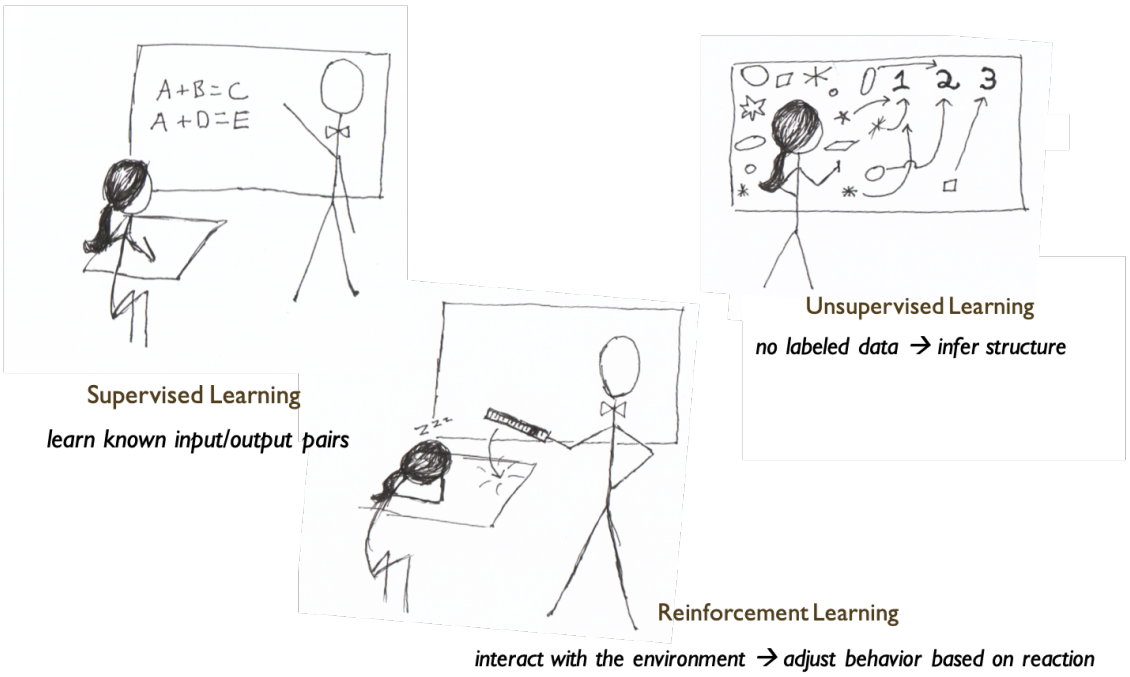


Figure 1.3: Intuition and human inspiration for the basic learning paradigms.

the ground truth, and the goal is to obtain an approximate input-output map $\tilde{y} = \tilde{g}(x)$. In the case of neural networks, the neural network is the map $\tilde{g}$, and training in this case is an optimization problem to find neural network parameters θ that minimize the prediction error, as codified by a chosen cost function. For example, the mean squared error (MSE) is a common choice of cost function, given by $C = \frac{1}{N}\sum_{i=1}^{N}(\tilde{y}_i - y_i)^2$ where N is the number of samples. The learned map should also be able to interpolate or generalize well to unseen data, i.e. given $x' \notin x$, the predicted $\tilde{y}'$ should be sufficiently close to $y' = g(x')$.

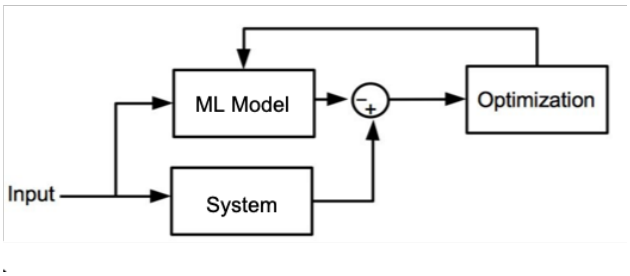


Figure 1.4: Example of supervised learning, where, for example, the parameters of model are optimized to learn an approximate input-output map (in this case of a particular system).

To contrast, a reinforcement learning scheme (see Figure 1.5), typically includes the following components:

- An evolving environment with which the agent interacts.
- A policy which maps observed system states to actions; these are rules by which control actions are chosen, and they are sometimes encoded in a neural network.
- A reward function that delivers a scalar value indicating how "good" the environmental response to the chosen behavior is.
- A means of estimating long-term future rewards for entering a given environmental state or for taking a certain action in a particular state. In this way, the benefit of accessing "better" states later on by entering "worse" states in the interim can be taken into account. This can also be parameterized with a neural network.
- An episode structure that determines how many iterations the agent is allowed to have with the environment before being evaluated and updated.

By observing states, choosing actions, and assessing the efficacy of those actions over time, the agent eventually learns to choose actions such that the highest long-term reward is received. Some architectures for reinforcement learning also use and/or learn a process model of the system to facilitate planning subsequent steps. RL is most often appropriate in cases where the order of actions taken matters (e.g. navigating a maze, controlling settings where hysteresis effects are important, etc.), or where the time-evolution of the system must be taken into account when taking actions (e.g. actions are taken at a rate faster than it takes the system to reach steady state).

Machine learning methods which combine these basic paradigms, as well as more specialized learning paradigms, are becoming increasingly common. For example, in **semi-supervised learning**, a small amount of labeled data is used to help train a model on a large unlabeled data set. In other cases, competition between algorithms is an essential component of training; two neural

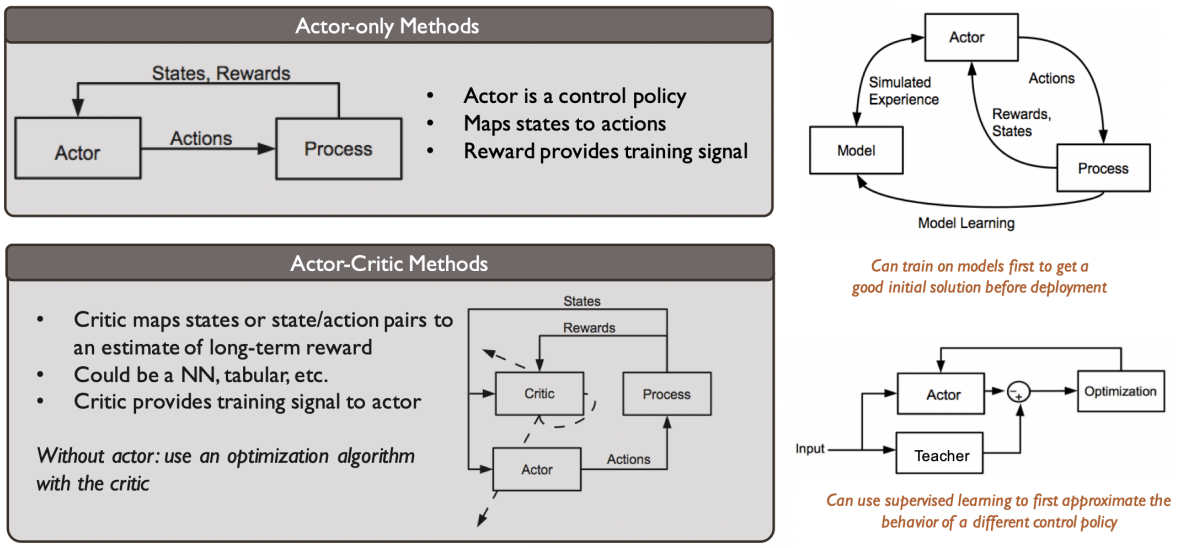


Figure 1.5: In reinforcement learning (RL), an intelligent agent (or "actor") interacts with an environment and improves its performance over time based on environmental feedback (the "reward"). An "actor" or "policy" maps observed system states to actions. The policy could be a direct map with a neural network, or a simple set of rules. A "critic" or "value function" is sometimes used to assess the merit of the actions taken by mapping actions to expected future rewards. This mapping can be learned over time. In the case where both the actor/policy and the critic are parameterized as neural networks, the signal from the critic can be used to update the actor/policy (updates are shown here by dashed lines). System models can be used to accelerate the learning process, as can pre-training with supervised learning to mimic an existing policy. This is just a basic set of examples to describe the general concept of reinforcement learning and highlight the flexibility. For more detail on specific reinforcement learning techniques, see [27].

networks may be jointly trained on competing tasks, and each spurs the other to improve performance. For example, in training of generative adversarial network (GANs) [28], one network (the generator) tries to produce realistic images and another network (the discriminator) tries to distinguish between real images and ones that were produced by the network. The generator is trained based on how well it can fool the discriminator, and as the generator becomes better at producing realistic images, the discriminator must become more sophisticated in how it detects the generated images. A popular analogy likens this to the way a person producing art forgeries and an art dealer might force each other to improve their respective skills over time.

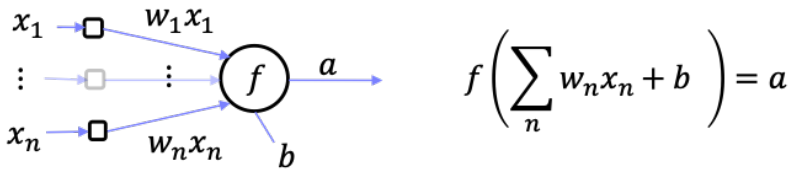
Meanwhile, a variety of approaches are used to tackle different kinds of ML problems. **Transfer learning** [29,30] focuses on enabling networks trained on one narrow set of tasks to generalize to a broader set of tasks, or to a new environment, or to data where the data distribution has shifted

from the training data (e.g. take, for example, training on a simulator of a system and then transferring the ML algorithm to the corresponding real-world system). **Active learning** (e.g. see [31–33]) queries for additional labeled data when there are high levels of uncertainty, so that these may be added to the training data (for example, an algorithm may ask for a new sample from a physical system, or ask a human operator to decide between two proposed settings). **Online learning** focuses on incrementally updating models with data as it become available. **One-shot learning** (e.g. see [34]) is focused on learning concepts that enable an ML model to generalize with only a few examples (for example, a child only needs to see a few examples of mammals and non-mammals before it can identify correctly whether a newly-observed animal is a mammal or not).

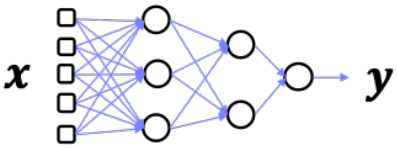
1.3.2 Basic Neural Network Structure and Learning

Artificial neural networks (NNs) are very loosely inspired by the hierarchical processing of information in animal brains [24]. They are universal function approximators [35–38] that are tailored specifically for a given task or computation. As such, they are highly flexible and are in principle able to operate effectively in many different situations, so long as appropriate architectures and training algorithms can be found.

In its simplest form, a neural network consists of a collection of functions ("activation functions") with weighted connections between them. These weighted connections can be adjusted until a desired output behavior is achieved, typically through an automated optimization procedure. This process of adjusting the neural network parameters is called "training". Elements of the neural network structure itself (for example, the number of nodes and layers), can also be adjusted as part of training.



(a) one node (or neuron)



(b) neural network

Figure 1.6: Basic structure of a feed-forward, multi-layer, densely-connected neural network. The input values from previous layers, the weights w, and the biases b are passed through a nonlinear activation function f to produce the output from each node. These are then connected to form a neural network.

What follows is a description of a simple feed-forward multi-layer network, like the one shown in Figure 1.6. "Feed-forward" means that information only flows from the input to the output in one direction. Each node in the neural network takes in a set of inputs x and transforms these according to $t(w, b, x) = \sum_{i=1}^{n} w_i x_i + b$, where n is the number of incoming connections to the node, w_i is the weight on the i^{th} input, and b is a node bias term. This is then connected to a nonlinear activation function $f(t)$ to produce the node output, i.e. output $a = f(t) = f(\sum_{i=1}^{n} w_i x_i + b)$. Many different activation functions can be used, and some common choices are described later. The neural network nodes are generally arranged in "layers" that each contain a certain number of nodes in parallel. A neural network with l hidden layers is called a "$(l+1)$-layer neural network". The neural network is thus defined in this case as a continuous function $g(x)$ constructed by a composition of linear transforms and nonlinear activation functions. For layer l consisting of j nodes and a previous layer $l-1$ consisting of i nodes, the output of the j^{th} node in layer l is determined by $a_j^l = f(\sum_{i=i}^{n} w_{ji}^l a_i^{l-1} + b_j^l)$. A neural network is said to be "densely connected" when all nodes of an earlier layer are connected to all nodes in the subsequent layer. The number of nodes and layers, along with types of interconnections, is collectively called the neural network "architecture."

A neural network is often mathematically expressed in practice by weight matrices, i.e. the j^{th} row and i^{th} column in a matrix w^l for layer l would contain the weight for connecting the i^{th} node in layer $l-1$ to the j^{th} node in layer l. The bias for the j^{th} node then becomes the j^{th} element in the bias vector b_j^l. By vectorizing, we then have the matrix operation $a^l = f(w^l a^{l-1} + b)$, which greatly simplifies and speeds up the computation of the neural network output.

The values of the weights and biases are randomly initialized and adjusted automatically during training to achieve some performance metric; this is an optimization problem where the goal is to find the most optimal weights and biases. Typically, prior to training, the weights and biases are randomly initialized with small values, which sometimes take into account the number of in-going and out-going layer nodes (e.g. see [39] for an example of how this is done, along with further discussion). Considering all undetermined parameters θ (the biases, weights, architecture), the

neural network can be considered as a map $\tilde{y} = \tilde{g}(x; \theta)$, where x is some input, $\tilde{y}$ is some output produced by the neural network, and during training one adjusts θ to achieve the desired map.

This is a very common type of neural network, and its basic components provide a foundation for more complicated neural network architectures. These range from simply adding recurrent connections between later and earlier layers (as opposed to having only feed-forward connections between layers), to more exotic cases that break the basic layer-by-layer structure altogether. For example, echo state networks [40], which have been used to model chaotic systems, have a central random reservoir of randomly-weighted connected nodes and then only train the input and output weights. Some architectures also include additional dynamic elements such as time-dependent activation functions (e.g. spiking neural networks [41]). A survey of neural network architectures can be found in [42, 43].

1.3.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) have become the state of the art in computer vision tasks over the past decade (e.g. see [44] for many examples of successes and some discussion on the history of deep learning). Proposed in 1989 [45], they are inspired by biological neural structures for visual processing, where an individual neuron is activated only by a local region of input (a " local receptive field"). One of their first real-world applications was for handwritten digit recognition [46], and this legacy in handwritten digit recognition is carried on in introductory tutorials today using the handwritten digit dataset MNIST [47]. Despite their early introduction, CNNs did not reach their modern state-of-the-art performance in computer vision until the early-to-mid 2010s. CNNs learn to process hierarchical features in an image. For example, it has been shown that earlier layers learn to recognize basic lines and curves, and later layers learn more complicated compound structures (e.g. eyes and noses in the case of facial recognition). See [48] for some examples.

A simplified example showing the basic CNN concept is shown in Figure 1.7, and the basic convolution operation is shown in Figure 1.8. In an individual CNN layer, learned filters (or "kernels") are convolved across the image (e.g. left-to-right, top-to-bottom), and the dot product (element-wise multiplication and summation) of the filter matrix and image patch at each point produces the corresponding value of the output matrix.

Each layer is specified by some number of unique filters (e.g. 16, 32 are common choices), the stride of the filter across the image (i.e. scanning the image pixel-by-pixel, or skipping a set of pixels each time), and the size of the filters (e.g. 3×3 and 5×5 are common filter sizes). The output of a given intermediate layer then becomes another set of matrices (also called "feature maps"). Padding of the image (i.e. the addition of null pixels around the exterior) is also sometimes used to make the output size of the image after the convolution operation meet some desired value.

The CNN hyperparameters (number of layers, filters, filter sizes, pooling operations, etc.) are generally determined by the expected features in the image (e.g. expected feature size, sparsity, etc.) and empirical tuning. A common approach is to take a publicly-available CNN architecture

and set of weights that has been trained on very large datasets (many of these are readily available now) and adapt it to a specific problem only adjusting the last few layer weights on new data (e.g. see [49–51]). This enables smaller data sets to be used by leveraging the primitive filters that have been previously learned.

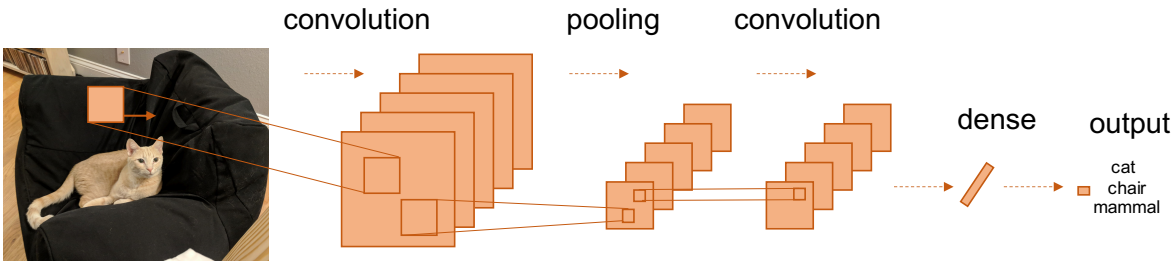


Figure 1.7: Basic concept of a convolutional neural network (CNN). A series of filters are convolved across the input image, producing corresponding output features. The output can then be reduced in size with pooling layers (e.g. taking the maximum of a set of adjacent pixels, for instance, or averaging a set of adjacent pixels). After going through several iterations of convolution and pooling operations, a few densely-connected layers are typically added with decreasing numbers of nodes. Finally an output layer provides the final network output, which could, for example, be a set of numbers coding for specific objects that the neural network determines are present in the image.

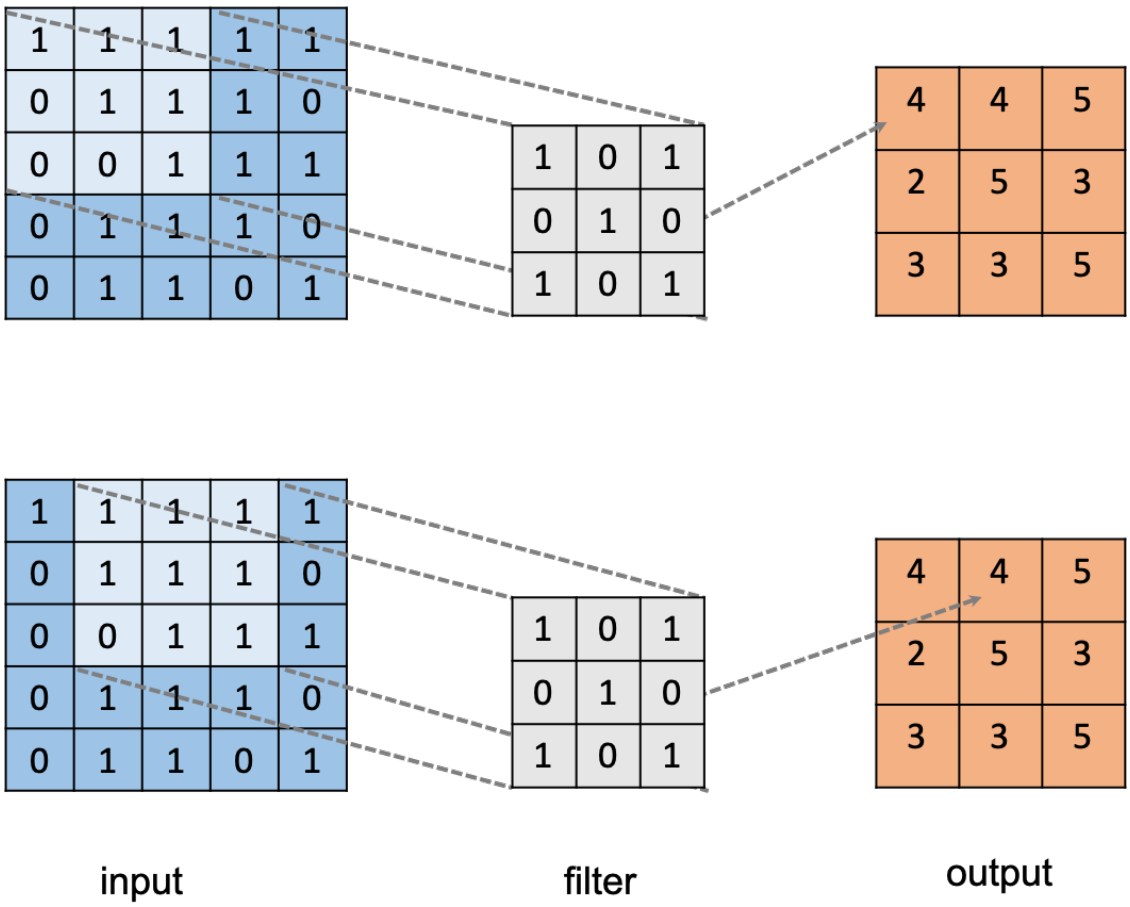
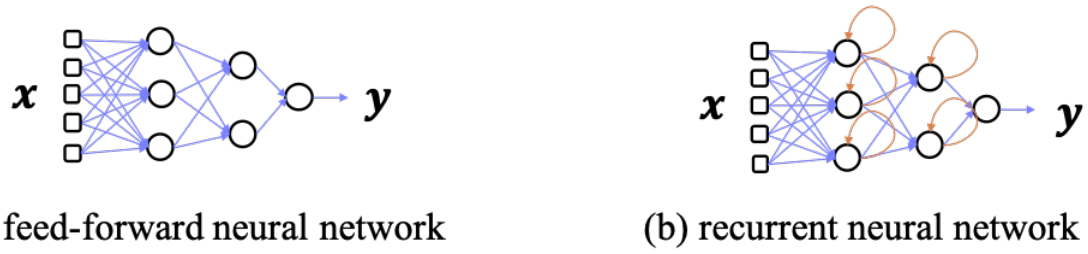


Figure 1.8: Example of the first two steps in a convolution operation on an input image. A filter is convolved across the image pixels, and the dot product of the filter and image patch (element-wise multiplication and summation) produce a processed feature output. This shows a 5×5 input with a 3×3 filter and a stride of 1. The stride indicates that the filter advances by one pixel horizontally across rows, and at the beginning of each subsequent row, it advances downward by one pixel. In a CNN, the filters are learned. Pooling operations are conducted in a similar fashion (e.g. taking the maximum value or average value in a patch).

1.3.4 Recurrent Neural Networks

In contrast to feed-forward neural networks, recurrent neural networks (RNNs) have connections that loop backward in the layer hierarchy (see Figure 1.9). This allows them to learn dynamic behavior, i.e. the impact that a previous input will have on the prediction for the next input, and so on. This makes them appealing for sequence modeling (e.g. data time series, language modeling). However, until recent theoretical and methodological advances were introduced, they were considered relatively difficult to train in a stable fashion due to a number of technical challenges. These challenges are summarized in [52], and some advances are summarized in [53,54]. A major challenge is that of vanishing gradients (where the gradient signal becomes extremely small in earlier layers), as one can consider a basic RNN to be equivalent to a very deep feed-forward neural network with cascaded inputs on each layer.



(a) feed-forward neural network (b) recurrent neural network

Figure 1.9: Feed-forward neural network and recurrent neural network with connections from later to earlier layers.

Specialized RNN architectures, such as long short-term memory networks (LSTMs) [55–57] and gated recurrent units (GRUs) [58], attempt to get around this problem by choosing selectively when to remember and when to forget information from new inputs and previous timesteps. Advances in training procedures and the incorporation of these architectures into common libraries (e.g. `keras`) contributed substantially to increasing their popularity for sequence prediction. More recent work [59–62] has shown that convolutional neural networks (CNNs) can also be used for sequence prediction and in some cases can out-perform RNNs.

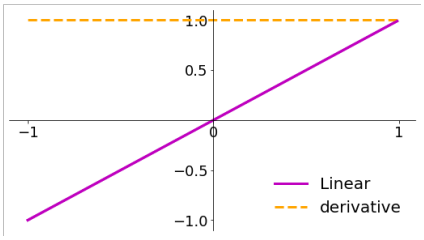
1.3.5 Activation Functions

Some examples of common activation functions are shown in Figure 1.10. Activation functions are typically chosen based on the expected functional form of the problem, the computation speed requirements, and the neural network architecture (e.g. CNN, RNN). For example, the hyperbolic tangent activation function (or "tanh") is a very common activation function that is useful for nonlinear modeling, but it can be prohibitively slow when trying to compute weight updates for large neural networks. Rectified linear unit (ReLU) activation functions [63–65] are significantly lower in computational overhead, and are commonly used in CNNs. Variants that replace the flat output on negative inputs for ReLU with a small linear slope were introduced in 2014 in [66] as the "Leaky ReLU." These are appealing because they avoid the complete loss of the error gradient signal that can happen with ReLU for inputs less than or equal to 0 (this is called the "dying ReLU" problem). For an arbitrary slope α, these become parameterized ReLU (PReLU) activations [67]. Sigmoid and tanh activation functions also suffer from vanishing gradients for inputs that have large positive or negative values. Although tanh is still a "sigmoidal" activation function in that it is just a transformed sigmoid, the basic sigmoid activation function in general has fallen out of favor for use in hidden nodes because it is not 0-centered and thus can shift the output of intermediate layers too drastically for stable training (see [39, 68] for more discussion). For outputs that are expected to lie between a set range of outputs or are binary, having a squashing function like tanh or sigmoid on the output is desirable. For cases where the outputs are expected to be unbounded, a linear activation function on the output layer allows arbitrary output magnitudes to be predicted by the network. The input and output data used with a neural network is also typically scaled according to the active range of the activation functions used (e.g. -1 to 1 for a neural network with tanh activation functions). For more discussion on activation functions, how to choose them, and how their choice affects the training process, see [24, 25, 39, 68].

Linear / Identity

$$f(x) = x$$

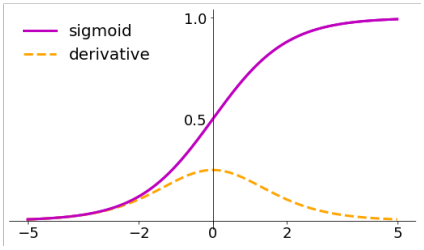
$$f'(x) = 1$$



Logistic / Sigmoid

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

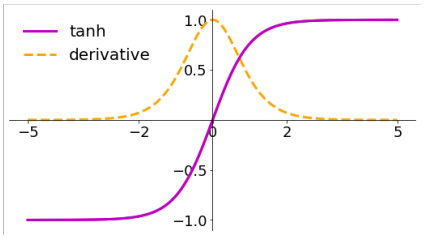
$$f'(x) = f(x)(1 - f(x))$$



Hyperbolic Tangent

$$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

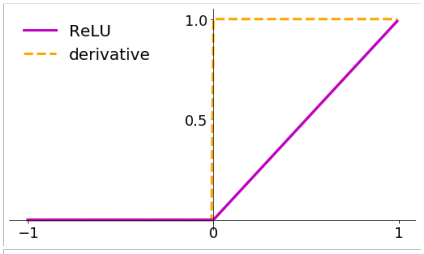
$$f'(x) = 1 - f(x)^2$$



Rectified Linear Unit (ReLU)

$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases} = \max\{0, x\}$$

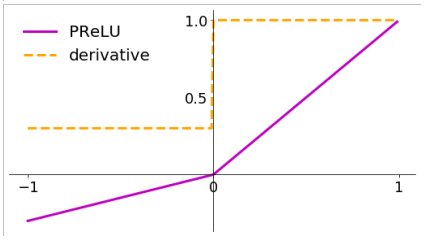
$$f'(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$$



Parametric ReLU (PReLU)

$$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

$$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$



Exponential Linear Unit (ELU)

$$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$$

$$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$$

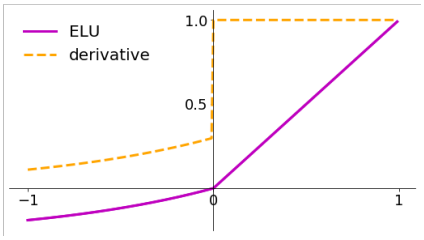


Figure 1.10: Common activation functions. Other activation functions, such as Gaussians, are used but are much less common. For parameterized activation functions (e.g. PReLU), the parameter α can be set arbitrarily to adjust the function.

1.3.6 Common Training Algorithms

Because neural network training is an optimization problem involving the adjustment of the network weights, biases, and architecture (number of layers and nodes, activation functions, etc.), it is itself a heavily-studied area in AI/ML. Here the topic of neural network training and common training algorithms is covered at a high level. More detail can be found in the references herein.

Automatic optimization of neural network weights and biases is susceptible to getting stuck in sub-optimal solutions (or "local minima"). One might then think that methods which are often used for approximate global optimization, such as evolutionary algorithms (EAs), would be the standard choice for finding good sets of neural network weights. However, with very large networks and large data sets, these approaches are at present too computationally inefficient to be practical.

Genetic algorithms [69,70] can be used to adjust both the neural network weights and the architecture, but these were not popular during the time of this work due to their computational expense. Early work using evolutionary approaches, such as Neuro Evolution of Augmenting Topologies (NEAT) [71], showed promise for finding very efficient network architectures (in terms of number of nodes and layers required to achieve a certain level of predictive performance). There has more recently been a resurgence in the popularity of these methods due to increased computational capabilities. Indeed, when trying to use NEAT for some of the problems in this dissertation, it was found that NEAT produced (as expected) substantially more compact neural network architectures than were found by hand; however, the computational expense made it prohibitively slow to use this approach. That said, older work in accelerators used NEAT successfully on smaller neural network architectures and data sets [72].

In contrast, the standard practice at present is to use variants of gradient descent for training, with a few modifications to avoid getting stuck in local minima. The weights and biases are updated by back-propagation of the error gradient through the network layers, in what amounts to repeated application of the chain rule. This is well-described elsewhere (e.g. see [24, 68]). For a general introduction to mathematical optimization, see [73]. For an older text describing many of

the nuances of training neural networks with gradient descent and backpropagation, see [68]. For more discussion of other training considerations, see [25].

The training set is typically broken into individual "mini-batches," or subsets of the data, for the weight updates. A new mini-batch is used at each weight update iteration. After going through one entire pass of the data (called an "epoch"), the batches are often shuffled, and the process starts again. In the context of using gradient descent as the optimization algorithm, this is called "stochastic gradient descent." Training on mini-batches serves several purposes. First, it increases the speed of each gradient update (and thus convergence of the neural network training) because calculations are done on a smaller data set. There is a trade-off between the batch size and the accuracy of the gradient estimate, but in practice for neural networks this is not a limiting factor. Second, training in batches reduces the need to hold a large amount of data in active computer memory (which is important for large data sets, particularly when each sample is large, as may be the case for array and volume data like images and video). Third, it introduces stochasticity into the weight updates. In practice this enables better minima to be found by reducing the likelihood of getting stuck in a poor local minimum and increases the ability of the neural network to generalize to unseen data. The exact impact of using stochastic gradient descent (versus other training methods) and the influence of learning rate and batch size on generalization are still active areas of study in ML research.

In practice, variants of stochastic gradient descent that use values from recently-sampled gradient estimates (e.g. [74, 75]) help stabilize the learning. The simplest version of this, stochastic gradient descent with momentum [76], uses a linear combination of the previous gradient estimate and the present estimate. Some variants also automatically adapt the learning rate; for example, the Adam optimizer [75] does this for each variable. For an explanation of momentum and other variants of gradient descent in the context of neural networks, see [77]. In addition, the conjugate gradient method [78], which uses the conjugate of the gradient for the weight update, is also commonly used.

In contrast to methods which look at the first derivatives of the cost function alone, so-called second-order methods attempt to use an approximation of the local error curvature as well (e.g. by approximating the Hessian). Examples include Newton and Quasi-Newton methods, such as L-BFGS [79]. These were not popular choices for training at the time of this work because although they typically converge in fewer iterations, each iteration is slower-to-compute than it is for first-order methods. For small networks (e.g. a few layers with a few tens of nodes in each layer), these methods can find good solutions in a reasonable amount of time. For example, in some of the work described in this dissertation with comparatively small datasets and nonlinear problems, they worked well and often found a better optimum than stochastic gradient descent and its variants (though this was not rigorously studied).

Architecture search itself is often done by hand with empirical tuning. For example, a popular approach is to do a grid search over neural network architecture parameters (number of nodes, layers, and specific inputs used). There are also rules of thumb for setting initial architecture sizes based on the number of inputs, outputs, and available training data; for more discussion, see [24]. Architecture search is likely to become more automated in the future. Along with evolutionary approaches (e.g. see [71, 80–82]), recent work has investigated a number of techniques for automated architecture search, including Bayesian optimization (e.g. see [83]) and reinforcement learning (e.g. see [84]). Removing unimportant connections ("pruning") as training progresses is also an older technique [85] that has recently been heavily revisited [86]. Further discussion can be found in the aforementioned references.

As with any optimization problem, neural network performance is very sensitive to the weight initialization, particularly when using stochastic gradient descent and its variants (e.g. see [39, 87] for more discussion). To ensure that a particular network design performs well (as opposed to just finding a lucky weight initialization), networks are often trained with multiple random weight initializations (e.g. 5 to 10), and the performance results are then averaged to give a final score to that architecture. This is sometimes coupled with k-fold cross-validation, where k subsets of the data are used for training separately to get an idea of how the network architecture performs. An

added advantage of this is that one can use the entire set of trained models in future predictions (called "model ensembling"), which collectively can provide more reliable predictions than a single trained model and also enables one to gain a rough estimate of the prediction uncertainty.

1.3.7 Model Validation, Overfitting, and Regularization

Model Validation

As highly-parameterized structures, neural networks can easily overfit the data or memorize training data sets if proper care is not taken, resulting in poor performance on new data (see Figure 1.11 for visual examples). When a neural network is overfitted to the data, it may not generalize to new data as well. When the neural network is underfitted, it may not provide satisfactorily accurate predictions. Underfitting can occur when too much regularization is introduced or when the neural network architecture is not expressive enough for the function to be learned (e.g. not enough layers and nodes). Overfitting can occur when the neural network architecture is too expressive, when too little data is used in training, and when insufficient regularization of the weights is used. Overfitting is a serious issue and can result in catastrophic failure on a task when new examples are presented.

To ensure that a valid predictive model has been produced, the neural network is typically trained and assessed with different data sets. The available data are typically divided into **training, validation, and testing sets**. Training sets are used to compute the actual model fit and determine updates to the model. The validation set is not used in any of the calculations for updating the model, but prediction errors on the validation set are used to monitor the training process (e.g. to ensure the neural network still interpolates or generalizes well). The training process can be automatically stopped before the performance on the validation set begins to decrease, which indicates the onset of over-fitting. This commonly-used approach is called "early stopping" (see Figure 1.12. The testing data is never used during training in any way and is used to assess the quality of the final solution.

Much care must be taken during data set splitting to avoid polluting the validation and testing data. For example, if the data is over-sampled, the three data sets can look statistically identical because too many samples are too close together, which defeats the purpose of splitting them in the first place. Similarly, data augmentation, such as adding noise to expand a data set, should only be done after splitting the data.

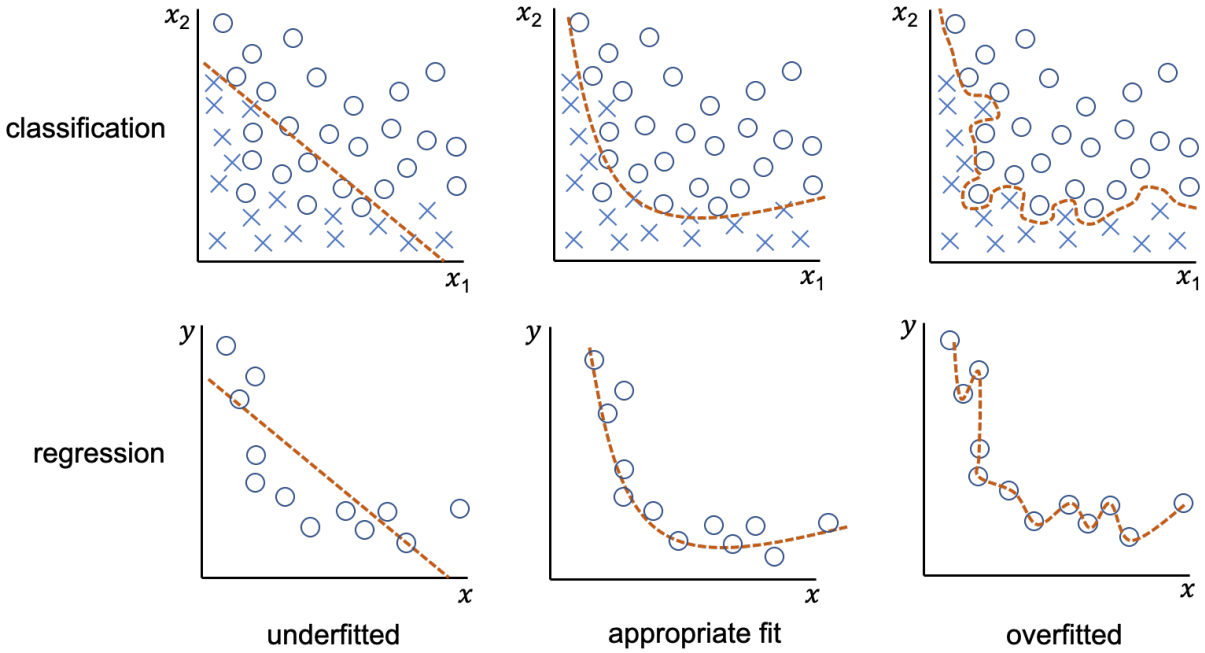


Figure 1.11: Examples of overfitting and underfitting. When a neural network is overfitted to the data, it may not generalize to new data well. When the neural network is underfitted, it may not provide satisfactorily accurate predictions. Underfitting can occur when too much regularization is introduced or when the neural network architecture is not expressive enough for the function to be learned (e.g. there are not enough layers and nodes). Overfitting can occur when the neural network architecture is too expressive, when too little data is used in training, and when insufficient regularization of the weights is used. Overfitting is a serious issue and can result in catastrophic failure on a task when new examples are presented.

A Note on Training Data Distributions

Note that training, validation, and testing data are often all randomly drawn from the main data set, and thus will have similar statistical distributions (e.g. it is common for samples to be **independent and identically distributed**, or i.i.d.). However, in real-world applications (such as accelerators), it is often the case that predictions will need to be made under shifting condi-

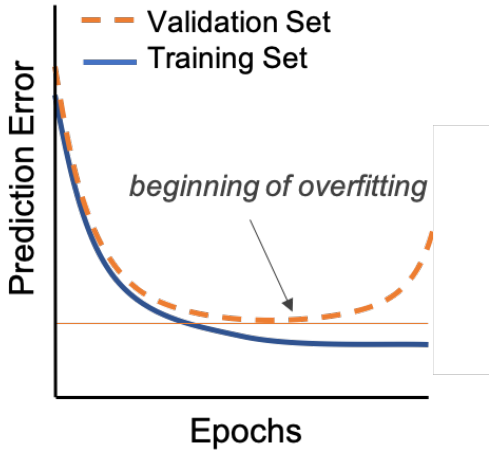


Figure 1.12: Example of learning curves and overfitting. This shows a simple case, but in general the prediction error on the training set is expected to decrease with increasing training time (here shown as the number of elapsed epochs). When the prediction on the validation set begins to increase, it is an indication that overfitting has begun.

tions or in new contexts. This makes generalization to new cases (and avoidance of overfitting) extremely important. In cases like these, it can be beneficial to evaluate the model on data that comes from a completely different distribution or contains *deliberately held out segments* of the relevant input-output parameter space. This is not commonly done in general for ML at present; however, breaking the common mold, we take this approach for most of the studies presented in this dissertation, specifically because model generalization to unseen regions of parameter space is important for accelerator applications.

Regularization

Careful monitoring of the prediction performance on the training and validation sets during training, along with simply tuning the model architecture to an appropriate complexity for the data set, will help avoid overfitting and underfitting. In addition, there are several direct regularization strategies that are at present standard practice for neural networks. First, the magnitude of the weights can be penalized directly in the cost function. L1 and L2 norms are common choices. The p-norm is given by $||w||_p = (\sum_i |w_i|^p)^{\frac{1}{p}}$. Thus the L1 and L2 norm are given by $L1 = ||w||_1 = \sum_i |w_i|$ and $L2 = ||w||_2 = (\sum_i |w_i|^2)^{\frac{1}{2}}$. These can be included in the regularized cost function as

$$C_{reg} = E(y, \tilde{y}) + \lambda ||w||_{(1,2)},$$

where E is the chosen error metric between the predictions $\tilde{y}$ and ground truth y, and λ is a weighting factor that is empirically determined. Typical values for λ are much less than 1.

L1 will prioritize sparsity (bringing weights to 0), whereas L2 will prioritize many small weights quadratically distributed around 0. The intuition behind this can be seen from the plots of the L1 and L2 norm and their derivatives in Figure 1.13. It is important to balance the weighting of the penalty in the cost function carefully, because if it is too high it will prevent the neural network from learning a good representation of the data (i.e. it will underfit the data).

Next, in 2014 the technique known as "dropout" was introduced [88]. This approach removes connections from the neural network during each weight update (see Figure 1.14). During testing, all weights are used. Dropout in effect forces the neural network to learn a more robust representation of the data by distributing the predictive responsibility across the weights. It can also be used as a weak form of model ensembling (i.e. by keeping dropout active in the forward pass of the network for prediction and predicting over the same inputs many times, one will obtain a distribution of outputs). It has also been used to provide approximate prediction uncertainty estimates for neural networks [89].

Adding noise to the data in various ways is another effective method to combat overfitting. This can be done internally via noise layers that add random noise at each forward pass through the network. These are now well-integrated into popular neural network libraries such as `keras` and allow for custom noise amplitudes and distributions. As a form of data augmentation, noise can also be added to copies of the training data directly.

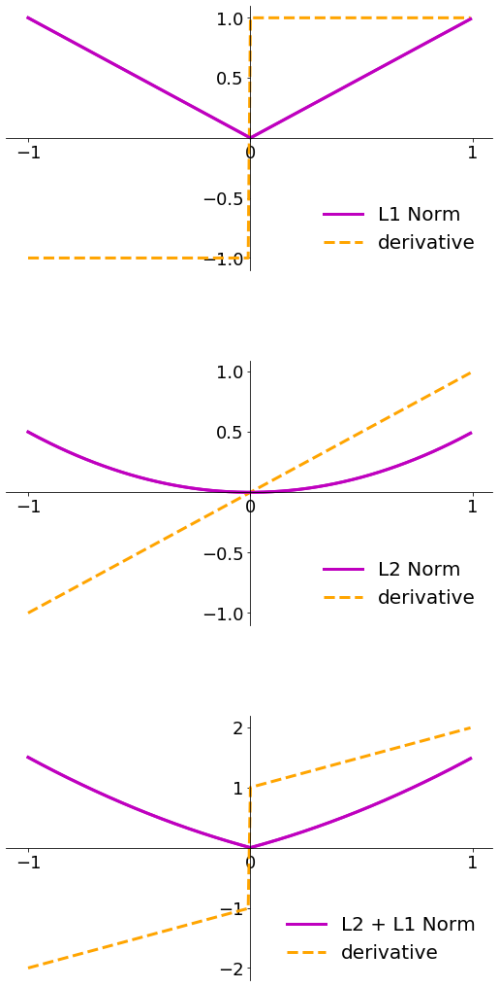
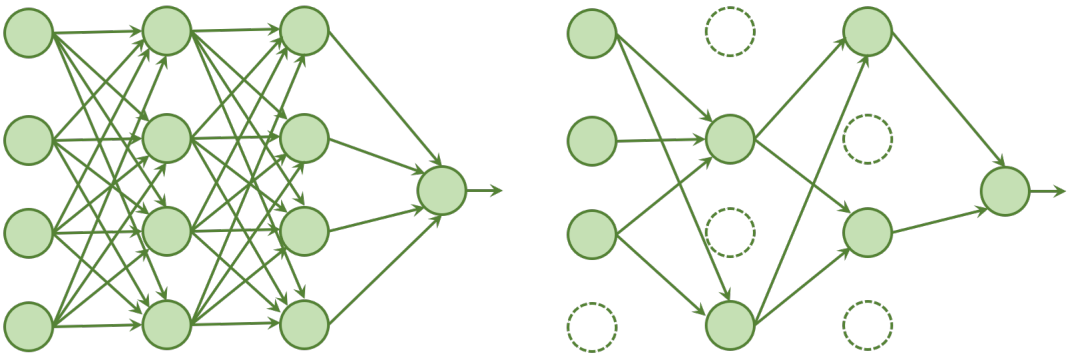


Figure 1.13: Intuition behind regularization using the L1 and L2 norm penalties on neural network weights. Plots for the L1 and L2 norms along with their derivatives for different weight values are shown. L1 regularization promotes sparsity in the weights and will bring weights to 0 more quickly, whereas L2 promotes many small weight values (and weight sharing).



(a) Standard fully-connected network

(b) Network with dropout applied

(c) Handling of nodes during training and testing

Figure 1.14: A standard densely-connected network (a), and a network with dropout applied (b). Nodes are kept with probability p during training and are present at all times during testing.

1.4 Relationship Between AI/ML and Optimization Methods Commonly Used in Particle Accelerators

In accelerator science, mathematical optimization is often used to adjust accelerator settings with the goal of producing particular beam characteristics. This occurs both in the context of online optimization for live operations and for offline optimization to aid experiment planning and to design new accelerator systems or components. For offline optimization, evolutionary algorithms such as genetic algorithms (GAs) [69, 70] and particle swarm optimization (PSO) [90] are popular choices (e.g. see [91–93] for examples). These algorithms are favored in part because they enable multiple objectives to be taken into account during optimization, they reliably find global optima (though this is not guaranteed), and they are relatively straightforward to implement. However, they are generally computationally expensive because many queries to the system to be optimized must be used (resulting in heavy reliance, for example, on parallel computation). They are also thus not favored for online tuning, but they have been used online in a few cases (e.g. [94]).These methods in their classical form do not learn an underlying representation of the system and thus are not considered to be ML-based methods. However, they attempt to mimic problem-solving processes found in nature (e.g. evolution and multi-agent swarms), so they are often considered to be an AI approach.

For online optimization of settings, gradient descent and variants, such as extremum-seeking (ES) [95–97] and robust conjugate direction search (RCDS) [98, 99], have been used extensively. Local heuristic methods such as Nelder-Mead Simplex [100] have also been used. These methods do not learn an underlying representation of the system, but rather, estimate the immediate local direction of improvement to choose subsequent samples in the search.

In contrast to standard optimization methods used by the accelerator community, ML-based optimization methods enable information about the underlying system behavior to be exploited more directly. For example, rather than searching the parameter space while only examining the performance at the present point, ML-based optimization methods can leverage information gleaned

from previously-observed points. This distinction is particularly important in cases where frequent switching between similar beam parameters is required; rather than always conducting a local search, an ML-based method can in principle interpolate between previously observed accelerator configurations to help find appropriate settings more quickly.

This can occur on a variety of scales, depending on how much information one wants to learn about the machine. For instance, one could try to learn a global machine model and use it to provide guesses of initial settings. However, even learning a local sub-system model as the search is conducted can aid tuning. For example, Bayesian optimization (BO) with Gaussian Processes (GPs) [101] is a model-guided optimization method that has recently (i.e. contemporary to this work) been used for online tuning [102]. This approach is appealing because it is very sample-efficient at sequentially optimizing black-box functions. In BO, an ML model (often a Gaussian Process model) is learned iteratively as each sample is taken, and the predictions of the model output and estimated uncertainty in the predictions are used to help inform where to sample next in the optimization process. An "acquisition function" is then used to select the next point (this is roughly analogous to the term "policy" mentioned above in the context of RL). For example, the upper confidence bound (UCB) acquisition function selects a point to sample that maximizes the sum of the objective and uncertainty (to try to sample potentially high-reward spaces). BO combines online system learning (active learning) with optimization, and it is intuitively similar to how a human might make a guess given a mental model about where to sample next. Gaussian Processes are often used in BO because they provide straightforward uncertainty estimates and can easily be updated iteratively with just a few samples. They are thus well-suited for learning a local model on-the-fly even with very little information about the overall system state (e.g. upstream changes to the accelerator are less important) or data from a machine archive. In contrast, neural networks are more computationally efficient at handling larger data sets than GPs, but they can be less straightforward to train online, and obtaining an accurate uncertainty estimate from a deep neural network is still an active area of research [89].

In terms of control and tuning with global models, a variety of methods are used that leverage simplified physics models and simple fits. For example, in correcting beam orbits, a standard method is to fit a linear response matrix to beam steering elements and beam position monitor responses (which can then be inverted to use in feedback control of the beam position) [103]. Similarly, simple physics models can calculate new magnet and accelerating cavity settings needed for beam energy changes. However, in all of these cases, differences between the model and the real machine (including offsets, nonlinear responses, etc) are not fully captured. Thus, another ML approach is to try to learn a better system model overall, which could then be used to provide better suggestions for machine settings. This however, makes the assumption that most of the variables needed to model the system response are actually even measured (an in some cases, they may not be; this varies from machine-to-machine). Along these lines, ML can also be used to form maps between a system state, desired target output, and the settings or setting changes needed to achieve that target output. In the case where a map from desired output to settings is made, this is an inverse system model that can be used to provide a warm start (i.e. an initial guess at settings) to local optimization methods.

Figure 1.15 roughly outlines some of these connections to help build intuition about what we mean in general when we describe AI and ML approaches in the context of accelerator physics and operation specifically. The delineation between these areas is of course not exact. For example, neural network-based approaches can be used to solve optimization problems as well (e.g. using neural networks in Bayesian optimization or neural network-based reinforcement learning, etc.).

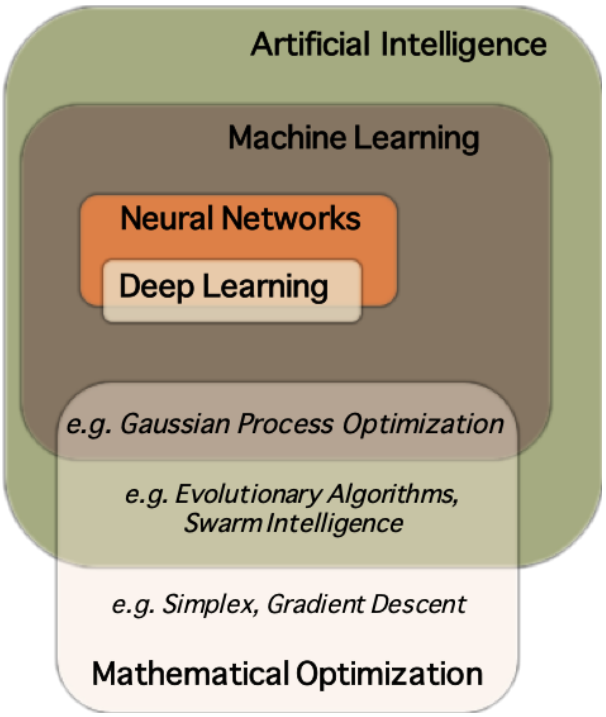


Figure 1.15: Broad categorization of terms relevant to AI/ML. The categories shown are not exact, but are aimed at building intuition for the connections between these areas. In accelerator physics, we often apply mathematical optimization to physical systems (e.g. optimizing the static accelerator settings to produce particular beam characteristics). This is what we are focusing on in the "mathematical optimization" box. For example, simple gradient descent or heuristic methods such as Simplex only use immediate local information from one sample to the next when searching a space; while these can find solutions to optimization problems, they are not ML-based approaches. Classical evolutionary algorithms (e.g. genetic algorithms) do not typically learn an underlying representation of the system. However, they are attempts to mimic problem solving processes found in nature (e.g. evolution and multi-agent swarming), so they are often considered to be an AI approach, but are not ML-based per se. In contrast, Bayesian optimization with Gaussian Processes learns an ML model iteratively to help solve an optimization problem. It also uses prediction uncertainty estimates to help inform where to sample next. Neural network-based approaches can be used to solve optimization problems as well (e.g. using neural networks in Bayesian optimization or neural network-based reinforcement learning), but they are far more commonly used in other settings (for example, supervised learning for computer vision tasks, control and game-playing tasks where dynamic evolution of states and actions must be taken into account, etc.).

1.5 Historical Context: Why Are neural network-based Approaches Now Relevant

1.5.1 Advances in Neural Networks

A full discussion of the recent advances in neural networks is beyond the scope of this work, but here we will highlight at a high level why neural network-based approaches have become popular in recent years for many real-world problems, and why these approaches are now technologically mature enough to be seriously applied to complex systems like particle accelerators.

Neural networks have had several boom-and-bust hype cycles; the early history of neural networks is described in [24]. With the introduction of new training methods in the mid-1980s (e.g. see [68] for discussion), multi-layer neural networks began to gain popularity, leading to many advances (running the gamut from computer vision to reinforcement learning [104] with neural networks). However, early real-world applications were limited by several factors, including limited available computing power for training, limited ability to fit large complicated networks and large data sets into memory, and difficulty in curating and sharing data sets. This led to an abandonment of neural network-based approaches in the 1990s.

Improvements in **computing technology** over the past decade have made successful implementation of more complicated neural network structures and their training algorithms feasible. These advancements have greatly increased the speed of training in general, allowing much larger training data sets and architectures to be used in practice. For example, training of CNNs on graphical processing units (GPUs) with improved computational efficiency started becoming increasingly common in the mid-2010s and has been critical in the success of CNN-based image processing algorithms. In addition, while not yet common, many advances have been made in implementing neural networks in hardware, such as FPGAs and neuromorphic chips, along with associated training algorithms (see [105–107] for examples and [108] for a comprehensive survey).

Another side effect of improvements in computing power and memory is improved **data set collection, and sharing of data and code**. Large data sets for training can now be easily collected,

stored, and accessed. Training data in many domains is now far more readily available and can be easily shared. The rise of the internet also has enabled large databases of data to be accessed by researchers world-wide, which both facilitates basic research and rigorous comparison of algorithm performance. Sharing of data and code has been a critical element in the rapid progress of neural network-based approaches to AI/ML.

Along similar lines, in the mid-2010s a variety of well-maintained **open source libraries** were introduced for constructing and training neural networks. This greatly helped to spur development by making it easier to share code and enabling newcomers to ML to readily use new neural network architectures and training algorithms. Libraries for auto-differentiation, like `Theano` [109], `TensorFlow` [110], and `PyTorch` [111], enabled custom neural network architectures to readily be built and efficiently trained with back-propagation. Higher-level wrapper libraries, like `lasagne` [112] and `keras` [113] were introduced in 2014 and 2015, and made the process even more accessible to novice users.

Critical advances in the **theoretical understanding** of how to effectively train various kinds of neural networks have also been made, along with the introduction of new learning methods and new neural network architectures. Much theoretical work over the past decade has been devoted to understanding the behavior of neural networks and developing more sophisticated architectures and associated training methodologies. Note that in the description of standard training practices described in earlier sections, many of these basic approaches were introduced in the mid-2010s.

Finally, the growing body of **experience with real-world applications** (e.g. see [44, 114] for many examples) has spurred further algorithmic development. The mid 2010s saw numerous advancements in neural networks, both in terms of new techniques and methods, and improvements in specific application areas like natural language processing and deep reinforcement learning for games. For example, in deep RL, algorithmic advances were made in deep reinforcement learning [115, 116], end-to-end RL with computer vision was developed and tested on robotic test stands (e.g. see [117], and for a historical survey of reinforcement learning in robotics see [118, 119]),

and super-human performance was achieved in StarCraft [120] and Go [121, 122] (both with and without leveraging human knowledge of the game).

The performance of CNNs in standard image recognition data sets is arguably the most widely-known example of neural network-based approaches starting to make revolutionary performance gains on real-world tasks. The introduction of very deep CNNs trained on large data sets, starting with AlexNet [123] in 2012 and with a series of others following [124, 125], began to overtake other methods in state-of-the-art performance on benchmark image recognition datasets such as ImageNet [126, 127] (see Figure 1.16). In 2015, human-level performance on the standard dataset was reached [67].

Combined, these advances enable significantly more complicated problems to be effectively addressed both in theory and in practice.

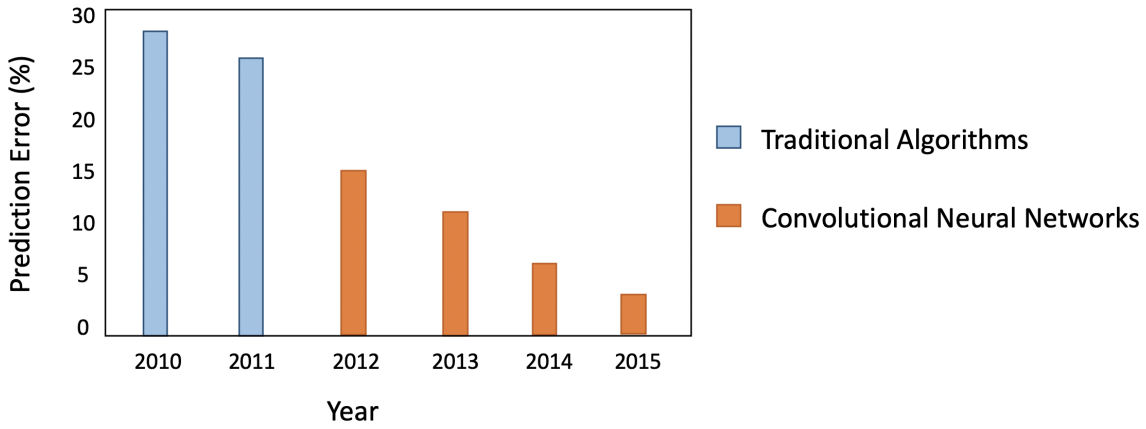


Figure 1.16: Performance of leading algorithms in the ImageNet challenge from 2010 to 2015. In 2012, algorithms based on convolutional neural networks took the lead.

1.5.2 Examples of Developments in Related Science and Engineering Fields

Numerous examples of success in using modern neural networks for various tasks are provided in [44, 114], but here we will also specifically highlight a few examples that intuitively relate to the problems found in modeling, diagnostic analysis, and control of particle accelerators. Neural network-based techniques to automatically process complicated measured scientific data have seen

great success in recent years. For example, they have become a useful tool in the analysis of astronomical data (e.g. see [128–131]). Some applications include image-based object classification from sky surveys and rejection of artifacts and interference in astronomical data. Neural networks have also proven useful in the analysis of data generated by high-energy particle physics experiments [132–135], which require detection and analysis of statistically improbable events from data sets with high backgrounds for event selection and particle identification.

Moving toward application of neural networks to real-time analysis of complex machines, an instructive example can be found in the recent literature surrounding fault prevention in tokamaks—a type of magnetic confinement device that is a promising candidate for thermonuclear fusion-based power production. In studies conducted within the last four years at the Joint European Torus, neural networks have shown promise in the modeling of tokamak behavior [136, 137], in instability detection [138, 139], and in disruption prediction [140, 141]. Furthermore, real-time early detection of potentially problematic features such as hot spots or instabilities through the use of neural network-based classification of video frames and has been demonstrated experimentally [142, 143].

The use of neural networks in process control (see [144] for a survey) has a somewhat older history of success. Examples of applications of neural networks in industrial control include the regulation of nonlinear chemical mixing processes [145, 146], temperature control of systems with nonlinearities and time delays (e.g. see [147]), optimization for energy savings in the temperature control of buildings (e.g. [148]) and self-tuning of traditional controllers such as proportional-integral-derivative (PID) feedback (e.g. [149, 150]). Approaches that specifically pair optimal control techniques with neural networks have also been the subject of numerous experimental and simulation-based studies (see, for example, neural network-based model predictive control [147, 151–157]).

1.5.3 Advances in Computing and Operations Infrastructure for Particle Accelerators

Advances in the infrastructure for modeling, control, and data acquisition for particle accelerators also has advanced significantly in the past 10 years, in ways that make them more amenable to ML-based solutions. Accelerator facilities are outfitted with a greater number and variety of diagnostics than was feasible in the past. Additional storage space also enables more historical data to be saved in the accelerator data archive. Newer diagnostic technologies enable finer measurements of beam distributions to be achieved (e.g. see [158] for a recent example). Finally, some important aspects of data acquisition systems, such as reliability of time-stamping across different acquisition devices, has begun to be addressed as ML has become a greater topic of interest.

In terms of simulation of particle accelerators, advances in modeling codes have increased the fidelity and computational efficiency of models (including fundamental algorithmic developments, for example see [159, 160]). In addition, the accessibility of high-performance computing platforms has improved the capability for large-scale simulation studies. Along with this, greater support for parallelization (e.g. see [161, 162]) and hardware acceleration (e.g. see [163]) of existing codes has developed.

Concurrent to these developments, improved support for open source frameworks (e.g. `keras`, `tensorflow`) and programming languages such as python have been incorporated into some accelerator control system environments. Challenges remain at many facilities in terms of computational power available directly on the accelerator control system, but this is improving.

1.6 Neural Networks and Particle Accelerators in the Context of this Dissertation

The idea of applying artificial intelligence and neural networks to particle accelerators is by no means a new one (e.g. see [164, 165]. During an initial wave of interest during the late 1980s and early/mid 1990s (near the peak of an AI hype cycle) efforts to apply neural networks to particle ac-

celerator systems obtained mixed results. As occurred with many other fields around this time, this quickly led to stagnation of efforts to apply AI to particle accelerators. Neural networks were investigated for orbit/trajectory control [166–168]. A neural network was successfully implemented to detect faulty beamline and diagnostic components [169]. In the early 1990s at Los Alamos, a neural network-based PID tuner for a low-level RF system was implemented [170]. Also at Los Alamos, several neural network schemes were used to control a negative ion source [171–173].

More recent work in the late 2000s used a combined feed-forward neural-network correction and PI controller to compensate for jitter at the Australian Synchrotron and the Linac Coherent Light Source [72,174]. Jitter in the upstream klystron phase and voltage was corrected using downstream settings, thus stabilizing electron beam energy and bunch length. In that control scheme, a neural network was used to predict future beam parameter deviations so that an appropriate correction could be applied. In another study, a multi-agent neural-network tuning tool was used to optimize machine settings for reduced electron beam energy spread and increased transmission at the Australian Synchrotron Linac [175]. This optimization agent was then used in a control experiment online at the FERMI@Elettra FEL to stabilize the beam energy. However, few comparisons to other methods were made, making it difficult to assess the utility of this approach.

It is clear from the recent achievements in neural network-based approaches to computer vision tasks, generative modeling (e.g. to produce images), control, and game playing that the following application areas in particle accelerators are now poised to benefit from the progress made in the last decade:

- **Faster simulations and more accurate machine models:** improving the speed and accuracy of accelerator models would open up new applications in online modeling and model-based control, and it would facilitate offline experiment planning.

- **Physics discovery / discovery of machine sensitivities:** learned models could also aid investigation of machine sensitivities that might not be represented in physics-based models.

- **ML-enhanced diagnostics:** Neural networks could be used to provide non-invasive predictions of beam parameters and phase space distributions (i.e. "virtual diagnostics) and facilitate sophisticated analysis of complex beam signals.

- **Tuning and control:** Neural networks could be used to facilitate rapid switching between beam parameters using inverse models, aid online control with model-based control routines, and aid online optimization of accelerator settings using techniques like deep reinforcement learning. An exciting area related to this is the ability that neural networks have to directly process image data as part of learned control policies (which could be used in conjunction with image-based beam diagnostics to potentially achieve finer control).

- **Anomaly detection and failure prediction (prognostics):** Neural network-based approaches could aid diagnosis and prediction of anomalous machine states or failures.

All of the areas above, with the exception of anomaly detection and failure prediction, are examined in this dissertation. The context for this work is important: in 2012, there was little interest in AI/ML in the particle accelerator community, a great deal of skepticism, and very little understanding of what the use cases might be. This is understandable - many of the advances in neural networks and deep learning that enabled its present success *had not yet been made* at that time. As such, this work was both an exploration of possible application areas, a generator of proof-of-concept demonstrations in these areas, and an exploration of common pitfalls that are relevant to accelerators (e.g. common time-stamping, calibration, and interfacing issues that are relevant to many accelerator data acquisition system standards). This work was also tightly coupled with many outreach efforts to encourage wider interest and support within the accelerator community. Finally, in early 2018, the first set of community workshops on AI/ML applications in particle accelerators were held [176], where this work was one of the few examples at the time of modern applications of neural networks to problems in particle accelerators.

Chapter 2

Accelerator System Modeling: Faster Simulations, Image-based Diagnostics, and Bridging the Gap Between Simulation and Empirical System Behavior

2.1 Introduction and Motivation

Accelerator systems are often extensively modeled with first-principles, physics-based simulations during the design stage. However, these models are rarely used to directly aid operation of accelerators, especially in cases where nonlinear beam effects like space-charge are important. This situation arises for two main reasons: simulation speed and accuracy.

First, simulations that include more of the relevant beam dynamics effects (e.g., space charge, coherent synchrotron radiation) are often too computationally intensive to execute in a useful amount of time on the live accelerator. This is exacerbated for older accelerator facilities which may have limited computational resources that are accessible in parallel with the accelerator control system. Online models can be used both for offline optimization (e.g. for experiment planning/design of new setups) and online tuning (e.g. using an optimizer with the model during operations to quickly converge to settings that are likely optimal). The computational burden of simulations also limits the extent of detailed optimization and experiment planning. The issue becomes more pressing in cases where collective effects, which are costly to model, begin to dominate the beam dynamics.

Second, predictions from physics-based simulations often deviate substantially from the observed machine behavior due to many compounding errors in the descriptions of the accelerator components. There are many sources of uncertainty in the operational accelerator, including calibration errors, hysteresis in magnet and motor settings, variable noise, misalignments in compo-

nents, and drift (in calibrations, observed variables, and unobservable variables). These discrepancies limit the usefulness of simulations for use in accelerator operations.

Machine learning is likely to play a significant role in improving both the execution speed and accuracy of physics simulations relative to the observed machine behavior. One solution is to learn a fast-executing representation of the beam dynamics based on a sparse sampling of the high-fidelity physics simulation, preserving the predictive power of high-fidelity simulations for online use and optimization. This could then potentially be calibrated to match the machine.

In addition, at present, a variety of image-based diagnostics are used in particle accelerator systems. Often times, these are viewed by a human operator who then makes appropriate adjustments to the machine. In some cases, scalar estimates of values of interest are derived from fits and then used in automatic feedback control. In both cases, some of the information present in the image is lost or not fully utilized. Similarly, image-based diagnostics also provide useful information about the present state of the machine. For example, images of the transverse distributions of photocathode drive lasers are used to provide some idea of what the initial electron beam shape might look like. Given recent advances in using convolutional neural networks (CNNs) for image processing, it should be possible to use image-based diagnostics directly in system models and control routines. CNNs have yielded impressive results in the area of computer vision, especially for image recognition tasks [44]. They are also starting to be used in physics-related applications, such as automatic classification of galaxies [177] and neutrino events [178]. Given the present success of CNNs, it may now be possible to use them as a means of incorporating image diagnostics directly into particle accelerator control systems. One could imagine incorporating images directly into a neural network-based control policy, as has been done recently in robotics [117].

Addressing these issues and corresponding open questions, this chapter shows proof-of-principle results in the area of using neural networks to enhance simulations and online predictions. This includes:

- Using image-based diagnostics as model inputs.

- Using a neural network to provide a virtual diagnostic prediction (in this case for an image-based multi-slit emittance measurement).
- Updating a neural network model that was pre-trained in simulation with measured data. This improves the model accuracy with respect to the machine behavior while retaining predictive performance in regions of parameter space not seen directly in the measured data.
- Significantly speeding up accelerator "simulations" (e.g. by a factor of 10^6) using neural networks to approximate the observed input-output relationship.

2.2 The FAST Low-Energy Beamline

For these studies, we focused on the low-energy beamline at the Fermilab Accelerator Science and Technology Facility (FAST). The electron linac at FAST [179, 180] was commissioned up to 300 MeV at the time of these studies, with the aim of providing electrons for the Integrable Optics Test Accelerator (IOTA) [181]. In the low-energy beamline, a 1.3-GHz PITZ-style photoinjector and two superconducting capture cavities (CC1 and CC2) provide a maximum beam energy of 50 MeV, after which a TESLA Type IV ILC-style cryomodule accelerates the beam to full energy.

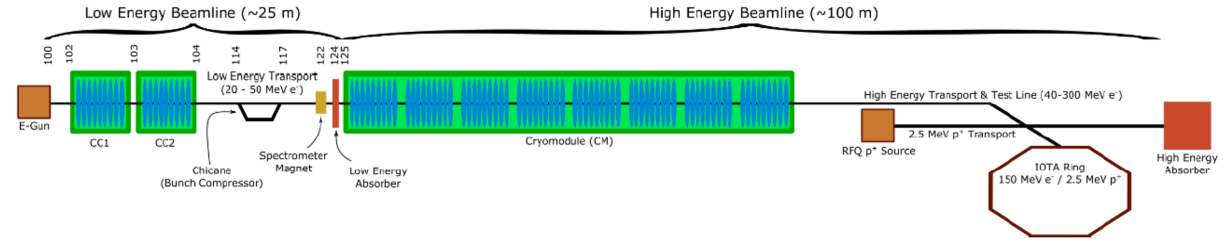


Figure 2.1: Layout of the Fermilab Accelerator Science and Technology Facility (FAST).

2.3 Incorporating Image-based Diagnostics as Model Inputs

For a given laser system it is not always easy to produce a top-hat transverse laser profile (which is a preferred transverse distribution used to obtain low-emittance beams), and any non-idealities in the initial laser distribution can impact the electron beam parameters for the rest of the machine. This includes, for example, the beam emittance, which one would typically like to minimize at the photoinjector. There is little control at FAST over the transverse profile of the photocathode drive laser, and it drifts substantially over time. Consequently, a skilled human operator manually tunes the photoinjector phase and solenoid strength to compensate for changes in the laser spot. The laser spot is observed on a virtual cathode camera (VCC), which is a camera placed at a location equivalent to that of the cathode and viewing light from the laser split off with a beam splitter.

Along with conducting scans to determine the appropriate photoinjector settings on a given day, over time an operator may develop intuition about how the settings need to be adjusted based on the VCC image. In principle, one could also take the VCC image and longitudinal laser information, convert it to an approximate initial beam distribution, and run a physics simulation to optimize the settings needed for that particular laser distribution. In practice however, it is impractical to run a computationally expensive simulation code that includes the relevant nonlinear effects for the low-energy injector system online during operation. This means that one cannot feed the observed laser profile to the physics simulation and optimize the settings on-the-fly.

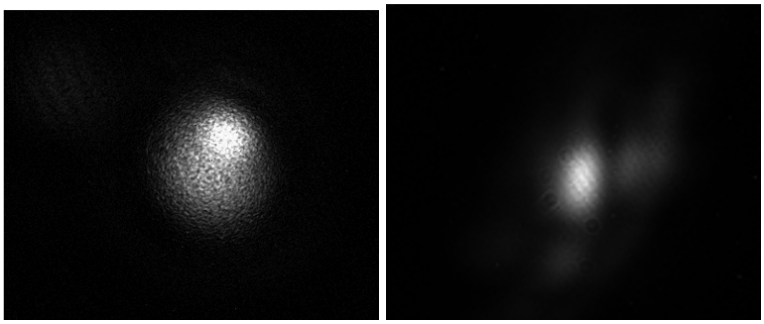


Figure 2.2: Example of changes in laser spot, as seen on the VCC, for nominally the same setup.

As such, it could be useful to have a neural network model that can take a measured laser distribution image (here, the VCC image) and controllable settings (such as the photoinjector phase and solenoid strength) as inputs to predict the downstream beam parameters. This could then be optimized on-the-fly to yield optimal photoinjector phase and solenoid strength settings. In principle, if one had fine control over the transverse laser distribution itself, one could also include it as a controllable parameter. Another approach (assuming there was no direct control over laser distribution beyond hand adjustment of the laser optics) would be to develop an inverse model that maps the observed VCC image directly to an initial guess at the optimal photoinjector settings.

As a first step, we decided to investigate whether a neural network can adequately predict downstream beam parameters from various input distribution images, photoinjector phase settings,

and solenoid strengths. We based this study on physics simulations of the photoinjector up through the exit of the second superconducting capture cavity, shown in Figure 2.3.

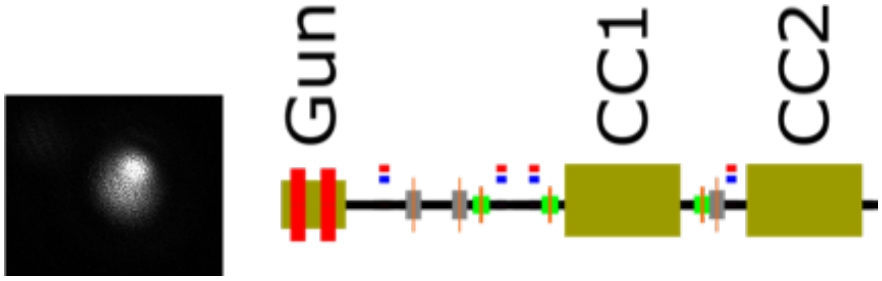


Figure 2.3: Portion of the FAST low-energy beamline simulated for this part of the study, including the photoinjector, an initial laser distribution (as would appear on the VCC image), the photoinjector (here denoted as the “gun"), and the remaining beamline up through the second superconducting capture cavity (CC2).

2.3.1 Simulations and Data Sets

Simulations of the first 8 meters of the FAST low-energy beamline, shown in Figure 2.3, were conducted using `PARMELA` [182]. Included in the simulations are the photoinjector, both superconducting capture cavities, and the intermediate beamline elements (steering and focusing magnets). The locations of the cavities and beamline elements were taken from a Fall 2015 mechanical survey. Two sets of simulation scans were conducted: one set of fine scans to predict beam parameters after the photoinjector, and one set of coarse scans to predict beam parameters after the second superconducting capture cavity (CC2). The photoinjector phase was scanned from -180° to 180° in steps of 10° and 5° for each case respectively, and the solenoid strength was scanned from 0.5 to 1.5 (relative strength) in 10% and 5% steps, where 1.0 represents the nominal setting that produces the peak axial field of 1.8 kGauss. Prior to scaling of the field maps, the bucking coil was tuned to produce zero magnetic field on the cathode. The field maps of the solenoid assembly, photoinjector, and capture cavities used for the `PARMELA` simulations were generated using `Poisson Superfish` [182, 183]. Examples of the scan data are shown in Figure 2.4.

For the photoinjector studies, we used three initial top-hat beam distributions with different radii. For the simulations up through CC2, we used nine different beam distributions. Three of these were the same top-hat distributions used for the photoinjector simulations, and the other six were Gaussian transverse distributions with different RMS widths in x and y (σ_x and σ_y). These distributions were also converted into the simulated VCC images (e.g. see Figure 2.5) that were later used to train the neural network. Prior simulation results using initial beam distributions derived from measured virtual cathode images suggest this is a sound approach (i.e. the cathode quantum efficiency is likely sufficiently uniform in this case that the VCC image provides a reasonably good representation of the initial beam). For all cases, the longitudinal distribution of the beam was Gaussian with a bunch length of 3 ps. The ranges of the produced data are shown in Table 2.1.

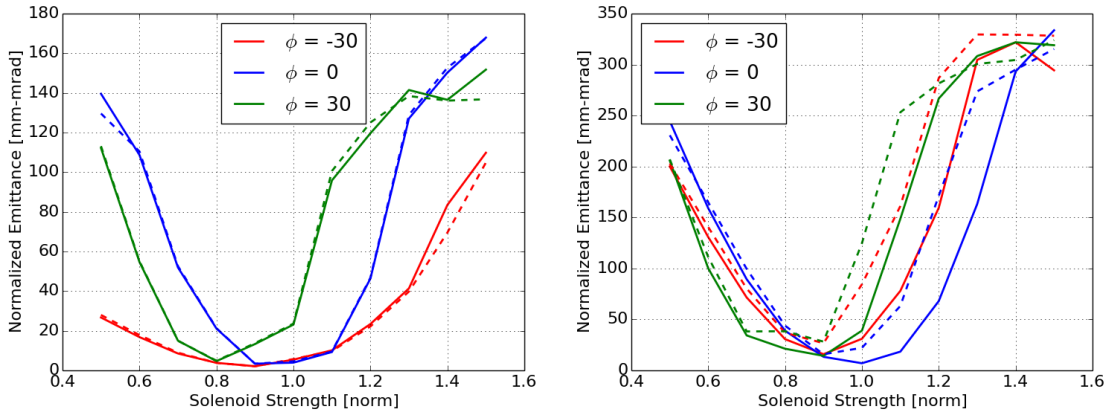


Figure 2.4: Examples of the transverse emittances as a function of the normalized solenoid strength for three different photoinjector phases. Dashed lines denote ϵ_{nx}, and solid lines denote ϵ_{ny}. The left plot is for a top-hat initial beam distribution with a width of 0.6 mm. The right plot is for a Gaussian initial beam distribution with σ_x of 0.6 mm and σ_y of 1.2 mm. These initial beam distributions were chosen because they are idealizations of two common ones that are typical in accelerator systems (often, an approximate flat top or super-Gaussian are aimed for in practice).

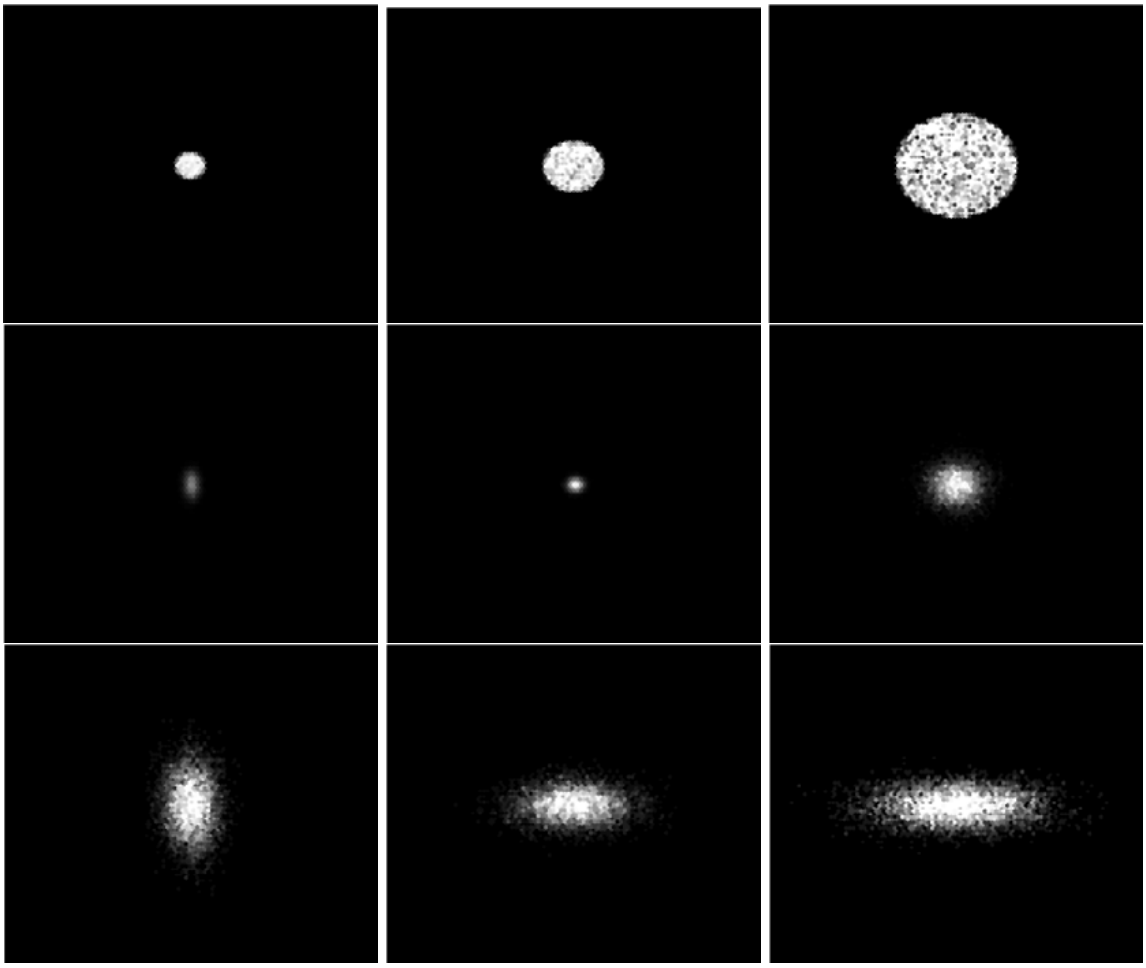


Figure 2.5: Transverse initial beam distributions used in the study. These are also used as the initial images to approximate the VCC images of the photocathode drive laser.

Table 2.1: Max and Min Values for Simulated Parameters

Parameter	Unit	Max Gun	Min Gun	Max CC2	Min CC2
Macroparticles	count	5001	1015	5001	1004
ϵ_{nx}	[m-rad]	2.5e-4	1.6e-6	4.0e-4	9.1e-7
ϵ_{ny}	[m-rad]	2.4e-4	1.6e-6	4.0e-4	8.5e-7
α_x	[rad]	14.1	-775.1	0.8	-149.8
α_y	[rad]	14.5	-797.0	0.7	-154.5
β_x	[m/rad]	950.4	7.9e-2	820.2	0.7
β_y	[m/rad]	896.8	8.4e-2	845.7	0.81
E	[MeV]	4.6	3.2	47.2	42.8

2.3.2 Neural Network Modeling

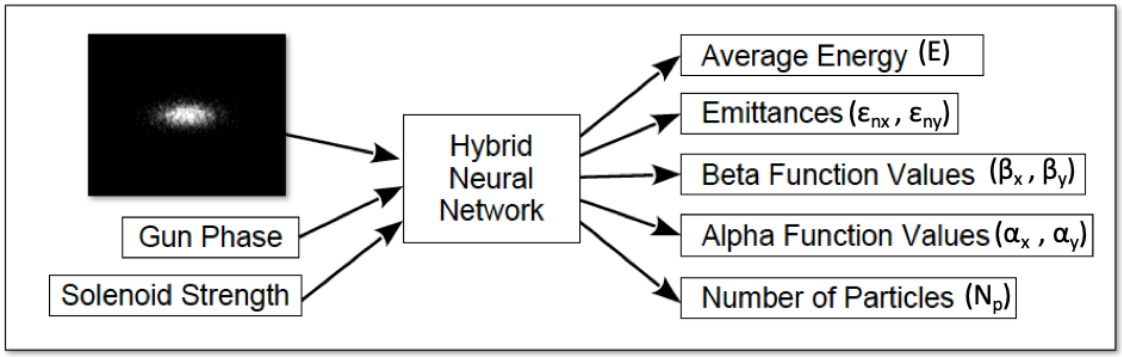


Figure 2.6: Neural network structure. The network takes in a VCC image (of the transverse laser distribution), photoinjector phase, and solenoid strength, and predicts the downstream beam parameters. One model predicts the outputs at the exit of the photoinjector, and another model predicts the outputs at the exit of the second superconducting capture cavity. The network itself combines a segment with convolutional layers for processing the image and a densely-connected section that maps both the scalar setting inputs and the output of the convolutional layers to the scalar beam parameter outputs.

Here we use a CNN to predict the downstream beam parameters after the photoinjector and CC2, given the photoinjector phase, solenoid strength, and VCC image (see Figure 2.6). We also adopt a hybrid structure that joins a set of convolutional layers and a set of densely-connected layers to incorporate both image-based data and the scalar data into the model (this also was a relatively novel approach at the time of the study). In this hybrid neural network the photoinjector phase and solenoid strength inputs bypass the convolutional layers of the network and are later combined in a final densely-connected set of layers. The outputs are the number of transmitted particles (Np), the transverse emittances (ϵ_{nx}, ϵ_{ny}), the average beam energy (E), and the beam transport envelope functions (α_x, α_y, β_x, β_y).

A variety of neural network structures were examined (including various number of layers, number of filters per layer, filter sizes, activation functions, etc.). The portion of the network for image processing consists of 3 convolutional layers: 16 5x5 filters, followed by 16 3x3 filters, followed by 10 3x3 filters. These are followed by 3 densely-connected layers with 150, 70, and 8 neurons respectively. With the exception of the linear output layer, hyperbolic tangent activation functions are used. The neural network is implemented in `lasagne` [112] and `Theano` [109].

Note that we found ReLU activation functions (which one might expect would be the activation function of choice for a CNN) performed poorly in this case, though after some assessment it is still unclear why; this could use further investigation.

The model was trained using a combination of the Adadelta [74] and Adam [75] optimization algorithms, and the network weights were initialized using the layer-by-layer method described in [39] with a uniform distribution. The network biases were initialized using a normal distribution with standard deviation of 0.01 and a mean of 0.

For the photoinjector data, the training set consisted of 1395 simulation data points and the validation set consisted of 200 simulation data points. For the CC2 data, the training set consisted of 894 simulation data points and the validation set consisted of 600 simulation data points. Note we were data-limited in this case because of the computational expense of the simulations; it takes approximately 15-20 minutes for one forward pass through the simulation on 1 core of an Intel Core i7 processor. To augment the data, we quadrupled the size of the training data set by adding Gaussian noise to both the images and the scalar inputs and outputs, and we took care to avoid overfitting by closely monitoring the validation loss. After training the photoinjector network, we used its weights as an initial solution to begin training the CC2 network (i.e. **transfer learning**) to allow us to make do with the smaller amount of data we had for that case.

Table 2.2 and Table 2.3 show the neural network's performance in terms of mean absolute error (MAE) over the training and validation sets, for the photoinjector and CC2 models respectively. For the photoinjector, all MAEs are between 0.4% and 1.8% of the parameter ranges, and for CC2, all MAEs are between 0.9% and 3.1% of the parameter ranges. In Figure 2.7, we highlight two representative data sets to show the neural network's performance in predicting downstream beam parameters.

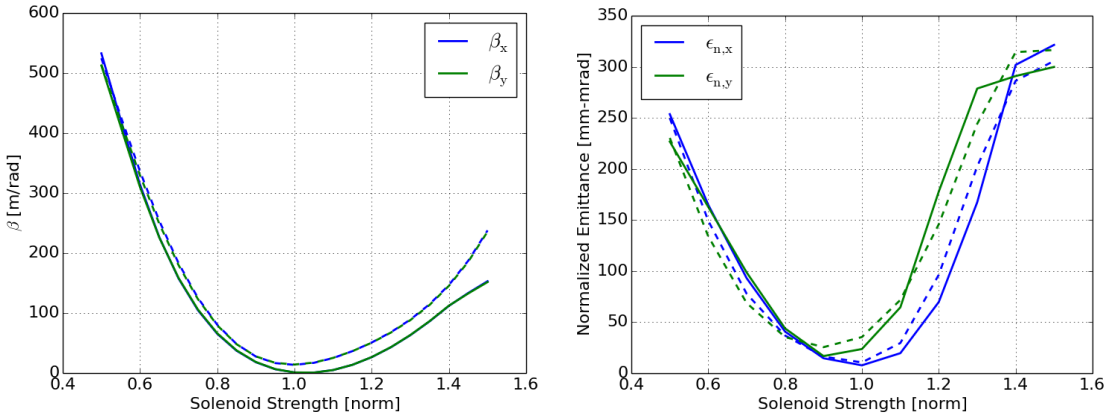


Figure 2.7: Examples of predictions for two beam parameters. The left plot shows transverse beta function values after the photoinjector as a function of solenoid strength for a top-hat initial beam distribution. The right plot shows normalized transverse emittances after CC2 as a function of solenoid strength for an asymmetric Gaussian beam distribution. Both are for an RF phase of 0 °. The dashed lines are neural network predictions and the solid lines are simulated values.

Table 2.2: Model Performance at Photoinjector Exit

Parameter	Units	Train MAE	Test MAE
Macroparticles	[count]	69.5	70.7
ϵ_{nx}	[m-rad]	2.3e-6	2.4e-6
ϵ_{ny}	[m-rad]	2.3e-6	2.4e-6
α_x	[rad]	9.0	10.9
α_y	[rad]	8.8	10.8
β_x	[m/rad]	12.1	14.8
β_y	[m/rad]	11.7	14.3
E	[MeV]	4.9e-3	5.5e-3

Table 2.3: Model Performance at CC2 Exit

Parameter	Units	Train MAE	Test MAE
Macroparticles	[count]	103.7	123.3
ϵ_{nx}	[m-rad]	1.0e-5	1.2e-5
ϵ_{ny}	[m-rad]	1.0e-5	1.2e-5
α_x	[rad]	3.4	3.1
α_y	[rad]	3.4	3.1
β_x	[m/rad]	16.3	14.7
β_y	[m/rad]	16.4	14.8
E	[MeV]	4.0e-2	4.6e-2

2.3.3 Conclusion and Future Work

We have shown for the first time in a particle accelerator setting that a combined convolutional and densely-connected neural network is capable of predicting simulated downstream beam parameters. In this case, solenoid strengths, photoinjector phases, and simulated virtual cathode images are varied widely as inputs. This was done only in simulation due to limitations in obtaining a sufficiently broad set of measured data; however, even showing this in simulation is an important first step toward creating a neural network model (or associated controllers) that can directly use image-based diagnostics along with scalar settings to predict output beam parameters. This could be useful for doing fast online optimization of accelerator settings to compensate for observed changes in the transverse laser distribution. Despite the small number of samples in the training set and the large number of predicted parameters, the network performs well. All errors are between 0.4% and 3.1%, and this is likely to improve with additional training data. Adding training data would also enable more thorough investigation into how well the model generalizes (e.g. using a very different laser distribution for the VCC image input and conducting scans with that).

In addition, the neural network model executes in under 1ms, in contrast to the `PARMELA` simulation, which takes around 20 minutes for one forward pass on 1 core of an Intel Core i7 processor. The neural network model itself thus could already be used on its own as a fast-executing stand-in for the `PARMELA` simulation to facilitate rapid optimization studies or to be used as an online model.

Beyond generating more data and more fully exploring the model performance for new VCC images, future work could include taking measured VCC images from the FAST accelerator and using these as inputs. Depending on the observed accuracy with respect to the empirical machine behavior, this may already be able to provide suggested settings for the photoinjector given drift in the laser distribution. However, it is expected that the model would need to be updated with measured data from the machine in order to be used in this fashion.

2.4 Virtual Diagnostics

2.4.1 Motivation

Many measurements of the beam parameters in charged particle accelerators rely on diagnostics that are either slow to update or destructive to the beam. In cases where the beam is needed for downstream experiments, destructive diagnostics cannot be used during normal operation. In cases where a measurement is also time-consuming, valuable information about the beam during a given machine state is unavailable in between these measurements. If a diagnostic measurement is both destructive *and* time-consuming, measurements may only rarely be made, particularly if there are high demands on beam time. When the configuration of the accelerator is relatively consistent and its responses are stable over time, augmenting some of these diagnostics with an online estimate of their output could reduce the need to interfere with the beam as frequently and allow otherwise missing information to be obtained between diagnostic updates.

A **"virtual diagnostic"** provides an estimate of what a particular measurement would likely show when that measurement is unavailable. This allows non-invasive estimates of measurements to be made in locations where it may be impossible to put an instrument, where only destructive instruments are available and thus cannot be used during normal operations, where the update rate of an existing instrument is lower than needed for experimental analysis or for control, or where the instrument is not sensitive across the entire operating range.

One approach to create a virtual diagnostic is to use high performance computing resources and specially-designed simulation codes that can take advantage of hardware acceleration (as is done, for example, with GPU-accelerated `PARMILA` [163, 184]). Another approach is to use fast, simplified physics-based models that have been empirically tuned to make a more accurate prediction [185]. Finally, another approach toward creating a virtual accelerator diagnostic, which we introduced with the following work, is to learn a fast-executing representation of the machine and its diagnostic outputs with a learned model.

In the following sections, we introduce an ML-based virtual diagnostic scheme that shows promise for providing online estimates of beam parameters and images resulting for a multi-slit

phase space diagnostic at the FAST low-energy beamline Figure 2.8. At the time of these studies (2016-2017), this was the first proof-of-concept demonstration of this approach aimed at predicting an accelerator diagnostic output based on wide parameter scans to produce a machine model. Concurrent to this study, the same approach was independently examined for filling in information between slow-to-update user measurements at the LCLS beamline [186] and now is being further investigated at LCLS and FACET-II for longitudinal phase space prediction [187]. While much work remains before the system described in the following sections would be viable to use at FAST, these studies represented important first steps in demonstrating the capability and the concept.

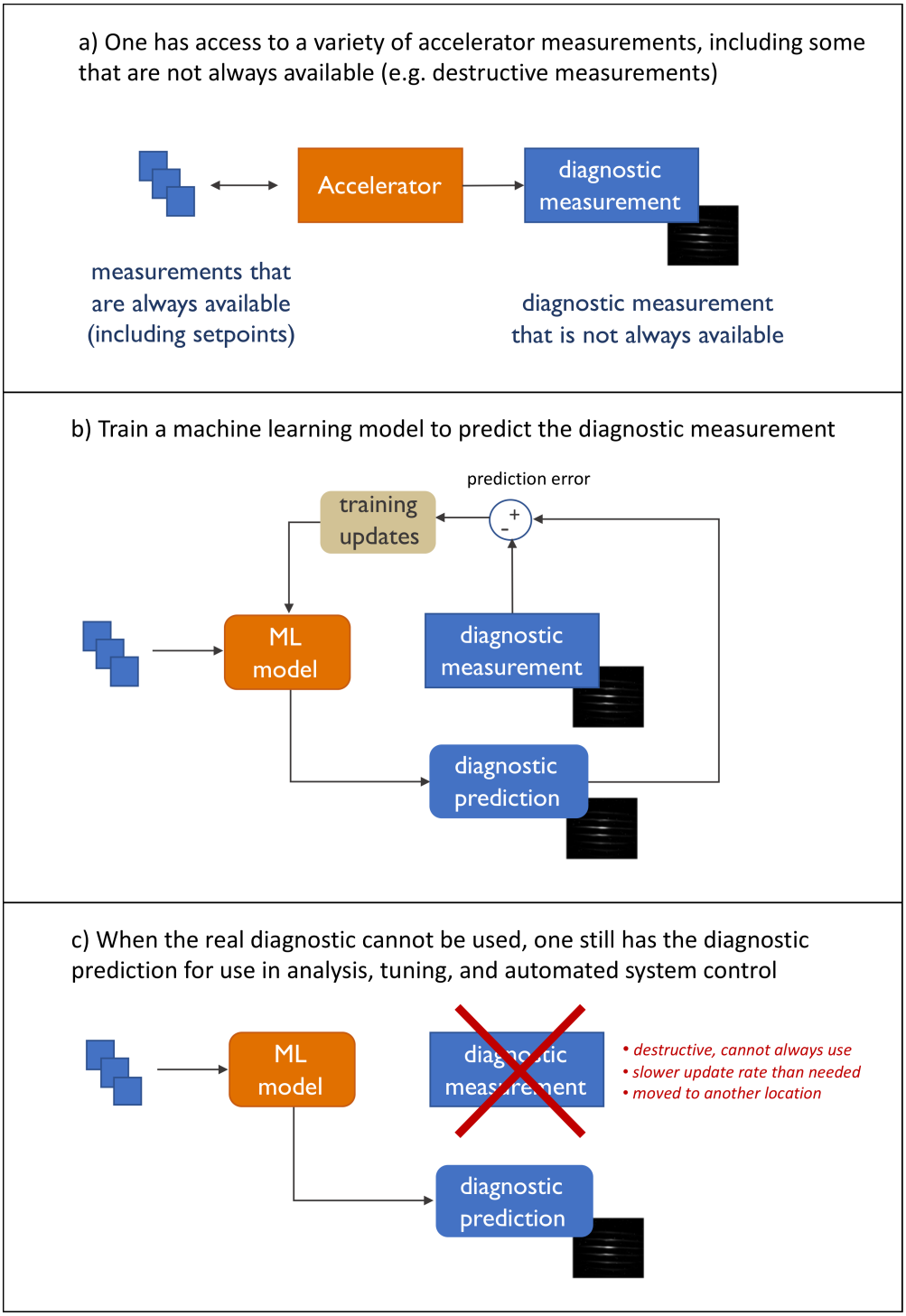


Figure 2.8: Machine learning-based virtual diagnostic. The model uses available measurements from the machine to predict the output of a destructive measurement, such as the multi-slit emittance measurement at FAST. Such online predictions could be used in hand tuning, data analysis, and automated feedback control.

2.4.2 Transverse Phase Space Measurements at FAST

At FAST, the emittance of the beam is measured using a multi-slit emittance measurement setup that was installed during the 2016 run (see Figure 2.9). This diagnostic works by passing the beam through a set of slits with a fixed width and spacing. Taking an image of the beam at a fixed distance from the slits enables calculation of the beam's transverse phase space based on the geometry of the setup and expected beam dynamics (e.g. beam divergence as it travels between the slits and the screen). A fitting procedure described in [188, 189] is used with the beam image and diagnostic geometry to calculate the phase space parameters. In this case, a lack of x-y coupling and a roughly Gaussian beam are assumed. Deviation from these assumptions is one of the sources of error when computing the emittance. At FAST, the setup consists of two separate masks for the vertical and horizontal planes that can be inserted into the beamline, after which the beam is projected onto a screen at X11 and the resultant image is recorded. Examples of measured images are shown in Figure 2.11.

Measuring the emittance this way is time-consuming. At FAST it takes about 15 seconds for each measurement plane to insert the slit mask and take the image. As a destructive measurement, it also blocks use of the beam downstream. This means that when trying to study the machine (e.g. dynamics through the cryomodule and bunch compressor) or conduct specialized experiments like the creation of flat beams [190], one does not have the luxury of non-invasively seeing what the upstream beam conditions are like relative to downstream measurements. As long as the beamline is sufficiently stable in its responses, it may be possible to construct a virtual diagnostic that can provide a quick estimate of what the multi-slit emittance measurement would show if actually measured.

In the virtual diagnostic scheme pursued here, shown in Figure 2.10, the inputs are the solenoid current, the rf phases of the photoinjector and the two super-conducting capture cavities, and the properties of the initial bunch at the cathode (charge, length, transverse emittance, and the x-y correlation in the emittance). The initial transverse bunch properties are inferred from virtual cathode images of the laser spot. It is worth noting in this case we are using only settings and

upstream inputs (e.g. the laser information), and so in addition to being a virtual diagnostic that predicts output from an invasive measurement, this is also an online model that could be used as a stand-in replacement for the first-principles physics simulation. Because it was difficult to obtain a comprehensive measured data set due to limited machine time and slow measurements, we first conducted studies in simulation and then used transfer learning [29,30] to reduce the error between the model and the real machine.

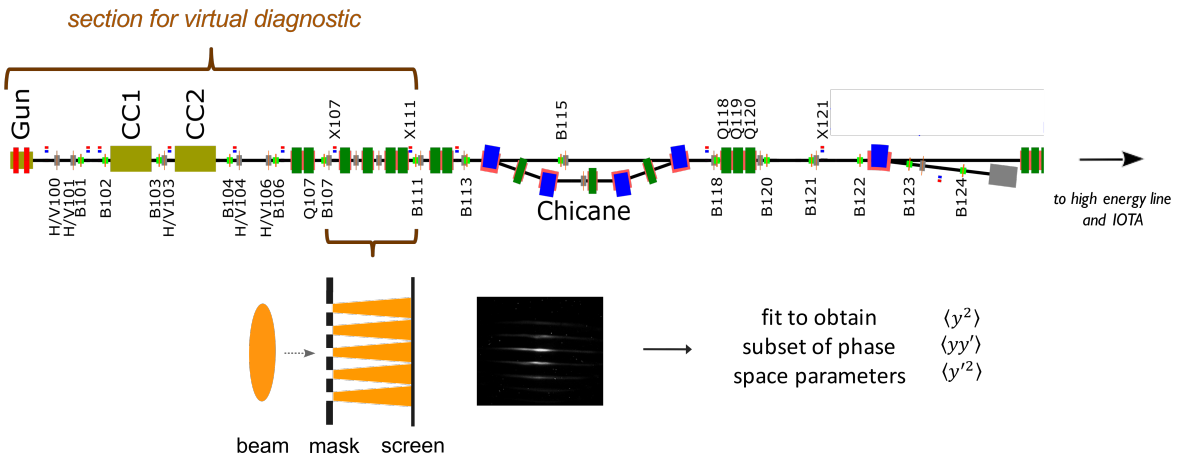


Figure 2.9: Multi-slit emittance measurement scheme at FAST. The beam hits a multi-slit mask and is then projected onto a screen. Based on a set of Gaussian fits to the projections, some of the phase space parameters can be determined. At FAST, a set of horizontal and vertical slits are used separately to measure the x- and y- phase space parameters.

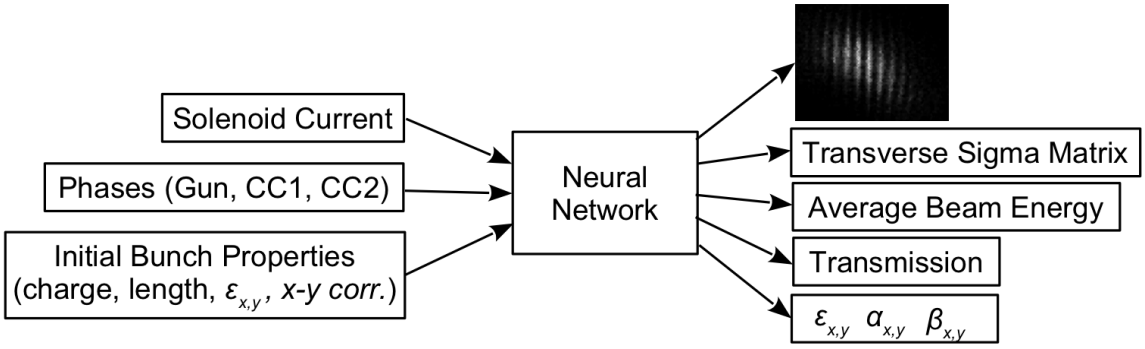


Figure 2.10: Corresponding concept for neural network-based virtual diagnostic of the multi-slit emittance measurement at FAST. The output includes scalar values for bulk beam parameters and direct image output. The initial transverse and longitudinal bunch parameters used as inputs can be inferred from the virtual cathode image of the laser spot at FAST.

2.4.3 Initial Measurements and Simulations

Solenoid and phase scans of the gun were performed at two different bunch charges (135 pC and 250 pC), while taking both horizontal and vertical multi-slit emittance measurements. The range of the solenoid current was 290 A to 306 A, and the range of the phase values was 174 ° to 212 °. In total 273 individual measurements were recorded (i.e. including both vertical and horizontal measurements separately). Examples of the measured images are shown in Figure 2.11, and examples of the scan data are shown in Figure 2.13.

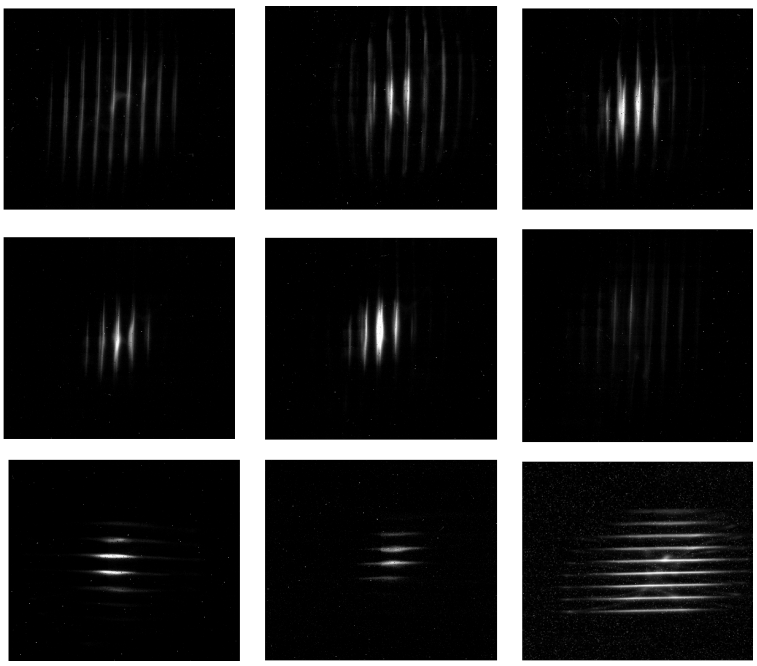


Figure 2.11: Example multi-slit measurement images from FAST for different combinations of gun solenoid strength and phase settings.

Simulations of the low-energy electron beamline were conducted using `OPAL` [161], an open-source particle-in-cell code. `OPAL` is well suited for this problem because it can handle 3-D space charge for low-energy electron beams and has been well-benchmarked against other simulation codes. The field maps for the cavities were generated using `Poisson Superfish` [183], and the solenoid and bucking coil were simulated in `Pandria`. During these studies the ratio between

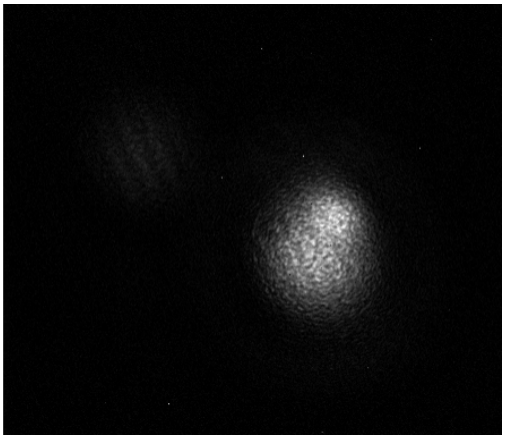


Figure 2.12: Image of the transverse laser profile measured with the virtual cathode camera at FAST. This was used directly along with the bunch length measurement to generate the initial electron bunch used for the simulations.

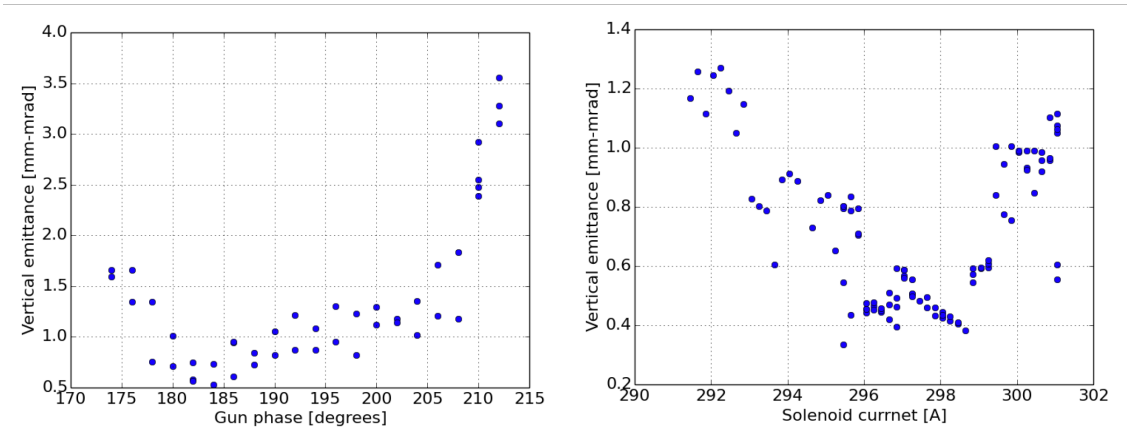


Figure 2.13: Example of measured emittance scan data from FAST for various gun phase and solenoid current settings.

the solenoid and bucking coil currents were kept constant. The virtual cathode image of the laser spot, shown in Figure 2.12, was used to generate the initial beam distributions for `OPAL`.

To have some idea of how much development could be reliably performed in simulation exclusively, we compared measured scan data with `OPAL` results, as shown in Figure 2.14. Later on, we included a simulation of the multi-slit mask and transport to the camera (as opposed to just predicting bulk parameters from a standard simulation). We also compared the agreement between simulated and measured multi-slit images for this case, and examples of the agreement for a couple of reasonable cases are shown in Figure 2.15.

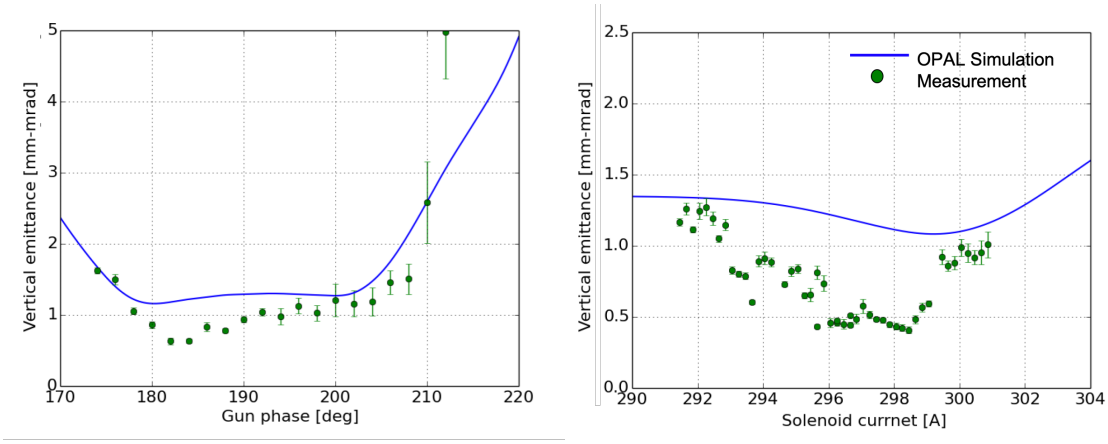


Figure 2.14: Comparison between measured data (green) and `OPAL` simulations (blue) for the vertical emittance versus the gun phase and solenoid strength. Note that the agreement is poor. This is likely due to differences in the as-built accelerator and the simulation. For example, field maps for the solenoid, bucking coil, and rf cavities, along with other sources of error like misalignments, may contribute. The difficulty in accounting for such deviations is part of what makes it appealing to leverage ML and related approaches for model calibration.

For the initial data set, a list of input and output parameters, along with their ranges can be found in Table 2.4. Broad scans of the input parameters around their operating points were conducted, as shown in Figure 2.16. Some subsections of the scans roughly correspond to the measured scan data as well. A subset of the main output parameters of interest is shown in Figure 2.17. Note that for some of the scans, CC1 and CC2 phases were not scanned independently but rather these were adjusted along with the gun phase so that acceleration was kept on-crest. A similar procedure is done on the actual machine, hence our choice to mimic it in the simulation setup.

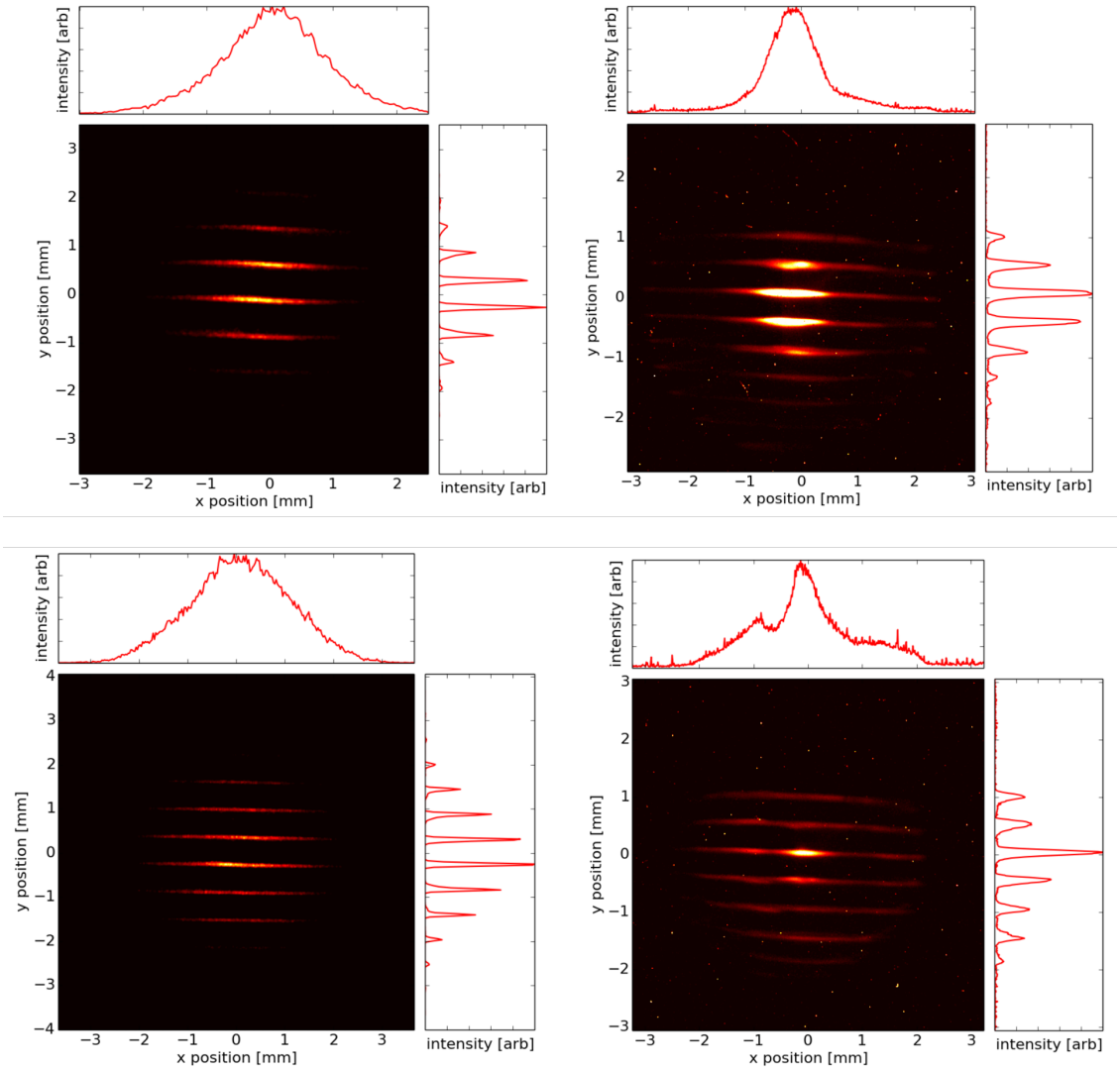


Figure 2.15: Example of comparison between the FAST multi-slit emittance diagnostic simulated in `OPAL` (left) and measured (right) for the same settings and laser profile.

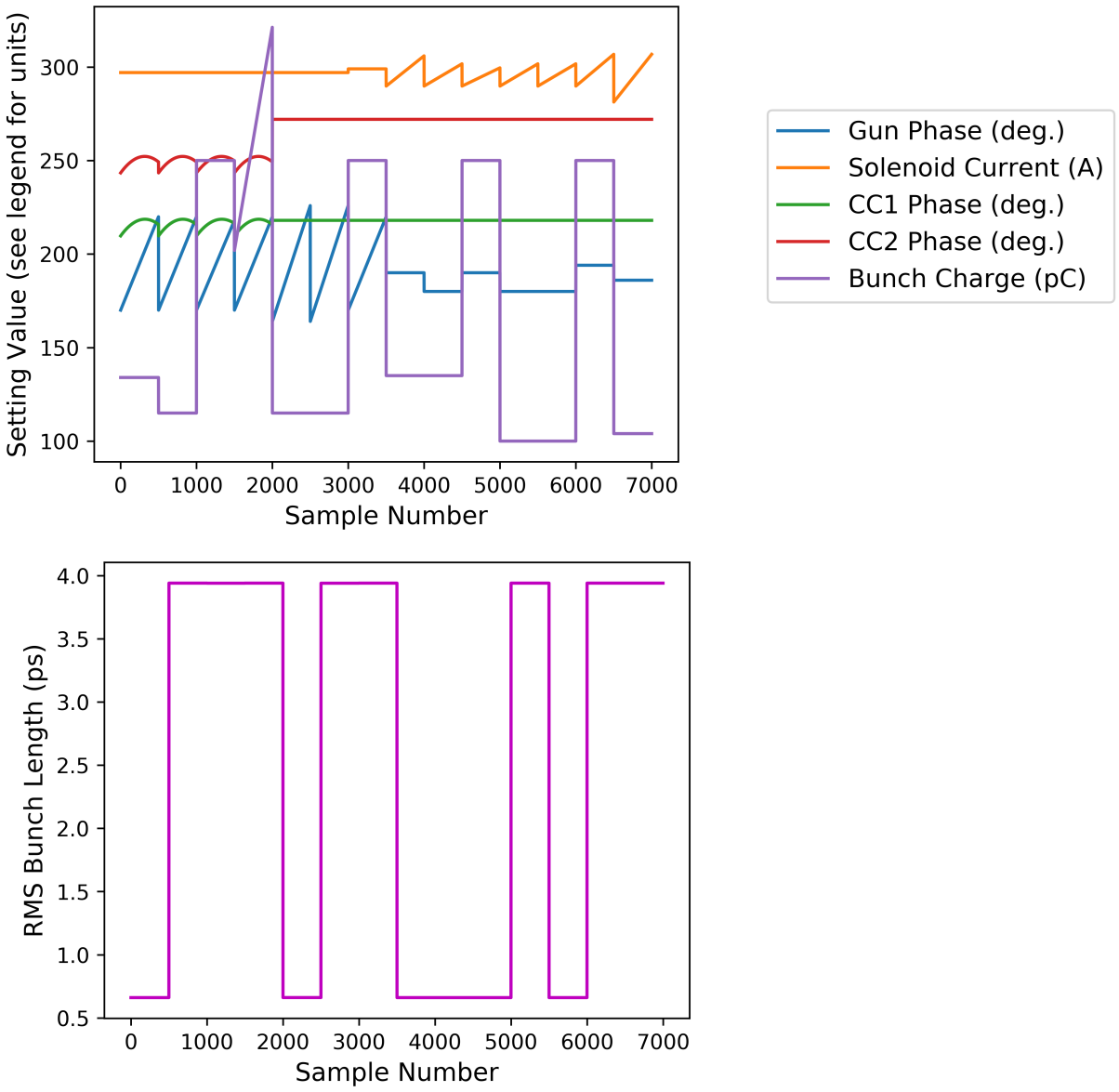


Figure 2.16: Plots of the main input variable scans for the FAST `OPAL` simulations. Each of these constitute large ranges around the nominal settings. The bunch length was also varied between 0.66 and 3.95 ps root mean squared (RMS); this is shown separately in the lower plot for readability. Some subsections of the scans roughly correspond to the measured scan data as well.

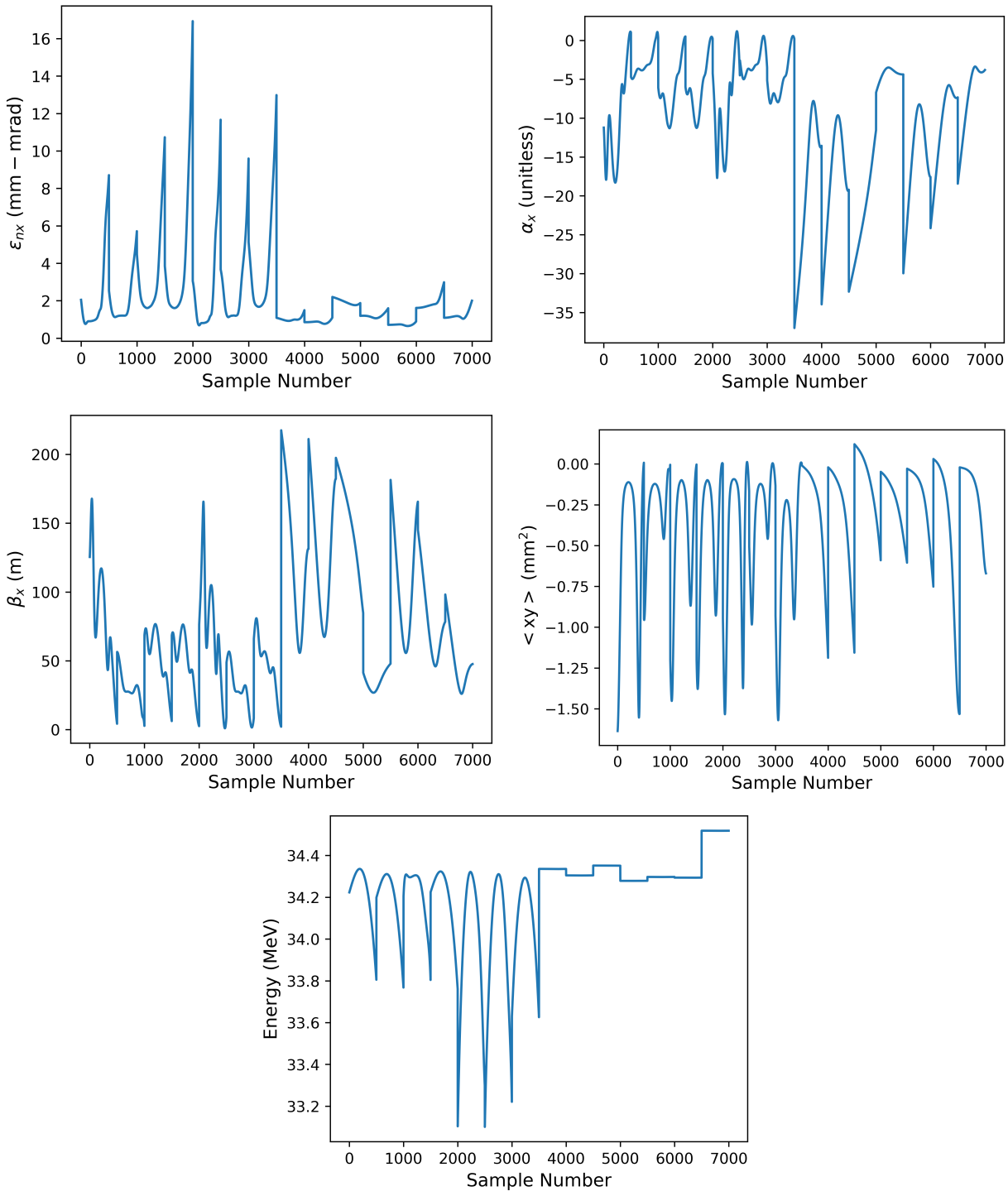


Figure 2.17: Plot of select output beam parameters for the FAST OPAL simulations to show rough ranges. Thirteen other output parameters are not shown for the sake of brevity.

Table 2.4: Simulated Data Ranges for FAST Virtual Diagnostic Scalar Predictions

Variable Setting	Unit	Min	Max
Gun Solenoid Strength	[A]	281	307
Gun RF Phase	[°]	160	226
CC1 Phase	[°]	203	219
CC2 Phase	[°]	237	272
Initial Bunch Charge	[pC]	100	321
Initial RMS Bunch Length	[ps]	0.66	3.94
Initial σ_x	[mm]	0.28	-
Initial σ_y	[mm]	0.35	-
Initial $corr_{xy}$	[unitless]	0.08	-

Beam Parameter	Unit	Min	Max
E	[MeV]	33.1	34.5
ε_{nx}	[m-rad]	6.45e-07	1.71e-05
α_x	[unitless]	-37.0	1.18
β_x	[m]	0.95	217.5
ε_{ny}	[m-rad]	7.21e-07	1.58e-05
α_y	[unitless]	-27.5	1.10
β_y	[m]	0.98	166.85
$< xx >$	[m^2]	1.20e-07	6.45e-06
$< yy >$	[m^2]	1.11e-07	6.40e-06
$< xy >$	[m^2]	-1.64e-06	1.21e-07
$< xx' >$	[m]	-1.37e-07	1.06e-06
$< yy' >$	[m]	-1.14e-07	1.06-06
$< xy' >$	[m]	-1.44e-07	3.98e-08
$< x'y >$	[m]	-1.55e-07	2.73e-08
$< x'x' >$	[unitless]	3.53e-09	4.25e-07
$< y'y' >$	[unitless]	3.16e-09	3.59e-07
$< x'y' >$	[unitless]	-3.32e-08	7.74e-09

2.4.4 Scalar Prediction on Simulation Data

A neural network was trained on the scalar data from the simulation scans described in the previous section. We predicted 18 total output parameters: beam energy E, beam transmission, ε_{nx}, ε_{ny}, α_x, α_y, β_x, β_y, and sigma matrix parameters $< xx >$, $< yy >$, $< xy >$,$< xx' >$,$< yy' >$,$< xy' >$,$< x'y >$, $< x'x' >$, $< y'y' >$,$< x'y' >$. For testing, in addition to the typical random split of the data, we added a segment that includes a range of scan data as well. This helps assess how the neural network performs in interpolating between unseen combinations of input variables. A range of gun phases was left out at a root-mean-squared (RMS) bunch length of 0.66 ps, and another range was left out at a bunch length of 3.94 ps. The two ranges left out of the testing set are shown in Table 2.5 and in Figure 2.18.

The neural network has a fully-connected, feed-forward architecture with 15-20-20-20-20-20-20 nodes in each hidden layer and hyperbolic tangent activation functions. As usual, the output layer has linear activation functions. For training, the Adam optimization algorithm was used, and a learning rate decay factor of 0.8 was applied if predictions on the validation set did not improve for more than 50 epochs. A batch size of 10 was used, as this was found to provide improved performance on the interpolation task compared with larger batch sizes (e.g. 100 - 200 samples). The mean squared error between predicted and simulated values was used as the cost function.

The neural network model performs well in predicting output parameters on the test set, even when interpolating in the held-out ranges of the scans. Predictions on the test set are shown as a function of gun phase in Figure 2.19 and as a function of solenoid current in Figure 2.20. Figure 2.21 shows a closeup of part of the testing data where ranges of the input data (as opposed to a random sample) were held out for testing. The neural network model captures the nonlinear behavior represented in the computationally expensive `OPAL` simulation, and thus it could be used as a way of bringing an online model to FAST. However, for a virtual diagnostic prediction that includes the measured data (rather than just providing estimates from simulation), we need to address directly how well the predictions match measured data. This is the subject of the next sub-section.

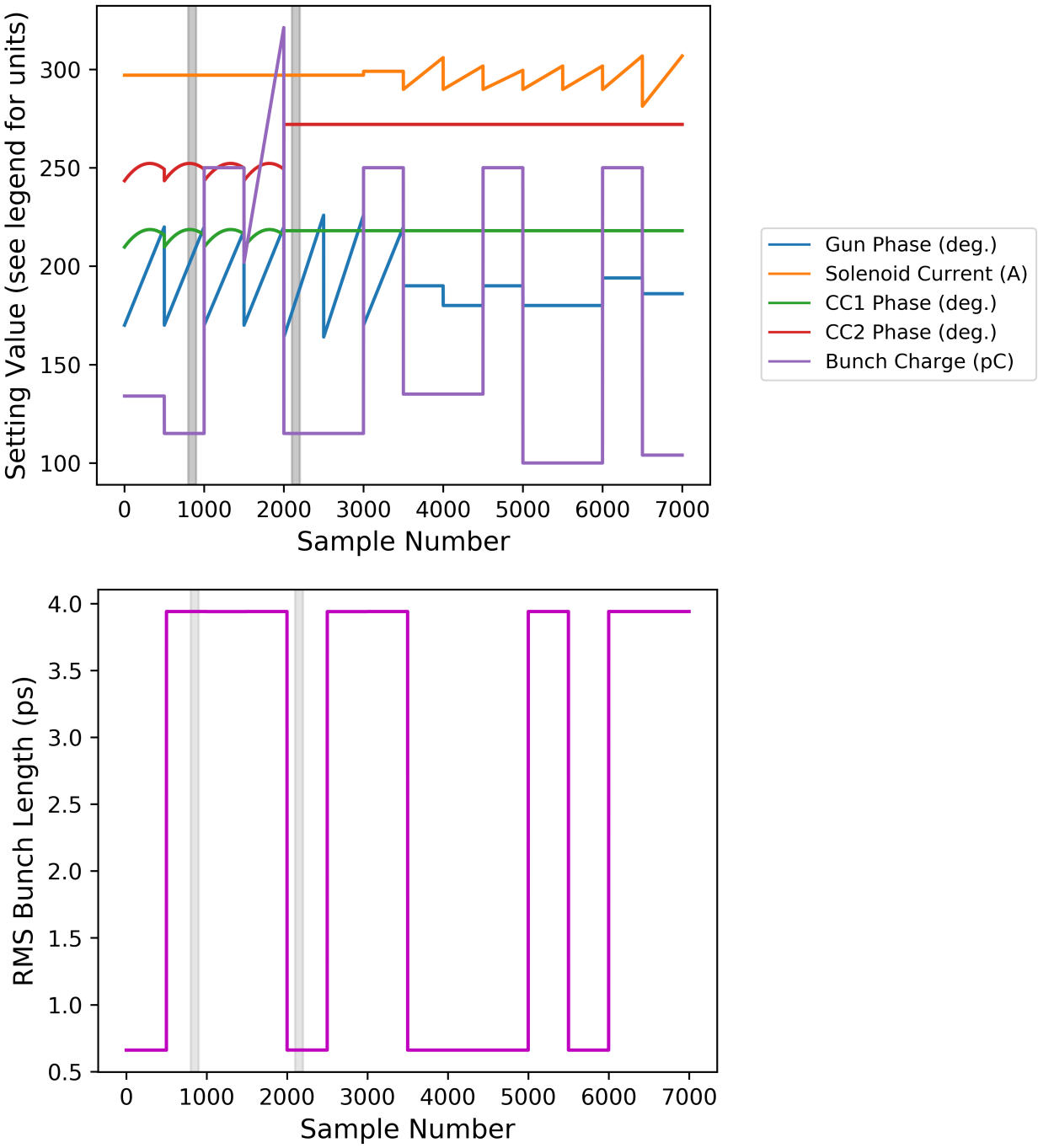


Figure 2.18: Inputs for simulation scan data. Grey markings denote which scan data was held out for testing in addition to the random test split. Pulling out distinct parts of the scan data helps assess the neural network's ability to interpolate when new combinations of input parameters are used.

Table 2.5: Two data ranges left out of training and validation. These were added to the test set for the scalar predictions.

Test Segment 1 Variable Setting	Unit	Range
Gun Solenoid Strength	[A]	297
Gun RF Phase	[°]	176 - 189
CC1 Phase	[°]	218
CC2 Phase	[°]	272
Initial Bunch Charge	[pC]	115
Initial RMS Bunch Length	[ps]	0.66

Test Segment 2 Variable Setting	Unit	Range
Gun Solenoid Strength	[A]	297
Gun RF Phase	[°]	200 - 210
CC1 Phase	[°]	218
CC2 Phase	[°]	251.6 - 252.1
Initial Bunch Charge	[pC]	115
Initial RMS Bunch Length	[ps]	3.94

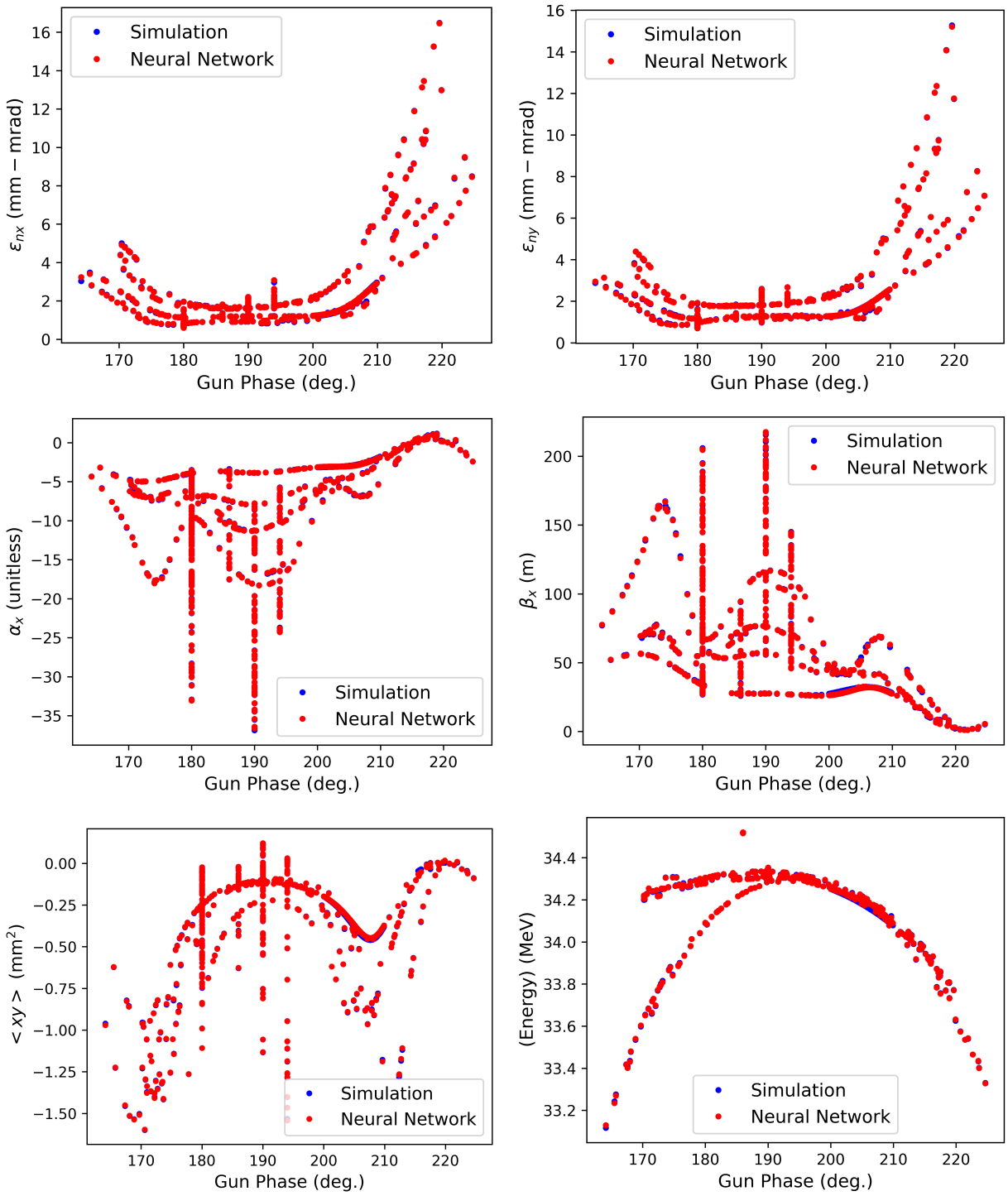


Figure 2.19: Scalar predictions on the test set from the neural network trained on simulation data for a subset of the output parameters (18 outputs in total), as a function of variation in gun phase. There are multiple output values (seen here as vertical lines and different levels of responses) for a given input value because this shows cases where *other* inputs are also simultaneously varied. Note that in this case the test set includes not just a random split of the data but a held out range of setting combinations (further highlighting the interpolation capabilities for this neural network model).

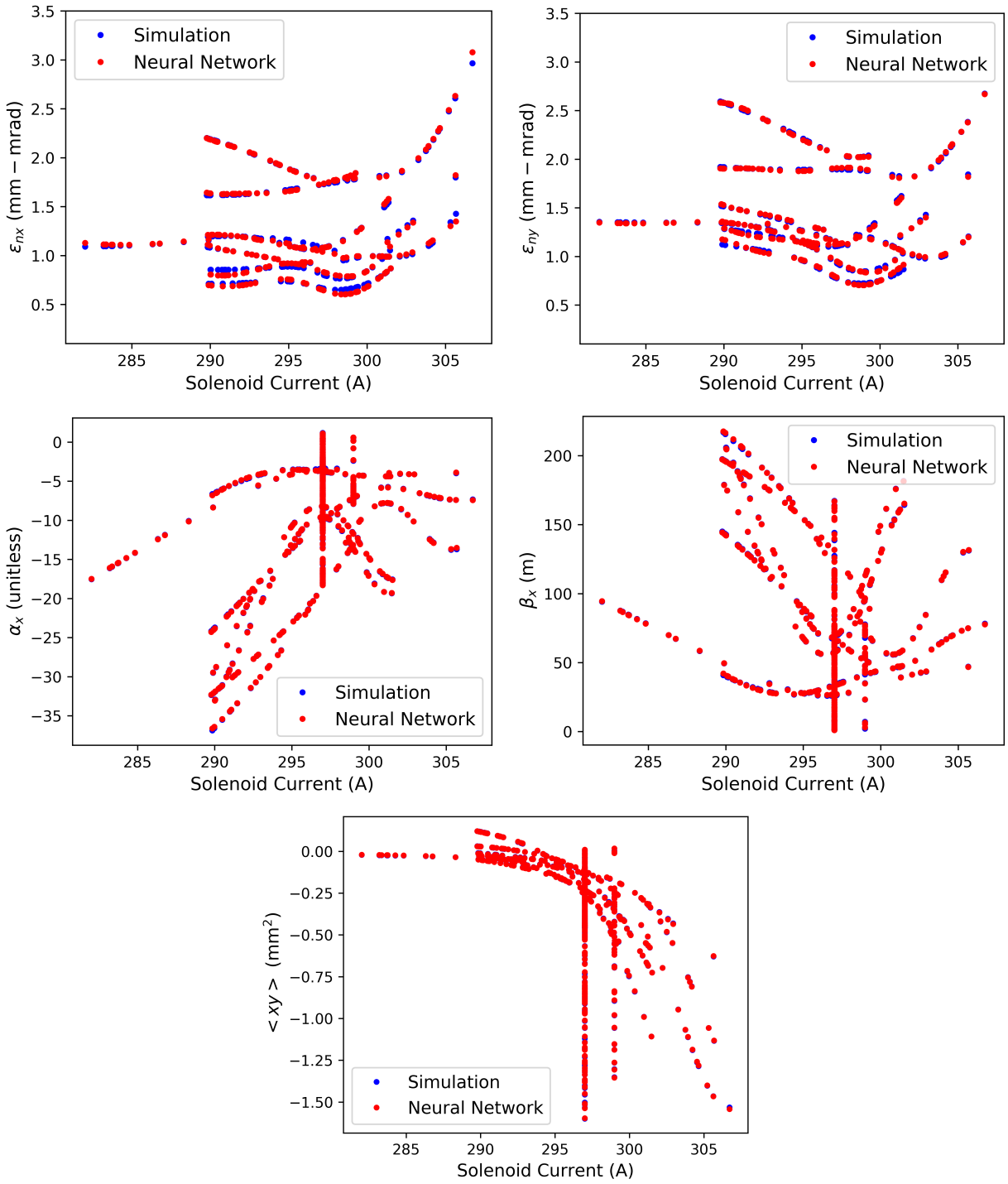


Figure 2.20: Scalar predictions on the test set from neural network trained on simulation data for a subset of the output parameters (18 outputs in total), as a function of variation in solenoid current. There are multiple output values (seen here as vertical lines and different levels of responses) for a given input value because this shows cases where *other* inputs are also simultaneously varied. Note that in this case the test set includes not just a random split of the data but a held out range of setting combinations (further highlighting the interpolation capabilities for this neural network model).

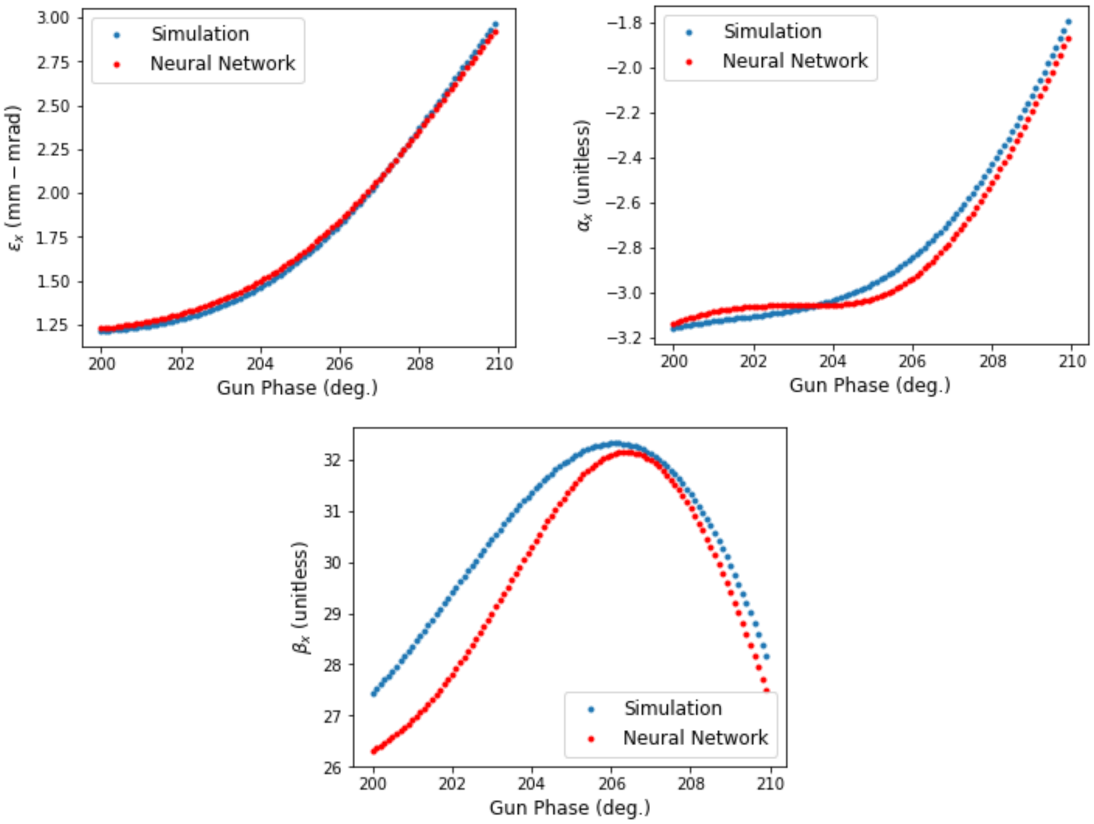


Figure 2.21: Closer example of prediction performance of neural network scalar model in interpolation for one of the ranges not seen during training (as opposed to the randomly selected portions of the test set). These correspond to the rightmost grey range of scan data in Figure 2.18, and this is the more challenging of the two held-out scan ranges. The data shown are subsets of the same data that is shown in Figure 2.19.

2.4.5 Updating with Measured Data (Transfer Learning)

In some cases only limited measured data can be obtained. For example, machine time may be in short supply, the measurement itself may be slow, or one may want to vary more parameters for training a model than would be feasible to scan on the real machine. In these cases, it would be appealing to first train a neural network model on broad sets of simulation data and then fine tune it with a smaller measured data set. This approach is an example of transfer learning [29,30] between the simulation and the machine. While transfer learning has been used for domain adaptation in large convolutional neural networks (for example in image classification tasks), at the time of these studies (2016-2017) it had not yet been attempted for transferring between simulation and measured data sets for accelerators.

For FAST, the simulation in `OPAL` does not perfectly capture the observed machine behavior. This is likely due to asymmetries and other deviations from the ideal design that are unaccounted for in the field maps of the cavities, solenoid, and bucking coil used in the simulation, as well as misalignments between elements, etc. Thus, when we train the neural network only on simulation data, we observe large errors in the prediction when varying the solenoid setting. However, when we adjust the photoinjector phase, the agreement between simulation and measurements (and thus between the pre-trained neural network and the measurements) is reasonable. To test whether we can update the neural network with just a portion of the measured data, we conducted a limited update of the neural network weights by training with a subset of the measured solenoid scan data only. In this case, we do not freeze any of the weights as is often done in transfer learning. We instead only allow a comparatively small learning rate (0.000001 in standard stochastic gradient descent). We also monitor both the prediction error on the unseen (i.e. unmeasured) ranges of simulation data and on the validation set of the measured solenoid scan data during training. We stopped the training when the prediction error on the unseen ranges of the simulation data begins to degrade. In this way, the model is not over-fitted to the new measured data and retains its predictive capability for regions of parameter space that are not measured.

After this procedure, the predictive performance is greatly improved, as shown in Figure 2.22. Furthermore, the performance for the phase scan data (which was not included in the update) is only slightly perturbed. While one could train on measured data alone, recording the scan data requires substantial machine time (in this case, approximately three 8-hour shifts were used to obtain 273 data points). In many cases, allocating this amount of time for such studies may not be possible. Pre-training with simulation in such cases may allow one to introduce enough training data to learn rough machine behavior, thus reducing the amount of measured data needed to complete the training (i.e. to reach an equivalent prediction accuracy).

Note that in Figure 2.22 we are showing a spline interpolation of the 273 measured data points and corresponding neural network predictions. We could manually do a series of 1-D interpolations reliably in this case because we did 1-D scans of the inputs; however, typically one would not have this luxury if using general archive data. Even in this case, the neural network interpolation was far more convenient than manual 1-D spline interpolations.

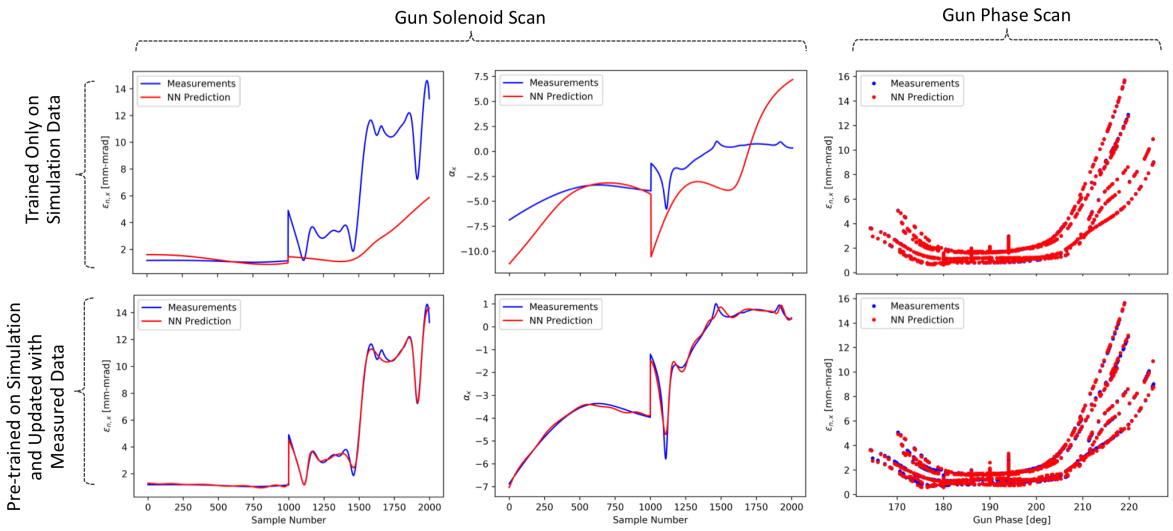


Figure 2.22: Example of updating the pre-trained neural network model with measured data. The top row shows the neural network predictions on the measured data when it has been trained only on simulation data. In this case the discrepancy represents the difference between the physics simulation and the measured data. The bottom row shows the neural network predictions after the pre-trained model has been updated with a small amount of retraining on measured data (transfer learning). Only data from solenoid scans were used in the retraining, and yet the performance on the phase scan data is not perturbed (ensuring that we can update the neural network with only partial scan data and still preserve some of the predictive performance of the original model). Note here that we are showing predicted and "measured" points from a spline interpolation of the 273 data points obtained on the machine.

2.4.6 Direct Prediction of Simulated Image Diagnostic Output

To obtain bulk parameters representing the phase space, one relies on the multi-slit fits to be accurate. However, these fits are not accurate for non-Gaussian beams, do not capture x-y coupling, and can degrade substantially when the quality of the image is reduced (e.g. when the measured image is noisy or has artifacts). In addition, when the multi-slit fitting procedure changes (e.g. as improvements are made to the peak selection algorithm, etc.), the neural network needs to be re-trained if it is using the scalar values from the slit fits.

By predicting the image output from the multi-slit measurement instead of the scalar fit values, we can reproduce the machine behavior more directly and are agnostic to the specific multi-slit fitting method chosen. We tested this approach in simulation, and we observe good agreement between the simulated images and the images produced by the neural network.

To include the multi-slit measurement, an additional step of masking and transporting the beam was added to the simulation, and additional simulation scans were conducted. These are shown in Figure 2.23. In addition, for testing, we again reserved parts of the ranges of the input scan data (i.e. unique series of setting combinations) rather than randomly sampling from the dataset alone. The scan regions reserved for testing are shown in Figure 2.24.

The neural network has 6 hidden layers with 50-80-100-200-400-500 nodes in each respective layer. Hyperbolic tangent activation functions are used in each hidden layer, and a linear output layer activation function is used. Because the dataset is small (603 images, each 620 by 480 pixels), we use Gaussian noise layers between each hidden layer to add noise with a standard deviation of 0.1 at each forward pass through the neural network. This helps to prevent over-fitting. In addition, after separating the data into training, validation and testing sets, the training data itself was augmented three-fold by carefully adding Gaussian random noise to the images at a level that was at max 10% of the minimum pixel deviation between adjacent settings in the scans. The neural network was implemented in `keras` [113] with `Theano` [109] as the backend. For the cost function, the sum of the absolute errors between the predicted and true pixel values in the images was used. In addition, the sum of the absolute errors between the predicted and true

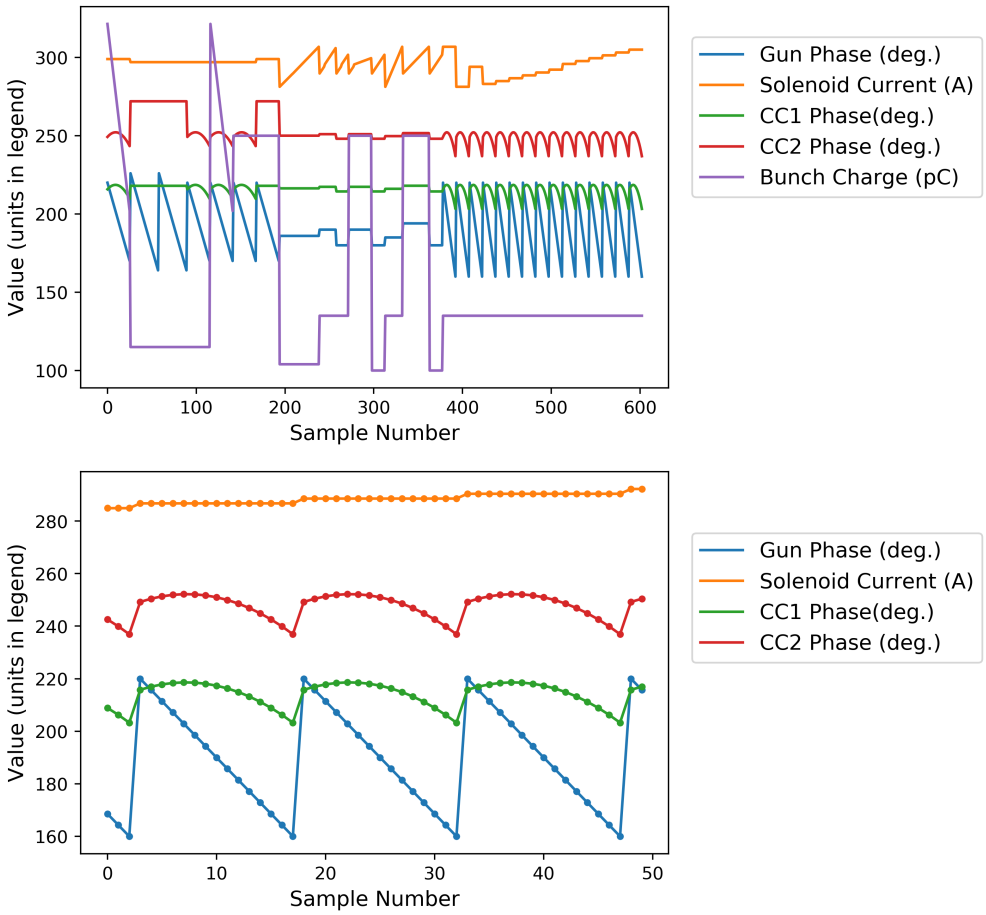


Figure 2.23: `OPAL` simulation scans conducted to include the multi-slit measurement directly (e.g. image predictions). The full range of varied parameters (top) and a close up of a scan section (bottom) are shown. These correspond to broad setting ranges around nominal values for FAST.

horizontal and vertical profiles were used as an additional penalty in the cost function. The neural network was trained with the Adam optimization algorithm until convergence. The training set consists of 515 examples, the test set consists of 43 examples, and the validation set consists of 45 samples.

Examples of the resultant image predictions on the test set are shown in Figure 2.25, and the corresponding slit profiles are shown in Figure 2.26. More examples of both are shown in Figure 2.27. Figure 2.28 shows the same thing but for the held out *ranges* of parameters specifically. The agreement is generally very good. For examples with poor agreement, these are in some cases for outliers where the beam would not be suitable for a multi-slit emittance measurement anyway

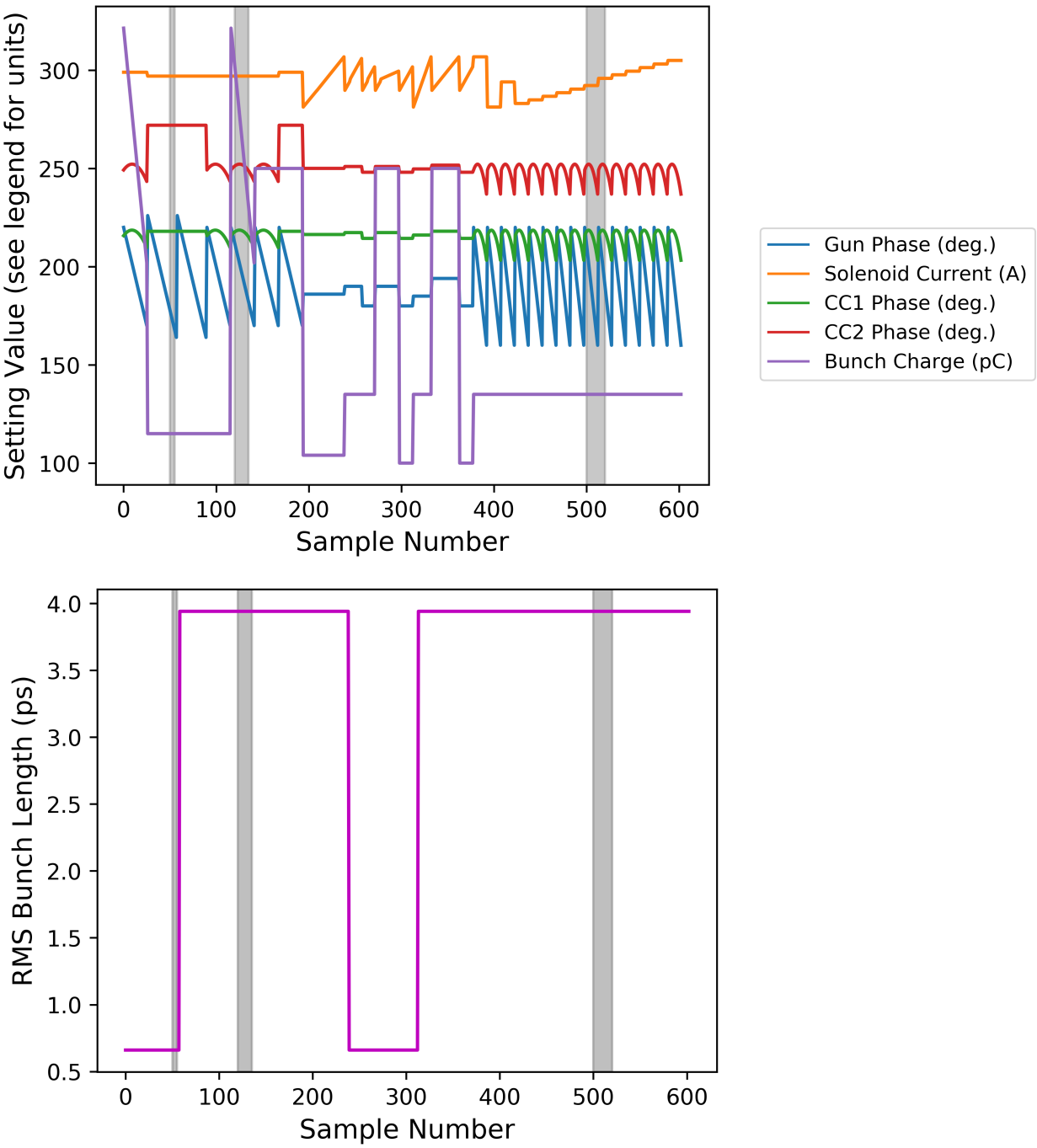


Figure 2.24: `OPAL` simulation scans showing regions of input scans left out of training and used in the test set (shown in grey) to examine interpolation of the neural network.

(e.g. high-divergence beams where the slit patterns become washed out). In this case, it is sufficient that the neural network also predicts a very poor beam.

The mean absolute error on the image prediction test sets were 0.009 counts (relative to a max normalized number of counts per pixel of 1) and the standard deviation was 0.02 counts. The mean absolute error on slit profiles was 0.87 counts and the standard deviation was 0.46 counts. In terms of percentage of max counts (i.e. max signal) per profile, this corresponds to 1.5 % error and 1.0% standard deviation. This is a reasonable level of accuracy, particularly for a proof-of-concept study with a small amount of data and a relatively large range of input settings.

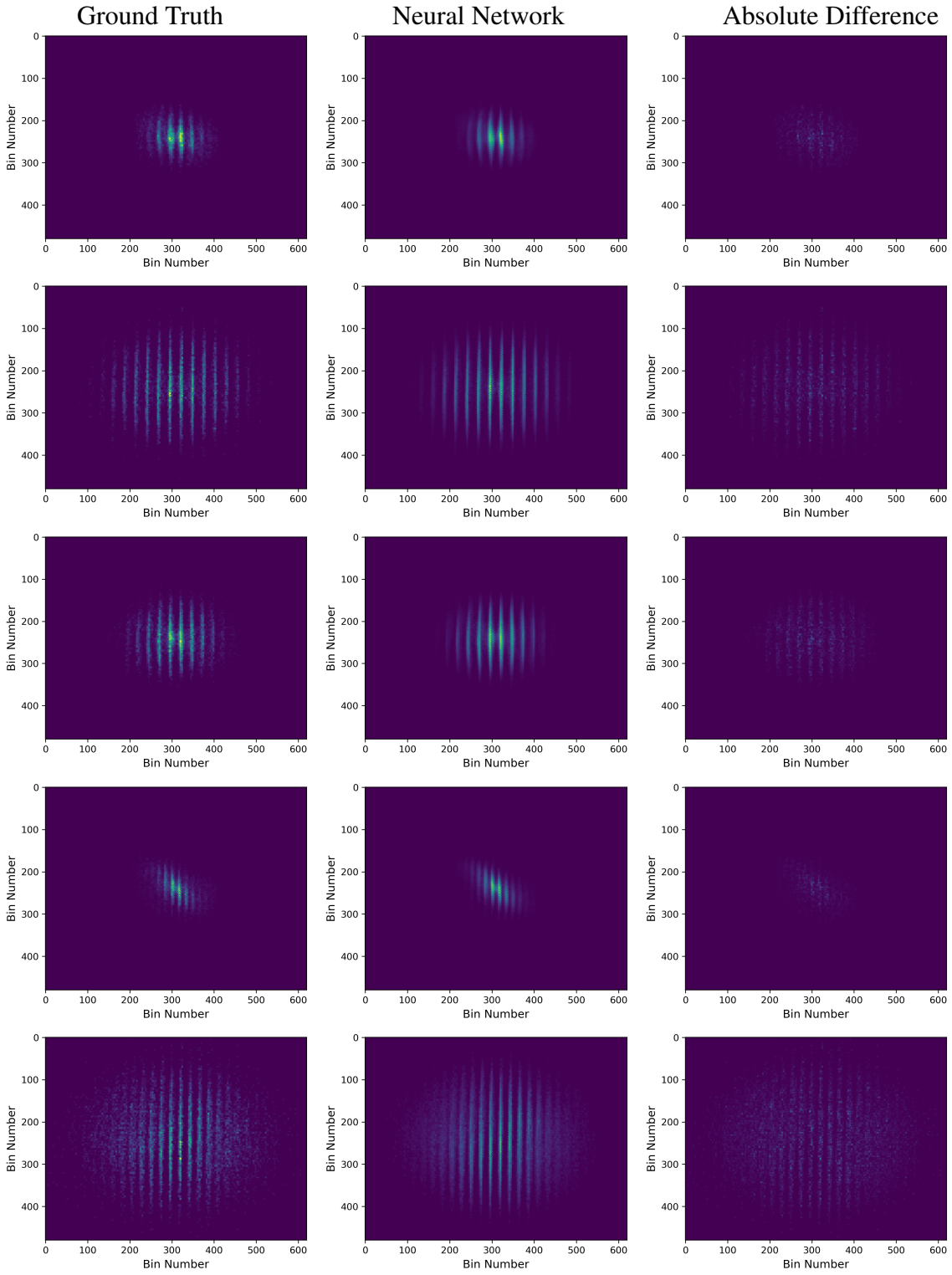


Figure 2.25: Comparison between the simulated multi-slit diagnostic images ("Ground Truth") and neural network predictions ("Neural Network") for some representative cases in the test set.

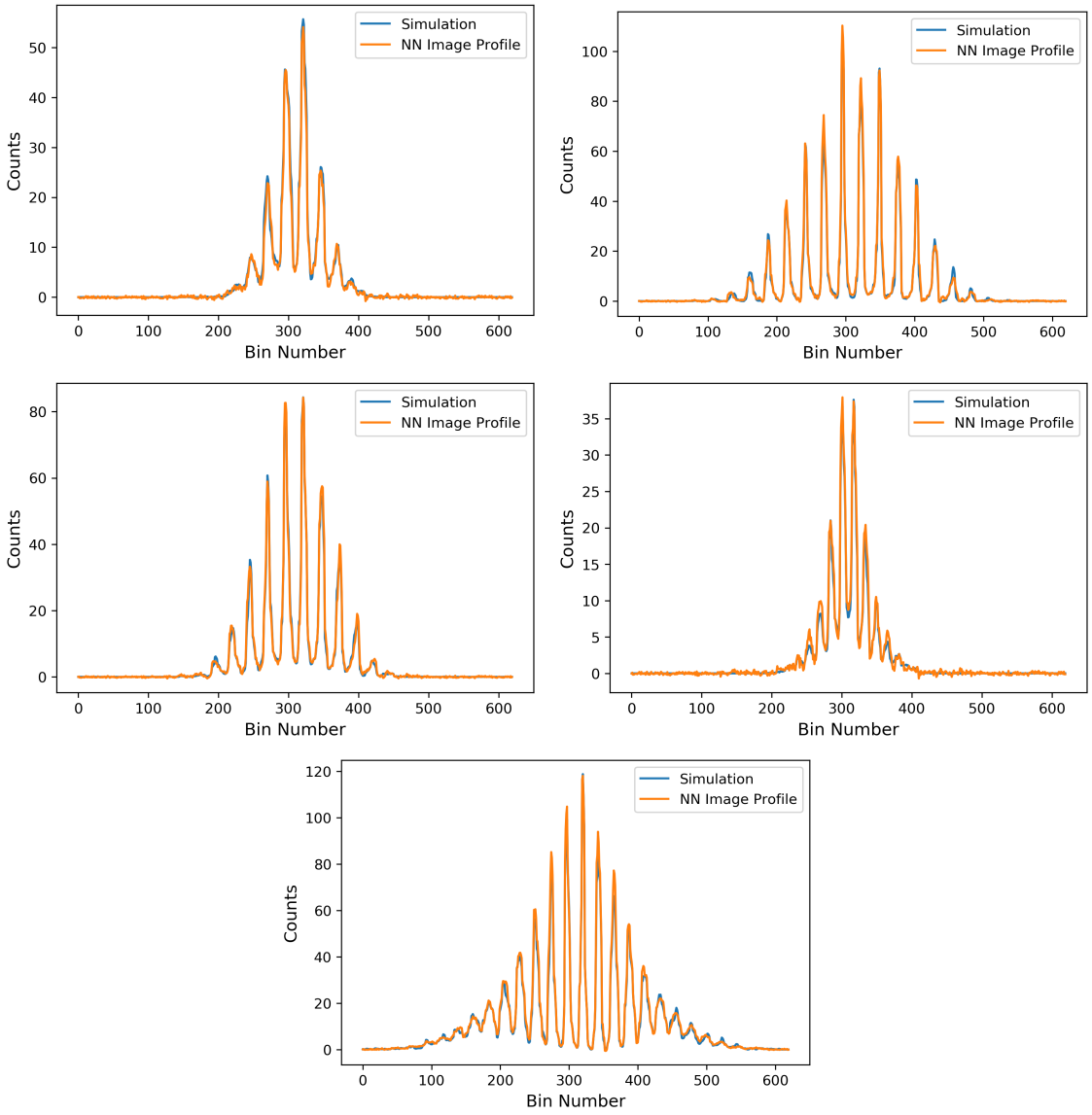


Figure 2.26: Comparison between the simulated multi-slit measurement images and profiles and neural network predictions for the same test set cases as shown in Figure 2.25.

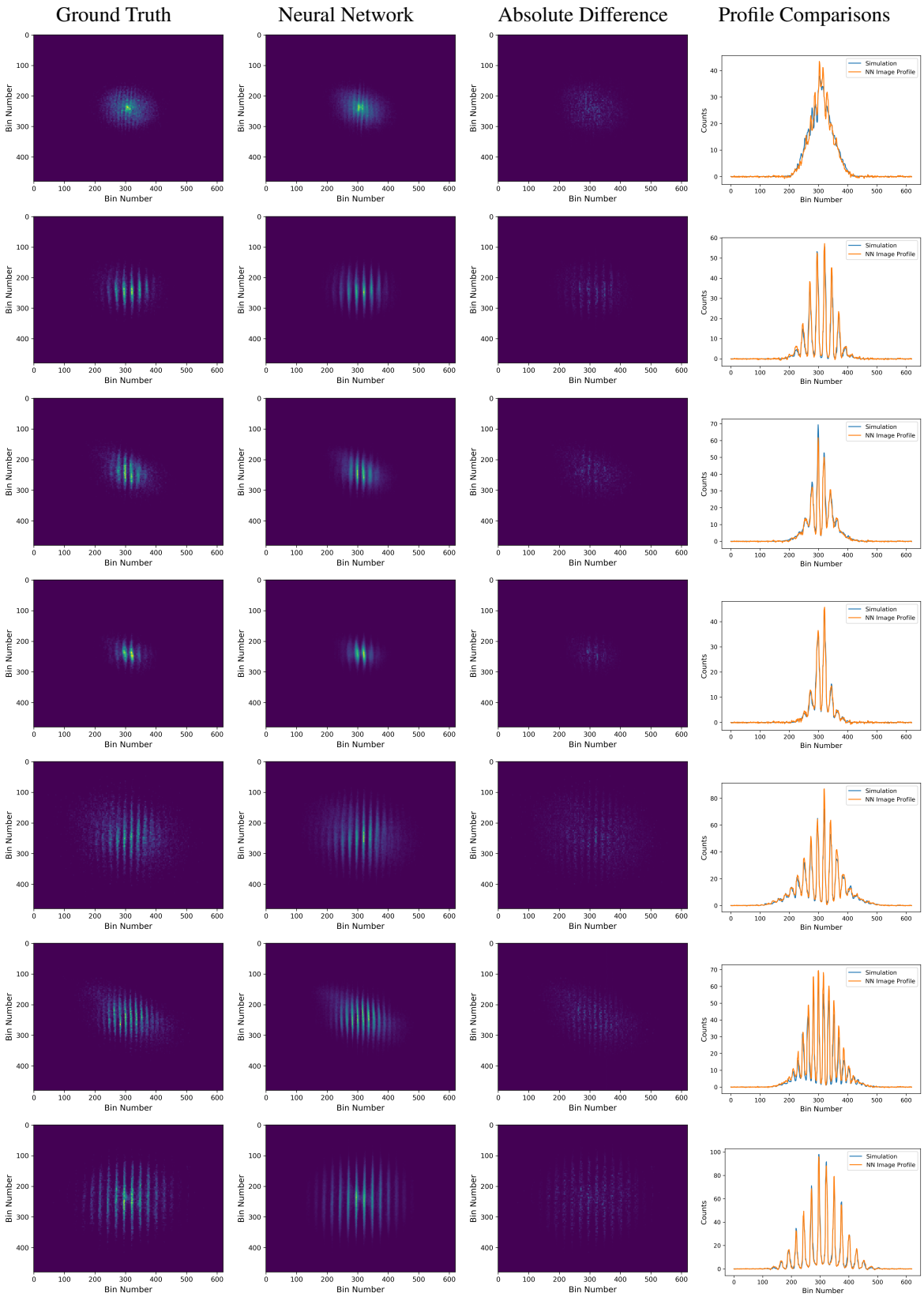
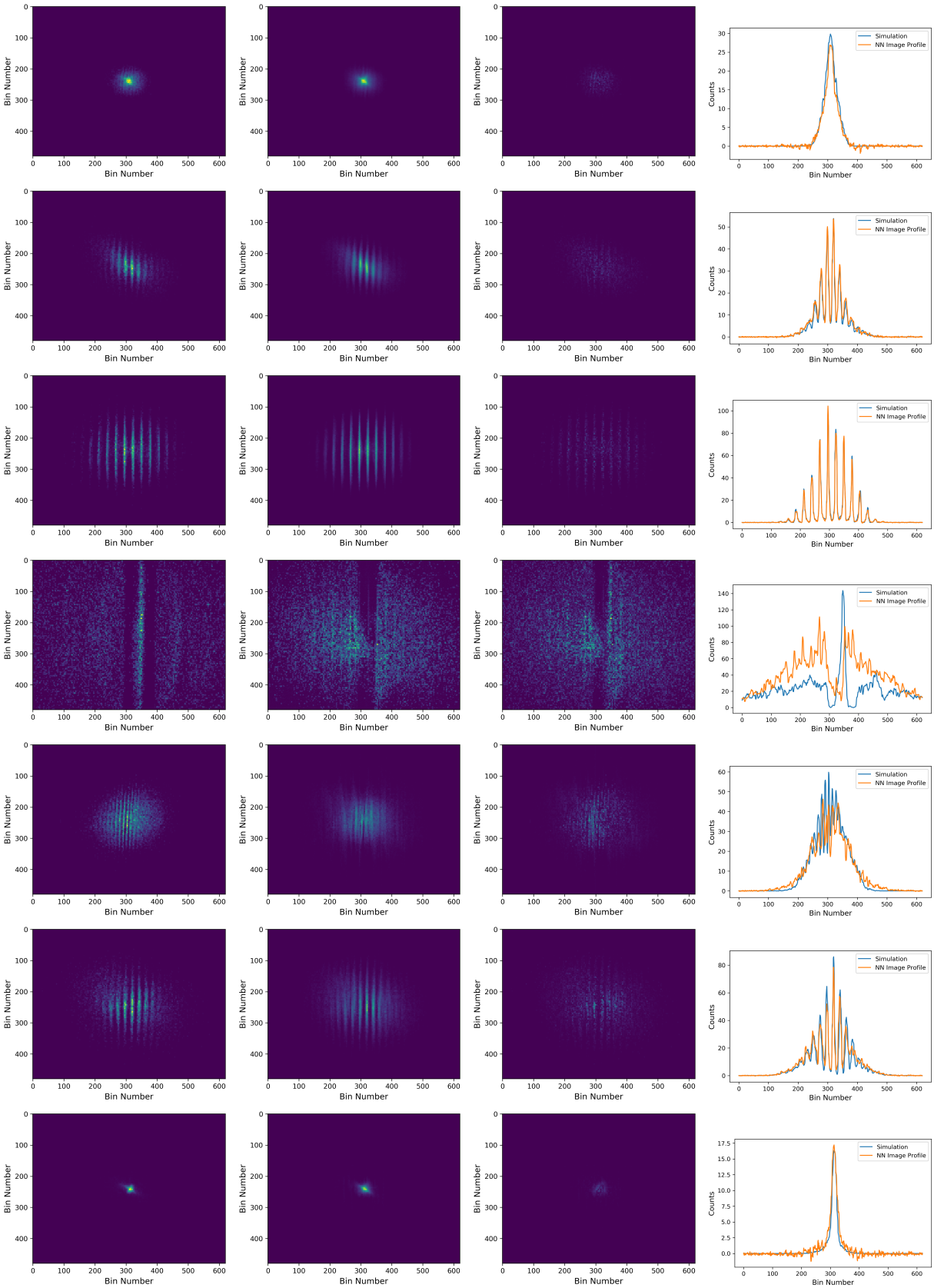


Figure 2.27: Comparisons between ground truth and neural network predictions on the test set for the multi-slit diagnostic images (continued on next page).


Bin Number
Counts
Simulation
NN Image Profile

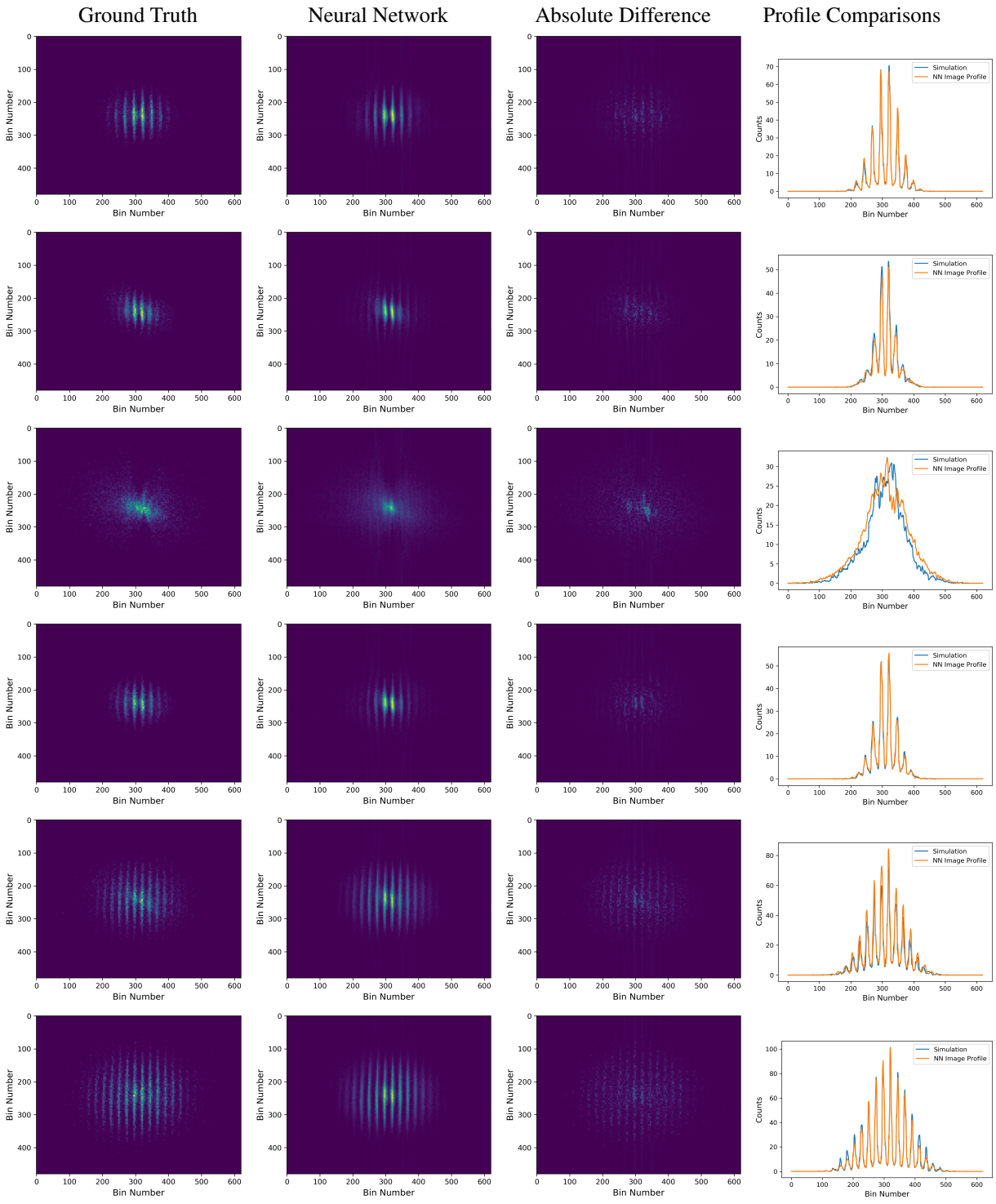


Figure 2.28: Comparisons between ground truth and neural network predictions on the test set for the multi-slit diagnostic images, specifically for the held out *ranges* of the scanned input parameters (continued on next page). In other words, this is looking specifically at how well the neural network interpolates for unseen combinations of input parameters (to a greater degree than is achieved with a randomly-selected test set).

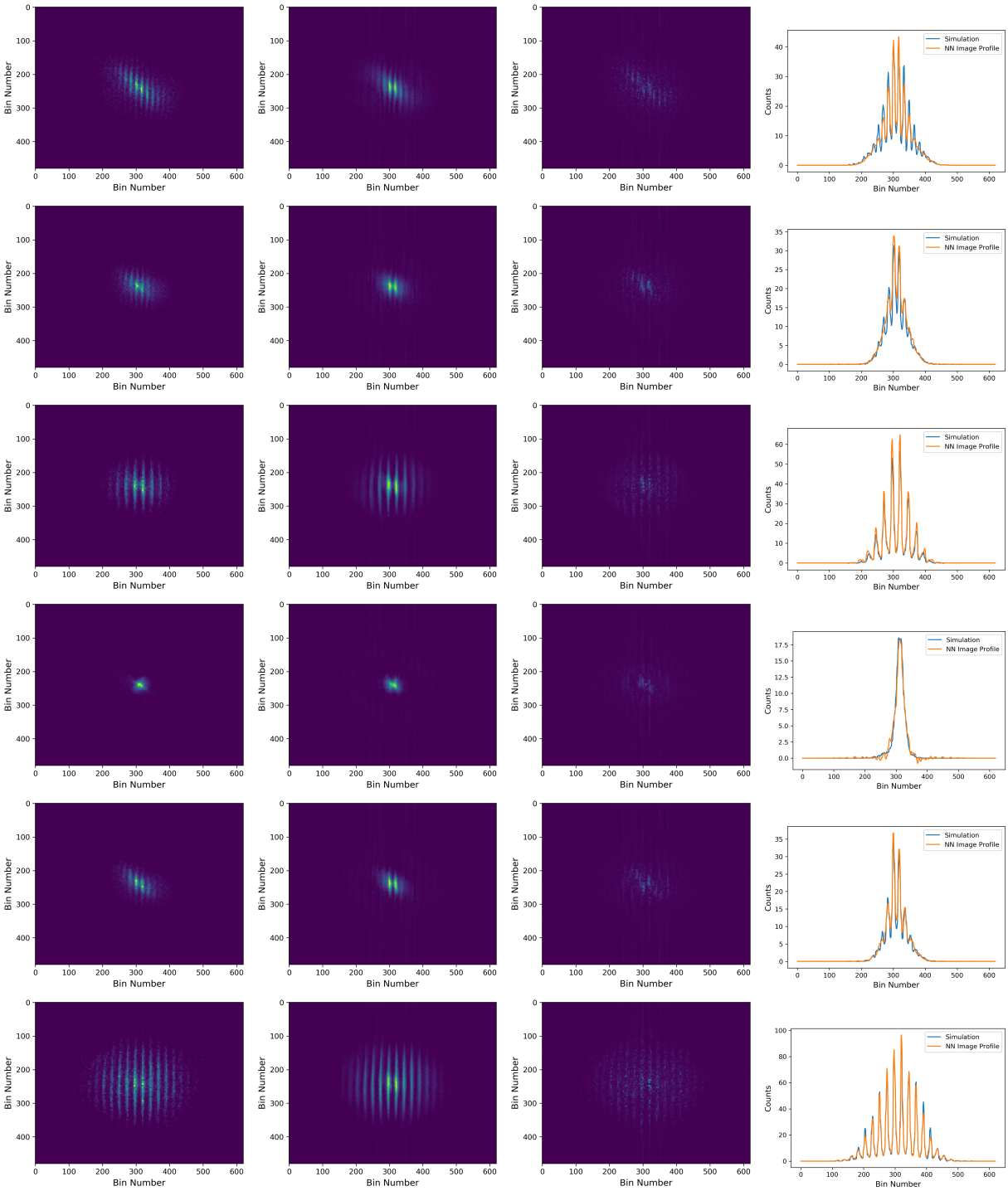

Bin Number
Counts
Simulation
NN Image Profile

Table 2.6: Data ranges left out of scans for the virtual diagnostic test sets

Test Segment 1 Variable Setting	Unit	Min	Max
Gun Solenoid Strength	[A]	297	-
Gun RF Phase	[°]	170	178
CC1 Phase	[°]	218	-
CC2 Phase	[°]	272	-
Initial Bunch Charge	[pC]	115	-
Initial RMS Bunch Length	[ps]	0.66	-

Test Segment 2 Variable Setting	Unit	Min	Max
Gun Solenoid Strength	[A]	288	290
Gun RF Phase	[°]	160	220
CC1 Phase	[°]	203	219
CC2 Phase	[°]	237	252
Initial Bunch Charge	[pC]	135	-
Initial RMS Bunch Length	[ps]	3.94	-

Test Segment 3 Variable Setting	Unit	Min	Max
Gun Solenoid Strength	[A]	299	301
Gun RF Phase	[°]	160	220
CC1 Phase	[°]	203	219
CC2 Phase	[°]	237	252
Initial Bunch Charge	[pC]	135	-
Initial RMS Bunch Length	[ps]	3.94	-

Table 2.7: Simulated Data Ranges for FAST Virtual Diagnostic Image Predictions

Beam Parameter	Unit	Min	Max
ε_{nx}	[m-rad]	6.5e-07	2.6e-05
ε_{nx}	[m-rad]	6.5e-07	6.1e-04
α_x	[unitless]	-37.0	16.8
β_x	[m]	1.0	217.5
$< xx >$	[m^2]	1.3e-07	1.1e-05
$< xx' >$	[m]	-1.5e-05	1.1e-06
$< x'x' >$	[unitless]	1.4-09	2.3e-05
ε_{ny}	[m-rad]	7.3e-07	2.6e-05
ε_{ny}	[m-rad]	7.24176277e-07	6.5e-04
α_y	[unitless]	-27.5	15.2
β_y	[m]	1.2	166.9
$< yy >$	[m^2]	1.3e-07	1.1e-05
$< yy' >$	[m]	-1.4e-05	1.1e-06
$< y'y' >$	[unitless]	1.5e-09	2.5e-05
$< xy >$	[m^2]	-1.7e-06	1.2e-06
$< xy' >$	[m]	-1.5e-07	1.4e-06
$< xy' >$	[m]	-4.9e-06	1.3e-06
$< x'y' >$	[unitless]	-1.9e-06	4.1e-06
E	[MeV]	4.5	34.6
transmission	[%]	100	-

Variable Setting	Unit	Min	Max
Gun Solenoid Strength	[A]	281	307
Gun RF Phase	[°]	160	226
CC1 Phase	[°]	203	219
CC2 Phase	[°]	237	272
Initial Bunch Charge	[pC]	100	321
Initial Bunch Length	[ps]	0.66	3.94
Initial σ_x	[mm]	0.28	-
Initial σ_y	[mm]	0.35	-
Initial $corr_{xy}$	[mm]	0.08	-

2.4.7 Conclusions and Future Work

This work has demonstrated that a neural network can predict simulated multi-slit emittance measurement images and bulk beam parameters for a wide range of input settings of the injector at FAST. This study was an early example of an ML-based virtual diagnostic. In addition, we showed that the prediction performance remains acceptable when interpolating between unseen ranges of input scan data. We also demonstrated a procedure for conducting transfer learning between the simulation and measured data, which could potentially reduce the burden on machine time when trying to build training data sets.

Collectively, these are promising steps toward creating an online model and virtual diagnostic for emittance prediction at FAST. However, to be put to reliable use on the machine, the performance of the model with respect to machine drift over time (e.g. due to known phase calibration drift, uncontrolled changes in the laser profile, and changes in the cathode quantum efficiency over time) needs to be examined. For example, our measured data set only spans three shifts over the course of two days. Coupled with this, reliable retraining strategies for keeping the model continuously updated would need to be examined. In addition, the transfer learning result between the simulation and the machine required careful weighting of the various data sets used, and more reliable methods of doing this in an automated fashion should be investigated. Finally, for virtual diagnostics to be put to use in practice, including some measure of neural network uncertainty in the predictions will be required. This would give some indication about whether a given prediction can likely be trusted and could be used to indicate when retraining is required. Although this is an important consideration for virtual diagnostic applications in general, it is even more acute for cases like the multi-slit emittance measurements where it will likely be difficult to take re-training data very often due to how slow the measurement process is. Methods such as model ensembling or Bayesian neural networks [191] may be suitable for generating such uncertainty estimates, but reliable uncertainty estimation for neural networks is still an open area of research within the ML community.

2.5 Notes on Simulation Speedup

In each of the cases above, we have also created a fast-executing representation of the simulation that is suitable for online modeling and experiment planning (e.g. by replacing the injector simulation with the neural network model for studies where it would be appropriate to do so). For example, one evaluation of `PARMELA` with 2-D space charge for the FAST linac up through CC2 took approximately 20 minutes on one Intel Core i7. However, the neural network model evaluates in under 1 ms on the same processor. Similarly, the `OPAL` simulation with 3-D space charge from the injector up to the X107 diagnostic screen took approximately 15 minutes on one Intel Core i7 for each forward pass, and once again the neural network model evaluates in under 1 ms. This is around a million times speedup in execution and opens up a new avenue for using particle accelerator simulations that include expensive-to-compute nonlinear collective effects during machine operation, without the need for on-site HPC resources for on-demand first principles simulations.

This of course comes with the caveat that it is only useful for the parameters and ranges varied during training; however, it is worth noting that even when using simplified or HPC-accelerated physics models, extensive model calibration often must be done, and not all models are valid for the full range of input parameters anyway. The results here at least are an encouraging indicator that we can add neural network-based surrogate models to the available options for online modeling and faster offline experiment planning.

Chapter 3

System Tuning and Rapid Switching Between Setups

3.1 Neural Network Controller for Fast Energy Switching in a Compact FEL

Free Electron Laser (FEL) facilities support a broad range of scientific endeavors. For example, FELs in the soft to hard X-ray regime are elucidating protein structures [192], investigating natural processes such as photosynthesis [193], and understanding the origin of different material properties [194]. In support of this, these machines must accommodate requests for a variety of electron beam parameters in order to supply their users with the appropriate photon beam characteristics for any given experiment. In the laboratory environment, this usually requires skilled human operators to tune the machine until the desired system output is achieved, thus reducing the amount of useful experimental time and increasing the overall operating cost of the facility relative to its scientific output. In scientific user facilities where there is a high demand on beam time, both the tuning speed and quality are important factors.

In principle, a neural network control policy that is trained on a broad range of operating states could be used to quickly switch between these requests without substantial need for human intervention. Additionally, this policy could be updated concurrently to machine operation as new states are visited or as drift occurs in the system. It also provides a relatively compact way of storing the information (as compared to, e.g., a database of previous settings and measured output).

To assess the basic feasibility of this concept, a simulation study was conducted for a compact THz FEL design based on the Twente/Eindhoven University FEL (TEU-FEL) [195] (see Figure 3.2). This is an appealing system for this study because it has a relatively small number of components, yet it exhibits non-trivial beam dynamics. In this case, the aim is to adjust the photoinjector and beamline settings to achieve specific electron beam parameters at the entrance of the undulator. Operationally, switching to a new operating configuration would involve specifying the

target beam energy and then choosing the appropriate injector settings and beamline settings such that the electron beam is properly matched into the undulator (see Figure 3.1).

We show that a neural network policy can be trained to switch rapidly between requested beam energies, including for energies that have not been seen directly in training. The neural network can do this within 10% accuracy in one iteration. In comparison, optimization with Nelder-Mead Simplex takes 80-200 iterations to converge for the cases examined.

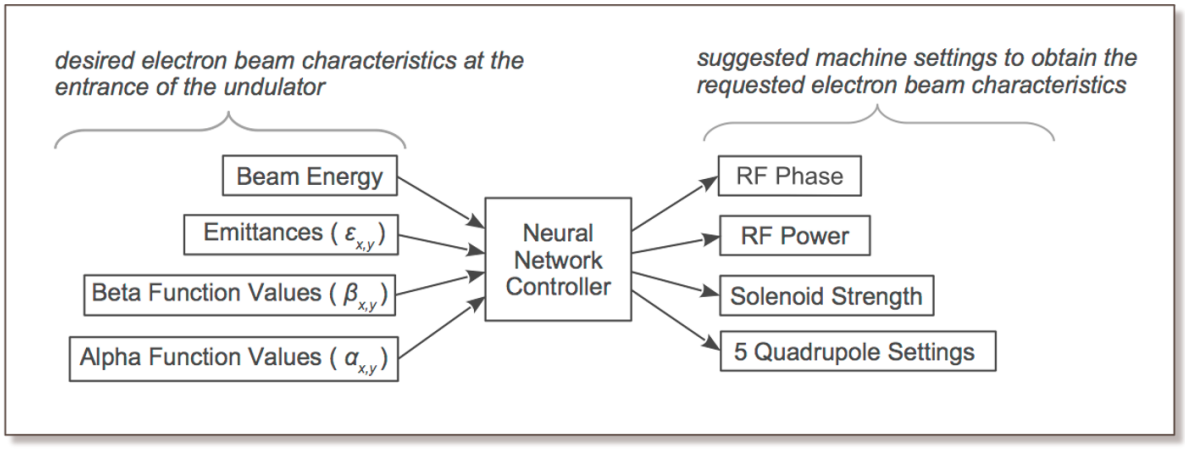


Figure 3.1: General scheme for the neural network control policy. Output parameters are specified at the entrance of the undulator. For a specified beam energy and target Twiss parameters, an initial guess at the machine settings is provided. The neural network's map from the beam request to the initial guess at settings is updated over time based on the resulting performance in achieving the target beam parameters.

3.1.1 The TEU-FEL System Description and First Principles Simulation

The TEU-FEL is designed to produce light with a wavelength that is tunable between 200 μm and 800 μm. It consists of a 5.5-cell, 1.3-GHz photocathode radio frequency (RF) photoinjector, a beam transport section with focusing quadrupoles, and a fixed-gap, permanent magnet undulator. The undulator strength parameter K is given by $K = \frac{\lambda_u eB}{2\pi m_e c}$, where λ_u is the period of the magnet placement, e is the electron charge, B is the magnetic field, m_e is the electron rest mass, and c is the speed of light. In the TEU-FEL undulator, K is equal to 1. The photoinjector is an RF structure that accelerates the initial electron beam. A bucking coil and solenoid provide appropriate focusing in the photoinjector. More details on this machine can be found in [195–198], and more details on the basics of FEL physics can be found in [12]. Figure 3.2 shows the relevant components for this study.

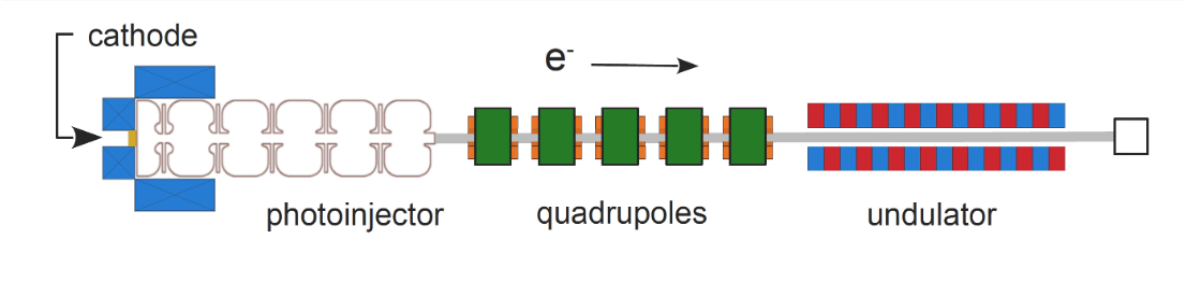


Figure 3.2: Layout of the accelerator system showing the 5.5-cell RF photoinjector with its bucking coil and solenoid, the beamline with quadrupoles, the undulator, and the beam dump. There also are steering coils, but these are excluded, as the simulated electron beam is generated on-axis.

While this machine is comparatively simple in terms of the number of components and interacting systems, the beam dynamics exhibit some subtleties that if not addressed properly will result in decreased performance of the FEL. In particular, the beam evolution throughout the machine is nonlinear. Because the bunch charge can be as large as 5 nC and the beam energy is low (3–6 MeV), space-charge effects will be significant throughout the system. This complicates the task of focusing the beam, which in turn complicates the task of matching the beam into the undulator. A solenoid is typically used to compensate for space-charge effects by matching the beam to

the invariant envelope [199]. This can be done in long (e.g. > 10-cell) accelerating structures or with injectors that have a short (e.g. 1.5-cell) photoinjector followed by a drift and a secondary accelerating stage. The TEU-FEL photoinjector, with 5.5 cells, makes it more difficult to perform adequate compensation of space-charge-induced emittance growth. Additionally, due to physical space constraints there is not enough room to allow the emittance to fully damp before the matching section into the undulator, and because the beam is low-energy, space-charge forces will continue to impact the emittance along the beamline. This further complicates the task of focusing the beam even after initial space-charge compensation is performed by the solenoid.

A simulation model of the TEU-FEL injector and beamline was constructed using `Superfish` [183] for the RF fields and `PARMELA` [182] for the beam dynamics. Because the coupling between the cells of the cavity is not axially symmetric, a full simulation of the photoinjector geometry could not be conducted using `Superfish`. Therefore, the geometry was approximated by simulating each of the individual cell types and splicing them together in `PARMELA` to create the proper field geometry. When compared with measurements of the field geometry this approach performs reasonably well [197]. The solenoid and bucking coil assembly were modeled in `PANDRIA` [183] using the nominal current settings in the solenoid. The bucking coil was then scaled to cancel the magnetic field on the cathode. The combined field map was then scaled in `PARMELA` in order to tune the space-charge compensation. The `PARMELA` simulations were performed using 5000 macro-particles and a 0.1° phase integration step. This was determined to be well within the stable region for reasonable estimation of bulk parameters [197].

3.1.2 Study Setup

For the initial study the aim was to have the neural network provide suggested settings such that specific $\alpha_{x,y}$ and $\beta_{x,y}$ are achieved at the entrance of the undulator for a given electron beam energy. In order to achieve proper matching into the undulator, the beam is focused to a waist at the entrance, which means the alpha parameter $\alpha_{x,y} = 0$ rad. The good magnetic field region for equal focusing in both planes requires the beam size to be less than 4 mm [196]. Based on this, the same

target beta function ($\beta_{x,y}$) value at the entrance of the undulator was set for each electron beam energy: $\beta_{x,y} = 0.106$ m/rad. This corresponds to a beam size that is within the optimal field region of the undulator for all energies. The approach taken was similar to how one might approach it for a real machine:

1. The initial training data is made to mimic how a machine archive might look (noisy data with local tuning around discrete operating points).

2. A neural network model is trained on this data. This creates a surrogate for the simulation that captures the major relevant behavior of the machine and can execute quickly to facilitate controller training.

3. A controller is then trained via interaction with the learned model. As the controller enters new regions of parameter space, the model is occasionally updated with new simulation data points.

4. For the above, a section of energy range is left out. The controller is then assessed on how well it can provide suggested settings for this left-out range (in our case, this assessment is done via interaction with the simulation after training). This is to assess the controller's ability to provide reasonable suggested settings in previously-unseen regions of parameter space.

This approach is shown schematically in Figure 3.3 and Figure 3.4. We wanted the distribution of the initial training data samples to mimic what one might find in measured data from different e-beam configurations in an FEL facility (i.e. noisy data with tuning around roughly optimal settings for different beam setups). As such, the training set consisted of the output from each iteration of an optimization routine (in this case, Nelder-Mead Simplex [100]) to find optimal current settings for each of the five quadrupoles ($Q1$–$Q5$) for 12 different beam energy scenarios between 3.1 and 6.2 MeV, with noise added. This dataset is used both for training the initial model and the initial control policy. An example of the quadrupole set points from the data is shown in Figure 3.5.

The simulation itself is slow-to-execute (appx. 5 minutes per forward pass, or around 26 days of continuous running on 1 core of a 2.9 GHz Intel Core i7 processor to generate the samples used in this study). Thus, the time required to produce the initial training set was reduced by optimizing the photoinjector and the beamline separately (i.e. photoinjector settings were coarsely optimized for each beam energy, given a rough guess at optimal quadrupole settings from previous TEU-FEL studies, and then the quadrupole settings were fine-tuned). This could impact the final solution quality observed in the initial training data; however, this is acceptable in this case because the intent of this dataset was to produce an initial "warm start" for the controller training. In addition, separate optimization of individual accelerator subsystems is consistent with standard practice.

Table 3.1: Data ranges for TEU-FEL. Note that beam parameters are only calculated in cases where transmission is above 90%

Beam Parameter	Unit	Min	Max
α_x	[rad]	-2.14	2.11
α_y	[rad]	-5.76	1.45
β_x	[m/rad]	0.06	1.86
β_y	[m/rad]	0.07	3.65
E	[MeV]	3.1	6.2
Transmission	[%]	0	100
Variable Setting	**Unit**	**Min**	**Max**
$Q1$	[T/m]	-0.98	0.83
$Q2$	[T/m]	0.65	1.98
$Q3$	[T/m]	-2.24	-1.07
$Q4$	[T/m]	0.89	2.26
$Q5$	[T/m]	-1.90	-0.23
Gun Solenoid Strength	[norm.]	0.67	1.05
Gun RF Phase	[°]	10.3	21.4
Gun RF Power	[norm.]	0.57	1.15

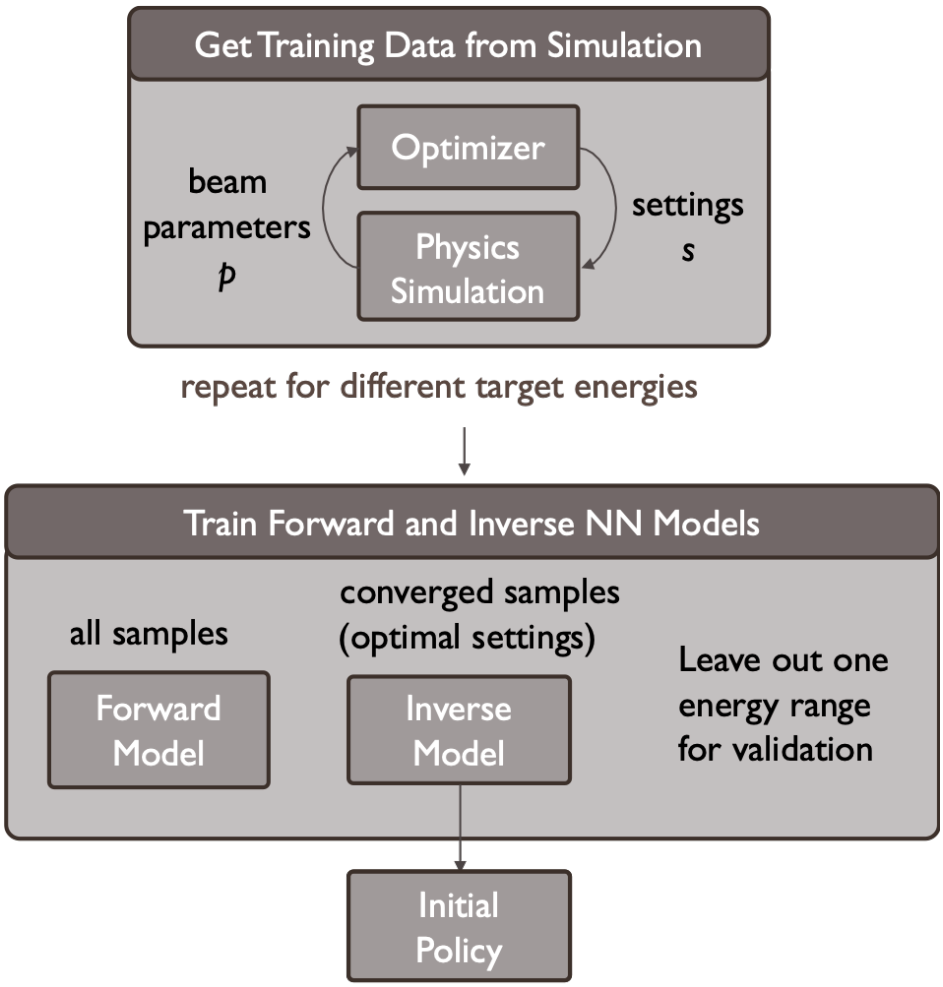


Figure 3.3: Schematic of generating training data and conducting initial pre-training of the control policy. The aim was to produce an initial training set that mimicked what one might find in a machine archive, use this to create a forward model that the controller would interact with as part of training, and then train an initial control policy using the same samples (in this case, this corresponds to an inverse model of the system).

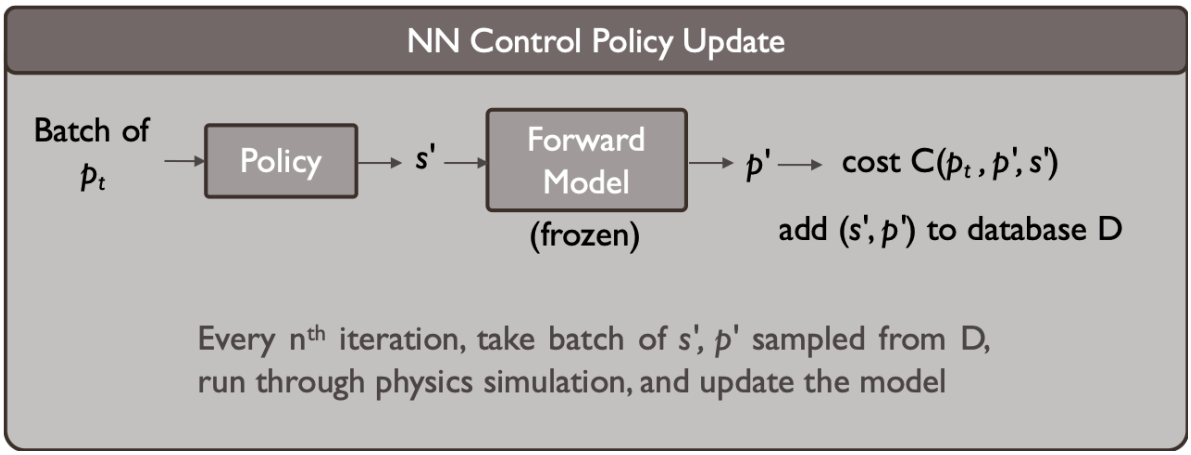


Figure 3.4: Schematic for neural network control policy updates. A batch of target beam parameters p_t are sent to the neural network control policy, which generates a predicted set of optimal settings s'. These are then sent to the neural network forward system model, whose weights have been frozen, to generate a predicted set of beam parameters p'. The weights of the policy are then updated according to the cost function, which includes the error between p_t and p' along with setting penalties. The s' and the corresponding p' from the model are saved, and a random batch of these is occasionally sent to the physics simulation to update the forward model.

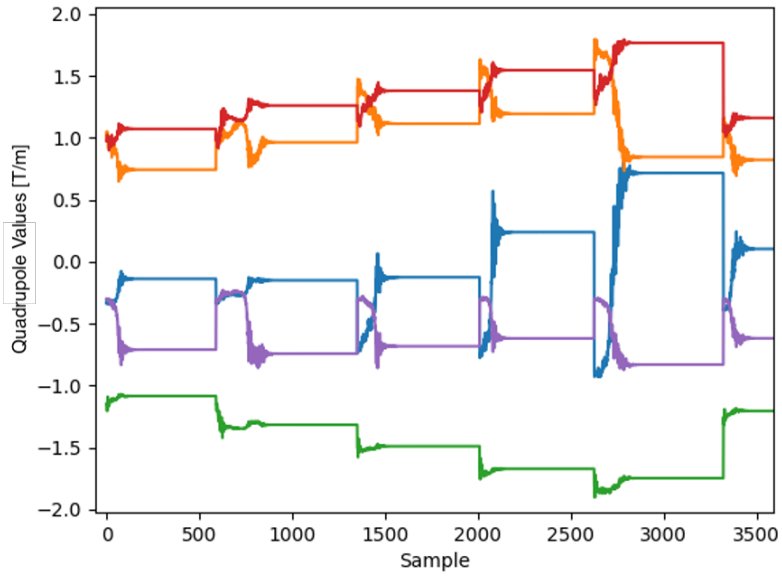


Figure 3.5: Training data example for initial training set generated according to Figure 3.3, showing variation in the five quadrupole settings.

3.1.3 Neural Network Model

The surrogate model was trained to predict the Twiss parameters, beam energy, emittance, and transmission at the entrance of the undulator, given the RF power, the RF phase, the solenoid strength, and the five quadrupole settings. Optimization data for the 5.7-MeV electron beam was excluded from the training set and used as the test set (310 samples total). This was chosen to assess the ability of the model to interpolate to unseen regions of parameter space, which is highly relevant to the task of giving suggested settings for a given beam request. The ranges of beam output parameters and input parameters are shown in Table 3.1, and an example of the initial training data is shown in Figure 3.5. The training data consisted of 7195 samples in total.

The neural network architecture consists of four hidden layers containing 50, 50, 30, and 30 nodes, respectively. Each node in the hidden layers uses a hyperbolic tangent activation function and a dropout [88] probability of 10%. The weights and biases of the network were trained using the AdaMax [75] optimization algorithm, along with subsequent fine-tuning using Møller's scaled conjugate gradient algorithm [200]. The cost was the mean squared error over the predictions for each batch of 200 samples. The network was constructed and trained using a combination of `lasagne` [112] and `Theano` [109].

Because the calculated beam parameters become inaccurate when only a small number of particles remains in the transmitted bunch, we have two output layers: one for transmission, which is trained on the full data set, and one for the beam parameters, which is only trained when the transmission is above 90%.

The performance of the model in terms of mean absolute error (MAE) and standard deviation (STD) is shown in Table 3.2. A representative plot from the test set is shown in Figure 3.6. From this we conclude that the model can interpolate in unseen regions of parameter space reasonably well, and that we have created a reasonably accurate, fast-executing surrogate model as a stand-in for the physics-based simulation to help speed up training (i.e. by not relying on the slow-to-execute physics simulation alone for controller training). The speedup in this case is over a factor of 10^6.

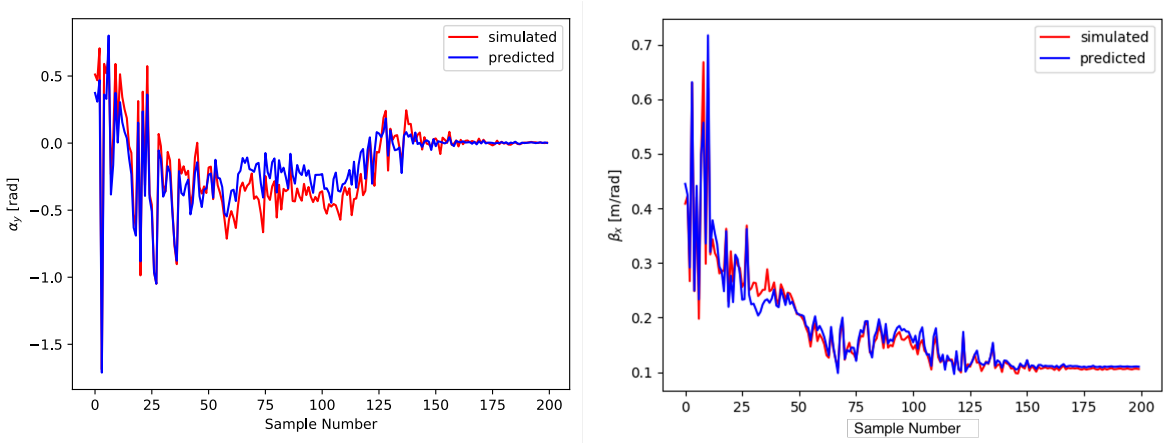


Figure 3.6: Neural network model predictions and simulated values for α_y and β_x on the test set, for samples generated via Simplex optimization. The test set was an optimization run for beam energy of 5.7 MeV, with the energy range for that run excluded from the training set. In other words, this is a measure of how well the model can interpolate in unseen regions of parameter space. This shows that we have created a reasonably accurate, fast-executing surrogate model as a stand-in for the physics-based simulation to help speed up training (i.e. by not relying on the slow-to-execute physics simulation alone for controller training). The speedup in this case is over a factor of 10^6. Note that in this case, we are also actually showing data that was generated by running Simplex optimization on the model for one of the energy settings, so we also see that in this case it takes Simplex 200 iterations to fully converge. In contrast, the neural network policy (trained on the neural network model shown) immediately makes an accurate guess of appropriate settings to reach a specified set of beam parameters, as shown in Table 3.2

Table 3.2: Prediction accuracy of TEU-FEL neural network model.

Variable	Units	Train MAE	Test MAE
α_x	[rad]	0.018	0.067
α_y	[rad]	0.022	0.070
β_x	[m/rad]	0.004	0.008
β_y	[m/rad]	0.005	0.012

3.1.4 Neural Network Controller

The controller selects an initial guess at settings given a desired beam energy. The controller was first trained as an inverse model following the same procedure as the forward model. The controller network is then trained more extensively via interaction with the model network, the output from which was used to directly calculate the cost (i.e. by back-propagating the cost through the model network). As such, this is a deterministic control policy that is trained first with supervised learning (on the data from simulation) and then with reinforcement learning (by interacting with the learned model, which itself is occasionally updated with new simulation samples).

Random desired beam energy values between 3.1 and 6.2 MeV at the entrance of the undulator were specified, along with a target set of α and β values, as inputs to the controller. The cost function included the mean squared error between the target parameters and those predicted by the model. In addition, losses of full transmission and larger quadrupole settings were penalized proportionally. To narrow the scope of the initial problem, deviation from the nominal optimal photoinjector values were also given a penalty. Thus, the quadrupoles are allowed to vary much more freely than the photoinjector parameters for this initial study.

Requested beam energy values between 4.8 and 5.2 MeV were excluded from training and used exclusively for testing. The controller network architecture consists of three hidden layers, with 30, 30, 20, and 20 nodes, respectively. As before, a hyperbolic tangent activation function and a dropout probability of 10% was used for each hidden node, and the batch size was 200 samples. AdaMax was used to adjust the weights.

Finally, interaction directly with the physics-based simulation was used to verify the performance of the controller. Given random requested energy values within 3.1–6.2 MeV, Table 3.3 shows the performance in reaching the desired Twiss parameters ($\alpha_{x,y} = 0$ rad, $\beta_{x,y} = 0.106$ m/rad) in **one iteration**. This shows that for a given energy, the controller will immediately reach the desired beam size to within about 10% and the beam will be close to a waist, requiring minimal further tuning to reach the target values. The maximum absolute errors for the energy range seen in the training data set were 0.063 rad, 0.023 m/rad, 0.067 rad, and 0.041 m/rad for α_x, β_x, α_y, β_y,

respectively. The maximum absolute errors for the test set were 0.141 rad, 0.140 m/rad, 0.008 rad, and 0.038 m/rad for α_x, β_x, α_y, β_y, respectively.

For the cases examined, traditional optimization with the local optimizer takes significantly longer to converge: around 80 - 200 iterations (see Figure 3.6, Figure 3.7). In contrast, the neural network policy immediately makes an accurate guess of appropriate settings to reach a specified set of beam parameters. Even in cases where the guess is not completely accurate, it could provide an initial guess to skip the early stages of convergence for Simplex or another local optimizer.

Table 3.3: Ability to achieve $\alpha_{x,y}$ = 0 rad, $\beta_{x,y}$ = 0.106 m/rad for 3.1–6.2 MeV electron beams in one iteration. The test set consists of leaving out an entire set of energy ranges, from 4.8 to 5.2 MeV.

Variable	Unit	Train MAE	Test MAE	Train STD	Test STD
α_x	[rad]	0.012	0.075	0.011	0.046
α_y	[rad]	0.013	0.079	0.012	0.045
β_x	[m/rad]	0.008	0.004	0.006	0.002
β_y	[m/rad]	0.014	0.011	0.011	0.011

3.1.5 Conclusions and Future Work

These were encouraging results from an initial study aimed at assessing the basic feasibility of using a neural network control policy to provide suggested settings for a compact FEL, given a target electron beam energy. The results indicate that in one iteration the controller can set up the machine to achieve close to the correct Twiss parameters for arbitrary beam energies between 3–6 MeV, even for energy ranges that have not been observed during controller training.

This is a first step toward the development of a neural network controller that can facilitate fast switching between operational parameters along with fine-tuning. The TEU-FEL presents a good platform to explore this technique because of its relative simplicity in terms of the number of control parameters and its nonlinear beam dynamics. The true merit of the approach won't be

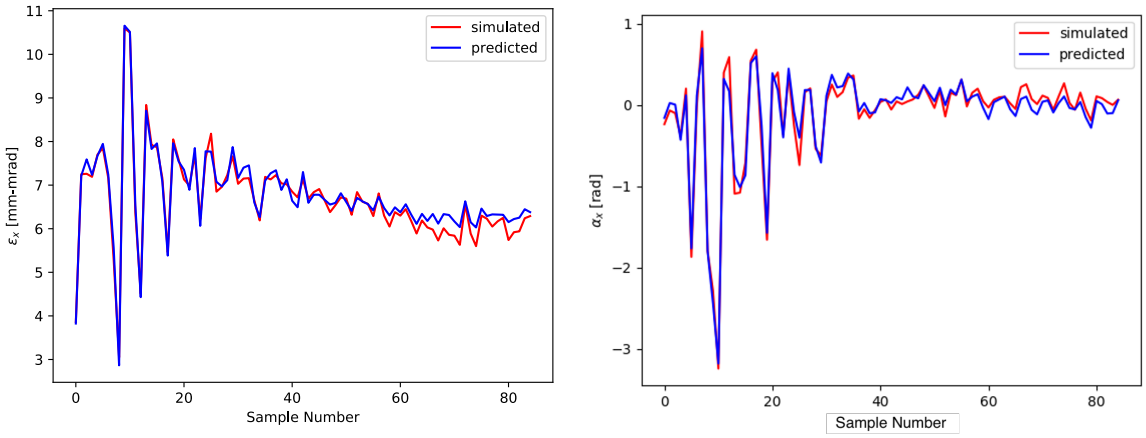


Figure 3.7: Neural network model predictions and simulated values for normalized ϵ_x and α_x on the new data set at 3.5 MeV for full the photoinjector and beamline study. Again, we are also showing data that was generated by running Simplex optimization on the model for one of the energy settings, so we also see that in this case it takes Simplex approximately 80 iterations to reach convergence. In contrast, the neural network policy immediately makes an accurate guess of appropriate settings to reach a specified set of beam parameters.

clear until it is tested experimentally, but these results suggest that a neural network control policy may be an expedient way of switching between operating states in an FEL.

Based on the results of this study, the next step would be to add the task of minimizing the emittance at the entrance of the undulator. As this requires finer adjustment of the RF phase, RF power, and solenoid strength in conjunction with the quadrupole settings, additional data sets consisting of the output from start-to-end optimizations and parameter scans were collected that include fine adjustment of these parameters. Figure 3.7 shows an example of the model's performance on the test set, which consisted of start-to-end optimization data for 3.5 MeV. This provides encouragement that the study could be extended to include emittance minimization with more complete variation of the photoinjector parameters allowed.

Along these lines, one could conduct further work with this test system by (1) adding emittance task to the controller training, (2) assessing the impact of noise and drift on the solution, and (3) incorporating FEL simulations and train the controller based on FEL output. However, the work here is sufficiently suggestive that the approach should work that it would be more worthwhile to

transition to a system where there is a possibility of doing experimental testing. This work was published in [201] and [202].

Chapter 4

Systems with Long-term Time Dependencies: Resonant Frequency Control in RF Cavities

Thus far, the discussion in this dissertation has focused on prediction and control with high-level machine settings (e.g. the magnet settings, steady-state settings for cavity phases and amplitudes, etc.). In the cases discussed, the evolution of the system behavior does not need to be taken into account when taking control actions (i.e. because the system evolution is faster than the rate of control actions taken). In other words, we can treat it as a single time step input, single time step output prediction or optimization problem. This means that there is no need to consider the impact of a series of actions taken when changing settings (i.e. the order of actions does not matter, and continuous dynamic control is not needed). However, this luxury does not extend to dynamic control over sub-systems. The RF system in accelerating cavities is one such sub-system, and in this case dynamic control is of critical importance for achieving high levels of beam quality and operational efficiency (where efficiency is considered in terms of both time spent tuning and recovering from trips, as well as the rate of bulk power usage during normal operation).

The RF pulse structure, cavity resonant frequencies, and related cooling and cryogenic systems all have a complicated dynamic interplay and ultimately impact the beam quality. To maintain proper acceleration of the beam to the appropriate energy and phase space properties, the resonant frequency is often partially controlled with cooling systems (e.g. water for normal-conducting cavities, cryogenic liquid for superconducting RF cavities). The cooling system behavior must be taken into account in tandem with the RF effects (e.g. variable heating from the RF, long transport delays of the cooling system, and thermal responses to both the cooling power and the RF power) when deciding how to dynamically control the combined cooling and RF system.

In this chapter, we examine model-based control over two types of normal-conducting RF cavities at Fermilab: the RF electron gun at the Fermilab Accelerator Science and Technology Facility

(FAST), and the radio frequency quadrupole (RFQ) at the Proton Improvement Plan II (PIP-II) injector test (formerly the PXIE injector experiment). Both of these systems exhibit long transport delays, large thermal time constants, and nonlinear responses. Improved resonant frequency control is an area that will be of critical importance for future superconducting linacs as well, including both the PIP-II superconducting linac and the LCLS-II superconducting linac.

4.1 Background on Resonant Frequency Control

In control of RF accelerating cavities, the frequency of the RF power entering the cavity (i.e. the drive frequency) needs to be well-matched with the natural resonant frequency of the RF cavity itself. When this condition is not met (i.e. when the cavity resonant frequency is "detuned" from the drive frequency), for a given RF power level the final accelerated beam energy will be below the maximum achievable value that is attained when the frequencies are matched. Detuning (i.e. shifting of the cavity resonant frequency away from the design value) also results in increases in the reflected power in the cavity, which leads both to additional waste heat and potential damage to components (e.g. RF amplifiers, RF windows, etc).

When small amounts of detuning occur during normal operation, the RF power can be increased to compensate for the loss of coupling efficiency and the same accelerating potential can be achieved. This ensures the beam will still be accelerated to the nominal required energy. However, this has several problems. First, the degree to which one can increase the RF power depends on the limits of the RF overhead (i.e. additional available RF power) built into the amplifiers. For accelerators that already require high RF power, it may not be possible to obtain amplifiers that have sufficient additional power to account for the observed detuning. Second, this procedure wastes a lot of energy, and operating costs are a major consideration for large accelerators. Third, the detuning also creates a phase shift in the RF cavity fields that also must be accounted for. For cases where the RF power is high, the RF heating itself is a major contributor to detuning, and even small changes in the RF power can have a very large impact on the resonant frequency.

Thus, in addition to using the RF forward power to make small corrections to the beam energy, active resonant frequency control is needed to keep the cavity at the same resonant frequency as the drive frequency. The resonant frequency is determined by the cavity geometry, which in turn can be adjusted via manual deformation of the cavity or by manipulating thermal expansion and contraction of the cavity with a dedicated cooling system. In the latter case, the cavity shape can be adjusted slightly by changing the rate and temperature of water flowing through internal or external cavity cooling channels. The flow rate and temperature of the cooling liquid (often water for normal-conducting RF cavities) needs to be adjusted when the RF power changes.

4.2 Background on Model Predictive Control

In Model Predictive Control (MPC), a system model and an optimization algorithm are used in conjunction to determine an optimal sequence of future controller actions such that the target output is reached within some future time horizon, subject to satisfying defined constraints (for example, there might be limits on the amount of transient overshoot allowable while the system is settling).

This is analogous to conducting optimization over a model where there is single time-step input and single time-step output (as would be done for the kinds of problems described earlier in this dissertation). In contrast, in this case instead of determining just one combination of settings to optimize a particular beam output, optimization over a time-ordered sequence of settings is needed. Such a scheme is useful for addressing system behavior where the response is dependent on the order of settings (e.g. as is the case with hysteresis), or where the time evolution of the system is slower than the rate at which settings are made (and thus the compounding impact of subsequent settings must be taken into account). In addition, if a series of future set points is known in advance (e.g. as may be the case in an RF ramp-up on system turn-on, or in pulsed operating modes where the pulse shape can be adjusted), the controller can act anticipatively to choose the best actions in other settings given the known future inputs for some of the components. Figure 4.2 and Figure 4.1 illustrate the basic concept.

The basic steps are to use a predictive model to assess the outcome of possible future actions by:

- Gathering state information for the present time step (e.g. previous measurements);
- Optimizing over the model to obtain the best series of actions to take;
- Selecting the first action in the optimal predicted series and execute it;
- Repeating the above steps.

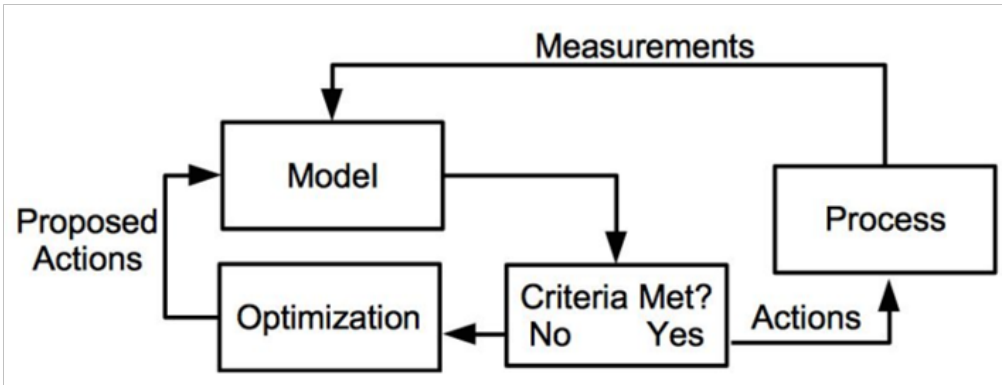


Figure 4.1: Basic concept of model predictive control.

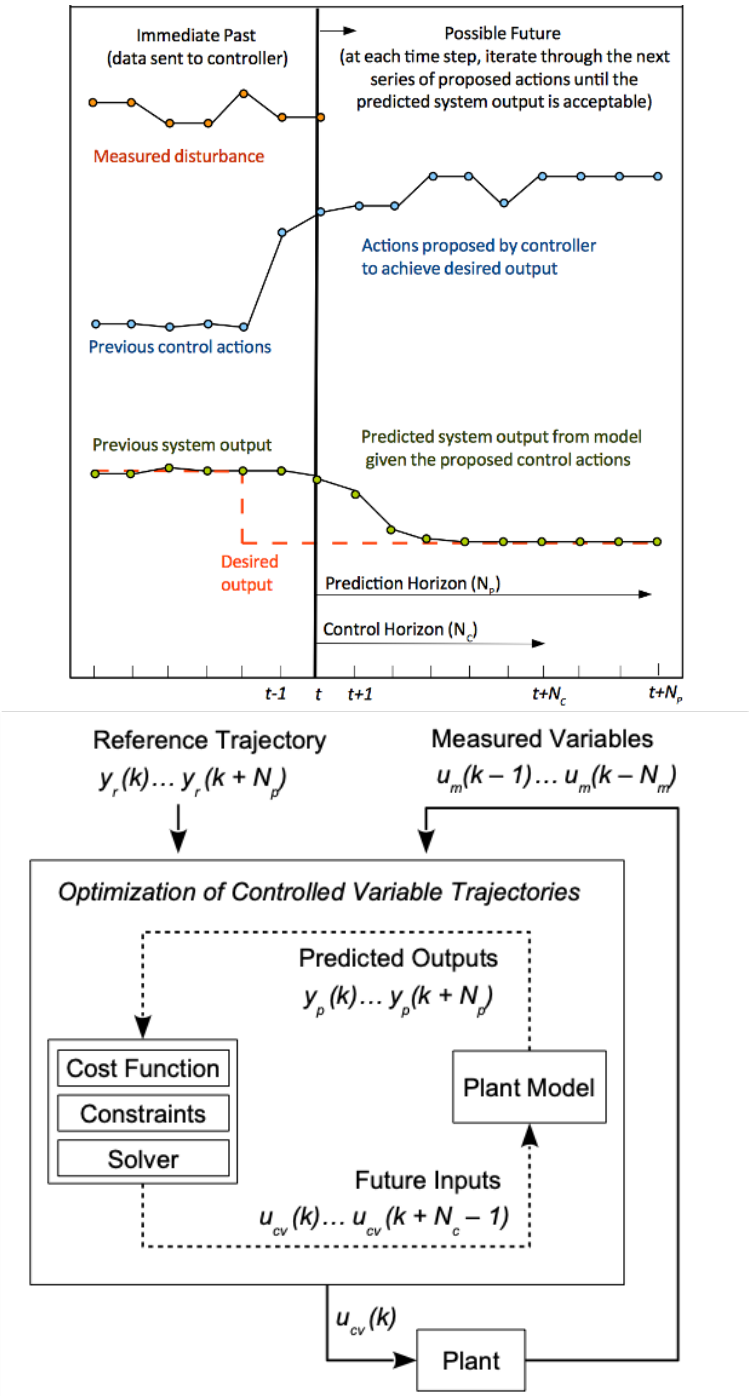


Figure 4.2: Basic elements of MPC. N_p is the prediction horizon, N_m is the number of previous measured values, k is the present time step, N_c is the control horizon, u_{cv} are the controlled settings, u_m are measured variables that provide additional input about the system state, y_p is the predicted output from the system (i.e. "plant") model given u_{cv} settings, and y_t is the target trajectory. Optimization is repeated with a suitable algorithm and constraints at each time step, and the first control settings in the predicted optimal sequence is sent to the system. The process then repeats in the next time step.

The major components of an MPC scheme are:

- A predictive system (or "plant") model. This could be any model that can execute quickly enough to make the optimization step of MPC feasible. Common choices are simple analytic or simple data-driven models (e.g. linear, state-space variable models).

- A prediction horizon consisting of (N_p) future time steps over which the system output is predicted at each iteration of the controller.

- A window of length (N_m) of recent measured values used as input to the system model. This gives an estimate of the present system state so that future predictions of the system evolution can be made. This window of recent values is moved along with the changing time step.

- A control horizon consisting of (N_c) future time steps. This corresponds to the number of time steps over which future control settings are optimized.

- A reference trajectory (y_t) which specifies the target output of the system evolution (for example this could be a desired step change in temperature in an output reading, a linear ramp, etc.)

Because MPC relies on repeatedly computing an optimal future trajectory for a series of future time-steps, there is a substantial tradeoff between model complexity and the ability to obtain a good solution within the control interval. This is particularly true as the optimization problem becomes more complicated (e.g. due to increasing model complexity, number and type of constraints, the specific cost function used, and the associated sophistication of the optimization algorithm that may be required for convergence).

For the optimization of future control actions there is a high degree of flexibility to customize behavior. For example, setting changes can be penalized to avoid wearing out motors and/or to avoid large step sizes that might shock the system. Similarly, target settings or secondary outputs that the controller should attempt to reach while optimizing the main output trajectory can be specified.

A standard formulation of the cost function in MPC is defined by terms quantifying the rate of controllable variable changes, the discrepancy between the present predicted output trajectory and the output reference trajectory, the discrepancy between the desired trajectory of controllable variables and their present trajectory (if applicable), and the degree to which any constraints are violated. Each of these terms is weighted in the cost function, which in turn is evaluated over the entire prediction horizon. If desired, small constraint violations can be allowed via constraint softening. A general formulation for the cost function is given by:

$$\begin{gathered}\sum_{j=1}^{n_{cv}} \sum_{i=0}^{N_p-1} \left(w_{1j}[u_j(k+i) - u_j(k+i-1)]^2\right. \\ \left. + w_{2j}[u_j(k+i) - u_{j,ref}(k+i)]^2\right) \\ + \textstyle\sum_{i=0}^{N_p} \left(w_3[y_r(k+i) - y_p(k+i)]^2\right) + w_4 b\end{gathered} \tag{4.1}$$

where w_{1j} is the weight for rates of change in the j^{th} controllable variable, w_{2j} is the weight for the j^{th} controllable variable target trajectory, w_3 is the weight for the output variable target trajectory, w_4 is a penalty weight for constraint softening, N_p is the prediction horizon, i is a future time step, y_r is the reference trajectory, y_p is the predicted output, k is the present control step, u_j is j^{th} the controllable variable value, $u_{j,ref}$ is the reference trajectory for the j^{th} controllable variable, b is a measure of constraint violation (e.g. proportional to sum of total deviation), and n_{cv} is the number of controllable variables.

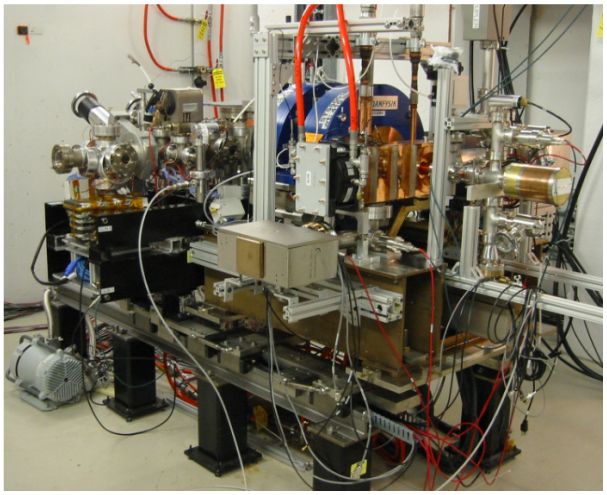
While MPC with a linear plant model has a long history of successful implementation in real-world applications, the task becomes significantly more challenging when a nonlinear model is used. Highly accurate but nonlinear models require an optimization algorithm that is suitable for nonlinear problems. Fortunately, this area has seen rapid development over the last fifteen years (for an overview see [157, 203–206]). Neural networks can be used as the model in MPC. A nonlinear neural network can be linearized at each time step and then used with standard linear optimization algorithms commonly used in MPC.

MPC can be considered to be analogous to model-based reinforcement learning in the following way. In some forms of reinforcement learning, a value function predicts the expected sum of future rewards given an action taken in a certain state. This is then used to improve the policy (map from states to actions). In MPC, the expected future reward is instead arrived at by optimizing settings and predicting the resultant reward over the entirety of the expected future time horizon. The action chosen by RL would be analogous to the first step of the trajectory that MPC solved for directly. The policy is simply the process of optimizing over the model, given the state input.

4.3 FAST RF Gun

4.3.1 Background on the RF Gun and Cooling System at FAST

The system that is used to regulate the resonant frequency of the electron gun at Fermilab Accelerator Science and Technology (FAST) facility was identified as an initial candidate for the application of neural network-based control methods. This was due to the large thermal time constants, long transport delays, recursive behavior in the cooling system, and long settling time and overshoot produced under PID control (described later). Waiting for the gun to settle while RF power changes were made (e.g. to change beam energies) was a substantial difficulty noticed during commissioning of the injector in 2013.



FAST RF Gun Parameters	
Gun Parameters	
Type	Photoinjector
Number of cells	1½
RF Mode	$TM_{010,\pi}$
Loaded Q	~11,700
RF Frequency	1.3 GHz
Frequency Shift	23 kHz/°C
Nominal Operating Parameters	
Macropulse Duration	1 ms
Repetition Rate	1–5 Hz
Bunch Frequency	3 MHz
Design Gradient	40–45 MV/m
Power Source	5 MW Klystron

Figure 4.3: FAST RF gun and nominal operating parameters (as of 2014).

The electron gun at FAST [179, 207] is a 1.5-cell copper RF photoinjector operating at 1.3 GHz in the $TM_{010,\pi}$ mode and is powered by a 5-MW klystron. It has a loaded Q of appx. 11,700. The resonant frequency of the gun is controlled using a water-cooling system, and it shows a measured 23-kHz shift per °C change in cavity temperature (consistent with that predicted at this RF frequency). The gun is designed to produce beam with 1-ms duration macropulses at a 1-5 Hz repetition rate at a bunch frequency of 3 MHz. The intended operational gradient is 40-45 MV/m, but as of 2014 it has also been operated at up to 47.5 MV/m. For adequate resonant frequency stability, the design requirements stated that the temperature of the water entering the gun should be regulated to within 0.02 °C [116]. At 40 MV/m and a 5-Hz macropulse repetition rate, the expected average dissipated power in the gun is 15 kW. An image of the gun and table of corresponding parameters is shown in Figure 4.3.

Resonant Frequency Control Hardware and Instrumentation

The resonant frequency of the gun is controlled entirely through a water-cooling system. A simplified schematic of the system is given in Figure 4.4. The two controllable water system variables are 1) the flow control valve setting (FCV) and 2) the heater power setting (HP). The T01 sensor reads the cold-water supply temperature, the T02 sensor reads the temperature just after the mixing chamber, the TCAV sensor reads the cavity temperature, and the TOUT sensor reads the temperature of the water leaving the cavity. The TCAV sensor is located in the iris of the gun.

Further description of the instrumentation specifications is given in [208], with several additional important details regarding the resistance temperature detectors (RTDs) and the associated data acquisition process. The original analog-to-digital converter (ADC) hardware units were found to have unstable read-backs, and they were replaced with new RTD sensors. This hardware change resulted in lower-resolution readings in T01, T06, and T02 (0.1 °C resolution rather than 0.01 °C resolution). Noise in the readings typically results in a variation of 0.2 °C. T02 was also converted to a digital reading by a Fluke 8846A multimeter with a resolution of 0.01 °C. The noise on the readings results in a variation of 0.02 °C. This is the same ADC setup that is used for TIN, TCAV, and TOUT.

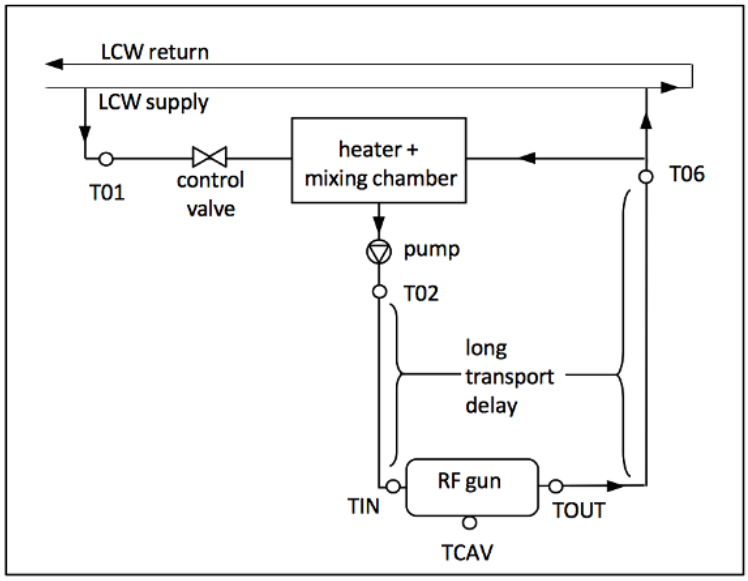


Figure 4.4: Layout of the water system and relevant instrumentation (not drawn to scale). T01, T02, TIN, TCAV, TOUT, and T06 are temperature sensors. The piping is not insulated, which introduces additional temperature disturbances from the ambient environment. For the regions marked "long transport delay," the piping traverses several parts of the building that at various times have different ambient temperatures. The control valve and heater/mixing chamber are not co-located with the gun (the latter is inside the shielded enclosure at FAST).

For this particular system there are several important considerations for control:

- Due to water transport time and thermal time constants, long, variable time delays exist between various elements in the system. A selection of these is shown in Table 4.1. This results in a minutes-long, damped oscillation in the temperature of the water entering the gun. An example of such an oscillation (under no control, i.e. "open loop") is shown in Figure 4.5.

- Without compensation, any change in the temperature of the water exiting the gun (either due to a change in the amount of waste heat from the RF power or a change in the temperature of the water entering the gun) will circulate back into the mixing chamber and have a secondary impact on the cavity temperature.

- The pipes through which the water flows are uninsulated and pass through several different areas of the building. Additionally, the closest ambient air temperature sensors, which read

the south cave temperature and south hall temperature, show variations of several °C from day-to-day, and more than 15 °C over longer durations. These two temperature readings are not always closely correlated. The relationships between T02, TIN, and TCAV vary measurably with these ambient temperatures, as does the relationship between TOUT and T06. Occasionally the steady state difference between TIN and T02 temporarily changes without a change in the ambient temperature readings being registered. Presumably, this is due to highly localized variation in temperature (e.g. from a fan, nearby equipment heating up during turn-on, etc.).

- Due to the TCAV sensor location at the iris and the cavity geometry, the temperature recorded will be higher than the real bulk cavity temperature under RF power. Thus, for resonance control using operator-specified TCAV set points, it is important to note that the set point required to maintain the proper resonant frequency will increase with increasing average RF power. This is also a good reason to regulate the measured resonant frequency directly, rather than regulating the TCAV sensor read-back (however, for FAST we first aimed to provide improved control over the TCAV setting, as it was also not outfitted initially with all of the needed probes for the resonant frequency calculation).

- There are fluctuations in the low conductivity water (LCW) supply temperature. While it is nominally regulated to within 0.5 °C, larger spikes do occur.

Table 4.1: Transport and Thermal Delays for the FAST RF Gun and Cooling System.

System Segment	Approximate Time [s]
Flow valve to T02	8-10
Heater to T02	5-10
T02 to TIN	32
TIN to TCAV	19-23
TOUT to T06	55-65
TIN to cavity frequency	16-18

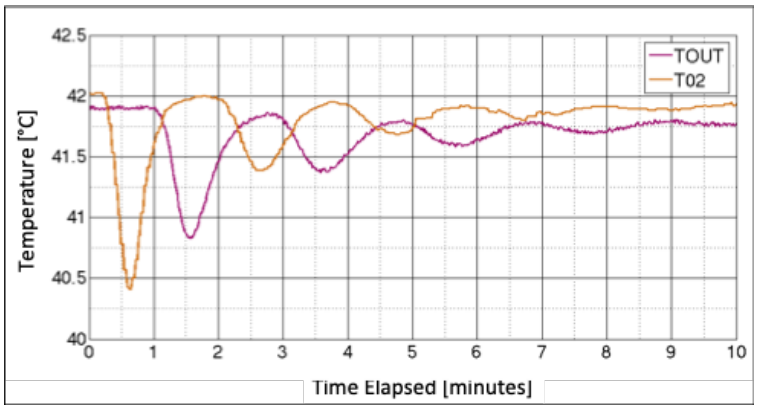


Figure 4.5: Oscillatory open-loop (uncontrolled) response in the water temperature at T02 due to mixing of the cold supply water and the water returning from the gun (TOUT). This oscillation was induced by a 20-second decrease in the heater power setting from 7 kW to 2.5 kW. Qualitatively, the response from an increase in the flow control valve is very similar (however, the system sensitivity to changes in the flow valve vs. the heater power setting are different).

These various issues (arising from space and cost constraints during construction) create a very complicated, dynamic interplay between the inputs. Space and cost constraints frequently arise when constructing large accelerator systems, and this sometimes results in subsystem designs that are not optimized for control (such as the water-cooling system in this case). While a good initial design that takes control into account is always preferred, adopting more sophisticated control methods can sometimes help achieve the desired performance in these cases. For example, as is shown in the following sections, MPC of the cooling system in this case is able to compensate for the limitations of standard PI control.

4.3.2 Assessment of Existing Feed-forward/PI Loop and Motivation for Improved Control

At the time our team was asked to collaborate on this problem, the cavity temperature was regulated using a feed-forward/proportional-integral-derivative (PID) controller that was developed at Fermilab. The feed-forward component was used to initialize the controller: it determines, using an analytic estimate, an appropriate flow control valve setting based on the RF power parameters and the expected cooling power of the water. This in practice provides roughly appropriate initial settings but still necessitates substantial fine tuning. The controller then continuously adjusts the

valve setting such that a desired TCAV set point is reached, while the heater power level is kept at a constant setting. An older version of the controller and its performance is described in [208].

Due to the TCAV sensor location at the iris and the cavity geometry, the temperature recorded will be higher than the real bulk cavity temperature under RF power. Thus, for resonance control using operator-specified TCAV set points, it is important to note that the set point required to maintain the proper resonant frequency will increase with increasing average RF power. This is also a good reason to regulate the measured resonant frequency directly, rather than regulating the TCAV sensor read-back; however, for FAST we first aimed to provide improved control over the TCAV setting. At the time, the gun was also not outfitted with all of the needed RF probes for a direct resonant frequency calculation.

The response of the PI controller to a 1-°C step change in the set point under no RF power is shown in Figure 4.6. While it may at first appear that these are oscillations due to poorly-tuned PID gains, in this case it is actually due to the thermal and transport delays between control actions and the sensor read-backs. The combined effect of the long time delays and the recirculation of the water through the system are responsible for the appx. 0.6-°C initial overshoot, the subsequent oscillations, and the long settling time. In the instance shown, the system takes appx. 23 minutes to reach a steady state.

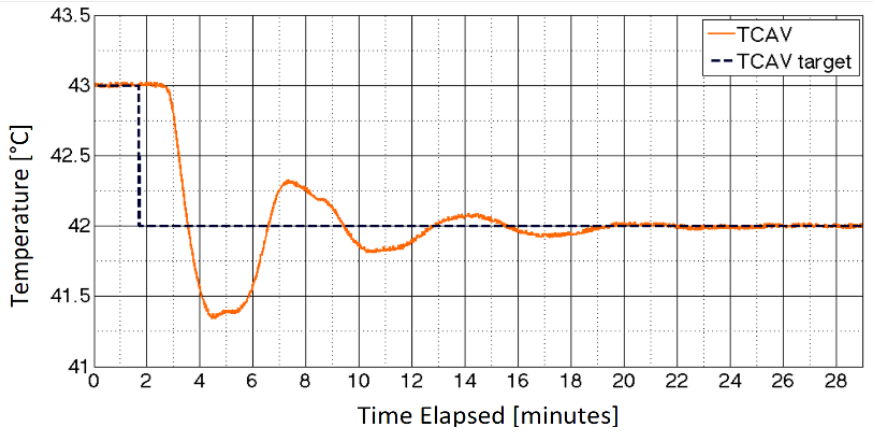


Figure 4.6: 1-°C step change under the existing PI controller. Note that the oscillations are due to the thermal and transport delays in the system response to controlled actions; they are not due to the PI loop gains.

Typically, without disturbances from the RF power or T01, the PI controller regulates the TCAV temperature to within 0.03 °C of the set point during steady state operation. The standard deviation of the TCAV temperature over 47,132 representative data points is 0.012 °C. For TIN, this corresponds to the temperature being kept within 0.04 °C of the mean temperature at steady state, with a standard deviation of 0.013 °C.

While this is acceptable for long periods of steady-state operation, regulation using the PI controller becomes problematic under dynamic conditions (e.g. for switching between RF power levels corresponding to different beam energies). For example, during RF turn-on, the overshoot results in reflected power often nearing and sometimes exceeding the threshold at which damage to ancillary components becomes a concern. Even with operator-mediated, gradual increase of RF power during normal operations at appx. 2.33 MW forward power, across 8 instances sampled the controller initially overshoots the TCAV set point by an average of 0.19 °C with a standard deviation of 0.02 °C. This results in a mean increase in reflected power of 70.6 kW over the steady state value with a standard deviation of 17.4 kW, culminating in a total mean reflected power of 103.0 kW. Damage can occur to the RF window at around 100 kW. There is no self-excited loop mode implemented for this cavity to circumvent this issue during start-up (i.e. by allowing the RF drive frequency to follow the cavity resonant frequency); therefore, one must exercise considerable caution when powering up the system.

Finally, by relying on manual adjustment of the TCAV set point in order to stay on resonance, operational time constraints combined with the slow settling time make it less likely that manual tuning to ensure the gun is exactly at the desired resonant frequency will be conducted consistently. This also reduces overall operational efficiency, as the low-level RF system increases the forward RF power in response to the reduction in field caused by moving away from the desired resonant frequency. In light of this, it becomes quickly apparent that having a control system that automatically adjusts TIN until the system is operating at the proper resonant frequency is absolutely needed.

The above situation motivates the use of MPC, where the system evolution under setting changes and disturbances is considered, enabling anticipative, feed-forward action to be continuously taken during operation. As the system is difficult to model analytically with a high degree of accuracy, the use of a learned model could enable the complicated-to-model interrelations between the settings, measured disturbances, and system output to be taken into account instead.

Significant improvement in the settling time, amount of overshoot, and disturbance rejection could likely be gained with an MPC setup. This would increase the operational efficiency of the gun by reducing the need to rely on the RF overhead to keep the cavity field constant, increase the total useful machine time by reducing the time spent waiting for the system to settle, and assist in the management of reflected power by more tightly regulating the cavity temperature under dynamic conditions. We approached the problem with the following steps in mind: 1) characterize the system, 2) develop appropriate neural network models of the system and assess their accuracy before proceeding, 3) use the neural network models in model-based control (e.g. MPC).

4.3.3 System Characterization

As this system was under commissioning during these studies, substantial initial characterization was needed before proceeding. The main goals for the characterization of the system were the following:

- Accurately identify the transport delays.
- Quantify the combined effects of TOUT, T01, flow control valve setting, and heater power setting on T02 (and subsequently on TIN and TCAV).
- Quantify the combined effects of TIN and RF power on the TCAV reading.
- Quantify the impact of ambient temperature on the temperature differences seen within the water system and the cavity (TOUT to T06, T02 to TIN, and TIN to TCAV).

Data Sets Obtained

In Figure 4.7 - Figure 4.10 a selection of the main measured data sets are shown. Variables that relay redundant information or did not undergo significant changes are not shown. The purpose of Set 1 was to obtain data for many combinations of flow control valve and heater power settings. The purpose of Set 2 was to obtain data for several power settings, primarily targeting the relationship between TIN, RF power, and TCAV. The purpose of Set 3 was to obtain larger variations in the flow control valve and heater power settings. Set 4 consists of changes in the temperature set point of the PI loop. Set 5 consists of data gathered over several days during normal operation at a cavity gradient of 42 MV/m. The sensor replacements described earlier (affecting the resolution and noise floor of the readings) occur after Sets 1 and Set 3. Several other small data sets were also examined.

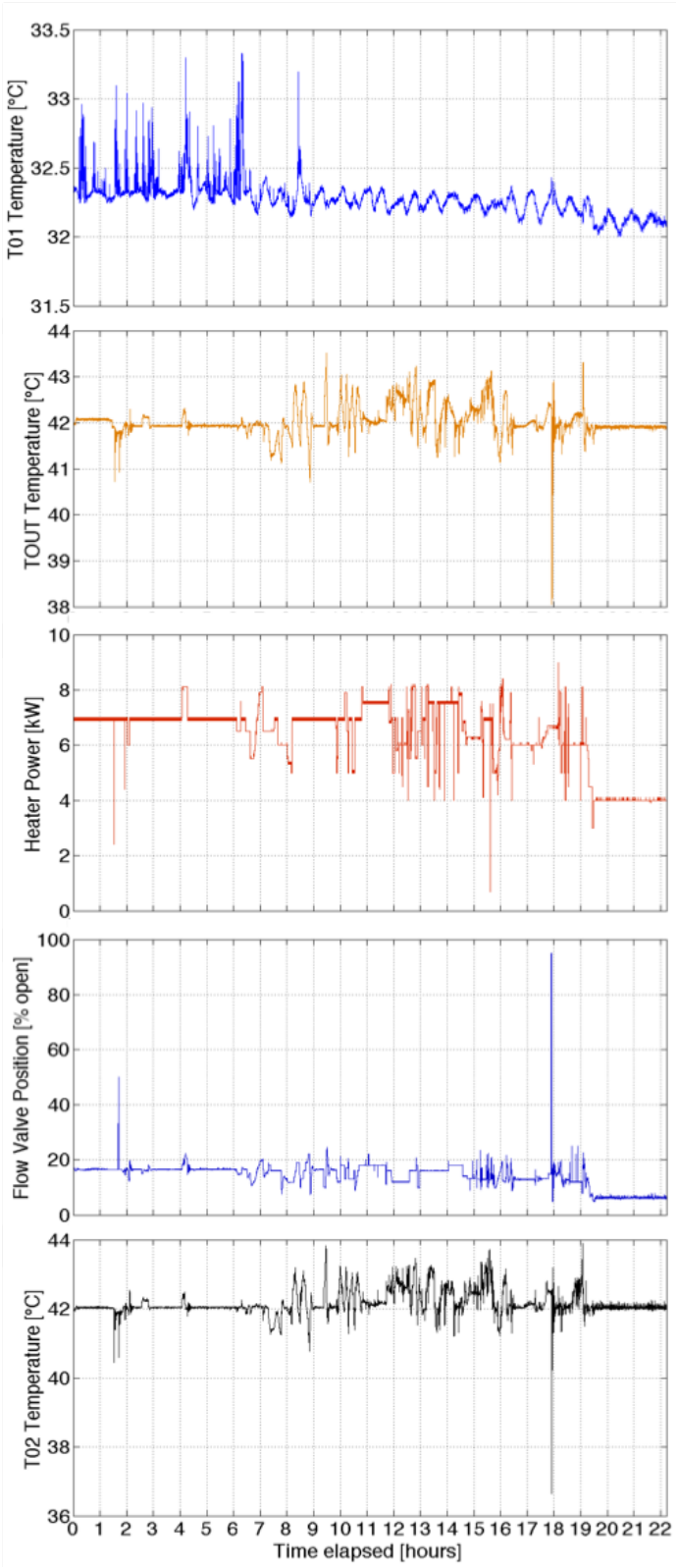


Figure 4.7: Data Set 1. Water system data with the gun running at low forward power (appx. 65 kW). Changes to the flow control valve and heater power settings were made. The PI controller was not active.

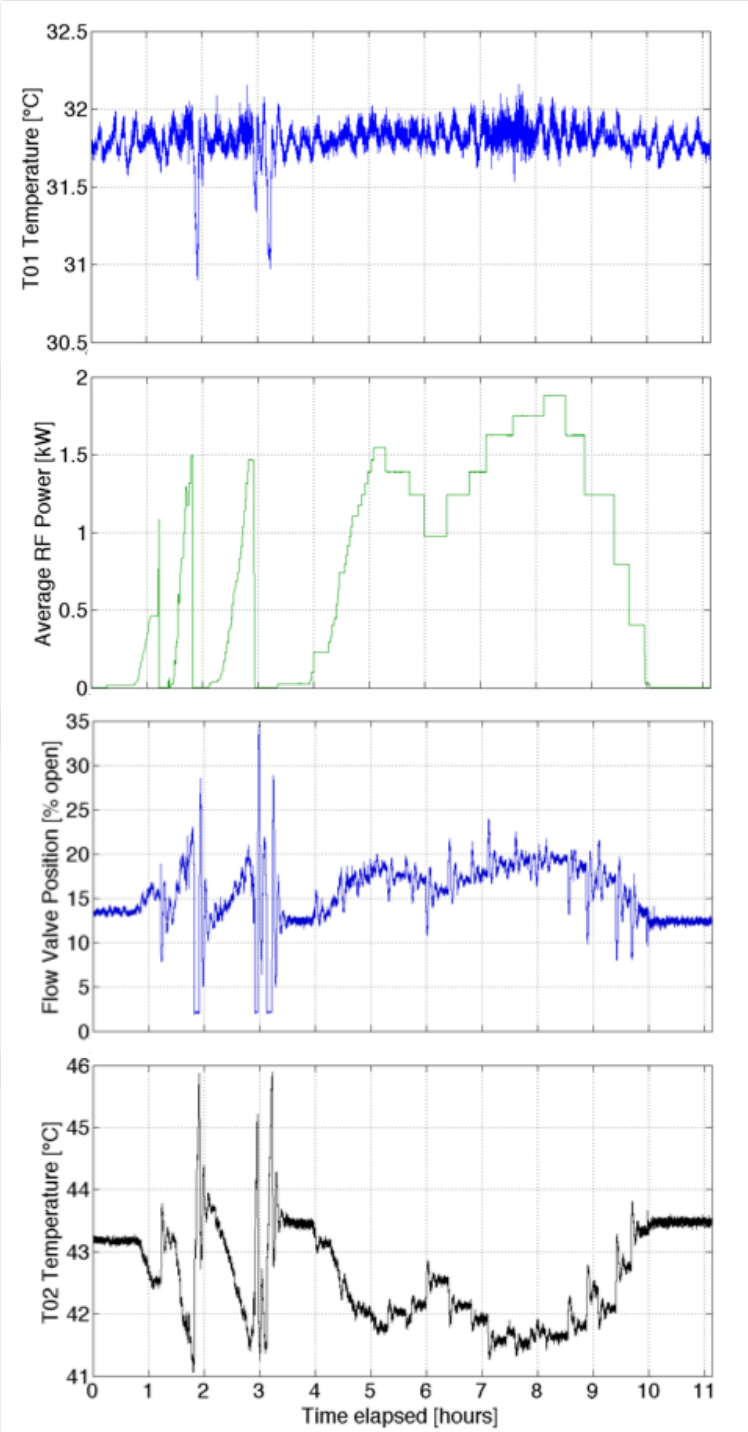


Figure 4.8: Data Set 2. System data under RF power to establish the relationship between the temperature of the water entering the gun, TCAV, and RF power. Due to operator concerns about reflected power, the PI loop was enabled during this time. Note that in between Set 1 and Set 2, the ADC hardware for some of the sensors (T01, T06, T02) was changed, resulting in lower-resolution readings.

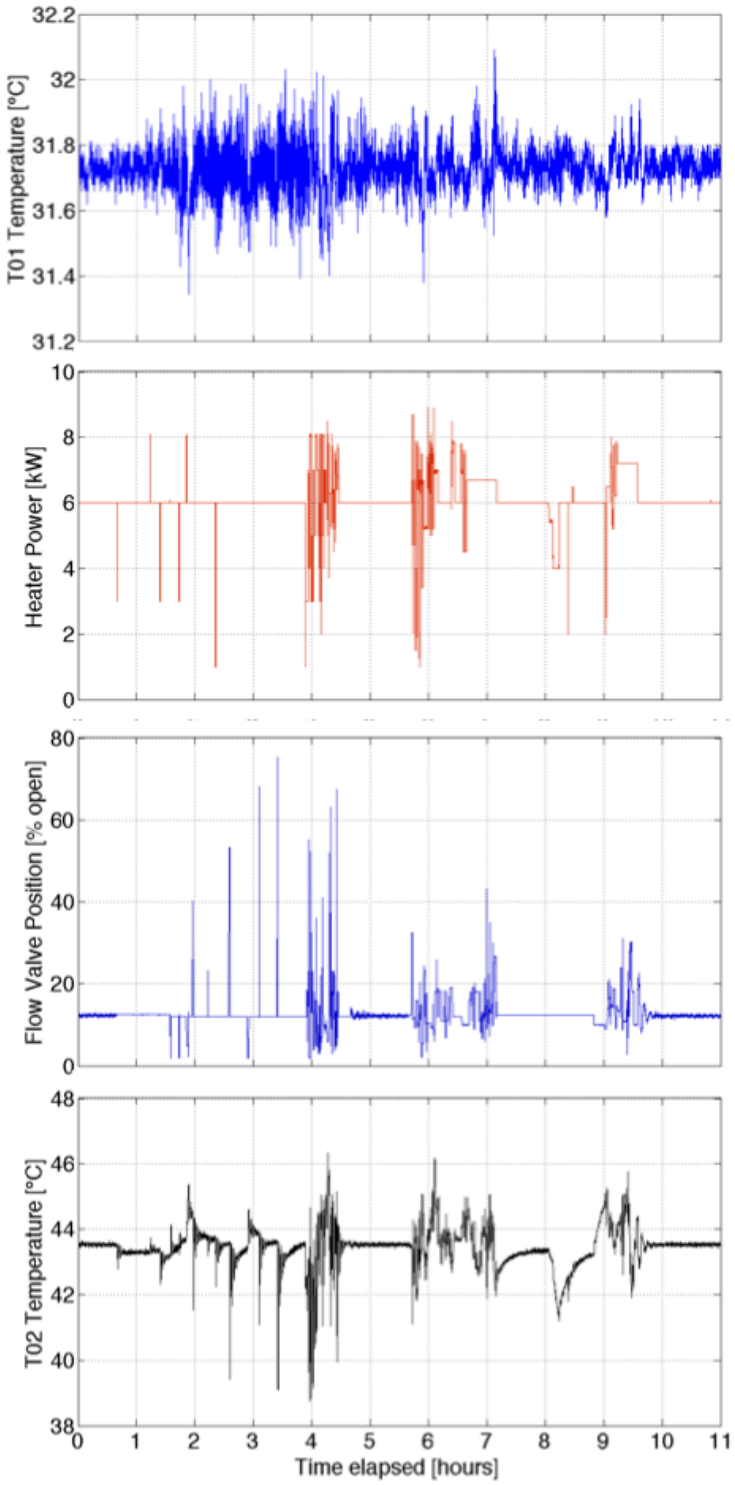


Figure 4.9: Data Set 3. Water system data for higher-magnitude changes in the flow control valve and heater power settings. The gun was off during this set, and the PI loop was not enabled.

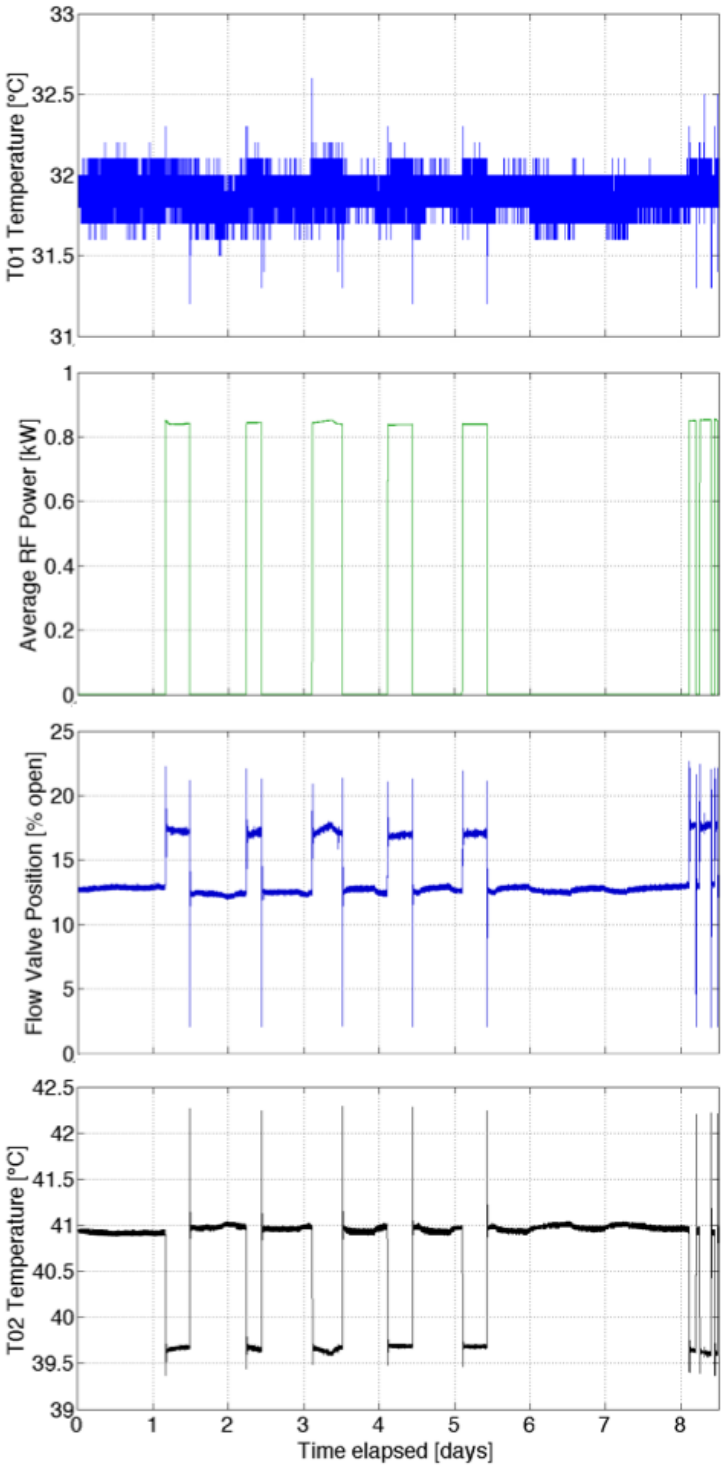


Figure 4.10: Data Set 5. Normal operations at 2.33 MW forward RF power. Note that, aside from the major changes in the un-powered vs. powered state, smaller changes in the T02 temperature roughly track changes in the ambient temperature, because the relationship between T02 and TCAV slowly changes and the PI loop compensates for this. Furthermore, changes in the T01 supply temperature can be seen as the RF power from the gun increases the thermal load on the entire water-cooling system (including recirculation back to the main chiller and cold water supply).

Influence of Ambient Temperature

The three primary areas where thermal losses could potentially impact the system are: (1) the water in transit from T02 to TIN, (2) the water in transit from TOUT to T06, and (3) losses from the cavity surface to air or attached components. Given the location of the ambient temperature sensors, their readings give us only a rough approximation of the air temperatures encountered by the bare pipes. Figure 4.11 shows the variation in the south hall and south cave temperatures over the course of several days. Figure 4.12 shows the temperature difference between TCAV and TIN as a function of the difference between TCAV and the cave temperature. The difference between T02 and TIN as a function of the ambient temperature shows a much less significant trend (the linear slope is 0.002, about a factor of 10 smaller), and the intercept changes slightly when the gun and its associated equipment is running, perhaps indicating that some local heating is raising the temperature of the TIN reading. Relative to the overall resolution of the sensors, this offset is not significant.

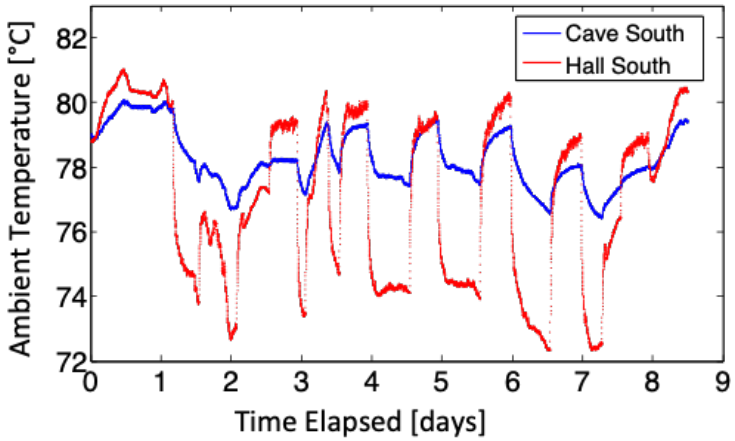


Figure 4.11: Variation in ambient air temperature over several days. Greater variation does occur.

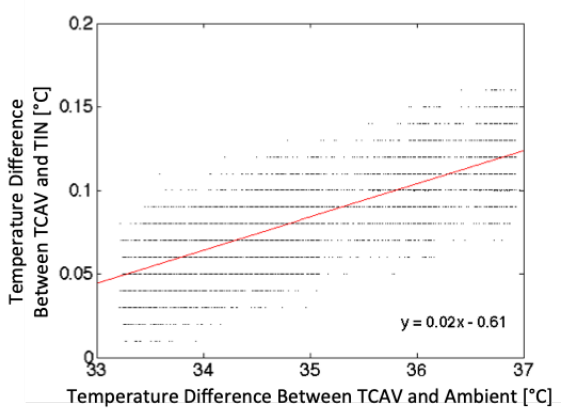


Figure 4.12: Temperature difference between TCAV and TIN, relative to the temperature difference between TCAV and the ambient air temperature reading. Heat is lost at varying rates from the un-insulated water transport pipes and exposed copper surface of the gun.

System Under Power

Figure 4.13 shows the steady state difference between TOUT and TIN and the steady state difference between TCAV and TIN as a function of average RF power. This is useful for determining what set point is needed for TIN such that the desired cavity temperature under a given average RF power level is reached. TOUT $-$ TIN diverges significantly from TCAV $-$ TIN at steady state. This is likely in part due to the effect that local heating from the RF has on the iris region where the TCAV sensor is located.

For a flow rate F in [GPM] and the water-cooling capacity C_c in [GPM-°C kW^{-1}], the cooling power is given by

$$P_{cool} = (TOUT - TIN) \times F \times {C_c}^{-1},$$

and at steady state, the cooling power is balanced with the average RF power input to the cavity (i.e. $P_{cool} = P_{IN} \approx P_{RF,avg}$). This relationship is shown as "TOUT $-$ TIN expected" in the line on Figure 4.13.

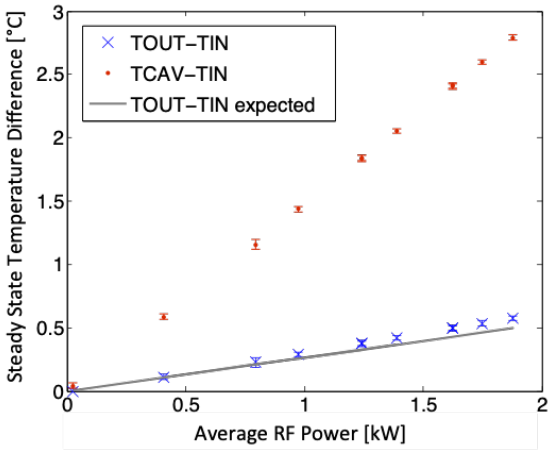


Figure 4.13: Difference between TIN and TCAV, as well as TOUT and TIN, as a function of RF power. The data were recorded at relatively constant cave temperature and a constant flow to the gun of 14.5 GPM. TOUT $-$ TIN diverges significantly from TCAV $-$ TIN at steady state. This is likely in part due the effect that local heating from the RF has on the iris region where the TCAV sensor is located.

4.3.4 Neural Network Modeling

In modeling the system, the primary aim was to investigate several model structures and use these to inform the controller design. While some individual elements of the system can be modeled analytically, it is difficult in this case to accurately model the entire system and the interplay between all relevant variables analytically.

The main variables examined were 1) the choice of model inputs and time windows of previous data for each input, 2) the combination of training data to use, 3) the neural network architecture. Based on the predictive performance and sensor resolution, there was also potential to break the problem into smaller MPC components (e.g. separate the cavity temperature / RF setting control from the water system control to various degrees). To investigate this, the following general model structures were assessed:

- A model to predict TCAV from TOUT, T01, flow control valve, HP, and RF power readings.
- A model to predict T02 from TOUT, T01, flow control valve, and HP readings.
- A model to predict T02 that also includes the south cave and south hall temperature readings as model inputs.
- A model to predict TCAV from TIN and RF power readings, and a similar model that uses T02 instead of TIN.

Data Pre-processing

The mean values were subtracted from the data, and the data were scaled to a range of - 0.5 to 0.5. The scaling was done in this way to allow for higher amplitude signals to later be included as future data was collected (e.g. so that all data would still be within the range -1 to 1). A 2.44 °C calibration offset in T02 for Sets 1-4 relative to Set 5 was adjusted for by subtracting it from the measured values. In addition, a zero-phase digital filter was applied to T02 readings for data sets containing the noisier, lower-resolution data that resulted from the instrumentation change. The remaining data were not filtered. This was found to produce more consistent performance than did

either filtering all of the data (and using incremental filtering for incoming data in online testing) or filtering none of the data.

Training Procedure

The training procedure described applies to all trials for ruling out different model structures and architectures. The model training data was used to teach the neural network the proper input-output relationship via supervised learning. In this case, the Broydon-Fletcher-Goldfarb-Shanno (BFGS) [79] algorithm, a popular quasi-Newton optimization method for unconstrained nonlinear problems, was used to find the optimal weights and biases of the network. The predictive performance on the testing data was used to assess the models, both in one-step-ahead prediction and in larger prediction horizons (20-to-200 steps ahead). To ensure that the particular training algorithm used was not skewing the results significantly, results were compared with those obtained using the Levenberg-Marquardt algorithm [209, 210] and scaled conjugate gradient [200].

For each candidate model architecture and set of inputs, multiple individual networks were trained (5-to-10) and tested. This helps to ensure that the network architecture and configuration of the training data itself is responsible for the performance, rather than a single exceptionally poor or good solution due to random initialization of the weights.

Different combinations of data sets were removed entirely from the training/validation data. The best performance was obtained when the networks were trained first on Set 1 and then subsequently trained on a portion of Set 3 (with the rest of Set 3 being reserved for testing). Set 1 had higher resolution T02 readings and contained a variety of flow valve and heater setting combinations, but with relatively small adjustments. Set 3 had fewer, but higher-magnitude adjustments and lower resolution T02 readings. For the models predicting the TCAV readings, additional training was also conducted with a selection of data from Set 2 under RF power.

Neural Network Architectures

A feed-forward architecture was adopted, where a sequence of input variables is provided for the prediction at every time step. This still captures the time dependencies and system evolution,

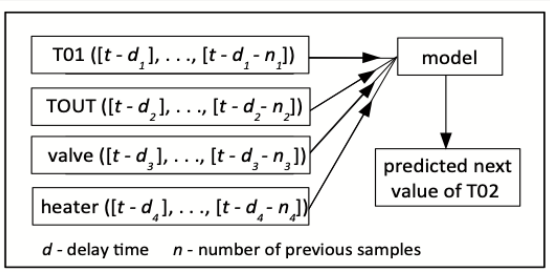


Figure 4.14: Illustration of T02 neural network model inputs and outputs. A series of measured samples are provided. Initial inputs are delayed by an appropriate amount determined by system dead time.

albeit not in as direct a fashion as could be done with a recurrent neural network (RNN) architecture. Because the time delays can vary, a short series of previous values is given for each prediction time step. For long transport delays, only relevant values are provided (i.e. dead time is removed by introducing a delay in the inputs). For the T02 models, the best performance was found when the neural network was provided with 15 prior seconds of flow control valve, heater, and T01 readings, and 30 seconds of prior TOUT data. The TOUT data inputs are delayed by 50 seconds and interspersed at a varied interval such that the 60-70 second range is represented more heavily. It was also found that the T02 models generally performed better when excluding the ambient temperatures. In addition, better solutions for long-term future predictions were obtained when excluding previous values of T02 as an input (thus forcing the neural network to really learn the relationships between T02 and the input variables, rather than relying on autocorrelation).

As a historical note, while RNN architectures were considered as another possible option, at the time of this work (2013), RNNs were considered to be very difficult to train in a stable fashion. The extensive suite of well-vetted training procedures and code bases for long short term memory (LSTM) and gated recurrent unit (GRU) networks that now exist were not yet introduced.

Table 4.2 shows the performance of several neural network model designs. For the best-performing network out of each model category, the average absolute error across all prediction instances in all of the data sets is reported, along with the standard deviation. For the TCAV predictions, the performance with and without RF power is reported separately. These models show reasonable accuracy on average for predicting the temperature. As expected, the least accurate

Table 4.2: Average Performance for FAST RF Gun Model Designs

Prediction	Activations	MAE [°C]	STD of Error [°C]	Max. Error [°C]
T02	tanh	0.018	0.037	1.049
T02 (with ambient temp. input)	tanh	0.022	0.043	1.317
T02	linear	0.058	0.266	2.915
TCAV (no rf power)	tanh	0.011	0.012	0.131
TCAV (with rf power)	tanh	0.259	0.287	1.390

model in terms of mean absolute error is the one for the TCAV temperature sensor readings when powered on (for which we had the least data). This could likely be improved with additional training data over a wider range of RF powers.

For the T02 models, a variety of configurations were examined (including variations in the number of layers, the number of nodes in each layer, and the type of activation functions used). Out of these, the best performing models used two hidden layers with 20 nodes in the first layer and 5 nodes in the second layer. The nodes in the hidden layers use a hyperbolic tangent activation function.

The T02 models with and without ambient temperature were the most extensively trained and vetted, and the final neural network performs well across all data sets. The best-performing full TCAV model is the same structure as the T02 model. Without power, it performs better than the T02 model, which may be due to the sets of lower-resolution T02 measurements included in training, or due to the fact that TCAV is less susceptible to oscillations and noise in the water system because of its large thermal mass. In testing the TCAV models under RF power, there were steady-state offsets in the predicted output. Additional training under a greater variety of RF power levels would be needed to improve this.

4.3.5 Model Predictive Control over the RF Gun and Cooling System

Establishing satisfactory control of the water temperature at the cavity entrance is the first step toward ensuring the gun is kept at the proper resonant frequency. Because the long transport delays

and recursion in the water system are a major challenge for the feed-forward/PI controller, it made sense to address this problem individually before moving onto a complete controller. Furthermore, while the cavity temperature (as reported by the TCAV sensor) is just an intermediate variable when considering the final goal (resonance control), framing an initial controller around TIN and TCAV also enables direct comparison with the existing feed-forward/PI loop.

As such, we aimed to design a modular controller that could be altered with ease to fit either TCAV-oriented regulation or resonant frequency-oriented regulation. The base controller regulates the temperature of the water entering the gun by modulating the flow control valve setting and heater setting. This controller can then be nested within another control loop that determines what the water temperature needs to be in order to either a) directly minimize the detuning or b) achieve an operator-specified cavity temperature set point.

MPC Controller for the RF Gun

First, we developed a benchmark controller for the water system to compare the performance of the feed-forward/PI controller with MPC. Because the temperature difference between T02 and TIN changes over time, T02 was chosen as the variable to control. The basic structure is shown in Figure 4.15. First, the operator provides a TCAV set point, which is then translated into an approximate T02 set point by exploiting the learned steady-state relationship between RF power, T02, and TCAV. MPC then is used to manipulate the flow control valve and heater power settings such that the desired T02 trajectory is obtained.

By monitoring temperature changes in the water leaving the gun, the controller can compensate for them before they reach T02. Monitoring TOUT provides plenty of time for actuation of the heater to take effect. By also monitoring T01 and using this as a model input, adjustments can be made to compensate for fluctuations in the LCW supply temperature. One critical weakness of this design is that the ambient temperature can affect the relationship between T06 and TOUT, and at present this is unaccounted for (e.g. by model updating or an adaptive offset on the TOUT input).

For the benchmark MPC, we were only concerned with getting a rough idea of how well a simple MPC system might perform compared with the PI loop. To this end, we linearized the

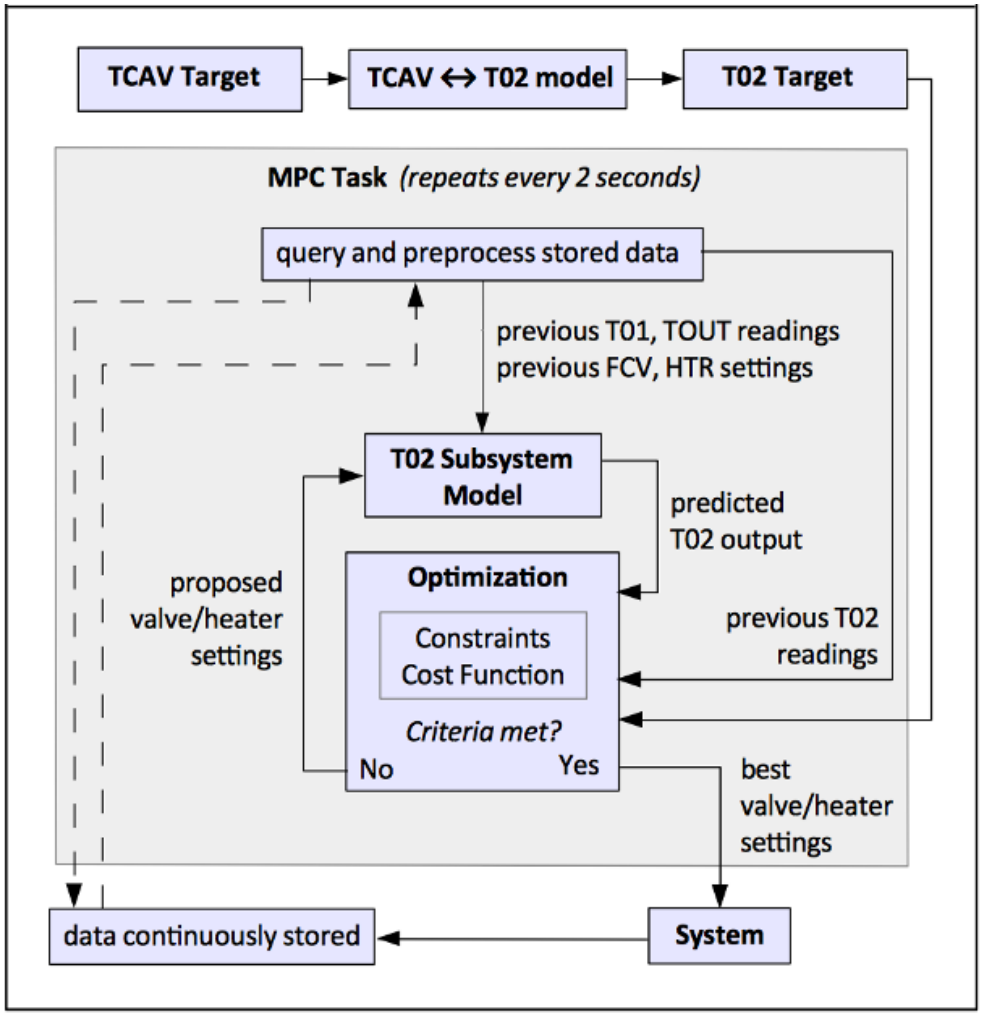


Figure 4.15: Structure of MPC for temperature control of the RF gun.

neural network model around the present operating point at each time step, and used the sequential quadratic programming solver QPKWIK [122].

In the benchmark controller, all constraints were strictly enforced and the output range was unconstrained. Through a combination of simulation with the learned neural network models and empirical testing on the gun, a set of the MPC parameters that achieve reasonably good performance were obtained. These parameters are given in Table 4.3. A rudimentary neural network model was used to translate between the TCAV set point and the T02 set point. One could instead use the steady state relationship, but we wanted to capture the dynamic responses as well.

Table 4.3: Benchmark MPC Parameters.

Variable	Value	Units
Valve max rate	10	% open/sec
Valve upper limit	70	% open
Valve lower limit	2	% open
Heater max rate	4	kW/sec
Heater upper limit	9	kW
Heater lower limit	1	kW
Prediction horizon	100	s
Control horizon	20	s
Control interval	2	s
Valve rate weight	0.4	-
Heater rate weight	0.5	-
T02 output weight	0.3	-

Performance Evaluation of MPC

Figure 4.16 shows the performance of the benchmark MPC for a 1-°C step change in the TCAV set point. The settling time is appx. 5× faster than that of the PI loop, and there is virtually no overshoot. After the step command for the cavity is issued, the MPC brings T02 to within 0.02 °C of its respective set point in about 3 minutes. Correspondingly, TCAV is brought to within 0.02 °C of its set point in about 5 minutes. While the transient behavior in this instance is clearly an im-

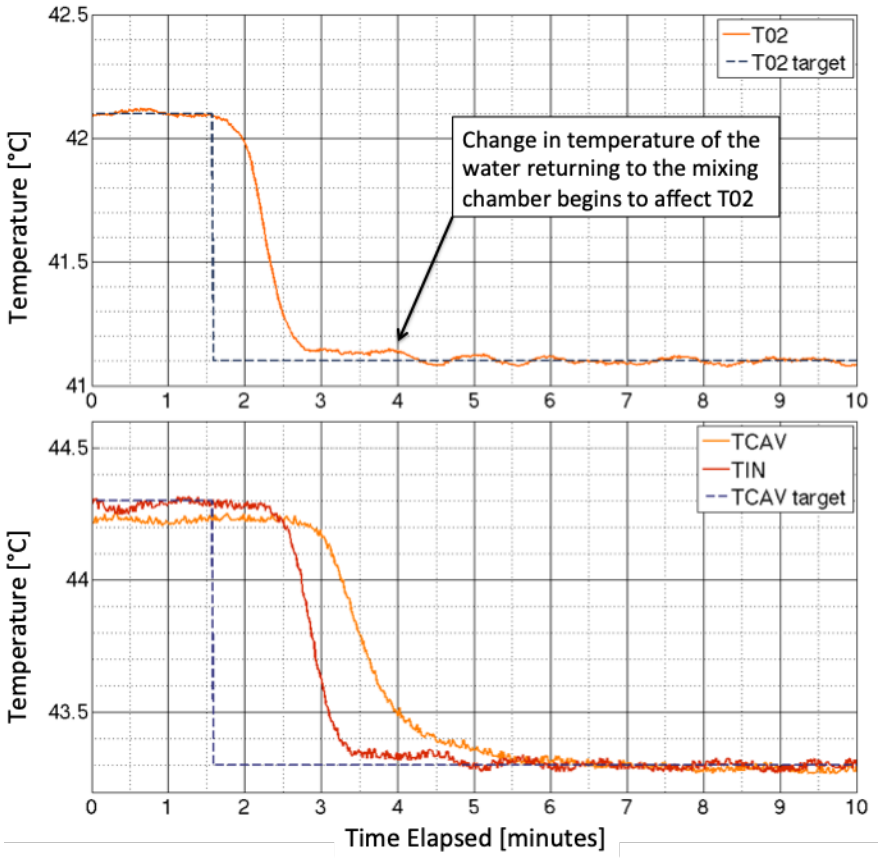


Figure 4.16: 1°C change in TCAV with the benchmark MPC. Note that the scales are smaller than those shown for the PID controller. These data were recorded as part of a series of steps in the TCAV set point. Note that this is not a perfect 1 °C step, as there is an offset between the original TCAV set point and the final value it obtained in the prior to step.

provement over the transient behavior of the PI loop, additional data is needed to fully characterize both the transient and the steady state performance.

Note that the scales in Figure 4.16 are smaller than those shown for the PID controller performance in Figure 4.6 (1.5 °C vertical extent in the former vs. 2.5 °C vertical extent in the latter, and 10 minutes extent in the former vs. 30 minutes extent in the latter). As with the case shown in Figure 4.6, no RF power is going to the gun.

Figure 4.17 shows the measured and requested flow control valve and heater power actions. We see an initial adjustment (the valve opens and the heater power decreases), followed by an adjustment to compensate for the lower temperature of the water exiting the gun. Overall, the control performance was a substantial improvement over the existing feed-forward/PI controller.

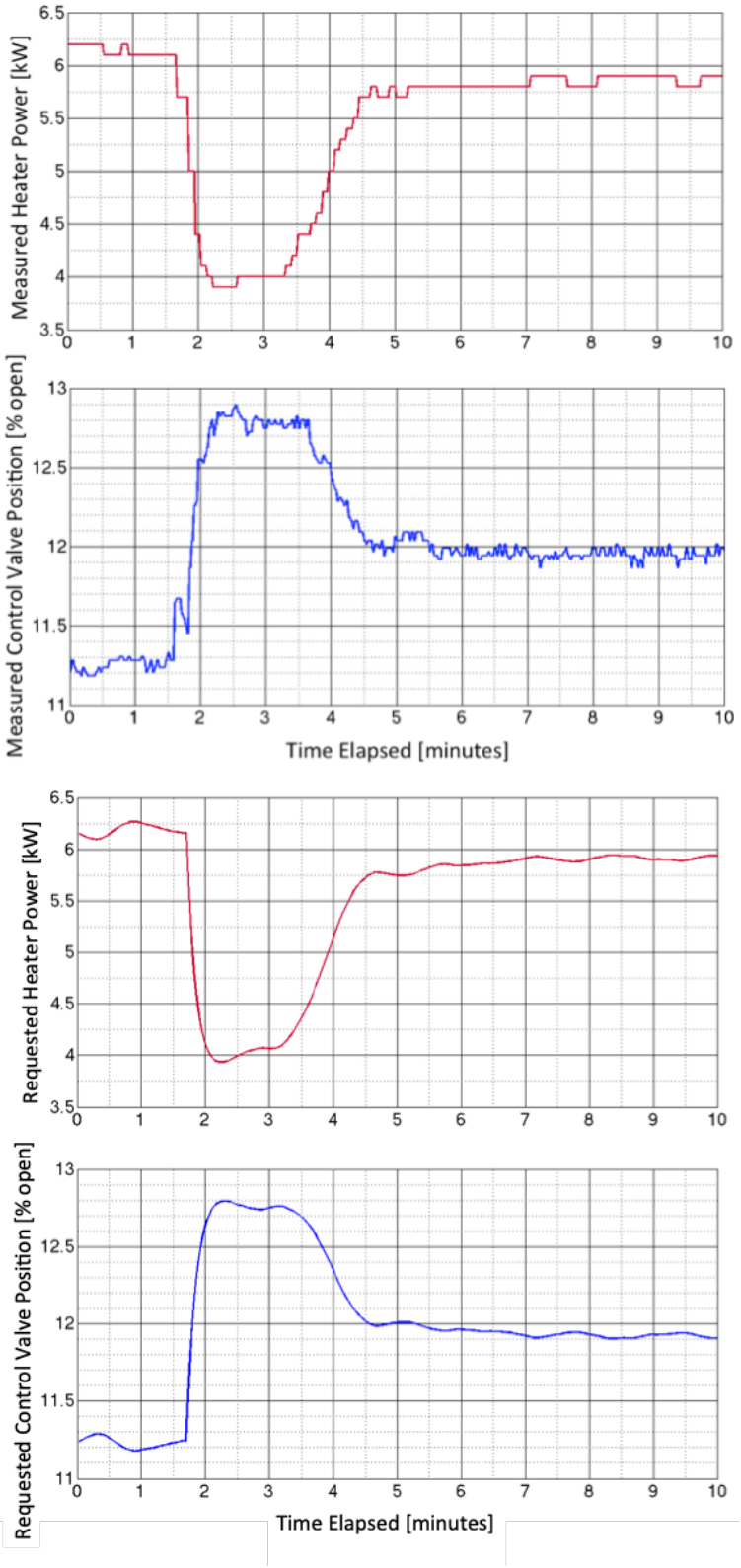


Figure 4.17: Measured (top) and requested (bottom) flow control valve and heater actions.

Limitations and Potential Improvements

One limitation of the controller is that it used previous requested flow control valve and heater power settings in the model, rather than using measured values. Given how much the requested values deviated from the measured values, the performance of the controller could likely be improved by using the measured values instead.

Improvement could also be made to the timing of control actions. The small oscillations in T02 that start around the 4-minute mark are the result of imperfect timing in the compensative actions for the recirculating water. They were replicated in simulation by introducing a mis-match in the timing of TOUT in the plant relative to that in the model used for the MPC controller. During training, the time delay between when the controller issues a command, when the command is received by ACNET (Fermilab's general control system) [211], and when the actuators in the hardware respond were not accounted for.

Attaching an adaptive offset to TOUT could also improve the performance. At present, the controller does not adjust the expected temperature reaching the mixing chamber as the ambient air temperature changes. This could result in over- or under-correction.

The controller also needs to be tested over the full power range. The T02 model performs well under powered conditions, and thus in principle the MPC should be able to compensate for temperature changes in the water exiting the gun associated with RF power adjustments. However, the component that converts the TCAV set point to a T02 set point needs to be more carefully designed before this is implemented for regular use. Steady state offsets in the modeling could be accounted for by adding slow feedback to the component that translates between TCAV and T02 under RF power. Alternatively, additional training data under a wider variety of RF power levels could be obtained. Because TCAV does have a slow thermal response, an extension of this is to use a second MPC to determine a desirable T02 trajectory (rather than a single set point given at each time step).

4.3.6 Conclusion and Future Outlook

Thus far, neural network system models have been constructed, and a benchmark model predictive controller that makes use of a neural network model has been designed and tested. By incorporating predictive control, anticipatory action can be taken that provides improvement over the standard feedforward/PI algorithm. This controller takes a TCAV set point, converts it to a T02 set point, and determines the best combination of flow control valve and heater settings to use over the prediction horizon.

Control with MPC has approximately a $5\times$ faster settling time than the existing feed-forward/PI controller, and it avoids the large overshoot seen in feed-forward/PI control. Avoiding overshoot will help to reduce the need to use excess RF overheat to compensate for detuning, thus potentially saving on operating costs. Adding RF overhead to amplifier specifications also is generally expensive (i.e. amplifiers with larger overhead are more expensive to purchase); thus, if advanced control methods can reduce the need for excess RF overhead, it could also impact the design budget of future accelerators.

Ideally, this set of studies would have continued on to create a full implementation of the MPC controller and examine additional algorithms (such as neural network-based reinforcement learning), as well as move toward designs which provide more direct control over the resonant frequency of the gun. Future work could be done in this arena. However, funding to do similar work on the PIP-II RFQ at Fermilab was obtained as a result of this initial study (necessitating a shift in focus to that application). PIP-II is a major high-power upgrade to the Fermilab accelerator complex, and thus was a high priority for the lab. That work is described in the next sections.

4.4 Resonant Frequency Modeling and Control for a High-Power RFQ

The Proton Improvement Plan II (PIP-II) is a major programmatic upgrade to Fermilab's main accelerator complex that will enable higher-power, higher-intensity proton beams to be delivered to particle physics experiments. In particular, Fermilab supports a number of neutrino experiments that need to maximize the number of protons per hour delivered by the accelerator that subsequently strike the graphite targets used to generated the neutrino beam. As the power and intensity of the proton beam is increased, complicated nonlinear dynamics effects (e.g. beam self-fields) make this more challenging. At high power, issues like stray beam (i.e. halo) around the main bunch can also damage components if the beam is not well controlled. To provide the power and intensity needed to open up new areas of study for neutrino and muon physics, PIP-II will require R&D in advanced accelerator technology, including improvements in control of high-power beams. Once built, PIP-II will also support the Long Baseline Neutrino Facility and the Deep Underground Neutrino Experiment (LBNF/DUNE) [212], as well as the muon-to-electron conversion experiment (Mu2e) [213]. These major experiments are expected to provide new insights into our understanding of fundamental particle physics.

At the time of this work (2015-2016), PIP-II was in the final stage of the design process, and an injector test stand, the PIP-II Injector Test, was being built and commissioned to provide proof-of-principle demonstrations of accelerator technology that will be used in PIP-II, including both hardware and software (e.g. advanced control of superconducting RF cavities, novel beam diagnostics, control of the high-power beam, and interlocks).

As part of the injector, a radio frequency quadrupole (RFQ) is used to simultaneously accelerate and focus an H- ion beam (see Figure 4.18 and Figure 4.19). Nominally, it accepts a 30-keV, 1-mA to 10-mA beam and accelerates it to 2.1 MeV. It is designed to operate at a frequency of 162.5 MHz with arbitrary duty factor, including both pulsed RF and continuous wave (CW) modes. The RFQ consists of a copper RF cavity with internal protrusions (i.e. the "vanes") to accelerate and focus

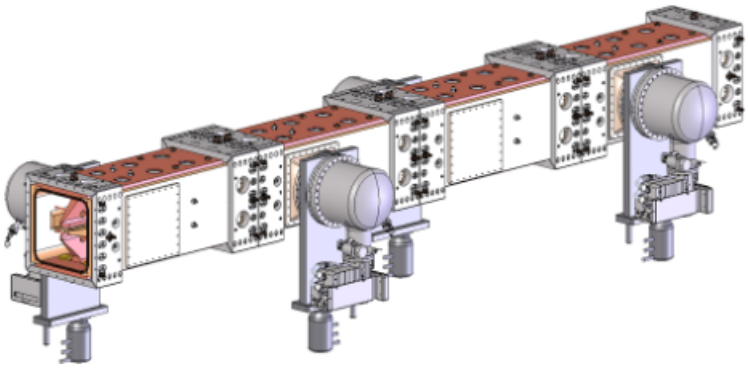


Figure 4.18: Solid model drawing of the RFQ (courtesy LBNL)

the beam (see Figure 4.20). Mechanical tuners can be manually adjusted to change the resonant frequency of the cavity. However, this is an involved process and cannot be done during operation. During operation, the resonant frequency is controlled entirely via thermal regulation, and cooling channels are built into the RFQ and attached to an externally-controlled water circuit to support this. The thermal expansion and contraction of the RFQ is leveraged to ensure that the desired resonant frequency is maintained, despite the heating of the cavity from RF power. The nominal RF power input to the RFQ is 100 kW, and the nominal cavity voltage is 60 kV. The nominal operating conditions are summarized in Figure 4.21.

Figure 4.19: The RFQ during installation.

One of the main challenges for operation is to maintain the specified field across the vane tips for beam acceleration by minimizing the cavity resonant frequency detuning at all times. Fine control over the detuning, minimal manual intervention, and fast trip recovery is desired. For

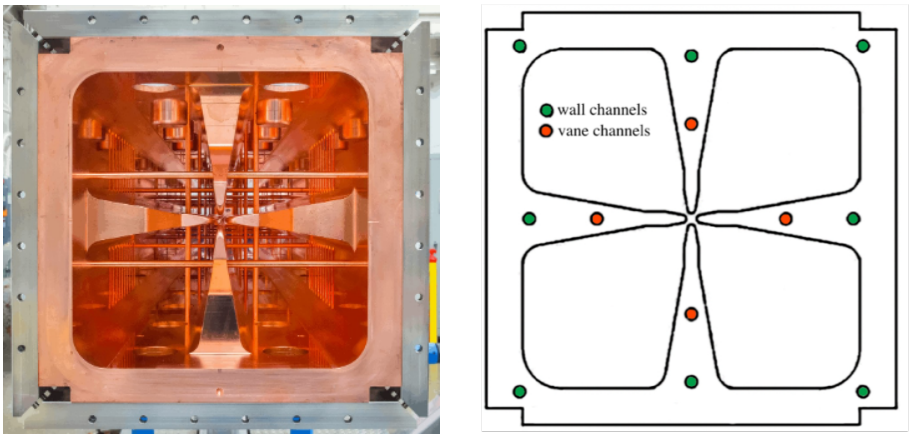


Figure 4.20: Profile cutaway of RFQ and drawing showing the locations of the vane and wall cooling channels.

PXIE RFQ Parameters	
RFQ Design Parameters	
RF frequency	162.5 MHz
Q-factor	~13,900
Loaded Q	~7,000
Physical Length	4.45 m (2.4 wavelengths)
Vane-to-Vane Voltage	60 kV
Estimated Power Dissipation	< 100 kW
RF Repetition Rate	pulsed – CW
Beam Parameters	
Current	0.5 – 10 mA (nominal 5 mA)
Input Energy	30 keV
Output Energy	2.1 MeV

Figure 4.21: The nominal operating parameters (as of 2015) of the RFQ.

example, the specifications for the RFQ operation state that the reflected power must be kept below 20% at all times in order to maintain cavity field with the RF amplifiers and avoid damaging the amplifiers. For trip recovery, the goal is to recover stable operation within 2 $\times$ the duration of the interruption, and the minimum specification is that it is within 10 $\times$ the interruption. Similarly, for startup of the RF (including to full RF power in CW operation), the goal is to take no longer than 10 minutes, and the minimum specification is that it takes no longer than 30 minutes. For an RFQ with high average-power RF, these are ambitious goals.

The available headroom of the RF amplifiers limits the maximum allowable detuning to just 3 kHz (at which point the forward power can no longer be increased to compensate for detuning). As the RFQ is expected to operate in both CW and pulsed RF modes with a wide range of duty factors, the expected RF heating will vary significantly, resulting in variable detuning of the cavity. The resonant frequency is particularly sensitive to changes in the vane pole tip positions resulting from both local and bulk heating (i.e. in the vanes and walls respectively). Because of their lower thermal mass, the vanes also contract and expand faster than the walls given the same heating/cooling power. Together, this results in a large nonlinear transient response in resonant frequency to changes in average RF power or disturbances from the water-cooling system that must be compensated for in order to stay below the maximum acceptable frequency shift of 3-kHz. With a Q-factor of 16,799, the increased amount of reflected power that results when the RFQ is slightly off-resonance is not severe in principle (e.g. compared to high-Q superconducting RF cavities), but the sensitivity of the RFQ to small thermal changes means that it can reach high levels of detuning very rapidly.

Based on the prior work with the RF gun at FAST (sponsored by the Office of Naval Research), we were then funded by Fermilab to design and commission a resonant frequency control system for the RFQ (including both software and hardware elements). We were also funded to help commission the RFQ operation itself. For example, the scope included reviewing the overall design of the exterior water-cooling system, reviewing and making key instrumentation choices, conducting system modeling to inform design and basic operating set-point decisions (e.g. to ensure both

pulsed and CW operation would be possible with the designed system), specifying and designing key communication interfaces for the software and hardware components of the resonant frequency control system, contributing to overall RFQ testing and commissioning, and, finally, implementing and testing the resonant frequency control system.

Each of these areas of scope was a critical component for addressing the technical challenges encountered in resonant frequency control for high-average-power RFQs, which are becoming increasingly of interest for high-power proton machines at the energy and intensity frontiers of physics. The results of this work can be used to inform and directly support the design and control of new RFQs designs, as well as support further control algorithm development and deployment on the PIP-II RFQ (e.g. through both associated software infrastructure and modeling approaches).

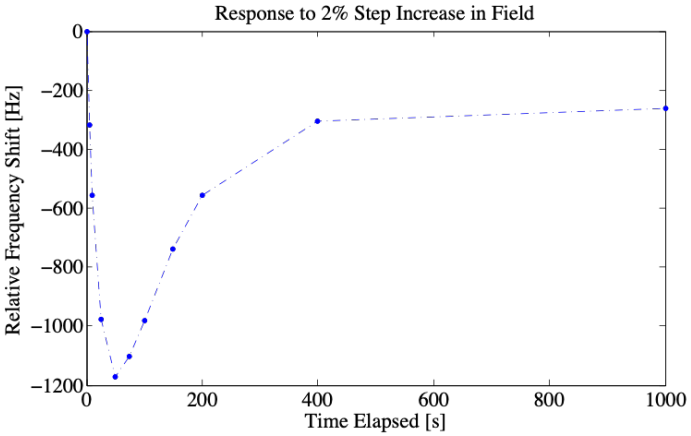
4.4.1 Description of the RFQ and Implications for Control

The RFQ was designed at Lawrence Berkeley National Laboratory (LBNL) [214]. It consists of a copper outer body (the walls) and four copper inner protrusions (the vanes) (see Figure 4.20). Four separate wall/vane modules along the length of the RFQ are bolted together to produce the entire structure, which is 4.46 m in length, 0.425 m in height, and 0.435 m in width (see Figure 4.18. Each module weighs approximately 1000 kg, and thus the RFQ has a large thermal mass. The temperature (and thus the exact geometry) of the walls and vanes are determined by water running through individual cooling channels, RF heating of the cavity, and heat dissipation to the environment from the exposed surface of the RFQ. For each module, there are 8 cooling channels in the walls and 4 in the vanes, as shown in Figure 4.20.

The resonant frequency is most sensitive to perturbations in the vane pole tip positions. As the body expands and contracts, it affects the pole tip positions along with any thermal expansion/contraction inherent to the vanes themselves. LBNL conducted initial 2-D thermal simulations in ANSYS [215] and found that over the data ranges examined, a -16.7-kHz shift is achieved per 1-°C change in the vane water temperature and a 13.9-kHz shift is achieved per 1-°C change in the wall water temperature. The common-mode shift is -2.80 kHz per 1-°C change in temperature.

Because of their lower thermal mass, the vanes contract and expand faster than the walls, producing a large dynamic transient under changing RF power or water-cooling temperature. Illustrating this, a simulated transient response to a small change in RF heating is shown in Figure 4.23. Existing requirements state that no more than a 3-kHz shift in resonant frequency is acceptable. At steady state, this corresponds to 0.1 °C stabilization in the water system. For the dynamic response, the transient needs to be mitigated through active control of the vane and wall water circuits.

At present, the resonant frequency of other RFQs is regulated with a PI loop around the vanes, while the walls are held constant [216, 217]. Independent control of the wall and vane circuits enables exploitation of the individual frequency responses to provide a wider frequency tuning range (i.e. through "differential tuning"), and the PIP-II Injector Test RFQ was designed with this functionality in mind by providing separate internal cooling channels across the cavity walls and



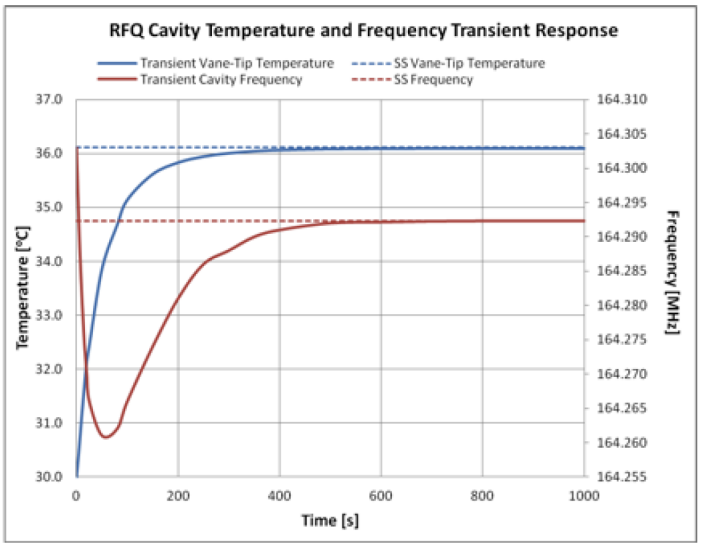


Figure 4.22: Simulated response in ANSYS of the RFQ to a 2 percent step increase in cavity field, provided by LBNL (courtesy Andrew Lambert, LBNL).

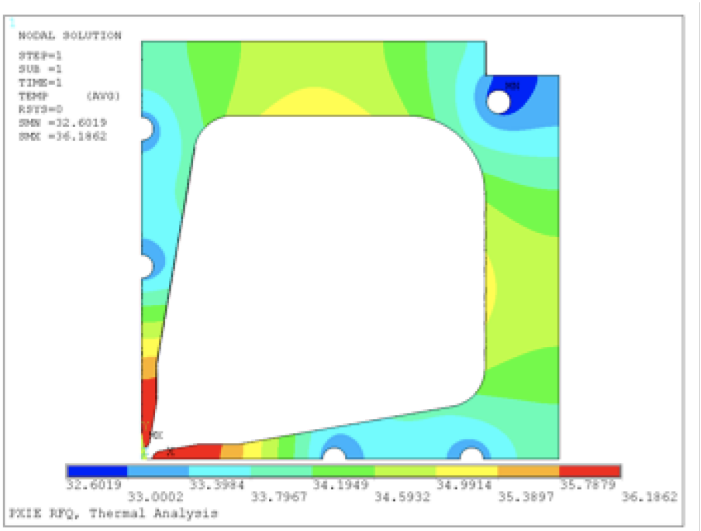


Figure 4.23: RFQ heating from the ANSYS model provided by LBNL (courtesy Andrew Lambert, LBNL) [215]. This shows the concentrated heating on the vane pole tips, which is responsible for the high degree of sensitivity of the resonant frequency to the variations in RF power.

the vanes. Coupled with a control algorithm that can handle adjustment of both the wall and vane valves in tandem (such as MPC), this should enable a very high degree of flexibility for running the RFQ.

4.4.2 RFQ Cooling System

Chilled water is supplied to the RFQ by two parallel loops: one for the inner (vane) channels and one for the outer (wall) channels, as illustrated in Figure 4.24. The channels are fed via an external cooling circuit that is supplied with temperature-regulated water. The external cooling circuit itself consists of water distribution, two pumps that maintain a constant flow of water into the RFQ (136-gpm for the wall and 64-gpm for the vane, see Figure 4.26), two flow control valves that regulate how much chilled water is mixed into each sub-circuit (see Figure 4.25), and an intermediate skid that regulates the average chilled water supply temperature. The intermediate skid has its own independent temperature control system that maintains the output temperature to within ± 0.28°C (± 0.5°F).

For each of the four individual modules comprising the RFQ, there are eight wall cooling channels and four vane cooling channels in the structure, and each module of the RFQ has its own exterior water distribution manifold (consisting of many parallel pipes, as shown in Figure 4.27). Temperature sensors, pressure sensors, and flow meters are outfitted throughout the system. A portion of the return water is mixed back into the intermediate skid loop, for pressure balancing in the system.

The chilled water from the skid mixes with warm water returning from the RFQ. At the mixing point, in-line helical mixers ensure that the mixing is thorough. The vane and wall sub-circuits are thus not completely independent, as they coupled through pressure balancing in the system, mixing in the skid return line, and heat flow through the RFQ itself.

Critically, the water system needed to be designed in such a way (both in terms of layout and set point temperatures and flow rates) to supply sufficient cooling capacity to keep the cavity at the desired resonant frequency over a range of pulsed and CW operating conditions. Fermilab has a dedicated group for the design and construction of water-cooling systems. However, as is the case at many accelerator facilities, these systems are not often designed with high-performance control over parameters of interest like the resonant frequency in mind. Additional constraints, such as limited space inside radiation shielding enclosures, impact the water-cooling system design. This

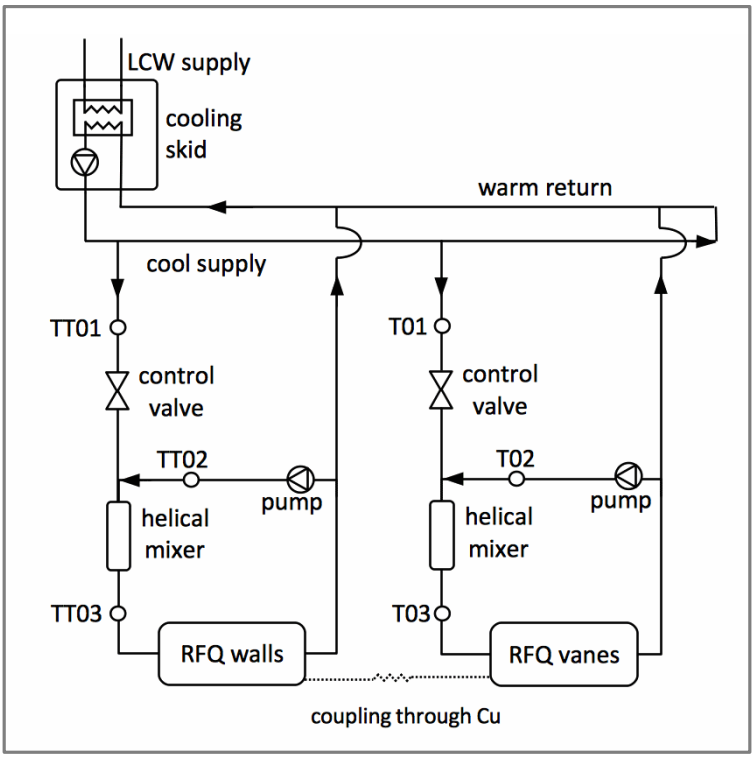


Figure 4.24: Simplified PIP-II RFQ cooling system diagram. TT01, T01, et cetera are temperature sensors. For illustration purposes, we have lumped the different modules of the RFQ together and displayed the water distribution manifolds as a single supply (after TT03 and T03) and return (before TT02 and T02) to the RFQ. It is important to note that the system is subject to transport delays for the return water and for the supply water due to the pump and supply lines being located outside the cave. The helical mixers are located just before the distribution manifold to minimize the transport delay between the water mixing point and the RFQ. There are six locations where high-resolution temperature sensors suitable for the use in the resonance control system are included. These sensors are located at the chilled water supply point (TT01 and T01 for the walls and vanes respectively), on the return line (TT02 and T02 for the walls and vanes respectively), and after the mixing points (TT03 and T03 for the walls and vanes respectively).

sometimes leads to transport delays when placing cooling skids outside of the radiation shielding. It also can lead to the introduction of thermal disturbance sources (e.g. when cooling pipes are placed near external doors, or near equipment inside the radiation enclosure that produces a lot of waste heat). This is one of many examples where systems engineering best practices (which take the wider interactions between machine design considerations into account) are essential.

Because of the tight tolerances on the resonant frequency, we were consulted on the initial design provided by the water group. We reviewed the water system and constructed a simulation model of the RFQ and cooling system (discussed later) to ensure the proposed design would be

Figure 4.25: Flow valves for the wall and vane cooling circuits (yellow).

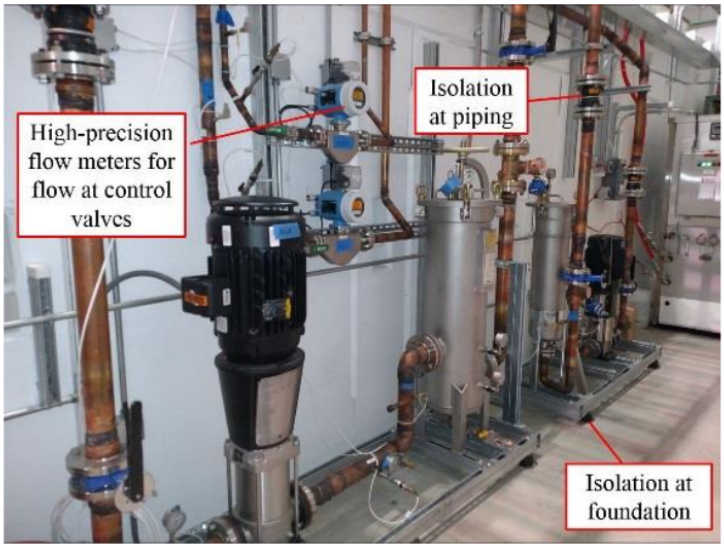


Figure 4.26: Pumps on vane and wall circuits for circulating water through the RFQ, courtesy Curt Baffes.

acceptable for RFQ resonant frequency control. Specifically, ensuring that the RFQ could be operated both in pulsed mode (i.e. including low levels of average RF heating) and in CW mode (i.e. including high levels of average RF heating) was a key goal. For example, the flow control valves have nonlinear responses (that is, the amount of change in flow achieved per change in aperture varies non-linearly), and thus having sufficient dynamic range in the valves to still have fine control in these disparate operational modes was a key issue. Another specific goal was to

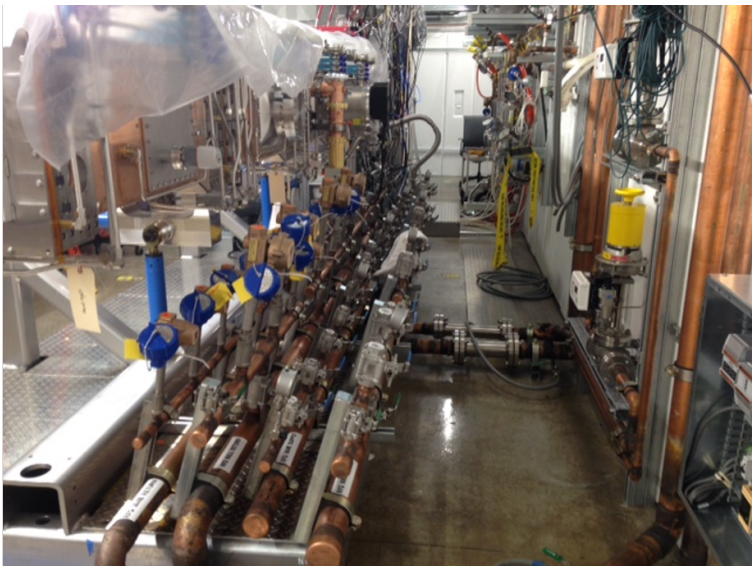


Figure 4.27: Distribution manifold for sending chilled water through the RFQ at multiple entry points (shown prior to final hosing installation.)

avoid some of the problems encountered with the design of the water system at FAST, such as transport delay and poor sensor resolution relative to the required control specifications.

We constructed an analytic model (described in the next section) of the combined RFQ and water-cooling system prior to the arrival of the RFQ at Fermilab. We then used this model to help assess the water system design specifications and ensure that the desired level of control over the resonant frequency could in principle be met for both pulsed and CW operation. It was also used to determine corresponding nominal set points for the water system (e.g. required flow rates through the RFQ, water system temperature set points, etc). Related to this, it was also used to inform specifications for the equipment, including some which had not yet been purchased (e.g. chiller specifications for handling heat load, pump flow rate specifications and flexibility for ensuring proper flow rates could be maintained, flow valve control accuracy, and temperature sensor resolution requirements).

Several adjustments to the water system were suggested and either approved or rejected based on discussions with the Fermilab members of the resonance control group and the water group, including:

- Moving the mixing point of the warm and cold water as close as possible to the RFQ to mitigate transport delay *(accepted)*;
- Adopting helical in-pipe mixers rather than a mixing chamber to further reduce system delays *(accepted)*;
- Replacing the previously-chosen temperature sensors and data acquisition units to enable the required accuracy for control *(accepted)*;
- Adding a pressure balancing valve to enable more constant flow through the system when the vane and wall valves are adjusted *(accepted but delayed)*;
- Insulating the copper pipes that carry the water *(rejected -to be revisited if need be)*;
- Decoupling of the vane and wall water circuits on the return line *(rejected - more expensive to re-route)*;
- Adding another flow control valve on each of the return lines to enable finer control over the mixing ratio of warm and cool water *(rejected - too expensive)*;
- Including a heater in the system to enable the flow control valves to be operated more flexibly in both CW and pulsed mode (i.e. to be cold enough at CW, and to be able to warm up the water for pulsed mode, thus reducing the need to operate at the extremes of the flow control valves) *(rejected - too expensive)*.

Most of the data from the water system is acquired through a programmable logic controller (PLC). This was only capable of providing a digital temperature resolution of 0.1 °C, which based on error propagation calculations was deemed unacceptable. As such, we proposed using a Cryocon model 18i for data acquisition for several key temperature sensors. This setup worked well and has since been adopted by other groups at the lab when high-resolution temperature sensors are needed.

Two of the changes that were not made had a significant impact on subsequent operation. The variation in ambient temperature (especially for pipes running outside of the accelerator enclosure)

result in disturbances to the system supply temperature, particularly as these are near to external doors. In addition, the additional coupling in the system between the vane and the wall water circuits hampers the ability to do differential tuning. The avoidable coupling comes from the water mixing in the joint return line and the lack of a pressure balancing valve (which results in changes in one valve setting affecting flow in the other circuit). These changes could still be made in the future if needed to more fully exploit differential tuning and improve the tuning range of the RFQ.

4.4.3 Analytic Modeling

Multi-physics modeling codes and 2-D or 3-D finite element analysis simulations are often used to simulate the thermal behavior of individual accelerating cavities, including RFQs [215, 218–221]. However, this modeling approach is too computationally expensive to be effective when studying dynamic control of large, distributed systems with many interdependent components. In the design of accelerator RF cavities, detailed thermal modeling usually is not done for the entire combined cooling and RF cavity system. Rather, it is typically done for just the cavity itself (with, for example, steady state cooling and heating sources). The ability to study dynamic temperature and frequency shifts at the system level with a reasonable degree of accuracy is paramount to the design of control algorithms.

For the RFQ, it was anticipated that conducting such simulations would be critical for ensuring the RFQ would be able to operate in pulsed and CW operation. To this end, an analytic model was constructed to help inform the design of the RFQ water system, ensure the RFQ could be operated in both pulsed and CW mode, and aid controller design choices for the combined RF and cooling system. The model needed to be able to run efficiently over the characteristic setting time response of the RFQ, and at the time of model construction this was estimated to be around 30-40 minutes. In order to construct a model that was computationally efficient enough to simulate these long timescales, a simplified physics model was needed. However, model accuracy was also a priority, and so the interactions between all major sub-components of the model needed to be included. To fulfill these needs, the thermal dynamics in the RFQ and the dynamics of the cooling system were both approximated. This model and modeling approach could be used for future design and control studies for RFQs and other water-regulated copper accelerating cavities. The model includes:

- All major layout considerations (pipe diameters and lengths, mixing points, etc).
- Heat generation from the pumps and friction from water flow
- Thermal losses from the pipes to air, and from the RFQ surface to air; air temperature is flexible

- Nonlinear flow curves from the valves
- The impact of cooling power on the vanes and walls individually
- The impact of RF heating on the vanes and walls individually
- Thermal coupling between the vanes and the walls through the RFQ copper body
- The combined resonant frequency shift induced by the vane and wall responses to the water system cooling and RF system heating powers

By including each of these components, the dynamic response of the system to changes in the flow control valves and in RF heating can be examined, along with resonant frequency shift induced in the cavity. In addition, the model is easily modified at each interaction point, which should facilitate shifting to a different system design or to calibrate the model to measurements (e.g. by using updated coupling or heat transfer coefficients).

Details of the Analytic Model

For the cooling system, the individual water distribution manifolds are combined into two supply and return pipes (i.e. one set for the vane and one set for the wall). This sacrifices some accuracy: water entering the RFQ at upstream manifold sections has an earlier thermal impact on the RFQ than later ones, resulting in small resonant frequency transients. However, making this assumption simplifies the transport and heat transfer models substantially. In addition, it is assumed that the delays in the plumbing manifold are small compared with the long transport delays associated with the distribution system. In the model, the pumps on the cooling skids completely compensate for changes in the chilled supply flow, as designed; in reality, this compensation is not perfect. Finally, it is assumed that the helical mixers perform ideal mixing between the chilled water and the return water (a reasonable assumption).

For the RFQ model, the geometry of the walls and vanes were simplified into two individual thermal capacitances. A coupling term is then added between the walls and the vanes to account for the heat transfer between them through the copper body of the RFQ. This ignores localized

effects on individual vanes and walls but should capture enough detail to predict temperature and resonant frequency shifts of the RFQ with reasonable accuracy. Estimates of the parameters used in this model are described later. This approach builds on more basic lumped thermal capacitance models described in [222–224].

Thermal response of the RFQ: The classical lumped capacitance [222] for an RF cavity is given as:

$$\begin{aligned} T_{\mathrm{out}}(t) =& T_{\mathrm{initial}} + \frac{1}{C}\int_0^t P_{\mathrm{rf}}(t)dt \\ & - \frac{1}{C}\int_0^t \left(\frac{(T_{\mathrm{out}}(t) - T_{\mathrm{in}}(t))\dot{V}}{A}\right) dt. \end{aligned} \tag{4.2}$$

Here, $T_{\mathrm{out}}(t)$ is the temperature of the water leaving the cavity, and we assume that it is approximately equal to the temperature of the cavity itself. C is the thermal capacitance [$\mathrm{J}/^{\circ}\mathrm{C}$] of the cavity, $P_{\mathrm{rf}}(t)$ is the RF heating, $T_{\mathrm{in}}(t)$ is the temperature of the water at the input to the cavity cooling channel, $\dot{V}$ is the volume flow rate of the water in the cavity, and A is the heat carrying capacity of water (3.814 C /gpm-kW). The time-dependent temperature of the walls and vanes in the RFQ are modeled using this relationship, as described below. A coupling term that accounts for heat transfer between the two subsystems through the copper and a term to account for thermal losses to the environment from the walls are also added. The thermal model of the vanes thus becomes:

$$\begin{aligned} T^{\mathrm{v}}_{\mathrm{out}}(t) =& T^{\mathrm{v}}_{\mathrm{initial}} + \frac{1}{C^{\mathrm{v}}}\int_0^t P^{\mathrm{v}}_{rf}(t)dt \\ & - \frac{1}{C^{\mathrm{v}}}\int_0^t \left(T^{\mathrm{v}}_{\mathrm{out}}(t) - T^{\mathrm{w}}_{\mathrm{out}}(t)\right) K_1 dt \\ & - \frac{1}{C^{\mathrm{v}}}\int_0^t \left(\frac{(T^{\mathrm{v}}_{\mathrm{out}}(t) - T^{\mathrm{v}}_{\mathrm{in}}(t))\dot{V}^{\mathrm{v}}}{A}\right) dt \end{aligned} \tag{4.3}$$

where $T^{\mathrm{v}}_{\mathrm{out}}$ is the temperature of the water at the output of the vane channels, C^{v} is the thermal capacitance of the vanes, P^{v}_{rf} is the heating of the vanes due to the incident RF power, $T^{\mathrm{v}}_{\mathrm{in}}$ is the water input temperature to the vane cooling channels, $T^{\mathrm{w}}_{\mathrm{in}}$ is the water input temperature to the wall cooling channels, K_1 is a coefficient that describes the coupling between the wall and vane water circuits through the copper body of the RFQ, $\dot{V}^{\mathrm{v}}$ is the volume flow rate of the water in the vane cooling channels, A is the heat carrying capacity of water, and $T^{\mathrm{v}}_{\mathrm{initial}}$ is the initial temperature of the vanes at the start of the simulation.

The thermal model of the walls (with many of the same terms as for the vanes) is given by:

$$\begin{aligned} T^{\mathrm{w}}_{\mathrm{out}}(t) =& T^{\mathrm{w}}_{\mathrm{initial}} + \frac{1}{C^{\mathrm{w}}} \int_0^t P^{\mathrm{w}}_{\mathrm{rf}}(t)dt \\ &- \frac{1}{C^{\mathrm{w}}} \int_0^t \left(T^{\mathrm{w}}_{\mathrm{out}}(t) - T^{\mathrm{v}}_{\mathrm{out}}(t)\right) K_1 dt \\ &- \frac{1}{C^{\mathrm{w}}} \int_0^t \left[\frac{(T^{\mathrm{w}}_{\mathrm{out}}(t) - T^{\mathrm{w}}_{\mathrm{in}}(t))\dot{V}^{\mathrm{w}}}{A}\right] dt \\ &- \frac{1}{C^{\mathrm{w}}} \int_0^t \left(T_{\mathrm{air}}(t) - T^{\mathrm{w}}_{\mathrm{out}}(t)\right) K_2 dt \end{aligned} \tag{4.4}$$

The last term describes the heat transfer between the RFQ walls and the environment via the coefficient K_2. T_{air} is the air temperature. This is only included for the wall component because the vanes are not in direct contact with the environment.

Using the above relationships, the resonant frequency shift of the RFQ is calculated as:

$$\begin{aligned} \Delta f_0(t) = (T^{\mathrm{w}}_{\mathrm{out}}(t) - T^{\mathrm{w}}_{\mathrm{initial}})\frac{\Delta f_0}{\Delta T^{\mathrm{w}}} \\ +(T^{\mathrm{v}}_{\mathrm{out}}(t) - T^{\mathrm{v}}_{\mathrm{initial}})\frac{\Delta f_0}{\Delta T^{\mathrm{v}}} \end{aligned} \tag{4.5}$$

Here $\frac{\Delta f_0}{\Delta T^{\mathrm{w}}}$ is the frequency shift per °C change in the wall temperature, and $\frac{\Delta f_0}{\Delta T^{\mathrm{v}}}$ is the frequency shift per °C change in the vane temperature. This group of equations represents an analytic thermal model for the RFQ body that can be combined with models for the water transport and

mixing to study system-level behavior (these are described next).

Heating due to friction: Additional heating of the water occurs throughout the system due to friction during transport, particularly in places where pressure drops occur. Here, the heating is approximated as a single term applied at the water pump:

$$\Delta T = \frac{P_{\text{pump}}}{c_{\text{water}} \dot{m}} \tag{4.6}$$

Here ΔT is the temperature change of the water due to frictional heating, P_{pump} is the pump power, c_{water} is the specific heat of water, and $\dot{m}$ is the mass flow rate of water through the pump. The pump power is calculated using the pressure drop across the pump multiplied by the flow rate of the water through the pump.

Heat transfer in the pipes: Heat transfer from the pipes to the environment (in this case, air in the accelerator enclosure) is modeled as a temperature change over a length of pipe for a given external air temperature:

$$T_{\text{out}}(t - t_d) = T_{\text{air}}(t) - (T_{\text{air}}(t) - T_{\text{in}}(t))\, e^{\left(\frac{-k' L}{r_{\text{i}}^2 c \rho v}\right)}. \tag{4.7}$$

Here T_{out} is the temperature of water after traversing the length of pipe, T_{air} is the air temperature, T_{in} is the temperature of water entering the section of pipe, k' is the effective thermal conductivity, L is the length of the pipe, c is the specific heat of water, ρ is the density of water at room temperature, r_i is the inner radius of the pipe, and v is the velocity of water in the pipe. The effective thermal conductivity is given by $k' = kr'\,(2hr_0/(2hr_0 + kr'))$, where r' is the effective pipe thickness. This is given by $r' = (r_0 + r_{\text{i}})/(r_0 - r_{\text{i}})$, where r_0 is the outer radius of the pipe. h is the convective heat transfer coefficient of stagnant air, which is typically 10-25 W/m^2 K. For the

RFQ mode, 17 W/m^2 K is used. The time delay t_d of the transport section is calculated using the volume flow rate and the pipe area, with the assumption that the velocity of the water in the pipe is constant.

Mixing: Because the RFQ water system uses helical mixers specifically designed to provide close to ideal mixing, this is a reasonable approximation. Ideal mixing of the cold and warm water is given by:

$$T_\mathrm{out} = \frac{T_1\dot{V}_1 + T_2\dot{V}_2}{\dot{V}_1 + \dot{V}_2} \tag{4.8}$$

Here T_1 and T_2 are the two input temperatures, and $\dot{V}_1$ and $\dot{V}_2$ are the two input volume flow rates.

Application to the PIP-II RFQ cooling system Using the equations, a model of the RFQ and water system was constructed using `Matlab`'s `Simulink` toolbox [225], which was later translated into Python code for portability. Figure 4.28 shows a block diagram of the model. Note that a superscript w or v denotes the wall or vane circuit.

In the block diagram, $T_\mathrm{out}(t - t_d)$ is the temperature of the water leaving a block; the output temperature is delayed by t_d in transport sections. $T_\mathrm{in}(t)$ is the temperature of water at the input to a block, $T_\mathrm{air}(t)$ is the ambient air temperature, $\dot{V}(t)$ is the volume flow rate of water entering the block, P_rf is the effective RF heating of the walls or vanes, T_initial is the initial temperature of the RFQ, L_pipe is the length of pipe for a transport section, D_pipe is the outer diameter of the pipe for a transport section, T_pipe is the pipe thickness for a transport section, and P_pump is the power of the pump.

The system parameters used for the model are given in Table 4.4. The thermal capacitances for the vane and wall circuits were determined by estimating the mass of the vanes and walls using the machining drawings for the RFQ. The pump heating powers were calculated from the nominal pressure drop and volume flow rate through the pumps. The frequency shift coefficients were obtained from 2-D ANSYS results [215]. The heat transfer coefficient K_1 between the vane and walls was estimated using a 1-D heat transfer model between the vane cooling channel and the

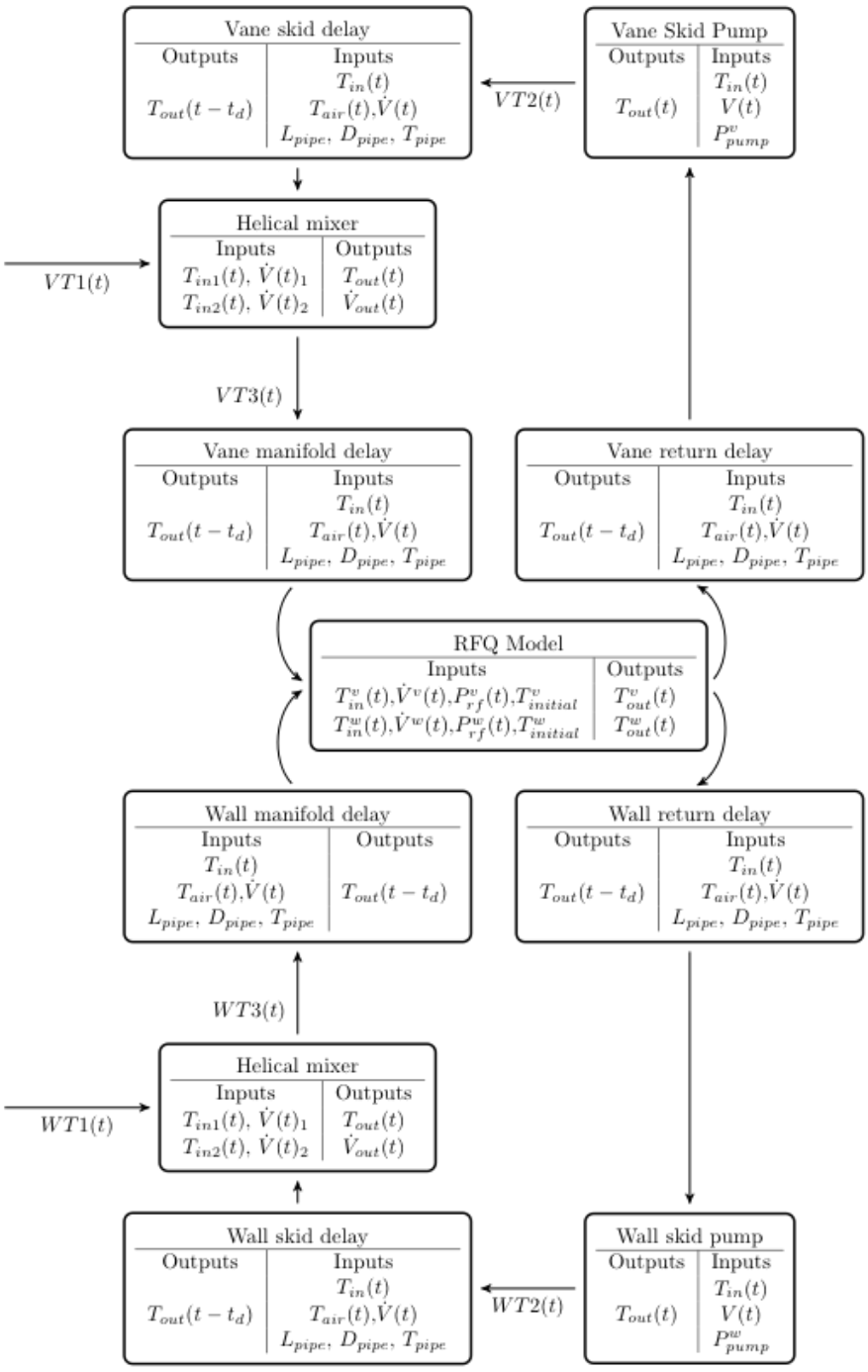


Figure 4.28: Block diagram of the water system model. Each block represents a system component that is modeled in Simulink. Using the inputs at each time-step the simulation calculates the water temperature at each location in the cooling system. The temperature of the RFQ model is used to calculate frequency shifts.

Table 4.4: *A priori* model parameters for the PIP-II RFQ.

Model parameter	Estimate	Units
C^{v}	160	kJ/°C
C^{w}	1430	kJ/°C
K_1	0.943	kW/°C
K_2	0.073	kW/°C
$P^{\mathrm{w}}_{\mathrm{pump}}$	2.9	kW
$P^{\mathrm{v}}_{\mathrm{pump}}$	1.6	kW
$\frac{\Delta f_0}{\Delta T^{\mathrm{w}}}$	13.9	kHz/°C
$\frac{\Delta f_0}{\Delta T^{\mathrm{v}}}$	-16.4	kHz/°C
Vane total flow	89	GPM
Wall total flow	167	GPM

nearest wall cooling channel; in doing so, the assumption is made that most of the heat transfer occurs between each vane cooling channel and the nearest wall cooling channel. The conductive heat transfer equation is given by:

$$Q_{\mathrm{cond}} = \frac{k_{\mathrm{cu}} A \Delta T}{d} \tag{4.9}$$

Here k_{cu} is the thermal conductivity of copper in [398 W / (m K)], A is the heat transfer area in [cm^2] estimated using the area of the walls in direct contact with the vanes (~ 0.192 m^2 over the whole RFQ), and d is the distance between the vane and wall cooling channels ($\sim$ 0.081 m). This gives a coupling coefficient of 943 W/ °C.

The coupling coefficient between the RFQ and the environment, K_2, is computed using the following equations to represent the conductive, radiative, and convective heat transfer rates respectively. For these calculations the surface area for the RFQ was estimated using the bulk RFQ dimensions from mechanical drawings ($A \sim 9$ m^2).

For the conductive heat transfer coefficient, Q_{cond}, given by:

$$Q_{\mathrm{cond}} = \frac{k A \Delta T}{d} \tag{4.10}$$

k is the thermal conductivity of air ($\sim 26.3 \times 10^{-3}$ W/m°C), ΔT is the temperature differential between the RFQ and the air in the building, and d is the effective distance for conduction (~ 1 m). This gives a heat transfer coefficient for conductive losses of ~ 2.38 W/°C.

For the radiative heat transfer coefficient, Q_{rad}, given by:

$$Q_{\mathrm{rad}} = A\epsilon\sigma(T^4 - T^4_{\mathrm{ambient}}) \tag{4.11}$$

ϵ is the emissivity (estimated at 0.78 due to oxidation [226]), and σ is the Stefaan-Boltzmann constant. This gives a heat transfer coefficient of $\sim$ 443 W when the RFQ is at 35 °C and the ambient temperature is at 25 °C. Because $(T^4 - T^4_{\mathrm{ambient}})$ is approximately linear for relatively small changes in temperature ($\pm$10°C), the radiative heat transfer coefficient can be approximated as 44.3 W/°C. While this approximation is not necessary for such a simple calculation, it significantly reduces the complexity of the implementation in Simulink.

For the convective heat transfer rate between the RFQ and the atmosphere, Q_{conv}, given by:

$$Q_{\mathrm{conv}} = h_c A \Delta T \tag{4.12}$$

the four different surfaces of the RFQ need to be treated separately in order to account for their different convection coefficients. The convection coefficient can be described by, $h = \mathrm{Nu_L} k/L$, where $\mathrm{Nu_L}$ is the Nusselt number, k is the thermal conductivity of air, and L is the effective length of convection. As the Nusselt number varies significantly for hot surfaces facing up vs. hot surfaces facing down, approximate values for different scenarios given by [226] were used to guide the estimates used in the RFQ model. Given the approximate Nusselt numbers, the convective heat transfer coefficient for the upper surface and the lower surface are estimated as $h_{\mathrm{t}} = 4.2$ and $h_{\mathrm{b}} = 1.56$ respectively. For the two sides of the RFQ the convection coefficient was estimated to be $h = 2.97$. Adding the different convective heat transfer rates for the RFQ gives a net heat transfer coefficient between the RFQ and the atmosphere due to convection of $\sim$ 26.4 W/°C. Adding each

of the derived heat transfer coefficients gives a total estimated coupling coefficient between the RFQ and the environment of $\sim$ 73 W/°C.

Model sensitivity analysis

The parameters that are used in the model rely on approximations. In order to understand which parameters most significantly affect the model predictions, the derivative of the ratio of the RMS error to the range with respect to a change in each of the parameters was computed, as shown in Table 4.4. Figure 4.29 and Figure 4.30 show the sensitivity of each error source to each of the parameters in Table 4.4 for pulsed and CW RF operation, respectively. Figure 4.31 shows the impact that removing different components of the model entirely has on the temperature predictions.

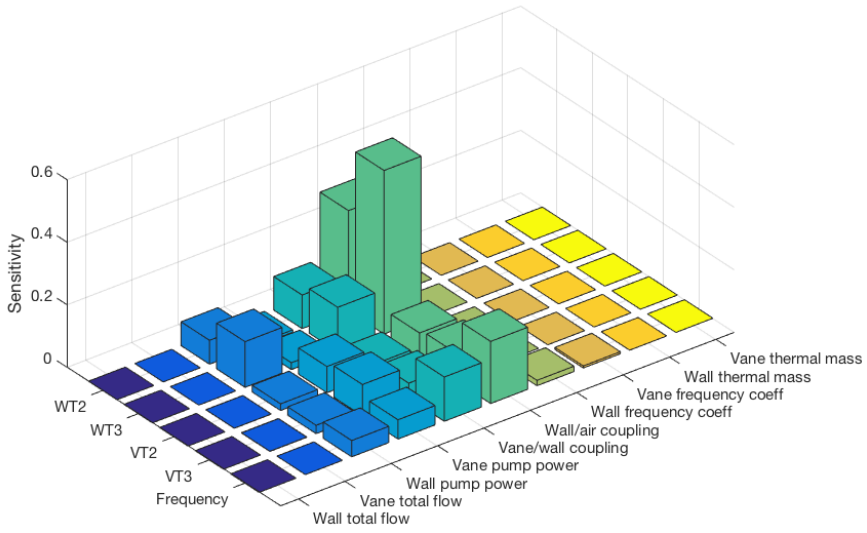


Figure 4.29: Absolute sensitivity of the model to 20% perturbation in the parameter during pulsed operation

Here we see that in general the model is most sensitive to perturbations in the coupling coefficients while the model is very insensitive to changes in the thermal mass and the flow rates. It is also interesting that the model is not very sensitive to the frequency coefficients for the vanes and walls. It is likely that in the current configuration of the model that the error in the coupling coefficients dominates the errors in the predictions.

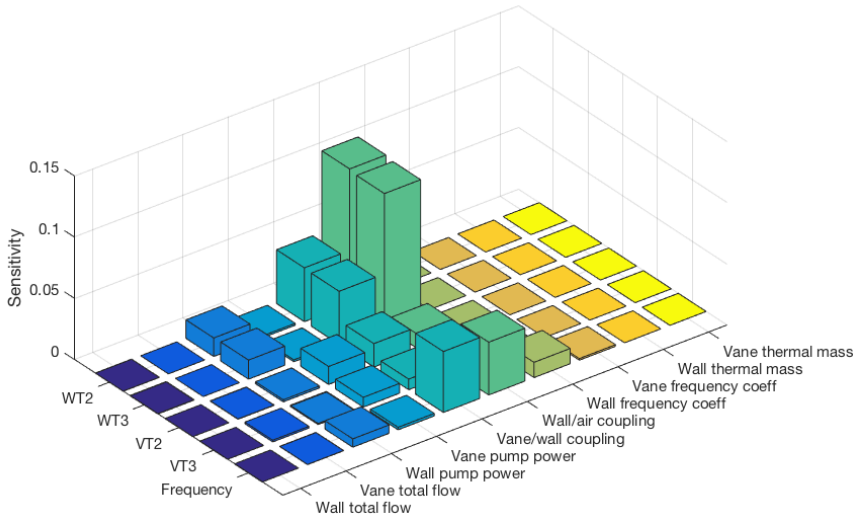


Figure 4.30: Absolute sensitivity of the model to 20% perturbation in the parameter during CW operation

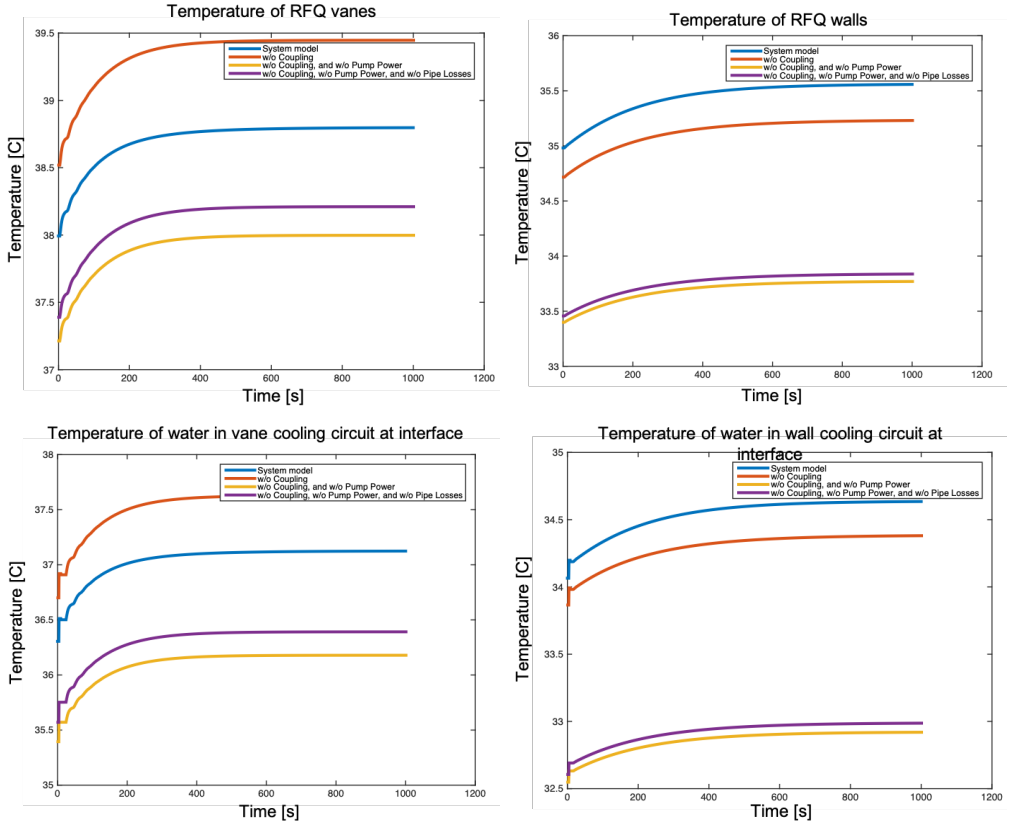


Figure 4.31: Impact on temperature prediction when different components of the model are removed.

Experimental Validation of the Analytic Model

To assess this modeling approach, the *a priori* analytic model was compared to measurements obtained once the RFQ arrived and the water system was commissioned. Note in this section, to make the correspondence to the wall and vane circuits more clear, TT01-TT03 (per Figure 4.24) are referred to as W1 - W03 and T01 - T03 are referred to as V1 - V3.

Low Average RF Power Studies: During low-average-power testing at the start of RFQ commissioning, the temperature and frequency response of the RFQ and water system was examined in three stages. This was done to assess different aspects of the model (e.g. prediction of water temperatures, valve responses, and responses to RF heating). First, only two RF power levels were examined with no changes in the flow control valve settings. Second, two RF power levels with 20 combinations of wall and vane flow valve settings were examined. Third, six RF power levels with five combinations of flow valve settings were examined.

Throughout these studies the RF system was operating in pulsed mode with a pulse length of 4-ms at a 10-Hz repetition rate. Data was collected at a 1-Hz rate on RF parameters, chilled water flow, ambient temperature, and water temperature throughout the system. For each of the studies the error metrics used are the root mean squared (RMS) error between the simulation and measurements, the maximum error between the simulation and measurements, the peak-to-peak temperature change during the study (temperature range), and the ratio of the RMS error to the temperature range.

The resonant frequency is calculated from the measured RF signals, as follows:

$$\delta f_0 = \frac{\tan\left(\phi_{\mathrm{cav}} - \phi_{\mathrm{fwd}}\right)}{2Q_{\mathrm{L}}} f_0 \tag{4.13}$$

Here, f_0 is the drive frequency, ϕ_{fwd} is the forward phase, ϕ_{cav} is the cavity phase, Q_{L} is the loaded quality factor, and δf_0 is the shift in resonant frequency. For these studies the drive frequency was 162.465 MHz, and Q_{L} was 6900. Because the RFQ is driven by two amplifiers, the

forward phase is determined by taking the vector sum of the two drive signals at the input to the cavity.

Figure 4.32 shows inputs to the model for changes in forward RF power level and flow valve settings. This includes the RF power settings, the chilled water flow settings, and the ambient and supply temperatures. Note that fast drops in the RF power were trips caused by reflected power. There are also significant fluctuations in the cave temperature as well as the chilled supply temperature, as shown. Figure 4.33 shows the comparison between the model predictions and the measurements of resonant frequency changes. Here we see a correlated increase in the error as a function of time. Upon inspection, it is clear that this is due to discrepancies between the model and the measurements in predicting changes due to the flow valves (e.g. due to inaccuracies in the flow response curve to different settings). The difference between the simulated and real flow response curves could easily be due to the lack of a pressure balancing valve, which, as described earlier, results in the settings for one valve affecting the flow rate through the other valve. Figure 4.34 shows a comparison between the simulated and measured data for the temperature sensors.

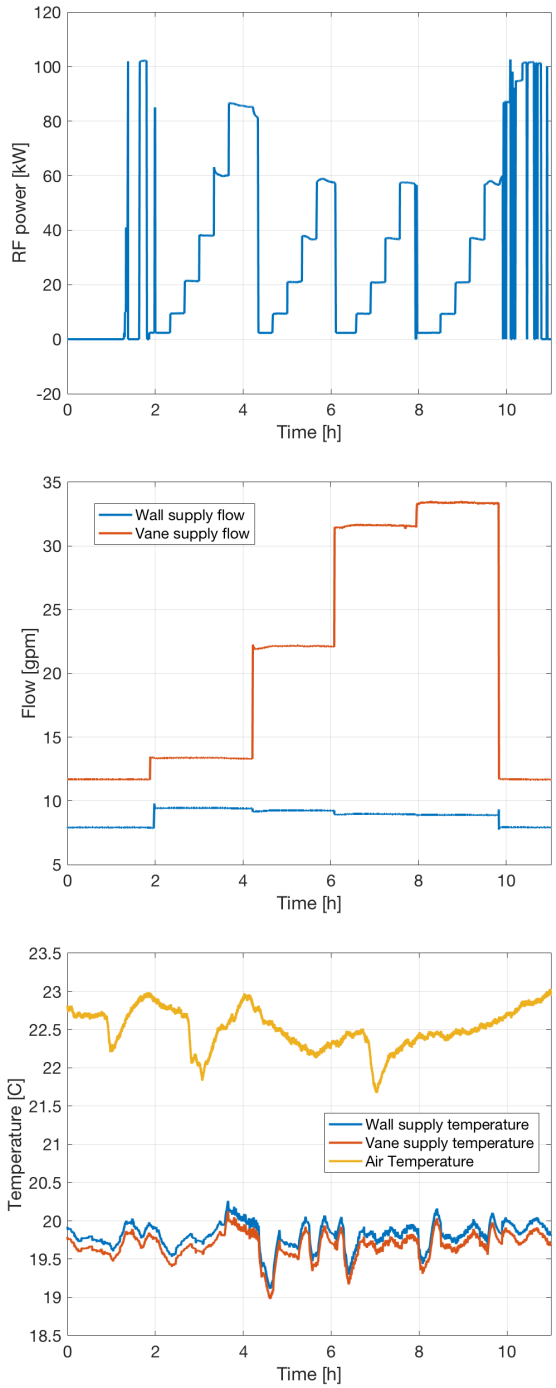


Figure 4.32: Inputs to the model for the third case study in pulsed mode; both the RF amplitude and the flow valves were scanned.

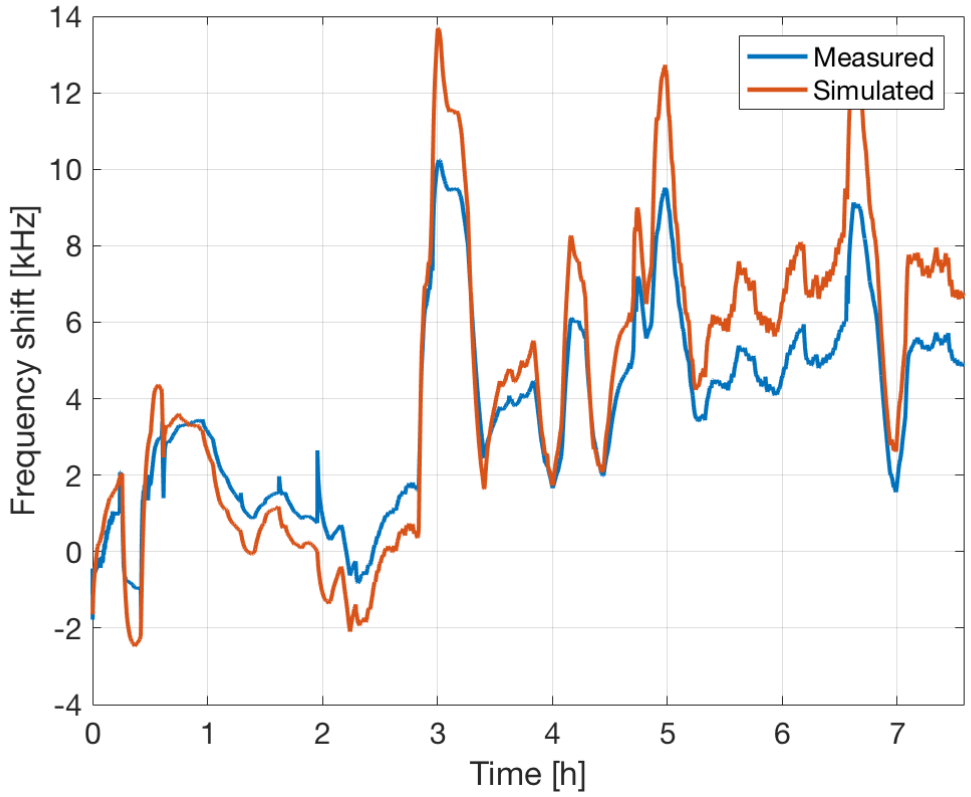


Figure 4.33: Comparison of measured and simulated frequency shift versus time for a case where both the RF amplitude and the flow valves were scanned.

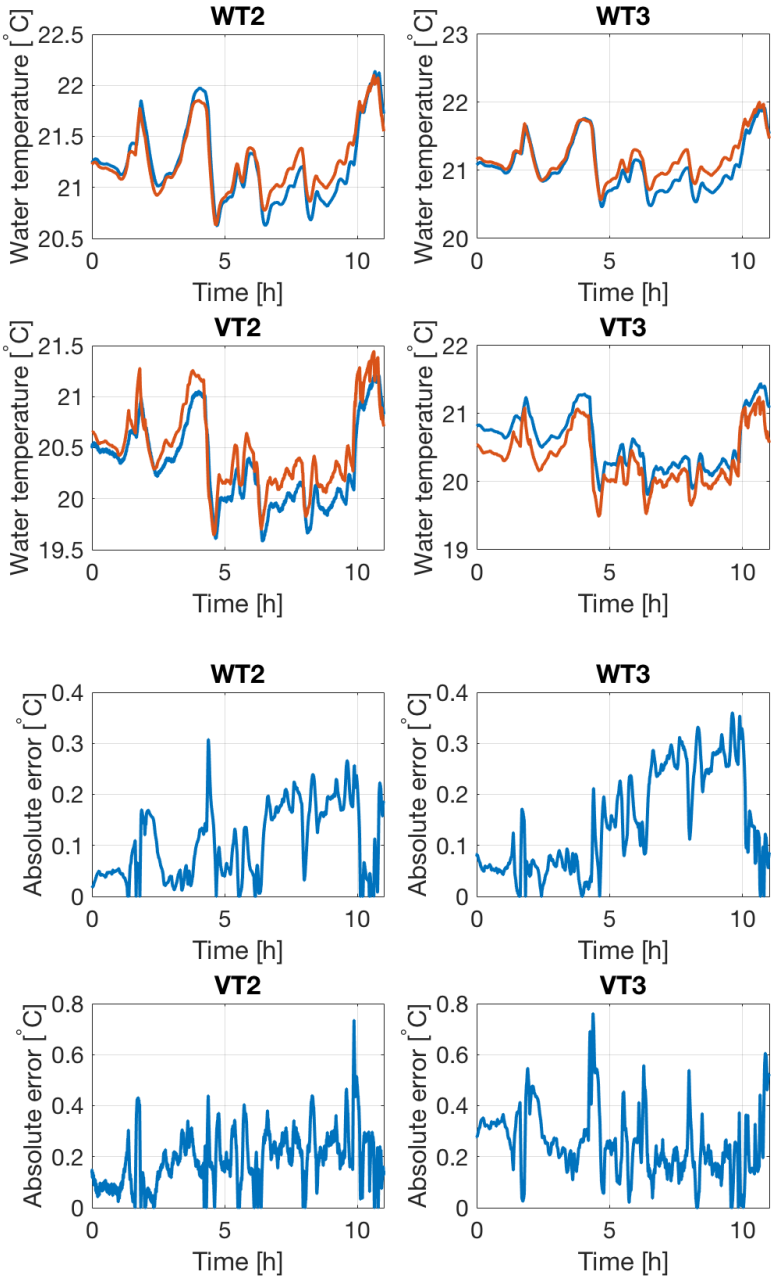


Figure 4.34: Comparison between measured and predicted water temperatures at low power; both the RF amplitude and the flow valves were scanned.

Table 4.5 - Table 4.7 show the RMS error, peak error, temperature range, and ratio of the RMS error to the temperature range for all three low-power studies. These results show that the model can be used to simulate absolute temperature transients in the system to within 20%. The RMS error in all cases was below 0.32 °C. Additionally, for all studies the ratio of the RMS error to the temperature range was below 0.17 °C. Of the four temperature sensors, the prediction at VT3 consistently has higher error than predictions at the other sensors (it is consistently underestimated in the simulation). This is likely due to a known calibration issue in the sensor, which creates an offset to the data that is not captured by our model. The standard deviation of the error, however, is less than or equal to 0.15 °C across all data sets. This indicates that relative changes in temperature are predicted with a higher degree of accuracy. Table 4.8 and Table 4.9 show the standard deviation of the error and the ratio of the standard deviation of the error to the temperature range for all four sensors across all three tests.

Examining the resonant frequency predictions, Table 4.10 shows the RMS, peak error, overall frequency range examined, and the ratio between the RMS error and the frequency range for tests two and three. This shows relatively good agreement between the simulations and the measurements.

Table 4.5: RMS error, max error, temperature range during the test, and the ratio of the RMS error to the temperature range during the first case study.

Sensor	RMS Error	Max. Error	Range	RMS/Range
WT2	0.10 [°C]	0.16 [°C]	1.20 [°C]	0.08
WT3	0.03 [°C]	0.09 [°C]	1.15 [°C]	0.03
VT2	0.15 [°C]	0.42 [°C]	1.07 [°C]	0.14
VT3	0.32 [°C]	0.53 [°C]	1.01 [°C]	0.32

Table 4.6: RMS error, max error, temperature range during the test, and the ratio of the RMS error to the temperature range during the second case study.

Sensor	RMS Error	Max. Error	Range	RMS/Range
WT2	0.17 [°C]	0.46 [°C]	1.94 [°C]	0.09
WT3	0.22 [°C]	0.54 [°C]	1.90 [°C]	0.12
VT2	0.16 [°C]	0.39 [°C]	2.20 [°C]	0.07
VT3	0.31 [°C]	0.58 [°C]	2.21 [°C]	0.14

Table 4.7: RMS error, max error, temperature range during the test, and the ratio of the RMS error to the temperature range during the third case study.

Sensor	RMS Error	Max. Error	Range	RMS/Range
WT2	0.13 [°C]	0.31 [°C]	1.51 [°C]	0.09
WT3	0.18 [°C]	0.36 [°C]	1.47 [°C]	0.12
VT2	0.22 [°C]	0.73 [°C]	1.65 [°C]	0.13
VT3	0.28 [°C]	0.76 [°C]	1.64 [°C]	0.17

Table 4.8: Standard deviation of the error, for each temperature sensor across all three tests.

Sensor	Test 1	Test 2	Test 3
WT2	0.030 [°C]	0.15 [°C]	0.12 [°C]
WT3	0.029 [°C]	0.15 [°C]	0.12 [°C]
VT2	0.11 [°C]	0.11 [°C]	0.15 [°C]
VT3	0.094 [°C]	0.095 [°C]	0.13 [°C]

Table 4.9: Ratio of the standard deviation of the error to the temperature range, for each temperature sensor across all three tests.

Sensor	Test 1	Test 2	Test 3
WT2	0.025	0.079	0.082
WT3	0.026	0.078	0.081
VT2	0.10	0.050	0.088
VT3	0.093	0.043	0.082

Table 4.10: RMS error, max error, frequency range during the test, and the ratio of the RMS error to the frequency range.

Test	RMS Error	Max. Error	Range	RMS/Range
Test 2	2.9 [kHz]	7.04 [kHz]	19.16 [kHz]	0.15
Test 3	1.67 [kHz]	4.01 [kHz]	12.02 [kHz]	0.14

High Average RF Power Studies: During commissioning of CW operation for the RFQ, the RF power was brought up to the full design level to evaluate operational stability and trip recovery times. During this time, the supply temperature, the vane valve settings, and wall valve settings were also varied. Because of how sensitive the system is in this operating mode, scans of input variables could not be conducted. Instead, manual adjustments to keep the cavity closer to resonance were made, while the low-level RF system used feedback to maintain the specified cavity field. In order to run CW at the desired frequency, the supply temperature needed to be decreased to 20°C.

Figure 4.35 shows the variations in the vane and wall flow readings, the RF power level, the ambient cave temperature, the vane supply temperature, and the wall supply temperature during the tests. Note that the initial RF load on the cooling system caused the building chiller to trip at 1.5 hours into testing (at which point the RF was turned off). After the chiller was reset, the temperature returned to the nominal set-point of 20°C, and the saw-tooth fluctuations in the chilled supply temperature are due to the intermediate skid's independent control system.

For this data set, the temperatures in the cooling system at the supply to the RFQ (VT3 and WT3 for the vanes and walls respectively) and at the return by the skid (VT2 and WT2 for the vanes and walls respectively) were predicted with the analytic model. Figure 4.36 shows the comparison of the model to the measurements for these four temperature sensors. Here we see reasonable agreement between the simulated and measured temperatures for all sensors. Figure 4.37 shows the measured frequency shift throughout the test compared with the simulated frequency shift, which in contrast to the results from pulsed operation shows significantly higher prediction error.

We see that the model is generally adequate at predicting the qualitative frequency response of the RFQ in CW operation. It does appear to consistently over-predict fast transients in the resonant frequency, and this is likely due to inaccuracies in one or more of the parameters used to construct the model (e.g. RF heating or conductive coupling). Table 4.11 shows the RMS, max error, range in the response, and the ratio of the RMS error to the range for the CW test. The general agreement

shows that the modeling technique can be extended to high average power operation (CW RF), albeit with much higher prediction error.

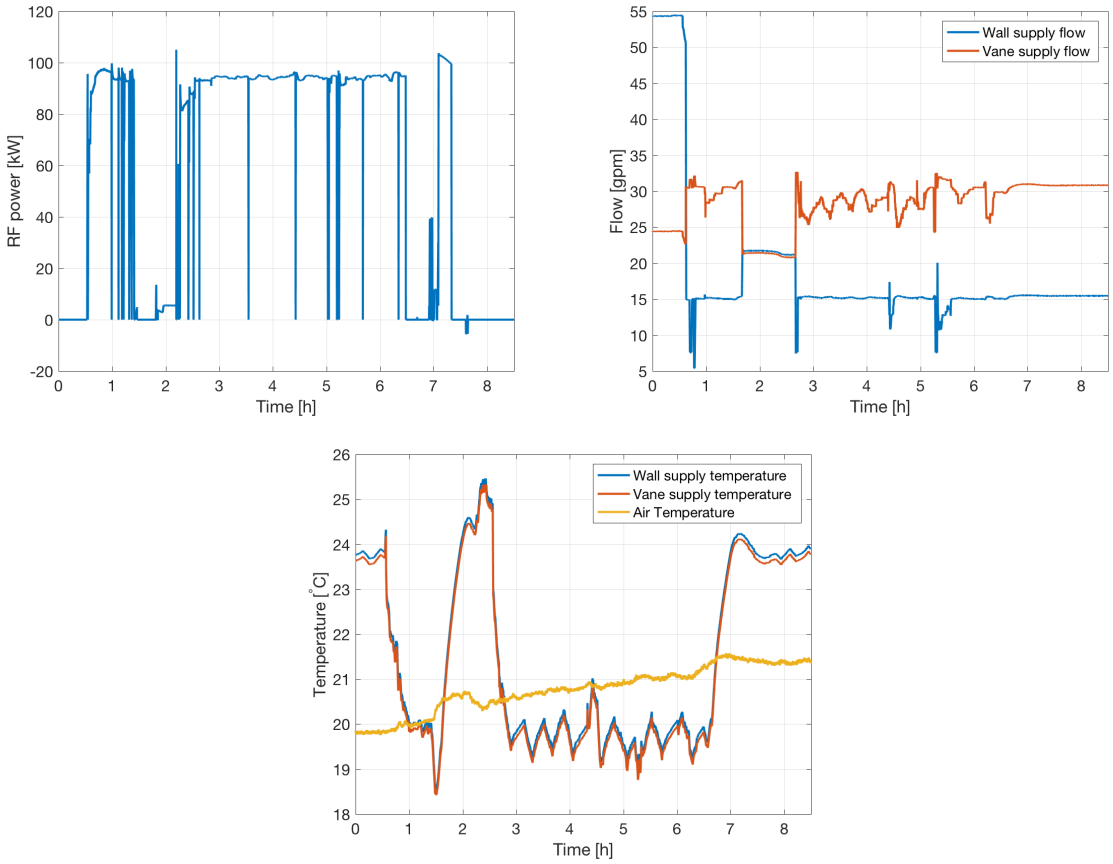


Figure 4.35: Changes in input variables during CW testing.

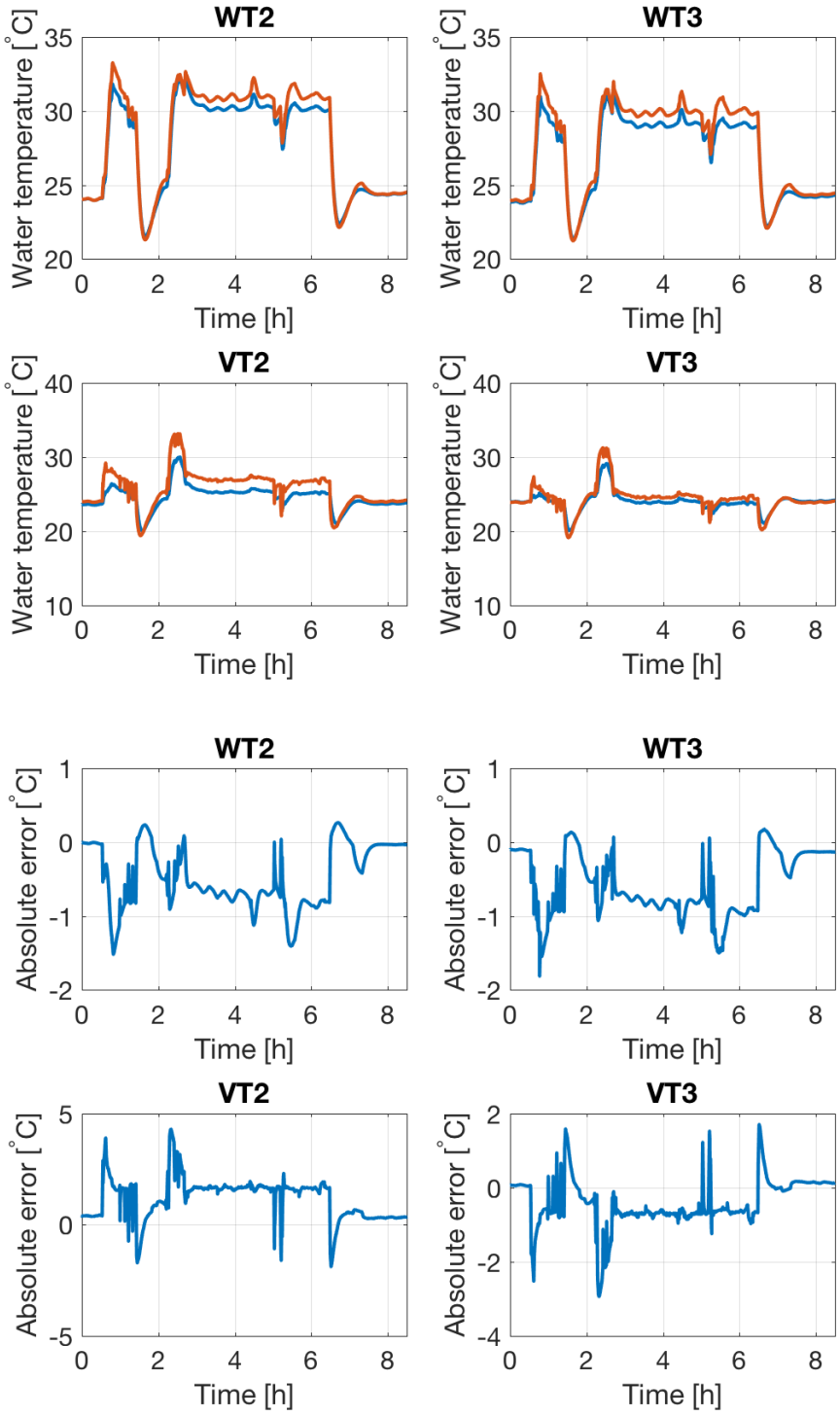


Figure 4.36: Comparison of measured and predicted temperature readings during high-power CW operation.

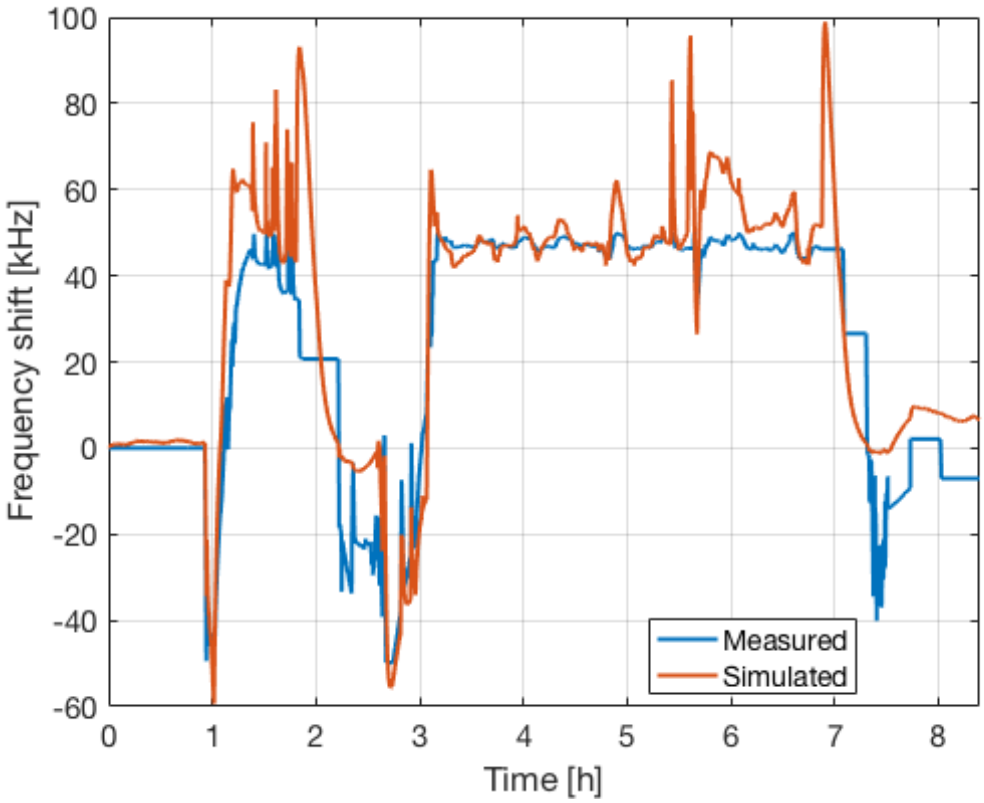


Figure 4.37: Comparison of measurements to simulations for the frequency shift in the RFQ versus time during high-power CW testing. Note that in contrast to the low-power pulsed operating mode, the absolute prediction error is significantly higher.

Table 4.11: RMS error, max error, frequency range during the test, and the ratio of the RMS error to the frequency range for CW testing.

	RMS Error	Max. Error	Range	RMS/Range
WT2	0.63 [°C]	1.52 [°C]	10.95 [°C]	0.058
WT3	0.71 [°C]	1.81 [°C]	10.27 [°C]	0.069
VT2	1.49 [°C]	4.29 [°C]	9.92 [°C]	0.150
VT3	0.77 [°C]	2.94 [°C]	9.04 [°C]	0.085
Frequency	12.63 [kHz]	52.77 [kHz]	100.00 [kHz]	0.126

Conclusions for Analytic Model

Prior to the delivery of the RFQ and obtaining the validation results described above, this model was used to aid in the design of the water-cooling system. Now that it has been validated, it can confidently be used as a design tool for future high-power RFQs, their cooling systems, and proposed control algorithms prior to installation. The modular design of the model enables sub-components to be easily replaced with other layout designs or parameters (e.g. copper body thermal mass, heat transfer coefficients, pipe diameters and lengths, etc.).

The analytic model is suitable for initial design studies and agrees reasonably well *a prioi* with the measured data, particularly with regard to the steady state system response. For example, Figure 4.38 shows the comparison between a simulated and measured frequency shift as the result of a roughly 5°C step change in the cold skid supply temperature.

In general, for low power the model can predict changes in frequency or temperature to better than 15% of the full range over a wide set of operational conditions. However, it should be noted that the prediction error is still quite high relative to the operating specifications of the RFQ for detuning. The prediction error for high-average-power RF operation (e.g. CW operation at full cavity field) was also significantly higher than for low-average-power operation.

Thus, quantitatively the dynamic error may be too high for it to be useful in online prediction and control or high-accuracy control design studies, particularly for high-power operation. The predictions for pulsed operation at 4 ms RF pulse duration and a 10 Hz rep rate had an average error of 1.67 kHz RMS in frequency shift, and a 4.01 kHz maximum error. For CW operation, the max error is 52 kHz and the RMS error is 12.63 kHz. The maximum acceptable detuning is 3 kHz, so it is unlikely that the analytic model can be used in MPC. This motivates the use of a learned model.

This approach in principle could be extended using machine data to fit some of the estimated coefficients, resulting in a more accurate representation of the system dynamics that can be used in model-based control schemes or further simulations (for example, a similar process is done for calibrating beam line lattices). A very limited attempt was made to tune uncertain parameters in

the model with an optimization algorithm, and thus minimize the difference between predicted and measured data; however, substantial improvement was not able to be readily made in this case. It is unclear what the limiting factor is (e.g. limits in model construction, optimization algorithm choice, or limitations in the data used). Further examination of this could potentially provide insight and utility.

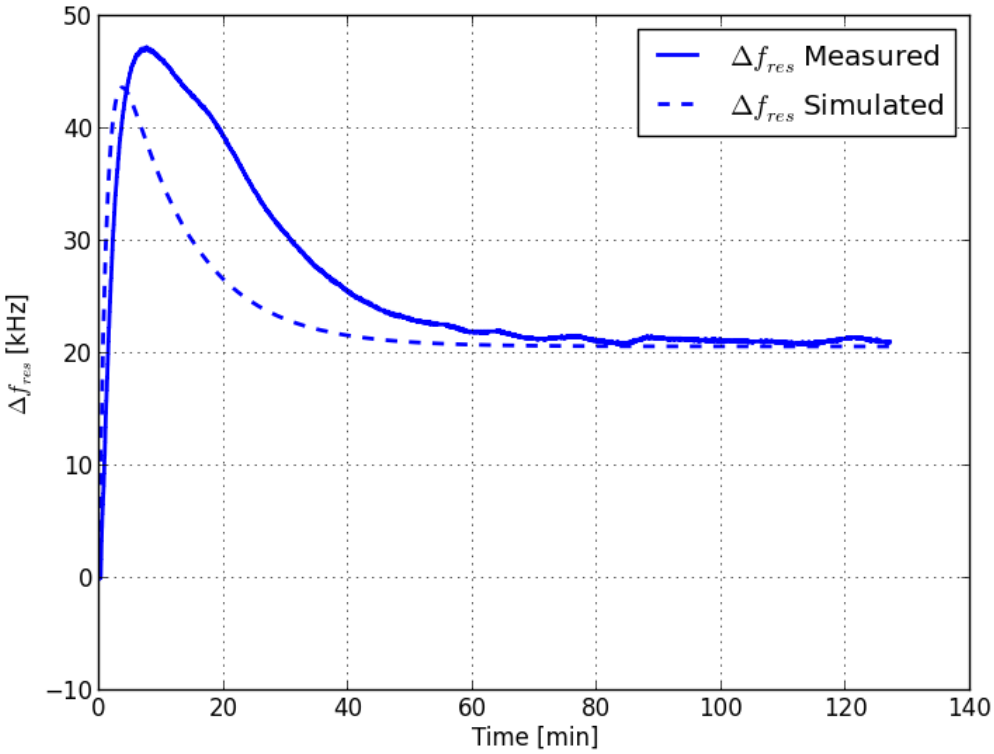


Figure 4.38: Comparison of resonant frequency measurements to analytic model predictions for a temperature transient.

4.4.4 RFQ Water System Control Assessment with Analytic Model

As discussed, prior to the arrival of the RFQ, the analytic model was used to assess and adjust the basic design of the water system and determine whether PI control on the vane valves would be sufficient to meet the RFQ specifications. It was also used to determine what the ideal temperature set points for the main water system should be to accommodate both pulsed and CW operation, with adequate PI control. Comparisons between uncontrolled responses in the entire system model, uncontrolled responses in the RFQ model alone (i.e. no transport delays, etc.), and controlled system responses with PI control on the vane valves were examined. Examples of the frequency responses are shown in Figure 4.39 and Figure 4.40.

After the arrival of the RFQ, the analytic model was used in conjunction with measured disturbance inputs to compare what the hypothetical system response would be under PI control compared to the uncontrolled responses. This is shown in Figure 4.41. The result highlights that active resonant frequency control is needed to maintain the performance specifications for the RFQ, and even PI control of the vanes alone can in principle substantially improve the magnitude and duration of detuning.

The analytic model was also used to make initial comparisons between PI control and MPC control. Figure 4.42 shows a comparison for control over the temperature of water entering the vane loop. As expected (and consistent with findings at FAST), MPC is faster to converge and provides lower overshoot.

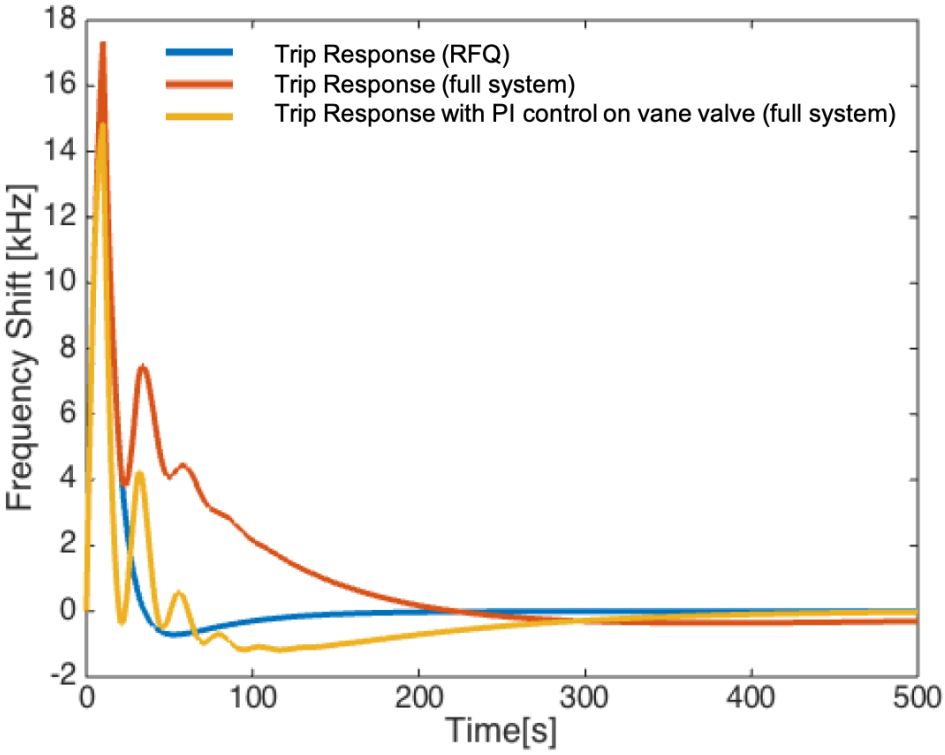


Figure 4.39: Simulated RFQ response to a 10s RF trip. The uncontrolled response of just the RFQ, the RFQ in the water system model, and the RFQ with PI control over the vane valves is shown.

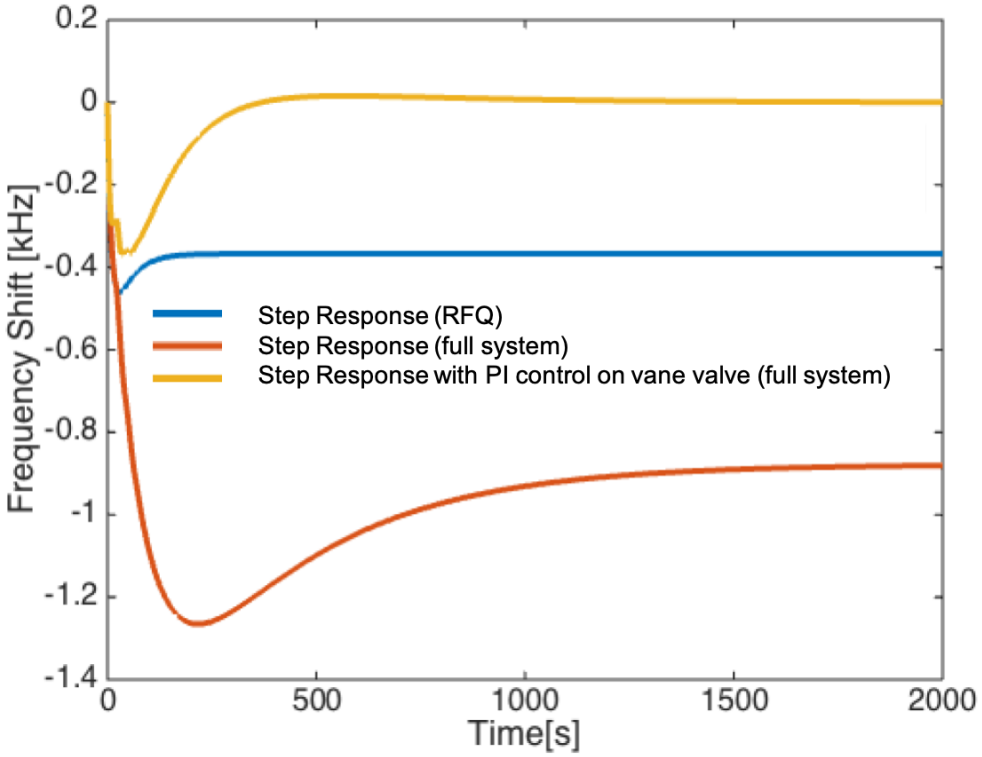


Figure 4.40: Simulated RFQ step response. The uncontrolled response of just the RFQ, the RFQ in the water system model, and the RFQ with PI control over the vane valves is shown.

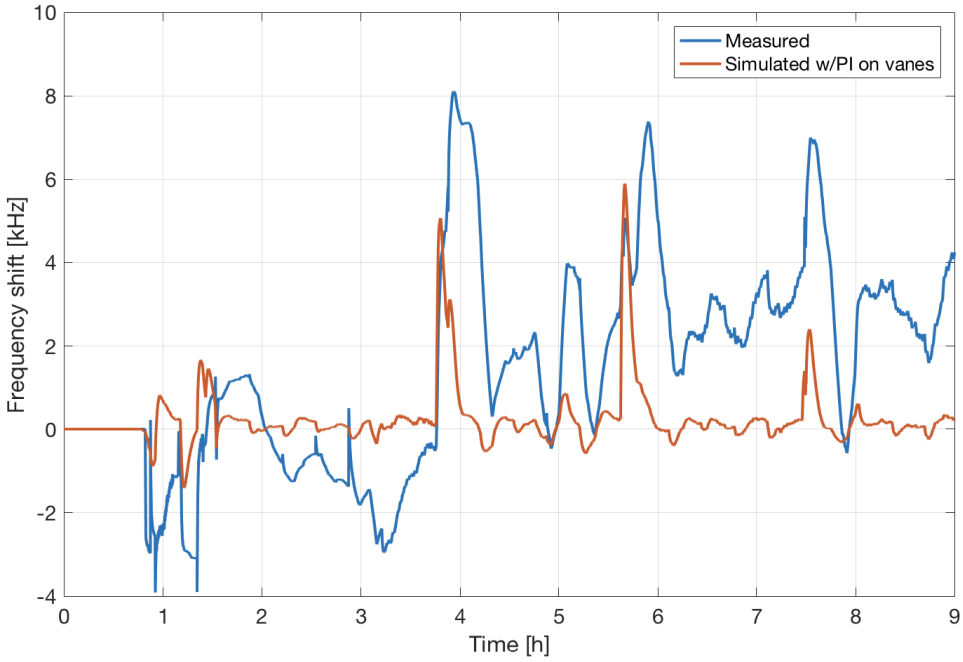


Figure 4.41: Once the RFQ arrived, a variety of characterization data were recorded. Here we show measured frequency shifts and the predicted performance of PI control on the vanes with the analytic model, given the disturbances from the measured data.

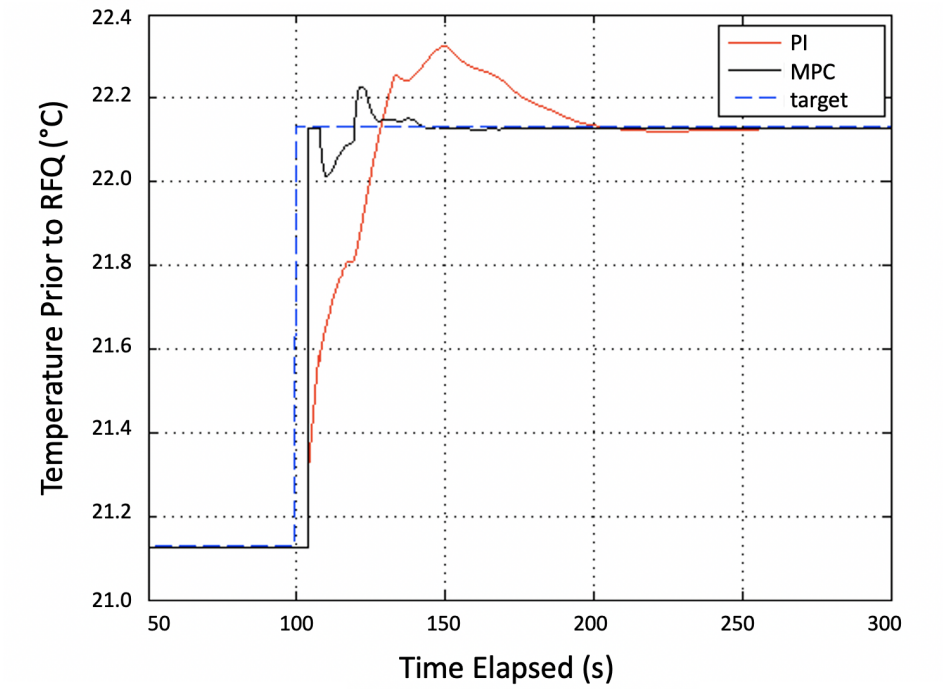


Figure 4.42: Comparison of PI control and MPC control over the temperature of the water entering the vanes, computed with the analytic model.

4.4.5 RFQ and Water System Characterization Data

The RFQ arrived at Fermilab in Fall 2015, after which the water system construction was completed and commissioning began. Once the system was built, characterization data were obtained from the RFQ. Initially, resonant frequency responses to changes in the water temperature were recorded to verify the predicted resonant frequency sensitivity from LBNL's ANSYS simulations.

For initial characterization prior to RF commissioning, we measured the resonant frequency response of the RFQ to sharp changes in the water temperature in the vane by closing the valve, reducing the intermediate skid temperature, and rapidly opening the valve to 50% of its full aperture as shown in Figure 4.43. We also examined the response to changes in intermediate skid temperature as shown in Figure 4.44. This was also used to characterize the timing between different elements in the system, as shown in Figure 4.45 and Figure 4.46. In addition, the flow valves have a nonlinear response that was characterized through scanning the settings and observing the resultant flow rates, as shown in Figure 4.47.

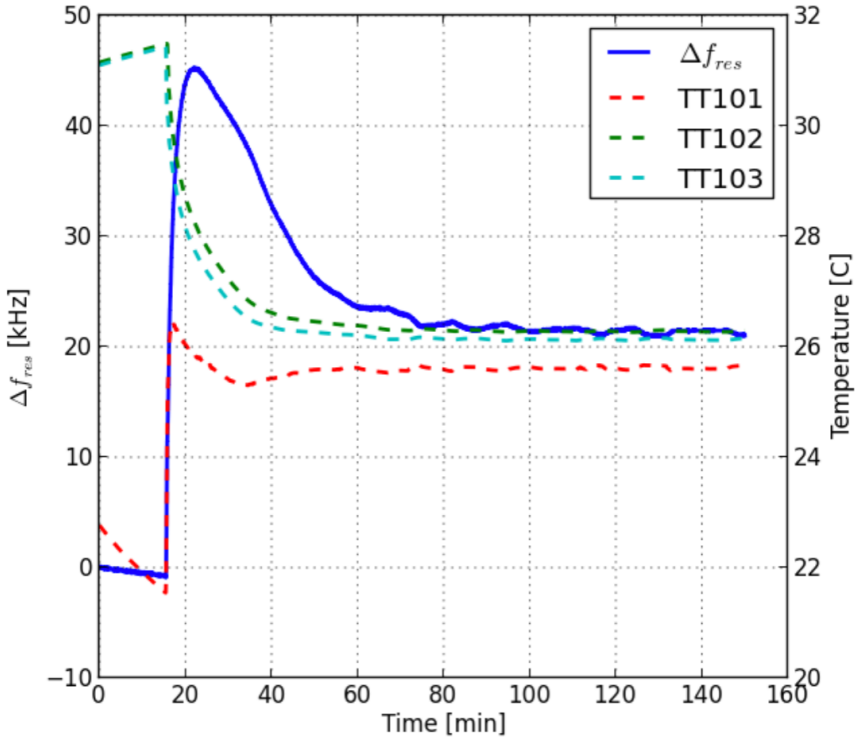


Figure 4.43: RFQ resonant frequency response to changes in water temperature in the vane.

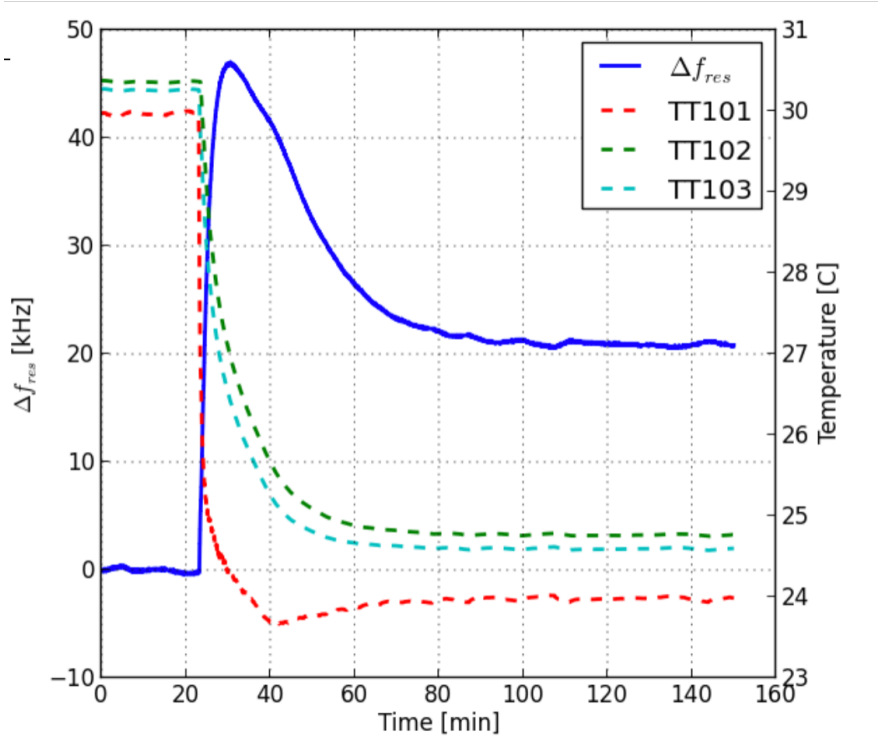


Figure 4.44: Measured, uncontrolled change in resonant frequency after a roughly 5-°C reduction in the cold supply temperature from the cooling skid.

	initial	transient	steady state
Time	0	7.9 min	124.2 min
Δf_{res}	0	47.38 kHz	20.80 kHz
TT_{101}	29.962 ^{o}C	24.278 ^{o}C	23.988 ^{o}C
TT_{102}	30.364 ^{o}C	26.910 ^{o}C	24.763 ^{o}C
TT_{103}	30.263 ^{o}C	26.371 ^{o}C	24.594 ^{o}C

Figure 4.45: Response times for changes in skid temperature.

	initial	transient	steady state
Time	0	7.2 min	115.4 min
Δf_{res}	0	46.13 kHz	21.15 kHz
TT_{101}	22.280 ^{o}C	25.796 ^{o}C	25.649 ^{o}C
TT_{102}	31.489 ^{o}C	28.126 ^{o}C	26.141 ^{o}C
TT_{103}	30.941 ^{o}C	27.639 ^{o}C	26.284 ^{o}C

flow path	time delay [s]
$TT_{101} \rightarrow TT_{103}$	1.0
$TT_{103} \rightarrow TT_{102}$	17.0

Figure 4.46: Response times for changes in vane temperature.

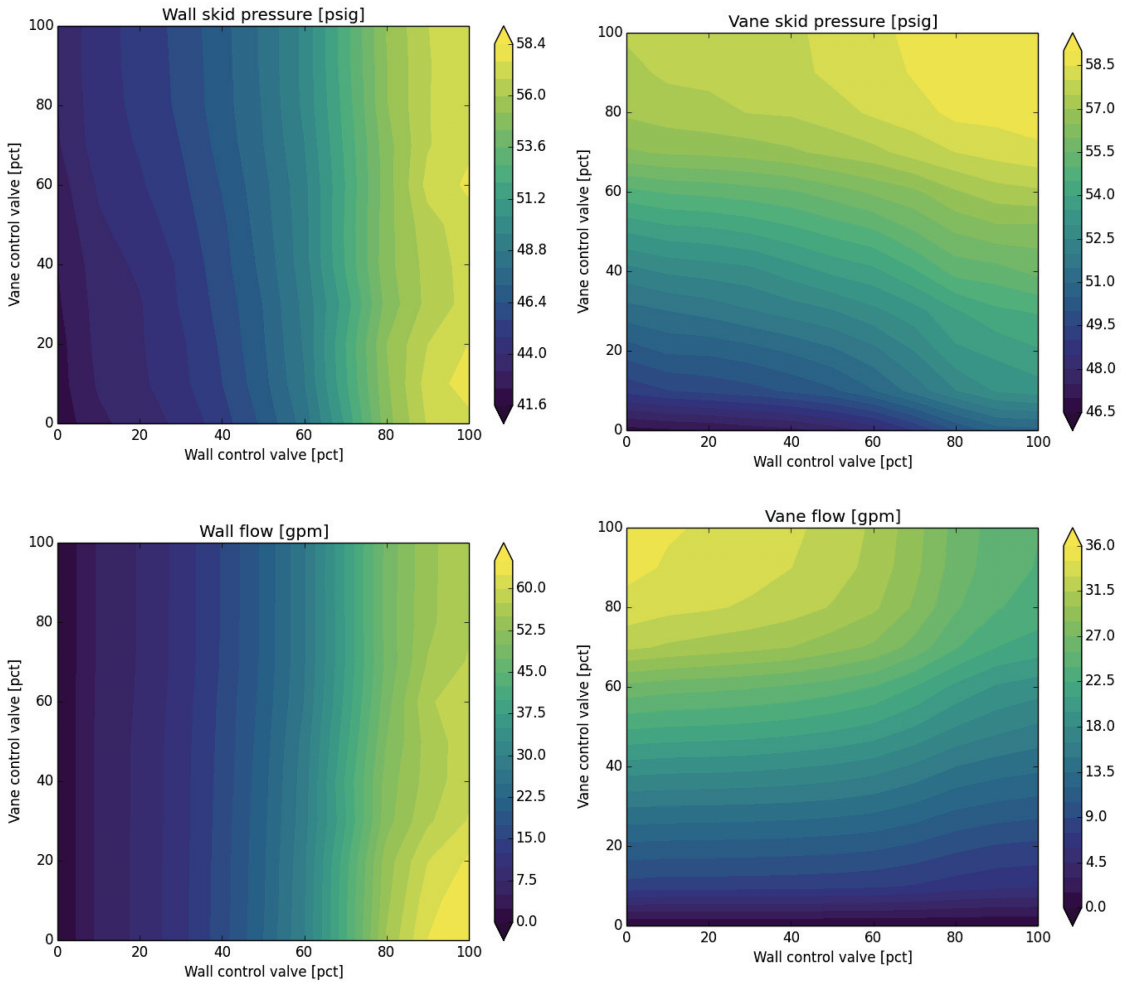


Figure 4.47: Flow rate and pressure relationships obtained from the cooling system during initial characterization studies.

4.4.6 Neural Network Modeling

Because of limitations in the prediction accuracy of the analytic model for CW operation, during RFQ commissioning we also built a neural network-based model. This could be used as an alternative model in an MPC routine. It could also be used in neural network-based reinforcement learning (both directly in the algorithm or to help train a controller).

Training Data

Parameter scans yielded the frequency response to various combinations of vane valve settings, wall valve settings, and RF field amplitudes. Table 4.12 shows the range of input parameters and resonant frequency values sampled. Figure 4.48 shows a representative scan over the vane valve setting and RF field amplitude. In this case, we used this entire scan as the testing data set. In total, the wall valve was varied from 0 to 99 % open, and the vane valve was varied from 0 to 99 % open. The vane-to-vane cavity field was varied from 0 kV to 70 kV. For all data sets, the repetition rate was 10 Hz and the pulse duration was 4 ms. During this time, as part of commissioning, the LLRF feedback was on to assess the ability of the LLRF system to regulate the cavity field and maintain the drive frequency at 162.465 MHz.

Approximately 64 hours of measured data (during which the average RF power and valves were scanned) were obtained. This includes RF trips and startup/shutdowns. To exclude suspect measurements from training, target resonant frequency values are ignored when the cavity field drops below 0.5 kV (e.g. during multipacting or sparking). However, as soon as the field recovers, the targets are used once again, thus including the thermal (and frequency) excursions caused by such interruptions.

The resonant frequency is calculated from the RF signals as follows:

$\phi_f = \arctan((\mathrm{E}_{f1}\sin(\phi_{f1})+\mathrm{E}_{f2}\sin(\phi_{f2}))/(\mathrm{E}_{f1}\cos(\phi_{f1})+\mathrm{E}_{f2}\cos(\phi_{f2})))$

$\delta = \tan(\phi_c - \phi_f)/(2\,Q_L)\,f_0$

where E_{f1} and E_{f2} are the magnitudes of the drive signals from each amplifier, ϕ_{f1} and ϕ_{f2} are the forward phases of each drive signal, ϕ_f is the calculated forward phase, ϕ_c is the cavity phase, ϕ_{f0} is the drive frequency, Q_L is the loaded quality factor of the cavity, and δ is the detuning (note

that this does not take into account beam loading). Because the RFQ is driven by two amplifiers, we need to calculate the vector sum of the two forward signals in order to obtain the forward phase for the calculation of resonant frequency shift.

The ranges of the training data are shown in Table 4.12. The validation data were interleaved with the training data (every-other sample). The testing data consists of a 2-D scan over vane valve settings and RF field amplitudes under a higher constant wall valve setting than was seen during training. The wall was set at 99% open for testing, whereas the highest prior value seen was 75% open. This was done specifically to assess the model's ability to generalize.

Table 4.12: Variable Ranges for Measured RFQ data.

Variable	Min	Max	Units
Wall Valve Setting	0	99	[% open]
Vane Valve Setting	0	99	[% open]
Cavity Field	0	70	[kV]
Wall Supply Temp	19.1	20.5	[C]
Vane Supply Temp	18.9	20.4	[C]
Wall Entrance Temp	19.8	22.8	[C]
Vane Entrance Temp	19.5	21.9	[C]
Resonant Frequency	162.4403	162.4738	[MHz]
Cave Temp	23.3	25.5	[C]
Cave Humidity	19.1	36.6	[%]

Architecture and Training Procedure

The variables used as inputs to the model are the temperature of the water entering each cooling sub-circuit (T01, TT01), the temperature of the water returning from the RFQ (T02, TT02), the two flow control valve setting read-backs (i.e. the actual values as opposed to the set point values), the ambient temperature and humidity, and a metric proportional to the power entering the cavity (given by the cavity field measurement and the duty factor). Here, the inputs are not instantaneous measurements, but, rather, we use a sequence of previous values that are sufficient to capture the

present system state. The output of the model is the predicted resonant frequency of the RFQ. For initial modeling, a simple feed-forward architecture with multiple previous time-steps embedded as inputs was selected. Initially, 30 minutes of previous system data were provided with a decaying sample interval. This resulted in 12,600 total inputs to the model at each time step.

An initial architecture and set of weights were obtained by conducting initial training and subsequently removing connections with small weights. The resultant network was then refined with further training. 10 networks with new initializations were trained using the scaled conjugate gradient optimization algorithm [200]. Two hidden layers with 25 and 7 nodes in each layer respectively were used. An approximate hyperbolic tangent activation function was used for all nodes, except the output node, which used a linear activation function.

This architecture choice was in part motivated by the lack of available computing hardware to deploy a more complicated model on the machine (e.g. an LSTM or a GRU). Later studies also revealed that LSTMs performed comparatively poorly in the prediction task (likely because the sequence of incoming data was too long). Note that subsequent developments in sequence predictions (e.g. see [59]) are likely better suited to this problem.

Neural Network Model Performance

The best-performing network has a mean absolute prediction error of 346 Hz on the test set, 98 Hz on the validation set, and 116 Hz across all training, validation, and testing data. Figure 4.49 shows the predicted and measured resonant frequency.

The neural network model predicts the resonant frequency of the RFQ under changes in the cooling system substantially more accurately than the analytic model, and sufficiently well for use in a model-based control routine. For comparison with another learned model, we also looked at a linear learned model, which for pulsed operation had 1.13 kHz RMS error and 2.66 kHz maximum error.

The neural network model needed to be validated for predictions far ahead of the present time-step in order to be useful in model predictive control. To this end, we trained models that include resonant frequency predictions across 600 future time steps. For predictions on the farthest time

step (i.e. in principle the hardest to predict), the mean absolute prediction error was 339 Hz, with a 1588 Hz maximum. The agreement is still quite good, as shown in Figure 4.51.

In typical MPC, one might set uncontrolled parameters, such as temperature and humidity, as constants (usually the most recent value observed). However, for the RFQ, there were substantial changes in temperature and humidity on the relevant timescales (e.g. due to heating from equipment and cycling of the environmental temperature control systems in the cave). Consequently, a model to predict the evolution of the temperature and the humidity over the prediction horizon was developed. These predicted disturbance inputs were then used during MPC.

The initial model has a few areas that could likely be improved. The model does not inherently include any state information (in contrast to an LSTM or a GRU). It is also likely that substantial reduction of the input parameter space could be conducted. Note that the development of the neural network model was not a high-priority goal compared with commissioning the initial control system for the PIP-II RFQ, detailed below. The model developed here was actually too computationally intensive to use with the available computing infrastructure on the Fermilab's control network (which was quite antiquated). This was the case even with attempted reduction of inputs as well as to the prediction horizon used with MPC. Now that the control system software infrastructure (described below) is in place and the basic systems are working, further effort on ML-based modeling and deployment (e.g. with a dedicated computer on the controls network) could be revisited, and both of these issues could likely be addressed.

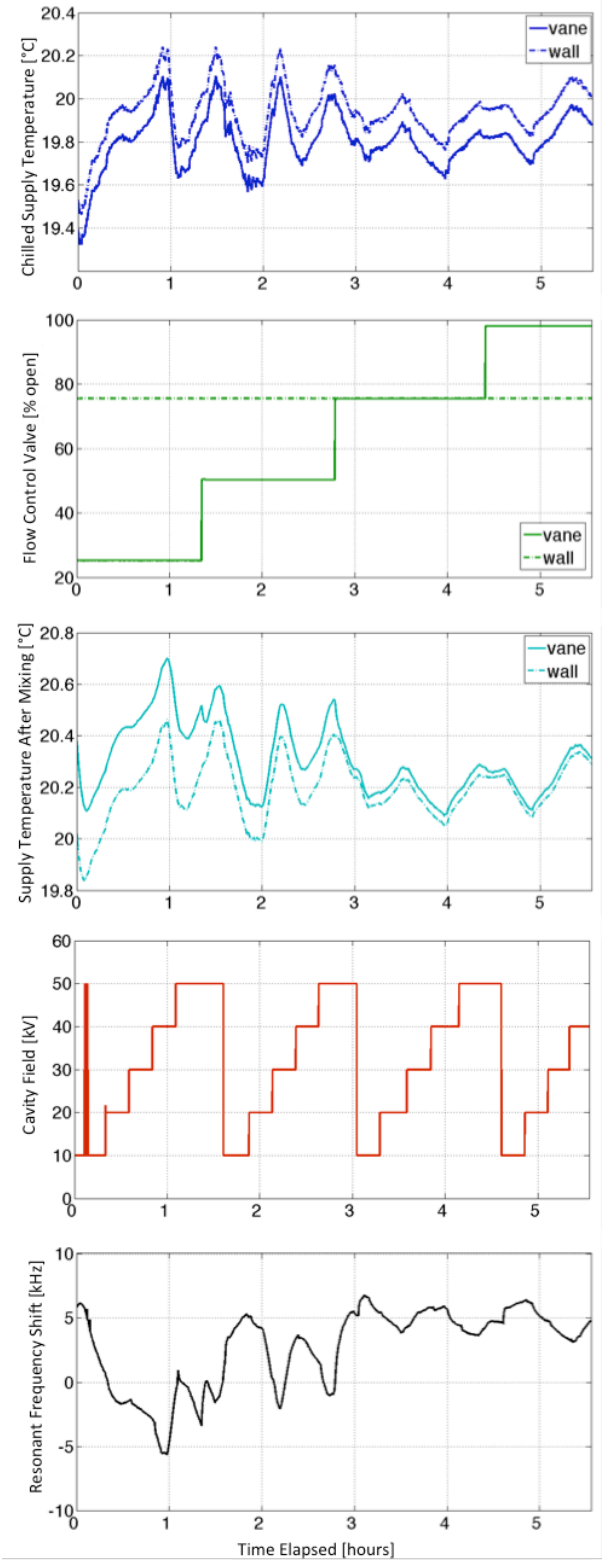


Figure 4.48: An example scan from the measured data; this one was used as the testing set. The repetition rate was 10 Hz and the pulse duration was 4 ms. Note that the fluctuations in the chilled supply temperature have a significant impact on the resonant frequency at these relatively low average RF power levels.

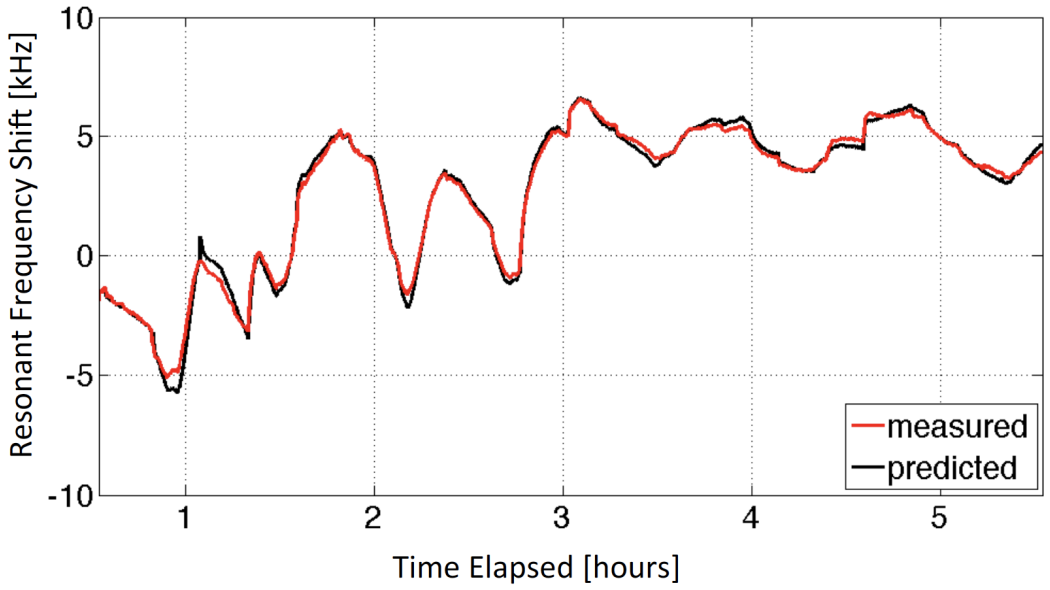


Figure 4.49: Measured and predicted resonant frequency shift values versus time for the test set scan shown in Figure 4.48. Note that the wall was set at 99% open for testing, whereas the highest prior value seen was 75% open. This was done specifically to assess the model's ability to generalize.

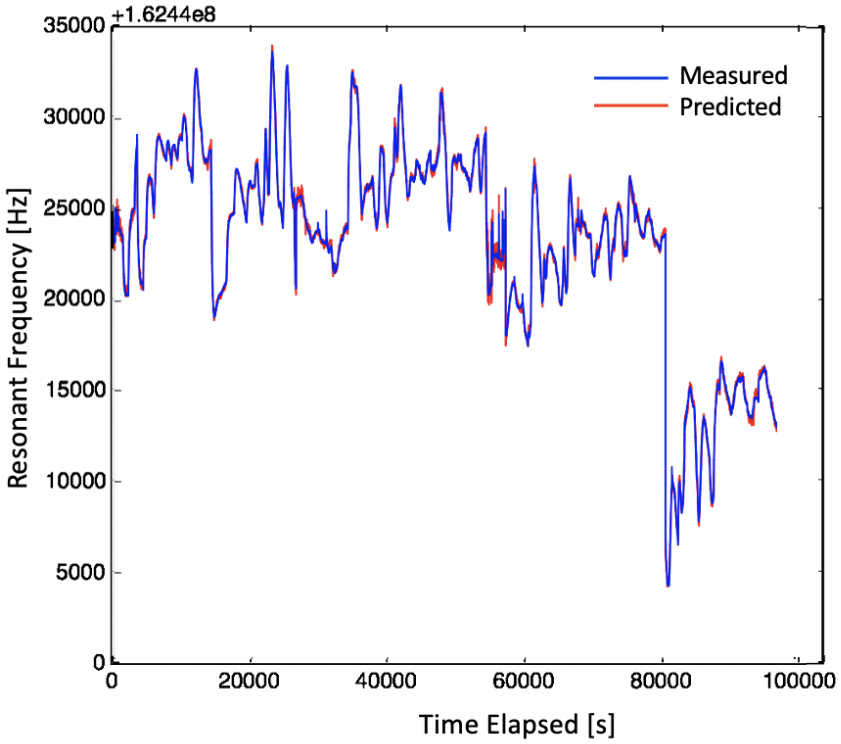


Figure 4.50: Measured and predicted resonant frequency values versus time for a wider data set.

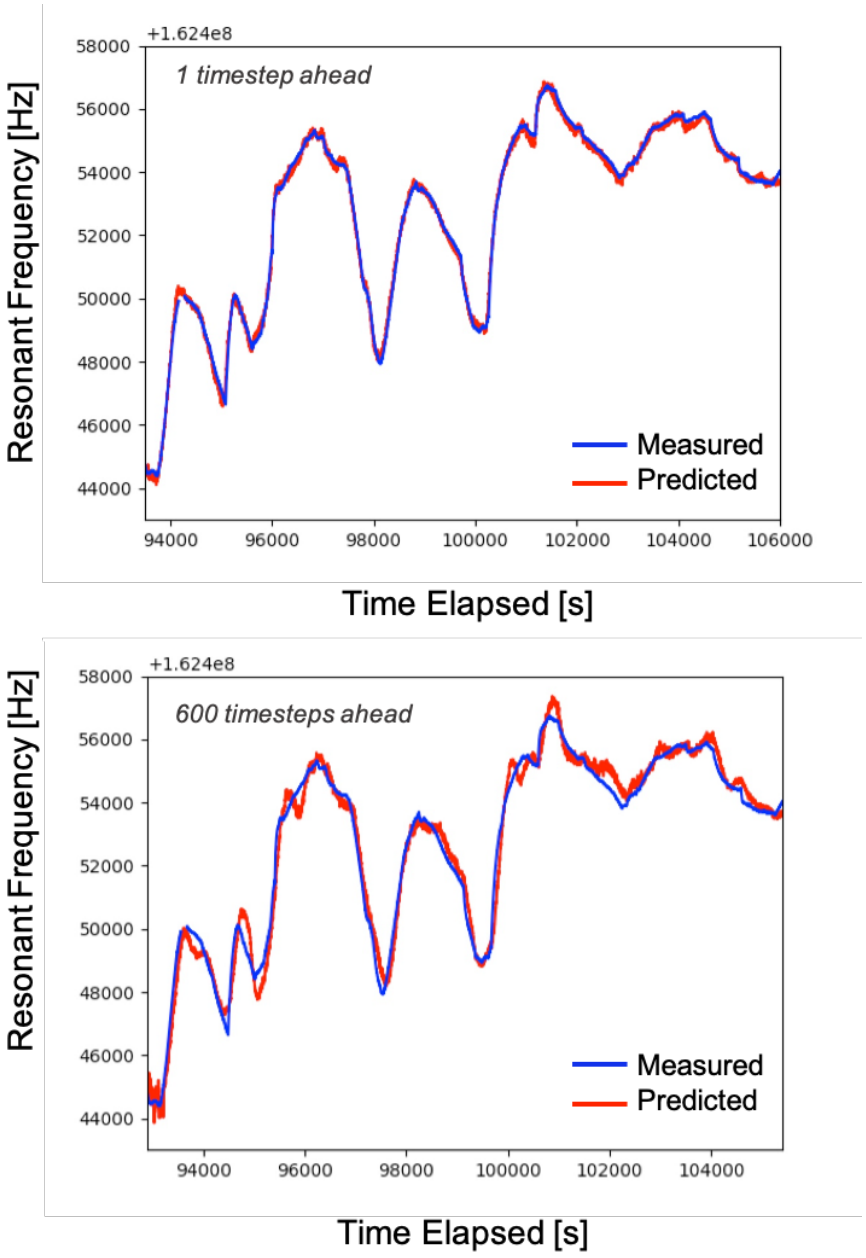


Figure 4.51: Resonant frequency measurements and neural network predictions versus time, for predictions 1 timestep ahead (top) and 600 timesteps ahead (bottom) of the last measured input. This shows that the neural network model is able to predict the future system evolution accurately. The model inputs include the predicted humidity and temperature evolution from a secondary model.

4.4.7 Resonant Frequency Control System

A critical component of this work was the development of a dedicated control framework to handle high-level decisions and execute control calculations for the PIP-II RFQ. The level of desired system flexibility motivated the development of a modular control system architecture that can be easily modified and would facilitate the use of multiple control algorithms and operating modes. The framework is written in Python in a modular fashion to facilitate easy modifications to the code. This framework is operational for the RFQ and could be readily modified for similar RF control tasks at Fermilab.

The LLRF system [227] is capable of operating in either self-excited loop (SEL) mode, in which the drive frequency follows the cavity resonant frequency, or generator-driven resonator (GDR) mode, in which the drive frequency is set. In SEL mode, the use of RF overhead is minimized due to the changing of the drive frequency to match the RFQ, thereby also minimizing the reflected power. As such it is useful to switch into SEL mode automatically when the detuning increases beyond a tolerable threshold. It is also needed in order to avoid machine trips and damage due to high amounts of reflected RF power. In addition to handling the actual control problem of resonant frequency control, the control framework handles a variety of high-level decisions related to RFQ operation, including:

- Switching between SEL and GDR mode automatically based on reflected power, measured detuning, and cavity field set-point vs. read-back (this includes activating LLRF feedback).
- Setting the LLRF averaging windows at startup, according to the desired pulse duration.
- Switching between user-requested operational states (e.g. temperature control, resonance control, manual override of automated switching between SEL and GDR).
- Detecting RF trips and taking appropriate action (e.g. transitioning to water temperature control) until power returns.
- Calculating the resonant frequency using a method appropriate for the current operational state (i.e. SEL vs. GDR mode with LLRF feedback require different calculations).

Additionally, accommodation of several high-level features were desired by the PIP-II team, including the following:

- Resonant frequency control using the water system valves.
- Temperature control, with consideration of RF. For trip recovery and commissioning it was desired to be able to control the water temperature as the target output (rather than the resonant frequency directly) as needed, while still enabling automated control over SEL/GDR switching.
- RF forward power control for startup and trip recovery. It was desired to control the RF forward power magnitude during a cold start or recovery from a trip. This is desired so that the impact of a target RF ramp can be taken into account when adjusting the flow valves. At present this uses a simple linear ramp to a user-specified RF power level, but it could eventually be incorporated into an MPC routine to optimize the ramp trajectory itself as well.
- Accommodation of multiple control algorithms. It was desired to be able to easily switch between PID and more experimental control modes, such as MPC, without affecting the other functional aspects of the controller.

Main Program Flow

The main program flow is as follows. (1) An operator selects an operational mode from the list. At each control interval, operator input is checked to determine whether a new state has been requested. (2) Upon startup, the program builds a data buffer for MPC (if applicable), sets the LLRF windows, and initializes variables. (3) At each execution cycle of the program, it gathers new readings, checks the requested state, pre-processes some of the data, makes relevant calculations (e.g. resonant frequency), and selects the appropriate action function. (4) The action is executed, which involves calling the control module when applicable. Mode-specific settings are then sent back to the main program. (5) Finally, the information is collected and commands are sent to the

Erlang front-end. At present, steps 2-4 repeat at a 1 Hz rate, but this rate may be increased in the future if need be (up to 10 Hz). Figure 4.52 shows the program flow for resonance control mode specifically.

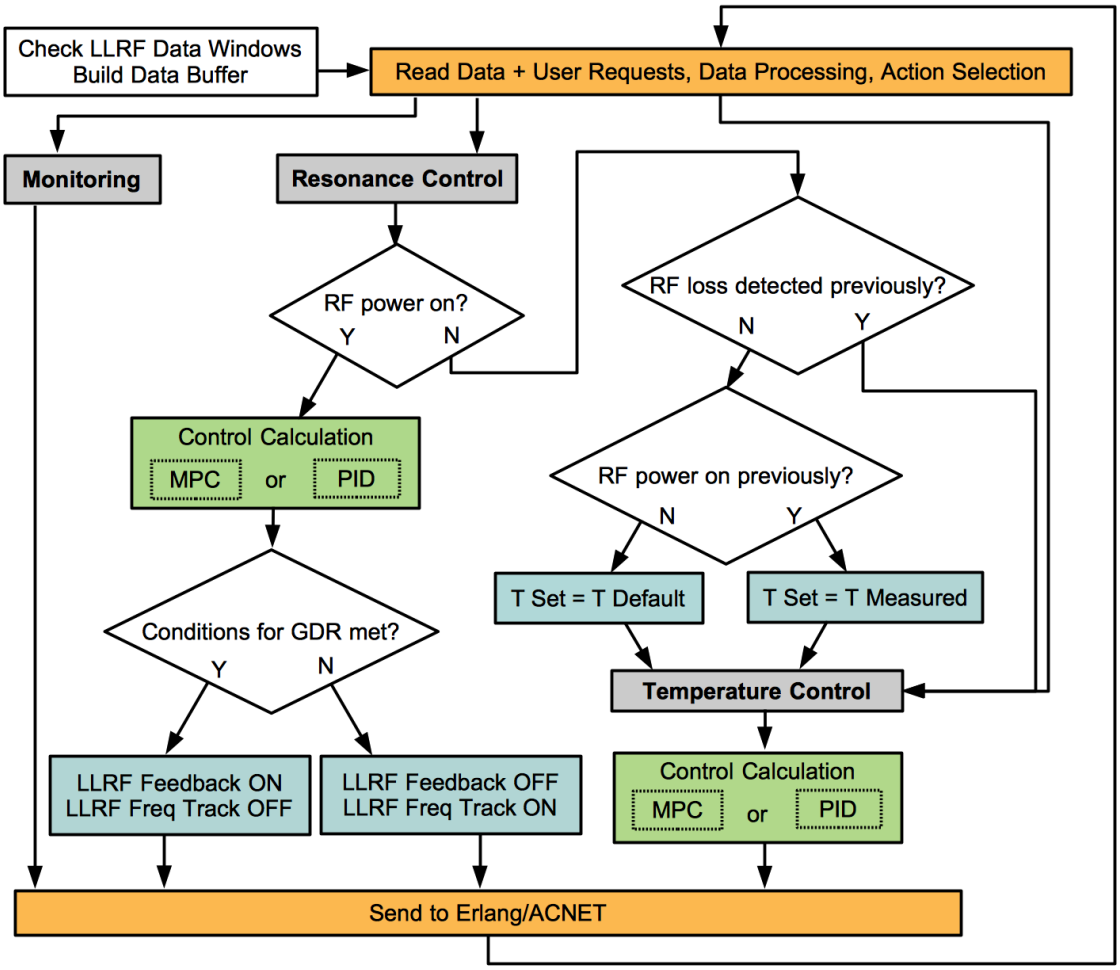


Figure 4.52: Overview of the program flow for the developed resonant frequency control system.

In resonance control mode, the controller adjusts the flow control valve settings such that the desired resonant frequency is reached and maintained. In this mode, checks are also carried out to ensure that the conditions for GDR are met. These conditions are that the resonant frequency is within a tolerable threshold and that any oscillations in the water system have settled (so that it will not perturb the RFQ above the threshold). If these conditions are met, the controller signals the LLRF either to stay in (or switch to) GDR mode. If they are not met, it signals the LLRF to

either switch to or (stay in) SEL model. The user can also request that the GDR/SEL switching by the resonance controller be disabled, in which case the decision of the controller is over-ridden and the GDR/SEL parameter can be set by the user directly from the ACNET GUI, PID for resonance control operates only on the vane circuit (with the wall valve kept open loop). This scheme (PID on the vanes) is the way most RFQ resonance controllers work. MPC for resonance control operates on both the wall and vane circuits simultaneously.

In temperature control mode, the flow control valve settings are adjusted such that the desired water temperature entering the RFQ is reached. PID for temperature control was requested so that more straightforward control could be used in the event that issues with the system arise. It operates on the wall first (bringing it to within a user-specified tolerance of the requested temperature) and then on the vane. This is to avoid having these systems (which are coupled through the copper of the RFQ) fight one another and lead to excessively long oscillation times while settling.

During operation, the controller also continues to check for the presence of RF power and takes action if it drops out. First, it waits to see if RF power comes back on immediately (i.e. within the next data acquisition, which is variable up to a 10 Hz rate). If not, it shuts the flow control valves to preserve heat in the RFQ and reduce the time it takes to restore the cavity to resonance (i.e. in the event of just a short RF power loss).

Once enough time has passed without RF (the length of time can be set by the user), the controller returns to an anticipatory state where the water system is set at an optimal starting point to smoothly counter RF heating once the cavity is turned on. There is also a startup mode which ramps the forward RF power to the desired level while also adjusting the flow control valves to reduce any transients due to initial RF heating.

For both startup and trips, the transient RFQ frequency error is initially very large. For PID control, only the feedback is used and the future natural settling from the transient is not predicted and taken into account (as it would with MPC). Thus, the feedback can actually perturb the system more in response to the large error and prolong the settling time. For this reason, in PID resonant frequency control mode, when a trip is detected or when startup is initiated, resonant frequency

control is blocked for an appropriate amount of time. Operational examples of this are shown in later sections.

Control System Architecture and Interfaces

The controller communicates with several other subsystems, including the LLRF hardware, a Cryo-con 18i temperature monitor [11], and a programmable logic computer (PLC) for the water system. Figure 4.53 shows a simplified view of the system interfaces. All of these communications occur through the ACNET control system via a front-end implemented in Erlang. This front-end is connected to the resonance controller via a User Datagram Protocol (UDP) interface. We use a custom protocol (i.e. specific to this controller) generated automatically with Fermilab's protocol compiler [228, 229]. The front-end then handles the communication with the other subsystems through standard ACNET messages, thus hiding that complexity from the controller.

The PLC is used to control the flow control valves and provide read-backs of the majority of the water system readings (flow valves, pressure valves, non-essential temperature sensors). The resonant frequency controller provides the flow control valve settings to the PLC, and the internal PLC feedback controls make the adjustment to the valves themselves. It also has a built-in controller for PID control of valves relative to the T03 temperature sensors and a user set point (which is useful, for example, when not running RF to the RFQ). The resonance controller tells the PLC whether to use this internal controller or not for temperature control.

An operator can specify the desired operational mode as an input to the controller. Error read-backs, a heartbeat, and the present operational mode are sent back from the framework to the user. From the LLRF system, readings include the present SEL/GDR state, an indicator for pulsed or CW mode, the forward and cavity phase readings, RF repetition rate and pulse length, forward and reflected RF power magnitudes, cavity field magnitude, and RF timing and averaging windows. The Cryo-con 18i and PLC return temperature sensor readings, pressure and flow readings, and flow control valve readings. The controller sets the LLRF system to SEL or GDR mode and sets appropriate window settings.

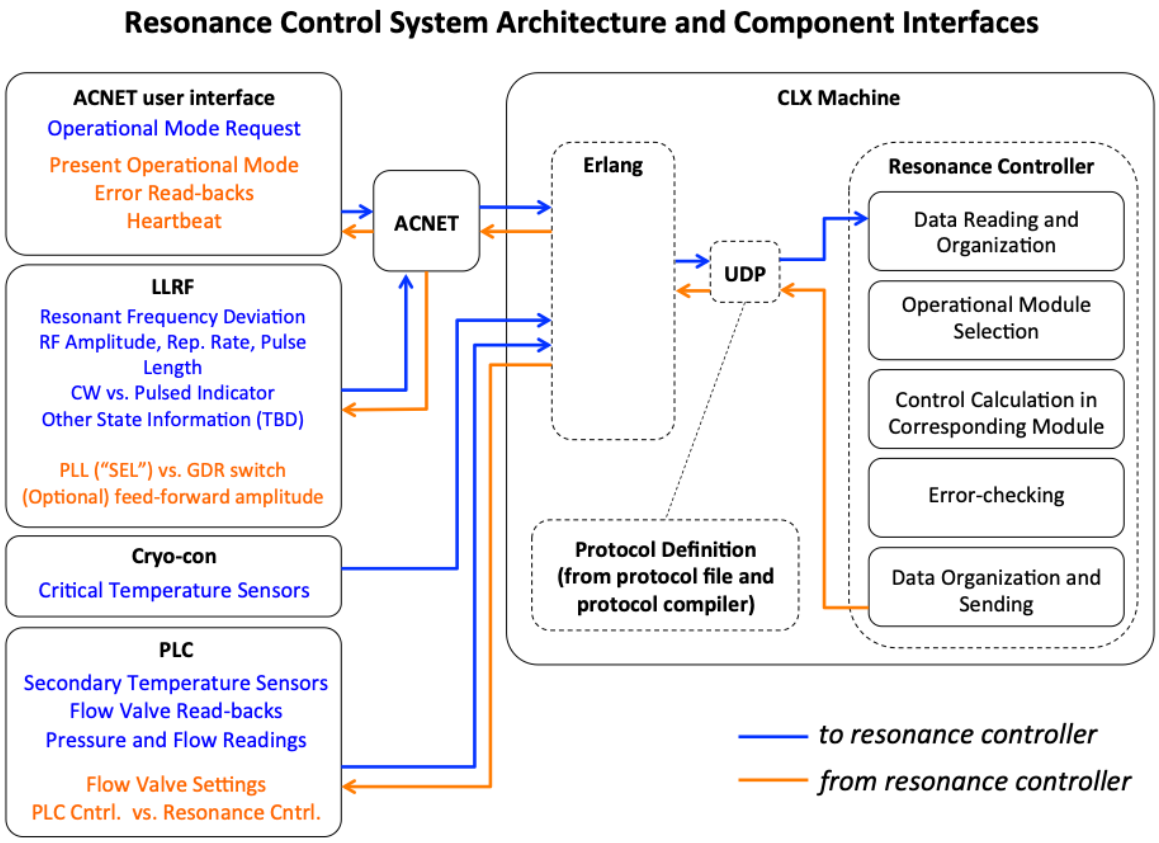


Figure 4.53: Simplified, conceptual diagram of the resonance control system architecture and interfaces built in this work to support RFQ operation.

Modular Structure and Software Implementation

The controller software consists of a hierarchical set of Python modules: one for tasks that must be done on each iteration (the main module), one for actions that are unique to specific operational modes (the action module), and one for specific control calculations (the control module), as shown in Figure 4.54.

The main module interprets user requests, reads in data from the water system and LLRF system, checks the LLRF and water system readings (as well as the internal resonance controller behavior) for errors, and selects the appropriate action in the action module-based user requests and readings from the machine, and sending commands back to the front-end.

The action module implements code for various actions to take in each operational state (e.g. resonance control, temperature control, PLC control, no control action on the valves - just SEL/GDR

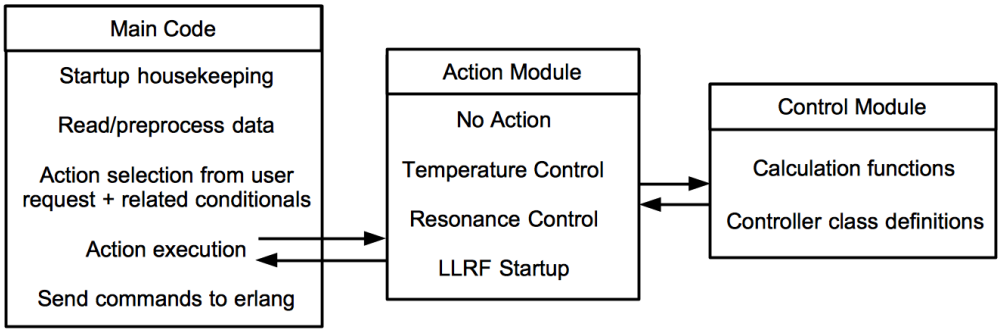


Figure 4.54: Modular control framework structure for the RFQ resonant frequency control.

switching). As part of their execution, these handle the logic for state variable changes (e.g. SEL/GDR toggling) and input/output to control calculations. The action module also handles computations and checks that need to be completed for different operational modes.

The actual control update calculations (e.g. changes in the flow control valve settings) are done in a separate module that has a standard interface to the action module (e.g. PID, MPC, MPC + RF ramp). This enables one to switch easily between different control algorithms.

4.4.8 Control Studies and Commissioning of Control Framework

The main framework was commissioned and PI control over the frequency in both pulsed and CW mode was tested. This included frequency recovery after RF trips, user-driven state switching, and automated switching between SEL and GDR modes. The PI controller regulates the resonant frequency of the RFQ by adjusting the vane flow valve aperture. Because the flow responses of the valves are coupled and nonlinear [230], a weighted correction is applied to the PI control output. The correction is inversely proportional to predicted change in flow for a given change in valve setting. Because of the transport delays in the system, a corresponding delay is placed into the frequency and temperature PI controllers. The weighting of the PI gains, the flow valve correction, and amount of delay were all tuned during commissioning.

Figure 4.55 shows the uncontrolled detuning in response to a small change in RF power (by increasing the cavity field). Figure 4.56 shows PI frequency control under CW operation for a similar step in cavity field as that shown in Figure 4.55.

Figure 4.56 - Figure 4.62 show examples of the PI frequency controller performance during pulsed and CW operation. In Figure 4.57, the pulse duration was increased by 2 ms while the cavity field was at 65 kV (vane-to-vane). Figure 4.58 shows the response for a decrease in cavity field. Figure 4.59, showing an overlay of the vane cold supply temperature on top off data from Figure 4.58, highlights the impact of disturbances in the vane supply temperature on the resonant frequency response in pulsed operation. The impact around the 13-minute mark due to a change in the water temperature is readily apparent.

Figure 4.60 shows frequency recovery from a 10-second trip at full specified field (60 kV), in which the detuning is reduced to 3 kHz in 140 seconds. As such, PI control does not fully meet the specification for trip recovery time at full field, although frequency trips at lower field values can be recovered in under 10 $\times$ the trip duration. Operational experience indicates that PI control may be brought into specification by slightly delaying the activation of the PI loop after power is restored. Figure 4.61 shows an example from a compound trip under CW operation for similar conditions, and Figure 4.62 shows performance under changing cavity field levels. Collectively,

these results show that PI resonant frequency control over the RFQ makes stable CW and pulsed operation in GDR mode possible.

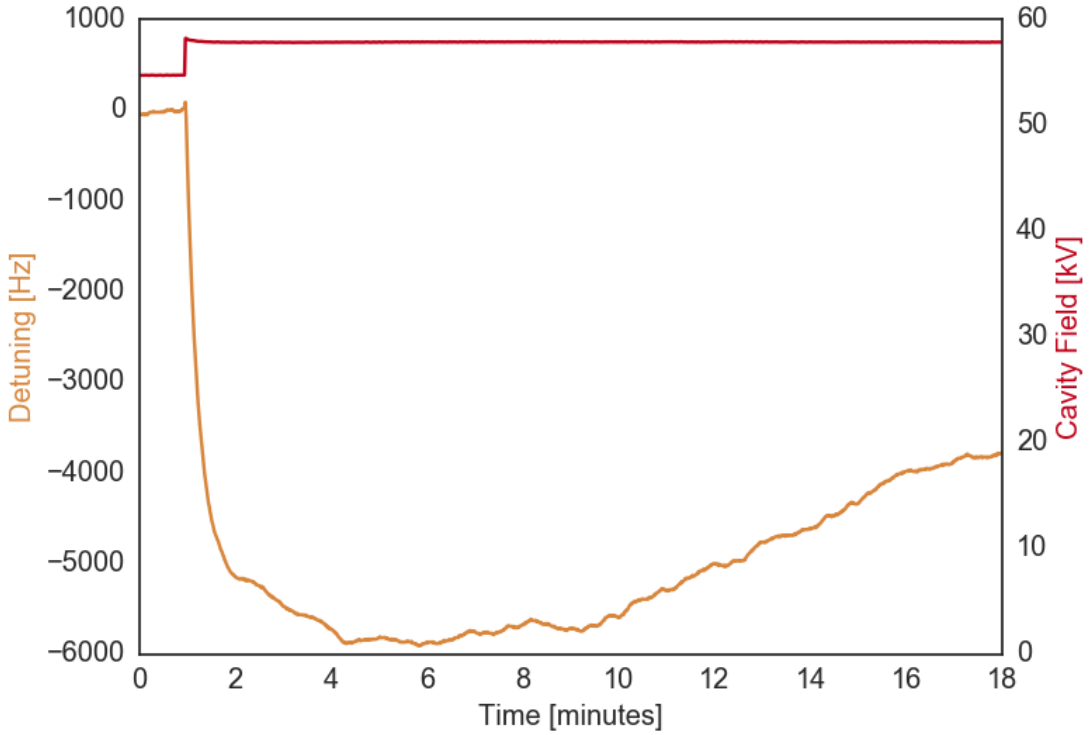


Figure 4.55: Example of uncontrolled detuning in CW mode under a small change in cavity field (55 kV to 58 kV).

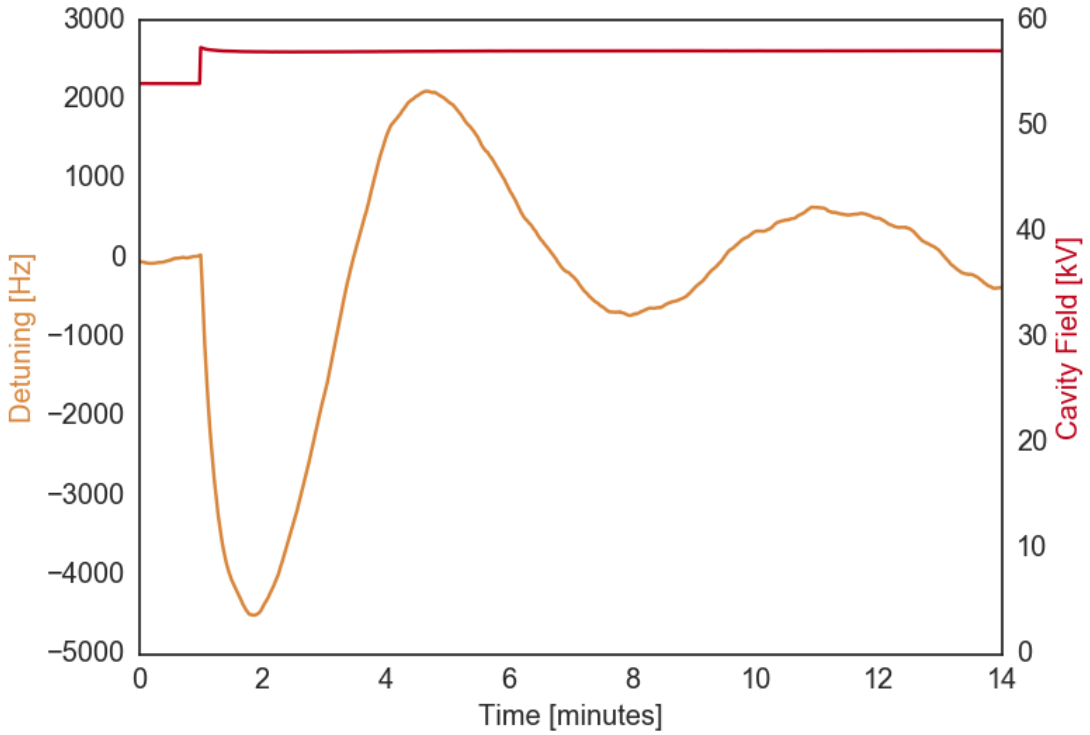


Figure 4.56: PI resonant frequency control under CW operation under a small change in cavity field. This is roughly the same step as used for the uncontrolled response shown in Figure 4.55.

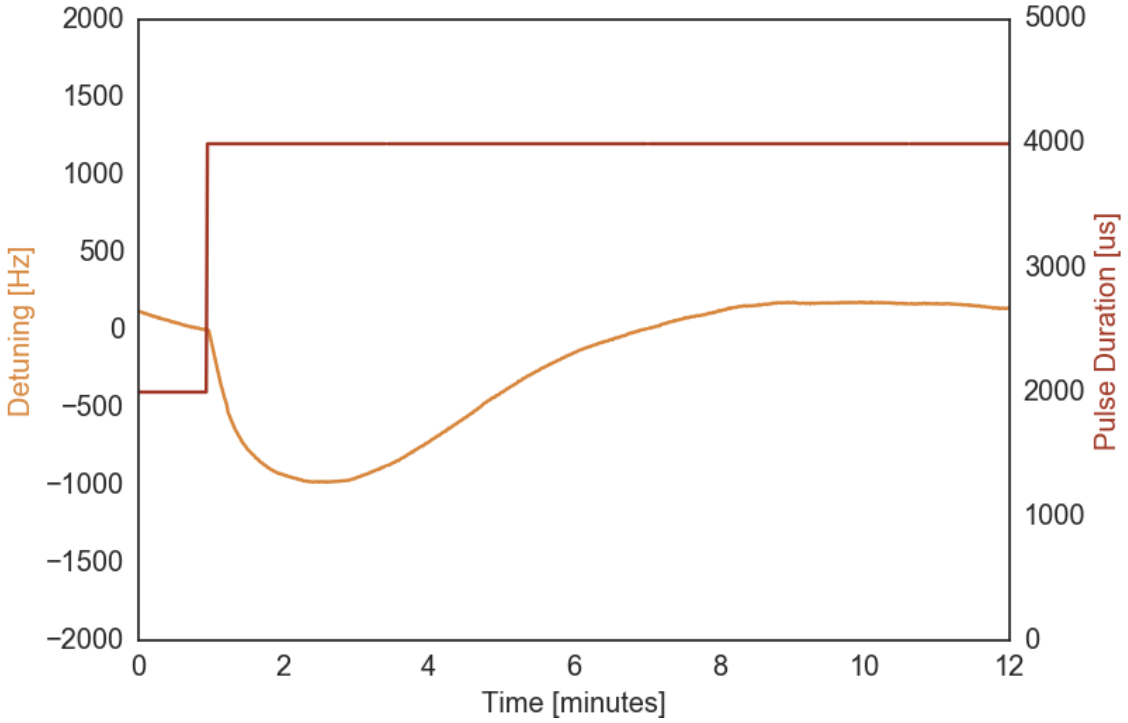


Figure 4.57: PI frequency control during pulsed RF operation for a 2 ms increase in pulse duration and a cavity field of 65 kV.

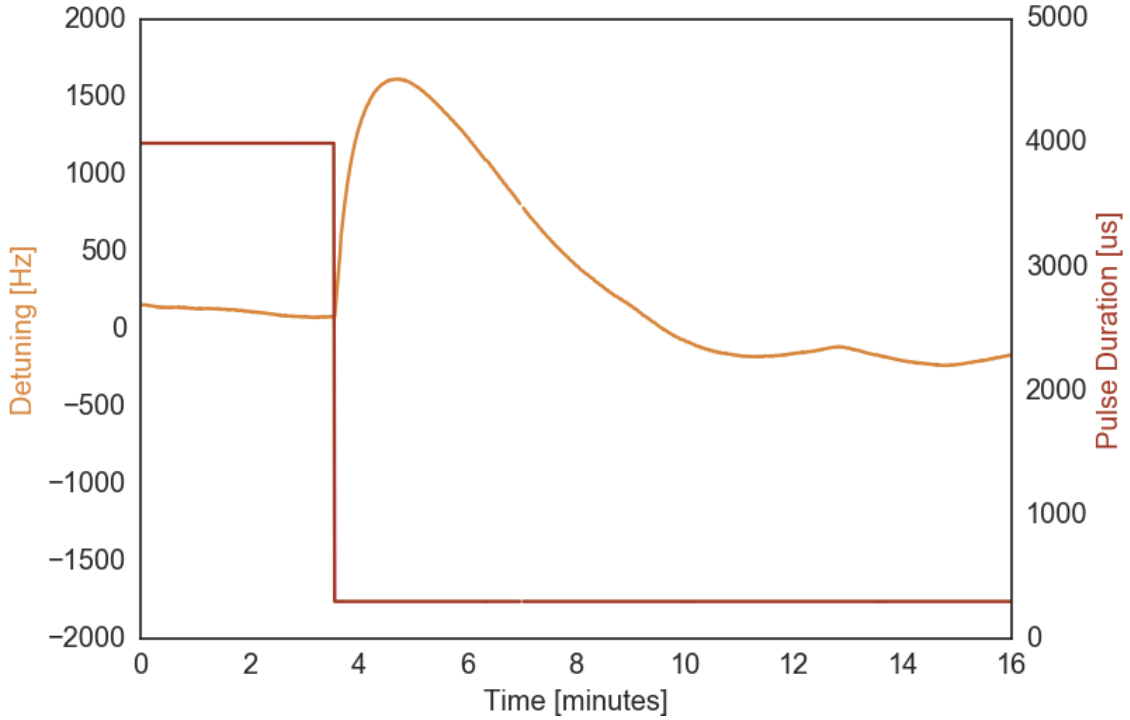


Figure 4.58: PI frequency control during pulsed RF operation for a decrease in pulse duration.

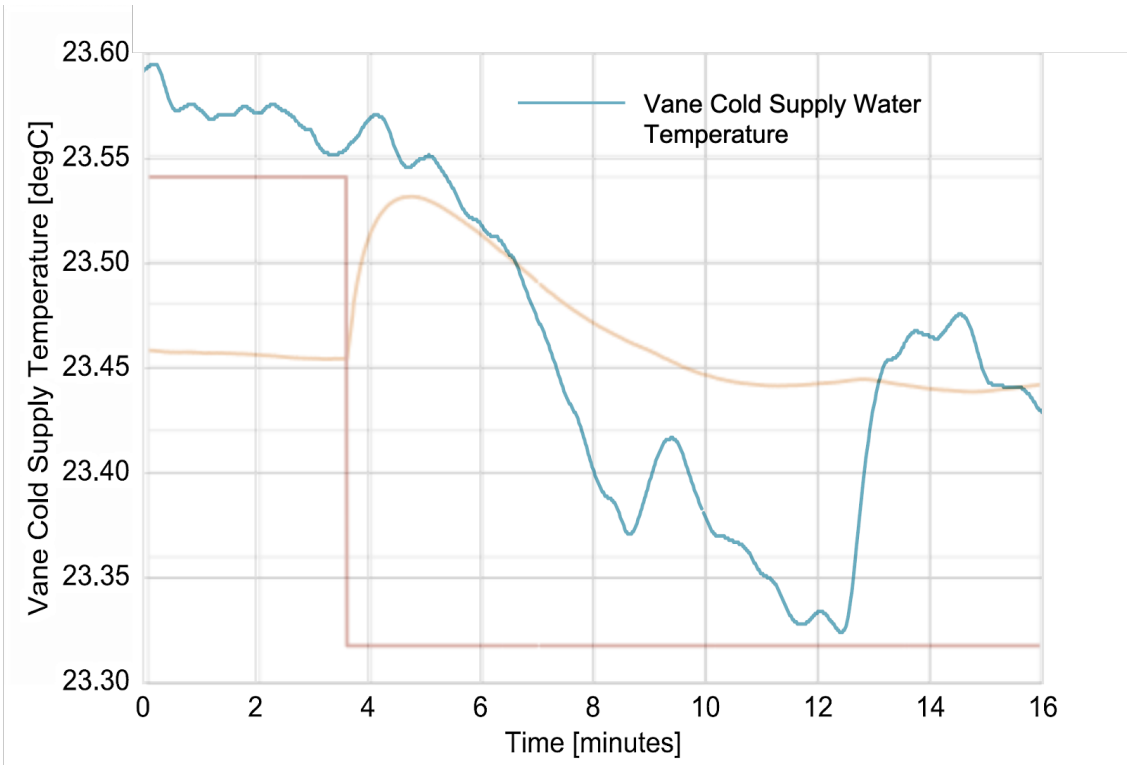


Figure 4.59: Overlay of vane supply temperature onto the same data as shown in Figure 4.58 showing the impact around the 13 minute mark of the abrupt disturbance in temperature.

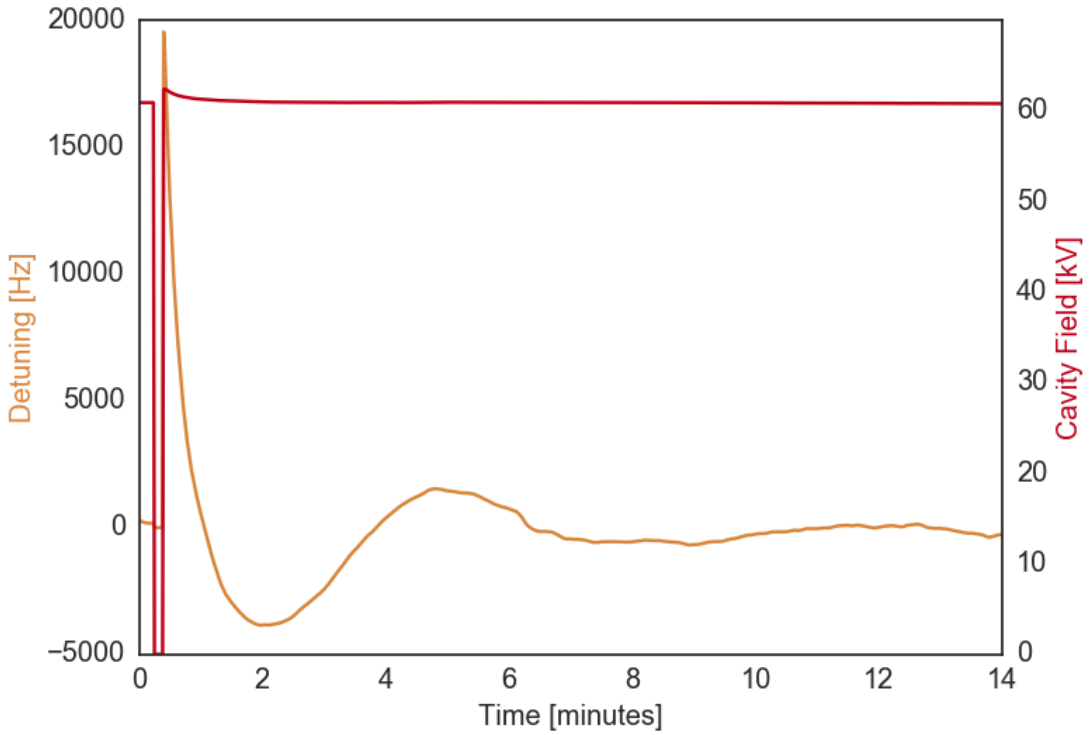


Figure 4.60: Frequency recovery from a 10-second RF trip at 60 kV using PI control over the vane flow valve.

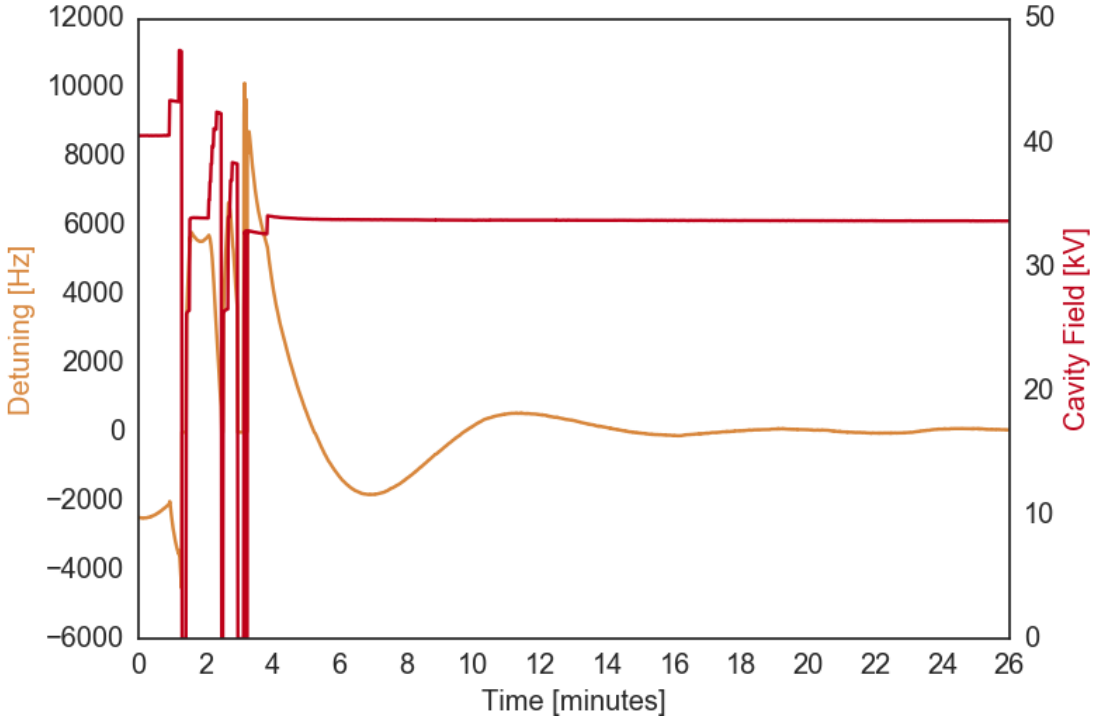


Figure 4.61: Frequency recovery from a compound RF trip with control over the vane flow valve. Conditions are similar to that shown in Figure 4.60. This is for a 121 seconds total trip time, with the most recent trip being 17 seconds. The PI controller brings the resonant frequency to within 3kHz (suitable for GDR mode) in approximately 60 seconds.

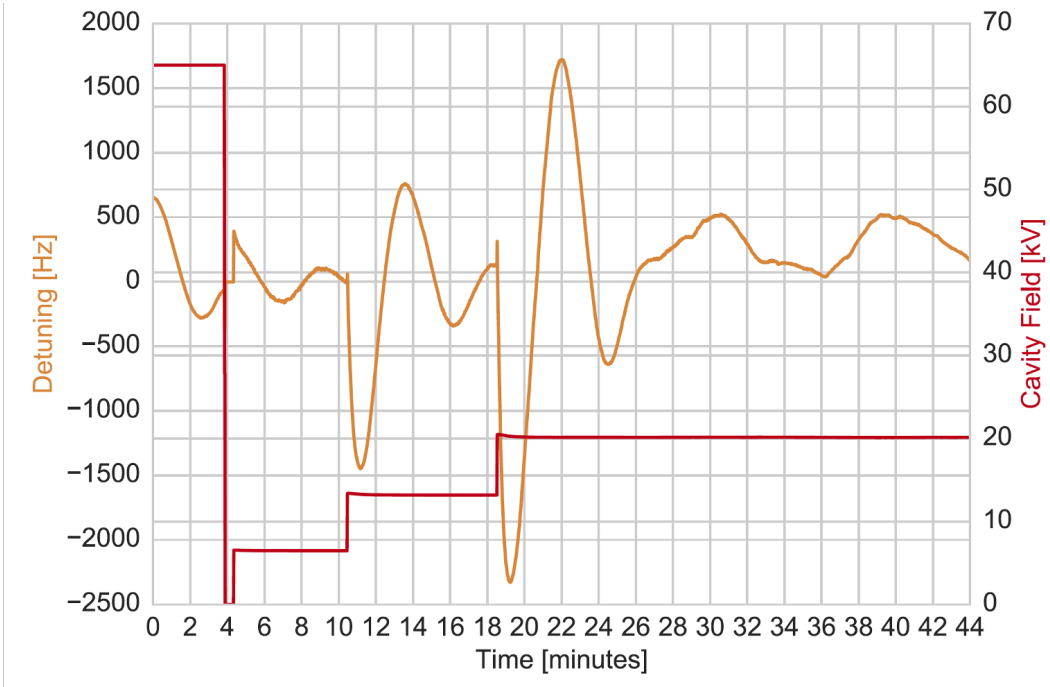


Figure 4.62: PI loop under CW operation with changing RF cavity fields.

Figure 4.63 and Figure 4.64 show examples of the resonant frequency control with the PI loop for recovery from a 10-second trip and for cold startup. The performance of the resonant frequency control system under PI control in pulsed mode was also evaluated with respect to trip recovery times. Figure 4.65 shows examples of trip recovery times compared to specification and stretch goals (recovery in 10 $\times$ and 2 $\times$ trip duration). This is for "long" trips where the RF itself takes a few minutes to recover. In most cases, GDR mode was achieved within the specified time, and in many cases was below the stretch goal time. The RF recovery time out of 44 trips was on average 3.3 minutes (2.1 minutes median), and after RF recovery the resonant frequency recovery took on average 7.8 minutes (1.6 minutes median). The major reason for failure to meet specification in some cases was large overshoot. Based on results from FAST, this overshoot is something MPC control should be able to alleviate.

Figure 4.66 shows examples from less common "medium' length trips where the RF takes less than a minute to recover. Only 3 out of the 73 trips analyzed comprised this type of trip. MPC should similarly be able to bring these better into specification. Shorter trips (e.g. a few seconds) do not require the resonant frequency controller to engage and can be handled entirely by the LLRF feedback, with recovery well within specification. Out of the 73 trips examined, 26 were this type of trip.

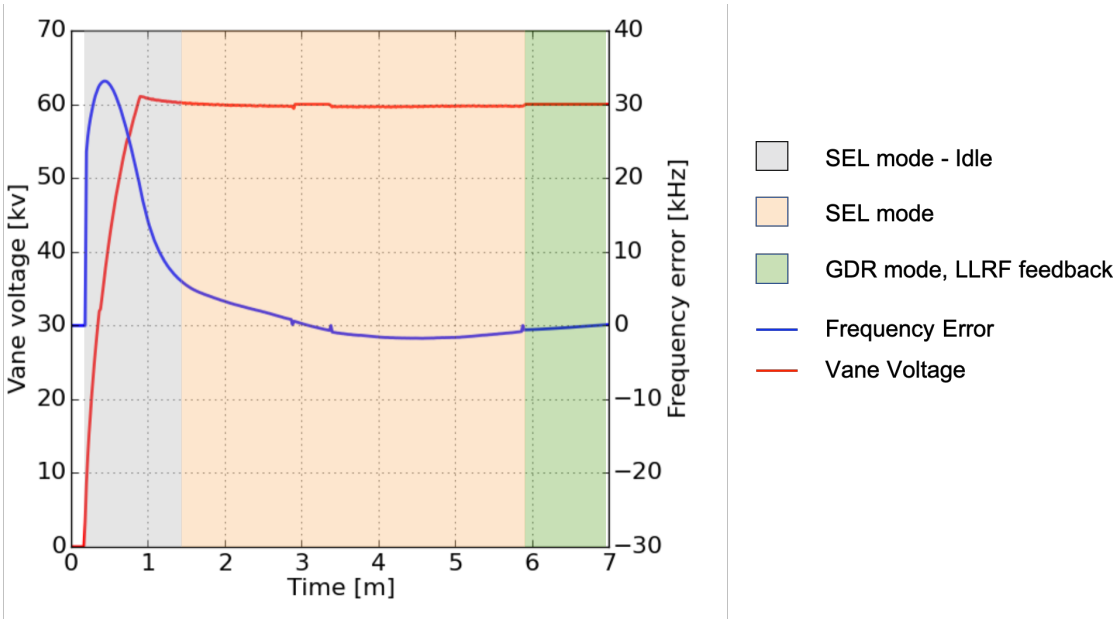


Figure 4.63: RF for recovery from a 10-second trip in pulsed mode under PI resonant frequency control on the vanes. The LLRF system starts in SEL mode. During the RF ramp, the resonance controller is not activated in PI mode. The resonant frequency controller is then activated. When sufficiently low detuning is reached, the resonant frequency controller switches LLRF system into GDR mode and turns on the LLRF feedback.

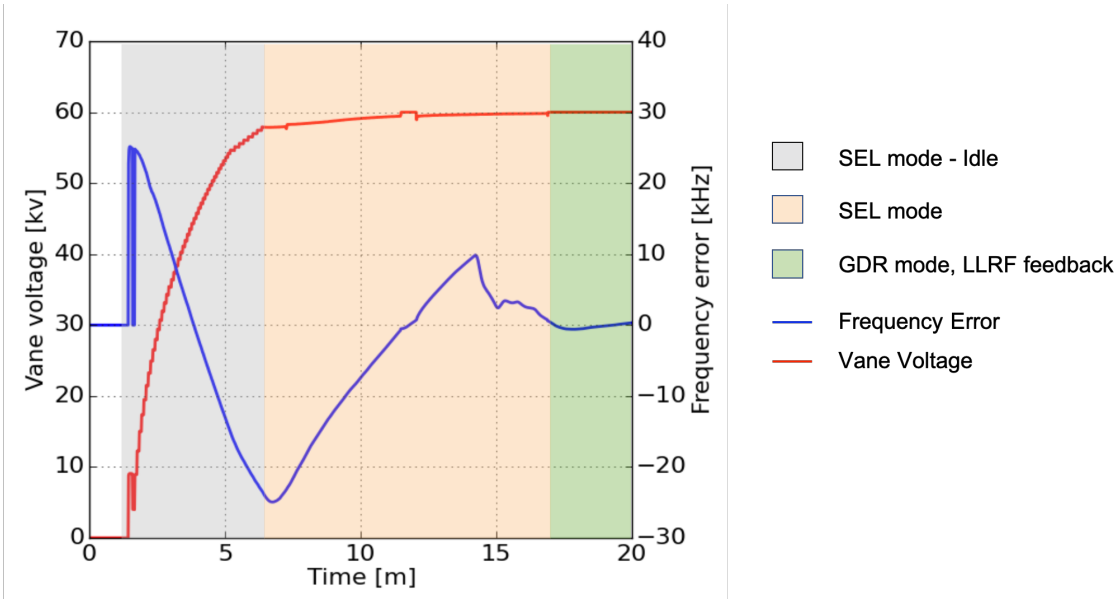


Figure 4.64: RF startup under PI resonant frequency control on the vanes. Note that a manual correction was made to an incorrect skid water temperature set point around the 14.5 minute mark.

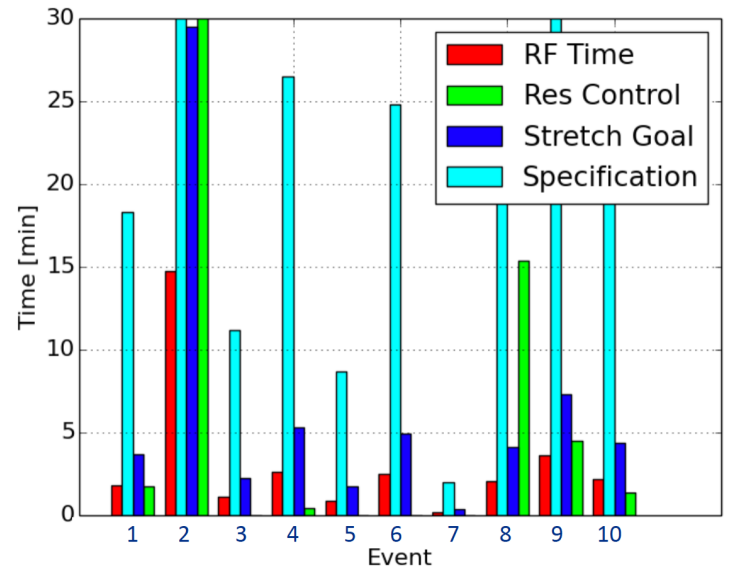


Figure 4.65: PI loop performance compared to specified and stretch goals for long trips (i.e. where RF was tripped off for more than a few minutes). This shows recovery times are typically within the specification for the RFQ.

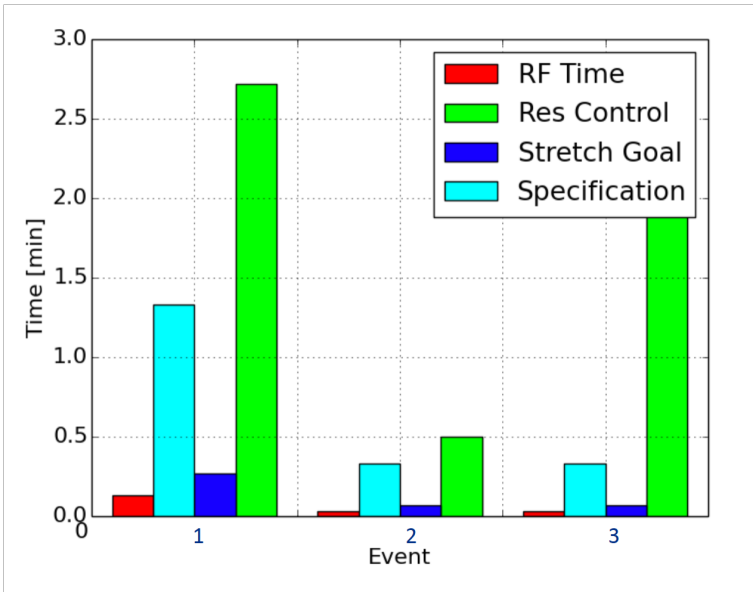


Figure 4.66: PI loop performance compared to specified and stretch goals for medium length trips (i.e. where the RF was tripped off for less than a minute). The main reason for failure to meet specification was large overshoot. These are more rare (e.g. 3 out of 73 total trips). Based on results from FAST, this is something MPC should be able to improve.

Early attempts to commission MPC proved to be a challenge due to the computational burden of the approach relative to the relatively limited computational power available on the Fermilab control network computers. This is an important lesson learned, as it highlights the fact that accelerator control system infrastructure (1) needs to be improved to support some ML applications and (2) should be taken into account from the start of the systems engineering and design process. Unfortunately, none of the neural network architectures, even after substantial attempts to prune the number of inputs and reduce the model size, were able to run on the actual control system infrastructure at the PIP-II Injector Test. Some initial attempts were made to calibrate the analytic model to use with MPC, but the predictions were not sufficiently accurate. With regard to purchasing a dedicated computer for this task, concerns about cost and availability of time for maintenance of such a system were raised. Another option was to connect via VPN with Fermilab's on-site HPC resources temporarily. While none of these could be attempted in the time before the long installation shutdown of the PIP-II injector test that occurred directly after resonant frequency control system commissioning, this should be revisited in the future.

4.4.9 Conclusion and Future Outlook for PIP-II RFQ

In this work, we developed and validated a general analytic modeling approach for combined cooling and RF system responses of normal-conducting copper accelerating cavities. The high execution speed of the model makes it suitable for design and control studies, and its validation against measured data from the RFQ provides confidence that it can be used for the design of cooling and control systems for future RFQs and other normal-conducting copper cavities. The lack of a readily available tool for this at the start of the PIP-II RFQ cooling and control system design is what prompted this part of the work. In part with help from this model and previous experience from FAST, we aided in the system design for the PIP-II RFQ water-cooling and resonant frequency control hardware based on control requirements. We also used it to conduct initial verification that the RFQ would be able to operate in both pulsed and CW modes while maintaining the required resonant frequency.

Next we designed, built, and commissioned a general framework that meets the major operational needs of the PIP-II RFQ for resonant frequency control (e.g. including many practical tasks such as automatic switching between SEL and GDR mode). This framework also represented a first for Fermilab with respect to putting in infrastructure to enable dedicated python-based control over parts of the accelerator system. We also aided in commissioning of the RFQ and water system itself, performed initial characterization work for the RFQ, and commissioned a PID controller for the resonant frequency for both pulsed and CW RF operation.

Within the control framework we built, the PID controller is capable of providing reasonable automatic recovery from short RF trips, and it also automatically detects when the cavity needs to be switched between SEL and GDR mode to protect the RFQ from reflected power and prevent machine trips due to detuning. It performs adequately across a wide range of RF heating, including both pulsed and CW mode.

During this process, we also gathered data for and constructed a neural network-based model of the resonant frequency response of the system to various disturbances and controlled inputs. The model was shown to be substantially more accurate than the analytic model, and the aim was to

use this model as part of MPC regulation of the RFQ. To that end, we also showed that the neural network model was able to make accurate predictions over a reasonable future time horizon relative to the last acquired data point. Unfortunately, MPC was not able to be fully commissioned on the machine. Initial designs using the neural network model were too computationally expensive to run on the control system hardware, and there was limited time before a long system shut-down to test ways of simplifying the setup. As a starting point, the analytic model could also possibly be used in MPC, but due to its lower predictive power than the neural network model is not expected to perform well. However, this could not be tested on the machine.

In addition, the funding for this work was aimed primarily at ensuring the RFQ would be able to operate within specifications for both pulsed and CW mode, along with engineering design support and commissioning of the RFQ. After commissioning the RFQ resonance control system general control interface, the PIP-II project was unable to fund us for further dedicated studies, nor the control computer upgrade that would be needed for MPC. Future testing of MPC may be possible with improved computer hardware (i.e. a more computationally powerful computer connected to the ACNET control system).

While there are many open questions to address, it remains the case that fully exploiting the potential of both MPC and other algorithms (e.g. neural network-based reinforcement learning) would be an interesting area to pursue for this problem. There are clear areas in the PI performance limitations for the RFQ that point toward possible benefits of adopting MPC for this system (e.g. reducing overshoot, as was demonstrated at FAST for the RF gun). In contrast to PID, MPC may provide compensation for water temperature fluctuations and enable finer control over the resonant frequency by leveraging both the wall and vane responses, but this needs to be tested. Automated RF recovery could also be added (e.g. setting an optimal RF ramp, as opposed to a standard linear one) to speed up startup.

4.5 Conclusion and Future Outlook for Neural Network-based Modeling and Control of Resonant Frequency Responses of RF Cavities

In the above work with FAST we demonstrated that MPC is a reasonable candidate to adopt for improving resonant frequency control in copper accelerating cavities. PID control is still the most common controller choice for this task. We also demonstrated that even simple neural network models can provide improved predictive performance for the resonant frequency, in comparison with analytic models for some copper RF systems (notably, the RFQ). As was found with the RFQ, in practice the development of analytic models for a complicated system like the RFQ can be extremely time-consuming.

However, despite the appeal of learned models for greater accuracy, during commissioning stages where many changes are being made to the machine a learned model may be a less reliable choice. Substantial difficulties were encountered chasing down instrumentation changes and un-documented calibration changes for both the FAST injector and PIP-II RFQ. For example, several weeks were spent debugging RFQ modeling, after which it was discovered that the difficulty was due to inconsistencies in the data created by an un-logged change to a calibration that affected the resonant frequency calculation. As a silver lining, this is an important set of lessons learned with regard to calibrations and machine logs. It has been used as an example (e.g. at recent workshops for ML in accelerators) to help advise other groups and facilities to avoid making the same mistakes; this includes, for example, helping to provide an example of pitfalls that the design of data acquisition systems for new accelerators might avoid.

The focus of this work was primarily on providing solutions that could be used to address the immediate operational needs of the FAST gun and PIP-II RFQ. Future work focused on applied controls research should compare MPC with analytic models and neural network-based models, along with control using reinforcement learning. The PIP-II RFQ would be a nice test bed for this.

The modeling and control infrastructure developed as part of this work can be directly leveraged to support such studies.

Complicated systems like the RFQ that need to be tightly controlled while taking into account system evolution are ubiquitous in modern accelerators. New accelerators (PIP-II, LCLS-II) will also use superconducting RF cavities, where the resonant frequency is extremely sensitive to small changes in cavity shape (even from the Lorentz force induced by the beam itself [231]) and disturbances in the RF and cryogenic cooling systems. Cryogenic cooling systems are also substantially more nonlinear than water-cooling systems, and can exhibit chaotic behavior. Some of the approaches initially investigated here (with, of course, further work first on normal-conducting cavities) may eventually be useful for achieving fine control over the combined RF and cryogenic systems of superconducting RF accelerators.

Chapter 5

Conclusions

We have described several studies that highlight potential application areas of neural networks to modeling and control of particle accelerators. At the time of each of these works (mid 2012 thru early 2017), the neural network-based approaches covered here were quite novel and have gone on to seed major ML-focused efforts in accelerators (for example, see [187, 232], which all were influenced directly by the work presented here). In summary we demonstrated important applications in the following areas:

- **Speeding up simulations**: We demonstrated that neural networks can be reliably used to produce fast-executing representations of computationally expensive accelerator simulations that include nonlinear beam effects. In particular, we showed this for two injector systems (including guns and downstream cavities and/or focusing elements). This enables higher-fidelity simulators to be used online during machine operation, could aid offline experiment planning, and also could help speed up start-to-end offline design studies by replacing upstream components with a neural network surrogate model and linking it to downstream simulations. This approach is now a major focus of an online modeling effort at SLAC. See [201, 202, 233].

- **Virtual diagnostics**: we introduced the concept of an ML-based virtual diagnostic to provide non-invasive estimates of destructive beam measurements, including for image-based diagnostics. This work built on other concepts for simulation-based virtual diagnostics [163, 185], and was conducted concurrent to related results on LCLS user data to fill in missing photon science data [186] (focused on vector and scalar predictions). In this work, we demonstrated that a neural network can provide accurate estimates of complicated image-based diagnostic outputs (in our case, these were images from a multi-slit emittance diagnos-

tic for the FAST low-energy beamline). This approach was later expanded on experimentally and in simulation for applications in phase space prediction at SLAC [187]. See [234, 235].

- **Incorporating image-based diagnostics directly into neural network machine models with Convolutional Neural Networks**: we showed the first proof-of-concept study in using image-based diagnostics (such as VCC images of the transverse laser profile) with neural network-based machine models, so as to directly incorporate the full complexity of the upstream information available to the model. At the time of the study (2016), this was an important first step toward demonstrating the feasibility of this approach. See [233].

- **Model updating with measured data (transfer learning)**: we demonstrated transfer learning between simulation data and measured data for an accelerator model. This improves the accuracy of the model, and although we did not investigate this aspect in great detail, this could reduce the burden of acquiring substantial amounts of measured training data from an operational accelerator. See [234, 235].

- **Inverse modeling and reinforcement learning for fast switching between operating modes**: we conducted a proof-of-concept simulation study showing that a neural network inverse model can be used to quickly switch between operating modes in a compact FEL (in this case, picking an appropriate set of injector and matching quadrupole settings to achieve a particular beam energy and match into the undulator). We further showed that this model can be updated in a reinforcement learning control scheme as it explores the parameter space. This inverse modeling approach to provide a warm start to local optimizers was later experimentally investigated at LCLS [232]. See [201, 202].

- **Model predictive control for temperature and resonant frequency control in normal conducting cavities**: we demonstrated that MPC is a viable replacement for the standard PID control that is commonly used in the accelerator community for control of water-regulated copper accelerating cavities. In our tests with the FAST RF gun, overshoot was minimized (saving forward RF power) and the settling time before stable operation was reduced by a

factor of 5. This is important because any time the beam energy is changed, the RF power is changed and the system is disturbed. In addition to improving stability for regular operation, this may also improve the minimum time of interruption due to accelerator enclosure access, where the RF has to be powered off and subsequently the system must be returned to full power and allowed to settle before resuming normal operations. We also demonstrated that a neural network model can accurately predict the resonant frequency for the combined RF and cooling system of a high-power RFQ over long time-scales and with a variety of average RF power levels. This model is substantially more accurate than an analytic treatment of the system. These approaches may also prove useful for future control of superconducting RF cavities and their associated cryogenic cooling systems, which exhibit substantial nonlinear behavior and are very difficult to control. See [230, 236–238].

- **Analytic modeling and control infrastructure for resonant frequency regulation in RF accelerating cavities**: while it is not an ML contribution, the analytic modeling and controls infrastructure developed for the RFQ was a substantial portion of the work conducted. The resulting framework provides a solid foundation for making MPC viable for dedicated use for the PIP-II RFQ, and it could be extended to other systems inside and outside of Fermilab. The analytic model could also be readily used for design studies of future RF/cooling systems (including other combined pulsed/CW RF systems). High-power RFQs are becoming increasingly common in upcoming accelerator designs, and the analytic modeling framework developed in this work may aid future efforts for resonant frequency controller and cooling system design. See [239, 240].

Finally, although it is an indirect contribution, the lessons learned from this work with regard to pitfalls encountered in timing and archiving data acquisition systems was used to help inform those designing new control / archive infrastructures for accelerators (i.e. with an eye toward incorporating ML-friendly features). In addition, this work was presented at numerous venues in the accelerator community as it was beginning to show interest in AI/ML, and these helped to introduce the concepts above more broadly.

These approaches also contributed to tutorial demos at the first workshop on AI/ML for accelerators (see [176] for a summary of that workshop). The tutorials were in some cases used directly by attendees for similar problems, thus helping to popularize these approaches. In addition, the survey article [237] written during this work helped re-introduce the particle accelerator community to neural network-based approaches for modeling and control.

This work focused on generating proof-of-concept results for neural network-based modeling and control of small-scale systems. Looking forward, to move from proof-of-concept demonstrations to dedicated deployment in accelerator operations, much R&D is needed. This includes addressing model retraining over time, robustness to anomalous machine behavior or equipment failure, thorough benchmarking of different algorithms, the inclusion of robust uncertainty measures (which is particularly challenging for neural network-based approaches), and methods for safe and efficient exploration of the large parameter spaces in accelerators.

A major open question that needs to be addressed is how to best determine when a neural network model or controller has moved outside its range of validity relative to the present machine operating condition, and how best to update it in a robust manner. In this case, uncertainty predictions may be used to decide when the model needs to be re-trained. Uncertainty predictions are also essential in model-based optimization tools if they are to be used safely in accelerator operations (i.e. "safe optimization"). For example, if model predictions are used for accelerator optimization, a prediction of the uncertainty is critical for evaluating the risk of allowing the machine to evolve into a new state vs. the reward from finding new optimal operating points (which may result in intermittent loss of beam if adequate care is not taken). Methods to incorporate uncertainty predictions into neural networks (e.g. see [191]) are a present topic of active research in the deep learning community. Scaling up to larger parameter spaces efficiently is also another area of work that will need attention if these techniques are to be used for larger-scale particle accelerator systems. Finally, for optimization, the trade-off between the cost of training neural network models or control policies (including data generation, data cleaning, and model training) and the utility gained needs to be compared with other methods, such as Bayesian optimization with Gaussian

Processes, for different kinds of accelerator systems. Finally, at present the approaches described here do not make full use of our rich knowledge of accelerator physics; approaches should be developed that instead make judicious use of this physics knowledge and ML together (for example, to improve sample efficiency and prediction robustness). These areas of research are now actively being investigated for particle accelerators.

In conclusion, the work presented here made significant conceptual contributions to neural network-based modeling and control of particle accelerators, along with first demonstrations of the proposed approaches. This work has spurred further promising research in this area, and many of the concepts introduced here are now being developed for dedicated use in accelerator operation.

Bibliography

[1] Accelerators for America's Future, Department of Energy Technical Report, 2010.

[2] SLAC National Accelerator Laboratory, https://www6.slac.stanford.edu/.

[3] Fermi National Accelerator Laboratory, https://www.fnal.gov/.

[4] CERN, https://home.cern/.

[5] M. Ball, A. Burov, B. Chase, A. Chakravarty, A. Chen, S. Dixon, J. Edelen, A. Grassellino, D. Johnson, S. Holmes, S. Kazakov, A. Klebaner, I. Kourbanis, A. Leveling, O. Melnychuk, D. Neuffer, T. Nicol, J. F. Ostiguy, R. Pasquinelli, D. Passarelli, L. Ristori, W. Pellico, J. Patrick, L. Prost, I. Rakhno, A. Saini, W. Schappert, A. Shemyakin, J. Steimel, V. Scarpine, A. Vivoli, A. Warner, V. Yakovlev, P. Ostroumov, and Z. Conway. The PIP-II Conceptual Design Report. 2017.

[6] P. Emma, R. Akre, J. Arthur, R. Bionta, C. Bostedt, J. Bozek, A. Brachmann, P. Bucksbaum, R. Coffee, F. J. Decker, Y. Ding, D. Dowell, S. Edstrom, A. Fisher, J. Frisch, S. Gilevich, J. Hastings, G. Hays, Ph. Hering, Z. Huang, R. Iverson, H. Loos, M. Messerschmidt, A. Miahnahri, S. Moeller, H. D. Nuhn, G. Pile, D. Ratner, J. Rzepiela, D. Schultz, T. Smith, P. Stefan, H. Tompkins, J. Turner, J. Welch, W. White, J. Wu, G. Yocky, and J. Galayda. First lasing and operation of an ångstrom-wavelength free-electron laser. *Nature Photonics*, 4(9):641–647, 2010.

[7] A. Marinelli, J. MacArthur, P. Emma, M. Guetg, C. Field, D. Kharakh, A. A. Lutman, Y. Ding, and Z. Huang. Experimental demonstration of a single-spike hard-x-ray free-electron laser starting from noise. *Applied Physics Letters*, 111(15):151101, 2017.

[8] G. Ha, M. H. Cho, W. Gai, K.-J. Kim, W. Namkung, and J. G. Power. Perturbation-minimized triangular bunch for high-transformer ratio using a double dogleg emittance exchange beam line. *Phys. Rev. Accel. Beams*, 19:121301, Dec 2016.

[9] Xiufang Tian, Kun Liu, Yong Hou, Jian Cheng, and Jiandong Zhang. The evolution of proton beam therapy: Current and future status. *Molecular and clinical oncology*, 8(1):15–21, 01 2018.

[10] Chungming Chu, M. Woodley, R. Iverson, P. Krejcik, G. White, Juhao Wu, and Quan Gan. XAL-based applications and online model for LCLS. In *Proc. of PAC*, page FR5REP022, 2010.

[11] Thomas Pelaia. Open XAL Status Report 2013. In *Proc. of IPAC*, page MOPWO086, 2013.

[12] K. Wille and J. McFall. *The Physics of Particle Accelerators: An Introduction*. Oxford University Press, 2000.

[13] M. Reiser. *Theory and Design of Charged Particle Beams*. Wiley Series in Beam Physics and Accelerator Technology. Wiley, 2008.

[14] E. Forest. *Beam Dynamics*. The Physics and Technology of Particle and Photon Beams. Taylor & Francis, 1998.

[15] S Y Lee. *Accelerator Physics*. World Scientific, 3rd edition, 2011.

[16] Alexander Wu Chao, Karl Hubert Mess, Maury Tigner, and Frank Zimmermann. *Handbook of Accelerator Physics and Engineering*. World Scientific, 2nd edition, 2013.

[17] Alexander Wu Chao. *Physics of collective beam instabilities in high energy accelerators*. Wiley, New York, NY, 1993.

[18] T. Rao and D.H. Dowell. *An Engineering Guide to Photoinjectors*. Createspace Independent Pub, 2013.

[19] E.L. Saldin, E.V. Schneidmiller, and M.V. Yurkov. *The Physics of Free Electron Lasers*. Advanced Texts in Physics. Springer Berlin Heidelberg, 2013.

[20] Z Huang and K J Kim. A review of x-ray free-electron laser theory. *Phys. Rev. ST Accel. Beams*, 10, 2007.

[21] C. Pellegrini, A. Marinelli, and S. Reiche. The physics of x-ray free-electron lasers. *Rev. Mod. Phys.*, 88:015006, Mar 2016.

[22] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[23] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

[24] Simon S. Haykin. *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition, 2009.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[26] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., USA, 1995.

[27] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[28] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[29] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, 2016.

[30] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[31] Burr Settles. Active learning literature survey. Technical report, Computer Sciences Technical Report 1648. University of Wisconsin–Madison., 2010.

[32] S. Das, W. Wong, T. Dietterich, A. Fern, and A. Emmott. Incorporating expert feedback into active anomaly discovery. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 853–858, 2016.

[33] Fredrik Olsson. A literature survey of active machine learning in the context of natural language processing. Technical report, SICS Technical Report T2009:06., 2009.

[34] Li Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[35] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.

[36] George Cybenko. *Analysis and Design of Neural Networks*. PN, 1992.

[37] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

[38] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[39] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10)*, 2010.

[40] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78 – 80, 2004.

[41] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659 – 1671, 1997.

[42] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E. Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[43] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3:e2, 2014.

[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[45] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[46] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[47] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[48] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

[49] C. Hentschel, T. P. Wiradarma, and H. Sack. Fine tuning cnns with scarce training data — adapting imagenet to art epoch classification. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3693–3697, 2016.

[50] Zongwei Zhou, Jae Shin, Lei Zhang, Suryakanth Gurudu, Michael Gotway, and Jianming Liang. Fine-tuning convolutional neural networks for biomedical image analysis: Actively and incrementally. *Proceedings. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2017.

[51] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:1655–1668, 2019.

[52] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013.

[53] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks, 2017.

[54] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2342–2350. JMLR.org, 2015.

[55] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[56] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. In *Proc. ICANN'99, Int. Conf. on Artificial Neural Networks*, pages 850–855, Edinburgh, Scotland, 1999. IEE, London.

[57] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.

[58] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Con-*

ference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[59] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. 2016. cite arxiv:1609.03499.

[60] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018.

[61] Roni Mittelman. Time-series modeling with undecimated fully convolutional neural networks. *CoRR*, abs/1508.00317, 2015.

[62] Anastasia Borovykh, Sander M. Bohte, and Cornelis W. Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv: Machine Learning*, 2017.

[63] Richard H. R. Hahnloser, Rahul Sarpeshkar, Misha A. Mahowald, Rodney J. Douglas, and H. Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, 2000.

[64] Richard H. R. Hahnloser and H. Sebastian Seung. Permitted and forbidden sets in symmetric threshold-linear networks. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, page 199–205, Cambridge, MA, USA, 2000. MIT Press.

[65] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress.

[66] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE International Conference on Computer Vision (ICCV 2015)*, 1502, 02 2015.

[68] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, page 9–50, Berlin, Heidelberg, 1998. Springer-Verlag.

[69] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1996.

[70] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Inc., New York, NY, USA, 1996.

[71] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10(2):99–127, June 2002.

[72] E. Meier, S.G. Biedron, G. LeBlanc, M.J. Morgan, and J. Wu. Development of a combined feed forward-feedback system for an electron linac. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 609(2):79 – 88, 2009.

[73] E.K.P. Chong and S.H. Zak. *An Introduction to Optimization, 4th Ed.* John Wiley Sons, Inc., 2013.

[74] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *ArXiv*, abs/1212.5701, 2012.

[75] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[76] Igor Gitman, Hunter Lang, Pengchuan Zhang, and Lin Xiao. Understanding the role of momentum in stochastic gradient methods. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9633–9643. Curran Associates, Inc., 2019.

[77] Sebastian Ruder. An overview of gradient descent optimization algorithms., 2016. arxiv:1609.04747.

[78] Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, 1994.

[79] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

[80] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[81] Hiroaki Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4, 1990.

[82] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.

[83] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabás Póczos, and Eric P. Xing. Neural architecture search with bayesian optimisation and optimal transport. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 2020–2029, Red Hook, NY, USA, 2018. Curran Associates Inc.

[84] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[85] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 598–605. Morgan-Kaufmann, 1990.

[86] Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018.

[87] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[88] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[89] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1050–1059. JMLR.org, 2016.

[90] J. Kennedy and R. Eberhart. Particle swarm optimization. *Proc. of ICNN'95 - International Conference on Neural Networks.*, 1995.

[91] I. V. Bazarov and C. K. Sinclair. Multivariate optimization of a high brightness dc gun photoinjector. *Phys. Rev. ST Accel. Beams*, 8:034202, 2005.

[92] X. Pang and L.J. Rybarcyk. Multi-objective particle swarm and genetic algorithm for the optimization of the LANSCE linac operation. *NIM A.*, 741:124–129, 2014.

[93] Alicia Hofler, Balsa Terzic, Matthew Kramer, Anton Zvezdin, Vasiliy Morozov, Yves Roblin, Fanglei Lin, and Colin Jarvis. Innovative applications of genetic algorithms to problems in accelerator physics. *Phys. Rev. ST Accel. Beams*, 16(1):010101, 2013.

[94] M. Borland, V. Sajaev, L. Emery, and A. Xiao. Multi-objective direct optimization of dynamic acceptance and lifetime for potential upgrades of the Advanced Photon Source. Technical Report ANL/APS/LS-319, Advanced Photon Source, Argonne National Laboratory, 2010.

[95] Alexander Scheinker, Xiaoying Pang, and Larry Rybarcyk. Model-independent particle accelerator tuning. *Phys. Rev. ST Accel. Beams*, 16:102803, Oct 2013.

[96] A. Scheinker and M. Krstić. Minimum-seeking for CLFs: Universal semiglobally stabilizing feedback under unknown control directions. *IEEE Transactions on Automatic Control*, 58(5):1107–1122, May 2013.

[97] A. Scheinker, X. Huang, and J. Wu. Minimization of betatron oscillations of electron beam injected into a time-varying lattice via extremum seeking. *IEEE Transactions on Control Systems Technology*, 26(1):336–343, Jan 2018.

[98] Xiaobiao Huang, Jeff Corbett, James Safranek, and Juhao Wu. An algorithm for online optimization of accelerators. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 726:77 – 83, 2013.

[99] Xiaobiao Huang and James Safranek. Online optimization of storage ring nonlinear beam dynamics. *Phys. Rev. ST Accel. Beams*, 18:084001, Aug 2015.

[100] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 1965.

[101] C. Rasmussen and C. Williams. Bayesian Regression and Gaussian processes. *Gaussian Processes for Machine Learning*, 2006.

[102] M. McIntire, T. Cope, D. Ratner, and S. Ermon. Bayesian optimization of fel performance at lcls. In *Proceedings of the IPAC'16*, pages 2972–2974, 2016.

[103] Y. Chung, G. Decker, and K. Evans. Closed orbit correction using singular value decomposition of the response matrix. In *Proceedings of International Conference on Particle Accelerators*, pages 2263–2265 vol.3, 1993.

[104] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[105] M. E. Dean, C. D. Schuman, and J. D. Birdwell. Dynamic adaptive neural network array. In *13th International Conference on Unconventional Computation and Natural Computation (UCNC)*, pages 129–141, London, ON, July 2014. Springer.

[106] Clément Farabet, Yann Lecun, Koray Kavukcuoglu, Berin Martini, Polina Akselrod, Selcuk Talay, and Eugenio Culurciello. *Large-Scale FPGA-Based Convolutional Networks*, page 399–419. Cambridge University Press, 2011.

[107] C. D. Schuman, J. S. Plank, A. Disney, and J. Reynolds. An evolutionary optimization framework for neural networks and neuromorphic architectures. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 145–154, 2016.

[108] Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, and James S. Plank. A survey of neuromorphic computing and neural networks in hardware. *ArXiv*, abs/1705.06963, 2017.

[109] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre-Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron Courville, Yann N.

Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman, Laurent Dinh, Mélanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian Goodfellow, Matt Graham, Caglar Gulcehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrancois, Simon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert T. McGibbon, Roland Memisevic, Bart van Merriënboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Pal, Razvan Pascanu, Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schlüter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, and Ying Zhang. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[110] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[111] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. https://openreview.net/pdf?id=BJJsrmfCZ.

[112] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo Moitinho de Almeida, Brian McFee, Hendrik Weideman, Gábor Takács, Peter de Rivaz, Jon Crall, Gregory Sanders, Kashif Rasul, Cong Liu, Geoffrey French, and Jonas Degrave. Lasagne: First release., August 2015.

[113] François Chollet et al. Keras: The python deep learning library, 2015.

[114] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

[115] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

[116] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page I–387–I–395. JMLR.org, 2014.

[117] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17(1):1334–1373, January 2016.

[118] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[119] M. P. Deisenroth, G. Neumann, and J. Peters. *A Survey on Policy Search for Robotics*. 2013.

[120] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[121] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[122] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[123] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[124] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[125] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[126] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[127] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[128] N. Ball and R. Brunner. Data mining and machine learning in astronomy. *International Journal of Modern Physics D*, 19(07):1049–1106, 2010.

[129] Michael J. Way, Jeffrey D. Scargle, Kamal M. Ali, and Ashok N. Srivastava. *Advances in Machine Learning and Data Mining for Astronomy*. Chapman Hall/CRC, 1st edition, 2012.

[130] Rahul Biswas, Lindy Blackburn, Junwei Cao, Reed Essick, Kari Alison Hodge, Erotokritos Katsavounidis, Kyungmin Kim, Young-Min Kim, Eric-Olivier Le Bigot, Chang-Hwan Lee, John J. Oh, Sang Hoon Oh, Edwin J. Son, Ye Tao, Ruslan Vaulin, and Xiaoge Wang. Application of machine learning algorithms to the study of noise artifacts in gravitational-wave data. *Phys. Rev. D*, 88:062003, Sep 2013.

[131] Philip Graff, Farhan Feroz, Michael P. Hobson, and Anthony Lasenby. SkyNet: an efficient and robust neural network training tool for machine learning in astronomy. *Monthly Notices of the Royal Astronomical Society*, 441(2):1741–1759, 05 2014.

[132] Alexander Radovic, Mike Williams, David Rousseau, Michael Kagan, Daniele Bonacorsi, Alexander Himmel, Adam Aurisano, Kazuhiro Terao, and Taritree Wongjirad. Machine

learning at the energy and intensity frontiers of particle physics. *Nature*, 560(7716):41–48, 2018.

[133] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5(1):4308, 2014.

[134] Pushpalatha C. Bhat. Multivariate analysis methods in particle physics. *Annual Review of Nuclear and Particle Science*, 61(1):281–309, 2011.

[135] Shimon Whiteson and Daniel Whiteson. Machine learning for event selection in high energy physics. *Engineering Applications of Artificial Intelligence*, 22(8):1203 – 1217, 2009.

[136] O. Meneghini, C. J. Luna, S. P. Smith, and L. L. Lao. Modeling of transport phenomena in tokamak plasmas with neural networks. *Physics of Plasmas*, 21(6):060702, 2014.

[137] A. Murari, D. Mazon, N. Martin, G. Vagliasindi, and M. Gelfusa. Exploratory data analysis techniques to determine the dimensionality of complex nonlinear phenomena: The l-to-h transition at jet as a case study. *IEEE Transactions on Plasma Science*, 40(5):1386–1394, 2012.

[138] A. Murari, G. Mazzitelli, A. Buscarino, L. Fortuna, M. Frasca, and M. Iachello. Identifying jet instabilities with neural networks. In *2012 16th IEEE Mediterranean Electrotechnical Conference*, pages 932–935, 2012.

[139] A. Murari, P. Arena, A. Buscarino, L. Fortuna, and M. Iachello. On the identification of instabilities with neural networks on JET. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 720:2 – 6, 2013.

[140] S. Dormido-Canto, J. Vega, J.M. Ramírez, A. Murari, R. Moreno, J.M. López, and A. Pereira and. Development of an efficient real-time disruption predictor from scratch on JET and implications for ITER. *Nuclear Fusion*, 53(11):113001, sep 2013.

[141] Jesús Vega, Sebastián Dormido-Canto, Juan M. López, Andrea Murari, Jesús M. Ramírez, Raúl Moreno, Mariano Ruiz, Diogo Alves, and Robert Felton. Results of the JET real-time disruption predictor in the ITER-like wall campaigns. *Fusion Engineering and Design*, 88(6):1228 – 1231, 2013. Proceedings of the 27th Symposium On Fusion Technology (SOFT-27); Liège, Belgium, September 24-28, 2012.

[142] G. Vagliasindi, A. Murari, P. Arena, L. Fortuna, G. Arnoux, E. Gauthier, and JET EFDA Contributors. First application of cellular nonlinear network methods to the real-time identification of hot spots in jet. *IEEE Transactions on Plasma Science*, 37(1):146–152, 2009.

[143] P. Arena, A. Basile, L. Fortuna, G. Mazzitelli, A. Rizzo, and M. Zammataro. Cnn-based real-time video detection of plasma instability in nuclear fusion applications. In *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, volume 3, pages III–77, 2004.

[144] K.J. Hunt, D. Sbarbaro, R. Żbikowski, and P.J. Gawthrop. Neural networks for control systems—a survey. *Automatica*, 28(6):1083 – 1112, 1992.

[145] Mohamed Azlan Hussain. Review of the applications of neural networks in chemical process control — simulation and online implementation. *Artificial Intelligence in Engineering*, 13(1):55 – 68, 1999.

[146] J.M. Zamarreño and P. Vega. Neural predictive control. application to a highly non-linear system. *Engineering Applications of Artificial Intelligence*, 12(2):149 – 158, 1999.

[147] Chi-Huang Lu and Ching-Chih Tsai. Generalized predictive control using recurrent fuzzy neural networks for industrial processes. *Journal of Process Control*, 17(1):83 – 92, 2007.

[148] Hao Huang, Lei Chen, Morteza Mohammadzaheri, and Eric Hu. A new zone temperature predictive modeling for energy saving in buildings. *Procedia Engineering*, 49:142 – 151, 2012. International Energy Congress 2012.

[149] Le Chen, Baoming Ge, and Aníbal T. de Almeida. Self-tuning pid temperature controller based on flexible neural network. In Derong Liu, Shumin Fei, Zeng-Guang Hou, Huaguang Zhang, and Changyin Sun, editors, *Advances in Neural Networks –ISNN 2007*, pages 138–147, 2007.

[150] Rodrigo Hernández-Alvarado, Luis Govinda García-Valdovinos, Tomás Salgado-Jiménez, Alfonso Gómez-Espinosa, and Fernando Fonseca-Navarro. Neural network-based self-tuning pid control for underwater vehicles. *Sensors (Basel, Switzerland)*, 16(9):1429, 09 2016.

[151] Maciej Lawryczuk. *Computationally Efficient Model Predictive Control Algorithms: A Neural Network Approach*. Springer Publishing Company, Incorporated, 1st edition, 2016.

[152] R.K. Al Seyab and Y. Cao. Nonlinear system identification for predictive control using continuous time recurrent neural networks and automatic differentiation. *Journal of Process Control*, 18(6):568 – 581, 2008.

[153] Stephen Piché, Jim Keeler, Greg Martin, Gene Boe, Doug Johnson, and Mark Gerules. Neural network based model predictive control. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 1029–1035, Cambridge, MA, USA, 1999. MIT Press.

[154] A. Draeger, S. Engell, and H. Ranke. Model predictive control using neural networks. *IEEE Control Systems Magazine*, 15(5):61–66, 1995.

[155] Bernt M. Åkesson, Hannu T. Toivonen, Jonas B. Waller, and Rasmus H. Nyström. Neural network approximation of a nonlinear model predictive controller applied to a ph neutralization process. *Computers Chemical Engineering*, 29(2):323 – 335, 2005.

[156] Maciej Ławryńczuk. A family of model predictive control algorithms with artificial neural networks. 2007.

[157] Maciej Lawryczuk. Suboptimal nonlinear predictive control based on multivariable neural hammerstein models. *Applied Intelligence*, 32(2):173–192, April 2010.

[158] Y. Ding, C. Behrens, P. Emma, J. Frisch, Z. Huang, H. Loos, P. Krejcik, and M-H. Wang. Femtosecond x-ray pulse temporal characterization in free-electron lasers using a transverse deflector. *Phys. Rev. ST Accel. Beams*, 14:120701, Dec 2011.

[159] J.-L. Vay. Noninvariance of space- and time-scale ranges under a lorentz transformation and the implications for the study of relativistic interactions. *Phys. Rev. Lett.*, 98:130405, Mar 2007.

[160] Jean-Luc Vay, Irving Haber, and Brendan B. Godfrey. A domain decomposition method for pseudo-spectral electromagnetic simulations of plasmas. *Journal of Computational Physics*, 243:260 – 268, 2013.

[161] Andreas Adelmann, Christian Baumgarten, Matthias Frey, Achim Gsell, Valeria Rizzoglio, Jochem Snuverink (PSI), Christof Metzger-Kraus, Yves Ineichen, Steve Russell (LANL), Chuan Wang (CIAE), Suzanne Sheehy, Chris Rogers (RAL), and Daniel Winklehner (MIT). The OPAL (Object Oriented Parallel Accelerator Library) Framework. Technical Report PSI-PR-08-02, Paul Scherrer Institut, 2008-2019.

[162] M. Borland. Elegant: A flexible sdds-compliant code for accelerator simulation. Technical Report LS-287, Argonne National Laboratory, 2000-2018.

[163] S. A. Baily, X. Pang, and L. Rybarcyk. High-performance beam simulator for the lansce linac. In *Proc. of IPAC*, page C1205201, 2012.

[164] T. Higo, Hamid Shoaee, and J. Spencer. Some Applications of AI to the Problems of Accelerator Physics. *Conf. Proc. C*, 870316:701, 1987.

[165] D. Weygand. Artificial intelligence and accelerator control. 01 1987.

[166] D Schirmer and P Hartmann. Electron transport line optimization using neural networks and genetic algorithms. pages 1948–1950, 01 2006.

[167] D. Nguyen, M. Lee, R. Sass, and H. Shoaee. Accelerator and feedback control simulation using neural networks. In *Conference Record of the 1991 IEEE Particle Accelerator Conference*, pages 1437–1439 vol.3, 1991.

[168] Eva Bozoki and Aharon Friedman. Neural network technique for orbit correction in accelerators/storage rings. In *American Institute of Physics Conference Series*, volume 315 of *American Institute of Physics Conference Series*, pages 103–110, August 1994.

[169] Y Kijima, M Mizota, K Yoshida, and K Suzuki. A Beam Diagnostic System for Accelerator using Neural Networks. 1992.

[170] Sungil Kwon, J. Davis, M. Lynch, M. Prokop, S. Ruggles, and P. Torrez. Gain scheduled neural network tuned pi feedback control system for the lansce accelerator. In *2007 IEEE Particle Accelerator Conference (PAC)*, pages 2379–2381, 2007.

[171] W.C. Mead, P.S. Bowling, S.K. Brown, R.D. Jones, C.W. Barnes, H.E. Gibson, J.R. Goulding, and Y.C. Lee. Optimization and control of a small-angle negative ion source using an on-line adaptive controller based on the connectionist normalized local spline neural network. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 72(2):271 – 289, 1992.

[172] W.C. Mead, S.K. Brown, R.D. Jones, P.S. Bowling, and C.W. Barnes. Adaptive optimization and control using neural networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 352(1):309 – 315, 1994.

[173] J.A. Howell, C.W. Barnes, S.K. Brown, G.W. Flake, R.D. Jones, Y.C. Lee, S. Qian, and R.M. Wright. Control of a negative-ion accelerator source using neural networks. *Nuclear Instru-*

ments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 293(1):517 – 522, 1990.

[174] E. Meier, S.G. Biedron, G. LeBlanc, M.J. Morgan, and J. Wu. Electron beam energy and bunch length feed forward control studies using an artificial neural network at the linac coherent light source. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 610(3):629 – 635, 2009.

[175] E. Meier, S.G. Biedron, G. LeBlanc, and M.J. Morgan. Development of a novel optimization tool for electron linacs inspired by artificial intelligence techniques in video games. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 632(1):1 – 6, 2011.

[176] A. Edelen et al. Opportunities in Machine Learning for Particle Accelerators. *White paper from the 1st ICFA Machine Learning Workshop for Particle Accelerators.*, 2018. Preprint at https://arxiv.org/abs/1811.03172.

[177] Sander Dieleman, Kyle W. Willett, and Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(2):1441–1459, apr 2015.

[178] A. Aurisano, A. Radovic, D. Rocco, A. Himmel, M.D. Messier, E. Niner, G. Pawloski, F. Psihas, A. Sousa, and P. Vahle. A convolutional neural network neutrino event classifier. *Journal of Instrumentation*, 11(09):P09001–P09001, sep 2016.

[179] D. Edstrom Jr. et al. 50-MeV Run of the IOTA/FAST Electron Accelerator. In *2nd North American Particle Accelerator Conference*, page TUPOA19, 2017.

[180] Elvin Harms, Jerry Leibfritz, Sergei Nagaitsev, Philippe Piot, Jinhao Ruan, Vladimir Shiltsev, Giulio Stancari, and Alexander Valishev. The Advanced Superconducting Test Accelerator at Fermilab. *ICFA Beam Dyn. Newslett.*, 64:133–156, 2014.

[181] S. Antipov, D. Broemmelsiek, D. Bruhwiler, D. Edstrom, E. Harms, V. Lebedev, J. Leibfritz, S. Nagaitsev, C.S. Park, H. Piekarz, P. Piot, E. Prebys, A. Romanov, J. Ruan, T. Sen, G. Stancari, C. Thangaraj, R. Thurman-Keup, A. Valishev, and V. Shiltsev. IOTA (Integrable Optics Test Accelerator): facility and experimental beam physics program. *Journal of Instrumentation*, 12(03):T03002–T03002, 2017.

[182] L. M. Young and J. Billen. The Particle Tracking Code PARMELA.

[183] K. Halbach and R. F. Holsinger. Superfish-a computer program for evaluation of rf cavities with cylindrical symmetry. *Part. Accel.*, 7:213–222, 1976.

[184] H. Takeda and J. E. Stovall. Modified parmila code for new accelerating structures. In *Proc. of PAC*, volume 4, pages 2364–2366, May 1995.

[185] Alexander Scheinker and Spencer Gessner. Adaptive method for electron bunch profile prediction. *Phys. Rev. ST Accel. Beams*, 18:102801, Oct 2015.

[186] A. Sanchez-Gonzalez, P. Micaelli, C. Olivier, T. R. Barillot, M. Ilchen, A. A. Lutman, A. Marinelli, T. Maxwell, A. Achner, M. Agåker, N. Berrah, C. Bostedt, J. D. Bozek, J. Buck, P. H. Bucksbaum, S. Carron Montero, B. Cooper, J. P. Cryan, M. Dong, R. Feifel, L. J. Frasinski, H. Fukuzawa, A. Galler, G. Hartmann, N. Hartmann, W. Helml, A. S. Johnson, A. Knie, A. O. Lindahl, J. Liu, K. Motomura, M. Mucke, C. O'Grady, J-E Rubensson, E. R. Simpson, R. J. Squibb, C. Såthe, K. Ueda, M. Vacher, D. J. Walke, V. Zhaunerchyk, R. N. Coffee, and J. P. Marangos. Accurate prediction of x-ray pulse properties from a free-electron laser using machine learning. *Nature Communications*, 8(1):15461, 2017.

[187] C. Emma, A. Edelen, M. J. Hogan, B. O'Shea, G. White, and V. Yakimenko. Machine learning-based longitudinal phase space prediction of particle accelerators. *Phys. Rev. Accel. Beams*, 21:112802, Nov 2018.

[188] Jinhao Ruan, Daniel Broemmelsiek, Darren Crawford, Auralee Edelen, Jonathan Edelen, Dean Edstrom, Alex Lumpkin, Philippe Piot, Alexander Romanov, and Randy Thurman-

Keup. Emittance Measurements at FAST Facility. In *9th International Particle Accelerator Conference*, page THPMF025, 2018.

[189] S. G. Anderson, J. B. Rosenzweig, G. P. LeSage, and J. K. Crane. Space-charge effects in high brightness electron beam emittance measurements. *Phys. Rev. ST Accel. Beams*, 5:014201, Jan 2002.

[190] A. Halavanau, P. Piot, D. Edstrom, A. Romanov, D. Crawford, J. Ruan, D. Mithalcea, V. Shiltsev, and S. Nagaitsev. Magnetized and Flat Beam Generation at the Fermilab's FAST Facility. In *9th International Particle Accelerator Conference*, page THPAK061, 2018.

[191] R. M. Neal. Bayesian learning for neural networks. *Springer Science & Business Media*, 2012.

[192] Jacques-Philippe Colletier, Michael R. Sawaya, Mari Gingery, Jose A. Rodriguez, Duilio Cascio, Aaron S. Brewster, Tara Michels-Clark, Robert H. Hice, Nicolas Coquelle, Sébastien Boutet, Garth J. Williams, Marc Messerschmidt, Daniel P. DePonte, Raymond G. Sierra, Hartawan Laksmono, Jason E. Koglin, Mark S. Hunter, Hyun-Woo Park, Monarin Uervirojnangkoorn, Dennis K. Bideshi, Axel T. Brunger, Brian A. Federici, Nicholas K. Sauter, and David S. Eisenberg. De novo phasing with x-ray laser reveals mosquito larvicide binab structure. *Nature*, 539(7627):43–47, 2016.

[193] Iris D. Young, Mohamed Ibrahim, Ruchira Chatterjee, Sheraz Gul, Franklin D. Fuller, Sergey Koroidov, Aaron S. Brewster, Rosalie Tran, Roberto Alonso-Mori, Thomas Kroll, Tara Michels-Clark, Hartawan Laksmono, Raymond G. Sierra, Claudiu A. Stan, Rana Hussein, Miao Zhang, Lacey Douthit, Markus Kubin, Casper de Lichtenberg, Long Vo Pham, Håkan Nilsson, Mun Hon Cheah, Dmitriy Shevela, Claudio Saracini, Mackenzie A. Bean, Ina Seuffert, Dimosthenis Sokaras, Tsu-Chien Weng, Ernest Pastor, Clemens Weninger, Thomas Fransson, Louise Lassalle, Philipp Bräuer, Pierre Aller, Peter T. Docker, Babak Andi, Allen M. Orville, James M. Glownia, Silke Nelson, Marcin Sikorski, Diling Zhu,

Mark S. Hunter, Thomas J. Lane, Andy Aquila, Jason E. Koglin, Joseph Robinson, Mengning Liang, Sébastien Boutet, Artem Y. Lyubimov, Monarin Uervirojnangkoorn, Nigel W. Moriarty, Dorothee Liebschner, Pavel V. Afonine, David G. Waterman, Gwyndaf Evans, Philippe Wernet, Holger Dobbek, William I. Weis, Axel T. Brunger, Petrus H. Zwart, Paul D. Adams, Athina Zouni, Johannes Messinger, Uwe Bergmann, Nicholas K. Sauter, Jan Kern, Vittal K. Yachandra, and Junko Yano. Structure of photosystem ii and substrate binding at room temperature. *Nature*, 540(7633):453–457, 2016.

[194] M. P. Jiang, M. Trigo, I. Savić, S. Fahy, É. D. Murray, C. Bray, J. Clark, T. Henighan, M. Kozina, M. Chollet, J. M. Glownia, M. C. Hoffmann, D. Zhu, O. Delaire, A. F. May, B. C. Sales, A. M. Lindenberg, P. Zalden, T. Sato, R. Merlin, and D. A. Reis. The origin of incipient ferroelectricity in lead telluride. *Nature Communications*, 7(1):12291, 2016.

[195] G. J. Ernst, W. J. Witteman, E. H. Haselhoff, J. I. M. Botman, J. L. Delhez, and H. L. Hagedoorn. Status of the dutch "teu-fel" project. In *Proceedings of the Eleventh International Free Electron Laser Conference*, pages 304–307, 1989.

[196] J.W.J. Verschuur, G.J. Ernst, and W.J. Witteman. The “teufel” undulator. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 318(1):847 – 852, 1992.

[197] Jonathan Paul Edelen. *Design and simulation of the Colorado State University linear accelerator system*. PhD thesis, Colorado State University, January 2014.

[198] Sandra Biedron et al. Colorado State University (CSU) Accelerator and FEL Facility. In *5th International Particle Accelerator Conference*, page THPRI074, 7 2014.

[199] Luca Serafini and James B. Rosenzweig. Envelope analysis of intense relativistic quasilaminar beams in rf photoinjectors:ma theory of emittance compensation. *Phys. Rev. E*, 55:7565–7590, Jun 1997.

[200] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525 – 533, 1993.

[201] A. Edelen, J.P. Edelen, S.G. Biedron, S.V. Milton, and P.J.M van der Slot. Using a neural network control policies for rapid switching between beam parameters in a FEL. *NeurIPS*, 2017.

[202] A. Edelen, J.P. Edelen, S.G. Biedron, S.V. Milton, and P.J.M van der Slot. Using a neural network control policy for rapid switching between beam parameters in a FEL. *International FEL Conference '17*, WEP031, 2017.

[203] Lars Grune and Jrgen Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer Publishing Company, Incorporated, 2013.

[204] J.B. Rawlings, E.S. Meadows, and K.R. Muske. Nonlinear model predictive control: A tutorial and survey. *IFAC Proceedings Volumes*, 27(2):185 – 197, 1994. IFAC Symposium on Advanced Control of Chemical Processes, Kyoto, Japan, 25-27 May 1994.

[205] David Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967 – 2986, 2014.

[206] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789 – 814, 2000.

[207] Philippe Piot, Elvin Harms, Stuart Henderson, Jerry Leibfritz, Sergei Nagaitsev, Vladimir Shiltsev, and Alexander Valishev. The Advanced Superconducting Test Accelerator at Fermilab: Science Program. *IPAC 2014*.

[208] P. Stabile, M. Ball, J. Czajkowski, J. Firebaugh, P. Kalsey, P.S. Prieto, and T. Zuchnik. RF Gun Water Temperature Control System at ASTA. In *1st North American Particle Accelerator Conference*, 9 2013.

[209] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, jul 1944.

[210] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[211] Frank J. Nagy. The fermilab accelerator controls networking system. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 247(1):208 – 214, 1986.

[212] R. Acciarri et al. Long-Baseline Neutrino Facility (LBNF) and Deep Underground Neutrino Experiment (DUNE) Conceptual Design Report Volume 1: The LBNF and DUNE Projects, 2016.

[213] The Mu2e Project, Collaboration, et al. Mu2e Conceptual Design Report, 2012.

[214] Andrew Lambert et al. High-Intensity Proton RFQ Accelerator Fabrication Status for PXIE. In *6th International Particle Accelerator Conference*, page WEPTY045, 2015.

[215] A. Lambert et al. RF, Thermal, and Structural Finite Element Analysis of the Project X (PXIE) CW Radio Frequency Quadrupole (RFQ. In *Proc. of PAC 2013*, page WEPMA20, 2013.

[216] P. N. Ostroumov, B. Mustapha, A. Barcikowski, C. Dickerson, A. A. Kolomiets, S. A. Kondrashev, Y. Luo, D. Paskvan, A. Perry, D. Schrage, S. I. Sharamentov, R. Sommer, W. Toter, and G. Zinkann. Development and beam test of a continuous wave radio frequency quadrupole accelerator. *Phys. Rev. ST Accel. Beams*, 15:110101, Nov 2012.

[217] Jing Wang, Jian-Long Huang, Yuan He, Xiao-Qi Zhang, Zhou-Li Zhang, and Ai-Min Shi. Multi-physics analysis of the RFQ for Injector Scheme II of C-ADS driver linac. *Chinese Physics C*, 38(10):107005, October 2014.

[218] S. Virostek and J. Staples. Analysis of thermally induced frequency shift for the spallation neutron source RFQ. *LINAC'00, THD04*, 2000.

[219] S. P. Virostek, M. D. Hoff, A. Lambert, D. Li, J. W. Staples, G. V. Romanov, and C. Zhang. Design and Analysis of the PXIE CW Radio-frequency Quadrupole (RFQ). *IPAC'12*, HPPC034, 2012.

[220] Yasuhiro Kondo, Kazuo Hasegawa, Takatoshi Morishita, Hiroshi Matsumoto, and Fujio Naito. Thermal Characteristics of a New RFQ for J-PARC. *IPAC 2010, MOPD043*, 2010.

[221] Yasuhiro Kondo et al. High-power test and thermal characteristics of a new radio-frequency quadrupole cavity for the Japan Proton Accelerator Research Complex linac. *Phys. Rev. ST Accel. Beams*, 16(4):040102, 2013.

[222] J. Crisp and J. Satti. Fermilab linac upgrade side coupled cavity temperature control system. In *Conference Record of the 1991 IEEE Particle Accelerator Conference*, pages 3189–3191 vol.5, 1991.

[223] J. Crisp and J. Satti. Temperature (or frequency) control system. Technical report, Fermilab, Batavia, IL, USA.

[224] J. Crisp. Linac temperature loops. Technical report, Fermilab, Batavia, IL, USA.

[225] Inc. The MathWorks. *Simulink*. Natick, Massachusetts, United State, 2015.

[226] M.J. Moran. *Introduction to thermal systems engineering: thermodynamics, fluid mechanics, and heat transfer*. Number v. 1 in Introduction to Thermal Systems Engineering: Thermodynamics, Fluid Mechanics, and Heat Transfer. Wiley, 2003.

[227] Jonathan Edelen, Brian Chase, Ed Cullerton, Joshua Einstein, and Philip Varghese. Low Level RF Control for the PIP-II Injector Test RFQ. In *2nd North American Particle Accelerator Conference*, page TUPOA18, 2017.

[228] R. Neswold and C. King. Generation of simple, type-safe messages for inter-task communications. In *12th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2009)*, 2009.

[229] R. Neswold and C. King. Further developments in generating type-safe messaging, 2012.

[230] Daniel Bowring, Sandra Biedron, Brian Chase, Jerzy Czajkowski, Auralee Edelen, Jonathan Edelen, Stephen Milton, Dennis Nicklaus, Jim Steimel, and Thomas Zuchnik. Resonant Control for Fermilab's PXIE RFQ. In *7th International Particle Accelerator Conference*, page MOPMW026, 2016.

[231] Yuriy Pischalnikov and Warren Schappert. Adaptive compensation of Lorentz force detuning in superconducting RF cavities, FERMILAB-TM-2476-TD. Technical report, 2011.

[232] Alexander Scheinker, Auralee Edelen, Dorian Bohler, Claudio Emma, and Alberto Lutman. Demonstration of model-independent control of the longitudinal phase space of electron beams in the linac-coherent light source with femtosecond resolution. *Phys. Rev. Lett.*, 121:044801, Jul 2018.

[233] Auralee Edelen, Sandra Biedron, Jonathan Edelen, and Stephen Milton. First steps toward incorporating image based diagnostics into particle accelerator control systems using convolutional neural networks. *NAPAC'16*, TUPOA51, 2016.

[234] A. Edelen, S.G. Biedron, D. Bowring, B.E. Chase, D.E. Edstrom, J. Steimel, J.P. Edelen, and P.J.M. van der Slot. Neural Network Based Approaches to the Modeling and Control of Particle Accelerators. *IPAC'18*, THYGBE2, 2018. https://accelconf.web.cern.ch/ipac2018/talks/thygbe2$_talk.pdf$.

[235] A. Edelen, J.P. Edelen, D. Edstrom, J. Ruan, S.G. Biedron, P. Piot, and A. Halavanau. Neural Network Virtual Diagnostic for the FAST Low Energy Beamline. *IPAC'18*, WEPAF040, 2018.

[236] A. L. Edelen, S. G. Biedron, S. V. Milton, D. Bowring, B. E. Chase, J. P. Edelen, and J. Steimel. Neural Network Modeling of the PXIE RFQ Cooling System and Resonant Frequency Response. *IPAC'16*, THPOY020, 2016.

[237] A. Edelen et al. Neural Networks for Modeling and Control of Particle Accelerators. *Transactions on Nuclear Science.*, 63, 2016.

[238] A.L. Edelen et al. Initial experimental results of a machine learning-based temperature control system for an RF gun. In *6th International Particle Accelerator Conference*, page MOPWI028, 2015.

[239] A.L. Edelen, S.G. Biedron, S.V. Milton, D. Bowring, B.E. Chase, J.P. Edelen, D. Nicklaus, and J. Steimel. Resonant Frequency Control For the PIP-II Injector Test RFQ: Control Framework and Initial Results. In *2nd North American Particle Accelerator Conference*, page MOPOB17, 2017.

[240] J. P. Edelen, A. L. Edelen, D. Bowring, B. E. Chase, J. Steimel, S. G. Biedron, and S. V. Milton. First Principles Modeling of RFQ Cooling System and Resonant Frequency Responses for Fermilab's PIP-II Injector Test. *IEEE Transactions on Nuclear Science*, 64(2):800–808, 2017.