# Optimisation of linear accelerator performance for single-pass free-electron laser operation

---

Evelyne Meier
BSc, MSc


School of Physics, Monash University

March 30, 2011

# Contents

# List of Figures

# Summary

This project is part of a collaboration between Monash University, the Australian Synchrotron (AS), the new FERMI@Elettra project, and the Linac Coherent Light Source (LCLS) at the SLAC National Accelerator Laboratory. The thesis investigates the use of Artificial Intelligence systems and their applicability to machine optimisation and control for linear accelerators, and in particular Free Electron Lasers (FEL). This research is motivated by the need to develop adaptive systems for beam tuning and stabilisation, in order to meet the increasingly stringent requirements of new generation light sources.

The thesis begins with the simulation of a feedback system for the FERMI@Elettra Linac, based on the Proportional - Integral - Differential[1] (PID) scheme developed for the LCLS. To facilitate these simulations, a Matlab Graphical User Interface was built in order to incorporate various control parameters, including possible actuators, observable variables and perturbations. These simulations highlight the difficulties encountered with PID control, which necessitates a more sophisticated approach to the control of the linear accelerator.

To address the intrinsic limitations of conventional PID control, a combination of a feedforward - feedback system was investigated. The feedforward component uses a neural network (NNet), which provides a prediction of the perturbation in the electron beam parameters based on fluctuations in the voltage and phase of the klystrons[2]. The feedback component consists of a simple PID algorithm, used to compensate for potential inaccuracies of the feedforward correction. Experimental results performed at the

[1] The PID controller involves three corrective terms: the proportional, the integral and derivative terms, denoted by P, I, and D, respectively. These terms can be interpreted as follows: P depends on the present error, I on the accumulation of past errors, and D on a prediction of future errors. See Secs 2.7 and 5.2.5 for further detail.

[2] A klystron is an electron tube used to generate or amplify electromagnetic radiation in the microwave regime by velocity modulation. In a klystron, electrons are accelerated by the application of a voltage. As the electrons leave the heated cathode of the tube, they are directed into resonating chambers tuned at or near the tube's operating frequency. Strong RF fields are induced in the chambers as the electron bunches give up energy, which are ultimately collected at the output resonating chamber to feed the cavities of the Linac.

AS show the viability of the system, by demonstrating the successful control of the energy at the end of the Linac. The experiments carried out at the LCLS show the applicability of the system to a multi-variable system with the simultaneous control of the energy and bunch length. Although these results demonstrate the ability of the NNet predictor to compensate for the deficiencies of the PID algorithm, further refinements of the technique are required to produce a system that can adapt to changes in machine parameters and jitter[3] conditions.

To correct the remaining deficiencies of the combined feedforward - feedback control system, we consider an intelligent system capable of self-learning. In this scenario the control system is treated from the perspective of an optimisation problem and a novel optimisation tool is designed, using state of the art developments in video games. The key principle is to exploit similarities between the navigation of a game agent in a battlefield and the navigation of an "optimisation agent" in a search space. This novel approach was tested using simulations and experiments conducted at the AS and on the FERMI@Elettra Linac. The experiments conducted at the AS showed the system's ability to simultaneously optimise the beam energy spread and transmission (i.e. the percentage of particles transmitted from the start to the end of the accelerator). We have demonstrated an increase in the transmission from 90% to 97% and a decrease in the energy spread of the beam from 1.04% to 0.91%. Control experiments performed at the new FERMI@Elettra FEL are also reported, which highlight the adaptability of the system for beam-based control, in the case where a static perturbation is applied to the klystron phase. These results show that NNets can be successfully exploited to build an optimisation tool that can self-learn from its interaction with the machine and operate a simple control task. Our results indicate that this optimisation tool can be used for the stabilisation of the electron beam parameters when it is subject to time dependent perturbations.

The thesis concludes with suggestions for future work. This includes the adaptation of the optimisation tool to N-dimensional search spaces, and the development of a novel control system which merges the NNet predictor with the structure of the NNet used for optimisation. By combining these two structures, it is anticipated that the resulting NNet will have the ability to correct time dependent perturbations, while self-adapting its response when machine parameters and jitter conditions change.

[3]Jitter is formally expressed as the movement of a signal edge from its ideal position in time and can be deterministic or random. Throughout this thesis the term jitter will be used to express a deviation from a desired setting, without specifically referring to time.

# Disclosure

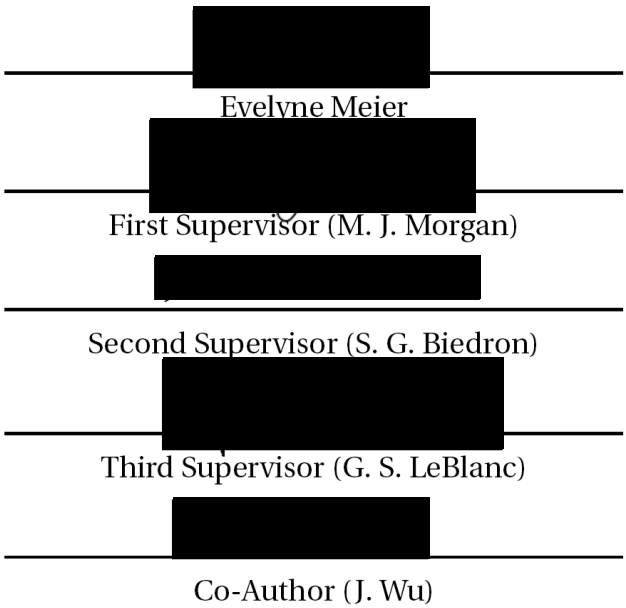
This thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other institution. To the best of my knowledge the thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis. Chapters 7 and 8 in this thesis are based on joint research and publications, the relative contributions to these chapters of the respective co-authors being as follows:

Chapters 7 and 8:

- **S.G. Biedron** (Argonne National Laboratory- US and Sincrotrone Trieste- Italy) : 5%
- **G.S. LeBlanc** (Australian Synchrotron) : 5%
- **M.J. Morgan** (Monash University-Australia) : 10%

Chapter 7 only:

- **J. Wu** (SLAC National Accelerator Laboratory - US) : 10%



# Acknowledgements

I consider myself particularly lucky to have had the chance to meet and work with a number of great people during my candidacy. I would like on this occasion, show my gratitude to those who have been important to the successful realisation of this work.

I owe my deepest gratitude to Sandra Biedron and Stephen Milton for having initiated a collaboration between FERMI@Elettra, Monash University, and the Australian Synchrotron, of which this thesis is part. It is also a pleasure to express my gratitude wholeheartedly to Sandra and Stephen for their kind hospitality during my stay in Trieste.

I wish to thank Monash University for the grant of scholarships that supported my work. I also want to thank Sincrotrone Trieste for equipment loan and for supporting the collaboration with the Australian Synchrotron and Monash University.

I gratefully acknowledge Michael Morgan for his advice, supervision, and crucial contribution in correcting my "poor" English.

I thank my supervisor Greg LeBlanc for providing the beam time on the Australian Synchrotron Linac and my officemate Elsa Van Garderen to keep me happy in the office.

I was extraordinarily fortunate in having Juhao Wu as my supervisor at the Stanford National Laboratory. I am thankful for all the enriching discussions and his help with planning experiments at the LCLS. I could never have done all of this without his teachings in accelerator physics and precious advice. Thank you.

My special thanks go to Paul Emma for his support and the allocation of beam time on the LCLS machine. Furthermore, I would like to thank the LCLS commissioning team members for their advice and constructive comments and ideas during my stay at SLAC.

I have also benefited by advice and guidance from Simone Di Mitri, Paolo Craevich and Giuseppe Penco who kindly granted me their time for answering my questions during my visits to Sincrotrone Trieste.

Also a special thank you to my aunt Martha Meier and my parents Paul and Andrée Meier for their support. To them I dedicate this thesis.

Finally, I wish to thank all my friends for being so supportive to me during my research, with a special thought for Agnes and Adam Michalczyk, as well as Rachel Mayer. I apologise for not mentioning them all personally one by one.

# 1 Introduction

## 1.1 Project background and motivation

FERMI@Elettra is a new 4th-generation light source located next to the third-generation synchrotron radiation facility ELETTRA in Trieste, Italy. It is a single pass Free Electron Laser (FEL) providing high peak-power photon pulses over the wavelength range from 100 nm (12 eV) to 10 nm (124 eV). The single-bunch structure electron beam is produced by a high-brightness RF photocathode with a 50 Hz repetition rate and accelerated to 1.5 GeV. Two bunch compressors provide the necessary beam brightness[1]. At the end of the accelerator, chains of undulators produce the photon beam with a seeded laser multistage mechanism. A total of 22 undulators provide the beamlines with tunable output over a range from ~100 nm to ~4 nm [2].

In order to ensure that free electron lasing can take place and that the quality requirements on the produced photon beams are met, a high degree of stability is required for the longitudinal parameters of the electron beam. For FERMI in particular, a maximum rms energy and peak current[2] variation of 0.1% and 10%, respectively has been specified for the electron bunches in order to guarantee FEL performance [2]. These parameters can be measured at three locations in the accelerator, i.e. at the two bunch compressors and the spreader (just before the entrance to the undulator chamber). They can be corrected using the voltages and phases of the Linac klystrons. The types of perturbations we expect to encounter include slow drifts due to temperature fluctuations and changes in the properties of the

[1] An important parameter describing an electron beam is the brightness, defined as $B = 2I_0/(\pi^2\epsilon^2)$, where $I_0$ is the beam current and $\epsilon$ is the emittance. To compare beams of different energies, it is convenient to use the normalised brightness, defined as $B_n = 2I_0/(\pi^2\epsilon_n^2)$, where $\epsilon_n = \beta\gamma\epsilon$ is the normalised emittance; $\beta$ is the ratio of the beam velocity to the velocity of light and $\gamma$ is the relativistic factor $1/\sqrt{1-\beta^2}$ [1].

[2] The peak current is defined by $I = Q/\tau$, where $Q$ is the total charge in the electron bunch and $\tau$ is the bunch duration. The peak current is used as a measure of the bunch length, which is the longitudinal parameter we are interested in stabilising (see Secs. 4.3.3 and 4.3.4).

accelerator components. Mechanical vibration of the quadrupole magnets and jitter of the RF accelerating fields are also expected.

A number of control loops are necessary to stabilise the parameters of the beam (i.e. energy and bunch length for longitudinal control). Although it is possible to run one local feedback loop for each of the parameters, the corrections performed by one loop will affect the corrections of the loops downstream. To ameliorate this problem, it is necessary to implement cascaded feedbacks, where the upstream loops communicate the correction downstream. The basic control algorithm currently used in facilities such as the Linac Coherent Light Source is a Proportional-Integral-Differential (PID) controller preceded by a low pass filter. For shot by shot control (i.e. for a sampling rate equal to the bunch repetition frequency), a maximum attenuation bandwidth of a few Hz is achievable. Consequently, only low frequency perturbations and slow drifts can be accommodated [2]. More sophisticated cascaded local control algorithms can be implemented to ensure better control. However, there is a real need to develop adaptive control systems as requirements on beam quality become increasingly stringent. The beam-based control studies of longitudinal parameters for the FERMI@Elettra FEL have motivated the research presented in this thesis; in the next section we discuss the proposed objective of this project.

## 1.2 Objective

The present thesis investigates the use of Artificial Intelligence (AI) Systems for optimisation of linear accelerator performance, with a particular focus on the FERMI Free Electron Laser. Here, the term performance includes two main aspects of accelerator operation. The first aspect relates to the necessary stabilisation of the electron beam to ensure that the FEL lasing process can take place. The second aspect considers the performance of optimisation procedures used to adjust beam parameters to meet users' requirements regarding the light produced.

To ensure the performance of the lasing process, it is critical to precisely control the accelerator parameters in order to produce a high quality electron beam during acceleration and compression. In particular, the final energy and peak current are very sensitive to system jitter [3]. To minimise sensitivity to jitter, a longitudinal feedback system is required [4]. In this thesis we first consider the beam-based feedback system implemented at the Linear Coherent Light Source (LCLS), where the correction is computed with a PID controller (described in Sec. 5.2). Although a PID algorithm is simple to implement, its tuning can be difficult and it suffers from several limitations [5, 6], including a lack of adaptability and poor response to high

jitter frequency (see Sec. 5.5.2). These limitations are observed in the non suppression or, in some cases, the amplification of jitter frequencies. The gains of the PID controller also need re-tuning when jitter characteristics change. A Matlab GUI is developed to implement a PID controller for the FERMI@Elettra machine (see Chapter 5). This work expands on previous studies that were done at the LCLS [4] and demonstrates the limitations of this type of controller. A coupled feedforward - feedback control scheme based on a Neural Network (NNet) is then proposed (see Chapter 5) to overcome some of these limitations.

The second aspect of linear accelerator performance optimisation lies in the fact that, unlike synchrotrons, FELs produce only a specific wavelength, which is obtained by adjusting of the electron beam parameters. This requires readjustment to elements of the linear accelerator in order to "tailor" the light produced to the users' specifications. This is currently performed by operators and is time consuming. To address this problem an Artificial Intelligence (AI) based system is considered. In this approach, NNets have been utilised along with a technique known as Neuro-Evolution of Augmenting Topologies (described in Chapter 6), to dynamically adjust their structure. This method draws on the latest developments in AI for video games. The present thesis considers how AI technology can be adapted and applied to optimisation problems for accelerator operation. Moreover, we show how, by building a system that has the ability to learn through time and from its interaction with the accelerator, it is possible to build a single system that can address problems in both beam-based control and beam line tuning.

## 1.3 Why neural networks?

NNets have been widely exploited in many domains, such as in astroparticle physics [7], chemistry [8, 9], robotics [10] and finance [11, 12], to name a few. They have provided solutions to a wide range of problems, including: classification problems [13], time series forecasting [14] and speech recognition [15]. Because NNets are universal approximators, they have the ability to learn from examples and can work naturally with a large number of inputs and outputs; in particular NNets are well suited to applications in control engineering [16]. For example, they have been used in industry for controlling mechanical disturbances [17] and heating regulation [18]. NNets have also shown facility in solving difficult analysis, designing new devices and learning from data. Their numerous qualities, as well as their proven record of success in many different fields make them an attractive solution to control applications in accelerator science.

## 1.4 Neural networks in accelerator science

NNets were first considered in accelerator science in 1987 by Higo *et al.* [19]. In this work, the application of AI was investigated to maintain a golden orbit, with a potential application at the Stanford Linear Collider. This problem was chosen, because maintaining a system in an orbit state is easier than achieving this state in the first place. However, the paper did not report a successful experimental implementation of the AI strategy. During this same year, Weygand [20] reported the development of a knowledge-based expert system at Brookhaven National Laboratory. The intention of this project was to help reduce the down time of the Heavy Ion Transfer Line, after a shut down of the machine, or when operation conditions are modified. This system showed the capability of simple tuning tasks, but it was static in the sense that it did not consider changes in the running conditions while tuning the beam.

In 1991 NNets were considered for feedback systems, and in particular for the stabilisation and optimisation of collisions in the Stanford Linear Collider (SLC) at SLAC [21]. Although the preliminary simulation results were promising, the project did not lead to a successful experimental realisation of a beam-based controller. The NNet controller was not able to operate the control task when presented with data from an operating accelerator environment; despite much effort to address pragmatic issues of implementation, they remained unresolved[3].

In 1992 a NNet-based beam diagnostic system was successfully implemented to detect large errors in measurements of beam parameters (i.e. faulty diagnostics) and magnetic field defects (i.e. faulty accelerator elements) [22]. However, this was not applied to control applications. In 1993 an attempt was made to build a NNet controller for orbit correction [23, 24]. Despite promising initial simulations, the implementation of the system for real-time machine operation was never reported. Another attempt to built a global orbit feedback using NNets at the Pohang Light Source was reported in 2000 by Kim *et al.* [25]. Again, despite encouraging simulations, a successful implementation of the system for real-time control was never reported.

A successful implementation of a NNet system for orbit correction was reported in 2004 by Hitaka *et al.* [26] at the KEK Proton Synchrotron. In their approach, a NNet was used to model the relation between beam loss and magnet settings, in order to optimise the configuration. However, the training was done off line using recorded data and was not implemented for real-time feedback.

[3] H. Shoaee, private communication, January 31, 2008.

In 1990, Howell *et al.* [27] reported an attempt to implement a NNet-based controller for a negative ion accelerator source. In their work a NNet was trained off line[4], with data collected from the ion source in order to model the response of the machine. This model was in turn used to optimise the parameters of the ion source. The approach assumed that the characteristics of the source did not change over time and that no anomalous operating conditions were encountered. However, in practice this is not the case and results were of limited utility.

In 1992 a successful NNet implementation for optimisation and control of a small-angle negative ion source was reported by Mead *et al.* [28]. Their approach consisted of mapping the operating space of the ion source coarsely with a scan procedure and fitting the data with a NNet. The system was employed over a number of years for tuning the the machine after short interruptions and cold startups. However, the system was not used for real-time control.

Between 1992 and 1995 the SETUP project was carried out as a collaboration between CERN and the Petersburg Nuclear Physics Institute [29]. Its purpose was to diagnose the control equipment of the CERN Proton Synchrotron. To this end, existing control procedures were programmed in an object-oriented language. When run online, a procedural reasoning system made decisions regarding the application of appropriate procedures. The system performed successful initialisation of accelerator equipment, but was never utilised to perform real-time beam-based control.

The ARCHON project was implemented at CERN between 1993 and 1996. This large project in the area of Distributed Artificial Intelligence (DAI) [30] was developed for the control and diagnosis of faults at the Proton Synchrotron (PS) [31]; it was also used to manage the electricity distribution system at the Spanish facility Iberdrola [32]. DAI was chosen to handle problems in highly complex systems, such as particle accelerators, which monitor thousands of variables. The approach consists of decomposing the problem into smaller tasks handled by expert systems (agents) and integrating these tasks through interaction and knowledge sharing between the agents. ARCHON was a successful project that was run online both at the PS and Iberdrola. However, its applications remained restricted to the detection and management of faults and was never considered for beam-based control systems or optimisation applications.

[4]The training of a neural network can be performed either online or off line. In the first approach, the network's parameters will be adjusted in real time as new data become available. In the second approach the network is only trained once when the whole data set has been collected.

In 1995 another attempt was made by Klein *et al.* [33] to implement an artificial intelligence system on an accelerator, specifically for beamline tuning. The system used a combination of methods, included NNets, fuzzy logic and fuzzy pattern matching, analytical techniques and genetic algorithms [34]. All these methods were incorporated into a single module and the project was developed over several years [35, 36]. The system prototype was tested at Brookhaven and Argonne National Laboratories [37]. Despite promising results, the Department of Energy (DOE) did not proceed with the project; subsequently developers of the prototype were unable to continue the research project[5].

In 1999 Atanasova *et al.* [38] proposed a NNet controller to enhance the stability of power systems. Their work included simulation results but not experimental outcomes. During that same year, Fortuna *et al.* [39] reported the successful implementation of an AI system for the detection of faults in Tokamaks. The strategy exploited a NNet to model expected sensor responses in order to help identify faults. The system was complemented with a fuzzy logic inference system to classify the detected faults. The system was implemented and reported to have successfully accomplished its tasks. In 2000 Ribes *et al.* [40] reported another successful system using NNets for fault diagnosis at the CEA high current linear accelerator in France. In 2001 Buceti *et al.* [41] reported the implementation of a similar AI system for the validation of interferometry density measurement at the ENEA-FTU Tokamak, confirming the suitability of the methodology.

Another interesting application of NNets in accelerator physics is the Parametric Universal Non - Linear Dynamic Approximator (PUNDA) model [42]. PUNDA has been successfully applied for modeling beam optics [43], beam loss in synchrotron light sources [44], and the phase space of an RF photoinjector [45]. However, PUNDA has not been applied to control systems.

In 2007 the use of NNets was reported at the LANSCE accelerator to tune the gains of a Proportional-Integral (PI) feedback controller for the low level RF system [46]. The NNet helped solve the tuning problem for the controller, but did not address deficiencies in the PI algorithm.

This brief survey of the literature shows that considerable effort has been devoted to adapting AI systems for operational accelerator environments over the past two decades. However, despite successful applications in accelerator science (i.e. data modeling, fault diagnosis or beam line tuning), to the best of our knowledge, NNets have not yet been successfully applied to real-time beam-based control. A promising approach by Klein *et al.* [33]

[5] P. Clout, private communication, May 4, 2010.

in 1995 was unfortunately not developed further because of a lack of funding. Nevertheless, in other areas of science and engineering, NNets have been utilised for real-time control. The demonstrable interest and need for these adaptive systems in accelerator science, combined with a successful record in many other areas of engineering is the principal motivation for the present thesis.

## 1.5 Outline of the thesis

#### Chapter 2

Chapter 2 gives an overview of the theory of control, introducing the necessary definitions that will be used throughout this thesis. The notions of Laplace and z-transforms are introduced, since they are most important for the design of controllers. In particular, the PID algorithm will be discussed and the concept of feedforward compensation introduced.

#### Chapter 3

The main components of the three Linacs studied in this work are described in Chapter 3; namely, the Australian Synchrotron Linac, the Linac Coherent Light Source (LCLS) at SLAC, and the FERMI@Elettra Linac. Similarities and differences between the machines are discussed, as well as the available beam diagnostic components that are used for control experiments.

#### Chapter 4

Chapter 4 covers the background on linear accelerator physics relevant to the control of longitudinal beam parameters. Mathematical expressions are derived to compute the deviation of the beam energy and peak current resulting from jitter. These equations form the basis for the simulations reported in Chapter 5.

#### Chapter 5

Chapter 5 focuses on simulation studies for the implementation of a PID controller at the FERMI@Elettra Linac. Here we expand on earlier work carried out at the LCLS, which is based on the equations governing longitudinal dynamics developed in Chapter 4. This includes the development of a Matlab Graphical User Interface (GUI), which facilitates the study of realistic scenarios, involving more components in the feedback system, e.g., a range of possible actuators, jitter characteristics and codes to simulate the

machine. This framework accomodates a range of studies, including jitter, control strategies and diagnostic systems that can be separately selected in the GUI.

### Chapter 6

The necessary background on NNets is discussed in Chapter 6. The structure of NNets and the salient training algorithms are described. Advanced concepts such as genetic algorithms, which are used to evolve the structure of NNets through time (see Chapter 8) are also introduced.

### Chapter 7

Chapter 7 discusses the design of a combined feedforward - feedback control system that overcomes some of the limitations related to the PID algorithm. Experimental results performed at the Australian Synchrotron are then presented, followed by coupled energy and bunch length control results obtained at the LCLS. The performance, limitations and adaptability of the control scheme are discussed.

### Chapter 8

Chapter 8 describes a system that overcomes the limitations of the feedforward - feedback control scheme, which was analysed in Chapter 7. The research is orientated towards developing an evolving controller. We explore the concept of an evolving NNet. The control system is considered as an optimisation problem and a tool, based on evolving NNets, is developed for optimisation purposes. Successful test results are reported for the tuning of the beam energy spread and beam transmission (i.e. percentage of electrons transmitted from the start to the end of the linear accelerator) at the Australian Synchrotron. To complement these studies, we present a preliminary control test performed at FERMI, which suggests the suitability of the methodology to beam-based control applications.

### Chapter 9

This chapter describes the generalisation of the optimisation tool developed in Chapter 8 to $N$ variables. Two approaches are proposed to integrate the systems developed in Chapters 7 and 8, resulting in a single system. We discuss how the integrated system can learn from its interaction with the machine in order to enhance the control of the accelerator. Finally, we review the results of the thesis and suggest areas for further investigation.

# 2 Elements of control theory

## 2.1 General considerations

This chapter gives an overview of control theory relevant to the work presented in the thesis. Because the electron beam consists of electron bunches, rather than a continuous flow of particles the system may be considered to be discrete in nature. We are therefore interested in intrinsically discrete time systems, without concomitant issues of sampling. The response of the longitudinal beam parameters will also be considered to be linear, or linearisable in the region of interest, for purposes of control. We therefore consider discrete linear time-shift invariant control systems. We start by describing the fundamental concepts of control theory, providing definitions that will be used later to characterise the controller performance. The system (in our case the accelerator) and its controller will be described using a mathematical model. In particular, we discuss the implementation of a PID controller as it is of particular interest in the work presented in Chapters 5 and 7. Finally, we introduce the salient features of feedforward control, which is utilised in later chapters.

## 2.2 Definitions

In control theory there are several factors that characterise a system, which must be taken into consideration. The most important of these is stability of the system. Other characteristics, such as the initial overshoot, the speed of response, the steady-state error, and parameter sensitivity are also critical determinants in designing a control system [47]. In what follows we formally define the notion of stability and describe the two main "regimes" associated with the response of a control system; namely, the transient and steady state responses. These two regimes are conceptually important as they form the basis for characterising a control system.

### 2.2.1 Stability

The stability of a system relates to its response to inputs or perturbations. A system that remains in a constant state unless affected by an external action and which returns to a constant state when the external action is removed is considered to be stable. The system is said to be unstable if its output increases without bound [48]. Formally, stability is defined as follows. Let us consider a system described by the general discrete time difference equation [49]:

$$x(i+1) = f(x(i), u(i), i), \tag{2.1}$$

where $x(i+1)$ denotes the state[1] of the system at time step $i+1$ whose input $u(i)$ in a time series with $i = 0, 1...i_r$. The solution $x_0(i+1)$ of the system is said to be stable if for any instant $i = i_0$ and for any $\epsilon > 0$ there exists a $\delta(\epsilon, i_0)$ such that:

$$\|x(i_0) - x_0(i_0)\| \leq \delta, \tag{2.2}$$

which implies:

$$\|x(i) - x_0(i)\| \leq \epsilon, \quad \forall i \geq i_0. \tag{2.3}$$

Here $\|x\|$ denotes the Euclidean norm. This definition of stability is commonly known as Lyapunov stability [49]. It means that the state of the system at any time $i > i_0$ is guaranteed to be bounded for an initial state $x(i_0)$ chosen not too far from the nominal solution $x_0$. In control theory, a stronger form of stability called asymptotic stability is often required. For the discrete time difference equation (2.1), the nominal solution $x_0(i)$ is said to be asymptotically stable if [49]:

1. It is stable in the sense of Lyapunov; and
2. $\forall i_0$ there exists a $\rho > 0$ such that $\|x(i) - x_0(i))\| \to 0$ when $i \to \infty$, and when the initial deviation is in the region defined by $\|x(i_0) - x_0(i_0))\| < \rho$.

The stability of a controller is a key criterion and is most often evaluated by applying a step change to a set point or load and evaluating the settling time, residual and maximum errors [50]. These characteristics are illustrated in Fig. 2.1, which we will use to define the associated concepts of residual error, overshoot and rise time.

[1] The internal state variables are the smallest possible subset of system variables that can represent the entire state of the system at any given time [49].

### 2.2.2 Transient and steady state regimes

The duration of a control process is commonly decomposed into two regimes following a step change; the transient and the steady-state regimes. The transient regime is the time required to achieve the output target value. The corresponding output variation is called the transient response. For example, in Fig. 2.1 this limit is set to 10% of the target value. This regime is then followed by the steady state response.

In control systems we also often refer to the rise time and the settling time. The rise time is commonly defined as the time for a signal to go from 10% to 90% of its final value [51]. The settling time is the time elapsed from the application of an ideal instantaneous step input to the time at which the output evolves to, and remains within a specified error band from the new set point (target value) as illustrated in Fig. 2.1. This includes a propagation delay, plus the time required for the output to recover from the overshoot and to settle within the specified error.

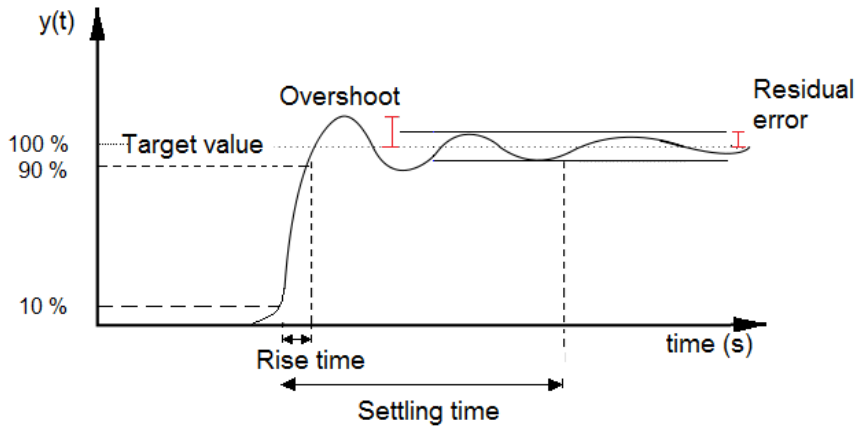



Figure 2.1: Definitions related to the transient and steady state responses. The transient response starts when a step input is applied and ends when the error falls below 10% of the target value. The steady state regime corresponds to the response of the system thereafter.

To tune a controller, one must optimise both the transient and the steady state responses. The optimisation of the transient response consists of minimising the maximum initial error, or overshoot, the settling time and the corresponding speed of response, i.e. to minimising the rise time [50]. To optimise the steady state response one must minimise the residual errors between a controlled variable and its set point. In what follows we will introduce the Laplace and z-transforms, which are useful for analysing signals and control systems, and for optimising the transient and steady responses of the system.

## 2.3 Transform analysis

The unilateral Laplace transform is pivotal to the description of linear time-shift invariant systems; it allows a system to be characterised in the complex frequency domain. The transform is defined as [52]:

$$\mathscr{L}(f(t)) = \int_0^\infty e^{-st} f(t) dt, \tag{2.4}$$

where $t$ is the time variable, $f$ is the signal to be transformed and $s$ a complex (frequency) parameter. The Laplace transform has a number of important advantages. For example, if $g(t)$ is the response of the system to a unit impulse at $t = 0$, then the response of the system output $y(t)$ to any input $u(t)$ is given by the convolution:

$$y(t) = \int_0^t g(t-\tau)u(\tau)d\tau. \tag{2.5}$$

The Laplace transforms of $u(t)$ and $g(t)$ are denoted by $U(s)$ and $G(s)$, which allows us to rewrite Eq. (2.5) in the complex frequency domain as:

$$Y(s) = G_s(s)U(s), \tag{2.6}$$

where the convolution in the time domain has become a multiplication in the frequency domain. The ratio $G_s(s) = Y(s)/U(s)$ is commonly referred to as the transfer function (or more generally the transfer matrix) of the system. The Laplace transform allows us to model the properties of a system in terms of its transfer matrix. The inverse transformation from $Y(s)$ back to $y(t)$ is not always required to design a controller, as we can use the pole placement technique, which will be discussed in Sec. 2.7. However, if it is necessary, the inverse Laplace transform allows us to recover $f(t)$, using:

$$f(t) = \mathscr{L}^{-1}\{G_s(s)\} = \frac{1}{2\pi j} \lim_{T\to\infty} \int_{\gamma-jT}^{\gamma+jT} e^{st} G_s(s) ds, \tag{2.7}$$

where $j = \sqrt{-1}$ and $\gamma$ is a real number chosen so that the integration path ensures convergence [2] of $G_s(s)$ [53]. For a discrete time system the integral in Eq. (2.4) is replaced by a summation over the discrete time steps $i$. This defines the unilateral z-transform of a discrete time series $Y(i)$:

$$\mathbb{Z}\{y(i)\} = Y(z) = \sum_{i=0}^{\infty} z^{-i} y(i), \tag{2.8}$$

[2] The convergence region defines the region of $t$ for which the Laplace transform converges, i.e. $|\int_{-\infty}^{\infty} g(t)e^{-st}| < \infty$ for a continuous time system. For a discrete time system the integral becomes a summation over the instants $i$, i.e. $|\sum_{i=-\infty}^{+\infty} g(i)z^{-i}| < \infty$.

where $z = Re^{j\theta}$ is a complex number with modulus $R$ and argument $\theta$. If the z-transform $X(z)$ of a time series $x(i)$ is known analytically, the inverse z-transform back to the time domain is given by:

$$x(i) = \frac{1}{2\pi j} \oint_{|z|=R} X(z) z^{i-1} dz, \tag{2.9}$$

where the contour integral is evaluated (around the origin) in the complex plane. In what follows we will be concerned with the z-transform, since we are interested in discrete time systems. However, many of the results presented in the following sections are derived from the continuous theory.

## 2.4 Linear discrete time systems

It often happens that we can only observe a physical system at a sequence of instants $t_i$, $i = 0, 1, 2...i_r$ rather than continuously. This is particularly relevant to an electron beam which consists of distinct bunches rather than a continuous flow of particles. The bunch index $i$ corresponds to the instant $t_i = i/f$, where $f$ is the beam repetition rate. Since we are principally concerned with linear systems, the response of the beam parameters to the actuators must be linear, or reasonably linearisable in the specified region of interest, for purposes of control. A general discrete time system is expressed by the state difference equation (2.1):

$$x(i+1) = f(x(i), u(i), i).$$

The corresponding output of the system $y(i)$ can be written as a function of the system's state $x(i)$ and its input $u(i)$:

$$y(i) = g(x(i), u(i), i). \tag{2.10}$$

These equations can also be expressed in the following matrix form [49]:

$$x(i+1) = A(i)x(i) + B(i)u(i), \tag{2.11}$$

and

$$y(i) = C(i)x(i) + D(i)u(i), \tag{2.12}$$

where in the general case, the parameters $x, u$ and $y$ are vectors whose components can have different units. For example, the state $x$ might include the

beam transverse coordinates (in mm) and the energy (in MeV), while the input $u$ might include the phase (in degrees) and voltage (in volts) of an accelerating structure. The matrix $A$ describes the relation between the state of the system $x(i+1)$ and its past state $x(i)$, while $B$ describes the response of $x(i+1)$ to the system input $u(i)$. The response of the system output $y(i)$ is related to the state of the system $x(i)$ and its input $u(i)$ via the matrices $C$ and $D$, respectively. Since the accelerator is considered to be a time-shift invariant system $A, B, C$ and $D$ are time-shift invariant. Applying the z-transform to Eqs. (2.11) and (2.12) we obtain, for a time-shift invariant system [49]:

$$X(z) = (zI - A)^{-1}BU(z) + (zI - A)^{-1}zx(0), \tag{2.13}$$

and

$$Y(z) = (CB(zI - A)^{-1} + DU(z) = G_s(z)U(z), \tag{2.14}$$

where $I$ is the identity matrix, $G_s(z)$ is the transfer matrix of the system, and $x(0)$ is the first element of the time series, i.e. the initial condition. It can be shown that [49]:

$$det[G_s(z)] = \frac{\psi(z)}{\phi(z)} = \frac{\psi(z)}{det(zI - A)}, \tag{2.15}$$

where $\psi(z)$ and $\phi(z)$ are polynomials in $z$. The transfer matrix $G_s$ is very useful for studying the frequency response of the system. Let us write the input of the system in the complex form:

$$u(i) = u_m e^{j\omega i}, \tag{2.16}$$

where $u_m$ is the magnitude of the input and $\omega$ is the angular frequency of the input. In can be shown [49] that the general solution of the inhomogeneous state difference equation (2.11) is given by:

$$x(i) = A^i a + (e^{j\omega} I - A)^{-1} B u_m e^{j\omega i}, \tag{2.17}$$

where $a$ is a vector of arbitrary constant values. The first term on the right hand side of Eq. (2.17) is the general solution of the homogeneous difference equation (2.11), while the second term is a particular solution of that same equation. We also note that these terms correspond to the transient and the steady state responses of the system, respectively. If the system is

asymptotically stable, the first term must vanish as $i \to \infty$, i.e. as the system reaches steady state. Introducing Eqs. (2.16) and (2.17) into Eq. (2.12), the steady state response of the system is then given by:

$$y(i) = C(e^{j\omega}I - A)^{-1}Bu_m e^{j\omega i} + Du_m e^{j\omega i}. \tag{2.18}$$

From Eq. (2.18) the transfer matrix $G_s(z)$ of the system in Eq. (2.14) can be rewritten as:

$$G_s(e^{j\omega}) = C(e^{j\omega}I - A)^{-1}B + D. \tag{2.19}$$

It is clear that the transfer matrix of the system depends on the frequency of the input. Because the frequency response of the system $G_s$ cannot be changed, the controller must be designed to compensate for undesirable frequency responses of the system, such that it can meet the stability criteria over the specified frequency band. This will be studied in more detail in Sec. 2.7.

In the following section we discuss how the system's response can be characterised in order to derive criteria for the evaluation of the controller performance. The results obtained will be used in Sec. 2.6, where the z-transform of the transfer function is exploited to design a PID controller.

## 2.5 Analysis of linear control systems

### 2.5.1 System description

In what follows we describe the equations that define a closed loop controller and discuss the basic design objectives related to the transient and steady state responses. First, we define the state equations of the system and its controller. The system described by Eqs. (2.11) and (2.12) can be generalised according to the following equations [49]:

$$x(i+1) = Ax(i) + Bu(i) + v_p(i), \tag{2.20}$$

$$x(i_0) = x_0, \tag{2.21}$$

$$y(i) = Cx(i) + v_m(i), \tag{2.22}$$

and

$$z(i) = Dx(i). \tag{2.23}$$

where $x$ and $u$ describe the state of the system and the input is defined in Sec. 2.4. The variable $v_p(i)$ is the disturbance, $v_m(i)$ is the noise associated

with the measurement of the variable $y(i)$, and $z(i)$ is the controlled variable. Here we will consider the general case where $v_p$ is a stochastic process. The matrices $A, B$ and $C$ are those defined in Sec. 2.4, while $D$ defines the response of the controlled variable $z(i)$ to the state $x(i)$ of the system. We also need to define the state equations for the controller, which are written in terms of the following difference equations:

$$q(i+1) = Lq(i) + K_r r(i) - K_f y(i), \tag{2.24}$$

$$q(i_0) = q_0, \tag{2.25}$$

and

$$u(i) = Fq(i) + H_r r(i) - H_f y(i). \tag{2.26}$$

where $q(i)$ is the state of the controller, $u(i)$ is its output, $r(i)$ is the reference variable and $q(i_0)$ is the first element of the controller state in the time series. The variable $u(i)$ also corresponds to the input of the system as depicted in Fig. 2.2. The response of the controller state $q(i+1)$ depends on its previous state $q(i)$ and the difference between the reference signal $r(i)$ and the actual output of the system $y(i)$. The response matrices $L, H_r, H_f, K_r$ and $K_f$ describe the relation between the parameters $q, r, u$ and $y$. The indexes $r$ and $f$ refer to the reference and feedback variables, respectively. We note that these equations correspond to a time-shift invariant system, since the matrices $A, B, C, D, F, H_l, H_r, K_l, K_r$ and $L$ are all time-shift invariant. Figure 2.2 illustrates the system and the controller scheme.

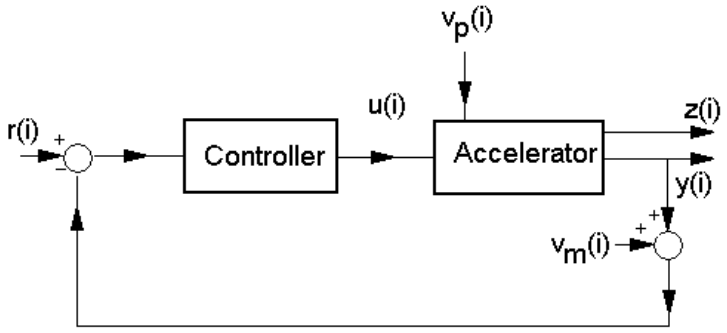



Figure 2.2: Schematic of a closed loop control system. The controller produces the input $u(i)$ to the system, given a desired reference value $r(i)$. The system is also subject to disturbances $v_p$(i). The output is $y(i)$, whose measurement can be subject to a disturbance $v_m$. The controlled variable is $z(i)$.

A measure of the system performance is the error between the output and the desired reference. This error is expressed as:

$$e(i) = z(i) - r(i), \tag{2.27}$$

or in terms of the corresponding z-transform:

$$E(z) = Z(z) - R(z). \tag{2.28}$$

In the regulator[3] problem, $r(i)$ is constant ($r(i) = r_0$), while according to Eq. (2.20) the controlled variable $z(i)$ contains the frequency components of the disturbance $v_p(i)$. Since the tracking error $e(i)$ is a function of $z(i)$, it also contains the frequency components of the disturbance $v_p(i)$. Moreover, $e(i)$ still contains information on the frequency components if we consider $v_p(i) = constant$ and if $r(i)$ contains the frequency components of $v_p(i)$. This allows us to study the system's stability as a function of its input rather than the perturbation $v_p(i)$, which simplifies calculations (see Secs. 2.5.2 and 2.5.3). For this, we first decompose the reference variable $r(i)$ into a constant part $r_0$ and a variable part $r_v(i)$:

$$r(i) = r_0 - r_v(i). \tag{2.29}$$

By doing so we can study the response of the system to the reference variable, which will simplify calculations and let us derive conditions for the system's stability. To evaluate the performance of a control system, we will use the mean square tracking error $C_e(i)$ and the mean square input $C_u(i)$, defined as follows [49]:

$$C_e(i) = \mathbf{E}\{e^T(i)W_e(i)e(i)\}, \tag{2.30}$$

and

$$C_u(i) = \mathbf{E}\{u^T(i)W_u(i)u(i)\}. \tag{2.31}$$

where $\mathbf{E}$ is the expectation value operator, $(\cdot)^T$ is the transpose, and $W_e(i)$ and $W_u(i)$ are weighting matrices. Usually $W_e(i)$ is diagonal and $C_e$ is the weighted sum of the mean square tracking error of the controlled variable. In particular when $W_e = 1$ (matrix of size 1x1) then $\sqrt{C_e}$ corresponds to the rms tracking error. Equations (2.30) and (2.31) are very useful because in a control system we are required to minimise the mean square tracking error, while maintaining the mean square input value within reasonable limits. The mean square input has the following relevance. Consider the need to

[3] In control theory we distinguish between the tracking problem, where the reference variable is changing frequently, and the regulator problem, where the reference variable remains constant for a long period of time.

impose an upper value on the voltage (i.e. the system input) that can be applied to an RF accelerating cavity. This can limit the range over which the input voltage can be changed (if the initial value is already close to the upper limit). Fixing a maximum value on the mean square input ensures that the input voltage will not exceed the prescribed value.

As described in Sec. 2.2.2, the response of the system has two characteristic regimes, the transient and the steady-state responses. The optimisation of the controller must consider both regimes, which have different criteria. In the following sections we will show how the transient and the steady states responses of the control system can be optimised. To do so we develop Eqs. (2.30) and (2.31) further and derive the conditions that guarantee the stability of the control system.

### 2.5.2 The steady-state response

In this section we study the response of the linear discrete time control system described by Eqs. (2.20) - (2.26) to the reference variable $r$. As discussed in Sec. 2.2.1 the most important objective of a control system is to ensure stability. Asymptotic stability can be guaranteed by ensuring that the steady state mean square tracking error and the steady state mean square input converge as $i \to \infty$ (corresponding to the the steady state response of the system). We are thus interested in the limit of $C_e(i)$ and $C_u(i)$ when $i \to \infty$. We introduce the definitions:

$$C_{e,\infty} = \lim_{i\to\infty} C_e(i), \tag{2.32}$$

and

$$C_{u,\infty} = \lim_{i\to\infty} C_u(i). \tag{2.33}$$

Now let us decompose the tracking error $e(i)$ into two parts; a deterministic or average part $\bar{e}(i) = \mathbf{E}\{e(i)\}$ and a non-deterministic part $\tilde{e}(i)$. We write:

$$e(i) = \bar{e}(i) + \tilde{e}(i). \tag{2.34}$$

Likewise for $u(i)$ we have $u(i) = \bar{u}(i) + \tilde{u}(i)$. Using these definitions, Eqs. (2.30) and (2.31) can be rewritten as a sum of deterministic and non-deterministic contributions [49]:

$$C_e(i) = \bar{e}(i)^T W_e(i)\bar{e}(i) + \mathbf{E}\{\tilde{e}(i)^T W_e(i)\tilde{e}(i)\} = C_{\bar{e}}(i) + C_{\tilde{e}}(i), \tag{2.35}$$

and

$$C_u(i) = \bar{u}(i)^T W_u(i)\bar{u}(i) + \mathbf{E}\{\tilde{u}(i)^T W_u(i)\tilde{u}(i)\} = C_{\bar{u}}(i) + C_{\tilde{u}}(i). \tag{2.36}$$

Moreover, we introduce the following z-transforms:

$$Z(z) = T(z)R(z), \tag{2.37}$$

and

$$U(z) = N(z)R(z), \tag{2.38}$$

where $T(z)$ and $N(z)$ are the response matrices in the z-domain. Inserting these equations into Eq.(2.28), the z-transform of $e(i)$ can be written as:

$$E(z) = (T(z) - I)R(z). \tag{2.39}$$

Now let us consider $C_{\bar{e}}(i)$ and $C_{\bar{u}}(i)$, which are the contributions arising from the constant part of the reference variable (see Eqs. (2.35) and (2.36)). Using the final value theorem[4] and the fact that $r_0$ is constant[5], when $i \to \infty$ $C_{\bar{e}}(i)$ and $C_{\bar{u}}(i)$ are given by:

$$\lim_{i\to\infty} C_{\bar{e}}(i) = \mathbf{E}\{r_0^T[T(1) - I]^T W_e[T(1) - I]r_0\}, \tag{2.40}$$

and

$$\lim_{i\to\infty} C_{\bar{u}}(i) = \mathbf{E}\{r_0^T N(1)^T W_e N(1) r_0\}. \tag{2.41}$$

where we assume that the weighting matrices $W_e$ and $W_u$ are time-shift invariant. Now let us consider the variable part $r_v(i)$ of the reference variable $r(i)$. It can be shown that for a stochastic process $\tilde{e}(i)$ [49]:

$$E\{\tilde{e}^T(i)W_e\tilde{e}(i)\} = \frac{1}{2\pi}\Big(\ tr\int_{-\pi}^{\pi}\Sigma_{\tilde{e}}(\omega)d\omega\ \Big), \tag{2.42}$$

[4]The final value theorem states that if $\lim_{i\to\infty} x(i)$ exists, then $\lim_{i\to\infty} x(i) = \lim_{z\to 1}(z-1)X(z)$. This theorem is valid only if the poles of $(z-1)X(z)$ are inside the unit circle.

[5]The z-transform of 1 is $z/(z-1)$ and since $\mathbb{Z}(r_0 1) = \mathbb{Z}(r_0)\mathbb{Z}(1)$, the z-transform of $r_0$ is given by $r_0 z/(z-1)$ .

where $\Sigma_{\tilde{e}}(\omega)$ is the power density matrix associated with the stochastic process $\tilde{e}(i)$, and $tr$ is the trace operation. Because $e(i) = z(i) - r(i)$, the matrix $\Sigma_{\tilde{r}}(\omega)$ is associated with the stochastic process $r_v$, which has a power spectral density matrix given by [49]:

$$\Sigma_r(\omega) = \sum_{i=-\infty}^{\infty} z^{-i} R_r(i), \tag{2.43}$$

where $R_r(i-k) = \mathbf{E}\{[r(i)-r_0][r(k)-r_0]^T\}$ is the covariance matrix of the process, which in this case is the reference variable $r(i)$. The power spectral density matrix represents the spectrum of the signal containing the frequency content of a stochastic process. The power spectral density matrix $\Sigma_{\tilde{e}}(\omega)$ is calculated from the "similarity" transformation:

$$\Sigma_{\tilde{e}}(\omega) = S(e^{j\omega})\Sigma_r(\omega)S^T(e^{j\omega}), \tag{2.44}$$

where $S$ is the z-transform matrix such that $E(z) = S(z)R(z)$. From Eq. (2.39) we identify $S(z) = T(z) - I$. We can thus write:

$$\Sigma_{\tilde{e}}(\omega) = (T(e^{j\omega}) - I)\Sigma_r(\omega)(T(e^{j\omega}) - I)^T. \tag{2.45}$$

Inserting Eq. (2.45) into Eq. (2.42), we obtain $C_{e,\infty}$ from:

$$\lim_{i\to\infty} C_{\tilde{e}}(i) = tr\{\frac{1}{2\pi}\int_{-\pi}^{\pi}[T(e^{-j\omega}) - I]^T W_e[T(e^{j\omega}) - I]\Sigma_r(\omega)d\omega\}. \tag{2.46}$$

Similar reasoning gives $C_{u,\infty}$ as:

$$\lim_{i\to\infty} C_{\tilde{u}}(i) = tr\{\frac{1}{2\pi}\int_{-\pi}^{\pi}N(e^{-j\omega})^T W_u N(e^{j\omega})\Sigma_r(\omega)d\omega\}. \tag{2.47}$$

Combining Eq. (2.40) with Eq. (2.46) and Eq. (2.41) with Eq. (2.47) we obtain the final expressions for $C_{e,\infty}$ and $C_{e,\infty}$:

$$\begin{aligned} C_{e,\infty} \quad = \quad & \mathbf{E}\{r_0^T[T(1) - I]^T W_e[T(1) - I]r_0\} + \\ & tr\{\frac{1}{2\pi}\int_{-\pi}^{\pi}[T(e^{-j\omega}) - I]^T W_e[T(e^{j\omega}) - I]\Sigma_r(\omega)d\omega\}, \end{aligned} \tag{2.48}$$

and

$$C_{u,\infty} = \mathbf{E}\{r_0^T N(1)^T W_e N(1) r_0\} + tr\{\frac{1}{2\pi}\int_{-\pi}^{\pi} N(e^{-j\omega})^T W_u N(e^{j\omega})\Sigma_r(\omega)d\omega\}. \quad (2.49)$$

The preceding analysis allows us to study the stability of the system. For clarity we consider a single input - single output case and refer to the literature [49] for the generalisation to multiple input-output systems. We can draw the following conclusions:

1. To maintain a small tracking error in the steady state, the transmission matrix $T$ must be designed such that (with $W_e$ time-shift invariant):

$$\Sigma_r(\omega)|T(e^{j\omega}) - 1|^2 \leq \delta_e, \quad (2.50)$$

where $\delta_e$ is small $\forall\omega$ and is defined according to the control requirement, i.e. the maximum mean square tracking error $C_{e,\infty}$ that can be tolerated in the steady state. The frequency band of the control system corresponds to frequencies for which

$$|T(e^{-j\omega}) - 1| \leq \epsilon, \quad (2.51)$$

where $\epsilon \ll 1$. This frequency band is commonly known as the bandwidth of the controller.

2. Since the first term in Eq. (2.49) is constant, to obtain a small mean square input in the steady state, the time-shift invariant control system should be designed such that (with $W_u$ time-shift invariant):

$$\Sigma_r(\omega)|N(e^{-j\omega})|^2 \leq \delta_u, \quad (2.52)$$

where $\delta_u$ is small $\forall\omega$ and is defined according to the control requirements, i.e. the maximum mean square input $C_{u,\infty}$ that can be tolerated in the steady state.

We can formulate the transfer matrices $N$ and $T$ in terms of the matrices $A, B, C, D, F, H_r, H_f, K_r, K_f, L$ that describe the system according to Eqs. (2.20) - (2.26). Doing so will allow us to design the controller to meet the requirements of Eqs. (2.50) and (2.52). First, we write the z-transform of the difference equations (2.20)-(2.23) of the system as:

$$X(z) = D(zI - A)^{-1}BU(z) + (zI - A)^{-1}zx_0, \tag{2.53}$$

$$Y(z) = C(zI - A)^{-1}BU(z), \tag{2.54}$$

and

$$Z(z) = DX(z). \tag{2.55}$$

Similarly, the z-transform of the controller equations (2.24)-(2.26) are given by:

$$Q(z) = (zI - L)^{-1}zq_0 + (zI - L)^{-1}K_rR(z) - (zI - L)^{-1}K_fY(z), \tag{2.56}$$

and

$$U(z) = FQ(z) + H_rR(z) - H_fY(z). \tag{2.57}$$

Here we impose the initial conditions $x_0 = q_0 = 0$. To simplify the above equations we define the following:

$$K(z) = D(zI - A)^{-1}B, \tag{2.58}$$

$$H(z) = C(zI - A)^{-1}B, \tag{2.59}$$

$$G(z) = F(zI - L)^{-1}K_f + H_f, \tag{2.60}$$

and

$$P(z) = F(zI - L)^{-1}K_r + H_r. \tag{2.61}$$

Using these definition and substituting Eq. (2.56) into Eq. (2.57), we can write:

$$U(z) = P(z)R(z) - G(z)Y(z), \tag{2.62}$$

$$Y(z) = H(z)U(z), \tag{2.63}$$

and

$$Z(z) = K(z)U(z). \tag{2.64}$$

Using the definitions of $T$ and $N$ in Eqs. (2.37) and (2.38), and by eliminating the appropriate variables we obtain:

$$T(z) = K(z)\Big(\ I + G(z)H(z)\ \Big)^{-1} P(z), \tag{2.65}$$

and

$$N(z) = \Big(\ I + G(z)H(z)\ \Big)^{-1} P(z). \tag{2.66}$$

We have thus arrived at expressions for $N$ and $T$ as a function of the matrices $A, B, C, D, F, H_r, H_f, K_r, K_f, L$ that characterise the system and its controller. We now discuss how the stability of the controller depends on these matrices. As mentioned previously, $A, B, C$ and $D$ are intrinsic to the system and can therefore not be modified. According to Eq. (2.24), the state of the controller $q(i+1)$ depends on its past state $q(i)$, and a term $K_r r(i) - K_f y(i)$, which is a function of the reference variable $r(i)$ and the output variable $y(i)$. Similarly, the output of the controller $u(i)$ depends on its state $q(i)$, and a term $H_r r(i) - H_f y(i)$ (see Eq. (2.26)). The matrices $K_r$ and $K_f$ are related to the properties of the sensors of the accelerator, whereas $H_f$ and $H_r$ relate to the properties of the actuators of the accelerator and must be designed according to these requirements.

From Eq. (2.26) one can see that $F$ has the form of a gain matrix between the controller output $u(i)$ and its state $q(i)$. The matrix $L$ defines how the past state of the controller influences its future state. When all the elements of $L$ approach zero, the state of the controller depends entirely on the measured error between the output and its set point. In this case the controller output can experience major fluctuations, when the output measurement has significant levels of noise. On the other hand, as the elements of the matrix $L$ increase, the contribution of the past state has an increasing influence on the future state in compared to the measured error. In this case, the response of the controller becomes very slow. Similar considerations apply to the gain matrix $F$. The design of $F$ and $L$ thus determines the sensitivity of the controller. According to Eqs. (2.65) and (2.66), $T(z) = K(z)N(z)$ and $K(z)$ is a function of $D$, which means that the response of the input variable and the controlled variable depend on $L$ and $F$ in exactly the same way; however, the response of the controlled variable can also be determined by the matrix $D$.

### 2.5.3 The transient response

To optimise the controller one must also optimise the transient response. This requires one to reduce the rise time and the associated overshoot, so that the system can reach its steady state in minimum time. An estimate of

the settling time $\tau_s$ can be calculated by considering the response of the system to the constant part of the reference variable $r(i)$. In this case a change in $r(i)$ corresponds to a step function. For the system described in Sec. 2.5.1, the solution of the state difference equation (2.20) is given by [49]:

$$x(i) = \Phi(i, i_0)x(i_0) + \sum_{j=0}^{i-1} \Phi(i, j+1)Bu(j), \tag{2.67}$$

where $\Phi(i, i_0)$ is the transition matrix defined by:

$$\Phi(i, i_0) = \begin{cases} A(i-1)A(i-2)...A(i_0) & \text{if } i \geq i_0 \\ I & \text{for } i = i_0. \end{cases} \tag{2.68}$$

If the matrix $A$ is time-shift invariant the transition matrix becomes:

$$\Phi(i, i_0) = A^{i-i_0}. \tag{2.69}$$

The corresponding solution for the system output is given by:

$$y(i) = \sum_{j=i_0}^{i} K(i, j)u(j). \tag{2.70}$$

with

$$K(i, j) = \begin{cases} C\Phi(i, i_0)B & \text{if } j \leq i-1 \\ 0 & \text{for } j = i. \end{cases} \tag{2.71}$$

Suppose that the matrix $A$ has $n$ eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ with corresponding eigenvectors $g_1, g_2, ..., g_n$, from which we form the matrices $\Omega = diag(\lambda_1, \lambda_2, ..., \lambda_n)$, and $T = (g_1, g_2, ..., g_n)$, respectively, where $T^{-1} = (h_1, h_2, ...h_n)^T$ and $h_1, h_2, ...h_n$ are row vectors. The transition matrix (2.69) can thus be rewritten as:

$$\Phi(i, i_0) = T\Omega^{i-i_0}T^{-1}. \tag{2.72}$$

The solution to the difference equation (2.67) is now given by:

$$x(i) = \sum_{k=1}^{n} \lambda_k^{i-i_0} g_k h_k x_0, \tag{2.73}$$

where $x_0 = x(i_0)$. Equation (2.73) shows that all responses are linear combinations involving the eigenvalues $\lambda_k$. If the system is asymptotically stable, the response of all elements of the state variable $x(i)$ are exponentially damped with time constants corresponding to the eigenvalues $\lambda_1, \lambda_2, ...\lambda_n$ [49]. The response of a linear system output can thus be written as:

$$y(i) = \sum_{k=1}^{n} C_k e^{i\lambda_k}, \tag{2.74}$$

where the coefficients $C_k$ are defined by the initial conditions. Since the 1% settling time $\tau_s$ of the eigenvalue $\lambda_k$ is given by $4.6/\lambda_k$, we can estimate the upper limit on $\tau_s$ as:

$$\tau_{s,1\%} \leq 4.6 \max_k \frac{1}{|\lambda_k|}, \tag{2.75}$$

where the max operation selects the largest value of $1/|\lambda_k|$ among all the eigenvalues $\lambda_k$. In most cases this provides a good estimate of the settling time. However, this method can poorly represent the real behavior of the system in the discrete time case. This is because, although the system's behavior is satisfactory at the sampling instants, it can exhibit significant overshoot between samples. This is particularly evident when the variable part $r_v(i)$ is non-zero. In our case this is not as important issue, since we are interested in the bunch-by-bunch control, i.e. there is no data loss since we can measure the parameters of every bunch.

Alternatively, the settling time can be obtained by calculating and plotting the values of $C_e(i)$. Suppose that we want to calculate this for the controlled variable $z(i)$, with corresponding reference variable $r(i)$. We can write:

$$C_e(i) = \mathbf{E}\{[z(i) - r(i)]^2\} = \mathbf{E}\{z^2(i)\} - 2\mathbf{E}\{z(i)r(i)\} + \mathbf{E}\{r^2(i)\}. \tag{2.76}$$

To calculate this we define the variance matrix:

$$Q(i) = \mathbf{E}\{\varepsilon(i)\varepsilon^T(i)\}, \tag{2.77}$$

where $\varepsilon(i)$ is the state vector $x(i)$ of the system augmented with the reference variable $r(i)$; we assume that the latter can be written in the form:

$$r(i+1) = c_r r(i) + w(i). \tag{2.78}$$

Here $w(i)$ represents a stochastic process and $c_r$ is a constant. In this case $\varepsilon(i)$ can be written as:

$$\varepsilon(i+1) = M\varepsilon(i) + Nw(i). \tag{2.79}$$

In the simplest situation, $\varepsilon(i)$ contains the controlled variable and its reference variable, i.e. $\varepsilon(i) = (z(i), r(i))^T$. The corresponding variance matrix is:

$$Q(i) = \begin{pmatrix} z^2(i) & z(i)r(i) \\ r(i)z(i) & r^2(i) \end{pmatrix}. \tag{2.80}$$

Using the definition in Eq. (2.77), the mean square tracking error in Eq. (2.76) can be rewritten in terms of the elements $Q_{kl}$ of the variance matrix. In particular, if $\varepsilon(i)$ only contains the controlled variable and the reference variable we can utilise Eq. (2.79) to write:

$$C_e(i) = Q_{11}(i) - 2Q_{12}(i) + Q_{22}(i). \tag{2.81}$$

It can also be established that the variance matrix $Q(i)$ obeys the following relation [49]:

$$Q(i+1) = MQ(i)M^T + NVN^T. \tag{2.82}$$

The mean square tracking error can thus be calculated for each time step $i$, using Eqs. (2.81) and (2.82). Plotting the results allows the settling time to be determined. This method is more laborious than the simple method leading to Eq. (2.75), but it has the advantage of being more reliable, since it contains the response to both the constant and variable part of the reference variable $r(i)$.

## 2.6 Feedback control

In this section we present the mathematical basis for discrete linear feedback systems. In what follows we assume that the state $x(i)$ of the system can be accurately measured at all times and is available for feedback. In what follows we consider the "regulator problem", which refers to the problem of maintaining the state of the system at a chosen value. This corresponds to our control problem, since we wish to maintain the beam parameters as constant as possible. We first consider the system defined by:

$$x(i+1) = Ax(i) + Bu(i), \tag{2.83}$$

with the controlled variable given by:

$$z(i) = Dx(i), \tag{2.84}$$

and with linear control defined by:

$$u(i) = -Fx(i). \tag{2.85}$$

This is a particular case of the general system described in Sec. 2.5.1, where the controller state corresponds to the system state but with opposite sign. The matrices $A, B$ and $D$ are those defined in Sec. 2.5.1, and $F$ is a feedback gain matrix. We note that the control is asymptotically stable if the system described by:

$$x(i+1) = (A - BF)x(i), \tag{2.86}$$

is asymptotically stable [49]. If $F = constant$, the stability is determined by the eigenvalues of $(A - BF)$. These can be placed at arbitrary points in the complex plane by designing $F$ accordingly.

To optimise the controller, we must establish how to bring the system from its initial state to its final state as quickly as possible. This can be expressed as an optimisation problem, for which we use the following quadratic minimisation criterion [49]:

$$\Gamma = \sum_{i=i_0}^{i_1-1} [z^T(i+1)R_a z(i+1) + u(i)^T R_2(i)u(i)] + x^T(i_1)P(i_1)x(i_1), \tag{2.87}$$

where $R_a(i+1) > 0$ and $R_2(i) > 0$ for $i = i_0, i_0+1, ..., i_1-1$ and $P(i_1) \geq 0$ are weighting matrices. The three terms in Eq. (2.87) have the following interpretation. The first term is a measure of the extent to which the state deviates from the zero state; the $R_a$ weighting matrix determines the respective weight of each component of the state vector. The second term in the sum is a measure of the deviation of the input from the optimal input and the matrix $R_2$ determines the respective weight of each component of the input vector. This term is important since the sum of the input deviations over time reflects how rapidity stabilisation can be achieved. Moreover, because the input must be bounded, the minimisation of this sum ensures a reduced overshoot. The third term reflects the fact that the final state $x(i_0)$ must be as close as possible to the zero state. We also note that in Eq. (2.87) the two terms in the summation are indexed differently. This is because the value of the controlled variable $z(i_0)$ is entirely dependent on the initial state $x_0$ and cannot be changed, i.e. it is a constant which does not need to be included in the sum; we will discuss this in more detail in Sec 2.7.

To minimise Eq. (2.87) let us assume that at the instant $i_0 - 1$ the input $u(i_0-1)$ is chosen arbitrarily, but at all subsequent instants $i_0, i_0+1, ...i_1-1$ the inputs are optimally chosen, such that the corresponding terms in Eq. (2.87) are minimised. The sum of these optimised terms is denoted by $\sigma_i$. The summation in Eq. (2.87) can therefore be written as the sum of the first element of the time series (instant $i_0$) and $\sigma_i$ (the sum over all other instants), i.e.

$$\Gamma = [z^T(i_0+1)R_a z(i_0+1) + u(i_0)^T R_2(i_0)u(i_0)] + \sigma_i. \tag{2.88}$$

To minimise Eq. (2.88), we must minimise the first two terms of the right hand side of the equation. Since $\sigma_i$ is already a sum of minimised terms,

only the first term needs to be minimised. It can be shown that the controller gain matrix $F$ has a unique solution [49]:

$$F(i) = \{R_2(i) + B^T(i)[R_1(i+1) + P(i+1)]B(i)\}^{-1} B^T(i)[R_1(i+1) + P(i+1)]A(i), \quad (2.89)$$

where

$$R_1(i) = D^T(i)R_a(i)D(i). \quad (2.90)$$

Here $P$ is the matrix difference equation defined by:

$$P(i) = A^T(i)[R(i+1) + P(i+1)][A(i) - B(i)F(i)]. \quad (2.91)$$

We can use Eq. (2.89) to design a controller with a gain matrix $F$ that minimises the mean square tracking error and the mean square input.

## 2.7 Controller design and PID control

Here we discuss how a controller can be designed using the z-transform, and how pole placement affects the system's stability. Consider the control system depicted in Fig. 2.3. The basic idea of such a system is to make the control output $y(i)$ follow a reference input $r(i)$. The desired transfer function for the control system can therefore be expressed in the z-domain as $T_c(z) = 1$, where:

$$T_c(z) = \frac{Y(z)}{R(z)}. \quad (2.92)$$

Here $R(z)$ is the z-transform of the reference input and $Y(z)$ is the z-transform of the system output. From the schematic in Fig. 2.3 we have the following relations:

$$E(z) = R(z) - H_s(z)Y(z), \quad (2.93)$$

and

$$Y(z) = G_s(z)E(z). \quad (2.94)$$

Utilising these conditions, Eq. (2.92) can be rewritten as:

$$T_c(z) = \frac{Y(z)}{R(z)} = \frac{G_s(z)}{1 + G_s(z)H_s(z)} = \frac{(z - z_1)...(z - z_m)}{(z - p_1)...(z - p_n)}, \quad (2.95)$$

where $m < n$. The poles of the system can be found by setting the denominator in Eq. (2.95) to zero, i.e. $G_s(z)H_s(z) + 1 = 0$. All coefficients of the polynomials $Y(z)$ and $R(z)$ are real, therefore the poles and zeros must be either purely real, or appear in complex conjugate pairs, i.e. $a \pm jb$. The unforced response of a linear system to a set of initial conditions is:

$$y(i) = \sum_{k=1}^{n} C_k e^{-i\lambda_k}, \tag{2.96}$$

where the coefficients $C_k$ are determined by the initial conditions and $\lambda_1$, $\lambda_2$,..., $\lambda_n$ are the eigenvalues of $R(z)$. According to Eq. (2.96), we can draw the following conclusions regarding the effect of the location of the poles on the stability of the system:

- A real pole $a$ in the left-half of the complex ($a < 0$) plane defines an exponential decay, whose rate is determined by the pole location. The farther from the origin, the more rapid the decay;
- A real pole $a$ in the right-half of the complex plane ($a > 0$) generates an exponentially increasing response, rendering the system unstable;
- A complex pair of poles $\pm jb$ on the imaginary axis generates an oscillatory response, whose constant amplitude is determined by the initial conditions;
- A complex conjugate pair of poles $a \pm jb$ in the left-half of the complex plane ($a < 0$) combine to generate a response that is a decaying sinusoid of the form $Ae^{-ai}sin(bi + \phi)$ and whose decay rate and frequency of oscillation are determined by $a$ and $b$, respectively. The amplitude $A$ and the phase $\phi$ are determined by the initial conditions;
- A complex pair of poles $a \pm jb$ in the right-half of the complex plane ($a > 0$) generates an exponentially increasing response.

These outcomes are illustrated in Fig. 2.4. In summary, the system is stable if the poles lie in the left-half of the z-plane. In an unstable system with a bounded input, the output becomes unbounded when the poles lie in the right-half of the z-plane.

The stability of the feedback system depicted in Fig. 2.3 can be improved with the addition of a controller $G_c(z)$, as shown in Fig. 2.5. By adding this controller, it is thus possible to modify the system's response by ensuring that all the poles are in the left-half of the z-plane.

With the addition of the controller $G_c$, the z-transform of the system output is $Y(z) = G_c(z)G_s(z)U(z)$. The corresponding transfer function of the system can be written as [51]:

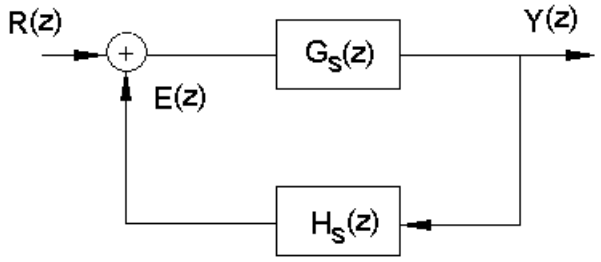



Figure 2.3: Schematic of a feedback control loop. $G_s(z)$ is the transfer matrix of the system and $H_s(z)$ is the feedback gain matrix.

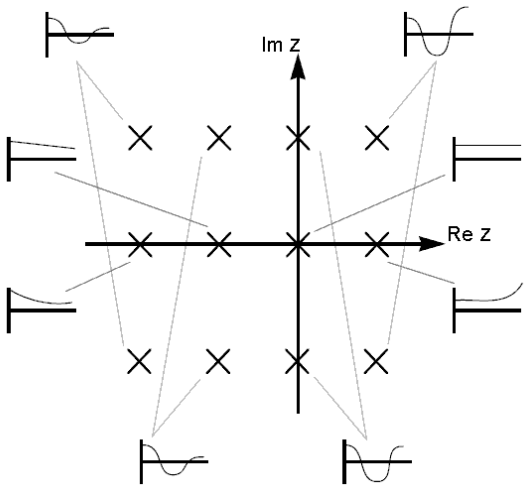



Figure 2.4: Pole placement and corresponding system response. To be stable the poles must lie in the left-half of the complex plan. When a pole is placed on the imaginary axis this corresponds to critical stability, where the perturbation is not damped or amplified. Figure adapted from [51].

$$T_c(z) = \frac{Y(z)}{R(z)} = \frac{G_c(z)G_s(z)}{1 + G_c(z)G_s(z)H_s(z)}. \tag{2.97}$$

Now consider Eq. (2.97) when $H_s(z) = constant$. One can see that increasing the gain of the controller $G_c(z)$, such that $G_c(z)G_s(z) \gg 1$, is enough to ensure that $T_c(z) \cong 1$. If the controller block $G_c(z)$ is independent of z (i.e. is a constant gain), the controller is called a proportional or "P" controller. By increasing the gain of the P controller, the rise time of the system can be decreased, allowing the output to follow the input more quickly. This method however, has a major drawback. When the gain is increased, the overshoot of the output is increased causing a damped oscillatory output. When the gain is increased further, the system reaches a critical point where the os-

cillation is not damped or amplified and the system is said to be critically stable. This corresponds to the poles of the closed-loop transfer function lying on the imaginary axis of the z-plane as shown in Fig. 2.4. For $Re(z) > 0$ the system is unstable. Another limitation of the "P" controller is that a constant gain tends to amplify high-frequency noise [51]. For these reasons, the proportional gain is often complemented with an integral and/or differential gain.

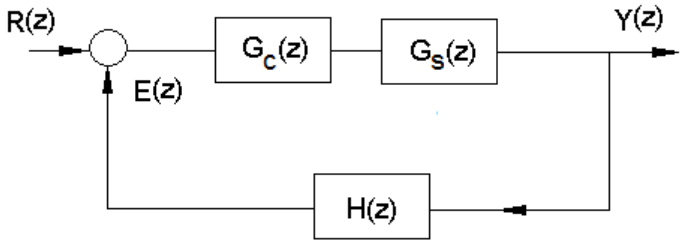



Figure 2.5: Schematic of a modified feedback control loop. The system is described by the transfer matrix $G_s(z)$, whilst the controller is described by $G_c(z)$.

To reduce the rise time without increasing the overshoot, it is possible to add a derivative term to the proportional controller. The transfer function of the controller can then be written as a sum of the proportional and derivative terms, with gains $P_g$ and $D_g$, respectively, i.e.

$$G_c(z) = P_g + D_g z. \tag{2.98}$$

When $D_g$ is properly tuned, the overshoot is avoided. In addition, it is also important to reduce the steady state error. For this we can use an integral term "I", with gain $I_g$. This term acts as a "memory", whence Eq. (2.98) becomes:

$$G_c(z) = P_g + \frac{I_g}{z} + D_g z = \frac{D_g z^2 + P_g z + I_g}{z}. \tag{2.99}$$

The integral term allows the PID controller to have a non-zero output for a zero input and ensures that the steady state error can be reduced to zero. This allows us to eliminate offsets. However, the addition of this term has two drawbacks. First, the integrator introduces another pole, which might reduce the stability of the system. Second, this term acts as a low-pass filter and reduces the transient response of the system. Because of this, and because the derivative term amplifies high-frequency noise, two or more poles must be added to ameliorate the response at higher frequencies. Ad-

ditional zeros may also be needed to obtain a specific frequency response for the controller[6].

## 2.8 Feedforward compensation scheme

In feedback control a correction signal is applied based on the measurement of differences between the desired outputs (set points) and the actual outputs. Consequently, it is not possible for a feedback controller to entirely suppress a perturbation and the residual deviation will increase with frequency. This is illustrated in Fig. 2.6, where two sinusoidal perturbations with the same amplitude are plotted; one with a 2 Hz frequency and a second with 4 Hz frequency. At step $k$, the two perturbations have zero amplitude. At step $k+1$ however, the deviation is larger for the 4 Hz perturbation. As the signal frequency is higher, the perturbation has "more time" to oscillate, which corresponds to the controller having "less time" to make adjustments.

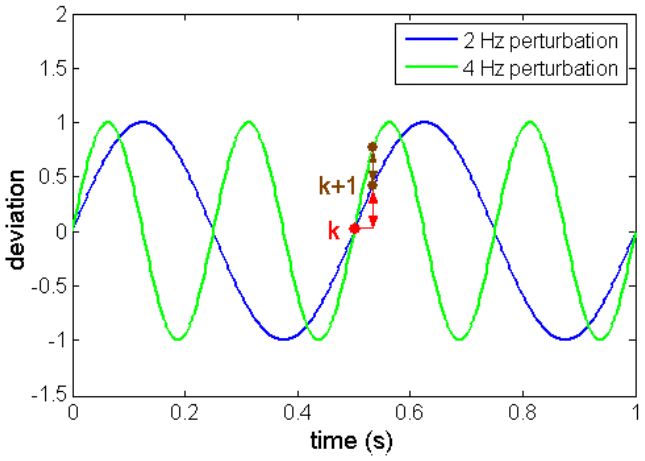



Figure 2.6: Perturbations with the same amplitude but different frequencies: 2 Hz (in blue) and 4 Hz (in green) are plotted over 1 second.

To eliminate higher frequency perturbations, control can be performed in feedforward. In feedforward control, changes are detected at the input and an anticipating corrective signal is applied before the output is affected. The success of a feedforward system will depend on its ability to measure

[6]The frequency response of the system is obtained by evaluating $T(z)$ for $z = e^{i\theta}$. From Eq. (2.95), the magnitude of the frequency response is governed by the ratio of two polynomials involving zeros and poles in the z-plane, with $\theta$ going from 0 to $\pi$. It is thus possible to shape the frequency response by placing poles at distances that will produce the desired response.

and predict the perturbation, and on accurate knowledge of the effect of the perturbation on the system [48, 54]. A feedforward compensation scheme is shown in Fig. 2.7, where some of the command signal is injected into the control signal without being affected by feedback. It can provide a way to speed up a system's transient response without impacting on its stability.

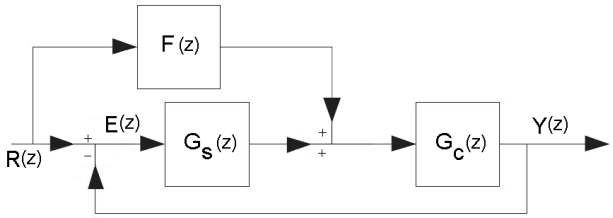



Figure 2.7: Feedforward control scheme with transfer matrix $F(z)$. Feedforward provides an extra input to the system, in addition to the feedback controller output.

The system transfer function for the feedforward compensation scheme shown in Fig. 2.7 is:

$$T_c = \frac{(F + G_c)G_s}{1 + G_c G_s}. \tag{2.100}$$

Stability is determined by the poles of $T_c$, as discussed in Sec. 2.7. Conventional feedforward techniques consist of designing filters for $F(z)$ to reject specific frequencies or frequency bands. Filters are categorised as either finite impulse response (FIR) filters or infinite impulse response (IIR) filters. The response of a FIR filter is said to be finite because its response settles to zero in a finite number of samples; unlike an IIR filter, which has an internal feedback. A FIR filter is described at instant $k$ by the following difference equation [55]:

$$y(k) = \sum_{i=0}^{N} b_i x(k-i), \tag{2.101}$$

where $x(k)$ is the input signal at instant $k$ and $y(k)$ is the output. The $b_i$ ($i = 0, 1, ..., N$) are the filter coefficients (also called the tap weights) and $N$ is the order of the filter. Consider the response of the FIR filter to a (Kronecker) delta impulse $x(k) = \delta(k)$. Equation (2.101) becomes:

$$y(k) = \sum_{i=0}^{N} b_i \delta(k-i) = b_k. \tag{2.102}$$

The corresponding z-transform of the FIR filter is given by:

$$T_c(z) = \sum_{k=-\infty}^{\infty} y(k)z^{-k} = \sum_{k=0}^{N} b_k z^{-k}. \tag{2.103}$$

In contrast, the output of an IIR filter is described at instant $k$ by the following difference equation [55]:

$$y(k) = \sum_{i=0}^{N} b_i x(k-i) - \sum_{j=0}^{M} a_j y(k-j), \tag{2.104}$$

where $a_j$ $(j = 0, 1, ..., M)$ are the feedback coefficients of the IIR filter and $M$ is the order of the filter. Unlike the FIR filter, the output of the IIR filter incorporates $M$ past outputs. The corresponding transfer function of the filter is given by [55]:

$$T_c(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{N} b_i z^{-i}}{1 + \sum_{j=1}^{M} a_j z^{-j}}. \tag{2.105}$$

The stability of the IIR filter is determined by the location of its poles, as discussed previously in the context of feedback controllers. Filter design requires coefficients to be selected to ensure that the system has specific characteristics; in particular the coefficients determine the frequency response of the filter. There are different methods to find the coefficients corresponding to specific frequency requirements;. These include the Least Mean Square (LMS) [56], Filtered-x Least Mean Square (FXLMS) [57], or the Augmented Error Algorithm [55]. In general FIR filters are simpler to implement as they only require knowledge of the present and past values of the input and are inherently stable. The disadvantage of FIR filters is that because of the lack of poles it is difficult to model a complicated transfer function, and a large number of taps may be required to model a dynamic system. The disadvantage of an IIR is that the presence of poles can make the system unstable during the adjustment of the coefficients. An algorithm is then required to monitor the stability of the filter during its optimisation and this process can be computationally expensive.

Combinations of feedback - feedforward control are also commonly used and have been applied successfully in many areas, such as in interferometry [58] and in automotive control systems [59]. They have the advantage of combining both the stability of feedback control and the ability of the feedforward to take corrective actions before the perturbation affects the system output - thus enabling the correction of higher frequency perturbations. The feedback loop eliminates offsets due to inaccuracies in the feedforward compensation and handles residual errors. Moreover, feedback systems are best adapted to those systems that are subject to unexpected disturbances.

For the feedforward method to be effective it is important that the effect of an actuator on the variable to control is well known.

## 2.9 Discussion

In this chapter we discussed the fundamental principles used for the control of linear time-shift invariant systems. In Sec. 2.2 we introduced important definitions related to the response of a control system, including a formal definition of stability. The control problem was divided into two parts - the transient response and steady state response of the system (see Sec. 2.2.2). The system was then formally described in Sec. 2.5.1, which allowed us to derive the conditions necessary to design a controller, in particular those conditions that ensure stability. The notion of the z-transform was also introduced in Sec. 2.3 and used to describe the transfer function of the system and its controller (see Sec. 2.7). The z-transform allows us to categorise the stability of the system in terms of the location of the poles in the z-plane. The PID algorithm was also discussed, which will be exploited in Chapter 5. Finally, we introduced the concept of feedforward control, which will be used in Chapters 7 and 8.

# 3 The machines

## 3.1 General considerations

This chapter describes the three machines that were used for simulation and experimental work in the thesis; namely, the Australian Synchrotron (AS) Linac, the Linac Coherent Light Source (LCLS) Linac, and the FERMI Linac. Our intention is to provide a concise overview of the main features of the accelerators. The reader is referred to [2, 60, 61] for a detailed technical account of these machines. The diagnostics used for experiments are discussed, including transmission measurements (fast current and wall current monitors), energy measurements (using screens and beam position monitors) and bunch length measurements (using coherent synchrotron radiation detectors). Information on other diagnostic components can be found in the corresponding design reports, i.e. reference [60] for the Australian Synchrotron, [61] for the LCLS and [2] for the FERMI@Elettra.

## 3.2 The Australian Synchrotron Linac

### 3.2.1 Electron source

The electrons are generated in a 90 kV DC electron gun with a thermionic cathode. A grid is used to modulate electron pulses when the system is triggered. The electron gun can operate either in short pulse mode (SPM)[1] or in long pulse mode (LPM). In the first case the pulse duration will be shorter than 1 ns, filling only one single 500 MHz bucket in the booster. Using this mode the Linac will supply the booster with a 1ns pulse at a repetition rate of 1 Hz.

[1] In the short pulse mode, also referred to as single bunch mode, a single bunch is defined as a train of no more than three S-band micro bunches that are to be injected into a single 500 MHz RF bucket of the booster. In multi-bunch operation the trains consist of single bunches.

In the second case, the LPM will deliver a beam at 500 MHz, with a pulse duration of 150 ns and at a repetition rate of 1 Hz to 5 Hz. The total charge in a train is ~3.1 nC. The maximum length of the trains does not exceed 1 $\mu$s, while the number of bunches and the inter-bunch distance are variable.

### 3.2.2 Pre-bunching and bunching systems

A sub-harmonic, pre-bunching system (SPB) working at 500 MHz is used to ensure good single bunch purity[2] in the booster. It also serves the purpose of increasing the modulation depth from the gun, i.e. of compressing the bunches. The pre-buncher consists of an RF accelerating cavity followed by a drift space. As a stream of particles passes through the pre-buncher, rear particles get accelerated and front particles are decelerated depending on the phase of the electric field in the pre-buncher at the time of passage. Concentrating particles by utilising the optimal phase maximises the particle density in the bunch. When the particles pass through the drift space bunching of the particle distribution occurs, which reaches a maximum at some distance $L$, corresponding to the chosen length of the drift space.

The 3 GHz primary buncher (PBU) increases the mean energy of the beam from 90 keV to approximately 300 keV and bunches the beam into a 3 GHz structure. It is followed by the 3 GHz final buncher, which provides the final bunching and increases the beam energy from about 300 keV to above 3 MeV. Figure 3.1 shows the different elements of the Australian Synchrotron Linac.

### 3.2.3 Accelerating sections

Two identical 3 GHz structures (which will be referred to as ACC1 and ACC2) are used in order to increase the beam energy up to 100 MeV and to ensure that the system meets reliability requirements. The first accelerating section (ACC1) brings the beam energy up to 50 MeV, while the second section increases it to 100 MeV. The phase of the second section is usually employed for adjusting the beam energy.

### 3.2.4 Focusing magnets

Solenoids and quadrupoles are used to ensure that most of the beam is kept in a radius of less than 0.5 cm from the beam axis. Thirty one solenoids are

[2]The bunch purity is defined as the percentage of particles in the central micro-bunch. Because the gun emits electrons at 500 MHz and the bunching sections operate at 3 GHz, a longitudinal modulation of the particle distribution occurs within the bunch. As a result of the modulation, the bunch consists of a main central part (the central microbunch) and adjacent microbunches or buckets [62].

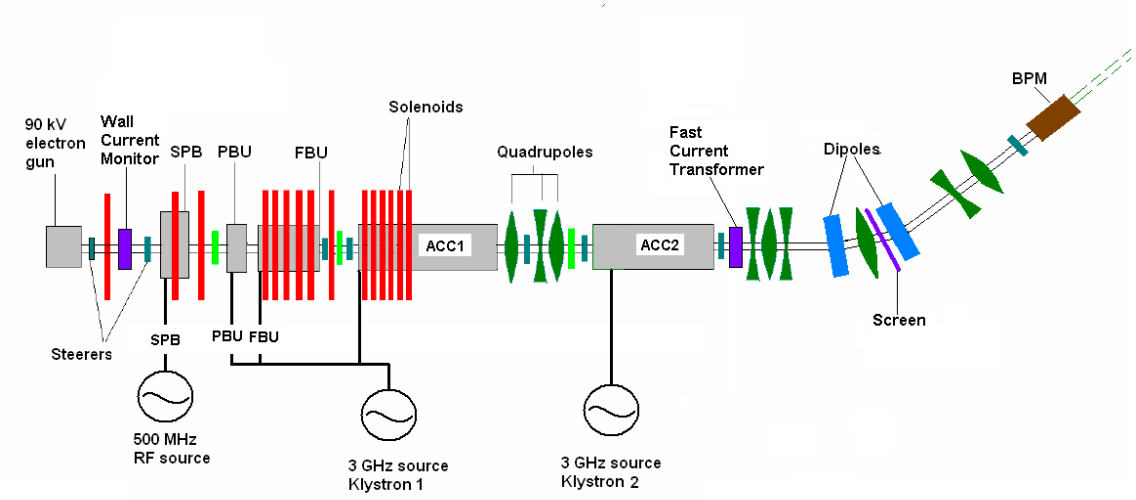



Figure 3.1: Layout of the Australian Synchrotron Linac. SPB, PBU, FBU provide the bunching, while ACC1 and ACC2 accelerate the beam to 100 MeV. Figure adapted from [63].

used in the low energy region (below 10 MeV); these are located between the gun and the first accelerating section. Quadrupoles are used in the high energy region (above 10 MeV) as they are the most efficient for beam focusing. Three quadrupoles form a focusing triplet where the beam reaches 50 MeV along the accelerator, between ACC1 and ACC2. This provides full transmission through ACC2; it also ensures that the required transverse beam parameters are met at the exit and provides the transfer line with the capacity to shape the beam for matching into the booster. Another quadrupole triplet is located directly after ACC2.

Two dipoles are located in the bend of the Linac-to-Booster Transfer Line, which adjust the beam trajectory. A quadrupole is placed in the middle to provide an achromatic bend, preventing the beam position from moving off axis due to energy fluctuations.

### 3.2.5 Diagnostic components

A removable Faraday Cup (FC) is located just after the electron source to allow the charge and pulse structure to be measured with a fast oscilloscope. The FC consists of an electrically isolated copper cup, which is connected to ground via a resistor. The induced voltage across the resistor, due to the passage of the beam, is a measure of the electron beam current. The integrated current is used to determine the number of particles [60].

A wall current monitor (WCM) is installed immediately after the electron source, which allows for a non-destructive measurement of the beam charge. When a bunch of electrons passes through the conducting beam pipe, image charges are created on the latter to cancel the field created by the electrons in the bunch (see Sec. 4.2.3). The WCM measures the resulting image current induced in the vacuum beam pipe [64]. A ceramic gap of a few mm is introduced into the beam pipe, which is bridged by a number of resistors. The wall current produces a voltage across these resistors. As with the Faraday cup, the WCM is suitable for time resolved measurements. It is incorporated into a stainless steel housing containing a ferrite ring to reduce excitation of the eigenmodes of the housing.

Four removable screens and cameras have been installed; one after the electron source, a second between the SPB and the bunching section and one each in front of the accelerating structures [60] (see Fig. 3.1). The screens are installed for a destructive measurement of the electron beam position and its transverse profile. The fluorescent screens convert the spatial distribution of the beam flux into visible radiation. The screens are viewed through a zoom lens with a CCD camera, whose spatial resolution is 30-$\mu$m. The CCD camera can be triggered externally and is able to digitise, store, and display a particular frame in real-time. The decay time of the fluorescent screen is of the order of milliseconds, which is important for injectors with a low repetition rate, since the image on a TV monitor can be "frozen" until the next shot arrives [64].

A fast current transformer (FCT) is located at the end of the Linac (as shown in Fig. 3.1), just after the second accelerating section. This allows non-destructive measurements of transmission[3] in combination with the WCM. The FCT consists of a main single turn toroid through which the beam passes and a secondary coil (with a variable number of turns). When the beam passes through the main core it produces a magnetic flux, which then induces a current in the secondary coil. The transformer yields a signal amplitude that is nominally proportional to the beam current and inversely proportional to the number of turns of the secondary coil, i.e.

$$V \propto \frac{I_{beam}}{N_{turns}}. \tag{3.1}$$

The Linac is equipped with a stripline beam position monitor (BPM), provided by Sincrotrone Trieste (from a former transport line) to allow relative energy measurements; this will be discused in Chapter 7. The beam position monitor consists of four stripline electrodes attached to either side

[3] In this work the transmission of the beam is defined as the percentage of particles that are still contained in the electron bunch after a given section of the accelerator.

of the beam tube. When the electron beam passes through the BPM, an image charge that uniquely depends on the position of the beam is induced on each of the four electrodes. The position $P(x, y)$ of the electron beam can then be determined knowing the voltage induced at the electrodes $V(e_1, e_2, e_3, e_4)$, where $e_i$ $(i = 1, 2, 3, 4)$ denotes the $i^{th}$ electrode (see [65] for more details). In order to make relative energy measurements possible, the quadrupole current in the achromatic bend is tuned to produce dispersion at the BPM. The spatial resolution of the BPM is of the order of 50 $\mu$m.

## 3.3 The Linac Coherent Light Source

### 3.3.1 The photoinjector and Linac sections

Figure 3.2 shows the various components of the Linac Coherent Light Source (LCLS). The electrons are generated with a RF photocathode driven gun. The system operates in pulse mode, generating 6 MeV electron bunches at repetition rates of 10 Hz, 30 Hz or 120 Hz. Two linear accelerating structures each 3 metres in length (L0-A and L0-B, which are referred to collectively as Linac L0) bring the beam energy up to 135 MeV.

In order to Landau damp the micro-bunching instability[4], a laser heater (LH) is installed before the first bunch compressor, where the beam reaches 135 MeV (immediately after the L0 Linac). It consists of a 0.5 metre long undulator located within a chicane, which allows an external IR laser to seed the electron beam. This provides a controlled increase of the uncorrelated energy spread to 40 keV rms [66]. When particles interact with the laser they gain an energy modulation, i.e. the energy of a particle in the electron bunch is a function of its position in the bunch. However, the corresponding particle density modulation is negligible. Because the induced energy/position correlation is "smeared" by the transverse motion in the chicane, the laser/electron interaction leads to an effective heating of the beam. More details on this process can be found in [67].

The three main Linac sections L1, L2 and L3 accelerate the electrons to a maximum of 14.3 GeV, before being injected into the undulator line. Figure 3.2 shows the LCLS layout and the energy reached at each stage of the machine. The working points for the voltage and phase of the accelerator used for simulations in Chapter 5 are summarised in Table 3.1.

[4]Micro-bunching instabilities occur during the bunch compression and are driven by coherent synchrotron radiation (CSR) and longitudinal space charge (LSC). These instabilities can significantly degrade the beam quality [66].

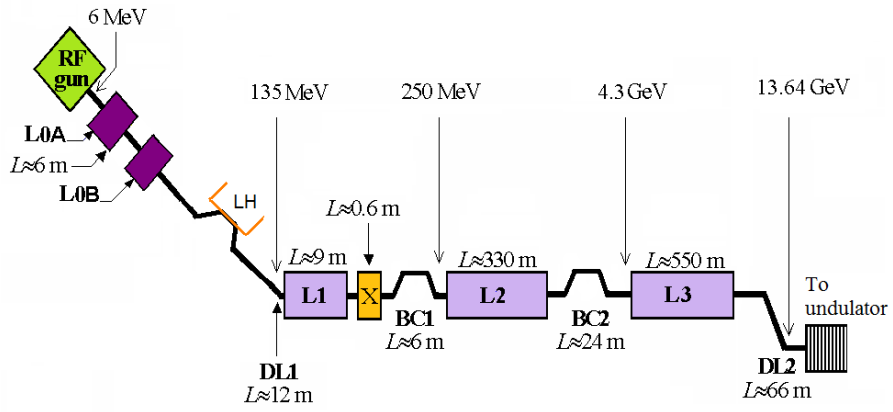



Figure 3.2: Layout of the LCLS linear accelerator. The setup consists of four Linac sections, L0, L1, L2, L3, two bunch compressors, BC1 and BC2, and an X-band lineariser X. It has two dispersion regions corresponding to the dog legs denoted by DL1 and DL2.

Table 3.1: Voltage and phase set points for the LCLS Linac.

| Element | Setting |
|---|---|
| Linac-1 :voltage ($V_1$) | 144 MeV |
| Linac-1 :phase ($\phi_1$) | 337.7° |
| X-band :voltage ($V_x$) | 20 MeV |
| X-band :phase ($\phi_x$) | 200° |
| Linac-2 :voltage ($V_2$) | 4.3 GeV |
| Linac-2 :phase ($\phi_2$) | 324° |
| Linac-3 :voltage ($V_3$) | 60 MeV |
| Linac-3 :phase ($\phi_3$) | 0° |

Two solenoids are placed after the electron gun and at the start of the L0A section for focusing. A quadrupole triplet is also placed before the first dog leg (DL1), where space charge is important [68].

### 3.3.2 X-band lineariser

The RF acceleration and wake fields create an energy chirp of the electron beam. In the energy/position representation of the beam this has the form of a "banana" when the quadratic order term dominates and the form of a "S" shape when the cubic term dominates (see Fig. 3.3). The energy $\delta(z)$ of a particle in the bunch is a function of the position $z$ of this particle within the bunch, where $z = 0$ corresponds to the front of the bunch.

The longitudinal wake fields, in combination with off-crest acceleration, are used to cancel the linear energy chirp. The quadratic energy chirp,

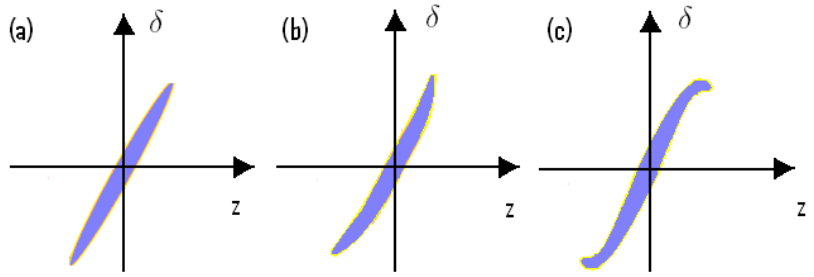



Figure 3.3: Representation of the different energy chirps induced by RF curvature. (a) Linear chirp only. (b) Positive linear and quadratic energy chirp ($d\delta/dz > 0$ and $d^2\delta/d^2z > 0$). (c) Positive linear and negative cubic energy chirp ($d\delta/dz > 0$ and $d^3\delta/d^3z < 0$).

which increases the energy spread, can in principle be canceled by the use of an X-band cavity. This shorter 0.6 m structure works at the fourth harmonic (11.424 GHz) of the S-band frequency (2.856 GHz). It can be shown that there is a solution (in terms of the X-band phase and voltage) for which the second and third order terms cancel [69].

Undesirable effects, resulting from the cubic term, produce a decrease in compression efficiency, possible deterioration of transverse emittance and energy distribution, and additional excitation of the wake field by the edges. When the beam is compressed, a negative cubic energy chirp will lead to a "two horn" density spectrum, whereas a positive cubic term will produce a "one horn" structure (see Fig. 3.4), as the edges are over compressed and create current spikes [70].

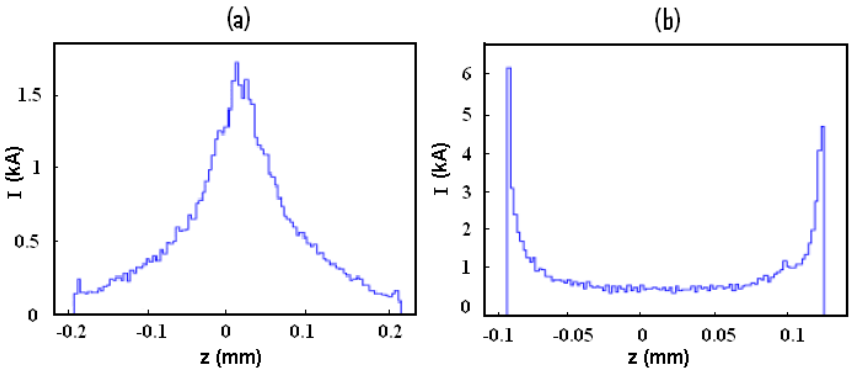



Figure 3.4: Undesirable effect due to remaining cubic chirp [71]. (a) "One horn" current profile after compression of a beam leading to a positive cubic energy chirp ($d^3\delta/d^3z > 0$). (b) "Two horn" current profile after compression of a beam produces a negative cubic energy chirp ($d^3\delta/d^3z < 0$). The spikes are due to over compression of the edges .

An analytical solution exists that allows us to correct for both the quadratic and the cubic terms [61]. However, this solution corresponds to a significant voltage for the X-band cavity, which results in a loss of beam energy, since the X-band decelerates the beam to compensate for the linear chirp. Therefore the X-band can only be used to compensate for the cubic chirp. The LCLS X-band lineariser operates at 180° and decelerates the beam by 22 MeV, setting the BC1 compression energy at 250 MeV.

### 3.3.3 Bunch compressors

The bunch has to be compressed by a factor of about 50, thereby reducing the initial bunch length from about 1 mm after the RF gun to 22 $\mu$m after the second bunch compressor (BC2), where it enters the undulator [61]. As depicted in Fig. 3.2, the LCLS accelerator has two bunch compressors; namely, BC1 and BC2. Each of these two compressors consists of four dipoles of equal length as illustrated in Fig. 3.5.

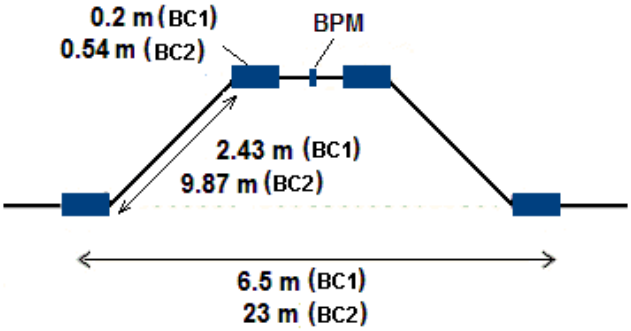



Figure 3.5: Structure of the four bending magnet bunch compressor, which constitutes BC1 and BC2. The characteristic lengths are given for both compressors (BC1 and BC2).

The energies at the compressors (0.25 GeV at BC1 and 4.3 GeV at BC2) have been chosen high enough to avoid space charge effects, but low enough to minimise wake field effects [61].

### 3.3.4 Diagnostic components

Energy variations are determined with stripline BPMs (with a 5-10 $\mu m$ spatial resolution) placed at high dispersion points in each of the following bending regions: DL1, BC1, BC2 and DL2. At DL1, a combination of three BPMs is used to distinguish energy variations from betatron oscillations [61]. For BC1 and BC2, a single BPM is placed at the centre of the chicane. Finally,

the DL2 energy measurement utilises two BPMs to cancel any betatron oscillation component from the signal.

To adjust the compression it is necessary to measure and control the electron bunch length after BC1 and BC2. This is realised by using non invasive relative measurements of the bunch length. To accomplish this we use the coherent synchrotron radiation (CSR) emitted by electrons passing through a chicane. Typically, only wavelengths larger than the beam size are coherently enhanced, i.e. $\lambda > 2\pi\sigma_z$, where $\sigma_z$ is the rms bunch length. Two types of detectors are used, depending on the bunch length to be measured, i.e. RF diodes and pyrodetectors. Pyrodectectors are used for small bunches, while diodes are used for longer bunches due to their higher sensitivity in the sub-THz range.

At wavelengths comparable to the bunch length, the radiated power in the pyrodetector will vary as a function of the bunch length. Filters are used to select wavelengths around the expected bunch wavelength ($\lambda \pm \Delta\lambda$); this avoids high frequency components (above 4 THz) arising from the double horn structure as a consequence of compression. In this way the radiated power measured by the detector does not depend on the temporal structure of the bunch. Because the light needs to be transported from the beam pipe to the detector (using a diaphragm, a window and filters), it is difficult to evaluate the signal loss. Therefore the CSR signal is not calibrated, but nevertheless it can be used to maintain a given value of the peak current using phase and voltage actuators [72].

After BC1 the wavelength corresponding to the bunch length is around 1 mm (300 GHz), while after BC2 it is about 100 $\mu$m (3 THz). The two types of relative bunch length monitors are installed after BC1; namely, a ceramic gap with waveguide coupled diodes and a pyroelectric detector [73]. The ceramic gap is simple to build and provides a good bunch length signal, but is not suitable for the higher frequencies required for BC2. The second bunch compressor uses a system similar to the pyroelectric detector for BC1. The pyroelectric detector includes millimetre wave filters to control the signal bandwidth.

## 3.4 The FERMI@Elettra Free Electron Laser Linac

### 3.4.1 The photoinjector and Linac sections

The FERMI photocathode radio-frequency (RF) electron gun uses a metallic photocathode illuminated by an intense 263 nm wavelength UV laser. A first stage gun is designed for a pulse repetition rate of up to 10 Hz, whereas an upgraded version will reach 50 Hz. The first stage gun is designed to provide a peak accelerating gradient of 110 MV/m, and an output beam energy

of about 5 MeV at 10 Hz. Figure 3.6 shows the layout of the machine. External emittance compensating solenoid magnets provide focusing to help transport the beam from the gun to the entrance of the main Linac structures. Part of the injection system into the main accelerator requires two short Linac sections L0A and L0B, which follow the gun and bring the beam energy up to 100 MeV.

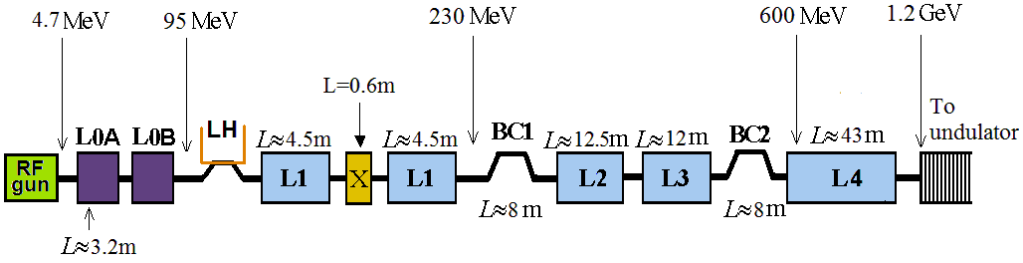



Figure 3.6: Layout of the FERMI linear accelerator. The accelerator includes four main Linac sections, an X-band lineariser located in the middle of Linac 1, and two bunch compressors (BC1 and BC2). The L1 and L2 Linacs were obtained from CERN, whereas the L3 and L4 sections were specifically designed for FERMI.

The main accelerator structure includes four Linac sections. The first two Linacs are composed of four and three sections, respectively. These were obtained from CERN after the decommissioning of the LEP Injector Linac. Linac 3 and Linac 4 are new elements built specifically for the FERMI project. A bend region, at the end of the accelerator (known as the spreader) allows for energy measurement and feedback. The voltage settings of the Linacs were chosen to achieve the desired compression and cancelation of the final energy chirp. Table 3.2 gives the working points for the voltage and phase of the Linacs. These settings will be used for our simulations in Chapter 5. Like the LCLS, the FERMI design includes a laser heater installed at the section where the beam energy is 100 MeV (before the first main Linac L1); this provides an increase in the uncorrelated energy spread of the order of 10 keV rms to 20 keV rms.

For FERMI, the remaining cubic chirp (described in Sec. 3.3.2) has a significant impact on the final current profile [74]. For this reason, the current distribution of the beam at the exit of the photoinjector is shaped, such that the cubic term is compensated for by the beam itself. This requires a ramp current distribution as shown in Fig. 3.7, which was evaluated using reverse tracking[5].

[5]Reverse tracking consists of tracking the desired final beam shape from the end to the start of the machine [75].

Table 3.2: Voltage and phase set points for the FERMI Linacs [2].

| Element | Setting |
|---|---|
| Linac-1 :voltage ($V_1$) | 188 MV |
| Linac-1 :phase ($\phi_1$) | 54° |
| X-band :voltage ($V_x$) | 18 MV |
| X-band :phase ($\phi_x$) | -90° |
| Linac-2 :voltage ($V_2$) | 141 MV |
| Linac-2 :phase ($\phi_2$) | 70° |
| Linac-3 :voltage ($V_3$) | 240 MV |
| Linac-3 :phase ($\phi_3$) | 70° |
| Linac-4 :voltage ($V_4$) | 600 MV |
| Linac-4 :phase ($\phi_4$) | 109° |

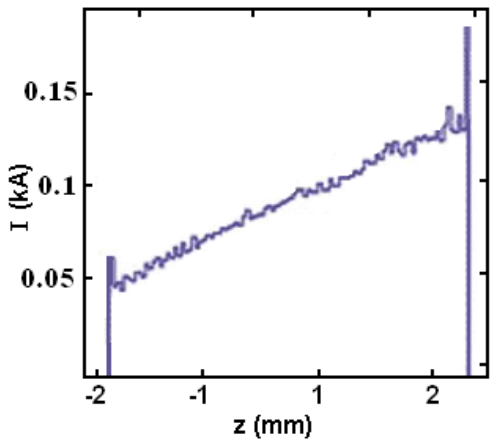



Figure 3.7: Ramped current distribution at the end of the FERMI photoinjector, which leads to the flat peak current distribution (in Fig. 3.4 (b)) at the end of the accelerator (case of medium length bunch).

### 3.4.2 Bunch compressors

As with the LCLS, the FERMI layout initially required two bunch compressors (BC1 and BC2), in order to achieve the desired compression factor. However, its design was revised for a lower compression factor and therefore only required the first bunch compressor (BC1). The installation of the second chicane (BC2) is foreseen for the study of micro bunching instabilities. This second chicane will therefore be movable. Both chicanes are 8 m long and comprises four 0.35 m long dipole magnets. The distance between the first two and last bending magnets is 2.5 m.

### 3.4.3 Diagnostic components

Dispersive beamlines (also known as spectrometer lines) are foreseen for the measurement of the beam charge, energy, energy spread and longitudinal phase space parameters. The two spectrometer lines that are used for experiments in this thesis (see Chapter 8) are located after the laser heater and the first bunch compressor. Typically, a spectrometer line consists of a bending magnet to extract the beam to the beamline, BPMs, screens, and a Faraday Cup placed at the end of the line for charge measurement. The rms energy spread is expected to be in the range 100 keV - 300 keV. The design goal for the spectrometers is to measure the energy with 1% resolution [76].

The diagnostics used for the FERMI longitudinal beam-based control (including the energy and the bunch length) are essentially of the same type as those described for the LCLS (see Sec. 3.3.4). Stripline beam position monitors are used to measure the beam energy at the bunch compressors and the end of the accelerator, with a spatial resolution of 5-10 $\mu$m. The energy measurement is performed using the BPM placed in the center of the bunch compressor, as this is a dispersion region. In the spreader region, where the electron beam is deviated towards one of the two FEL undulator chains, the energy is measured by combining the readings of two BPMs located in positions with high absolute dispersion (0.1 m) and opposite signs. Scintillation screens (with a spatial resolution better than 10 $\mu$m) are also installed to measure the transverse profile of the electron beam at low energy in the photoinjector.

The peak current is measured using a ceramic gap with RF diodes and a CSR power detector as described for the LCLS [77]. Assuming that the bunch charge is almost constant, the peak current is controlled by measuring the bunch length. As for the LCLS, CSR monitors are used at the end of each bunch compressor, providing a relative accuracy of about 5% in the measurement of the peak current. The central wavelength of the pyrodetector is ~470 $\mu$m (~1.5 THz) at BC1 [78].

# 4 Longitudinal dynamics in Linacs

## 4.1 General considerations

As discussed in Chapter 1, the electron beam parameters that require control are the beam energy and the bunch length. In order to perform simulation studies to evaluate control strategies, we first need to identify the potential sources of perturbation in these parameters and model the dynamics of the acceleration process. Section 4.2 introduces the three main physical effects that need to be taken into account in the computational modeling of longitudinal dynamics. These include compression, RF curvature and wake field effects. Based on these results we derive expressions for the energy and bunch length deviations of the beam in the presence of perturbations (see Sec. 4.3). These equations will be calculated using Matlab in Chapter 5, where we perform energy and bunch length control studies. In the present chapter we assume a uniform particle distribution. More realistic particle and wake field distributions require the use of tracking codes such as LiTrack [1] or Elegant[2], as discussed in Chapter 5.

## 4.2 The key physical components

### 4.2.1 Compression

The electron beam can be represented in terms of the 6D phase space vector $\overrightarrow{X} = (x, x', y, y', z, \delta)$, where $x, y, z$ are the spatial coordinates, $x'$ and $y'$ are the angular coordinates, and $\delta = (p_z - p_{z,0})/p_{z,0}$ is the error in the momentum $p_z$

[1] LiTrack is a fast 2D code used to study longitudinal dynamics without considering transverse focusing. It is a Matlab-based code that employs a Graphical User Interface (GUI). RF acceleration and bunch compression are calculated up to $3^{rd}$ order [79] in the phase space coordinates (see Eq.(4.1)).

[2] Elegant is a C-based 3D particle tracking code used for linear accelerator simulations. The beam line elements are described by transport elements up to $3^{rd}$ order [80] in the phase space coordinates (see Eq.(4.1)).

relative to the design momentum $p_{z,0}$ (i.e. the particle momentum without deviations). By convention $z = 0$ corresponds to the head of the bunch and the longitudinal position of any particle within the bunch has $z < 0$. When passing through an accelerator element, the transformation from the initial coordinates (start of the accelerator element) to final coordinates (at the exit of the element) $X_0 \rightarrow X_f$, can be represented to arbitrary order by a power series expansion of the phase space coordinates [81]:

$$(X_f)_i = R_{ij}(X_0)_j + T_{ijk}(X_0)_j(X_0)_k + U_{ijkl}(X_0)_j(X_0)_k(X_0)_l + ..., \tag{4.1}$$

where $R_{ij}$, $T_{ijk}$ and $U_{ijkl}$ ($j,k,l = 1,...,6$) are the transport matrices of first, second and third order, respectively. The elements of these matrices are the coefficients of the Taylor expansion of the dynamical equations transforming initial coordinates into final coordinates for a specific accelerator element (e.g., bending magnet, quadrupole, etc). The index $i$ ($i = 1,2,...6$) refers to the element of the final phase space vector $\overrightarrow{X}_f$, approximated by the Taylor expansion around the initial phase space vector $\overrightarrow{X}_0$. For example, the element $R_{12}$ is the coefficient by a Taylor expansion of $x$ around $x'_0$ (first order). Similarly, the element $T_{134}$ is the coefficient for the Taylor expansion of $x$ around $x'_0$ and $y'_0$ (second order). The expansion assumes small deviations in the coordinates of any particle from the optimum design particle (i.e. the particle with the ideal phase space coordinates). The matrices **U** and **T** are therefore small corrections to the linear transport described by the matrix **R**.

Since we are concerned with the transformation of the longitudinal phase space of the beam, we only consider the phase space coordinate $z$ (i.e. $i = 5$). The dominant terms in Eq. (4.1) are related to the longitudinal dispersion, which are dependent on the momentum error $\delta$ ($i = 6$). Therefore we can approximate the expression for $z$ up to the third order terms as [81]:

$$z = z_0 + \delta(R_{56} + T_{566}\delta + U_{5666}\delta^2), \tag{4.2}$$

where $z_0$ and $z$ are the internal coordinates of the particle before and after compression, respectively. The coefficient $R_{56}$ is also called the "compression factor" and is given by $R_{56} = \alpha_{comp}L$, where $L$ is the length of the compressor and $\alpha_{comp}$ is the compression factor. For a compressor chicane (small bending angles) we have $T_{566} = -\frac{3}{2}R_{56}$ [69]. Equation (4.2) will be used to derive the expression for the bunch length in Sec. 4.3.3.

### 4.2.2 RF curvature

As illustrated in Fig. 4.1, the intrinsic sinusoidal form of the RF acceleration creates an energy curvature of the beam. Due to the finite longitudinal size of the bunch not all the particles can have the same phase on the RF waveform (see Fig. 4.1); some particles will receive more energy and some less. This results in an energy deviation from the reference particle at $z_{ref}$ (i.e. the particle at the ideal phase of the RF field). Moreover, altering the phase or the amplitude of the RF field will modify it, thereby modifying the energy deviation.

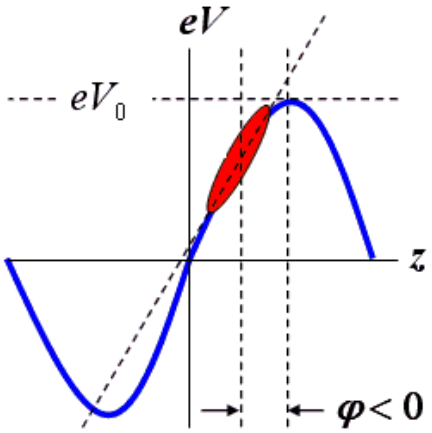



Figure 4.1: Energy chirp caused by the RF acceleration field. Because of the harmonic form of the RF acceleration field, a quadratic energy chirp is induced between the head and the tail of the bunch.

In what follows we derive an expression for the slope of this curvature as a function of the deviation in the phase and voltage of the RF field. This will allow us to calculate the relative energy deviation of the beam in the presence of perturbations in the amplitude and phase of the RF field [61]. The energy $\Delta E_{0,i+1}$ gained by an electron passing through the $(i+1)^{th}$ accelerating structure with RF phase $\phi$ and amplitude (voltage) $V$ is given by:

$$E_{0,i+1} = eV_{i+1}cos(\phi_{i+1}). \tag{4.3}$$

From Eq. (4.3), the overall energy gained by an electron at position $z$ within the bunch, from its passage through the $(i+1)^{th}$ accelerating structure of the accelerator, in the presence of jitter can be written as:

$$E_{i+1}(z) = E_i(z) + eV_{i+1}\left[\ 1 + \left(\ \frac{\Delta V}{V}\ \right)_{i+1}\ \right] cos(\phi_{i+1} + \Delta\phi_{i+1} + k_{i+1}z + k_{i+1}c\Delta t_i), \tag{4.4}$$

where $E_i(z)$ is the energy of the electron at the exit of the $i^{th}$ accelerating structure and $k_{i+1} = 2\pi/\lambda_{i+1}$ is the RF wave number of the $(i+1)^{th}$ accelerating structure. The voltage jitter (in the $(i+1)^{th}$ accelerating section) is included in the term $\Delta V_{i+1}$, while the total phase jitter is comprised of a local RF phase jitter $\Delta\phi_{i+1}$ and the timing jitter from the gun $\Delta t_{i+1}$. The energy of the reference electron (where $z = z_{ref}$ refers to the position of the design particle) without jitter is therefore given by ($\Delta t = \Delta\phi = \Delta V = 0$):

$$E_{i+1}(z_{ref}) = E_i(z_{ref}) + eV_{i+1}cos(\phi_{i+1}). \tag{4.5}$$

The energy deviation of an electron located at $z$ (from the design particle) is given by:

$$\begin{aligned}\delta_{RF,i+1}(z) &= \frac{E_{i+1}(z) - E_{i+1}(z_{ref})}{E_{i+1}(z_{ref})} \\ &= \frac{eV_{i+1}}{E_{i+1}(z_{ref})}\left[\left[\,1 + \left(\frac{\Delta V}{V}\right)_{i+1}\right]cos(\phi_{i+1} + \Delta\phi_{i+1} + k_{i+1}z + k_{i+1}c\Delta t_i) - cos(\phi_{i+1})\right].\end{aligned} \tag{4.6}$$

Since the bunch only occupies a small portion of the RF waveform, the energy deviation of the whole bunch caused by the RF curvature can be approximated by the following linear relation:

$$\delta_{RF,i+1} \equiv \Gamma_{RF,i+1}\Delta z, \tag{4.7}$$

where $\Delta z = \sqrt{12}\sigma_z$ is the bunch length for a uniform longitudinal bunch, $\sigma_z$ is the corresponding rms bunch length, and $\Gamma_{RF,i+1}$ is the slope of the RF accelerating field given by:

$$\begin{aligned}\Gamma_{RF,i+1} &= \left(\frac{\partial\delta_{RF}}{\partial z}\right)_{i+1} \\ &= -\frac{2\pi}{\lambda_{i+1}}\left[\,1 - \frac{E_i(z_0)}{E_{i+1}(z_0)}\,\right]\left[\,1 + \left(\frac{\Delta V}{V}\right)_{i+1}\right]\frac{sin(\phi_{i+1} + \Delta\phi_{i+1} + 2\pi c\Delta t_i/\lambda_{i+1})}{cos(\phi_{i+1})}.\end{aligned} \tag{4.8}$$

The energy deviation will also be affected by the chirp induced by wake fields created by the interaction of the electrons with the vacuum chamber, which is discussed in the next section.

### 4.2.3 Wake fields

When a bunch of electrons travels through the vacuum chamber an image charge is created in the beam pipe to cancel the field created by the electron bunch. The field created by the bunch is transverse to the beam trajectory and only a small section of the beam pipe will have an image charge. To entirely cancel the field created by the electrons in the bunch, the image charge should travel at the speed of light. Because of electrical resistance in the beam pipe, this is not achieved and a temporal lag is created, leading to imperfect cancelation and a remanent field, called the wake field. The effective field seen by an electron in the beam is the superposition of individual field contributions created by upstream electrons. This field produces an energy deviation in the beam causing energy loss. The effect of the wake field on the beam is amplified when the surface of the beam pipe has irregularities. In this case, the path traveled by electrons near the surface of the beam pipe is lengthened (since the electrons follow the shape of the irregularities), producing a temporal lag and modifying the remaining field seen by the electrons in the bunch. A example relevant to our study is the periodic array of cavities that form the structure of the accelerating section. In this case the longitudinal wake field function at a given position is given by [82]:

$$w(z) = \frac{Z_0 c}{\pi a^2} e^{-\sqrt{z/s_0}}, \tag{4.9}$$

where $Z_0$ is the characteristic impedance of the vacuum, $s_0$ is the spatial extent of the wake field and $a$ is the mean iris radius of the accelerating section. A particle at $z$ will experience an induced voltage due to the wake field created by all the particles ahead of it. The induced voltage is thus given by a convolution integral involving the wake field and the particle distribution within the bunch:

$$V(z) = -NeL \int_0^z w(z - z') f(z') dz', \tag{4.10}$$

where $L$ is the Linac length, $N$ is the number of electrons in the bunch, $e$ is the magnitude of the charge of the electron, and $f(z)$ is the particle distribution function of the bunch. We note that the limits of integration are from 0 to $z$, where $z = 0$ corresponds to the head of the bunch. The position of the particle experiencing the induced voltage is at $z$, as depicted in Fig. 4.2. To a first approximation $f(z)$ can be represented by a uniform distribution, although we could consider a Gaussian distribution. The beam generated by the RF gun at the LCLS and FERMI has a more complicated distribu-

tion; namely a "ramped parabolic" distribution designed to compensate for wake fields. Complicated distributions can be convolved numerically; however, here we only consider the simple case of a uniform distribution. For a uniform particle distribution in a bunch with extent $\Delta z$, the distribution function is:

$$f(z) = \begin{cases} \frac{1}{\Delta z} & \text{if } 0 \leq |z| \leq \Delta z \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

Equation (4.10), thus becomes:

$$V(z) = -\frac{NeL}{\Delta z} \int_0^z w(z - z')dz', \tag{4.12}$$

which gives, after integration:

$$V(z) = -\frac{2NeLZ_0cs_0}{\pi a^2 \Delta z} \left[ \; 1 - (1 + \sqrt{z/s_0})e^{-\sqrt{z/s_0}} \; \right]. \tag{4.13}$$

The energy deviation associated with the induced voltage for the entire bunch is $eV(\Delta z)/E_{i+1}(z_{ref})$. In the approximation of small bunches $\partial/\partial z \approx 1/\Delta z$ and we can write the slope of the energy chirp caused by the wake field as:

$$\Gamma_{W,i+1} = \left( \frac{\partial \delta_W}{\partial z} \right)_{i+1} \approx \left( \frac{eV(\Delta z)}{\Delta z E_{i+1}(z_{ref})} \right)_{uniform} = \\ - \frac{2Ne^2LZ_0cs_0}{\pi a^2 (\Delta z)^2 E_{i+1}(z_0)} \left[ \; 1 - (1 + \sqrt{\Delta z/s_0})e^{-\sqrt{\Delta z/s_0}} \; \right]. \tag{4.14}$$

The slope corresponds to the energy deviation caused by the wake field. The presence of wake fields will also cause an energy loss of the beam and therefore affect its mean energy. Equation (4.14) will be used in Sec. 4.3.1 to compute deviations in the mean energy of the beam. Moreover, the energy chirp associated with the wake field adds to the chirp caused by the RF curvature and will also affect the energy deviation. Since the bunch length is a function of the energy deviation, Eqs. (4.8) and (4.14) will be used to calculate the bunch length and the peak current in Secs. 4.3.2 and 4.3.3, respectively.

## 4.3 Observables

In the following sections we derive the formulas that give the "Observables" which the feedback aims to control. By Observable we mean a measurable

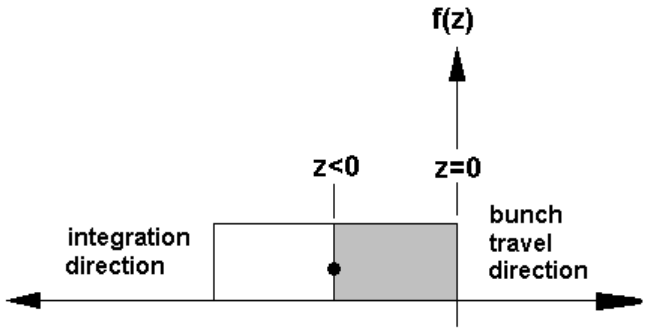



Figure 4.2: Induced voltage of the wake field for a uniform particle distribution. By convention $z = 0$ corresponds to the head of the bunch and $z < 0$ to any particle within the bunch. A particle at position $z$ within the bunch will experience a voltage induced by all the particles ahead, corresponding to the gray region of the bunch.

quantity that parameterises the beam, which can be modified by using appropriate actuators. As mentioned earlier, we are interested in controlling the mean energy of the beam and its bunch length. While the mean energy is an Observable (directly measurable), the peak current is an Observable related to the bunch length (since the bunch length cannot be directly measured in a non-destructive way). To calculate the peak current, we will derive the expression for the energy deviation in Sec. 4.3.2 and use this result to derive an expression for the bunch length in Sec. 4.3.3, which in turn is used to calculate the peak current in Sec. 4.3.4.

### 4.3.1 Mean energy deviation

The relative mean energy deviation of the beam passing through the $i^{th}$ accelerating section is defined as:

$$\left( \frac{dE}{E} \right)_i = \left( \frac{E-E_0}{E_0} \right)_i, \tag{4.15}$$

where $E$ and $E_0$ are the perturbed and unperturbed beam energies, respectively. The mean energy deviation after the $(i+1)^{th}$ Linac can be expressed as a sum of the energy deviation at the $i^{th}$ Linac, plus a term associated with the energy gain and jitter occurring in the $(i+1)^{th}$ section:

$$\left( \frac{dE}{E} \right)_{i+1} E_{i+1} = \left( \frac{dE}{E} \right)_i E_i + \left( \frac{dE}{E} \right)_{jitter,i+1} E_{gain,i+1}, \tag{4.16}$$

where $(dE/E)_{jitter,i+1}$ is the relative energy deviation due to voltage and phase jitter in the $(i+1)^{th}$ Linac and $E_{gain,i+1}$ is the net energy gain:

$$E_{gain,i+1} = (E_{i+1} - E_i) + E_{loss,i+1}. \tag{4.17}$$

Here $E_{loss,i+1} = -E_{i+1}\Gamma_{W,i+1}\frac{\sigma_z}{2}$ is the energy loss in the $(i+1)^{th}$ cavity due to the wake fields. A deviation in the mean energy can arise due to a perturbation in the phase $(d\phi)$ of the RF field or in its amplitude $(\Delta V)$. From Eq. (4.3), the unperturbed energy is:

$$E_{0,i+1} = eV_{i+1}cos(\phi_{i+1}). \tag{4.18}$$

Using Eq. (4.18), the perturbed energy can be written as:

$$E_{i+1} = e\Big[\ V_{i+1} + \Delta V_{i+1}\ \Big]cos(\phi_{i+1} + d\phi_{i+1}) \tag{4.19}$$

Using Eqs. (4.18) and (4.19), we obtain the energy deviation due to voltage and phase jitter. This is given by:

$$\Big(\ \tfrac{dE}{E}\ \Big)_{jitter,i+1} = \frac{E_{i+1} - E_{0,i+1}}{E_{0,i+1}} = \Big[\ 1 + \Big(\ \tfrac{\Delta V}{V}\ \Big)_{i+1}\ \Big]\Big[\ \tfrac{cos(\phi_{i+1}+d\phi_{i+1})}{cos(\phi_{i+1})} - 1\ \Big], \tag{4.20}$$

where $d\phi_{i+1} = dt_i\frac{c}{\lambda_{i+1}} + d\phi_i$ is the total local phase error (gun and Linac/chicane plus local RF). Assembling all terms one obtains [61]:

$$\begin{aligned}&\Big(\ \tfrac{dE}{E}\ \Big)_{i+1} = \\ &\Big(\ \tfrac{dE}{E}\ \Big)_i \frac{E_i}{E_{i+1}} + \Big[\ 1 - \tfrac{E_i}{E_{i+1}} - \Gamma_{W,i+1}\tfrac{\sigma_z}{2}\ \Big]\Big[\ \Big[\ 1 + \Big(\ \tfrac{\Delta V}{V}\ \Big)_{i+1}\ \Big]\tfrac{cos(\phi_{i+1}+d\phi_{i+1})}{cos(\phi_{i+1})} - 1\ \Big].\end{aligned} \tag{4.21}$$

This expression gives the relative mean energy deviation caused by phase and voltage perturbations in the accelerating sections. A relative measure is used to specify the machine stability. For example, the maximum energy deviation that can be tolerated at the entrance of the undulator chamber for FERMI is specified as 0.1% of the final beam energy.

### 4.3.2 Relative energy deviation

The relative energy deviation is directly related to the bunch length and consequently the peak current. This is because an energy deviation corresponds to a position deviation of a particle from the design particle. The relative energy deviation of a particle at $z_i$ within the bunch after the $i^{th}$ accelerating section is defined as:

$$\delta_i = \frac{E_i(z_i) - E_i}{E_i}. \tag{4.22}$$

To calculate the deviation induced by an accelerating section, we write the energy deviation after the $(i+1)^{th}$ Linac as a sum of the energy deviation after the $i^{th}$ section plus the contribution $\delta_{z,i+1}E_{i+1}$ of the RF curvature and wake field of the $(i+1)^{th}$ Linac:

$$\delta_{i+1}E_{i+1} = \delta_i E_i + \delta_{z,i+1}E_{i+1}, \tag{4.23}$$

where the deviation can be written as the sum of the RF and the wake field contributions:

$$\delta_{i+1}E_{i+1} = \delta_i E_i + E_{i+1}z_{i+1}(\Gamma_{RF,i+1} + \Gamma_{W,i+1}). \tag{4.24}$$

Here $\Gamma_W$ and $\Gamma_{RF}$ are the slopes of the linearized energy chirp, due to the wake field and RF curvature, respectively. The expression for $\Gamma_{RF}$ is given in Eq. (4.8) and the expression for $\Gamma_W$ is given in Eq. (4.14) for a uniform distribution. The relative energy deviation after the $(i+1)^{th}$ stage of the accelerator can be rewritten as:

$$\delta_{i+1} = \delta_i \frac{E_i}{E_{i+1}} + \Gamma_{i+1}z_{i+1}, \tag{4.25}$$

where $\Gamma_{i+1} = \Gamma_{W,i+1} + \Gamma_{RF,i+1}$ is the total slope along the bunch. Using the definition of the standard deviation[3], the energy deviation is given by [61]:

$$\sigma^2_{\delta,i+1} = \Gamma^2_{i+1}\sigma^2_{z,i} + \left(\frac{E_i}{E_{i+1}}\right)^2 \sigma^2_{\delta,i} + 2\Gamma_{i+1}\left(\frac{E_i}{E_{i+1}}\right)\langle z_i\delta_i\rangle. \tag{4.26}$$

This result will be used to calculate the bunch length in the next section.

[3]The standard deviation $\sigma_{XY}$ of the sum of two variables $X$ and $Y$ is given by $\sigma^2_{xy} = \mathbf{VAR}(X+Y) = \mathbf{VAR}(X) + 2\mathbf{COV}(X,Y) + \mathbf{VAR}(Y)$, where the variance is defined by: $\mathbf{VAR}(X) = \mathbf{E}(X^2) - (\mathbf{E}(X))^2$ and the covariance is defined by $\mathbf{COV} = \mathbf{E}(XY) - \mathbf{E}(X)\mathbf{E}(Y)$, with $\mathbf{E}$ the expectation value operator.

### 4.3.3 rms bunch length

Here we derive the expression for the bunch length that will be used to compute the value of the peak current, which is the second Observable we are interested in. We start by noting that for a relativistic beam, the energy $E$ is related to the momentum $p_z$:

$$E^2 = (p_z c)^2 + (m_0 c^2)^2, \tag{4.27}$$

where $m_0 = 0.511$ MeV$/c^2$ is the electron rest mass and $c$ is the speed of light in vacuum. For beams of a few hundreds MeV the second term in Eq. (4.27) can be neglected and the momentum deviation is thus equal to the energy deviation, since $E \simeq p_z c$. Equation (4.2) can thus be rewritten as a function of the energy deviation $\delta$. To second order this is given by:

$$z_{i+1} = z_i + \delta_{i+1} R_{56} + \delta_{i+1}^2 T_{566} \cong z_i + \delta_{i+1} R'_{56}, \tag{4.28}$$

where the term linear in $R'_{56}$,is defined by [81]:

$$R'_{56} = R_{56} + 2T_{566} \left( \frac{dE}{E} \right)_{i+1}. \tag{4.29}$$

Substituting for $\delta_{i+1}$ from Eq. (4.25), we can write Eq. (4.28) as:

$$z_{i+1} = (1 + \Gamma_{i+1} R'_{56}) z_i + R'_{56} \frac{E_i}{E_{i+1}} \delta_i. \tag{4.30}$$

According to the definition of the standard deviation, the rms bunch length is given by:

$$\sigma_{z,i+1}^2 = (1 + \Gamma_{i+1} R'_{56,i+1})^2 \sigma_{z,i}^2 + (R'_{56,i+1})^2 \left( \frac{E_i}{E_{i+1}} \right)^2 \sigma_{\delta,i}^2 + 2R'_{56,i+1}(1 + \Gamma_{i+1} R'_{56,i+1}) \left( \frac{E}{E_{i+1}} \right) \langle z_i \delta_i \rangle. \tag{4.31}$$

This result will be used to compute the peak current of the beam in the next section.

### 4.3.4 Peak current

We can now derive an expression for the peak current in the case where the particles are uniformly distributed within the bunch. The total current in the bunch is $I = Ne/\tau$, where $N$ is the total number of electrons and $\tau$ is the bunch duration. In the case of a uniform particle distribution on the

interval $[z_0, z_f]$, representing the edges of the electron bunch, the rms bunch length is $\sigma_z = 1/\sqrt{12}(z_f - z_0)$. Since the electrons are relativistic, after the first chicane the bunch duration can be written as $\tau = (z_f - z_0)/c$, and the peak current is:

$$I = \frac{Ne}{\tau} = \frac{Nec}{\sqrt{12}\sigma_z}, \tag{4.32}$$

where $\sigma_z$ is the rms bunch length given in Eq. (4.31). The measure of the peak current allows us to obtain a relative measurement of the bunch length in a non-destructive way, as discussed in Sec. 3.3.4. In this thesis, when we refer to peak current control it is understood that the beam parameter we wish to control is the bunch length, but experimentally the control is based on values of peak current measurements.

## 4.4 Controllables

The notion of "Controllable" refers to adjustable quantities used to correct an Observable value. Controllables are used to exert control over the beam energy and bunch length (the Observables). In Sec. 4.3 we identified that jitter in the phase and the voltage of the accelerating sections will be the most important source of perturbations for the beam energy and peak current. Since we can control the phase and the voltage of the accelerating section it makes sense to use these parameters as our Controllables.

The relation between Observables and Controllables is expressed using a response matrix. A simple approach is to assume linearity of the Observables to the Controllable in the range of interest for the control, i.e. the coefficients of the response matrix are assumed constant. However, as simulations will show in Chapter 5, this simplistic approach can degrade the quality of control. In Chapter 5, a Graphical User Interface is built with Matlab in order to determine the effects of the Controllables over the Observables, which will help us to determine appropriate Controllables for feedback.

# 5 Simulations of a PID controller for the FERMI FEL Linac

## 5.1 Motivation and objectives

A feedback system includes many parameters, such as different Controllables (Linac phase and voltage settings), Observables (energy and peak current), and jitter types; consequently, a systematic approach is essential to determine the optimal feedback system. To facilitate such a study, a Matlab Graphical User Interface (GUI) has been developed. The schematic of the GUI is given in Fig. 5.1; its components are numbered to correspond to the numbers in Sec. 5.2.

Figure 5.2 shows the stages for the LCLS longitudinal feedback system [4]. The numbering ($i = 1, 2, ..., 5$) corresponds to the five stages of the machine. A stage includes an accelerating structure and a dog leg (or chicane). For example, the first stage ($i = 1$) includes Linac 0 and the first dog leg.

The FERMI feedback structure shown in Fig. 5.3 is based on the LCLS and comprises five stages. At each stage the energy and the peak current are computed, which will allow us to study the ability of the PID control algorithm to reduce deviations in these parameters. In practice, the energy and the peak current cannot be measured at these five stages, because not all stages contain a dispersion region (dog leg or chicane). According to Fig. 5.3, energy measurements can only be made at stages $i = 2$ (first bunch compressor), $i = 4$ (second bunch compressor) and $i = 5$ (end of the accelerator). Peak current measurements can only be made at stages $i = 2$ and $i = 4$. For the LCLS, the accelerator comprises two more dog legs, thereby allowing for additional energy measurements. FERMI was also modelled as a five stage structure thereby obviating the need to adapt the script written for the LCLS. However, this will not affect the results, as only those stages where measurements of the peak current and/or energy are used for computing corrections with the PID algorithm.

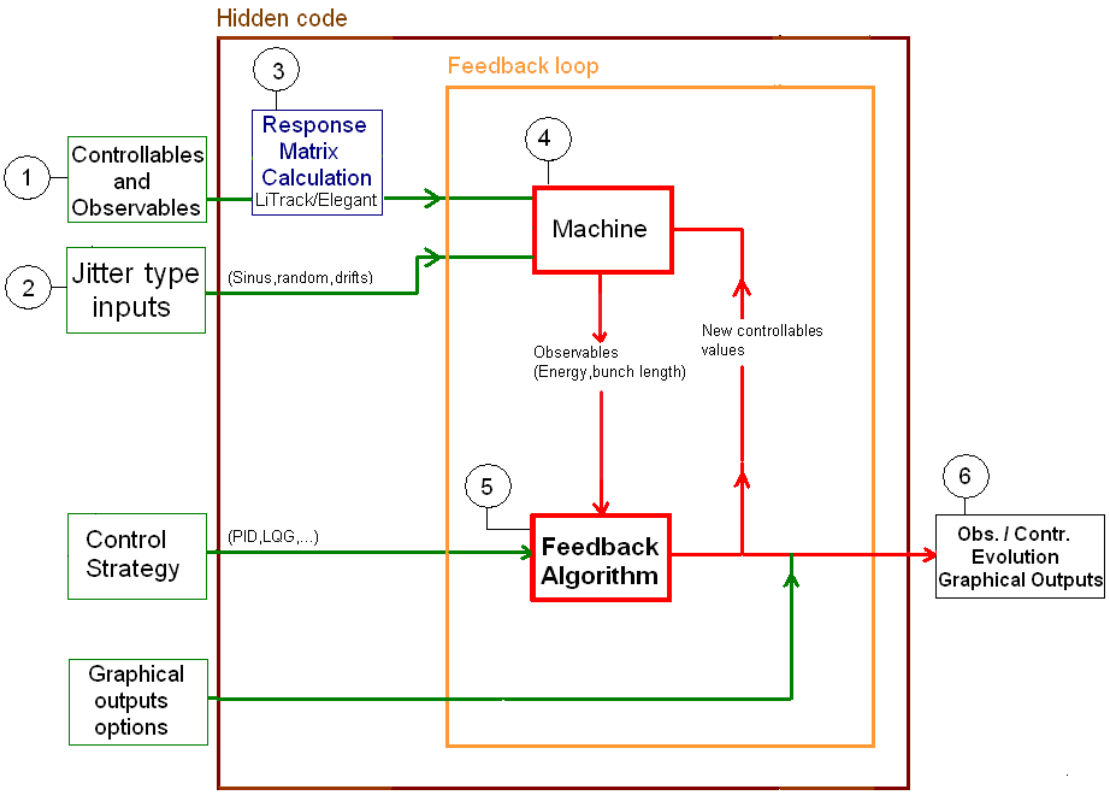



Figure 5.1: Schematic of the GUI for the FERMI simulation feedback system. The user can select the Observables and Controllables to be included in the feedback, the jitter characteristics, the control strategy, and some graphical outputs options. The hidden part of the code includes the response matrix of the selected Observables and Controllables, and the feedback loop. The feedback loop contains the PID algorithm and the machine to be simulated; it computes the equations developed in Chapter 4, or utilises tracking codes such as LiTrack and Elegant.

## 5.2 The elements of the Matlab GUI simulation framework

### 5.2.1 Controllables and Observables

Observables and Controllables can be selected from the GUI. For simplicity, the same number of Controllables and Observables will be used. Depending on the machine configuration (i.e. the number of bunch compressors used) there will be more or less freedom in the choice of Controllables.

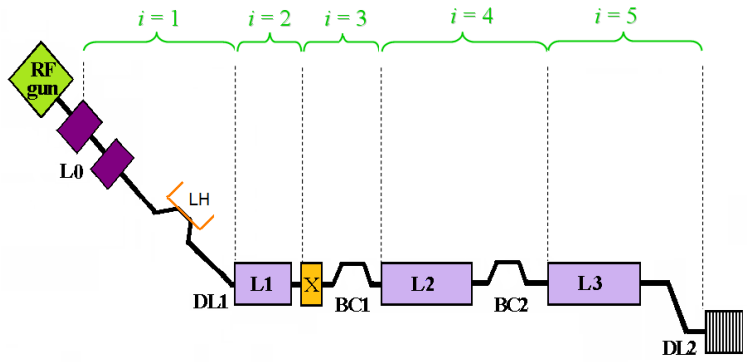



Figure 5.2: Five stage feedback structure for the LCLS machine. The LCLS control scheme includes five stages, encompassing the accelerating sections and bunch compressors [4].

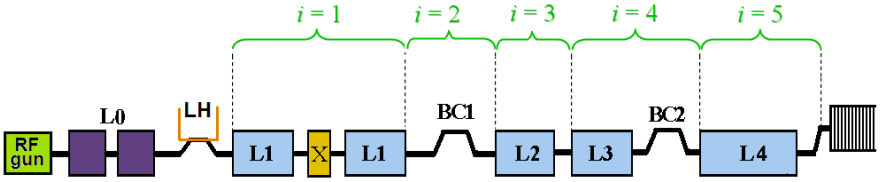



Figure 5.3: Five stage feedback structure for the FERMI machine. Energy feedback can be operated on stages 2, 4 and 5, whilst peak current feedback can be operated on stages 2 and 4 (if the machine configuration uses two compressors).

### 5.2.2 Jitter inputs

It is important to first identify the origin of jitter in the energy and bunch length. The beam energy jitter is clearly dominated by the Linac phases and voltages, since the energy gained by an electron passing through the accelerating section is $E = eVcos(\phi)$. Moreover, the bunch length is a function of the energy deviation (see Eq. (4.31)). The bunch length deviation is therefore also a function of the Linac phases and voltages. We develop a model for the perturbation in the energy and the bunch length as a function of the Linac phases and voltages [4], i.e.

$$\frac{dV}{V}[\%] = A_v rand(1) + \sum_{i=1}^{Nv} B_{v,i} sin(2\pi f_{v,i} t) + C_v t + D_v \sum_{j=1}^{Nstep} H(t - t_{step,j}) \qquad (5.1)$$

and

$$d\phi[^\circ] = A_\phi rand(1) + \sum_{i=1}^{N\phi} B_{\phi,i} sin(2\pi f_{\phi,i} t) + C_\phi t + D_\phi \sum_{j=1}^{Nstep} H(t - t_{step,j}). \qquad (5.2)$$

The first term in the right hand side of Eqs. (5.1) and (5.2) is a uniformly distributed random error, representing white noise that can arise from measurements or which originates in the phase and voltage of the Linac. The second term is a sum of sinusoids, reflecting periodic variations (e.g., a cyclic variation in the cooling system) that can affect the phase and voltage of the Linac. The third term is a slow drift that can occur because of accelerator components heating over time or when the machine is restarted after a shut down. The last term represents sudden jumps that might occur; for example, in a power supply feeding several accelerator components, such as klystrons and magnets. When a change of load occurs (e.g., a component is turned on or off), this will cause a jump in the power supply output, if the regulation system is not fast enough. The characteristics of the jitter described in Eqs. (5.1) and (5.2) can be selected and modified with the GUI.

### 5.2.3 Response matrix

The response of a Controllable to a change in the value of an Observable, is described by a response matrix. Initially we had assumed that the Observable - Controllable relation is linear, or acceptably linearisable in the region of interest for operating the control. However, further investigation showed that, in some cases, the Observable-Controllable relation is strongly nonlinear. Details of these studies will be given in Sec. 5.3. The response matrix can either be evaluated analytically, as with the LCLS, or by using a tracking code, such as LiTrack or Elegant. The matrix elements are scanned and their linear(ised) forms included in the GUI scripts. Each time the Observable-Controllable configuration is modified, the elements are automatically rearranged to form the corresponding new response matrix. The response matrix is defined as follows:

$$[O_i] = [M_{ij}][C_j], \qquad (5.3)$$

where $M_{ij} = \partial O_i / \partial C_j$, $O_i$ is the $i^{th}$ element of the Observable vector **O** and $C_j$ is the $j^{th}$ element of the Controllable vector **C**. For the FERMI machine, and for the one bunch compressor configuration, this expression takes the following explicit form:

$$\begin{bmatrix} \frac{dE_{BC1}}{E_{BC1}} \\ \frac{dI_{BC1}}{I_{BC1}} \\ \frac{dE_{S\,pr.}}{E_{S\,pr.}} \end{bmatrix} = \begin{bmatrix} M_{11} & 0 & 0 & 0 & M_{15} & M_{16} & 0 & 0 & 0 & M_{110} \\ M_{21} & 0 & 0 & 0 & M_{25} & M_{26} & 0 & 0 & 0 & M_{210} \\ M_{31} & M_{32} & M_{33} & M_{34} & M_{35} & M_{36} & M_{37} & M_{38} & M_{39} & M_{310} \end{bmatrix} \begin{bmatrix} \frac{dV_1}{V_1} \\ \frac{dV_2}{V_2} \\ \frac{dV_3}{V_3} \\ \frac{dV_4}{V_4} \\ \frac{dV_x}{V_x} \\ d\phi_1 \\ d\phi_2 \\ d\phi_3 \\ d\phi_4 \\ d\phi_x \end{bmatrix}, \quad (5.4)$$

where $dE_{BC1}/E_{BC1}$ and $dI_{BC1}/I_{BC1}$ are the relative deviations in the energy and the peak current from their desired settings at the first bunch compressor. $dE_{S\,pr.}/E_{S\,pr.}$ is the relative energy deviation at the spreader (i.e. at the end of the accelerator). The Controllable vector includes the deviations in the voltage and the phase of the X-band linearisers and the four main accelerating structures, as shown schematically in Fig. 5.3.

### 5.2.4 Machine simulation codes

Three different codes can be selected in the GUI to simulate the machine. The first is a fast Matlab script (FMS) that was implemented at the LCLS [4] and adapted to include the characteristics of the FERMI machine. This code computes the main equations describing longitudinal dynamics discussed in Chapter 4, i.e. RF acceleration, second order compression, and wake fields. Although it is not very accurate, the FMS code has the advantage of being very fast (2-3 seconds for the computation of 1000 bunches) and gives good insight into the trends of the system response.

A second means to simulate the machine is LiTrack. This tracking code also accounts for the main longitudinal dynamics, but with more precision, including a third order compression term. For more realistic scenarios LiTrack reads particle and wake field distributions from files produced by other simulation codes (such as ASTRA[1]), whereas the FMS code assumes a uniformly distributed beam at the end of the photoinjector.

The third code uses Elegant, which is C-based and takes much longer to run than the two preceding codes. Typically, a LiTrack run will take 7 to 8 seconds, whereas Elegant will take 3 to 4 minutes depending on the speed

[1]ASTRA is a space charge tracking algorithm. It tracks the bunch as a distribution of macro particles through the combined external and self induced fields generated by the charged particles. It is used for beams where space charge effects are important [83].

of the processor. Unlike the other codes, Elegant models the machine as a sequence of physical components, in which parameters are read from a sequence file. For this reason, Elegant and LiTrack produce different outputs for the compressors.

### 5.2.5 Feedback algorithm

A Proportional-Integral-Differential (PID) algorithm was first implemented because of its simplicity and demonstrable robustness [84]. From Chapter 2 we recall the PID algorithm in the discrete time domain is defined by:

$$u(i) = P_g e(i) + I_g \sum_{i_s=0}^{i} e(i_s) + D_g(e(i) - e(i-1)), \tag{5.5}$$

where $e(i) = r(i) - y(i)$ is the tracking error, with $y(i)$ the measured output and $r(i)$ the desired output. The response is obtained by adjusting the three parameters $P_g$, $I_g$ and $D_g$, also called the PID gains (see Chapter 2). Stability can often be ensured using only the proportional term $P_g$. The integral term $I_g$ permits the rejection of a step disturbance. The derivative term $D_g$ is used to provide damping or shaping of the response. In the GUI, the controller gains can be modified for each particular Controllable along the machine.

### 5.2.6 Simulation outputs of Observables/Controllables

When launched, the main GUI panel looks as shown in Fig. 5.4. One can recognise the options described above. A complementary "Detector and Actuator" panel serves to adjust the characteristics of the CSR detector.

After Matlab has performed the simulation, it launches the display panel shown in Fig. 5.5. On the far right, one can select the display of a specific Controllable response. The blue curve shows the chosen Controllable behavior without feedback, whereas the green curve gives the Controllable response when feedback is on.

On the left side of the panel shown in Fig. 5.4 two columns of four graphs are displayed. The four graphs in the left column show the deviations in the system Observables when the feedback is off, whereas the deviations in the Observable responses with feedback on are displayed in the right column. The displayed Observables are the energy (first row), the peak current and corresponding power in the CSR detector current (second and third row) and the timing jitter (fourth row). The Observable can be displayed for each stage of the machine.

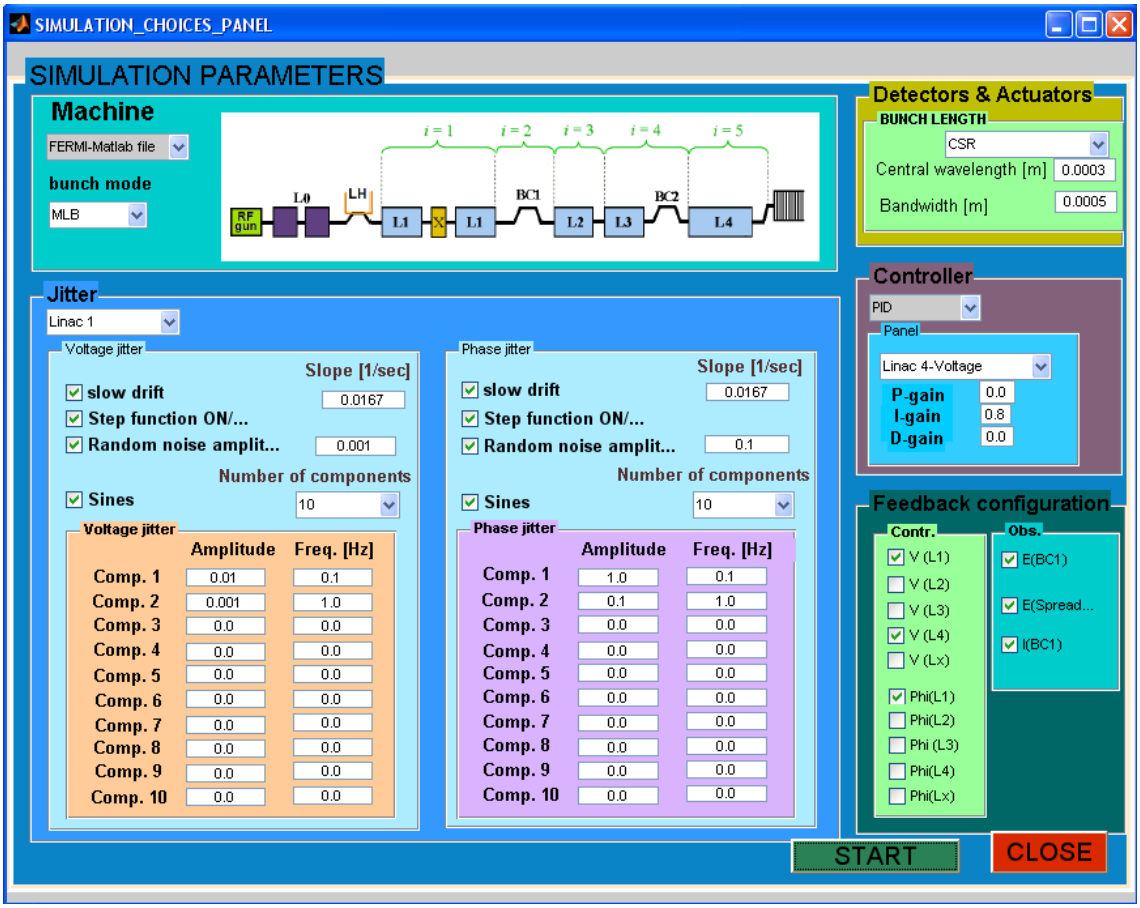



Figure 5.4: Main GUI simulation panel. The upper left "Machine" panel allows the user to choose between the LCLS and FERMI machines, as well as the code to be used (fast longitudinal computation, LiTrack or Elegant). The lower left "Jitter" panel allows us to enter jitter parameters according to the model given in Eqs. (5.1) and (5.2). Feedback strategies and their specifications can be accessed via the "Controller" panel, whereas Observables and Controllables can be selected from the bottom right panel.

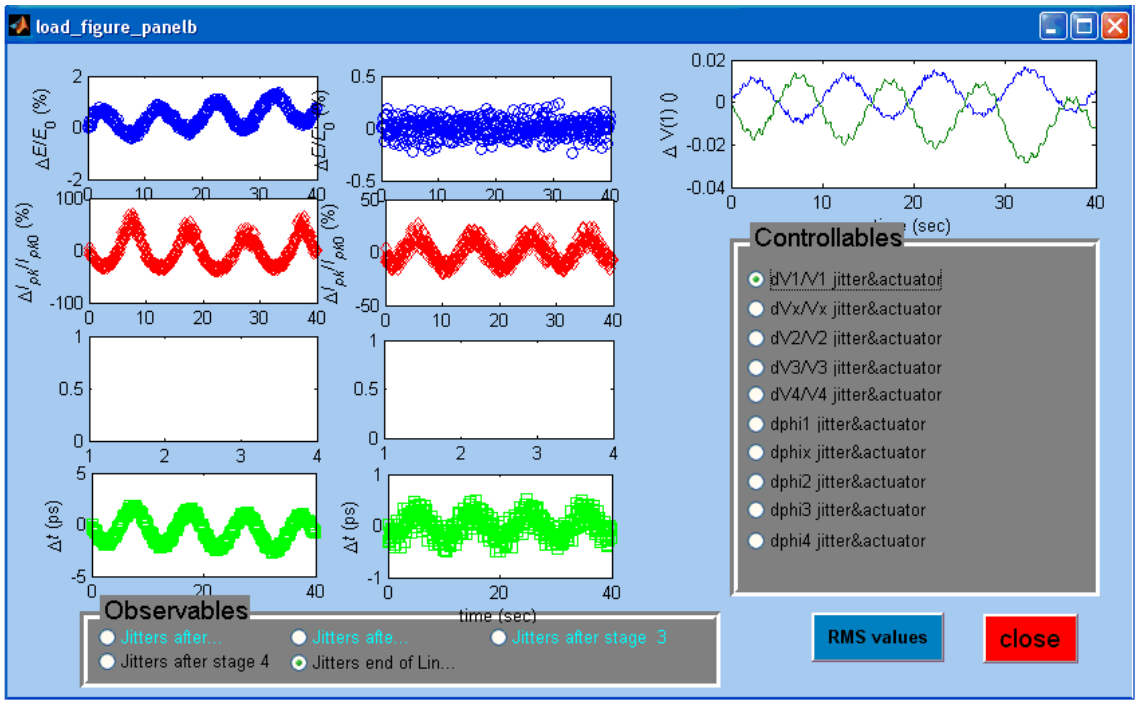



Figure 5.5: GUI simulation display panel. The left side of the panel displays two columns of four graphs. The deviations in the Observables (energy, peak current and timing) with feedback off are shown in the left column, while the deviations in the Observables with feedback on are displayed in the right column. The display corresponds to the machine stage selected in the "Observable" menu. A specific Controllable can be displayed on the far right plot by selecting it from the "Controllable" menu located below the corresponding graph.

## 5.3 Evaluation of the response matrix elements

The response matrix is a function of the chosen Controllables and Observables. To find which Controllables are best suited for control, their response is examined. A Controllable should provide a linear response in the Observable it aims to correct. When the response is not linear the elements of the response matrix are a function of the machine settings and should be fitted from measurements. Results show that the response of some of the elements exhibit strong non-linearities that can result in a lack of accuracy in the correction. Moreover, the sensitivity of an Observable to a change in a Controllable is also important. Sufficient sensitivity is required to ensure a fast response and to avoid saturation of the klystrons. On the other hand, over sensitivity of a corrector might cause a non acceptable overshoot during correction.

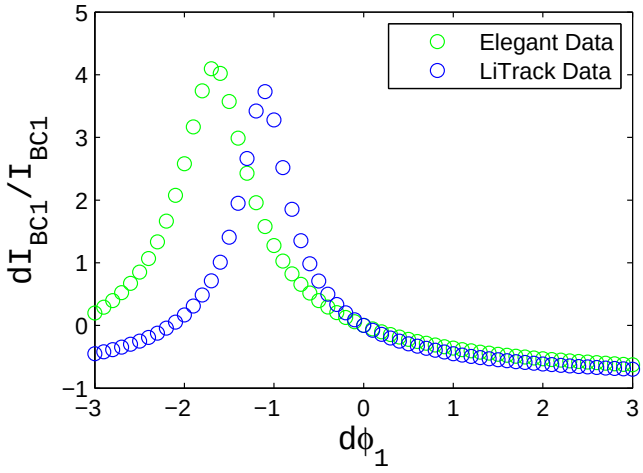



Figure 5.6: Calculation of the relative peak current deviation $dI_{BC1}/I_{BC1}$ at BC1 for FERMI versus the deviation in the phase of Linac 1 $d\phi_1$. The simulation was performed with LiTrack and Elegant. The shape and amplitude of the response is preserved, with a phase shift due to differences in compressor implementation between LiTrack and Elegant.

For the FERMI machine, the response matrix was calculated using both LiTrack and Elegant. Whilst the two codes gave identical results for matrix elements for a Controllable located downstream from the bunch compressor, Figs. 5.6 and 5.7 show that they give different results for matrix elements associated with a Controllable located before the bunch compressor. As an example, one can see from Fig. 5.6 that the response of the relative peak current deviation is non-linear, with a maximum phase shift at -1.1° and

−1.8° from the initial settings using LiTrack and Elegant, respectively. Looking at the response of the relative energy deviation at the end of the machine (Fig. 5.7) one sees the slope of the response is changed, whereas both curves are linear with a change in their slope at around -1.5°. These effects are due to differences in modeling the compressor between the two codes. As discussed in Sec. 5.2.4, LiTrack computes only longitudinal dynamics, whereas Elegant requires detailed information on the transverse lattice of the accelerator in order to compute the transverse dynamics.

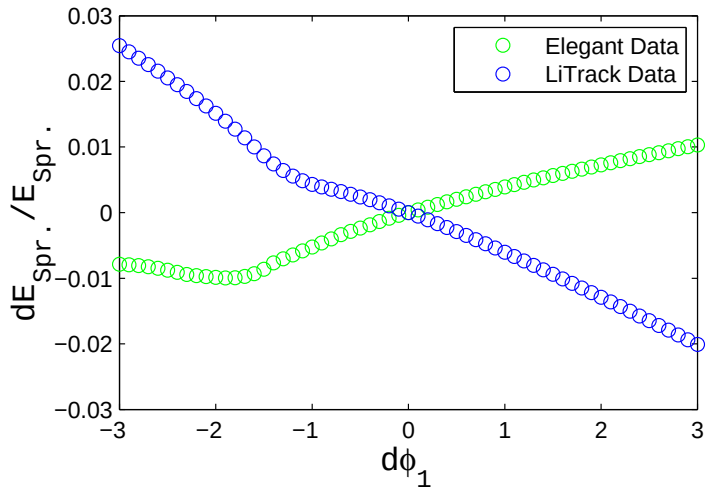



Figure 5.7: Evaluation of the relative energy deviation $dE_{S\,pr.}/E_{S\,pr.}$ at the end of the FERMI accelerator versus the deviation in the phase of Linac 1 $d\phi_1$. A rotation of the curve between Elegant and LiTrack is due to differences in the way the two codes model the compressor.

Table 5.1 summarises the response matrix elements predicted by Elegant for the one bunch compressor configuration of FERMI. They were calculated for a ±4% change in the voltages of the Linacs and a ±1.5° change in the phases (to avoid the peaks). The results indicate that the relative deviations in the energy and the peak current at the bunch compressor are most affected by a change in the voltage of the first Linac ($M_{11}$ = 0.612 and $M_{21}$ = 16.2). Similarly, the relative energy deviation at the end of the machine is most affected by the voltage of Linac 4 ($M_{34}$ = 0.487). The system is therefore very sensitive to jitter in the voltages of Linac 1 and Linac 4.

For comparison the LCLS matrix elements were derived using LiTrack. Compared to FERMI, the LCLS accelerator structure includes two bunch compressors and two dog legs, which allow for energy measurements at more locations along the machine. The results indicate very good linearity for all the energy dependent matrix elements over a relatively wide con-

Table 5.1: FERMI response matrix elements evaluated with Elegant for a ±4% change in voltage and a ±1.5° change in phase. The first column identifies the matrix elements as defined in Eq. (5.4). Columns 2 and 3 are the associated Controllables and Observables, respectively; while column 4 gives the actual values of the elements.

| Element | Controllable | Observable | Element value |
|---|---|---|---|
| $M_{11}$ | $V_1$ | $E_{BC1}$ | 0.612 |
| $M_{15}$ | $V_x$ | $E_{BC1}$ | -0.0835 |
| $M_{16}$ | $\phi_1$ | $E_{BC1}$ | 0.0128 |
| $M_{110}$ | $\phi_x$ | $E_{BC1}$ | -0.000743 |
| $M_{21}$ | $V_1$ | $I_{BC1}$ | 16.2 |
| $M_{25}$ | $V_x$ | $I_{BC1}$ | -2.52 |
| $M_{26}$ | $\phi_1$ | $I_{BC1}$ | -0.849 |
| $M_{210}$ | $\phi_x$ | $I_{BC1}$ | 0.227 |
| $M_{31}$ | $V_1$ | $E_{S\,pr.}$ | 0.0667 |
| $M_{32}$ | $V_2$ | $E_{S\,pr.}$ | 0.126 |
| $M_{33}$ | $V_3$ | $E_{S\,pr.}$ | 0.214 |
| $M_{34}$ | $V_4$ | $E_{S\,pr.}$ | 0.487 |
| $M_{35}$ | $V_x$ | $E_{S\,pr.}$ | -0.00893 |
| $M_{36}$ | $\phi_1$ | $E_{S\,pr.}$ | 0.00458 |
| $M_{37}$ | $\phi_2$ | $E_{S\,pr.}$ | 0.000715 |
| $M_{38}$ | $\phi_3$ | $E_{S\,pr.}$ | 0.00122 |
| $M_{39}$ | $\phi_4$ | $E_{S\,pr.}$ | 0.00491 |
| $M_{310}$ | $\phi_x$ | $E_{S\,pr.}$ | -0.00878 |

troller range (i.e. ±10% in voltage and ±3° in phase). However, matrix elements related to the peak current Observable exhibit significant non-linear behavior. The results also illustrate that the initial machine settings (working point in terms of cavity voltages and phases) are critical for determining the peak current response to jitter. In Fig. 5.8 the relative deviation of the peak current exhibits a maximum for a +2.5% deviation in $dV_0/V_0$ (i.e. the voltage of Linac 0 from its initial setting). At that particular setting the relative peak current deviation $dI_{BC2}/I_{BC2} = 8$, which corresponds to a relative deviation of 800% from the initial set point for the peak current (at BC2 the set value of the peak current is 3.2 kA). In Fig. 5.9 one can see similar behavior, with the shift being to the left. However, the peak is reached for a larger deviation from the initial settings of Linac 2 (-7%). Therefore one would chose the voltage of Linac 2 to correct the jitter rather than Linac 0, since it is linear over a larger voltage range.

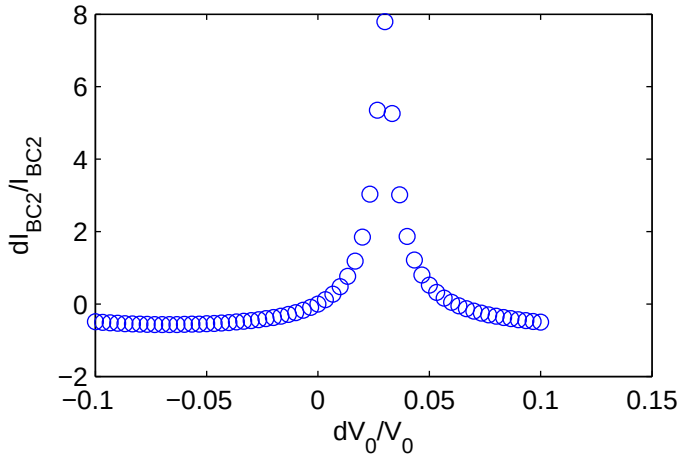



Figure 5.8: Calculation of the relative deviation in the peak current $dI_{BC2}/I_{BC2}$ at BC2, for the LCLS, versus $dV_0/V_0$ (the relative deviation of the voltage of Linac 0). The simulation was performed with LiTrack. The relative peak current deviation (after the second bunch compressor) to a change in the second Linac voltage shows a maximum for a +2.5% change in the voltage of Linac 0. At this setting the peak current has a relative deviation of 800% from its initial setting.

Figure 5.10 shows the response of the relative peak current deviation at BC2 to a shift in the phase of Linac 2. An ill-defined peak is observed at around -2.2°. For a -2.2° deviation in the phase of Linac 2 the peak current experiences a 8000% deviation from its set point of 3.2 kA, which is almost ten times the amplitude of the deviation experienced by the peak current to a shift in the voltage of Linac 2 (see Fig. 5.9). Consequently, any change in the machine's initial settings will play an important role in the feedback performance and one must give careful consideration to the range over which a matrix element is defined. If the necessary corrections exceed ±4% of the voltage, or ±1.5° of the phase, the matrix elements need to be re-evaluated.

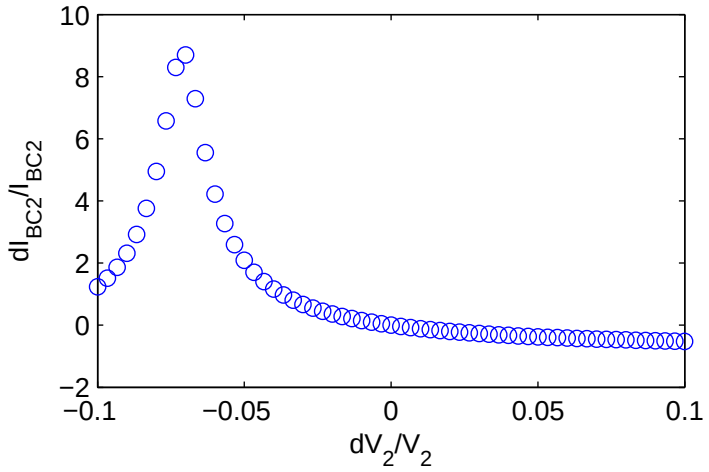



Figure 5.9: Calculation of the relative deviation in the peak current $dI_{BC2}/I_{BC2}$ at BC2, for the LCLS, versus $dV_2/V_2$ (the relative deviation of the voltage of Linac 2). The simulation was performed with LiTrack. Changes in the voltages of Linac 2 and Linac 0 produce a maximum of similar magnitude to that shown in Fig. 5.8. However, the maximum is reached for a larger deviation in the initial voltage setting of Linac 2, i.e. at about -7.5%.

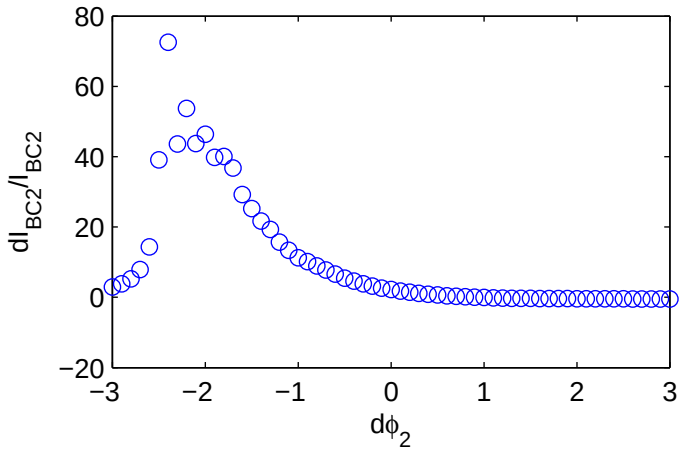



Figure 5.10: Calculation of the relative deviation in the peak current $dI_{BC2}/I_{BC2}$ at BC2, for the LCLS, versus $d\phi_2$ (the deviation in the phase of Linac 2). The simulation was performed with LiTrack. The peak current after the second bunch compressor is evaluated for a change in the phase of Linac 2. This shows an ill-defined peak around -2.2°. For such a deviation the peak current experiences a relative deviation of 8000% from its initial setting, which makes the Linac 2 phase a critical parameter to control for peak current stabilisation.

## 5.4 Feedback configuration for one bunch compressor

Because the first stage of the FERMI accelerator comprises one bunch compressor, there are only three Observables (see Eq.(5.4)); consequently we investigate a feedback configuration that consists of three Controllables. As two of the three Observables (energy and peak current) are measured at the bunch compressor, two of the three Controllables must be chosen among the phases and voltages of Linac 1 and the X-band section (located upstream from the compressor). The X-band cavity is much shorter than Linac 1 and its contribution to the beam energy is much smaller, thereby necessitating large changes in its phase or amplitude to correct for jitter. Moreover, because the phase and amplitude of the X-band cavity must remain stable to compensate for wake field effects, this cavity is not suitable for control. Therefore, the first two Observables must be the voltage and the phase of Linac 1. This leaves only one more Controllable to choose. It is sensible to directly consider the voltage and phase of the last Linac, since this section contributes the largest fractional change to the total energy of the beam (compared to the other Linacs). Therefore Linac 4 will allow for smaller changes in the phase and voltage in order to achieve the desired corrections to the energy and peak current of the beam. According to Table 5.1, the phase and voltage have a similar ability to attenuate jitter. However, since the voltage actuators have a much faster response (of the order of 5% of the initial value per $\mu s$) than the mechanical phase shifter, whose response is slower than the beam repetition rate, we will consider the voltage of Linac 4 as our third Controllable.

The feedback will therefore use the Linac 1 phase and voltage, and Linac 4 voltage as the set of Controllables. The Observables will comprise the energy and peak current at the bunch compressor, together with the energy at the end of the accelerator.

## 5.5 Performance of the PID algorithm

We present results for the simplest of the three available codes, i.e. the FMS code, which includes longitudinal dynamics. For this simple case we evaluate the residual energy and peak current jitter over 1500 bunches, for a set of proportional and integral gains ranging from 0.1 to 1.5 in steps of 0.05, and for only one Observable and Controllable at a specific jitter frequency. The same studies with Elegant are more computationally intensive and fewer results are presented. In general, we will use Elegant in order to validate the results of the simpler FMS code.

### 5.5.1 Optimisation of the PID gains

The usual way to tune the gains of a PID controller is to first tune the proportional gain $P_g$, then the integral gain $I_g$, and finally the differential gain $D_g$ [85]. The gain $D_g$ is used to compensate for high frequency jitter by increasing the system's response and is the most difficult to tune correctly. This is why many controllers use proportional and integral gains only. In conventional control theory, methods are given to tune PID controllers which assume the system is at least of first order. However, in our case the system is modeled with the response matrix, which is of zero order. For this reason, scans of the residual energy and peak current jitter were made as a function of the gain settings in order to tune the gains of our PID controller. Three types of scans were carried out. The first scans aimed to optimise the combination of integral gain only (gain $P_g$ and $D_g$ are zero). A second set of scans combined $P_g$ and $I_g$ gains for the different Observables to see the effect of the proportional gain. Finally, the last scans investigated the efficacy of the gain $D_g$.

#### Scans of integral gain

In the first set of scans, the standard deviations of the relative peak current and energy deviation were evaluated for combinations of the integral gains, $I_g(V_1)$ (Linac 1 voltage), $I_g(V_4)$ (Linac 4 voltage) and $I_g(\phi_1)$ (Linac 1 phase). The scans were performed with the FMS code and compared with Elegant. In both cases a combination of sinusoidal perturbations with frequencies of 0.1 Hz and 1 Hz was applied to both the phase and the voltage of Linac 1, with amplitudes of $1^\circ$ and 1%, respectively. Figure 5.11 compares the results obtained with the FMS code and Elegant. The relative energy deviation at the bunch compressor was evaluated as a function of the integral gains on the voltages of Linac 1 and Linac 4. Figures 5.12 and 5.13 show the standard deviation of the relative peak current and energy deviations at the end of the accelerator. Note that since there is no second compression stage the peak current is the same after the bunch compressor and the end of the accelerator.

The low resolution in Figs. 5.11, 5.12 and 5.13 is due to the significant time required to run Elegant. Although better resolution can be achieved with the FMS code, for purpose of comparison, the same resolution was used in comparing the FMS code and Elegant. The scans were performed for gains ranging from 0.2 to 1, with a step size 0.2 (giving a total of 30 plots). In what follows we present the plots obtained for a fixed gain $I_g(\phi_1) = 0.2$.

This choice was made because the corresponding plots clearly show the interplay of the gains (i.e. the effect of a combination of gains on the control).

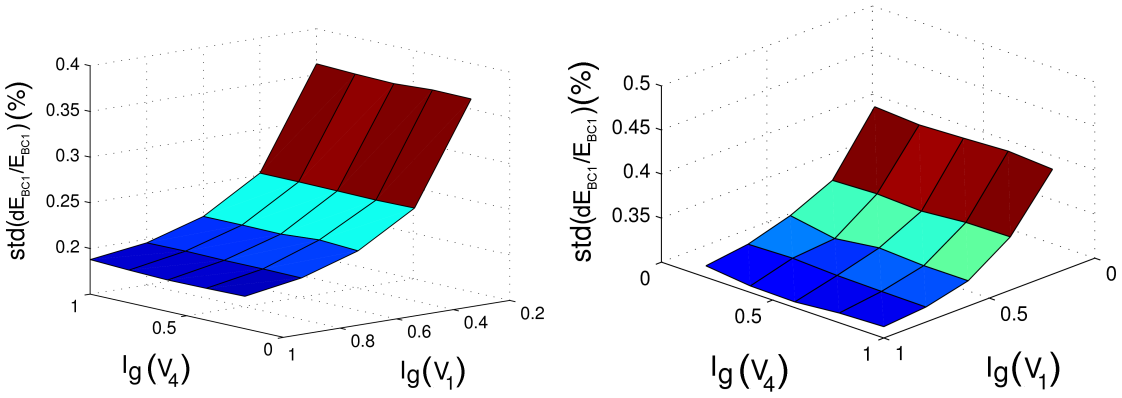



Figure 5.11: Standard deviation of the relative energy deviation $std(dE_{BC1}/E_{BC1})$ at BC1 for FERMI, calculated with the FMS code (left) and Elegant (right) as a function of $I_g(V_1)$ and $I_g(V_4)$. Both the fast code and Elegant show that an increase in $I_g(V_1)$ reduces the standard deviation in the peak current, while an increase in $I_g(V_4)$ has no effect, since Linac 4 is located downstream of the bunch compressor.

Figure 5.11 shows good agreement between the results obtained with the FMS code and Elegant. Both codes predict that a higher $I_g(V_1)$ will reduce the relative energy deviation. An increase in $I_g(V_4)$ has no effect, since Linac 4 is located downstream of the bunch compressor. However, Fig. 5.12 shows the opposite trend, i.e. the FMS code predicts a high value of $I_g(V_1)$ should reduce the relative peak current deviation, while Elegant predicts that a low gain should produce better results. This can be explained by the differences in the way the compressor is modeled in the two codes. $I_g(V_4)$ has no effect since Linac 4 is located downstream of the bunch compressor. In Fig. 5.13 Elegant and the fast code both predict that high $I_g(V_1)$ and $I_g(V_4)$ gains minimise the deviation. However, the effect of $I_g(V_1)$ is small compared to $I_g(V_4)$, since Linac 1 provides much less energy to the beam than does Linac 4.

The FMS code shows that there is an interplay between the gains (see Fig. 5.13). It appears that a combination of high gains for both actuators provides the best correction. Nonetheless, a combination of high $I_g(V_1)$ and low $I_g(V_4)$ (and vice versa) amplifies the perturbation. Indeed, a gain of 0.2 for both actuators gives rise to a lower residual perturbation than for the low-high gain combination. This effect can have different causes. First, as it

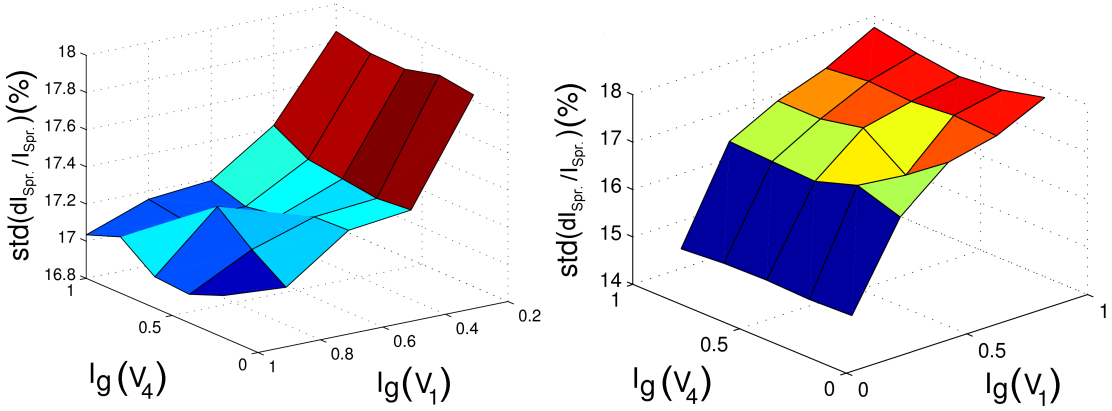



Figure 5.12: Standard deviation of the relative peak current deviation $std(dI_{BC1}/I_{BC1})$ at BC1for FERMI, calculated with the FMS code (left) and Elegant (right) as a function of $I_g(V_1)$ and $I_g(V_4)$. Left: High $I_g(V_1)$ provides better reduction of the peak current. Right: Elegant shows that lower $I_g(V_1)$ provides better attenuation of the perturbation. $I_g(V_4)$ has no effect since Linac 4 is located downstream of the bunch compressor.

will be shown in Sec. 5.5.2, the gain is frequency dependent. Second, there is also a dependence on the machine settings. In the previous section it was shown that some elements of the response matrix are strongly non-linear, especially $M_{26}$, which corresponds to the slope of the relative deviation of the peak current to the phase of Linac 1 (see Fig. 5.6). We note that that the relative deviation in the peak current at BC1 and at the spreader (end of the accelerator) are equal, since there is no second compression stage here. In fact, for a deviation just above 1° in amplitude, the response matrix element derived with LiTrack changes sign. Thus, the gain used for $\phi_1$ can either amplify or damp the perturbation, depending on the jitter amplitude in $\phi_1$.

Table 5.2 lists the optimised gains and their corresponding performance in terms of residual energy and peak current deviation. As one can see, the optimised gains are very similar for both codes. Whilst it is expected that Elegant would give higher residual deviations, this is only true for the energy. Indeed, the residual peak current deviation is smaller (5.65%) with Elegant than it is with the FMS code (6.3%).

#### Scans of proportional and integral gains

Here we consider the combined effect of the proportional and integral gains on $\phi_1$, which is used to correct the deviations in the beam energy at the bunch compressor and at the end of the accelerator. A 0.5 Hz jitter frequency was applied to the voltage and to the phase of Linac 1, with am-

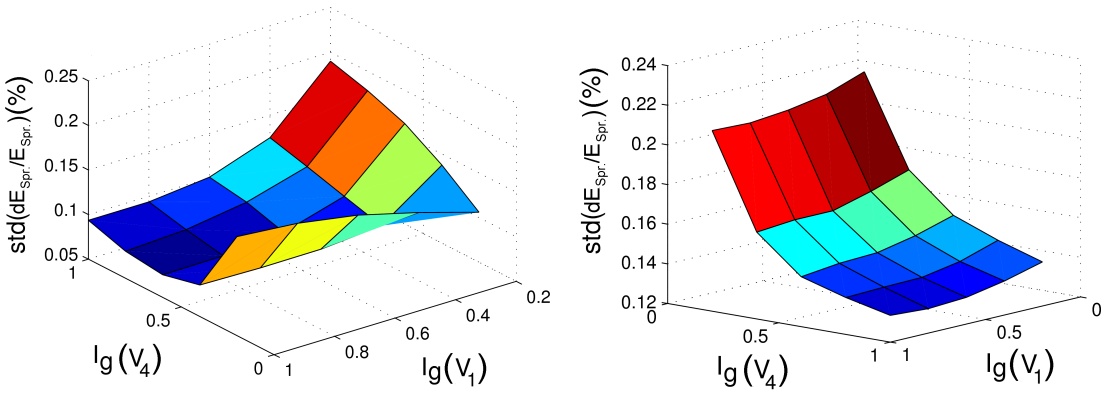



Figure 5.13: Standard deviation of the relative energy deviation $std(dE_{S\,pr.}/E_{S\,pr.})$ at the end of the FERMI accelerator, calculated with the FMS code (left) and Elegant (right), as a function of $I_g(V_1)$ and $I_g(V_4)$. Left: A decrease in $I_g(V_1)$ reduces the energy standard deviation (for small $I_g(V_4)$). Right: High $I_g(V_4)$ achieves better attenuation of the energy jitter, while $I_g(V_1)$ does not attenuate the perturbation significantly.

Table 5.2: Optimised integral gains, residual energy and peak current deviations obtained with the fast computation code (FMS) and Elegant for a jitter comprising 0.1 Hz and 1 Hz components.

| | FMS code | Elegant |
|---|---|---|
| Linac 1 voltage integral gain $I_g(V_1)$ | 1.1 | 1.0 |
| Linac 1 phase integral gain $I_g(\phi_1)$ | 1.0 | 1.0 |
| Linac 4 voltage integral gain $I_g(V_4)$ | 0.8 | 1.0 |
| std$(dE_{S\,pr.}/E_{S\,pr.})$ | 0.064 % | 0.079% |
| std$(dI_{BC1}/I_{BC1})$ | 6.30 % | 5.65% |

plitudes of 1° and 1%, respectively. The integral gains on the voltages of Linac 1 and Linac 4 were set to 0.5, while their proportional and differential gains were all set to 0. It was found necessary to include non-zero integral gains for the voltages of Linac 1 and Linac 4 ($V_1$ and $V_4$) of the feedback configuration, otherwise no correction was observed at the end of the accelerator. The value of 0.5 for $I_g(V_1)$ and $I_g(V_4)$ was chosen to ensure that the effect of $P_g(\phi_1)$ and $I_g(\phi_1)$ could be observed (gain sufficiently low) and such that the system does not become unstable (gain too high). The need to include non-zero integral gains for the voltages of Linac 1 and Linac 4 reflects the fact that a one Observable - one Controllable configuration located at the beginning of the machine is unable to overcome jitter occurring downstream. This can be understood by examining the values of the response matrix elements $M_{34}$ and $M_{36}$ listed in Table 5.1. This shows that there is a

factor of 100 between the values of these two elements, with $M_{34} = 0.487$ and $M_{36} = 0.00458$, respectively. This means that to correct a 1% deviation in the beam energy at the end of the accelerator, a 1% change in the voltage of Linac 4 is required, while a change of 100° in the phase of Linac 1 would be required. Similarly, to correct a 1% perturbation in the voltage of Linac 4, using the phase of Linac 1, would require a 100° phase change. The results observed at the bunch compressor are shown in Fig. 5.14, where the proportional gain reduces the energy deviation.

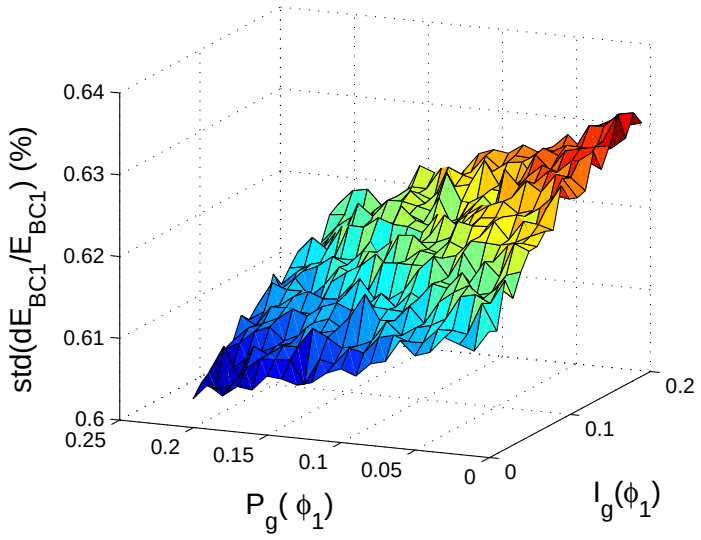



Figure 5.14: Standard deviation of the relative energy deviation $std(dE_{BC1}/E_{BC1})$ at BC1 for FERMI as a function of the gains $P_g(\phi_1)$ and $I_g(\phi_1)$ (for a 0.5 Hz jitter). Integral gains of 0.5 were used on the voltages of Linac 1 and Linac 4. The use of an integral gain increases the amplitude of the deviation, whereas the proportional gain reduces it. The addition of an integral correction to the proportional correction produces an increase in the residual energy perturbation at the bunch compressor.

However, Fig. 5.15 shows the opposite trend for the energy at the end of the accelerator. As was observed in the study of the integral gain, there is an interplay between the gains that causes the energy deviation at the end of the accelerator to be further reduced when large integral gains are applied. This leads us to conclude that gains optimising the reduction of the peak current jitter will not be the same as those optimising the reduction of the energy jitter at the end of the machine. Consequently, it is difficult to arrive at general rules regarding the optimum values of the gains.

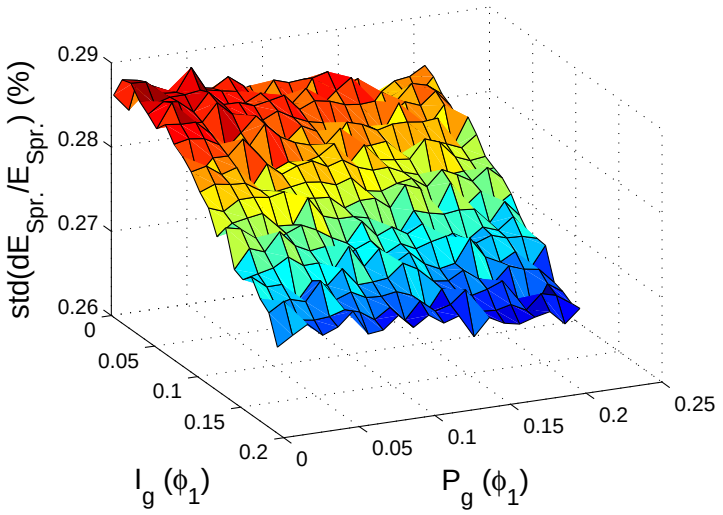



Figure 5.15: Standard deviation of the relative energy deviation $std(dE_{S\,pr.}/E_{S\,pr.})$ at the end of the FERMI accelerator as a function of gains $P_g(\phi_1)$ and $I_g(\phi_1)$ (with a 0.5 Hz jitter). Integral gains of 0.5 were used on the voltages of Linac 1 and Linac 4. Unlike the energy deviation after the bunch compressor, the use of an integral gain decreases the amplitude of the deviation, whereas the proportional gain has little effect. Better jitter reduction is achieved at the end of the Linac if only the integral gain is used.

#### Scans of differential and integral gains

Scans were performed in order to investigate the effect of the differential gain $D_g$ combined with the integral gain $I_g$. The differential gain tends to make the system highly sensitive to instabilities and amplify jitter. No useful information could be extracted from these scans, which would allow us to determine the required differential gains for attenuating the perturbation. Moreover, considering the difficulty in tuning the integral and proportional gains (see Sec. 5.5), we will not pursue further study of the differential gain.

### 5.5.2 Bode plots

The feedback performance can be expressed in terms of bandwidth. For this, a Bode plot is used to express the attenuation in the energy and peak current jitter. The attenuation factor $\eta$ is defined by:

$$\eta = 20log_{10}\left( \frac{(dE_{S\,pr.}/E_{S\,pr.})_{on}}{(dE_{S\,pr.}/E_{S\,pr.})_{off}} \right), \tag{5.6}$$

where $(dE_{S\,pr.}/E_{S\,pr.})_{on}$ is the residual relative deviation in the energy at the end of the accelerator when the feedback is on and $(dE_{S\,pr.}/E_{S\,pr.})_{off}$ is the relative deviation when the feedback is off. A similar formula is used for the peak current. The Bode plot was recorded over 30 Hz to reveal a 10 Hz periodicity corresponding to the beam repetition rate. The PI gains used for this scan corresponded to those used to attenuate a 0.1 Hz jitter frequency in the Linac 1 voltage, i.e. $I_g(V_1) = 0.5$, $I_g(\phi_1) = 1.5$ and $I_g(V_4) = 0.8$.

The Bode plot shows a symmetry at 5 Hz, corresponding to the Nyquist frequency $f_{Nyquist}$ (see Fig. 5.16). When a sinusoid of frequency $f$ is sampled with frequency $f_s$, the resulting samples are indistinguishable from those of another sinusoid of frequency $f'(N) = |f - Nf_s|$, with $N \in \mathbb{Z}$. All signal frequencies with $N > 0$ will have the same effect on the beam as those with $f'(N = 0)$. Because of this effect, jitter with frequencies higher than the Nyquist frequency $(N > 0)$ cannot be distinguished from those with frequencies lower than the Nyquist frequency $(N = 0)$. Consequently, it is sufficient to design a PID controller whose frequency range extends up to $f_{Nyquist}$.

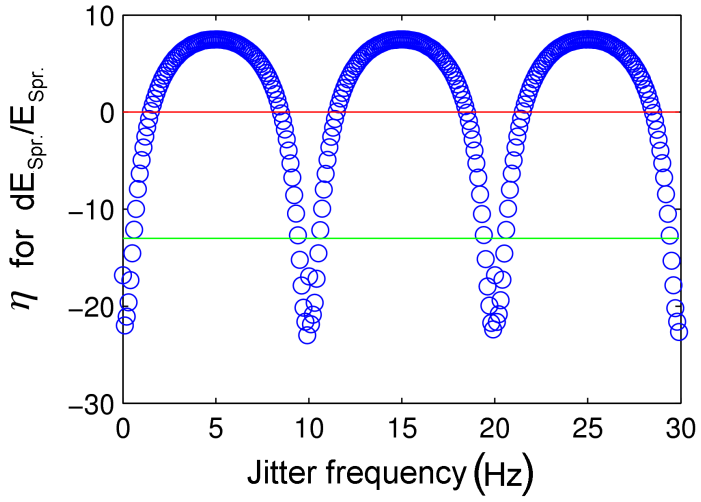



Figure 5.16: Bode plot of the relative energy deviation $dE_{S\,pr.}/E_{S\,pr.}$ at the end of the FERMI accelerator, obtained with the FMS code. The 10 Hz periodicity corresponds to the bunch repetition rate.

To ensure the lasing process can take place in the FERMI FEL, the energy stability is required to be $dE/E \leq 0.1\%$, with peak current stability of $dI/I \leq 10\%$ [2]. These criteria correspond to an attenuation factor $\eta$ of about -13 and -8, for the energy and the peak current, respectively. The red horizontal line in Fig. 5.16 represents zero attenuation ($\eta = 0$), whereas the green

line represents the minimum attenuation ($\eta = -13$) one has to achieve in order to meet the jitter attenuation requirements (the same criteria are used for the peak current in Fig. 5.17). Only the points lying below the green line in these figures are sufficiently attenuated. From these figures one can see that the correction is satisfactory only for a narrow window, i.e., energy jitter up to around 1 Hz and less than 0.5 Hz for the peak current. As discussed in Sec. 2.8, when feedback control is implemented[2], the missing information between the last pulse and the pulse that receives the correction leads to a residual deviation. The higher the jitter frequency, the higher the residual perturbation, since the perturbation can affect the beam more rapidly. Because of this, the PID algorithm is unable to attenuate the perturbation. Here this limit was approached around 1.5 Hz.

The Bode plot of the peak current shows a local minimum at the Nyquist frequency and another local minimum at half the Nyquist frequency (2.5 Hz). These features are due to the relation between the peak current and the frequency $f$ of the voltage jitter: $\Delta V(f) = V_0 - V_a \sin(2\pi f t)$, where $V_a$ is the jitter amplitude and $V_0$ is the unperturbed value of the voltage (with $V_a << V_0$). From Eq. (4.21), one can see that the energy deviation is directly proportional to the voltage deviation $dE/E(f) \sim \Delta V(f)$. However, the peak current is a function of the rms bunch length $I \propto 1/\sigma_z$ (see Eq. (4.32)), which is a function of $\Gamma_{RF} \propto \Delta V(f)$, i.e. the energy chirp induced by the RF curvature given by Eq (4.8). Based on Eq. (4.31) the rms bunch length has the form $\sigma_z = \sqrt{a_0 + a_1\Gamma_{RF} + a_2\Gamma_{RF}^2}$, where $a_0, a_1$ and $a_2$ are constants introduced to simplify the form of Eq. (4.31). The peak current is a function of the voltage deviation, $I(f) \propto 1/\sqrt{a_0 + a_1\Delta V(f) + a_2\Delta V^2(f)}$, which relation explains the more complex shape of the Bode plot for the peak current (Fig. 5.17).

## 5.6 Discussion

Our simulations have shown that optimised gains for the PID controller depend on the frequency of the system jitter and the settings of the machine parameters. These dependencies, in combination with the interplay of the gains themselves, makes it difficult to draw general conclusions for designing a PID controller. Moreover, the FMS code is not reliable for accurately simulating the machine, whereas Elegant is too time consuming. For example, the study with Elegant in Sec. 5.5.1 took one month (one data point taking about six hours), for the study of a limited range of integral gains and for the case of fixed jitter frequencies. Scanning the set of integral gains used

[2]Unlike the feedforward approach, the feedback correction uses past values of the deviations in the beam parameters and does not anticipate the deviation.

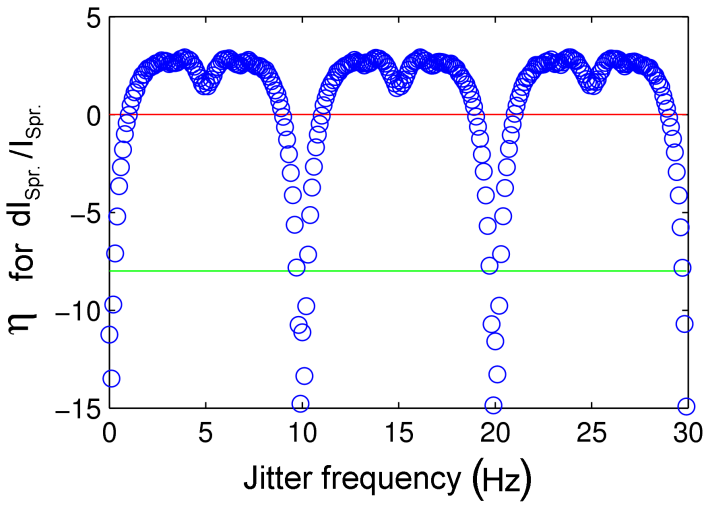



Figure 5.17: Bode plot of the relative peak current deviation $dI_{S\,pr.}/I_{S\,pr.}$ at the end of the FERMI accelerator, obtained with the FMS code. The 10 Hz periodicity corresponds to the bunch repetition rate.

in Sec. 5.5.1, for the frequency range 0 Hz - 5 Hz in steps of 0.1 Hz would take about 4 years. Further, any change in the machine parameters from the design settings would invalidate the results. Elegant is thus not suitable for the determination of the PID gains.

In addition to the difficulty in tuning the PID gain, the control system had a limited bandwidth and was unsuitable for frequencies above 1 Hz or 2 Hz. These limitations call for a more sophisticated approach to beam-based control. In the following chapters we address these deficiencies by considering more complex control systems that do not have bandwidth restrictions, are able to correct high frequencies, and can adapt their behavior when jitter conditions change, or when the machine settings are modified.

The limited measurement precision of the diagnostics (e.g., BPM and CSR detectors) was not taken into account in these studies. Unfortunately, the diagnostic characteristics were not available when the work reported in this chapter was carried out. Nevertheless, including the measurement precision of the diagnostics would be of interest, allowing us to evaluate its potential impact on the performance of the control algorithm, which will eventually be implemented.

# 6 An overview of neural networks

## 6.1 General considerations

As it was discussed in Chapter 5, the limitations of a simple PID controller, in combination with the more demanding requirements on beam stability in new generation Light Sources calls for more sophisticated control strategies. Moreover, as discussed in Chapter 1, NNets have been widely adopted in many domains of science and industry, including applications in control [17, 18]. However, to the best of our knowledge, NNets have not been successfully applied to the implementation of a beam-based control system, which is capable of adapting to the dynamics of the system through time and learning from its interaction with the accelerator.

The present chapter describes how NNets can be used for accelerator control. Section 6.2 gives a description of the structural components of NNets. Section 6.3 focuses on the principles used to evolve the structure of these systems, and in particular a method known as the Neuro-Evolution of Augmenting Topologies, developed by Stanley *et al.* [86]. Finally, in Sec. 6.5 we provide information on the integration of available information into the NNet structure.

## 6.2 Neural network architecture and learning techniques

An artificial NNet consists of an interconnected group of artificial neurons, based on a mathematical model of biological neurons. In most cases it is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. A schematic diagram of an artificial neuron is shown in Figure 6.1. Just as a biological neuron receives signals from the dendrites of other neurons, the artificial neuron (or node) receives stimuli from other nodes. Each of

these inputs to a node has a "weight" $w$ associated with it. A node has an activation function, which, like the threshold potential of a biological neuron, tells a node when to fire. A neuron may also have a "bias" value $\theta$; the weighted inputs and any bias is passed through the activation function, with the resulting value made available as the node output.

The type of activation function that is used depends on the application. For example, the linear function $f(x) = x$ may be used to find a linear approximation to a non-linear function, while the step function $f(x) = sign(x)$, may be used to classify input vectors by dividing the input space into two regions [87]. The hyperbolic tangent, $f(x) = tanh(x)$, and the sigmoid, $f(x) = 1/(1 + e^x)$, are common choices in modeling and control, especially when it comes to non-linear applications [16]. Radial Basis Function (RBF) Networks, $f(\mathbf{x}) = f(\|\mathbf{x} - \mathbf{c}\|)$, where $\mathbf{c}$ is the coordinate vector of the centre of the radial[1] function, are often used for modeling [88] or forecasting [89, 90]. An example of a radial basis function is the Gaussian, $f(\mathbf{x}) = e^{\|\mathbf{x}-\mathbf{c}\|^2}$. Figure 6.2 shows plots of the activation functions and corresponding mathematical expressions for each of the functions discussed above. A symbolic representation of a node with the associated activation function is also shown below each corresponding equation. These symbolic representations will be used throughout this thesis to specify a node's activation function in NNet diagrams.

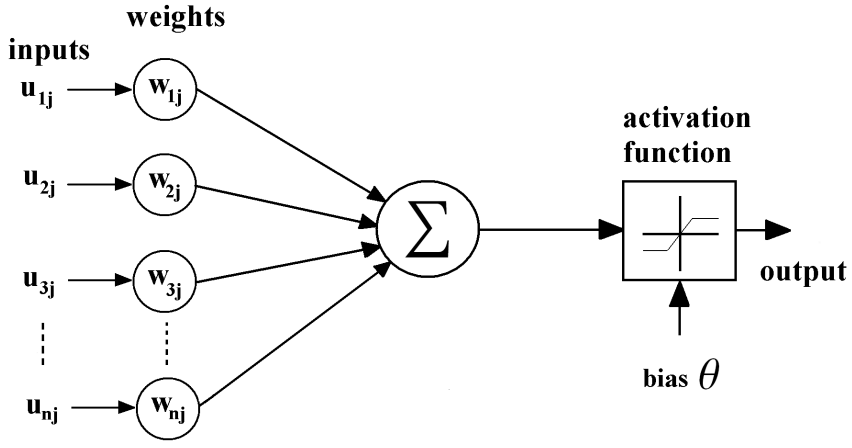



Figure 6.1: Schematic of an artificial neuron. The $j^{th}$ node in a given layer receives $n$ inputs $u_{1j}, u_{2j}, ..., u_{nj}$ from other nodes, which are multiplied by their respective weights $w_{1j}, w_{2j}, ..., w_{nj}$ and fed into the activation function with a bias $\theta$. The result is the node's output. Figure adapted from [63].

[1] By definition, the value of a radial basis function only depends on the radial distance from the centre $\mathbf{c}$, so that $\rho(\mathbf{x}, \mathbf{c}) = \rho(\|\mathbf{x} - \mathbf{c}\|)$ [16].

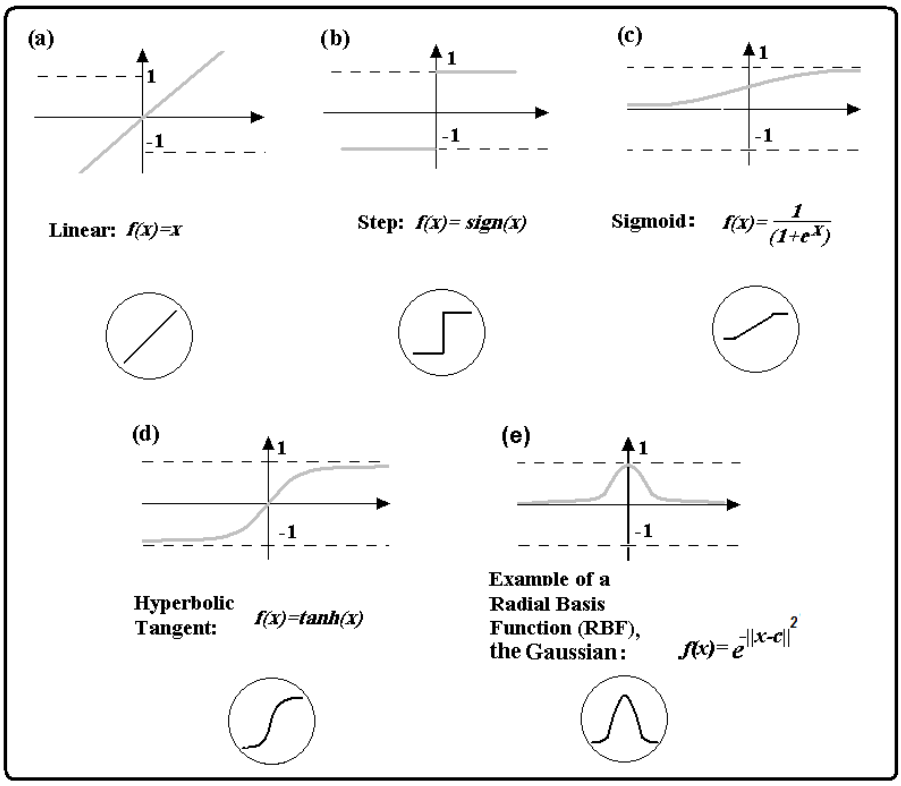



Figure 6.2: Commonly used activation functions. (a) The ramp, (b) the step function, (c) the sigmoid,(d) the hyperbolic tangent, and (e) a radial basis function. A node is represented schematically by a circle, which depicts the corresponding transfer function.

In an artificial NNet, neurons are arranged in input, output and hidden layers as shown in Fig. 6.3. The nodes in the input layer are passive since they do not modify the data; their only function is to relay the values from the single input to multiple outputs. In comparison, the nodes of the hidden and output layers are active as they modify the input data. Hidden layers are used to introduce additional non-linearities to the network's behavior; one hidden layer is enough to approximate any function with an arbitrary level of precision, given an appropriate number of nodes in the hidden layer [16].

NNets are classified as either feedforward or recurrent. In the first case, the information moves in only one direction (forward), from the input nodes, through the hidden nodes (if any) to the output nodes. There are no cycles or loops in the network. While a feedforward network propagates data linearly from input to output, recurrent networks (RN) propagate data from later processing stages to earlier stages.

Once one has decided on a network architecture, a training algorithm is chosen. Learning techniques can be classified into three categories [91]; namely, (i) supervised learning, (ii) reinforcement learning, and (iii) self organisation or unsupervised learning.

In the first case, sets of inputs and outputs are presented to the network, i.e. the network receives an input vector and produces an output vector. This output vector is compared to the desired output vector and the network weights are then adjusted by the learning algorithm. The process is repeated for the entire set of input and output training vectors.

In supervised learning it is assumed that there is a target output value for each input value. However, in many situations there is less detailed information available. Reinforcement learning is a particular case of supervised learning, where the network is trained by receiving appreciations[2] rather than target values. Supervised and reinforcement learning methods will be discussed in more detail in Secs. 6.2.1 and 6.2.3; these methods will be used in Chapters 7 and 8 to perform control and optimisation experiments.

Unsupervised learning is also referred to as self-organisation, in the sense that it self-organises data presented to the network to uncover correlations. Unlike supervised training, which uses the output vector, the input vector is compared with the weight vectors. The node with a weight vector most closely matching the input vector is called the winning node. The weights of the winning node and those of nodes within a certain neighborhood are then updated, such that the next time a similar vector is presented to the network, the node is more likely to win; conversely it is less likely to win with a different input vector. After training, each input vector of the training set will only activate a distinct output node. This type of training is often used for classification or recognition problems [93].

NNets are best suited to problems where there is no, or only a very complex analytical solution, but for which a significant amount of data is available. They are sensitive to statistical regularities in the input data and can therefore derive implicit relationships within the data (i.e. correlations between inputs and outputs). The fact that they are parallel processing elements increases the computation speed. Further, the distribution of memory elements makes them fault tolerant in a hardware implementation [94].

### 6.2.1 The back propagation algorithm

A number of supervised training algorithms have been developed, e.g., the Delta rule [95], Widrow-Hoff algorithm [96] and the backpropagation algo-

[2] An appreciation can take the form of a number between 0 and 1, which tells the NNet how well it performed, but without any information on the correct output [92].

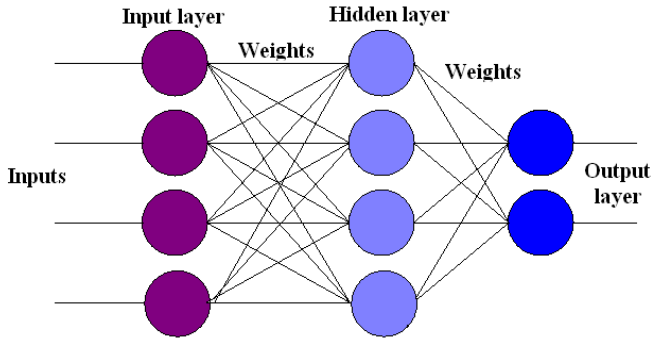



Figure 6.3: The three layers of a NNet: inputs, outputs and intermediate layers hidden between the two layers. Figure adapted from [63].

rithm. Here we describe the backpropagation algorithm used for training in our control and optimisation experiments. This algorithm also forms the basis for more advanced algorithms using the gradient descent technique [96].

As depicted in Fig. 6.1, the $j^{th}$ artificial neuron in a given layer receives an input vector[3] $\mathbf{u}_j = [u_{1j}, u_{2j}, ..., u_{nj}]$ from $n$ other nodes in the network. Each of these inputs has a "weight" associated with it, from which we form the weight vector $\mathbf{w}_j = [w_{1j}, w_{2j}, ..., w_{nj}]$. The weighted sum at the $j^{th}$ neuron is:

$$\mathbf{z}_j = \mathbf{w}_j \cdot \mathbf{u}_j, \tag{6.1}$$

and the output is:

$$\mathbf{y_j} = f_j(\mathbf{z_j}), \tag{6.2}$$

where $f_j$ is the activation function of the node. For the $j^{th}$ node in the output layer, the sum-squared error at the output is defined by:

$$E = \frac{1}{2} \sum_{o \in Outputs} (y_o - \hat{y}_o)^2, \tag{6.3}$$

where $y_o$ is the output produced by the network at the $o^{th}$ node of the output layer and $\hat{y_o}$ is the corresponding desired value (i.e. the corresponding value in the training set).

The change in weight $\Delta w_{ij}$ is calculated using the gradient descent technique [95]:

$$\Delta w_{ij} = -\mu \frac{\partial E}{\partial w_{ij}}, \tag{6.4}$$

[3]For example, if one is to model the energy of the electron beam at the end of a linear accelerator (which would correspond to the NNet's output), relevant inputs would include, but are not limited to, the phase and voltage of the accelerating sections.

where $\mu$ is a positive constant representing the learning rate. For the output layer one can write (Note that the summation convention does not apply):

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}} = \frac{\partial E}{\partial z_j}u_{ij}. \tag{6.5}$$

For the other layers we can expand the error of the hidden node in terms of its posterior nodes using the chain rule:

$$\begin{aligned}\frac{\partial E}{\partial w_{ij}} &= \sum_{k\notin Output}\frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial f_j(z_k)}\frac{\partial f_j(z_k)}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}} \\ &= \sum_{k\notin Output}\frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial f_j(z_k)}\frac{\partial f_j(z_k)}{\partial z_j}u_{ij}.\end{aligned} \tag{6.6}$$

The weights of the hidden and output layers are then updated by using Eqs. (6.5) and (6.6) with $w_{ij} + \Delta w_{ij} \rightarrow w_{ij}$. The process is repeated for the whole training set until there is either convergence of the weights to a local minimum, or until the remaining error given by Eq. (6.3) falls below a specified limit [16]. At the start of training all the weights are initialised to small random numbers. With the gradient descent technique the accuracy with which a NNet can reproduce training data depends on the initial weights of the network. This is illustrated in Fig. 6.4, where the global minimum $E_g$ is reached for the residual error starting with a weight $w_2$, whereas only the local minimum $E_l$ is reached when one starts with a weight $w_1$. This effect can be compensated for, by allocating more neurons to the network. However, this tends to overfit the data, a problem that will be discussed in Chapter 7.

### 6.2.2 Training of RBF neural networks

The weights applied to the RBF function outputs, as they are passed to the summation layer, are determined by the training algorithm. However, the RBF training also determines the number of neurons in the hidden layer and the coordinates of the centre of each hidden neuron in the network. In practice, the centres are often chosen to be a random subset of the training data set, which leads to a network with poor performance or having a large number of hidden nodes. In what follows we describe two training methods that overcome this problem by training both the weights and centres of the NNet. The first method is called the Orthogonal Least Squares (OLS) method. With this method, nodes are added to the network until the sum-squared error falls below an error goal, or until a maximum number of nodes has been reached. The second method is based on the least squares algorithm and only trains the weights of the network.

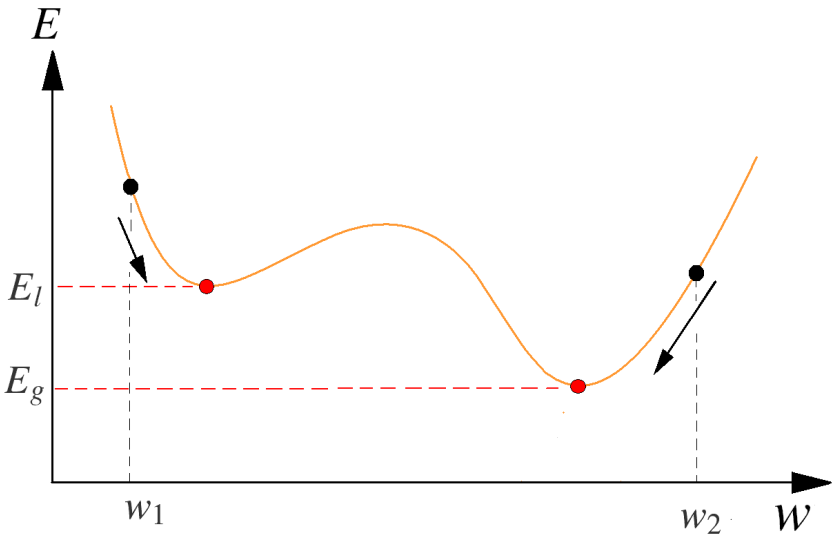



Figure 6.4: Local minimum in backpropagation training. During training with the gradient descent technique the accuracy attained by the NNet depends on the value of the initial weights of the network. Starting with an initial weight $w_2$, the global minimum $E_g$ of the sum-squared error is reached, whereas only the local minimum $E_l$ is found when starting with weight $w_1$. The arrows indicate the direction in which the weights are adjusted by the backpropagation algorithm.

#### Orthogonal least squares training

Let us consider the RBF network depicted in Fig. 6.5. The network receives $J_1$ inputs forming the vector $\mathbf{x} = [x_1, x_2, ..., x_{J_1}]$ and has $J_3$ outputs forming the vector $\mathbf{y} = [y_1, y_2, ...., y_{J_3}]$. It has $J_2$ hidden nodes, with radial basis functions $\phi_k(\mathbf{x}) = \phi(\mathbf{x} - \mathbf{c}_k)$, $(k = 1, 2, ..., J_2)$, forming the vector $\mathbf{\Phi} = [\phi_1, \phi_2, ..., \phi_{J_2}]$. The weights connecting the hidden layer to the output layer are given by the matrix $\mathbf{W}$, where the element $w_{ki}$ connects the $k^{th}$ hidden node to the $i^{th}$ output node. With these definitions, the $i^{th}$ output of the RBF network is given by:

$$y_i(\mathbf{x}) = \sum_{k=1}^{J_2} w_{ki}\phi(\|\mathbf{x} - \mathbf{c}_k\|). \tag{6.7}$$

As with the backpropagation algorithm, the Orthogonal Least Squares (OLS) methods aims to minimise the error $E$ between the NNet's output and the outputs of the training set, where

$$E = \frac{1}{N}\|\mathbf{Y} - \mathbf{W}^T\mathbf{\Phi}\|_F^2, \tag{6.8}$$

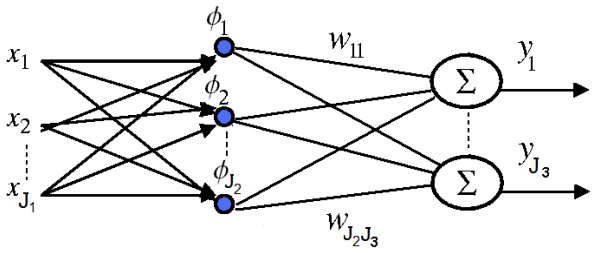



Figure 6.5: Schematic of a radial basis function neuron. The NNet contains $J_1$ input nodes, $J_2$ hidden nodes and $J_3$ output nodes.

Here $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N]$, $N$ is the number of input/output samples in the training set and $\|\cdot\|_F$ is the Forbenius norm[4]. When all the centres are distinct, $\mathbf{\Phi}^T$ can be decomposed as follows [97]:

$$\mathbf{\Phi}^T = \mathbf{Q}\begin{bmatrix} \mathbf{R} \\ ... \\ \mathbf{0} \end{bmatrix}, \tag{6.9}$$

where $\mathbf{Q}$ is an $N \times N$ orthogonal matrix and $\mathbf{R}$ is a $J_2 \times J_2$ upper triangular matrix. Because this matrix is orthogonal, Eq. (6.8) can be rewritten as:

$$E = \frac{1}{N}\|\mathbf{Q}^T\mathbf{Y}^T - \mathbf{Q}^T\mathbf{\Phi}^T\mathbf{W}\|_F^2. \tag{6.10}$$

Now let rewrite the first term in the right hand side of Eq. (6.10) as:

$$\mathbf{Q}^T\mathbf{Y}^T = \begin{bmatrix} \tilde{\mathbf{B}} \\ ... \\ \bar{\mathbf{B}} \end{bmatrix}, \tag{6.11}$$

where $\tilde{B}$ and $\bar{B}$ are $J_2 \times J_3$ and $(N - J_2) \times J_3$ matrices, with elements $\tilde{b}_{ij}$ and $\bar{b}_{ij}$, respectively. With this, Eq. (6.10) becomes:

$$E = \frac{1}{N}\left\|\begin{bmatrix} \tilde{\mathbf{B}} - \mathbf{RW} \\ ... \\ \bar{\mathbf{B}} \end{bmatrix}\right\|_F^2. \tag{6.12}$$

From this equation one can see that the weights of the system that minimise the error $E$ are such that $\mathbf{RW} = \tilde{\mathbf{B}}$. The OLS algorithm adds one neuron at a time to determine the weights and centres of the NNet. The centres are

[4]The Frobenius norm of a vector $\mathbf{H}$ is defined as $\|\mathbf{H}\|_F^2 = tr(\mathbf{H}^T\mathbf{H})$, where $tr$ is the trace operation.

chosen from the training set. For this, we introduce the error reduction ratio for the $k^{th}$ neuron being implemented in the hidden layer [98]:

$$\epsilon_k = \frac{\left( \sum_{i=1}^{J_3} \tilde{b}_{ki}^2 \right) \mathbf{q}_T^k \mathbf{q}_k}{tr(\mathbf{Y}\mathbf{Y}^T)}. \tag{6.13}$$

Equation (6.13) is evaluated for each sample of the data set and the sample that results in the largest decrease in the error $\epsilon_k$ is selected. The criterion to stop the centre selection (and the addition of a new neuron) is a threshold $\rho \in (0,1)$ such that:

$$1 - \sum_{k=1}^{J_2} \epsilon_k < \rho. \tag{6.14}$$

Training stops when the error criterion (6.14) is satisfied (via incorporating neurons in the hidden layer). A drawback of the OLS algorithm is that it is computationally intensive for large data sets, scaling as $O(N, J_2^2)$; this is due to the complexity of the orthogonal decomposition. On the other hand, because hidden nodes are added one at a time, it avoids incorrectly sized networks, i.e. a NNet with too few neurons will show poor fitting properties, while a NNet that is too large will overfit the data.

#### Least squares training

Consider how the gradient descent technique can be used for the training. Unlike the OLS method, this method does not determine the number of neurons in the hidden layer, but trains the centres and weights of a pre-determined number of hidden nodes. We first write the error function as [99]:

$$E = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{J_3} (e_{n,i})^2, \tag{6.15}$$

where $e_{n,i}$ is the error at the $i^{th}$ node for the $n^{th}$ pair of input/output samples of the training set, and is given by:

$$e_{n,i} = y_{n,i} - \sum_{m=1}^{J_2} w_{mi} \phi(\|\mathbf{x}_n - \mathbf{c}_m\|). \tag{6.16}$$

Taking the first derivatives of $E$ with respect to $w_{mi}$ and $\mathbf{c}_m$, we obtain:

$$\frac{\partial E}{\partial w_{mi}} = -\frac{2}{N} \sum_{n=1}^{N} e_{n,i} \phi(\|\mathbf{x}_n - \mathbf{c}_m\|), \tag{6.17}$$

and

$$\frac{\partial E}{\partial \mathbf{c}_m} = \frac{2}{N}\sum_{i=1}^{J_3} w_{mi}\sum_{n=1}^{N} e_{n,i}\dot{\phi}(\|\mathbf{x}_n - \mathbf{c}_m\|)\frac{\mathbf{x}_n - \mathbf{c}_m}{\|\mathbf{x}_n - \mathbf{c}_m\|}. \tag{6.18}$$

Using Eqs. (6.17) and (6.20), the update equations for the weights and centres are thus given by:

$$\Delta w_{mi} = -\eta_1 \frac{\partial E}{\partial w_{mi}}, \tag{6.19}$$

and

$$\Delta \mathbf{c}_m = -\eta_2 \frac{\partial E}{\partial \mathbf{c}_m}, \tag{6.20}$$

where $\eta_1$ and $\eta_2$ are the learning rates. With this method, the centres and weights of the networks are selected randomly at first, and then updated until the error $E$ falls below a specified limit. Although this method is not as computationally demanding as the OLS method, the number of hidden neurons must be pre-determined.

In general, radial basis networks tend to have many more nodes than a comparable network constituted of sigmoidal or hyperbolic tangent neurons in the hidden layer. This is due to the fact that sigmoidal neurons can have outputs over a large region of the input space, while RBF neurons only respond to relatively small regions of the input space. As a result, the larger the number of inputs, the more RBF neurons are necessary [87].

### 6.2.3 Reinforcement learning

Reinforcement learning is used when data is not directly available for supervised training. In this case the NNet is connected to its environment via perception and action. At each step $k$ of a time series, the network receives a set of inputs that define its state $T_k \in T$ in the environment, where $T$ represent the set of possible states. It then chooses an action $a_k \in A$ to produce an output, leading to a new state $T_{k+1}$ ($A$ is the set of possible actions). The state transition is then evaluated and its value communicated to the NNet through a reinforcement signal. This signal is a value that we denote by $Rw_k$, and which rewards or penalises the NNet for its actions. A reward is defined by $Rw_k > 0$, and a penalty by $Rw_k < 0$. The value of $Rw_k$ is often an integer, linked to the outcome of the action. For example, in a video game, an agent would receive a positive signal $Rw_k = 1$ for shooting an enemy and a negative signal $Rw_k = -2$ for being shot itself. The NNet is not told at any stage what the ideal action would have been. Based on these interactions, the

network must develop a policy[5], $\pi : T \rightarrow A$ that maximises the total reward $R_{tot} = \sum_k Rw_k$ in the long term. In order to develop the optimum policy, it is necessary for the network to gather experience about the possible system states, actions, transitions and rewards.

This can be illustrated with the following simple example. Consider a video game, where an game agent must shoot its enemies while avoiding being shot itself. The actions of the agent can be directed by a NNet receiving sensor inputs, which provide information on the proximity of obstacles and enemies. These inputs define the state $T_k$ of the network in its environment. The agent can then take an action $a_k$, such as moving forward, turning or shooting. After the network has taken an action, it receives a reward or penalty $Rw_k$, according to the outcome of its action. For example, it will receive positive points for having shot an enemy, or negative points for not moving when a sensor indicated an enemy was firing in the agent's direction.

This concept is of particular interest for control purposes in an operational accelerator environment. Indeed, let's assume that we wish to develop a control system to stabilise the beam energy using a NNet to act on the phase of the klystrons. If supervised training is used and a shift occurs in the absolute phase of an accelerating cavity, the system would require re-training (i.e. re-adjustment of the NNet's weights). This would require the detection of the phase shift and the gathering of data for new training. The amount of data required for training increases exponentially[6] with the number of cavities, becoming a highly time consuming task. Instead, it would be highly desirable to build a system that develops policies to adapt its behavior through time. In Chapter 8 we will discuss the implementation of such a system for optimisation purposes and in Chapter 9 we discuss its adaptation to control applications.

In what follows we build and train NNets using the Matlab programming language and its neural network toolbox. This choice is based on the use of Matlab for communication with the machine actuators and diagnostics at the AS, the LCLS and FERMI.

[5] In the context of a neural network a policy is a map that defines the action $a_k$ that should be taken at every step, given the state $T_k$ [100].

[6] To obtain training data for one phase $\phi$, an interval of interest $[\phi_a, \phi_b]$ is divided into $n_L$ segments of identical lengths, where $(n_L + 1)$ corresponds to the number of data points required for training. For two phases $\phi_1$ and $\phi_2$, if the number of segments are identical (i.e. $(n_{L1} + 1) = (n_{L2} + 1)$), the number of data point is $(n_{L1} + 1)(n_{L2} + 1) = (n_L + 1)^2$. Similarly, for $N$ phases, the number of training data points is $(n_L + 1)^N$.

## 6.3 Neuro-Evolution of Augmenting Topologies (NEAT)

In practical control or optimisation applications, environmental conditions are subject to change through time. For example, a robot might have a faulty sensor connection or an airplane may loose an engine and there is no opportunity to re-train the controller. To perform optimally in such situations, adaptation of the controller is necessary.

In nature, organisms encounter new situations frequently and learn from them by changing their policies. This is possible because their nervous system is not static. Instead, new connections can arise and some connections can become more or less stimulated depending on the environments. This motivates us to develop an evolving NNet that can adapt its structure through learning from interactions with the environment. Promising advances have been made in this domain by Stanley *et al.* [101]. This approach, called Neuro-Evolution of Augmenting Topologies (NEAT), combines the use of genetic algorithms to encode and evolve the NNets' structures with reinforcement learning to select the most promising elements amongst a population of NNets. In a similar way to natural organisms, the artificial NNet can therefore adapt its structure through interacting with its environment. NEAT has already proven to be highly effective, outperforming other neuro-evolution methods [102]. The development of NEAT has been motivated by current limitations of contemporary video games, where game agents show a high level of graphical realism and in their actions, but lack the ability to adapt to new situations. Their actions quickly become predictable and this results in a loss of interest from the player . The method has been used successfully in the NERO[7] video game [101, 103, 104, 105] and the roving eye for Go[8] [106].

NEAT is based on the three following key ideas. First, the network structure must be genetically encoded in an efficient way [86]. This is described in Sec. 6.3.1, which also discusses a method used to keep track of each gene during evolution, called "historical marking". Second, because in a pop-

[7] NERO is a new style of game that requires the training of intelligent armies. At the start of the game, each player trains his own robots to form a team for combat. The agents thus created gain their fighting techniques from the guidance of the player and not from a pre-programmed algorithm [101].

[8] Go is an ancient two-player Chinese board game where black and white pieces are placed alternately at intersection points on the grid by two players. The purpose of the game is to control more territory than the opponent. Any region of the board totally surrounded by one player's pieces is counted as that player's territory. Although Go has very simple rules it is very difficult to master and it is a popular and challenging testbed for different AI techniques.

ulation of NNets some individuals might take more time to optimise their structure, it is important to protect them from being eliminated too early during evolution, before they can be competitive with the population at large. This is addressed by protecting innovation through speciation as is explained in Sec. 6.3.2. Finally, NEAT starts the evolution process from a minimum NNet architecture[9], only increasing the complexity of the structure when necessary. The initial population consists of simple networks with no hidden nodes, which become more complex through evolution to create increasingly sophisticated behaviors. The fittest networks of a population are selected using a fitness function, which is based on reinforcement learning. This is described in Sec.6.3.3.

### 6.3.1 Genetic encoding and historical marking

The structure of a NNet must be encoded in an efficient way. For this, NEAT encodes the nodes and their connections separately. The node's genes identify the node and its type (input, output, bias or hidden). The connection genes must include information on the start and end nodes of the connection, its associated weight and whether the connection is activated or disabled [107]. Figure 6.6 shows an example of encoding for a network comprised of three inputs, one output and one hidden neuron.

During evolution, mutations can occur in both the weights and the structure of the network. Structural changes act on nodes and connections. A connection can be created or inactivated. When a new node appears, it is introduced where there is an existing connection. That connection is then split into two new connections in order to integrate the new node into the structure. To minimise the non-linearity introduced by the addition of the new node, the weight of the input connection is set to 1 and the output connection is assigned the weight of the old connection. In this manner the functionality of the newly created network is only slightly modified [107].

Evolution takes place in the following way. For each generation, the performance of each network in the population is assessed by evaluating the fitness function (discussed in Sec. 6.3.3). The fittest individuals are then selected to generate the next generation by crossing over. When this occurs, genes of both parents are compared and those that do not match are either disjoint or in excess, depending on whether they occur within the reach[10]

[9] The architecture includes not only the NNet's weights but also the number of nodes in the hidden layer(s) and the connections. In NEAT, the first generation of the NNet consists of input and output nodes only (i.e. no hidden nodes).

[10] The reach of one parent is defined as the highest innovation number possessed by that parent. A gene is said to be in excess when its innovation number is higher than the other parent's highest innovation number, otherwise it is disjoint.

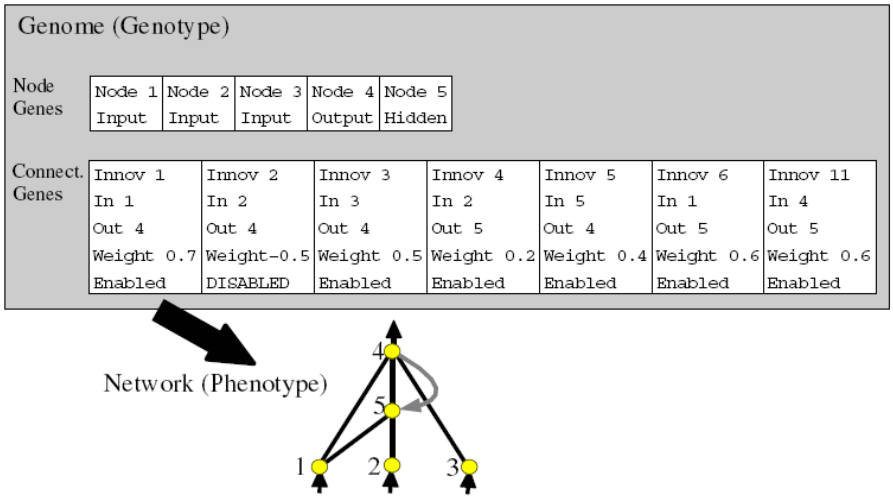



Figure 6.6: Example of a NNet structure encoded with NEAT. The node genes include the node identity number and its corresponding type (input, output, bias or hidden). The connection genes identify the in and out nodes for this connection, its weight, and specifies whether the connection is active. A unique innovation number is also attributed to each gene in order to track its introduction through evolution [108]. Figure taken from [109].

of the other parent (see Fig. 6.7). Genes that are present in both parents will be randomly selected from both parents to form the offspring. Genes that are in excess or disjoint will be selected from the fittest parent, or randomly from both parents if they share the same fitness. Each time a new gene appears it is assigned a global "innovation number" that allows for the easy identification of matched and unmatched genes during cross-over (i.e. when two parents are selected and their genes used to create an offspring). This mechanism is called "historical marking" and avoids having a connection copied multiple times [109].

### 6.3.2 Protection of innovation

Because structures can take time to be optimised, NEAT divides the population into species. In that way, networks will be competing against individuals of the same species before competing with the population at large. For this, individuals are re-assigned to a species on the basis of their architectural similarities (number of hidden nodes, connections and weights) at each generation. A topological distance is used to quantify the structural differences between two individuals. This distance is calculated as a linear combination of the number of in excess and disjoint genes ($E$ and $D$, respectively) and the average weight differences for the matching genes ($\bar{W}$) [86]:

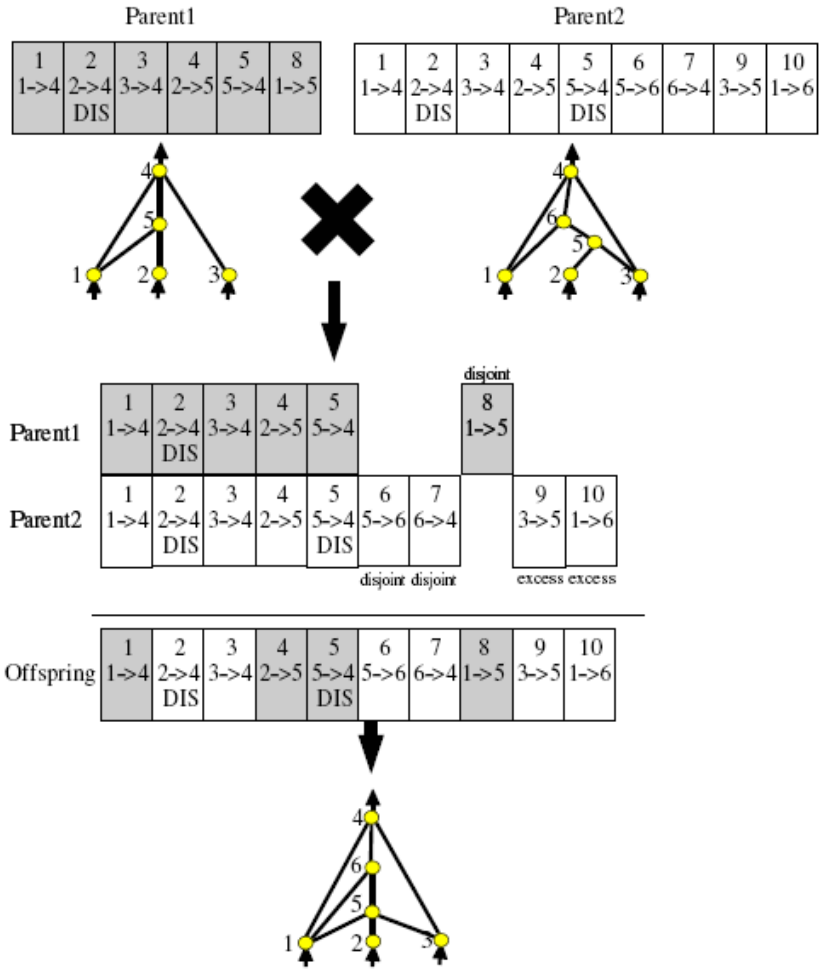



Figure 6.7: Cross-over with NEAT. The genes of both parents are aligned and matching genes are randomly selected from both parents to form the offspring. Disjoint or in excess genes are chosen from the most fit parent, or randomly chosen from both parents if their fitness is identical. Figure taken from [109].

$$\delta = \frac{c_1 D}{N} + \frac{c_2 E}{N} + c_3 \bar{W}, \tag{6.21}$$

where $N$ is the number of genes of the larger genome (between the two parents). The three constants $c_1$, $c_2$ and $c_3$ are used to adjust the importance of the different contributions. At the start of each generation, networks are evaluated one after another and placed in the first species for which $\delta < \delta_t$, where $\delta_t$ is a threshold value that defines how similar two networks must be in order to be considered the same species. If the network does not match

an already existing species a new species is created. This first network will then be the "reference" network from which the distance $\delta$ of other networks is calculated according to Eq. (6.21). Moreover, each species is allowed to have a maximum number of offspring $n_k$, based on their average fitness. This number is given by:

$$n_k = \frac{\bar{F}_k}{\bar{F}_{tot}}|P|, \tag{6.22}$$

where $\bar{F}_k$ is the average fitness of individuals in species $k$, $|P|$ is the size of the population, and $\bar{F}_{tot} = \sum_k \bar{F}_k$ is the sum of the average fitnesses of all the species. This prevents the number of individuals in the population from continuously increasing through evolution. At each generation, the best individuals are then selected to create new individuals, and after cross-over the entire population is replaced by the new offspring.

### 6.3.3 The fitness function

During evolution, an individual's fitness is assessed according to how well it has performed on a given task. This fitness is evaluated by a fitness function, which serves to select and evolve the most promising networks of a population through successive generations. The performance is evaluated on the basis of reward and penalty points that are accumulated by an individual during the evaluation phase; this is based on the principle of reinforcement training discussed in Sec. 6.2.3.

Because evolution aims to increase an individual's fitness, it is very important to carefully define the notion of fitness. It is necessary to have a clear idea of what succeeding in a task means, be able to quantify it, and accordingly reward or penalise actions taken by a network during its evaluation. This is discussed in more detail in Chapter 8, where a fitness function is constructed to evolve a NNet that has to navigate within a search space without crossing pre-set boundaries.

## 6.4 Real-time Neuro-Evolution of Augmenting Topologies (rtNEAT)

As described above, NEAT is designed to be run offline. This means that all the individuals of a population have to be evaluated before a new generation of neural networks can be generated. This method is not suited to a video game involving many game agents. Indeed, this implies that all the agents would be replaced at the same time, which would look incongruous to the player and precludes interaction with the agents as they evolve [103].

To overcome these limitations, the real-time adaptation of NEAT replaces an individual in the game every few game ticks (interval of time in the game) with an offspring created from the fittest individuals. The real-time adaptation of NEAT (i.e. rtNEAT) includes the following refinements. First, the fitness $f_i$ of an individual must be adjusted to the number of individuals $|S|$ in its species. We define the "sharing" fitness as [104]:

$$f_{s,i} = \frac{f_i}{|S|}. \tag{6.23}$$

This adjustment of the fitness is necessary to keep the same dynamics as in NEAT. Indeed, if the size of the species was not taken into account, innovation would no longer be protected and new species would be removed as soon as they appear. After a given number of game ticks the agent with the worst shared fitness is removed from the game. However, it is important not to remove agents that are too young, and for this reason only agents who have been playing for $m$ game ticks will be chosen. To evaluate the appropriate number of game ticks between replacements we introduce the fraction of agents $I$ too young to be replaced. This is expressed as a fraction of the population that is ineligible after the evolution has reached a steady state:

$$I = \frac{m}{|P|n}. \tag{6.24}$$

From Eq. (6.24) we can determine the number of game ticks before replacement [103]:

$$n = \frac{m}{|P|I}. \tag{6.25}$$

For example, in the NERO video game $I$ is typically chosen to be 50%. Before a new agent can enter the game it is necessary to perform a few operations in order to preserve the dynamics of NEAT. First, the average fitness $\bar{F}_k$ of individuals in species $k$ must be recalculated when the worst agent is removed. This is important because $\bar{F}_k$ is used in the next step to chose the parent species for the new individual. Next, one must chose the parent species. Previously this was done by determining the number $n_k$ of offspring that a species could generate (c.f. Eq. (6.22)). In our case, since only one individual is generated when a replacement occurs, we define the probability $Pr(S_k)$ of choosing a species $S_k$ as [103]:

$$Pr(S_k) = \frac{\bar{F}_k}{\bar{F}_{tot}}. \tag{6.26}$$

In rtNEAT the number of species is kept relatively stable by adjusting a threshold $C_t$ that determines the compatibility of an individual with a

species. If too many species exist, the threshold can be increased so that more individuals belong to the same species. Similarly, if there are too few species, the threshold $C_t$ is decreased. When $C_t$ is modified the entire population must be reassigned to the different species.

## 6.5 Knowledge integration into the network's structure

As discussed earlier, NEAT starts its search from a minimal NNet structure. If the task is complex, evolving a NNet with a structure that can accomplish that task can take considerable time. For example, evolving a NNet that implements the binary operation XOR[11] takes a few tens of generations, while evolving a NNet structure that can navigate in 2D space without crossing fixed boundaries can take up to a few hundred generations (this structure will be discussed in Chapter 8). Moreover, it is sometimes possible to decompose a problem into a series of actions that should be taken by the NNet in order to carry out a task correctly under specified conditions [110]. Whenever possible, it is desirable to integrate functional structures into the minimal NNet that NEAT will evolve. Even if only partial knowledge is available it will provide a basis for evolution and can save considerable time [111]. There are two basic structures we will highlight here, as they are the most relevant to the analysis in Chapter 8.

The first structure corresponds to the "if-then" rule. This structure is depicted in Fig. 6.8 (a). The first node receives the conditional inputs and depending on the response of that node, Action 1 or Action 2 will be triggered. For this structure to work, the correct weights must be determined. To illustrate its use, consider the example of a video game agent that has the task of navigating through a battlefield. Let us also assume that the NNet is provided with an input that gives the distance of the agent to an obstacle located in front of it. If that obstacle is detected, the agent should turn in order to avoid it. This information will therefore lead to a conditional action of the type: "if there is an obstacle, then go forward and turn, otherwise go straight ". The integration of this structure into the NNet will be discussed in Chapter 8.

Similar reasoning can be applied to the "while-then-repeat" action. The corresponding structure is illustrated in Fig. 6.8 (b). In this case two nodes receive the "while" and "until" conditions. The when-condition node is directly connected to the repetition-rule node, whereas a negation node

[11] The XOR binary operation yields 1 if exactly one of the two binary inputs is 1, and 0 otherwise.

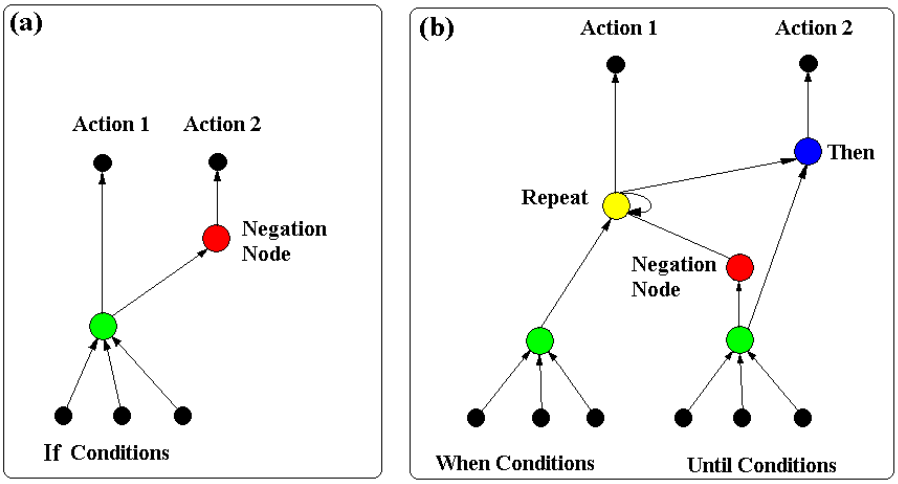



Figure 6.8: Building blocks used to construct knowledge-based structures. (a) The "if" structure is built using two hidden neurons (shown in green and red). The first hidden neuron (in green) receives the conditions. Depending on the activation of this neuron Action 1 will be triggered or Action 2 by the application of a negation node (in red). (b) The "while" structure uses five hidden neurons (two shown in green, one in red, one in yellow and one in blue). The two neurons of the first hidden layer (in green) receive the "when" and "until" conditions. When both conditions are true Action 1 is activated and the last repeat state is recorded by the recurrent connection (on the yellow node). Otherwise Action 2 is activated because the until condition is "true", which means the repeat action is inactivated, or the "when conditions" are no longer valid [111].

is used to connect the until-condition node to maintain the repeat-action node. The structure that connects the right green node to the red, blue and yellow nodes is similar to the "if-then" structure described above. Indeed, if the "until condition" is true, the "then" action rule will be activated. In this case, the negation gives a "false" input that will inhibit the repeat-action rule. If the until condition is "false" this means that the condition to stop is not met, and the negation node will provide an excitatory input to the repetition-rule node. This structure can be used, for example, to tell the game agent to keep moving forward until it encounters an enemy. It will be exploited in Chapter 9 for the optimisation of the transmission and energy spread of the electron beam, and for the control of the beam energy when a static perturbation is applied to the phase of an accelerating structure.

# 7 Building a NNet hybrid feedforward - feedback control system

## 7.1 Motivation and objectives

The work presented in Chapter 5 demonstrated the limitations of conventional PID controllers, including poor response for high jitter frequencies and limited bandwidth. The present chapter considers the development of a feedforward-feedback type of controller in order rectify these deficiencies, while still ensuring robustness of control.

The chapter is organised as follows. Section 7.2 describes the proposed controller structure, while Sec. 7.3 presents the method used to define an appropriate topology for the NNet. Sections 7.4 and 7.5 show real-time control tests carried out at the Australian Synchrotron and the Linac Coherent Light Source.

## 7.2 The controller structure

The controller scheme is shown in Fig. 7.1. It consists of two distinct parts, a NNet predictor (feedforward part) and a conventional control algorithm (feedback part). The NNet is used to make a prediction for the perturbation of the next bunch (i.e. energy and/or bunch length) in order to anticipate corrective action, while the conventional controller ensures that the residual jitter (not corrected by the feedforward algorithm) is attenuated by a feedback action. Thus, if the feedforward should fail, the feedback will still ensure partial control. An important feature of a NNet lies in the fact that its response is bounded, due to the nature of the activation function. This is relevant to control systems because it ensures that the actuators cannot be driven beyond fixed limits.

As discussed in Chapter 4, the main source of jitter for the beam energy and bunch length comes from perturbations occurring in the phase and voltage of the accelerating structures. Records of these variables will be

used to compute future deviations in the beam parameters. For example, consider the case where a first klystron (klystron 1) is subject to phase and voltage jitter and a second klystron (klystron 2) is used to correct for the energy deviations induced by the first klystron. The delay operator $D$ in Fig. 7.1 provides the NNet predictor with $m$ delayed input values of the voltage jitter $dV_1$ and $n$ delayed input values of the phase jitter $d\phi_1$ of the first klystron ($dV_1$ and $d\phi_1$ are deviations from the set points of the voltage and phase of klystron 1). For the bunch number $k$ of a time series, the $p^{th}$ delayed phase value (or lag) is given by $D^{(-p)}(\phi(k)) = \phi(k-p)$, with $p$ ranging from 1 to $n$. The NNet output in Fig. 7.1 is a prediction of the deviation in the horizontal position $dx(k+1)$, measured at a BPM, and corresponds to a prediction in the energy deviation of the beam.

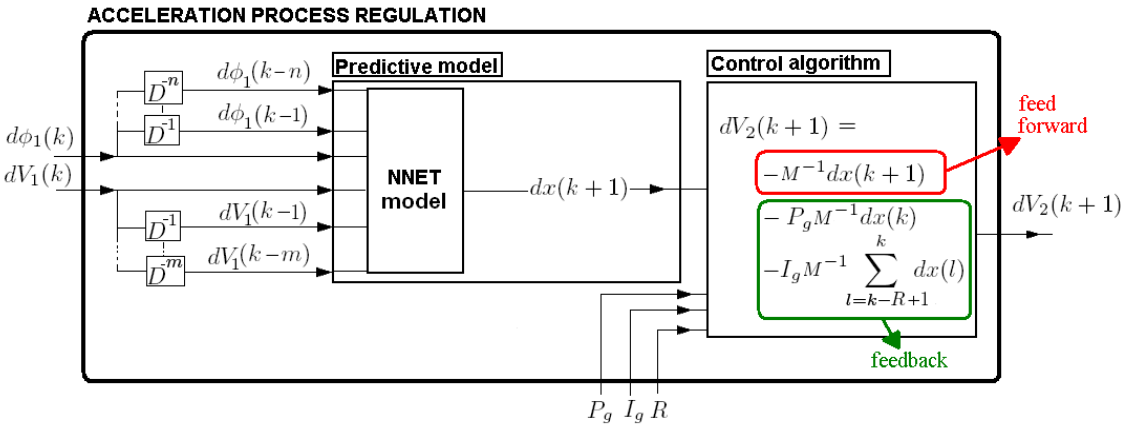



Figure 7.1: Feedforward-feedback control scheme consisting of a NNet predictor (left block), connected in series to a control algorithm (right block). The predictor receives $m$ lagged values of $dV_1$ and $n$ lagged values of $d\phi_1$ (the voltage and phase jitter of the first klystron). The predicted horizontal deviation $dx(k+1)$ in the beam position is used by the control algorithm in the right block to compute the forward correction (red block). The feedback correction (green block), with user specified gains ($P_g$ and $I_g$) and sum range ($R$) uses measurements of the deviations in the current and the past bunches. The sum range specifies the number of bunches taken into account in the integral correction. Figure adapted from [63].

The chosen control algorithm is based on a Proportional-Integral (PI) algorithm, where the gains $P_g$, $I_g$ and the number of elements $R$ in the sum of the integral term are chosen externally. The gain $D_g$ is not considered in this study because of the tuning difficulty and its tendency to make the system unstable. Moreover, this term is often used to compensate for higher frequency jitter, which is taken care of here by the NNet structure. The fac-

tor $M$ is the response coefficient of the deviation in the klystron voltage (in mm/kV). When the NNet provides predictions for multiple variables, it does so by providing a control proportional to the predicted variation. This is complemented by a feedback term to ensure stability. In Fig. 7.1 the feedback control is applied to the voltage (of klystron 2), but it could also be applied to its phase.

More generally, a network can be trained to directly operate the control, acting as the inverse of the system model. The network's outputs are then used directly for control in an additive feedforward path [112, 113].

## 7.3 Determining the NNet predictor structure

The success of the feedforward control is determined by the ability of the network to predict disturbances. In principle, an arbitrary level of precision can be achieved as long as the network has a sufficient number of hidden neurons [16]. The quality of the training set (i.e. the amount of data available and its relevance) will contribute as well. If the training is too long or the number of neurons too large, over fitting may result. Similarly, if the training set or number of neurons is insufficient, fitting will be poor [114]. One problem often associated with the use of a NNet is that there is no way to determine in advance an appropriate architecture and training method for a specific problem. In this section we describe the method used in order to ensure that the control network has an appropriate size and is adequately trained.

If records of the inputs to the accelerating process $u(t)$ (i.e. phase and voltage of the klystrons) and outputs $y(t)$ (i.e. beam energy and bunch length) are available, the network can be assigned a number of lagged values for the input variables, output variables or a combination of both as its input nodes. In the case where outputs are used, a model of the noise is often required, otherwise the resulting predictor might be biased [115, 116]. In such a case, the model would operate well on training data, but poorly on data that was not presented to the network during its training phase. In our case, since the bunch-to-bunch correlation (output history) would be much weaker than the input-output correlation, there would be no point to feed the network lagged outputs. Thus, it was decided not to include past outputs of the predictor model to the network.

In what follows, we describe the systematic approach that was adopted to define the number of lagged inputs (for each input variable) and hidden neurons. The number of hidden neurons will be denoted by $N_h$. To begin with, we introduce the error index (E.I.), which is a measure of the quality

of the prediction [115]:

$$E.I. = \left( \frac{\sum(\hat{y}(t)-y(t))^2}{\sum y^2(t)} \right)^{1/2}, \tag{7.1}$$

where $\hat{y}(t)$ is the model output and $y(t)$ is the target value. Alternatively, the number of lagged input and output values can be evaluated by using correlation tests. If only lagged inputs are used, the relevant correlation tests are $\phi_{u\epsilon}(\tau)$, $\phi_{u^2\epsilon}(\tau)$ and $\phi_{u^2\epsilon^2}(\tau)$ (see [115, 116] for more details), where $\tau$ is the sample time difference (from the current time). In our case $\tau$ is the time interval separating two bunches in the beam. If a correct number of lags is assigned, the following conditions are valid [115]:

$$\phi_{u\epsilon}(\tau) = \mathbf{E}(\epsilon(t)u(t-\tau)) = 0, \tag{7.2}$$

$$\phi_{u^2\epsilon}(\tau) = \mathbf{E}(\epsilon(t)(u^2(t-\tau) - \bar{u}^2(t))) = 0, \tag{7.3}$$

$$\phi_{u^2\epsilon^2}(\tau) = \mathbf{E}(\epsilon^2(t)(u^2(t-\tau) - \bar{u}^2(t))) = 0, \tag{7.4}$$

where $\mathbf{E}$ is the expectation value operator, $\epsilon(t)$ is the error between the network prediction and its desired output value and $u(t)$ is the input variable (e.g., the voltage and phase of klystron 1). The sample correlation function between two sequences is given by:

$$\phi_{u\epsilon}(\tau) = \frac{\sum_{t=1}^{N-r} u(t)\epsilon(t+\tau)}{\left( \sum_{t=1}^{N} u^2(t) \sum_{t=1}^{N} \epsilon^2(t+\tau) \right)^{\frac{1}{2}}}. \tag{7.5}$$

To understand the significance of the correlation functions (7.2)-(7.4), let us consider the general case where we model a non-linear system subject to some noise $e(t)$. The noise $e(t)$ is additive to the output when, for example, it arises from the measurement of $\hat{y}(t)$ (e.g., noise in a BPM reading); it is not additive when it is internal to the system (e.g., noise in the magnetic field of a bending magnet). The output $\hat{y}(t)$ of the system at time $t$ can be modeled with the following polynomial [117]:

$$\hat{y}(t) = G^u[u(t)] + G^{ue}[u(t), e(t)] + G^e[e(t)], \tag{7.6}$$

where $G^e[e(t)] = \sum_{n=1}^{N} a_n e^n(t)$ and $G^u[u(t)] = \sum_{m=1}^{M} a_m u^m(t)$ are polynomials in $e(t)$ and $u(t)$, and $G^{ue}[u(t), e(t)] = \sum_{n=1}^{N} \sum_{m=1}^{M} a_n a_m e^n(t) u^m(t)$ contains all cross products. The polynomial coefficients are denoted by $a_m$ and $a_n$.

When the model of the system (7.6) is incomplete (i.e. polynomial terms are missing), the third correlation test $\phi_{u^2\epsilon^2}(\tau)$ detects missing polynomial terms in $G^u[u(t)]$ and $G^{ue}[u(t), e(t)]$. When the system is correctly modeled $\phi_{u^2\epsilon^2}(\tau) = 0$ for all $\tau$ [117]. When $\phi_{u^2\epsilon^2}(\tau) \neq 0$, it is necessary to detect odd and even polynomial terms that should be included in the model. This is achieved by using the first and second correlation tests, $\phi_{u\epsilon}(\tau)$ and $\phi_{u^2\epsilon}(\tau)$, which detect odd and even terms, respectively. Sometimes, when the terms $u^2(t)$ and $\epsilon^2(t)$ are small, the correlation $\phi_{u^2\epsilon^2}(\tau)$ may also be small although $u^2(t)$ and $\epsilon^2(t)$ are correlated. It is therefore important to evaluate the tests $\phi_{u\epsilon}(\tau)$ and $\phi_{u^2\epsilon}(\tau)$ [117].

In practice the correlations cannot be zero, so we refer to the 95% interval of confidence defined by $95/\sqrt{N}$ [%], where $N$ is the number of samples. In what follows Eqs. (7.2), (7.3) and (7.4) will be refereed to as tests 1, 2 and 3, respectively.

## 7.4 Energy control results at the Australian Synchrotron Linac

### 7.4.1 Description of the experiment

An experiment was carried out to test the ability of the NNet to predict deviations in the beam energy, which can then be used for corrective action. Because the Australian Synchrotron Linac is not equipped with bunch length measurement diagnostics, this first study is limited to energy control using the beam position monitor (BPM) that was provided by Sincrotrone Trieste. Perturbations of different amplitudes and frequencies were injected into the phase and voltage of the first klystron (see Fig. 7.2), and corrected by the second klystron. The energy deviation of the beam is measured as a deviation in the horizontal position of the beam at the BPM.

Two types of networks were tested to validate the predictions: a hyperbolic tangent (HT) network and a radial basis function (RBF) network (the type of network is defined by the activation function in the hidden layer, as discussed in Chapter 6). The HT network was used for single frequency jitter control and to evaluate the use of the combined feedforward - feedback algorithm; however both networks were evaluated for the control of multiple frequency jitter.

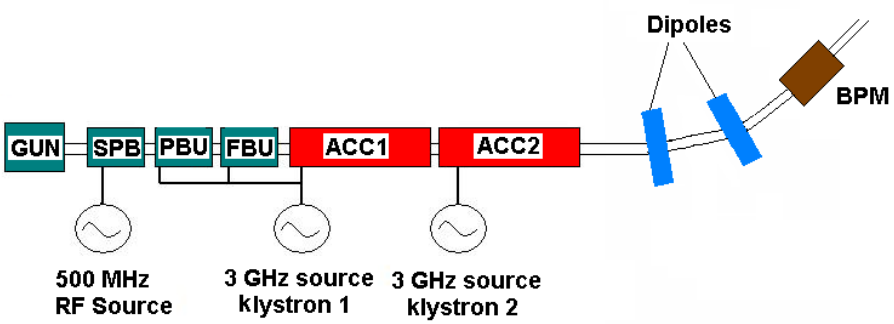



Figure 7.2: Simplified layout of the Australian Synchrotron Linac shown in Fig. 3.1. The phase and voltage of klystron 1 are used to induce a perturbation in the beam energy, while the phase and voltage of klystron 2 are used to correct for it. The resulting beam energy perturbation is measured as a horizontal deviation of the beam position, recorded at the BPM.

### 7.4.2 Collecting data

Three sets of data were collected in order to evaluate the network structure and its training. The first data set, consisting of 115 sequences of 100 bunches each, was recorded to evaluate appropriate NNet topologies for the single frequency case. The bunch repetition rate was 1 Hz. For each sequence, different jitter frequencies and amplitudes are introduced to excite the first klystron phase and voltage (i.e. $\phi_1$ and $V_1$). The maximum amplitude was chosen to be 3° rms for the phase and 0.5 kV rms for the voltage. In both cases the maximum jitter frequency introduced was 0.1 Hz. We chose an upper limit for the frequency that was quite low due to the slow response of the klystron and the information travel time from the control room to the actuators. For each sequence the position deviation and the phase and voltage of klystron 1 were recorded.

A second data set, consisting of 16 sequences of 100 bunches, was then recorded. This set was used for training the network. The induced single-frequency jitter ranged from 0.05 kV to 0.08 kV and from 0.05 Hz to 0.08 Hz (with steps of 0.01 kV and 0.01 Hz, respectively), for the first klystron voltage. The phase jitter was maintained at 3° amplitude at 0.05 Hz.

The last set of measurements consisted of 40 sequences of 100 bunches, each with 3 induced frequencies ranging from 0.01 Hz to 0.07 Hz, and with amplitudes ranging from 0.04 kV to 0.06 kV. The phase jitter was again maintained at 3° amplitude at 0.05 Hz. This data set was used to evaluate the topology and train a HT network and a RBF network in our multi-frequency study.

### 7.4.3 Determination of the network structure

#### NNet structure for a single frequency

In order to determine if the topology depends on the jitter frequency, the network was first trained for a specific frequency. Since one hidden layer has been shown to be adequate for fitting any function to an arbitrary degree of accuracy [16], the network comprises one input layer, one hidden layer and one output layer. There is one output (see Fig. 7.1), which corresponds to the position deviation recorded at the BPM. The activation functions for the hidden layer and the output layer were chosen to be hyperbolic tangents.

In what follows we present results for one of the sequences of the data set, with a voltage jitter of 0.96 kV at 0.078 Hz and phase jitter of 1.6° at 0.018 Hz (for klystron 1). The recorded data from the machine, which was used for the evaluation, is shown in Fig. 7.3. The deviation ranges from 0 mm to 1.8 mm corresponding to a deviation of 2 MeV in amplitude, or 2% since the unperturbed beam energy is 100 MeV.

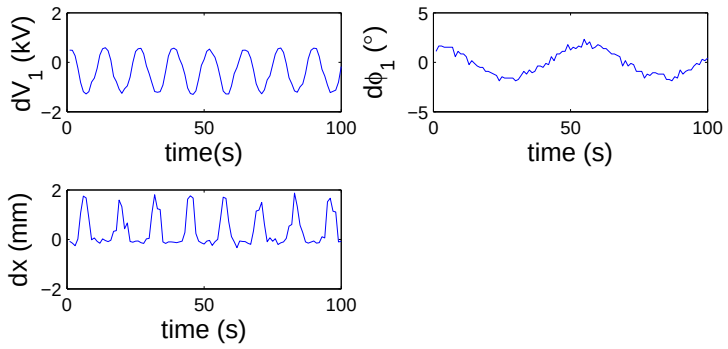



Figure 7.3: Example of a single sequence recorded for training the NNet predictor. The data includes records of the excited voltage jitter: 0.96 kV at 0.078 Hz (upper left plot), the excited phase jitter: 1.6° at 0.018 Hz (upper right plot), and the induced horizontal beam position jitter at the BPM (lower left plot). Figure adapted from [63].

Equations (7.2)-(7.4) were used to evaluate the required number of $dV_1$ and $d\phi_1$ lags. To do this, these equations were evaluated for combinations of 1 to 12 lags of $dV_1$ and $d\phi_1$, with $\tau = [-30, -29, ..., 0, 1, 2, ..., 29, 30]$. For each combination, the standard deviation (over $\tau$) of the results were plotted against the number of $dV_1$ and $d\phi_1$ lags (see Fig. 7.4). It was noticed that the number of phase input lags only has a small influence on the results due to the small effect of the phase jitter compared with the voltage jitter. This

is seen in Fig. 7.4, where the first correlation test $\phi_{u\epsilon}(\tau)$ given in Eq. (7.2) was evaluated for the input variable $u = dV_1$. This figure shows that at least 3 or 4 lagged values of $dV_1$ are required to minimise the correlation between $dV_1$ (the input variable) and $\epsilon$ (the error between the NNet prediction for the position deviation and the actual deviation). The other two correlation tests (Eqs. (7.3) and (7.4)) showed the same trends.

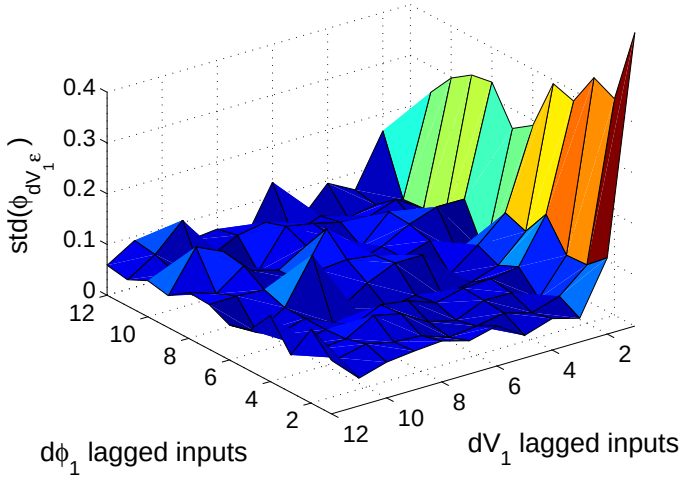



Figure 7.4: Standard deviation for the first correlation function given in Eq. (7.2), evaluated for the input variable $u = dV_1$, as a function of the input lags $dV_1$ and $d\phi_1$. The HT network had 6 hidden nodes and was trained for the data shown in Fig. 7.3. The results show that at least 3 to 4 lagged values of $dV_1$ are necessary to minimise the correlation between $dV_1$ and $\epsilon$, while the number of $d\phi_1$ lags is not relevant. Figure adapted from [63].

The error index was evaluated using Eq. (7.1). Figure 7.5 plots the index error as a function of the number of hidden neurons $N_h$ and lagged values of $dV_1$. The error index does not decrease further for 6 hidden neurons and 5 to 6 lagged voltage inputs. Similar results are obtained with $d\phi_1$, showing that the phase has little influence on the error index and that no further decrease is achieved above 2 lagged values. Tests were repeated for a range of frequencies and amplitudes, which showed that the network performance (in terms of error index) and topology are frequency independent. Based on these results we chose a topology comprised of 5 lagged voltage input values and 2 lagged phase input values.

The performance of the network was then studied when the whole data set was used for training the network. The network received 5 lagged values of $dV_1$ and 2 lagged values of $d\phi_1$ as its inputs. Results for different sequences (see Fig. 7.6), show that the error index increases with decreasing amplitude of the jitter. As one can see from Fig. 7.6, the two upper curves

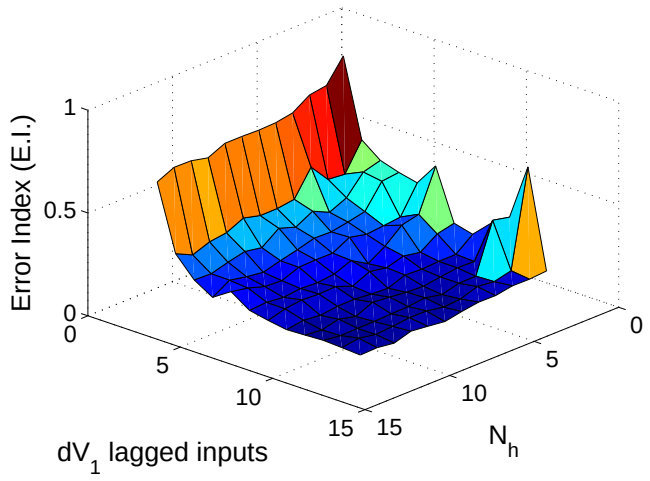



Figure 7.5: Error index (E.I) given in Eq. (7.1) as a function of the number of neurons in the hidden layer $N_h$ and the numbers of lagged values of $dV_1$. The HT network received 2 lagged values of $d\phi_1$ and was trained for the data shown in Fig. 7.3. The error index is minimised for 5 to 6 lags of $dV_1$ and 6 hidden neurons. Figure adapted from [63].

correspond to a low jitter amplitude (0.25 kV and 0.13 kV), whereas the two lower curves correspond to a higher amplitude (0.86 kV and 0.96 kV). This is directly related to the network training algorithm, which it is designed to minimise the error between the network output and its desired output in a least squares sense. The residual error in the output (beam position) for different data sequences will be of similar magnitude, and consequently will be independent of the amplitude of the input data (phase and voltage). If the network is trained over a wide range of input amplitudes, the smaller the amplitude the larger the residual error. With a sufficient number of hidden neurons ($N_h > 12$), the error index will be similar for all amplitudes. Although one is tempted to increase the number of neurons in order to improve the network performance, this may result in over fitting of the background white noise in the BPM measurements.

Whilst error indices indicate how well the network performs, correlation tests will confirm the appropriateness of our choice for the number of lagged inputs. From Fig. 7.7 it appears that at least 4 to 5 hidden neurons are needed in order for the standard deviation of the correlation tests to remain below about 0.05, corresponding to the 95% confidence interval. In fact, 6 hidden neurons is adequate to minimise the correlation and will be chosen for the NNet predictor.

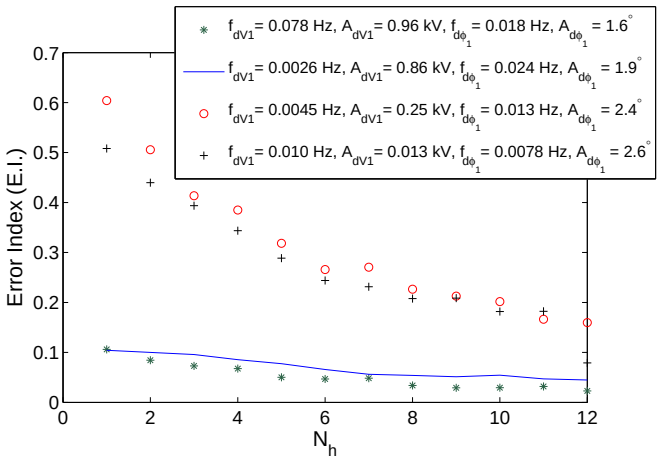



Figure 7.6: Error index (E.I.) as a function of the number of hidden neurons $N_h$ for various jitter characteristics. The HT network was provided with 2 lagged values of $d\phi_1$ and 5 lagged values of $dV_1$. Higher voltage jitter amplitudes result in a lower error index (blue curve and green data points). Figure adapted from [63].

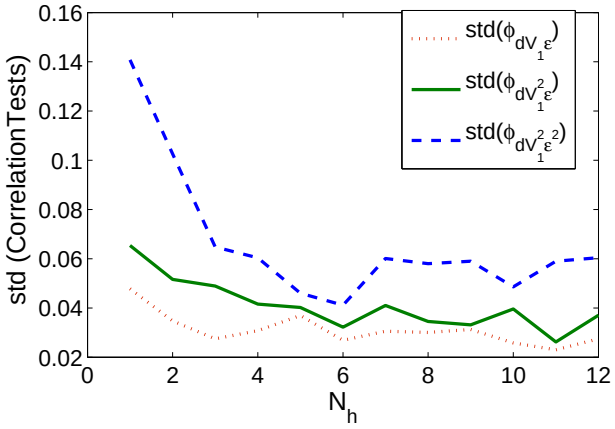



Figure 7.7: Standard deviation of the correlation functions (7.2)-(7.4) for the input variable $u = dV_1$ versus $N_h$. The HT network was provided with 2 lagged values of $d\phi_1$ and 5 lagged values of $dV_1$, and was trained with the data shown in Fig. 7.3. There is no significant reduction in the correlations when $N_h > 6$. Figure adapted from [63].

The correlation tests shown in Fig. 7.8 correspond to the evaluation of the topology for the recorded data given in Fig. 7.3. These tests were evaluated for each sequence in the collected data set; all results were within the 95% confidence interval. This confirms that the topology is viable for any combination of voltage and frequency in the data set.

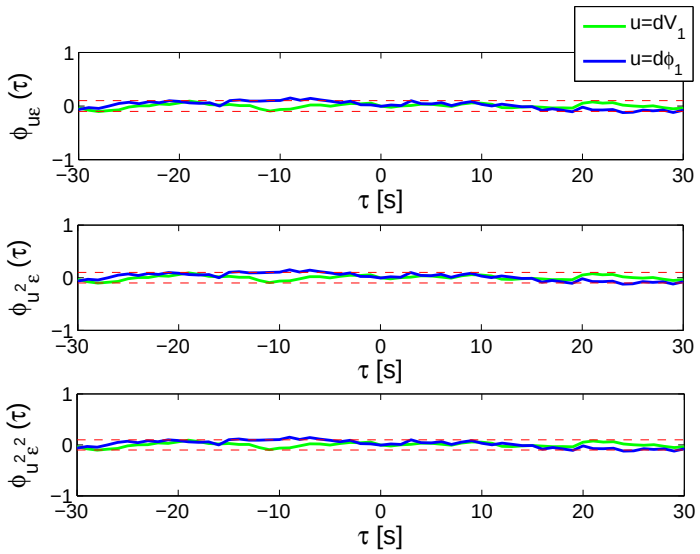



Figure 7.8: Evaluation of the correlation functions (7.2)-(7.4) for the input variables $dV_1$ (green curves) and $d\phi_1$ (blue curves) for $\tau = [-30,-29,...,-1,0,1,...,29,30]$. These results were obtained for a HT network consisting of 6 hidden neurons, receiving 5 lagged voltage inputs and 2 lagged phase inputs; the network was trained with the data shown in Fig. 7.3. All three correlation tests are within the 95% confidence interval, delimited by the horizontal dashed red lines. Figure adapted from [63].

#### NNet structure for multiple frequencies

The structure of the network for the multi-frequency case was evaluated with the same procedure as for the single frequency case, using the criteria outlined in Sec. 7.3. Studies were carried out for a HT and a RBF network.

(a) *Hyperbolic tangent network*

The evaluation of the topology for the multi-frequency case is not as obvious as for the single frequency case. First, it was noted that the number of neurons does not have a significant impact on the error index as one can see from Fig. 7.9. This figure also shows that the error index is not significantly affected by the number of $dV_1$ lags.

A better indication of the appropriate number of lagged values of $dV_1$ and $d\phi_1$ is given by the correlation tests. Figure 7.10 shows that including 4 to 6 lagged values of $dV_1$ reduces the correlations, whereas the number of lagged $d\phi_1$ values does not have a significant effect.

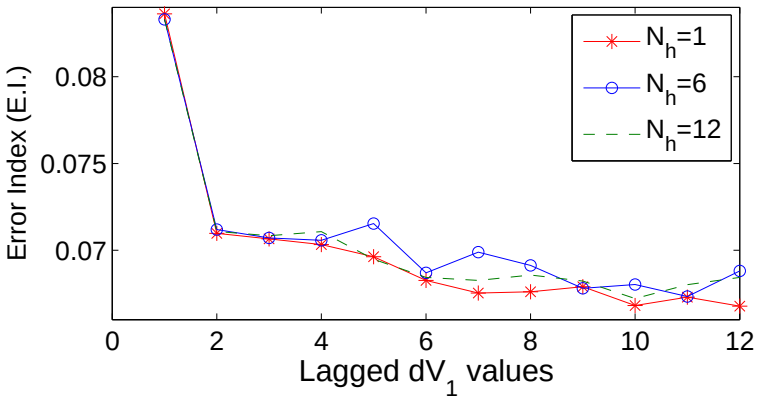



Figure 7.9: Error index (E.I.) versus number of $dV_1$ lags for 1 (red curve), 6 (blue curve) and 12 (dashed green curve) hidden neurons of an HT network. The network was provided with 2 lagged values of $d\phi_1$. The number of neurons in the hidden layer has a negligible effect on the error index for the whole range of lagged values of $dV_1$. Figure adapted from [63].

Figure. 7.11 shows the correlation tests for a HT network with 6 hidden neurons, 6 lagged values of $dV_1$ and 2 lagged values of $d\phi_1$. The upper two graphs show some points outside the 95% confidence bands, indicating a residual correlation with the last 10 bunches. This was observed for different numbers of lagged inputs and neurons. Typically, when the correlation tests show a peak for a given $\tau$, this means that the network is lacking some information at that specific time difference. The solution is to simply add the corresponding lag to the network inputs. In our case, if the correlation is to be decreased further, some of the lagged values could be given to the network a second time (i.e. another input can be added to the NNet to feed the same lagged value a second time). However, since the level of correlation was not significant it was decided to proceed with the same network topology. Therefore we chose a topology for online testing consisting of 6 hidden neurons, receiving 5 lagged values of $dV_1$ and 2 lagged values of $d\phi_1$ as its inputs.

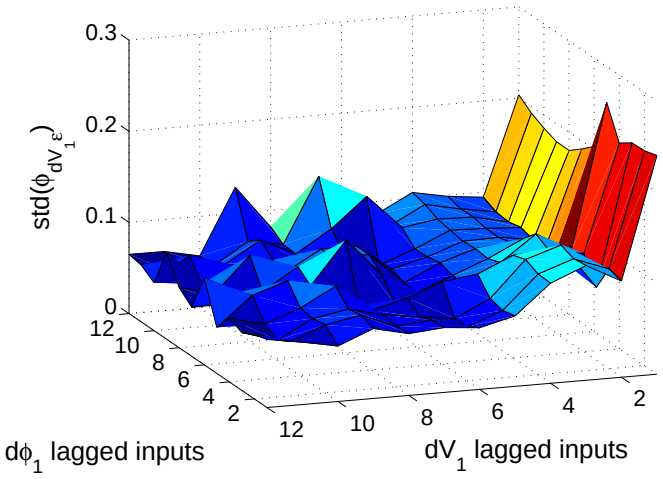



Figure 7.10: Standard deviation for the first correlation function given in Eq. (7.2), evaluated for the input variable $u = dV_1$, as a function of the input lags $dV_1$ and $d\phi_1$. The HT network was trained for the data shown in Fig. 7.3. These results show that 4 to 6 lagged values of $dV_1$ reduce the correlation, while the number of $d\phi_1$ lags does not have a significant effect. Figure adapted from [63].

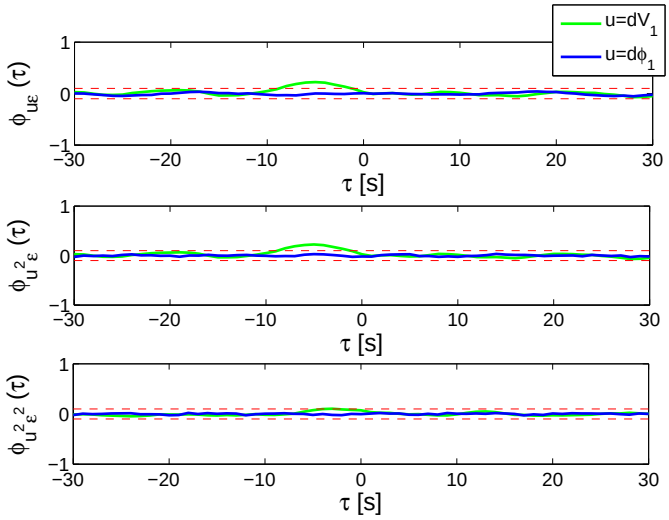



Figure 7.11: Evaluation of the correlation functions (7.2)-(7.4) for the input variables $dV_1$ (green curves) and $d\phi_1$ (blue curves) for $\tau = [-30, -29, ..., -1, 0, 1, ..., 29, 30]$. The HT network consisted of 6 hidden neurons, receiving 6 lagged values of $dV_1$ and 2 lagged values of $d\phi_1$. The network was trained for a sequence with jitter frequencies 0.015 Hz, 0.05 Hz and 0.065 Hz, with respective amplitudes of 0.55 kV, 0.4 kV and 0.45 kV. Points between $\tau = -10$ and $\tau = 0$ lie slightly outside the 95% confidence bands for the two first correlation functions (upper plots).

(b) *Radial basis function network*

For a hyperbolic tangent network one specifies the number of inputs, outputs and hidden neurons that constitute the network. Then a training algorithm is used to adjust the weights of the network, reproducing desired outputs for given inputs. However, RBF networks are constructed by adding one neuron at a time, until the sum-squared error falls below an error goal [87, 118]. Consequently, the error index estimated using the training set is nearly constant over the whole range of $dV_1$ and $d\phi_1$ lags, since training stops as soon as the goal is attained. Therefore, only the number of lags needs to be determined using the correlation tests.

Figures 7.12 and 7.13 show the results for the first correlation test given by Eq. (7.2). This is the most relevant of the three tests since it is likely to lie outside the confidence bands. The tests were carried out at frequencies 0.01 Hz, 0.02 Hz and 0.07 Hz, and for an amplitude of 0.06 kV. Larger correlations in Fig. 7.12 (c.f. Fig. 7.13), reflect the higher sensitivity of the system to the voltage. According to Fig. 7.12, the voltage correlation is reduced with a small number of lagged $d\phi_1$ and $dV_1$ inputs, i.e. 5 to 6 lagged voltage values and 1 to 3 lagged phase values.

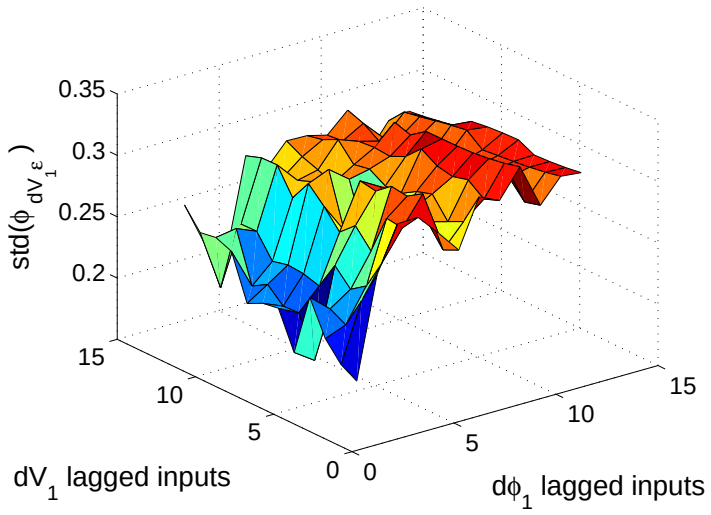



Figure 7.12: Standard deviation for the first correlation function given in Eq. (7.2), evaluated for the input variable $u = dV_1$, as a function of the input lags $dV_1$ and $d\phi_1$. The RBF network was trained for a single sequence of the data set (i.e. 0.06 kV at 0.01 Hz, 0.02 Hz and 0.07 Hz and 3° at 0.05 Hz). Increasing the number of $d\phi_1$ lags leads to an increase in the correlation, while the number of $dV_1$ lags does not have a significant effect. Figure adapted from [63].

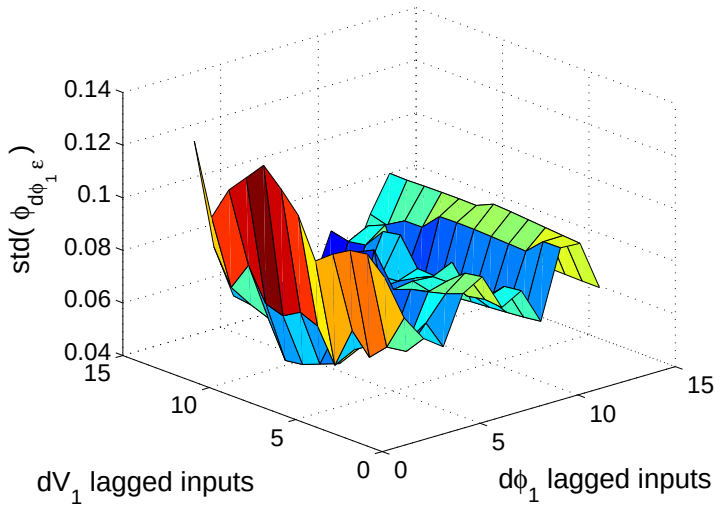



Figure 7.13: Standard deviation for the first correlation function given in Eq. (7.2), evaluated for the input variable $u = d\phi_1$, as a function of the input lags $dV_1$ and $d\phi_1$. This evaluation was carried out using a RBF network trained with a single sequence of the data set (i.e. 0.06 kV at 0.01 Hz, 0.02 Hz and 0.07 Hz and 3° at 0.05 Hz). Increasing the number of $d\phi_1$ lags decreases the correlation, while the number of $dV_1$ lags does not have a significant effect.

The correlation in the phase is minimised for two or three lagged values (see Fig. 7.13). The number of lagged voltages has less effect, tending to reduce the correlation as more lags are added. However, it was found that the voltage correlation tends to increase with the number of lagged $dV_1$, depending on the test data sequence. An upper limit of 5 lagged inputs was found to be a good compromise in these cases.

Based on these observations the network topology was configured to include 5 lagged values of $dV_1$ and 2 lagged values of $d\phi_1$. The absolute value of the correlation can be reduced by setting a lower error goal during training, without affecting the trends of the correlation tests (as functions of the number of the $dV_1$ and $d\phi_1$ lags). However, to ensure an optimum error goal is attained, the error between the network outputs and the training outputs should be comparable to the rms amplitude of the white noise of the BPM. Imposing this criterion on the training phase ensures that over fitting the white noise does not occur.

The whole data set was then presented to the network for training. According to Fig. 7.12, when the network is trained for a single sequence, increasing the number of lagged values of $d\phi_1$ increases the correlation; however, it is only slightly affected by the number of lagged values of $dV_1$. These trends remain when the network is trained for the whole data set, as shown in Fig. 7.14. The average number of neurons used for training on a single data sequence was 60 to 75. For training over the whole data set, an upper limit of 150 neurons was utilised together with an error goal corresponding to the level of white noise at the BPM. The training stopped when the network size reached 150 neurons, before the error goal was reached. Consequently, the network prediction error is larger than the white noise and correlations show some points outside the confidence bands. This is illustrated in Fig. 7.15, where the three tests are represented for one specific sequence, using 5 lagged voltage values and 2 phase input lags.

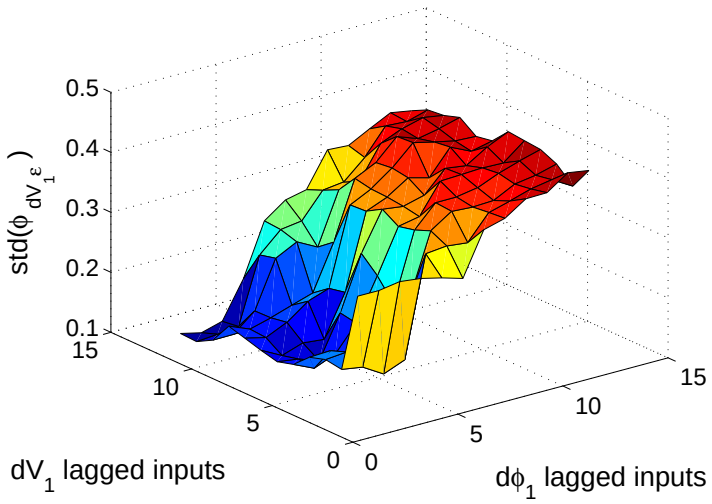



Figure 7.14: Standard deviation for the first correlation function given in Eq. (7.2), evaluated for the input variable $u = dV_1$ and for $\tau = [-30, -29, -28, ..., 29, 30]$. This evaluation was carried out for a RBF network trained using all sequences of the data set. An increasing number of $d\phi_1$ lags leads to an increase in the correlation, while the number of $dV_1$ lags does not have a significant effect. Figure adapted from [63].

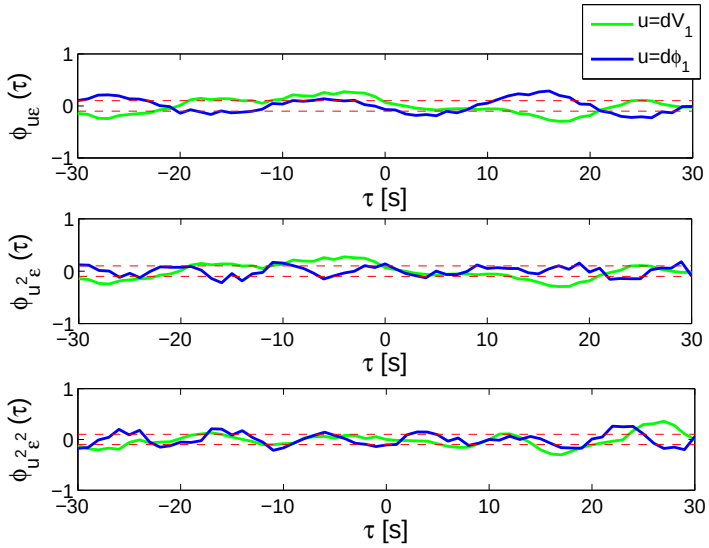



Figure 7.15: Evaluation of the correlation functions (7.2)-(7.4) for the input variables $dV_1$ (green curves) and $d\phi_1$ (blue curves) for $\tau = [-30,-29,...,-1,0,1,...,29,30]$. The RBF network consisted of 150 hidden neurons, receiving 5 lagged values of $dV_1$ and 2 lagged values of $d\phi_1$. The network was trained using the whole data set. Points lying outside the 95% confidence bands for the three correlation functions are due to the interruption of training before the error goal was reached (i.e. before the maximum number of hidden neurons was reached).

### 7.4.4 Control of a single frequency jitter

Here we evaluate the real-time performance of the topology discussed in Sec. 7.4.3. The NNet was trained for jitter frequencies ranging from 0.05 Hz to 0.08 Hz and with amplitudes ranging from 0.05 kV to 0.08 kV, using steps of 0.01 Hz and 0.01 kV, respectively. The upper limit of 0.08 Hz is due to the slow response of the actuators. The response of klystron 2 was measured and found to be linear in the region of interest, with the correction proportional to the network prediction. The performance index (*P.I.*) is computed as:

$$P.I.(\%) = \left( 1 - \frac{std(dx_f) - WN}{std(dx_i) - WN} \right) \times 100, \tag{7.7}$$

which takes into account the irreducible white noise (WN) present, even when no jitter is excited in klystron 1. Here $std(dx_f)$ and $std(dx_i)$ denote the standard deviations of the final (under NNet control) and initial (without control) horizontal deviations. Thus, jitter is entirely canceled if the residual standard deviation is of the same magnitude as the white noise.

Figure 7.16 shows the resulting performance index, as a function of the excited jitter amplitude and frequency. Training used a record of 100 bunches, however, the evaluation of the network was done with 1000 bunches. The performance index was evaluated for amplitudes and frequencies different from the training data, but included in the training range, from 0.045 Hz to 0.075 Hz and with amplitudes ranging from 0.055 kV to 0.085 kV; steps of 0.01 Hz and 0.01 kV were used for the frequency and amplitude, respectively. The performance decreases with increasing frequency, which is not due to the network itself, but to the slow response of the actuators. In fact, the phase and voltage of the first klystron are needed for the NNet to compute predictions before their values reach the set point. Thus, the information is sent too early to the network and its prediction is shifted in time. Because of the time shift the network acts as a feedback system and not predominantly in the feedforward mode. The network was trained on a range of small amplitudes; however, there were no significant performance differences between sample sequences.

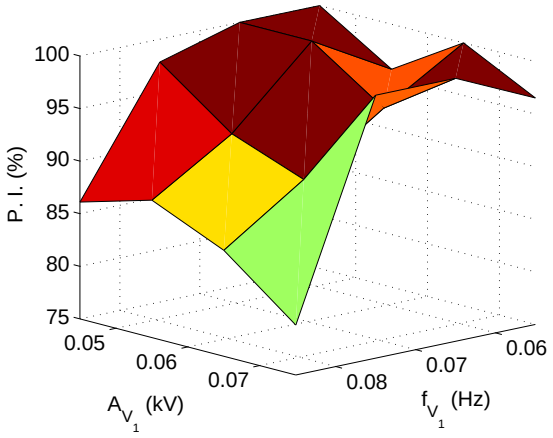



Figure 7.16: Performance Index (P.I.) versus jitter amplitude $A_{V_1}$ and frequency $f_{V_1}$ induced in klystron 1. The P.I. was evaluated over 1000 bunches.

Figure. 7.17 shows an example of recorded data acquired during evaluation. The induced jitter is a sine wave of amplitude 0.05 kV and frequency 0.05 Hz. The solid red curve corresponds to the data recorded for training the network, whereas the dashed blue curve corresponds to the network prediction. The residual deviation with NNet control is given by the green circles and has a rms deviation of 0.116 mm, which corresponds to the background noise level. Figure 7.18 shows the Fast Fourier Transform (FFT) for the controlled jitter sequence (lower green curve) with data points indicated by stars; the uncontrolled jitter (upper red curve) is represented by crosses. This confirms that the 0.05 Hz perturbation was suppressed.

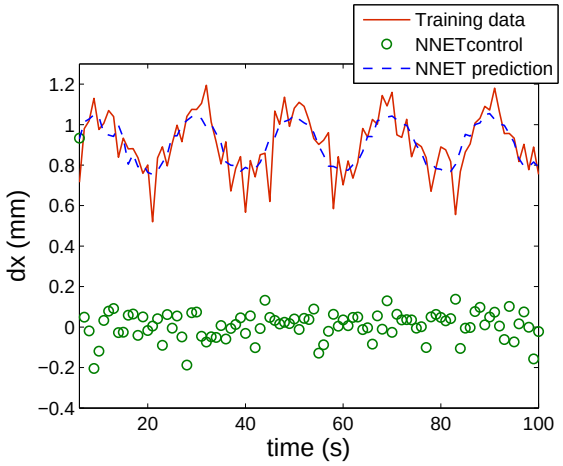



Figure 7.17: Example of real-time control operated over 100 bunches for a jitter of 0.05 kV and 3° at 0.05 Hz. The red curve shows a record of the perturbed beam position (training data), the dashed blue curve shows the fit of the NNet and the green circles show the record of the position deviation when the NNet operated the control. Figure adapted from [63].

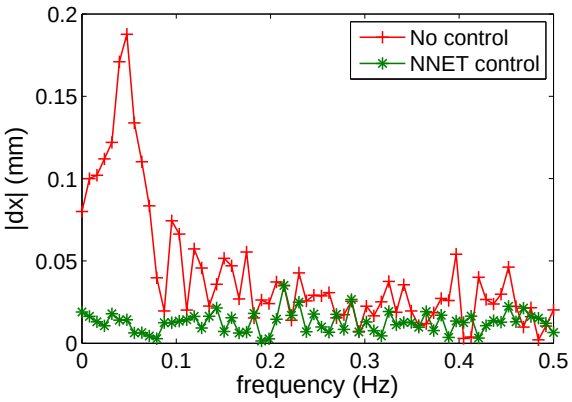



Figure 7.18: FFT for controlled jitter (lower green curve) and uncontrolled jitter (upper red curve), corresponding to the data in Fig. 7.17. Figure adapted from [63].

### 7.4.5 Control of multiple frequency jitter

Although the evaluation of network topology is more difficult for the control of multiple frequency jitter, the results of real-time control for the multi-frequency case were comparable to the single frequency case. Tests were made for both hyperbolic tangent and Gaussian (RBF) networks.

(a) *Hyperbolic tangent network*

The network was trained over a set of 200 bunches with excited jitter of 0.01 Hz, 0.02 Hz and 0.05 Hz, each of 0.06 kV amplitude (see recorded data in the upper plots of Fig. 7.19). The network consists of 6 hidden neurons receiving 6 lagged values of $dV_1$ and 2 lagged values of $d\phi_1$ as its inputs. The online control results are shown in Fig. 7.19 (middle plots), where it is evident that the residual noise is at the background level. Thus, the network with the same topology is performing for the multi-frequency jitter as well as for a single frequency. Fourier analysis shown in Fig. 7.19 indicates that there is no obvious residual component of jitter frequency.

As a second test, the network was trained over a set of 24 sequences with frequencies ranging from 0.01 Hz to 0.05 Hz and with amplitudes ranging from 0.04 kV to 0.06 kV. The network was then tested in real-time by inducing frequencies and amplitudes different from the training set, but included within the training range. Each sequence was 250 bunches long. The standard deviation of each of the 16 test samples was below 0.09 mm rms, corresponding to the background noise level. (The white noise was measured to be 0.085 mm rms at the time of this experiment). For each sample test, all frequencies were suppressed. This confirms the ability of the network to interpolate its prediction when encountering frequencies that are different from the training set, but included within the training range.

(b) *Radial basis function network*

The RBF network, consisting of 76 hidden neurons, was trained using the same sample data as for the hyperbolic tangent network. The data was required to be fitted with a precision of 0.085 mm rms (corresponding to the background noise level).

The trained RBF network was then tested for the same jitter conditions as for the HT network, i.e. same jitter frequencies and amplitudes. Although the residual noise is slightly higher than for HT network, Fourier analysis reveals that the frequency components were successfully suppressed (see lower plots in Fig. 7.19). This test was repeated several times for both the

HT and RBF network. The residual noise is always slightly lower using the HT network. Moreover, the RBF network does not totally cancel the frequency components; the RBF network shows better results when less data is presented in the training phase. Usually one to two periods of the slowest jitter component is enough.

As for the HT network, the controller was trained over a set of 24 sequences with frequencies ranging from 0.01 Hz to 0.05 Hz and amplitudes ranging from 0.04 kV to 0.06 kV. The same 16 jitter configurations were tested over 250 bunches. The standard deviation of each of the 16 test samples was comparable to the background noise level, which corresponds to the successful cancelation of all frequencies for all sequences.

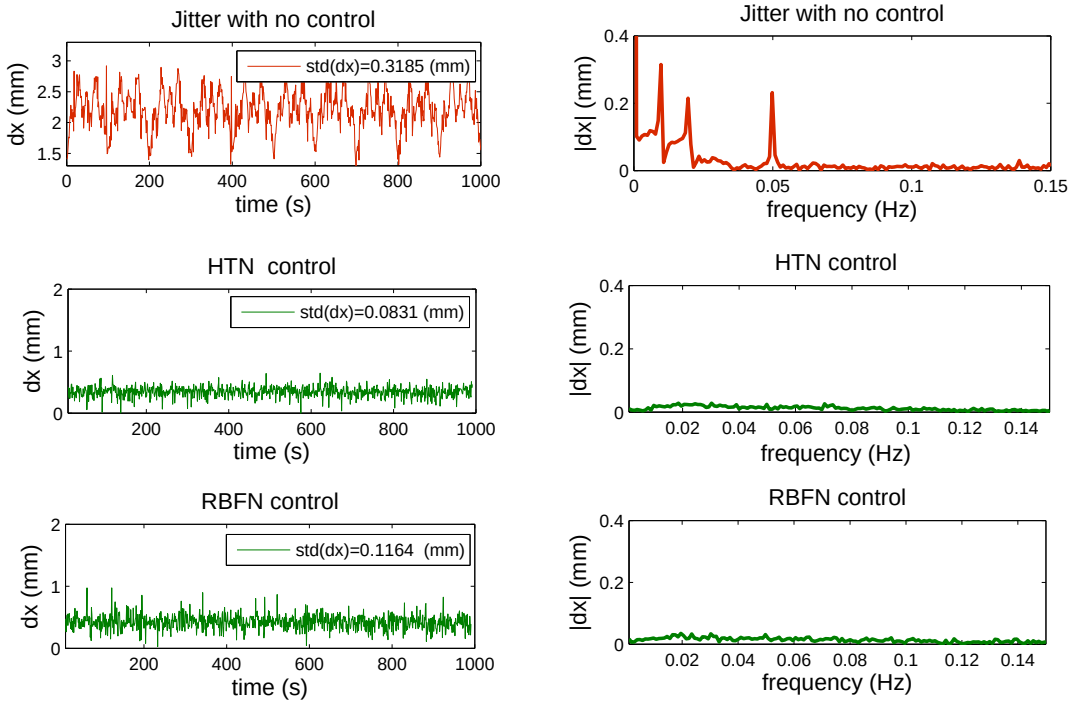



Figure 7.19: Example of real-time control operated over 1000 bunches, for voltage jitter frequencies of 0.01 Hz, 0.02 Hz and 0.05 Hz. Each frequency has an amplitude of 0.06 kV. The phase jitter was 1.5° at 0.05 Hz. The upper plots give the recorded BPM readings (left) and the corresponding FFT (right) without control. The middle and lower plots give the BPM reading and the corresponding FFTs, when the control is operated with the hyperbolic tangent network (HTN) and the radial basis function network (RBFN), respectively. Figure adapted from [63].

### 7.4.6 Feedforward-feedback combined control

To minimise the residual jitter, in those situations where the NNet predictions are not accurate enough to totally cancel the perturbation, it is possible to couple the network to a feedback control. A very simple scheme consists of using a Proportional-Integral (PI) control as shown in Fig. 7.1. The total correction is given by:

$$dV_2(k+1) = -M^{-1}dx(k+1) - P_g M^{-1}dx(k) \\ -I_g M^{-1}\sum_{l=k-R}^{k} dx(l), \tag{7.8}$$

where $dV_2$ is the correction applied to the voltage of klystron 2 and $M$ is the response of the horizontal position $dx$ of the beam to a change in the voltage of klystron 2. The PI gains are denoted by $P_g$ and $I_g$, and $R$ is the sum range (i.e. the number of past bunches taken into account in the sum of the integral correction). If the NNet is operating correctly, the residual deviation is small and there will be very little feedback correction. When the network performance is decreased, the sum of deviations over time will increase, and so too will the feedback correction term. In those situations where the network is not performing optimally, the combination of feedforward-feedback will ensure stability of the system; in this case its mis-predictions can be compensated by a feedback term.

The system was tested for the case of a single frequency jitter. A HT network with 6 hidden neurons, 5 lagged values of the voltage and 2 lagged values of the phase was trained over a sequence of 200 bunches with a jitter of 0.1 kV at 0.04 Hz in klystron 1. The PI gains were then coarsely tuned with no additional control from the NNet. A control test showed that both the NNet and the PI controller were able to entirely suppress the perturbation; no peak could be identified from the FFT and the rms deviation of the beam position corresponded to the white noise level (0.92 mm rms for the NNet control and 0.090 mm rms for the PI control). The combined controller was not tested since the perturbation was entirely suppressed by the NNet. These results are listed in Table 7.1.

To evaluate the system response, the network was used to correct jitter with frequency and amplitude different to the training set. In the first experiment, a jitter of 0.1 kV and 0.06 Hz was excited in klystron 1 (different frequency to the training set but with the same amplitude). In a second experiment the excited jitter was 0.2 kV at 0.04 Hz (different amplitude to the training set but same frequency). The residual standard deviation and peak amplitudes of the FFT are listed in Table 7.1.

Table 7.1: Recorded rms deviation and FFT peak values in mm, for a change in jitter frequency (second column) and amplitude (third column) from the training set (first column). The deviation was evaluated for 1000 bunches.

| | $A_{V_1} = 0.1$ kV<br>$f_{V_1} = 0.04$ Hz | $A_{V_1} = 0.1$ kV<br>$f_{V_1} = 0.06$ Hz | $A_{V_1} = 0.2$ kV<br>$f_{V_1} = 0.04$ Hz |
|---|---|---|---|
| No control (rms dev.) | 0.153 | 0.150 | 0.250 |
| NNet control (rms dev.) | 0.092 | 0.089 | 0.146 |
| FFT peak value | - | 0.056 | 0.165 |
| PI control (rms dev.) | 0.090 | 0.113 | 0.146 |
| FFT peak value | - | 0.081 | 0.177 |
| Combined control (rms dev.) | - | 0.087 | 0.120 |
| FFT peak value | - | 0.039 | 0.126 |

In the first experiment (with increased frequency) the PI operates better than the NNet. The combined system only leads to a small improvement in the total rms deviation, since it is already close to the white noise level. However, the peak at 0.06 Hz is further decreased as shown in Fig. 7.20.

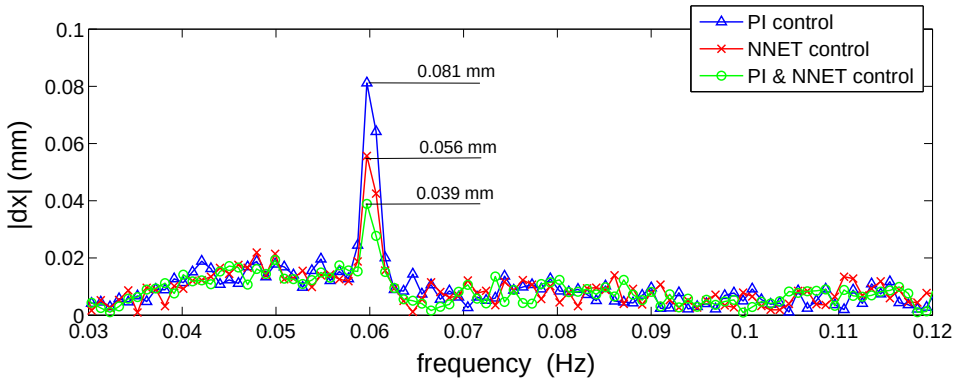



Figure 7.20: FFTs of the response of the PI controller, NNet and combined controller to a 0.1 kV jitter at 0.06 Hz, tuned to a 0.1 kV jitter at 0.04 Hz. The change in jitter frequency results in a 0.081 mm and 0.056 mm peak when the control is operated by the PI controller (blue curve) and the NNet (red curve), respectively. The combined controller (green curve) decreases the residual peak to 0.039 mm. The results are listed in Table 7.1. Figure adapted from [63].

For the second experiment (with increased amplitude), the NNet controller and the PI controller (using the same gains as the first experiment) show similar performance in terms of rms deviation. The combined con-

troller gives improved performance, decreasing the total rms noise from 0.25 mm to 0.12 mm. Similarly, the peak value in the FFT is lower in the combined controller (0.126 mm) than it is with the PI controller (0.177 mm), or the NNet alone (0.165 mm).

The combined controller showed a reduction in the rms deviation and the FFT peak value compared to the NNet or the PI controller alone. This was the case for both a change in the frequency and amplitude. Even though adding a PI controller requires more tuning of the system, it can easily be re-tuned automatically when the control system is initialised. However, because of its limited performance at high frequency, the PI controller is only capable of reducing low frequency jitter. The high frequency regime is then left entirely to the NNet, and training in this frequency regime will require special attention. One way to proceed consists of pre-training the NNet for a wide range of frequencies and in the expected range of amplitudes that might occur (this should be specified by the performance of the low level feedback acting on each klystron separately). "Final" training of the network can then be done using the latest recorded data, so that minor weight adjustments can be made for specific frequencies.

## 7.5 Energy and bunch length control results at the LCLS

### 7.5.1 Description of the experiment

The experiments conducted at the LCLS extended on the earlier work carried out at the Australian Synchrotron. The principal objective was to test the system for simultaneous control of the energy and bunch length. The first experiment was carried out to confirm that the NNet is capable of operating individual control of the beam energy and the peak current. To this end, multi-frequency jitter was introduced in the phase and voltage of the L0A section. The induced energy and peak current deviations were measured at the first bunch compressor (BC1), while the correction was applied to the voltage and the phase of the L0B section. Due to the slow response of the actuators of L0 and L1, the simultaneous control of the beam energy and peak current was performed using the L2 sections, powered by six klystrons; namely, 24-1, 24-2, 24-3, 24-4, 24-5 and 24-6 (see Fig. 7.21). Only the phases of klystrons 24-1, 24-2 and 14-3 were allocated for beam-based control. The jitter was injected into the phases of klystrons 24-1 and 24-2, while the correction was applied to the phase of klystron 24-3. The induced energy and peak current deviations were observed at the second bunch compressor (BC2).

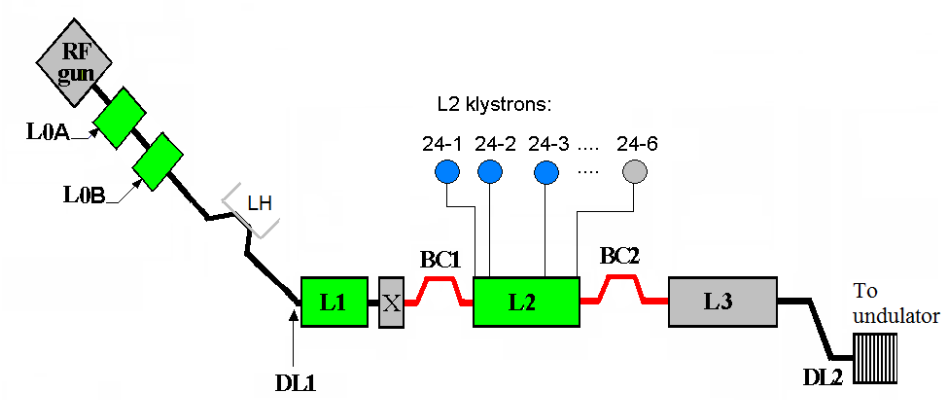



Figure 7.21: Simplified layout of the LCLS Linac shown in Fig. 3.2. For individual control of the beam energy and peak current, jitter is introduced into the phase and the voltage of L0A, while the correction is operated using L0B; the induced beam energy and peak current deviations are measured at BC1. For simultaneous control, jitter is introduced into the phases of klystrons 24-1 and 24-2, while the correction is operated using the phase of klystron 24-3; the induced deviations in the beam energy and peak current are measured at BC2.

Unlike the Australian Synchrotron Linac, the Linac sections in the LCLS are powered by several klystrons. In our study we will apply a change to all the klystrons of a section simultaneously, except when using the L2 section for the simultaneous control of the beam energy and peak current. Measurements were also conducted to compare the performance of the PI controller with the combined control system.

### 7.5.2 Collecting data for training

Machine data were acquired to determine an optimal network topology as described in Sec. 7.4.3. The data set comprises records of the phase and voltage of the L0A section, records of the horizontal beam position (measured at BC1) and peak current deviation (measured at BC1). A 3-frequency jitter was induced in the phase and voltage of the L0A klystron, with frequencies ranging from 0.1 Hz to 0.6 Hz. The horizontal deviation at BC1 varies from about -1.1 mm to 1.1 mm and the peak current ranges from ~230 A to ~330 A.

### 7.5.3 Determination of the network structure

The number of hidden neurons and lagged inputs was determined following the procedure developed for the Australian Synchrotron Linac described in Sec. 7.4.3. For the individual control (either horizontal position or peak current deviation), the optimal topology has 6 hidden neurons and 6 phase lags of the L0A section with 6 voltage lags. For the simultaneous control experiments (horizontal position and peak current deviation), the NNet comprises 8 hidden neurons, 8 voltage lags and 8 phase lags.

### 7.5.4 Results of real-time control application

#### Individual energy and bunch length control at BC1

First, the energy and peak current controls were evaluated individually. A deviation was induced in the phase and voltage of the L0A section. It comprised 0.05 Hz, 0.1 Hz and 0.3 Hz components, with voltage amplitudes of 0.1 MV, 0.2 MV and 0.15 MV, respectively, and a phase amplitude of 0.5°. The control was operated over 1000 bunches at 5 Hz and the voltage of the L0B section was used as the corrective actuator. Figure 7.22 shows the FFTs of the induced perturbation (dashed red curve) and the corrected beam (solid green curve).

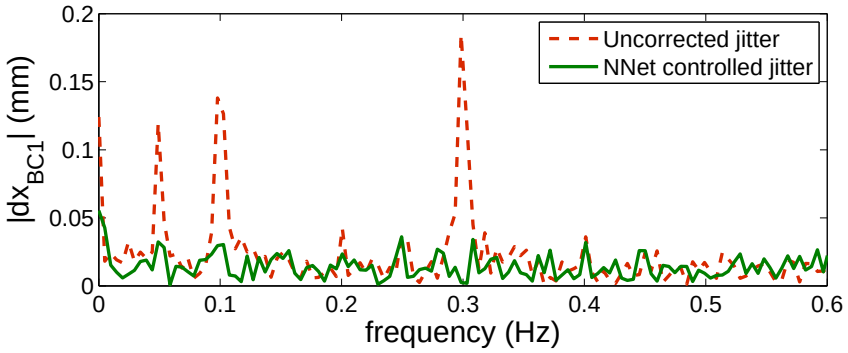



Figure 7.22: Results of individual energy control at BC1, measured in terms of the horizontal deviation of the beam. A 3-frequency jitter (0.05 Hz, 0.1 Hz and 0.3 Hz) was injected into the phase and the voltage of the L0A section and corrected using the voltage of L0B. The dashed red curve shows the FFT of the uncorrected positional deviation, while the green curve shows the FFT of the positional deviation when the correction is operated by the NNet. Figure adapted from [119].

For the peak current, the same jitter amplitudes were introduced, while the frequency components were increased to 0.1 Hz, 0.2 Hz and 0.6 Hz, respectively. To operate the simultaneous control with the L0B voltage as the energy actuator, the L0B phase was used as the peak current actuator. Unlike the energy, the FFT of the peak current reveals a modulation (see Fig. 7.23), which originates from the limited time response of the actuator. This modulation will be discussed further in Sec. 7.5.4.

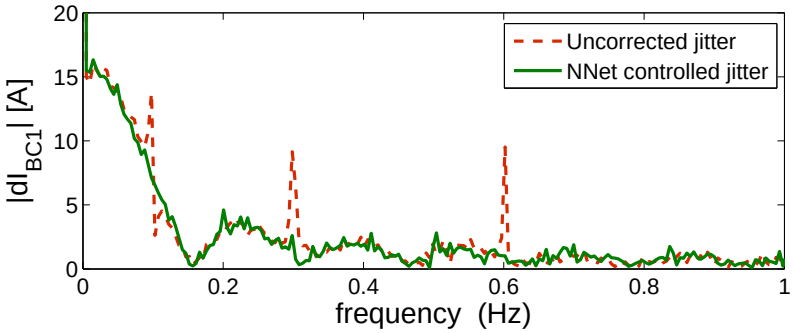



Figure 7.23: Results of individual bunch length control at BC1, measured in terms of the deviation in the peak current of the beam. A 3-frequency jitter (0.1 Hz, 0.2 Hz and 0.6 Hz) was injected into the phase and the voltage of L0A and corrected using the phase of L0B. The dashed red curve shows the FFT of the uncorrected peak current deviation, while the green curve shows the FFT of the peak current deviation when the correction is operated by the NNet. Figure adapted from [119].

#### Limitations of the actuators

It was observed that jitter control was unsuccessful for frequencies above 0.6 Hz, when the L0B voltage was the actuator; control was unsuccessful above 1 Hz when the L0B phase was the actuator. These limitations were due to a combination of slow communication with the actuators (via channel access using Matlab) and the klystrons running close to saturation.

When saturation occurs, a delay is caused, since a large change in the klystron input voltage is required to generate a small change in its output RF amplitude. As a result, the klystron response becomes non-linear and the voltage exhibits a slower response than the phase (see Fig. 7.24). Because the computation of the correction (Eq. (7.8)) assumes linearity of the actuator response, the calculated correction is under-estimated, leading to residual perturbations (i.e. perturbations due to incomplete corrections). This effect is shown in Fig. 7.25, where the theoretical correction (obtained by subtracting the uncorrected beam data from the NNet prediction) is plot-

ted against the actual correction of a 3-frequency energy jitter induced in L0A and observed at BC2. In this experiment all frequency components (0.2 Hz, 0.4 Hz and 1 Hz) had the same amplitude and the voltage of the L1 section was used as the actuator. The amplitudes of the two lower frequency components (0.2 Hz and 0.4 Hz) are larger than the background level due to the klystron running in the saturation regime.

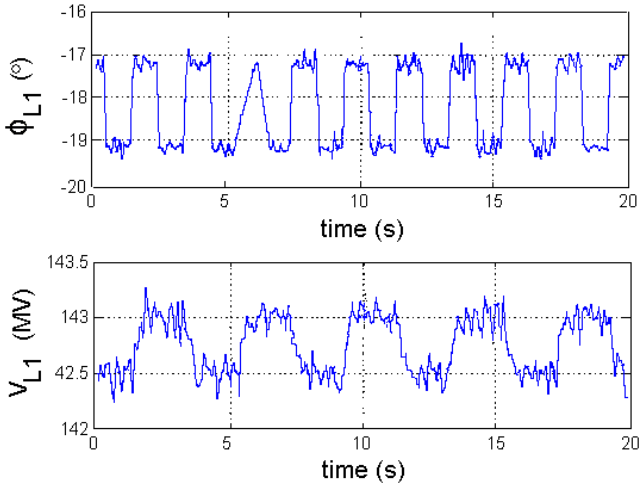



Figure 7.24: Time response of the phase and voltage of the LCLS Linac 1. The phase (upper plot) can respond to a 0.5 Hz square wave, whereas the voltage (lower plot) is not able to follow a 0.2 Hz square wave. Figure adapted from [119].

The slow response of the actuators, which in part is due to accessing the actuators using Matlab, results in a delayed correction. In this situation the system acts in feedback mode, which in turn results in poorer response with increasing frequency. This effect is seen in Fig. 7.25, where the NNet correction should have eliminated all frequency components (dashed green curve), but where the FFT of the controlled beam still shows a peak at 1 Hz (solid red curve). The travel time through the Matlab channel access layer can also lead to a loss of information. This is illustrated in the phase record (upper plot) of Fig. 7.24, which shows a triangular rather than rectangular signal shape at $t \simeq 7s$ due to missing data points.

In order to improve the communication channel, the Low Level RF (LLRF) smoothing factor[1] was increased from 0.4 (its nominal setting) to 1. Although this helps with the phase actuators it does not lead to a significant improvement in the amplitude because of saturation effects.

[1]The smoothing factor sets the percentage of the correction that is implemented in one step. A smoothing factor of 1 means that the full correction (i.e. difference between the current reading and the new set point) is applied in one step.

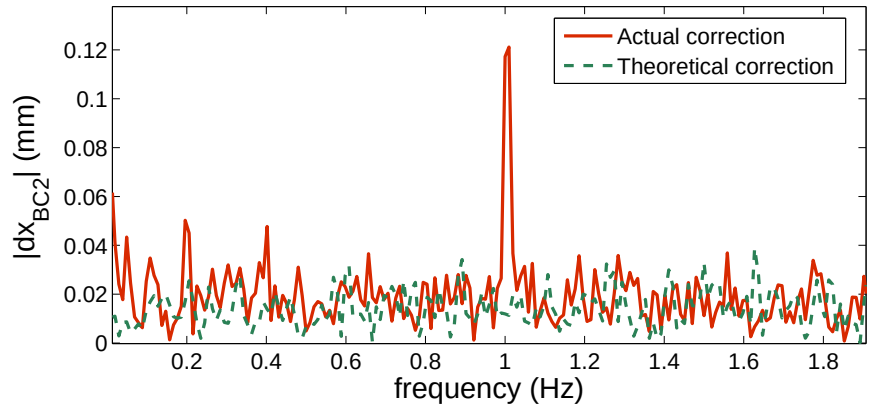



Figure 7.25: Effects of klystron saturation and slow actuator response. The voltage of L1 was used to correct a 3-frequency jitter induced in the phase of L0A, which perturbs the horizontal position of the beam at BC2. The jitter frequencies were 0.2 Hz, 0.4 Hz and 1 Hz. The dashed green curve shows the FFT for the correction that should have been achieved at BC2 with the NNet if the actuators were responding correctly, while the red curve shows the FFT for the actual control. The lower frequencies (0.2 Hz and 0.4 Hz) were almost suppressed, while the 1 Hz perturbation was not attenuated. Figure adapted from [119].

As alluded to in Sec. 7.5.4, in some cases a modulation can be observed in the FFT data. This modulation is due to the slow response of the actuators and reading devices. Figure 7.26 (a) displays the record of a 0.2 Hz sinusoidal perturbation imposed on the L1 phase (dotted green line) and the corresponding readback of the actuator (solid red curve). The latter contains repeated hard steps showing that the actuator reading is not updated frequently enough. The induced square waveform translates into a modulation in the frequency domain as shown in Figs. 7.26 (b) and (c). The 0.2 Hz modulation seen in the FFT corresponds to the period of the square waveform in Fig. 7.25. In the control results given in Sec. 7.5.4 (see Figs. 7.22 and 7.23), only the peak current shows such a modulation. This is because the energy, unlike the peak current, is relatively insensitive to the changes in the voltage of L0B. The implementation of the control system on local control boards using C++ should ameliorate limitations with the actuator speed.

#### Simultaneous energy and bunch length control at BC2

To overcome the slowness of the correcting actuators at L0 and L1 (located before BC1), a simultaneous energy-peak current experiment was performed

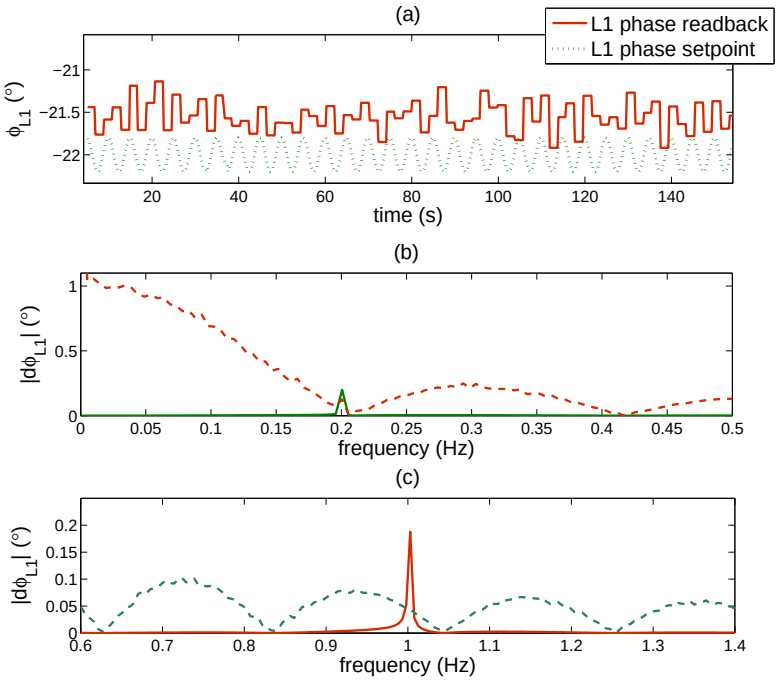



Figure 7.26: Set point, readback records and FFTs of the phase of the L1 section $\phi_{L1}$. (a) Induced jitter with frequency 0.2 Hz and amplitude of 0.2°. (b) FFTs for the induced jitter shown in (a). (c) FFTs for an induced jitter with 1 Hz and an amplitude of 0.2°. Figure adapted from [119].

at BC2, using the klystrons in the L2 section. Three out of the six klystrons in this section (i.e. 24-1, 24-2 and 24-3) were allocated for beam-based control and their phases manipulated. These klystrons respond much faster, since they are controlled in an open loop mode and are therefore much better candidates for high frequency jitter experiments. A 3-frequency jitter was injected into the phase of klystron 24-1, comprising 0.3 Hz, 0.4 Hz and 0.6 Hz components. The phases of klystrons 24-2 and 24-3 were chosen as corrective actuators for the energy and peak current, respectively. Figure 7.27 shows successful suppression of the perturbation for both the energy and the peak current.

### PI versus NNet at BC2

Figure 7.28 compares the performance of the NNet feedforward controller with the conventional PI controller. The NNet was re-trained for each frequency and the PI gains also re-tuned for each frequency. First, the sum range $R$ (i.e. the number of past bunches) in the third term on the right hand side of Eq. (7.8) was chosen such that the correction does not result in

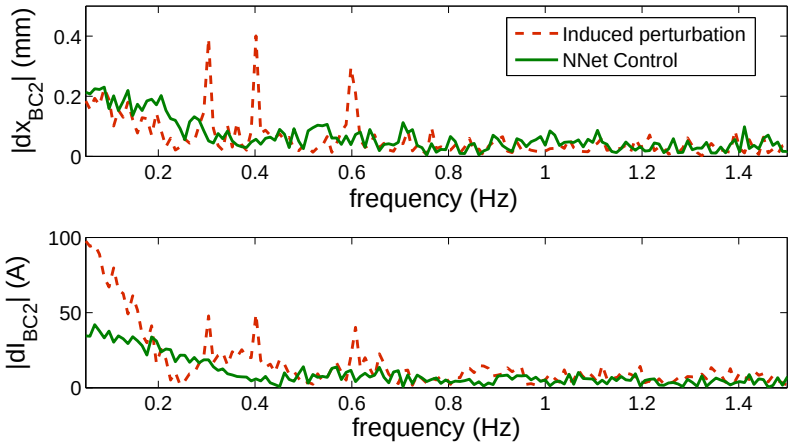



Figure 7.27: Simultaneous control of the energy and bunch length, measured in terms of the deviation of the beam position (upper plot) and peak current (lower plot), respectively. The three induced frequencies, 0.3 Hz, 0.4 Hz and 0.6 Hz (dashed curves) are successfully suppressed (solid curves) by the NNet controller. Figure adapted from [119].

excitation of higher frequencies when operating the control. The residual rms deviation was evaluated by scanning the $P_g$ and $I_g$ gains from 0.1 to 0.8 in order to find an optimal combination. Since the proportional gain tends to make the system unstable it was not used. Optimal integral gains ranged from 0.2 to 0.4 for the chosen frequency range 0.6 Hz - 2.4 Hz.

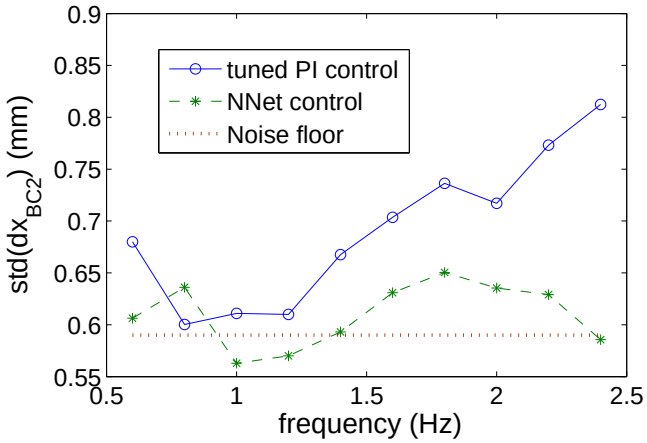



Figure 7.28: Comparison of the PI controller (blue curve) versus the NNet controller (green curve) for energy control at BC2. The jitter was induced in the phase of L1 and the control was operated using the phase of klystron 24-1. The dotted red line indicates the white noise level observed when no jitter was induced. Figure adapted from [119].

According to Fig. 7.28 both systems have similar operational performance at low frequencies (i.e. below 1.5 Hz). The dashed red line represents white noise, which fluctuates between 0.55 mm and 0.65 mm. From about 1.5 Hz the NNet starts to show better performance. The initial induced jitter had a rms amplitude of 0.98 mm.

#### System adaptability and dynamic response matrix

Until now it was assumed that the machine remained in a fixed state (in terms of the phase and voltage of the klystrons), and the response matrix was measured for that particular state. The response matrix was therefore static and re-measured for different machine states. However, during real-time operation, the machine parameters will change, due to phase drifts, resets, or in order to meet different beam energy or peak current requirements. One way to deal with this is to record enough data to train the NNet to recognise the machine state and reconstruct the related response matrix. In practice this approach is not feasible as there are too many parameters on which the machine state depends. To address this problem, we introduce a dynamic response matrix, in which the elements are re-calculated with the help of a model, when the settings of the klystrons change. The model uses the FMS code (See Chapter 5) that computes the main longitudinal dynamics comprising the RF acceleration, compression and wake fields, as discussed in Chapter 4.

To test the model, the phase of klystron 24-2 was scanned from 0° to 360°. The measured and modeled energy and peak current at the second bunch compressor are shown in Figs. 7.29 and 7.30, respectively.

Figure 7.29 shows good agreement between the model and the measured positional deviations. Differences appear when the deviation is large, i.e. more than ±12 mm. This is due to BPM saturation when the beam is away from the pipe axis, which translates to an under-estimate of the deviation. The theoretical and real-time response of the horizontal beam position to a change in the phase of klystron 24-2 is given by the slope $dx_{BC2}/d\phi_{24-2}$ of the curves in Fig. 7.29.

Figure 7.30 shows the modeled and measured peak current. Despite good qualitative agreement between the curves, the model produces a maximum of 3 kA when the machine actually reaches 5 kA. Consequently, the corresponding slope $dI_{BC2}/d\phi_{24-2}$ differs significantly between the model and the machine (see Fig. 7.31).

To compensate for inaccuracies in the correction, for example, due to mis-prediction of the model in the non-linear region of the BPM, the following algorithm was used for online operation. Every 200 bunches the

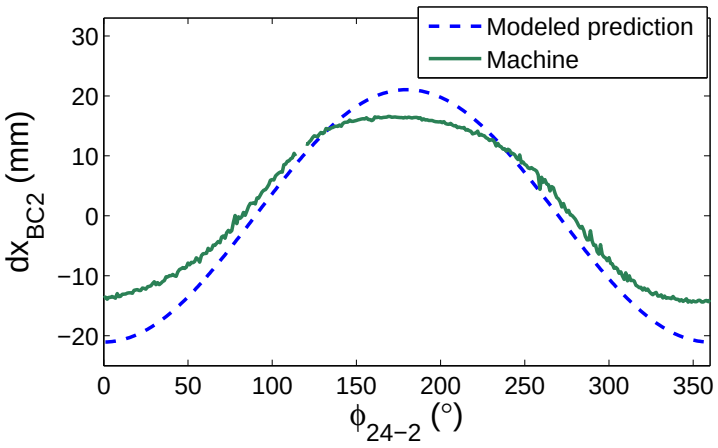



Figure 7.29: Modeled (dashed blue curve) and measured (solid green curve) deviation in the horizontal position of the beam at BC2 as a function of the phase of klystron 24-2. Figure adapted from [119].

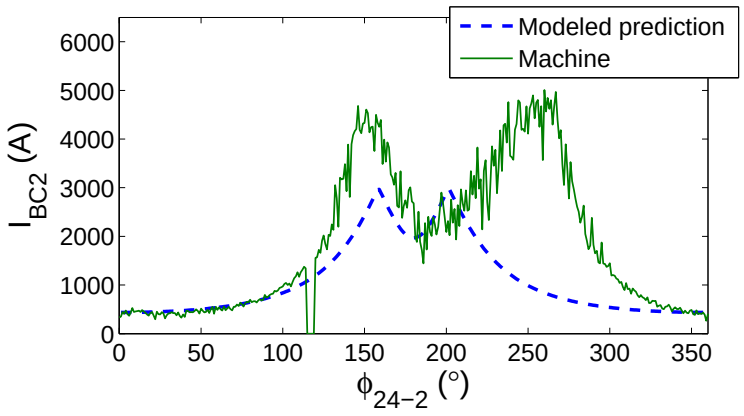



Figure 7.30: Modeled (dashed blue curve) and measured (solid green curve) peak current of the beam at BC2 as a function of the phase of klystron 24-2. Figure adapted from [119].

FFT of the energy and/or the peak current are/is computed. If the amplitude of the jitter is above the white noise threshold, the response matrix element is modified to further decrease the residual perturbation. To determine whether to increase or decrease the response matrix element, a slight change is first implemented. Next we take the corresponding sign and amplitude of the first iteration into account and then make further adjustments.

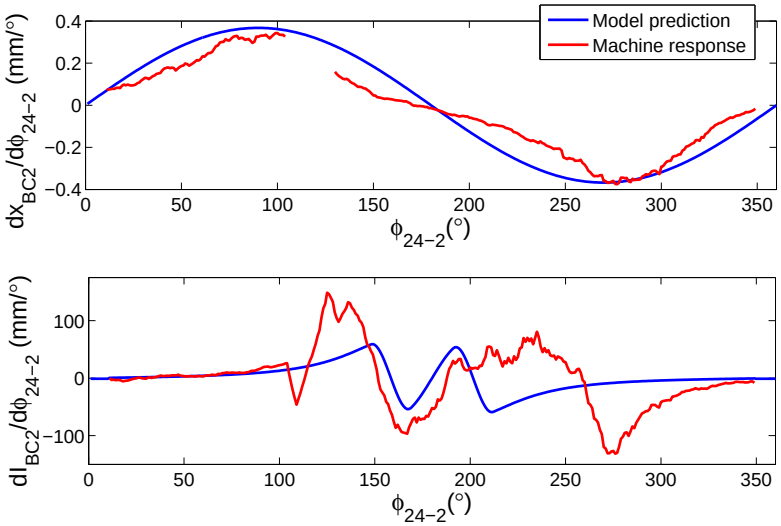



Figure 7.31: Theoretical (blue curves) and real-time machine response (red curves) for the horizontal position of the beam (upper plot) and peak current deviations (lower plot). Figure adapted from [119].

In our experiments the phase of klystron 24-2 is first set to 60°, where, according to Fig. 7.31 (upper plot) the response of the horizontal position of the beam to the phase of klystron 24-2 is $dx_{BC2}/d\phi_{24-2} = 0.32$ mm/°. The NNet is then trained to recognise a single 0.6 Hz frequency energy jitter introduced in klystron 24-1. The phase of klystron 24-2 is then changed to 300°, where $dx_{BC2}/d\phi_{24-2} = -0.3$ mm/°(according to Fig. 7.31). To test the algorithm, we assume an incorrect prediction of the model and initialise the dynamic matrix element to $dx_{BC2}/d\phi_{24-2} = -0.1$ mm/°. In this case the response is sufficiently different from the real machine response so that the full correction cannot be achieved. The algorithm then searches for adjustments that reduce the peak in the FFT to the background noise level. Figure 7.32 illustrates the process. The algorithm first applies a small positive change to $dx_{BC2}/d\phi_{24-2}$ from -0.1 mm/° to -0.08 mm/°. As the 0.6 Hz peak of the second FFT (solid red curve) is higher than the first (dashed blue curve), the algorithm implements a correction of opposite sign. The difference in amplitude between the first two FFTs is also taken into account by the algorithm to compute the next correction.

This experiment showed that inaccuracies in the correction, due to errors in the prediction of the beam energy deviation, or in the estimate of the response matrix elements, can be successfully corrected by a simple online algorithm. The method can take a few thousands bunches before the correction is completed. However, if the algorithm operates at 120 Hz (beam repetition rate), this is only a matter of seconds.

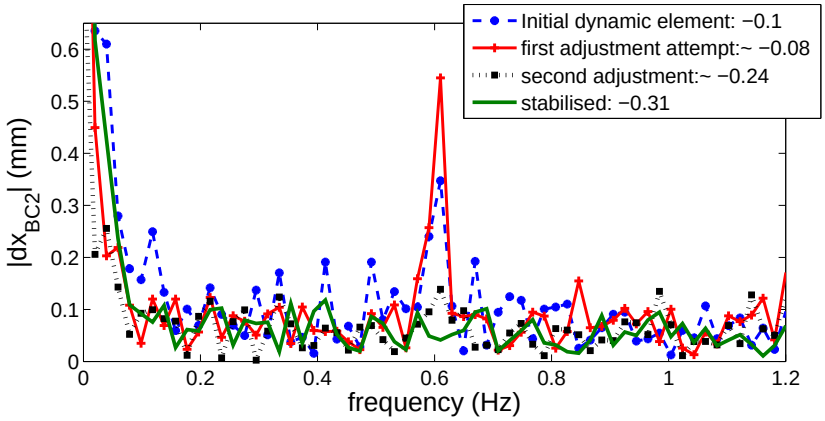



Figure 7.32: FFT of the deviation of the beam position at BC2 for real-time adjustments of the response of the beam position to the phase of klystron 24-2. A change in the phase of klystron 24-2 (from 60° to 300°) causes a change in the response of the beam position to this actuator, causing a residual jitter (blue curve). The algorithm makes a first attempt to adjust this response, leading to an increased deviation in the beam position (red curve). A second attempt decreases the amplitude of residual deviation (dotted black curve), while the last attempt (green curve) brings the residual amplitude of the deviation down to the white noise level. Figure adapted from [119].

## 7.6 Discussion

In this chapter we discussed how to minimise residual jitter using a feedforward system that anticipates deviations for fixed settings of the machine. The experiments performed at the Australian Synchrotron demonstrated the ability of the NNet to predict perturbations occurring in the klystrons; this provides a useful augmentation of feedforward control by a PI controller. Similarly, the results obtained at the LCLS have shown that the NNet hybrid controller can simultaneously operate the control of the energy and bunch length. A comparison of the combined feedback-feedforward controller with a simple PI controller also showed enhanced performance of the combined control system.

It was also shown that it is possible to derive the response matrix elements of the machine using a simulation code (LiTrack, Elegant and the FMS). However, because the response matrix elements cannot be derived with a high level of precision, it is necessary to adjust the matrix elements to ensure the accuracy of the control. This can be achieved with a second NNet whose output would slightly modify the response matrix elements. A possible structure would consist of providing the NNet with records of

the response matrix elements and the corresponding residual perturbation in the beam parameters. The outputs would be the values of $\Delta M_{ij}$, corresponding to the correction to the response matrix elements $M_{ij}$.

Current limitations in testing the correction at higher frequencies are due to the slow response of the actuators via the Matlab Process Variables (PVs). To study the performance of the system at higher frequencies will require the algorithm to be implemented on the low level RF control units using compiled C/C++ code.

# 8 Using neuro-evolution to build an adaptive control system

## 8.1 Motivation and objectives

Previous chapters demonstrated how NNets can be used to predict and control perturbations in an electron beam. However, as discussed in Chapter 7, there is a need to develop a system that can take into account the dynamic response matrix. Ideally, it would be desirable for the system to continuously learn from its interaction with the environment. This would avoid retraining the NNet, should the jitter conditions change. To do this, we consider beam stabilisation as an optimisation problem, where the aim is to minimise the deviation in the beam parameters from their desired settings. To proceed, we develop an online optimisation tool for beam tuning (i.e. optimising the beam parameters by adjusting the settings of the accelerator) and discuss its adaptation to control problems.

Implementing an online beam tuning tool is also of great interest for FEL operation. As synchrotron radiation has a wide range of wavelengths; specific spectral components can be selected using monochromators for different beamline applications. However, in SASE FELs, a single wavelength is produced by Self-Amplified Spontaneous Emission (SASE) occurring as the beam passes the undulator magnets. The production of different wavelengths therefore requires re-optimisation of the linear accelerator parameters. Currently, operators optimise the machine parameters to meet users' requirements. This is accomplished by loading the closest known configuration and making adjustments to the machine settings. This procedure is tedious and can be very time consuming.

The present chapter shows how to build an optimisation tool based on an Artificial Intelligence (AI) system that learns the response of beam parameters to actuators. The system must be computationally inexpensive and must not require prior knowledge of the machine, except for limits placed on the actuators. We will show how such an AI system can be used to

operate feedback tasks. The system discussed in this chapter can perform optimisations in a 2-dimensional search space; however, its generalisation to an N-dimensional search space will be discussed in Chapter 9.

## 8.2 The approach

The approach used here is inspired by the work of Miikkulainen *et al.* [86]. These authors use NNets which are genetically evolved to create "game agents" that can learn from their environment (see Chapter 6). The game agents learn to develop skills in shooting, avoiding enemies and navigating in a maze. Here we make parallels between the evolution of such a game agent in a battlefield and the evolution of an "optimisation agent" in a search space. The optimisation agent must develop skills to navigate in the search space to find the global maximum of an objective function[1], while staying within specified boundaries (i.e. between the lower and upper limits of each machine parameter in the search space). This is similar to a game agent that must navigate in the battle field and develop skills to win in combat. The basic idea is to implement an intelligent system that can mimic the actions an operator would take to optimise beam parameters in real-time, when the only available knowledge of the system is the actuators' limits. In what follows, the optimisation agent is equipped with sensors that will serve to provide inputs to the NNet. The decisions the agent makes (i.e. change in the actuators' settings) are dictated by the NNet outputs.

To perform an optimisation one would explore the local search space by making small changes, which tend to improve the value of an objective function. Moreover, one would choose to explore a given region of the search space based on past trials. This region is changed when a local maximum is encountered that is not optimum[2]. We can summarise the actions of an operator as follows:

1. Ensure that the machine remains within the actuators' limits;
2. Move in the direction that is most likely to increase the value of the objective function;

[1] In order to solve a control problem or beam tuning optimisation problem, we use an objective function $H$, which is a function of the parameters to be optimised, subject to certain constraints.

[2] To determine whether a solution is optimum, one must set a goal (or value) for the objective function $H$. For example, in the case where the transmission $T$ (i.e. the number of particles transmitted from the start to the end of the accelerator) is to be optimised (see Sec. 8.4), we could set 95% as a goal. In this case the value of the objective function must be $T \geq 95\%$ (since $H = T$).

3. Jump to another location in the search space when the optimisation agent is trapped at a local maximum.

The goal is to evolve a system with these characteristics. In order to ensure that each of the three actions is correctly addressed, the problem is decomposed into three distinct parts, each of which is handled by a substructure of the NNet. This is similar to the structure of the brain, where different regions have different functionalities. Our three structural parts are called: *the boundary navigation substructure*, *the performance navigation substructure* and *the local maximum avoidance substructure*. The first and second substructures handle actions 1 and 2, while actions 3 is handled by the third substructure. Because not crossing boundaries is essential, the boundary navigation substructure will take over the two other substructures whenever the optimisation agent is within a chosen threshold distance from a boundary. The performance navigation substructure is otherwise the active substructure, i.e. the response of the optimisation agent at a given time step is given by only one of the three substructures, which is referred to as the "active" substructure. The local maximum avoidance substructure is only activated when the density of trial points exceeds a given threshold. When this occurs, re-localisation of the optimisation agent's position within the whole search space takes place. This decision process is illustrated in Fig. 8.1.

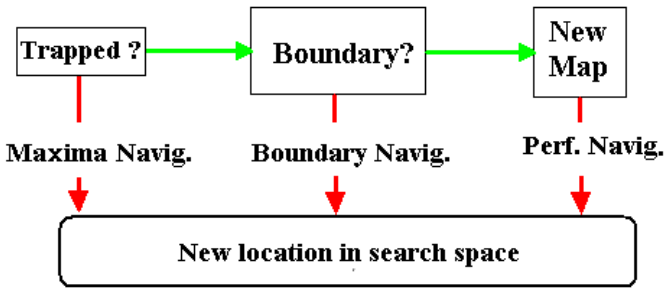



Figure 8.1: Decision processes of the optimisation agent. Before the optimisation agent takes a step in the search space, it verifies that it is not trapped at a local minimum; this task is handled by the *the local maximum avoidance substructure*. If not, it checks for the proximity of boundaries. If a boundary is not in the local neighborhood (i.e. within a given radius of the optimisation agent), the agent's next step is based on the decision of the NNet's *performance navigation substructure*. Otherwise, the next step is calculated by the *boundary navigation substructure*.

Each substructure is discussed in detail below, but first, we must establish the basic parameters that define the state of the optimisation agent in the search space; these parameters are common to all substructures. For visualisation purposes the system is depicted using a two-dimensional search space. Figure 8.2 (a) shows the optimisation agent's position in the search space at step $k$ of a time series $k \in [1..N]$. The state of the agent is defined by its position $\vec{P}_k = (x_k, y_k)$ (corresponding to the values of actuators X and Y) and the angle $\theta_k$, giving the direction faced, as illustrated in Fig. 8.2 (a).

For the agent to find optimum settings, the outputs of the NNet must be related to the next position in the search space, $\vec{P}_{k+1} = (x_{k+1}, y_{k+1})$. In analogy with a game agent, we limit the step size that can be taken by the optimisation agent. For this reason it is convenient for the outputs of the NNet to correspond to "move forward" and "turn" actions from the current position $\vec{P}_k$ in the search space. The NNet's outputs will be denoted by $r_k$ and $\phi_k$, corresponding to the step size and direction, respectively, as illustrated in Fig. 8.2 (a). With this definition, the new state parameters of the optimisation (at step $k+1$) can be expressed in terms of the NNet's outputs $\phi_k$ and $r_k$:

$$\begin{cases} x_{k+1} = x_k + r_k cos(\theta_k + \phi_k) \\ y_{k+1} = y_k + r_k sin(\theta_k + \phi_k) \\ \theta_{k+1} = \theta_k + \phi_k. \end{cases} \tag{8.1}$$

Note that the outputs of the NNet range from -1 to 1 for the sigmoid activation function used in NEAT. In practice, the outputs of the NNets need to be re-scaled such that $\phi_k \in [\phi_{min}, \phi_{max}]$ and $r_k \in [r_{min}, r_{max}]$. In our study we chose $\phi_k \in [0, \pi/2]$, with $r_{min}$ and $r_{max}$ corresponding to 2% and 5% of the size of the search space. For example, if the lower and upper values of an actuator are denoted by $X_{min}$ and $X_{max}$, respectively, then $r_{min} = 0.02(X_{max} - X_{min})$ and $r_{max} = 0.05(X_{max} - X_{min})$. The search aims to explore the local neighborhood from the current location, rather that the whole search space at once (i.e. the value of the actuators can only by changed by a limited amount at each step). This principle is key to the methodology since it reduces dramatically the amount of data that must be considered and therefore the computation time at each iteration.

## 8.3 The three substructures of the NNet

### 8.3.1 Substructure one: boundary navigation

#### The necessary inputs

First, it is necessary to determine the inputs that must be provided to avoid crossing boundaries. Just as a game agent needs to detect the presence of a wall, an optimisation agent must have sensors that detect the proximity of boundaries. Our optimisation agent is equipped with five sensors to detect the proximity of boundaries; one in front, two at ±45° (left and right) and two at ±90° (left and right) as depicted with blue arrows in Fig. 8.2 (b). Each sensor provides the NNet with an input ranging from 0 (when there is no boundary detected) to 1 (when the current position is already on the boundary). The $i^{th}$ input at step $k$, denoted by $I_{i,k}$ (with $i = 1, 2, .., 5$ depicted in Fig. 8.2 (b)) is given by:

$$I_{i,k} = 1 - \frac{d_{i,k}}{R}, \tag{8.2}$$

where $d_{i,k}$ is the distance of the optimisation agent in the search space to the boundary in the direction of the $i^{th}$ sensor at step $k$ (see Fig. 8.2), and $R$ is the maximum distance at which a boundary can be detected by the sensor.

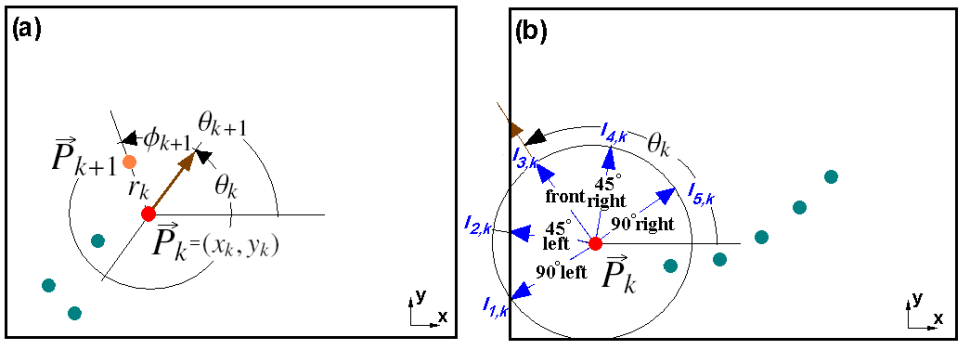



Figure 8.2: (a) State parameters of the optimisation agent in a 2D search space. At step $k$ the state of the optimisation agent is defined by its position $(x_k, y_k)$ in the search space (shown with a red dot) and the direction it is facing, specified by the angle $\theta_k$ (indicated with the brown arrow). The new position $\vec{P}_{k+1}$ is calculated from the NNet's outputs $\phi_{k+1}$ and $r_{k+1}$, giving the new direction and step size (see Eq. (8.1)). (b) The NNet receives inputs proportional to the distance to the boundary, in front, at ±45° (on the left and right) and ±90° (on the left and right). For example, the distance of the game agent to the boundary in the direction of sensor number 2, and at step $k$ is $d_{2,k}$. Figure adapted from [120].

### Building a fitness function

In what follows we describe the fitness function discussed in Sec. 6.3.3, which is implemented using a genetic algorithm[3] to evolve the boundary-based substructure. We need to consider situations that can be encountered by the optimisation agent during its navigation through the search space. At each step, the new position can be classified as:

**(A)** Away from all boundaries (within the search space);

**(B)** Facing a boundary (in this case only input $I_{3,k}$ is non-zero);

**(C)** Close to, but not facing a boundary;

**(D)** Facing a corner (intersection of two boundaries); or

**(E)** Anywhere outside the region defined by the upper and lower limits of the search space parameters, i.e. outside the boundaries.

For cases **(A)** to **(D)**, the position of the optimisation agent is contained within the boundaries, otherwise we must consider case **(E)**. The fitness function takes these different cases into account and rewards or penalises the actions of the NNet accordingly, as discussed in Secs. 6.3.3 and 6.2.3. To do this, we consider what decisions an operator would make in the above situations and formulate them to attribute reward and penalty units for the agent's actions. At each step taken by the optimisation agent, the NNet receives a reinforcement signal ranging from -1 to 1, which reflects how well the agent has performed in the above situations. Since case **(A)** is handled by the performance-based substructure, the boundary navigation substructure must handle situations **(B)** to **(E)** only.

Let us first consider the presence of a boundary detected ahead (case **B**). This case corresponds to inputs $I_{1,k} = 0, I_{2,k} = 0, I_{4,k} = 0, I_{5,k} = 0$ and $I_{3,k} \neq 0$ (inputs at the $k^{th}$ step in a time series defined by Eq. (8.2)), i.e. only the sensor pointing in the direction $\theta_k$ detects a boundary. To make sure the region close to the boundary is explored (the agent must not cross a boundary but must be able to get sufficiently close to it), the optimisation agent should turn through an ideal angle $\phi_{id,k}$, proportional to its proximity to the boundary. The ideal angle will be zero when the boundary is detected at a distance $d_{3,k} = R$ from the agent and equal to $\phi_{max}$ (the maximum angle

[3] We evolve the boundary navigation substructure of the NNet using a genetic algorithm, specifically using the NEAT technique discussed in Sec. 6.3. This method encodes the structures of the NNets and includes mutations, such as adding or removing connections and nodes from the NNet.

the agent can turn through) when the agent is already on the boundary, i.e. $d_{3,k} = 0$. With these conditions, the ideal angle at step $k$ is given by:

$$\phi_{id,k} = \left( 1 - \frac{d_{3,k}}{R} \right) \phi_{max}. \tag{8.3}$$

Now let us consider how to reward and penalise the NNet, according to the actions taken by the optimisation agent. A positive reward unit[4] is assigned if the agent turns though the ideal angle $\phi_{id}$ and a negative reward unit (penalty) is given when the difference between the ideal angle $\phi_{id,k}$ and the angle $\phi_k$ taken by the agent is equal to $\phi_{max}$. According to this, the reward or penalty received by the NNet in relation to the angle it turns though is given by:

$$f_{B,\phi_k} = 1 - \frac{2||\phi_k| - \phi_{id,k}|}{\phi_{max}}, \tag{8.4}$$

where $\phi_{id,k}$ is the ideal angle specified by Eq. (8.3). Note that $\phi_k$ can be of either sign, which is reflected by using the absolute value of $\phi_k$ in Eq. (8.4). This is because the optimisation agent can turn left or right, since no boundary is detected on either side. There is therefore no reason to reward/penalise the agent for turning in one direction rather than the other.

The NNet must also be rewarded or penalised for the step size it takes. Intuitively, the ideal step size corresponds to a maximum $r_k = r_{max}$ when $d_{3,k} = R$ and a minimum $r_k = r_{min}$ when $d_{3,k} = 0$. We can write the ideal step size as:

$$r_{id,k} = \frac{d_{3,k}}{R}(r_{max} - r_{min}) + r_{min}. \tag{8.5}$$

The optimisation agent should also be given a positive reward unit when it takes a step of ideal size ($r_k = r_{id,k}$); likewise a negative reward unit should be given when the difference between the size of the step $r_k$ and the ideal step $r_{id,k}$ is equal to $r_{max} - r_{min}$. The reward or penalty received by the NNet for the step size it therefore given by:

$$f_{B,r_k} = \left( 1 - \frac{2|r_k - r_{id,k}|}{r_{max} - r_{min}} \right). \tag{8.6}$$

The reward or penalty received by the NNet at step $k$ consists of the reward received for the angle through which the optimisation agent turns and

[4] In reinforcement learning (see Chapter 6), the NNet is rewarded by receiving positive units or penalised by receiving negative units. For example a game agent would score -2 for being shot by an enemy and +4 for killing an enemy. The sum of individual actions gives the score of the game agent at the end of its evaluation. In the above case the NNet would score two reward units.

the size of the step it takes. When equal weights are given to both contributions, the reward or penalty received by the NNet is given by (using Eqs. (8.4) and (8.6)):

$$\begin{aligned} f_B(k) &= \frac{1}{2} f_{B,\phi_k} + \frac{1}{2} f_{B,r_k} \\ &= \frac{1}{2}\left( 1 - \frac{2||\phi_k| - \phi_{id,k}|}{\phi_{max}} \right) + \frac{1}{2}\left( 1 - \frac{2|r_k - r_{id,k}|}{r_{max} - r_{min}} \right), \end{aligned} \tag{8.7}$$

where $\phi_{max} = \pi/2$ is the maximum angle the optimisation agent can turn at each step (from its current direction $\theta_k$). A boundary is detected on one side when the sum of the inputs exceeds the sum of the inputs on the other side. Moreover, either $I_{1,k}$ or $I_{5,k}$ must be zero. If this is not the case and $d_{3,k} = 0$ the agent is on the boundary and the reward will be computed according to Eq. (8.7). If $d_{3,k} \neq 0$ the agent encounters a corner and case **(D)** is relevant. The following conditions define the presence of a boundary on either the left or the right:

$$\begin{cases} Left: & I_{1,k} + I_{2,k} > I_{4,k} \quad \text{and} \quad I_{5,k} = 0 \\ Right: & I_{2,k} < I_{3,k} + I_{4,k} \quad \text{and} \quad I_{1,k} = 0. \end{cases} \tag{8.8}$$

When a boundary is detected on either the left or right (case **C**), the NNet receives a reward when the optimisation agent turns away from the boundary and a penalty when it turns towards it. We express this by introducing the ideal angle, which is proportional to the distance the optimisation agent is from the boundary. The ideal angle is expressed as:

$$\phi^{\perp}_{id,k} = \phi_{max}\left( 1 - \frac{d_{\perp,k}}{r_{max}} \right), \tag{8.9}$$

where $d_{\perp,k}$ is the distance perpendicular to the boundary from the current position and $r_{max}$ is the maximum step size the optimisation agent can take. At step $k$ the reward or penalty can be expressed as:

$$f_C(k) = -\delta_{0,DT}\frac{|\phi_k|}{\phi_{max}} + \delta_{1,DT}\left( 1 - \frac{|\phi_k - \phi^{\perp}_{id,k}|}{\phi_{max}} \right), \tag{8.10}$$

where $DT = 0$ if a turn is taken towards a boundary and $DT = 1$ otherwise; here $\delta_{z,DT}$ $(z = 1, 0)$ is the Kronecker delta symbol.

In the case of a corner (case **D**), the most appropriate response can be characterised by a maximum angle $\phi_k = \phi_{max}$, in combination with a minimum step size, since turning through a small angle with a large step size increases the chances of falling outside the boundaries. If a corner is encountered at step $k$, the reward or penalty is given by:

$$f_D(k) = \frac{3}{4}\left( 2\frac{|\phi_k|}{\phi_{max}} - 1 \right) + \frac{1}{4}\left( 1 - 2\frac{r_k - r_{min}}{r_{max} - r_{min}} \right). \tag{8.11}$$

Intuitively, turning through an angle close to the ideal angle (8.9) should have more importance than the exact step size; consequently we introduce the empirical factors $\frac{3}{4}$ and $\frac{1}{4}$ in Eq. (8.11), which assign three times more importance to the angle compared to the step size. The agent is free to turn either left or right, which is accounted for by using the absolute value of $\phi_k$ in Eq. (8.11).

Finally, when case **(E)** is relevant, the NNet receives a penalty unit ($out(k) = 1$). If the optimisation agent remains within the boundaries ($out(k) = 0$), the NNet will receive a reward unit. This can be expressed as follows:

$$f_E(k) = (-1)^{\delta_{1,out(k)}}. \tag{8.12}$$

At each step in the search space, the position of the optimisation agent is classified according to cases **(A)**,**(B)**,**(C)**,**(D)** or **(E)**. Therefore, the fitness $F_{indiv.}$ of a NNet (or individual[5]) over a number of steps is equal to the averaged sum of the fitnesses for each case. Because case **(A)** does not have any reward or penalty associated with it (since no boundary is detected), its contribution to the fitness of an individual over any number of steps is zero. The fitness function can be expressed by combining Eqs. (8.7), (8.10), (8.11) and (8.12) to give:

$$\begin{aligned} F_{indiv.} = {} & \frac{b}{N_{front}} \sum_{n_f=1}^{N_{front}} f_B(n_f) + \frac{c}{N_{side}} \sum_{n_s=1}^{N_{side}} f_C(n_s) + \\ & \frac{d}{N_{corner}} \sum_{n_c=1}^{N_{corner}} f_D(n_c) + \frac{e}{N_{out}} \sum_{n_o=1}^{N_{out}} f_E(n_o), \end{aligned} \tag{8.13}$$

where $N_{front}$ is the number of times a boundary is faced (case **B**), $N_{side}$ is the number of times a boundary is detected but not faced directly (case **C**), $N_{corner}$ is the number of times a corner is faced (case **D**), and $N_{out}$ is the number of times the optimisation agent steps outside the boundaries of the search space (case **E**). The corresponding weighting factors $b, c, d$ and $e$ take values between 0 and 1 and can be adjusted to enhance the importance of a certain behavior. For example, it is more important to remain within boundaries than to take the ideal turning angle when facing a corner; therefore we choose $e > d$. Also, since the fitness ranges from $-1$ to 1, we have $b + c + d + e = 1$, and each term in the sum ranges from $-1$ to 1. In

[5]In Sec. 6.3, we define an individual as a NNet in a population, which is evolved using the NEAT technique.

what follows we describe the different terms and their importance in more detail. Introducing Eqs. (8.7), (8.10) and (8.12) into Eq. (8.13), the fitness function of an individual is given by:

$$\begin{aligned} F_{indiv.} &= \frac{2b}{N_{front}} \sum_{n_f=1}^{N_{front}} \left[ \frac{1}{2}\left( 1 - \frac{2||\phi_k|-\phi_{id,k}|}{\phi_{max}} \right) + \frac{1}{2}\left( 1 - \frac{2|r_k-r_{id,k}|}{r_{max}-r_{min}} \right) \right] \\ &+ \frac{c}{N_{side}} \sum_{n_s=1}^{N_{side}} \left[ -\delta_{0,DT} \frac{|\phi_k|}{\phi_{max}} + \delta_{1,DT}\left( 1 - \frac{|\phi_k-\phi^{\perp}_{id,k}|}{\phi_{max}} \right) \right] \\ &+ \frac{d}{N_{corner}} \sum_{n_c=1}^{N_{corner}} \left[ \frac{3}{4}\left( 2\frac{|\phi_k|}{\phi_{max}} - 1 \right) + \frac{1}{4}\left( 1 - 2\frac{r_k-r_{min}}{r_{max}-r_{min}} \right) \right] \\ &+ \frac{e}{N_{out}} \sum_{n_o=1}^{N_{out}} (-1)^{\delta_{1,out(k)}}. \end{aligned} \tag{8.14}$$

### Running NEAT

What level of fitness should an individual achieve in order to be considered successful in a navigation task? Whilst the fitness function expressed in Eq. (8.14) describes the "ideal" behavior it is not necessary to achieve a fitness of 1 to succeed in a navigation task. In fact, the main goal is to remain within the boundaries at each step. If this criterion can be satisfied, regardless of the starting point $(x_0, y_0)$ and direction $\theta_0$ in the search space, the NNet structure is functional. For a number of configurations $N_{conf}$ of the starting points $(x_0,y_0)$ and angles $\theta_0$, the sum of reward units accumulated by the NNet for case (**E**) must satisfy the condition:

$$\sum_{n_t=1}^{N_{conf}} f_E(n_t) = e. \tag{8.15}$$

Thus, a NNet is successful as long as the condition (8.15) is met. The case where $F_{indiv.} < e$ can occur when the sum of the contributions from the terms $b$, $c$ and $d$ in Eq. (8.14) is negative. This happens when, for example, a NNet accumulates negative rewards because the optimisation agent turns towards a boundary instead of turning away. Whether the optimisation agent finds the ideal angle to turn through when it encounters a boundary, or corner, is therefore not the key criterion, but an indication of the fitness of the NNet (i.e. the fitter the NNet the more the optimisation agent will turn through the ideal angle with the ideal step size). In order to avoid a NNet getting "lucky", because the optimisation agent never has to avoid

facing a boundary or a corner, it is necessary to operate the selection over different starting points and angles. The fitness of an individual tested over $N_{conf}$ is defined as the average of the fitnesses obtained for each configuration, i.e.

$$F_{indiv} = \frac{1}{N_{conf}} \sum_{n_t=1}^{N_{conf}} F_{indiv.}(n_t). \tag{8.16}$$

In our evaluation the optimisation agent is typically required to start from 40 different locations in the 2D search space (the search space is normalised such that its values range from 0 to 1 for each parameter). These start locations were selected such that the optimisation agent is located close to a boundary or corner at the start of its evaluation. The coordinates $(x_0, y_0)$ of the chosen start locations are given by four sets of ten points each: $(x_0 = 0.05, y_0 = \mathbf{b}(l))$, $(x_0 = 0.95, y_0 = \mathbf{b}(l))$, $(x_0 = \mathbf{b}(l), y_0 = 0.05)$ and $(x_0 = \mathbf{b}(l), y_0 = 0.095)$, where $\mathbf{b}(l)$ is the $l^{th}$ element of the coordinate vector $\mathbf{b}$ = [0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95]. For each start position, the start angle $\theta_0$ was chosen randomly.

#### Simulation results

In this section we present results that were obtained when NEAT was evaluated using two different approaches. First, we evolve the NNet from the knowledge-based (KB) structure illustrated in Fig. 8.3 using the "if" building block described in Sec. 6.5. Only the NNet's weights are evolved with NEAT. Second, we evolve a NNet from a minimum structure (without any hidden neurons). In this case the initial NNet is fully connected, i.e. each node in the input layer is connected to all output nodes.

When the knowledge-based NNet structure was evolved, we found that even with a single configuration and one generation, it is possible for the optimisation agent to remain within the boundaries at each step. NEAT was run nine times for populations containing between 100 and 500 individuals; for each run the corresponding percentages of successful NNets are listed in Table 8.1. The percentage of NNets capable of accomplishing the task, ranges from 1.2% to 1.7%. The fact that some individuals in the first generation remain within the boundaries indicates that the structure of the NNet is already operational, i.e. there is no need to evolve the number of hidden nodes and their connections.

NEAT was then run five times to evolve 200 individuals over 800 generations and five times to evolve 700 individuals over 800 generations. These numbers were chosen so that the smaller population ensures initial topo-

Table 8.1: Percentage of individuals in the initial population of NNets for which the optimisation agent successfully remains within the boundaries. During each test the optimisation agent is required to detect a boundary 100 times.

| Num. Individuals | 100 | 150 | 200 | 250 | 300 | 350 | 400 | 450 | 500 |
|---|---|---|---|---|---|---|---|---|---|
| Success rate [%] | 1.20 | 1.13 | 1.60 | 1.67 | 1.23 | 1.34 | 1.25 | 1.20 | 1.36 |

logical diversity, while the larger population was not significantly more computationally intensive. The number of generations was chosen to ensure that the populations have enough time to evolve their structure. The maximum fitnesses are listed in Table 8.2. For each run, the NNets were evaluated with four configurations (i.e. starting position and angle) corresponding to the optimisation agent facing each corner. According to Table 8.2, increasing the number of individuals in the initial population increases (on average over the five runs) the fitness from 0.69 to 0.71. The maximum fitness varies slightly between runs, but is consistently between 0.70 and 0.74. An example of a typical run is given in Fig. 8.4 (blue curve) and shows that most of the improvement in fitness is achieved within the first 100 to 200 generations. Figure 8.5 shows a typical path followed by an optimisation

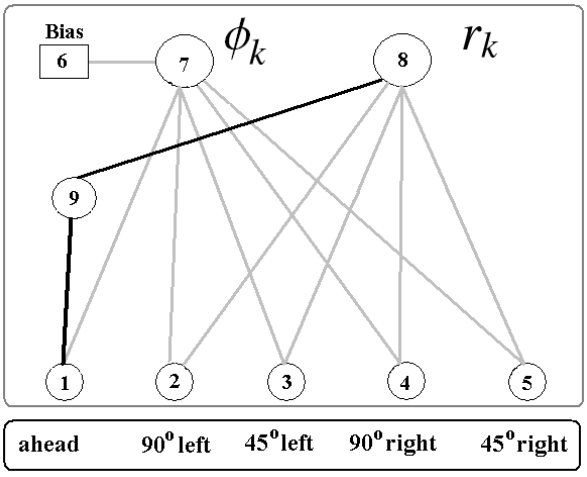



Figure 8.3: Knowledge-based structure of the NNet exploiting the "if" structure described in Sec. 6.5. The five inputs (nodes 1 to 5) are calculated according to Eq. (8.2). The two outputs correspond to the "turn" and "move forward actions" of the optimisation agent. This structure is inspired by the structure used for the NERO video game [121].

agent navigating within the 2D search space. As one can see from this figure the optimisation agent does not follow a straight line when it is away from boundaries. This is due to a non-zero bias for the "turn" output of the NNet.

Table 8.2: The fitnesses attained for five runs of NEAT over 800 generations with the knowledge-based structure. An increase in the number of individuals in the initial generation shows a slight improvement in the fitness. Further, this improvement is only achieved when the NNet is evolved over a larger number of generations.

| | 200 Individuals | | 700 Individuals | |
|---|---|---|---|---|
| Run | Fitness | Generation | Fitness | Generation |
| 1 | 0.67 | 207 | 0.70 | 588 |
| 2 | 0.70 | 154 | 0.74 | 915 |
| 3 | 0.71 | 775 | 0.70 | 827 |
| 4 | 0.72 | 190 | 0.72 | 844 |
| 5 | 0.67 | 687 | 0.70 | 573 |
| Average: | 0.69 | 403 | 0.71 | 750 |

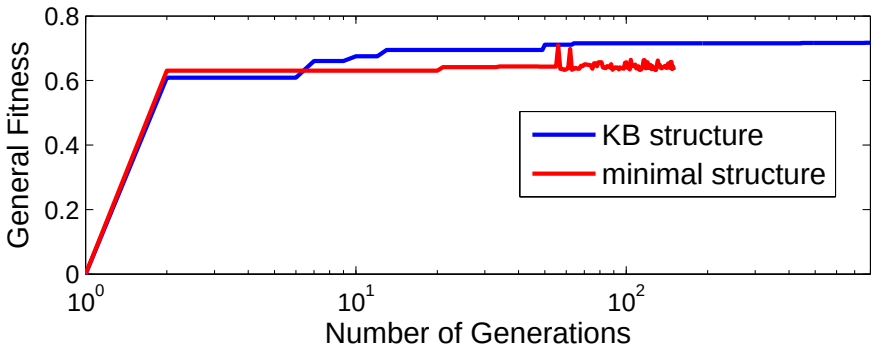



Figure 8.4: Evolution of fitness through generations for the knowledge-based structure and the minimal structure. NEAT evolved the weights of 200 individuals over 800 generations for the KB structure (blue curve). The structure of 300 individuals was evolved over 150 generations for the minimal structure (red curve).

In the second approach NEAT was run five times. The initial population contained 300 individuals that were evolved over 150 generations. These numbers were chosen based on the time required to perform the simulations (a typical run takes around 20 hours) and the fact that most of the

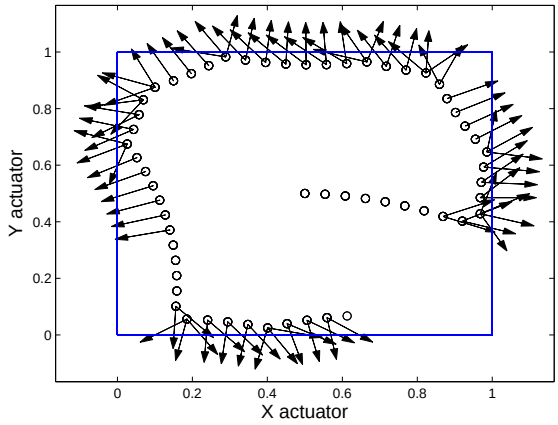



Figure 8.5: Example of a NNet succeeding in a navigation task. This utilised the knowledge-based structure. The arrows show the direction in which a boundary is detected. The search was started from the center of the search space at (0.5,0.5). Figure adapted from [120].

fitness improvement appears within the first 100 to 200 generations. The number of individuals was chosen to ensure topological diversity, while the number of generations was limited to 150 as the computation time increases rapidly with generational number, i.e. the number of connections and nodes in the NNet increases at each generation. In this approach the whole structure is evolved, which results in the appearance of species[6] with more complex structures, and therefore an increase in the computation time to evaluate their performance. Table 8.3 presents results obtained for the five runs.

Figure 8.6 shows a typical structure obtained with NEAT. After 34 generations a fitness of 0.65 was achieved (see Fig. 8.4). The initial population (first generation) of NNets had no hidden neurons; all inputs and the bias were connected to both output neurons. During evolution, apart from adjusting the weights of the existing connections, NEAT made two structural modifications. First, a hidden neuron (node 9 in Fig. 8.6) was added to connect the "boundary ahead" input node (node 1) to the "move forward action" (node 8). In this figure the corresponding connections are shown in black. Because the inputs range from 0 to 1 and the hidden node is connected to a bias, the hidden node's output will necessarily be below 0.75, according to the definition of the sigmoid transfer function (Sec. 6.2). Moreover, the connection of the hidden neuron to the output node also has a weight that

[6]Every individual in a population of NNets is assigned to a "species" (see Sec. 6.3.2) based on topological similarities (i.e. individuals of a species have similar numbers of connections, nodes, weights, etc).

ranges between 0 and 1. This structure therefore acts as a "moderator" that decreases the step size (output $r_k$) taken by the optimisation agent when a boundary is detected ahead. Evolution chose this structure because the optimisation agent tends to be more at risk of crossing a boundary when it directly faces it. In this way, the optimisation agent is given more time to proceed in a different direction.

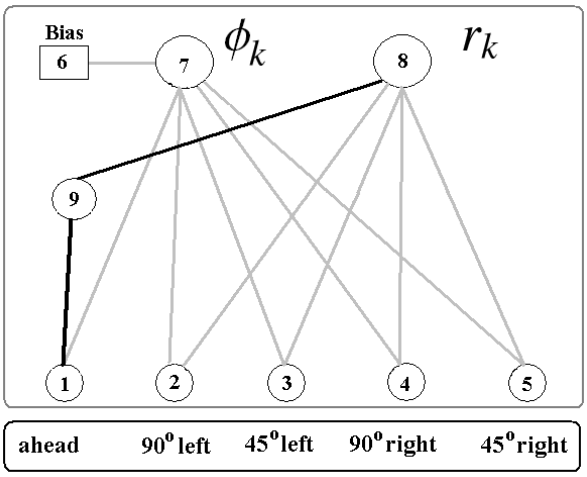



Figure 8.6: Structure evolved with NEAT. The gray connections are already present in individuals in the initial generation. The black connections, as well as node 9, were added by NEAT. All initial connections were unchanged by NEAT, except for the connection from node 6 (bias) to node 8 (move forward output), which was removed.

NEAT also de-activated the connection from the bias to the "move forward" output; whereas the connection from the bias to the "turn action" remains active. This evolution can be interpreted using the expression for the fitness function. When navigating within the search space, boundaries are most likely to be classified as being on the side (case **C**) or at a corner (case **D**), rather than being in front (case **B**); the condition for a boundary to be ahead is that only node 1 in Fig. 8.6 has a non-zero input. Therefore, the third term on the right hand side of Eq. (8.14) will have the most influence in the evolution of the NNet's structure. The positive reward term related to the step size (output node 8, Fig. 8.6) shows a linear dependence in $1 - r_k/(r_{max} - r_{min})$, with $r_{max} - r_{min} = R$ and which is similar to the expression used to define the inputs (Eq. (8.2)). In other words, there is a linear relation linking the positive reward in the fitness function and the inputs to the NNet. The nodes that constitute the NNets also have a sigmoid transfer function; the addition of a bias tends to make the system operate in the

non-linear region of the sigmoidal function. This is contrary to the assumption that there is a linear relation between the inputs and positive rewards, which explains why evolution tends to deactivate that connection. On the other hand, the positive rewards related to the turning angle are not proportionally related to the inputs, which explains why the bias connection remains activated.

Table 8.3: Finesses for five runs of NEAT starting from the minimum structure. Both the weights and the structure of a population containing 300 individuals was evolved over 150 generations.

| 300 Individuals | | |
|---|---|---|
| Run | Fitness | Generation |
| 1 | 0.65 | 34 |
| 2 | 0.72 | 12 |
| 3 | 0.70 | 32 |
| 4 | 0.69 | 33 |
| 5 | 0.71 | 56 |

For both the knowledge-based structure and the structure evolved with NEAT, the maximum fitness function did not exceed 0.76, whilst the maximum fitness that could in principle be reached is 1. This can be understood as follows. The fitness function is built from the contributions of cases **(B)**,**(C)**,**(D)** and **(E)**. In particular, cases **(C)** and **(D)** (see Sec. 8.3.1) contribute significantly to the fitness function, because they arise statistically more often than case **(B)**. When the agent is close to a boundary (case **C**), it should turn away through an angle proportional to the distance to the boundary. However, when a corner is faced (case **D**), the optimisation agent should turn through the maximum angle. Because the NNet only receives inputs proportional to the distances from the boundaries it cannot decide between these two cases, and there is a contradiction between both requirements. Evolution attempts to reconcile this contradiction, which impacts on the optimisation of the fitness function (Eq. (8.14)). The contribution of the corners ranges between -0.05 and -0.1, whilst the maximum contribution should be 0.2. This explains why the maximum fitness ranges between 0.70 and 0.75. To correct this, the second and third term terms on the right hand side of Eq. (8.14), corresponding to cases **(C)** and **(D)**, should be written in such a way to avoid this contradiction.

### 8.3.2 Substructure two: performance-based navigation

#### Basic approach

When the optimisation agent navigates within the search space, it must be provided with relevant information to decide which direction is more likely to improve the value of an objective function[7]. A simple and intuitive approach consists of providing an input that informs the NNet whether the value of the objective function has increased or decreased since the last step. This input is written as a difference between the values of the objective function $H$ at steps $k$ and $k-1$, normalised by the maximum decrease $\Delta H_{max}$ that can occur, i.e.

$$Input = \Delta H(k)/\Delta H_{max} = (H(k) - H(k-1))/\Delta H_{max}. \tag{8.17}$$

Intuitively, if the value of the objective function increases ($\Delta H(k) > 0$), the optimisation agent should take a step of maximum size, i.e. $r_k = r_{max}$ and keep going in the same direction, i.e. $\phi_k = 0$. Similarly, if the value of the objective function decreases ($\Delta H(k) < 0$), the optimisation agent should take a step of minimum size, i.e. $r_k = r_{min}$ and turn through an ideal angle $\phi_{id,k}$, defined as follows. When $\Delta H(k)/\Delta H_{max} = -1$ (maximum decrease that can occur), the optimisation agent should turn through a maximum angle $\phi_{max}$. When $\Delta H(k)/\Delta H_{max} = 0$, the value of the objective function neither decreases or increases, an angle of zero is applied. Accordingly, the ideal angle is given by:

$$\phi_{id,k} = \pm\phi_{max}\frac{\Delta H(k)}{\Delta H_{max}}, \tag{8.18}$$

where the ideal angle can have either sign, since the optimisation agent can decide to go either left or right. Based on these considerations, we build a fitness function that includes two terms, one for $\Delta H(k) > 0$ and a second for $\Delta H(k) < 0$:

$$\begin{aligned} F_{indiv.} = &\sum_{k=1}^{N}\Big[\ \delta_{1,p(k)}\Big(\ \tfrac{1}{2}\tfrac{r_{max}-r_k}{r_{max}-r_{min}} + \tfrac{1}{2}\Big(\ 1 - \tfrac{|\phi_k|}{\phi_{max}}\ \Big)\Big)\ \Big] + \\ &\sum_{k=1}^{N}\Big[\ \delta_{-1,p(k)}\Big(\ \tfrac{1}{2}\tfrac{r_k-r_{min}}{r_{max}-r_{min}} + \tfrac{1}{2}\Big(\ 1 - \tfrac{|\phi_k-\phi_{id,k}|}{\phi_{max}}\ \Big)\Big)\ \Big], \end{aligned} \tag{8.19}$$

where $N$ is the total number of steps taken by the optimisation agent in evaluating the fitness of the NNet and $p(k) = sign(\Delta H(k))$. The two Kronecker

[7] The form of the objective function $H$ depends on the optimisation problem to be solved and the parameters to be optimised.

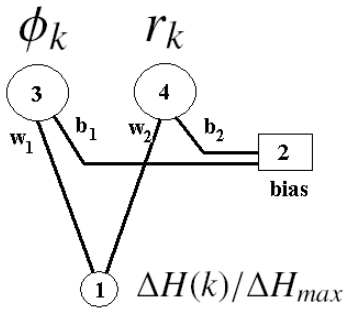



Figure 8.7: Simple structure used for the performance-based decision process. This structure decides on the new direction and step size taken by the optimisation agent based on its performance $\Delta H(k)/\Delta H_{max}$.

delta symbols correspond to an increase and a decrease in the value of the objective function, respectively. Thus, $\Delta H(k) > 0$ activates the first term of Eq. (8.19), whereas $\Delta H(k) < 0$ activates the second term; if $\Delta H(k) = 0$ no action is taken. According to the first term, the NNet scores one reward unit when the optimisation agent takes a step of maximum size $r_{max}$ and does not turn. A reward of zero is given when the optimisation agent takes a step of minimum size $r_{min}$ and turns through a maximum angle $\phi_{max}$. Based on the second term, one reward point is given if the optimisation agent turns through the ideal angle $\phi_{id,k}$. A reward of zero is given when the difference between the ideal angle $\phi_{id,k}$ and $\phi_k$ is equal to $\phi_{max}$.

Although the substructure could be evolved with NEAT it can also be built according to the structure in Fig. 8.7. As depicted in Fig. 8.7, the NNet receives $\Delta H(k)/\Delta H_{max}$ at the single input (node 1) connected to output nodes 3 and 4, corresponding to the move forward ($r_k$) and turn actions ($\phi_k$), respectively. A bias (node 2) is also connected to both outputs. For nodes with a linear activation function, the NNet outputs can be written as a function of the weights for the biases ($b_1$ and $b_2$) and the inputs ($w_1$ and $w_2$) as depicted in Fig. 8.7:

$$\begin{cases} \phi_k = w_1 \Delta H(k)/\Delta H_{max} + b_1 & : \text{node 3} \\ r_k = w_2 \Delta H(k)/\Delta H_{max} + b_2 & : \text{node 4.} \end{cases} \tag{8.20}$$

The weights $w_1$, $w_2$ and biases $b_1$ and $b_2$ can be calculated by considering the boundary conditions. When $\Delta H(k)/\Delta H_{max} = -1$ (maximum decrease in performance) the output of node 3 should be a maximum (turn through a maximum angle) and the output of node 4 should be a minimum (smallest step size). Similarly, when $\Delta H(k)/\Delta H_{max} = 1$ the output of node 3 should

be a minimum (zero turning angle) and the output of node 4 should be a maximum (maximum step size). Thus, the boundary conditions are:

$$\phi_k = \begin{cases} 0 & \text{if } \Delta H(k)/\Delta H_{max} = 1 \\ 1 & \text{if } \Delta H(k)/\Delta H_{max} = -1, \end{cases} \tag{8.21}$$

and

$$r_k = \begin{cases} 1 & \text{if } \Delta H(k)/\Delta H_{max} = 1 \\ 0 & \text{if } \Delta H(k)/\Delta H_{max} = -1, \end{cases} \tag{8.22}$$

where the angle and the step size range from 0 to 1, since we do not need to re-scale the NNet outputs to calculate the weights and biases (i.e. the outputs of the NNet range from 0 to 1 for a sigmoid activation function). Introducing conditions (8.21) and (8.22) into Eq. (8.20), we construct a system of four equations and four unknowns. The solutions for $w_1$, $w_2$, $b_1$ and $b_2$ are listed in Table 8.4. The optimisation agent can be tested using an objective function that has minima and maxima. In this way we can easily verify whether the three substructures are able to complete their specific tasks. In what follows we use the following simple test function[8]:

$$H(x,y) = \frac{sin(6\pi x)}{x} - \frac{sin(7\pi y)}{2y}. \tag{8.23}$$

Table 8.4: Connection genes for the structure depicted in Fig 8.7, corresponding to a fitness of 1 (see Eq. (8.19)). The first two rows identify the start and end nodes for each connection. The third row gives the weights according to Eq. (8.20) and the conditions given by Eqs. (8.21) and (8.22).

| | | | | |
|---|---|---|---|---|
| Connection from | 1 | 1 | 2 | 2 |
| Connection to | 3 | 4 | 3 | 4 |
| Weight | $-\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{2}$ |

An example of a run using the performance navigation substructure is shown in Fig. 8.8. Although the optimisation agent can find its ways towards a maximum the method is not optimal. Because the optimisation agent only takes the last step into consideration, it can sometimes approach the global maximum, but not reach it without taking multiple turns. This suggests that we need a more sophisticated approach to navigating in the 2D search space.

[8]Tests were performed for a range of smooth objective functions. Here we present the results for a typical function, which has local maxima and maxima, and a global maximum.

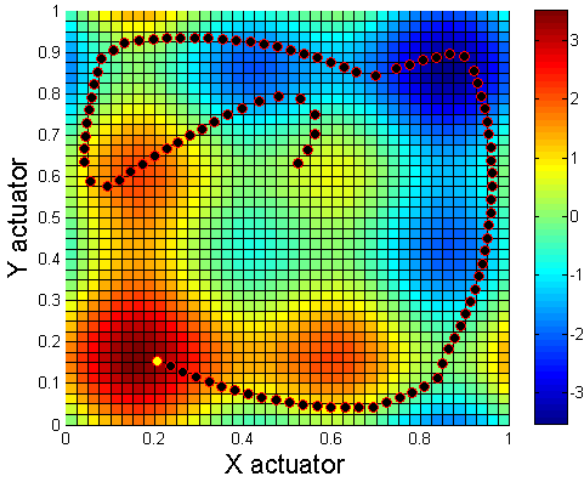



Figure 8.8: Example of navigation in a 2D search space using the NNet structure depicted in Fig. 8.7, starting at (0.53,0.63) and ending at (0.2, 0.15). Although the optimisation agent finds the global maximum, the search method is not optimal.

### A more effective approach

The decisions made by the optimisation agent should ideally be guided by a "local map" of the search space based on past trials. Indeed, since the optimisation agent has a limited range of actions, it is not necessary to build a map of the whole search space. Because data containing information on past trials are available to train a NNet, it is more convenient to proceed with a conventional NNet rather than using reinforcement learning, which requires modeling a fitness function. Re-training the NNet is a fast process when the amount of data is small and the NNet's structure is not large. This is ensured when the NNet models the objective function for only a small region of the search space, including past trials within a given radius of the optimisation agent's location. At each step, the NNet will be re-trained with new data corresponding to past trials within a given radius of the current location of the optimisation agent. Points taken within this radius are then selected, and the value of the objective function is evaluated with the model built using the NNet. The coordinates of the point that are most likely to increase the value of the objective function are then selected for the next step (i.e. inverse modeling). The NNet that is used to perform the modeling task consists of two input nodes (for the $x$ and $y$ coordinates of a past trial), one output node corresponding to $H(x,y)$ and four hidden nodes. The number of hidden neurons is arbitrary and further investigation is required to determine the optimum number.

Figure 8.9 shows two examples for the test objective function given in Eq. (8.23). In both cases the search starts from the center of the 2D space at $(0.5, 0.5)$. On the left figure, the optimisation agent is able to find its way towards the global maximum. In the right figure, however, the optimisation agent is trapped at a local maximum. This shows the necessity for a substructure to handle local maxima. The difference in behavior between the two simulations comes from the initial weight configuration of the performance navigation substructure. When the NNet is re-trained, there are multiple solutions to the NNet's weights that fit the available data. This leads to differences in the local maps of the objective function fitted by the NNet. This is similar to the decisions taken by different operators when performing an optimisation task. It is only when more data are available in the same region that the fit becomes more accurate.

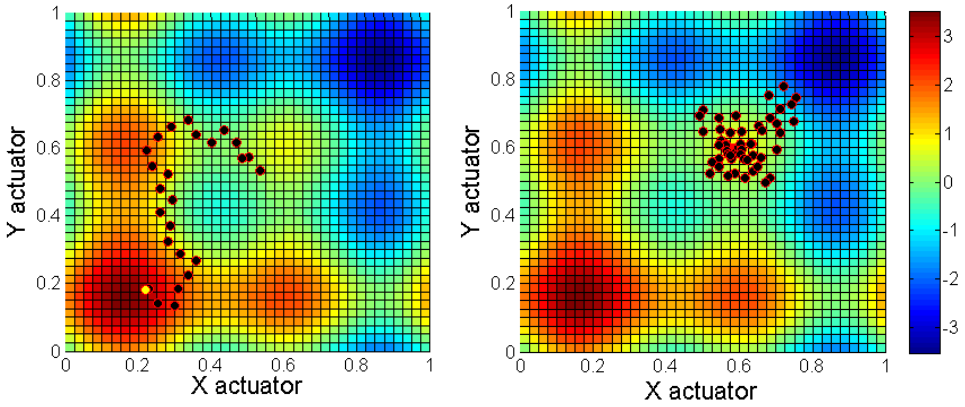



Figure 8.9: Examples of a search with the performance-based substructure. The search starts at (0.5,0.5) with $\theta_0 = \frac{3}{4}\pi$. On the left, the optimisation agent finds its way towards the maximum, whereas on the right, it is trapped at a local minimum.

### 8.3.3 Substructure three: local maxima avoidance

The optimisation agent needs to be able to move to a different region of the search space when the local density of trials is too high (for example, when there is a local maximum). The input to this substructure is a Boolean variable, which is set to 1 when the local density $D_k$ of trials (at step $k$) in the search space exceeds a given threshold $D_{thres.}$, otherwise it is set to zero. When this Boolean variable is "true" (i.e. equal to 1), the optimisation agent must decide on a new region of the search space to explore, based on past trials for the entire search space. The approach is based on what the actions of an operator are. An operator would look at the distribution of the

explored parameters in the search space and start the actuators in an unexplored (or a lower density) region. To accomplish this, the NNet must be able to identify new regions of the search space that have not yet been explored.

To detect a new region in the search space we proceed as follows. When the local density sensor is activated (i.e. local density exceeds a given threshold), a "long distance" density sensor is activated in order to detect a new region to re-initialise the search. This sensor evaluates the local density for points located between the current location of the optimisation agent and the boundary it faces (at step $k$ the direction is given by $\theta_k$). The new point must be located at a minimum distance from the current location to avoid climbing back to the same maximum and the new local density must be less than $D_{thres.}$. This process is illustrated in Fig. 8.10.

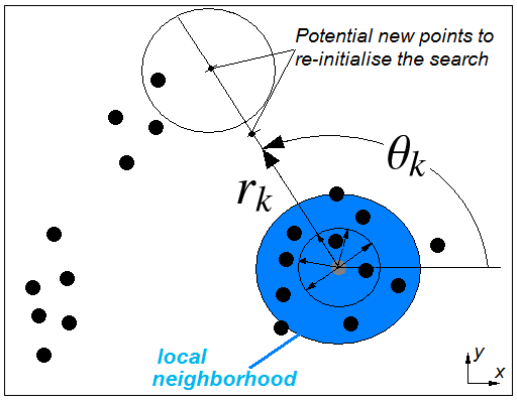



Figure 8.10: Identification of a local maximum by the optimisation agent. When the density of trials in the local neighborhood (blue region) exceeds a given threshold, the density sensor is activated to find a lower density region in the direction faced at the current step. If no satisfactory region is found, the optimisation agent rotates with no forward motion and senses regions in another direction (i.e. $r_{k+1}$=$r_k$ and $\theta_{k+1}$=$\theta_k$+$\Delta\theta$, where $\Delta\theta$ is a small angle generated randomly). This continues until a new region is found.

Figure 8.11 shows examples of searches where the three substructures of the NNet were utilised. In these examples the search started from the middle of the search space with $\theta_0 = \frac{3}{4}\pi$. On the left figure, the optimisation agent first gets trapped at the maximum located around $(x,y)$=(0.6,0.6). It then relocates to the vertical right boundary around $(0.85, 0.65)$. At the second step the boundary is detected and the optimisation agent decides to take a right turn. It then follows the boundary until it detects an increase in the value of the objective function at $(0.9, 0.25)$, leading it to the local maximum located at $(0.6, 0.15)$. When it reaches $(0.45, 0.2)$, the opti-

misation agent just misses the local maximum; however, the performance-based navigation substructure of the NNet redirects the search towards that local maximum. Upon reaching $(0.62, 0.05)$, the boundary is detected by the agent's boundary sensors and for a few steps the navigation is operated by the boundary navigation substructure. The optimisation agent then works its way to the top of the global maximum.

In the second example (right figure of Fig. 8.11), the optimisation agent slowly goes towards the top right corner at $(1, 1)$. The density of points exceeds the density threshold $D_{thres.}$ and the optimisation agent decides to jump to $(0.57, 0.11)$. Navigation through the search space is then directed by the boundary navigation substructure until it reaches the region containing the global maximum. The performance navigation substructure takes over and directs the optimisation agent towards the global maximum.

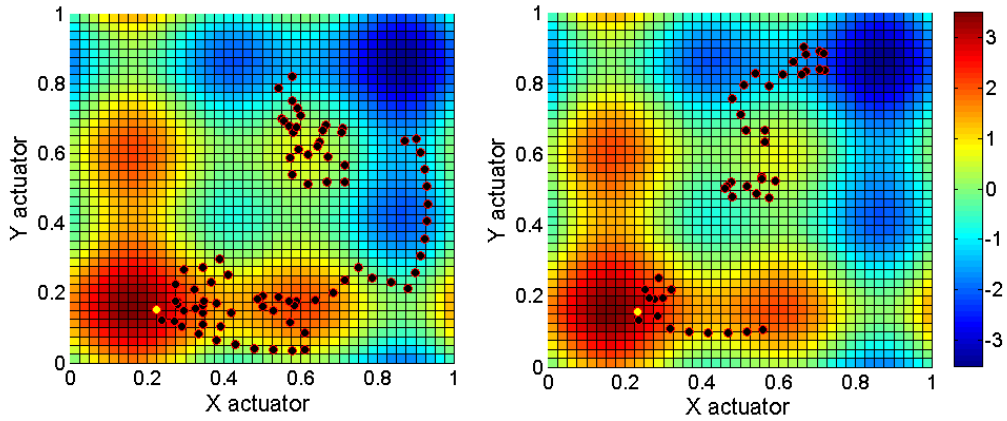



Figure 8.11: Examples of searches that utilise the local maximum avoidance substructure. The search starts from (0.6,0.6) and (0.5,0.5) on the left and right figures, respectively. In both cases the optimisation agent is able to find its way towards the global maximum, using: the boundary navigation substructure to remain within boundaries, the performance-based navigation substructure to detect maxima, and the local maxima avoidance substructure to re-initialise the search, when the local density of past trials exceeds a threshold $D_{thres.}$. Figure adapted from [120].

### 8.3.4 Global structure of the system

The final structure of the whole system is shown in Fig. 8.12. From left to right, the eight inputs correspond to the local density $D_k$, the five boundary sensor inputs (as depicted in Fig. 8.3) and two vectors $\vec{x}_k$ and $\vec{y}_k$ giving the coordinates of past trials in the local neighbourhood. The first input (node 1) is connected to a Boolean node $B_1$ (far left in the second middle layer)

that is also connected to the bias (on the far left of the figure). This node will therefore only fire if $D_k$ exceeds $D_{thres.}$. When this occurs, a long distance density sensor is activated in order to find a new region to re-initialise the search (see Sec. 8.3.3), until a new region of the search space is found. This is represented by a red circle with clockwise arrows.

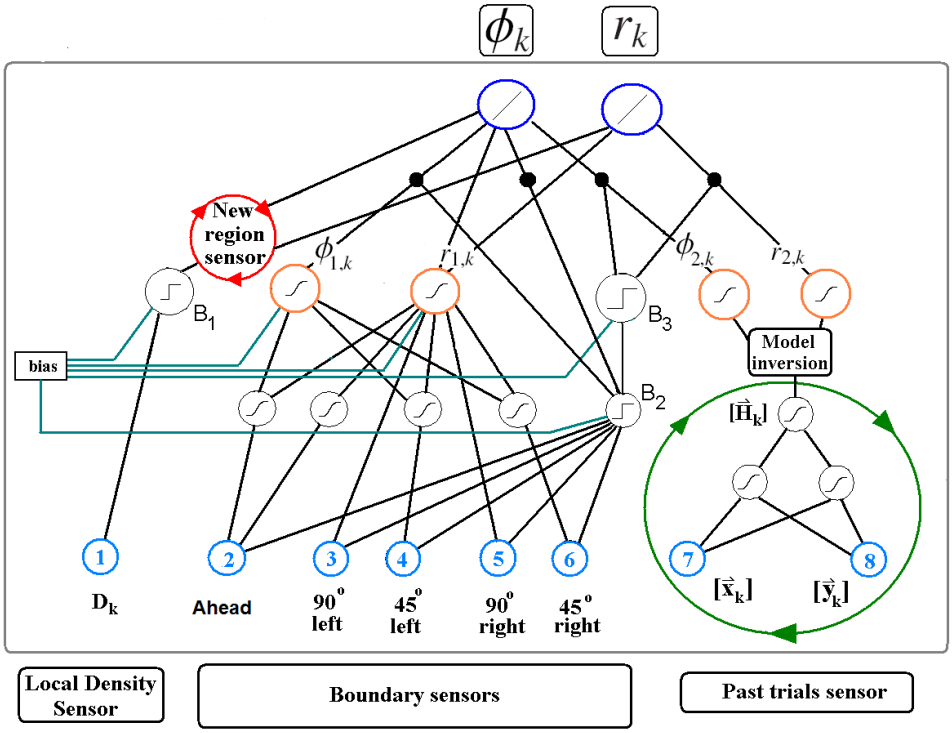



Figure 8.12: Structure of the entire optimisation NNet. The three groups of inputs correspond to the three substructures. The first input node (node 1) is used by the local maxima avoidance substructure. The next five inputs (nodes 2 to 6) correspond to the five boundary sensors and are used by the boundary navigation substructure. The last two inputs (nodes 7 and 8) receive the coordinate vectors $\vec{x_k}$ and $\vec{y_k}$ of past trials in the local neighbourhood of the optimisation agent and are used by the performance-based navigation substructure.

Inputs 2 to 6 are used to feed the knowledge-based boundary navigation substructure described in Sec. 8.3.1. The boundary navigation substructure has two outputs $\phi_{1,k}$ and $r_{1,k}$ (the left two orange nodes in Fig. 8.12), corresponding to the "move forward" and "turn" actions of the boundary navigation substructure. The bias that connects to the two output nodes of this structure is used to increase or decrease the sensitivity of the response. Input nodes 2 to 6 are also connected to the Boolean node $B_2$ (far right in the first middle layer). When the bias connection to node $B_2$ is zero, the neuron

fires if any of the sensors have detected a boundary. The sensitivity of this neuron can then be decreased by the addition of a value to the bias connection. The neuron will only fire when the sum of the inputs exceeds the value of the bias. The output of node $B_2$ is used as a Boolean weight for the substructure connection from $\phi_{1,k}$ and $r_{1,k}$ to the final outputs $\phi_k$ and $r_k$ of the NNet. When the sum of inputs 2 to 6 exceeds a set threshold, the NNet's final outputs are based on the boundary navigation substructure. This Boolean node is also connected to a second "negation" Boolean node $B_3$. The output of node $B_3$ will serve as a weight that activates or deactivates the output of the performance navigation substructure. When the output of node $B_2$ is 0 (no boundaries are detected), the output of node $B_3$ is 1, which activates the performance-based navigation substructure.

Input nodes 7 and 8 (the coordinates of the past trials) feed the performance navigation substructure represented by the large green circle in Fig. 8.12. The clockwise arrows indicate that this substructure is re-trained when new data are available to the system to update the model of the objective function. The outputs used for the training exploit the vector $\vec{H}_k$ containing the values of the objective function for the past trials with coordinate vectors $\vec{x}_k$ and $\vec{y}_k$. Inverse modelling is used to evaluate the value of $H$ for a series of points (angles $\phi$ and distances $r$ from the current location) within reach of the optimisation agent (i.e. within a radius $R$ from the location of the optimisation agent in the search space), and select the coordinates of the search space that are most likely to improve the value of $H$.

## 8.4 Optimisations at the Australian Light Source

### 8.4.1 Experimental setup and calibration

A schematic of the Linac at the Australian Synchrotron is shown in Fig. 8.13 (see Sec. 3.2.2 for more details). In what follows we consider the optimisation of the energy spread and transmission as a function of the phase of the primary and final bunchers (shown in red in Fig. 8.13).

The energy spread measurement is calibrated in the following way. The bending section from the Linac to the LTB contains two dipoles (see Fig. 8.13). Because a direct measurement of the beam energy is not available, we emulate a change in the beam energy by decreasing the current of the first dipole and measure the beam displacement on the LTB1 screen. The current of the dipole is decreased in steps of 0.5% down to 93% of its nominal value. Decreasing the current further would cause loss of the beam from the screen. For each point, an image of the beam is taken and a Gaus-

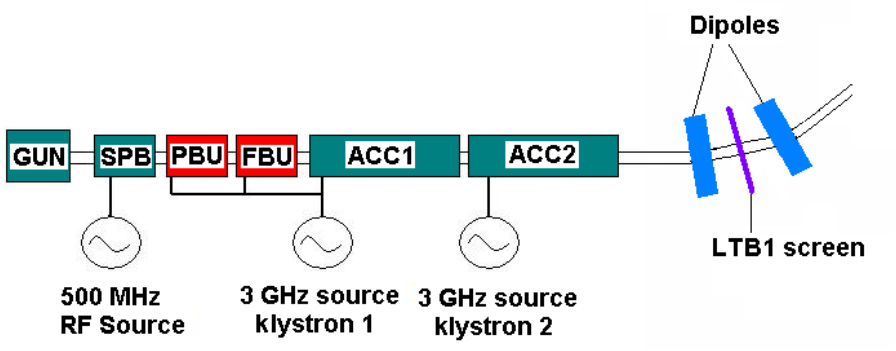



Figure 8.13: Simplified schematic of the Australian Synchrotron Linac (see also Fig. 3.1). The parameters for our 2D search space are the phases of the PBU and FBU. The first dipole in the LTB is used to calibrate the energy spread measured on the LTB1 screen.

sian profile fitted to the horizontal projection as shown in Fig. 8.14. The mean value of the fit corresponds to the position of the center of the beam (in pixels), while the width corresponds to the energy spread (in pixels); the latter is converted to a percentage of the beam energy.

To establish the calibration, the mean of the Gaussian is plotted against the relative change (in [%]) of the magnet's current as shown in Fig. 8.15. Since a decrease of 1% in the current of the dipole results in a decrease of 1% in the beam energy, the conversion factor corresponds to the gradient of the calibration curve. The calibration gives:

$$F_{cal.} = 18 \pm 0.5 \frac{Pixels}{Energy\ Change\ [\%]}. \tag{8.24}$$

The energy spread is then obtained by converting the width of the Gaussian from pixels to a [%] using the conversion factor (8.24).

The transmission is measured using the wall current monitor in concert with a fast current transformer, which measures the beam charge at the end of the Linac (as described in Sec. 3.2.5). The calibrated transmission is directly available through a Matlab PV.

### 8.4.2 Experimental results of the energy spread and transmission optimisation

In this section we present the results obtained when the beam energy spread $\delta E$ and transmission $T$ were first optimised independently and then optimised simultaneously.

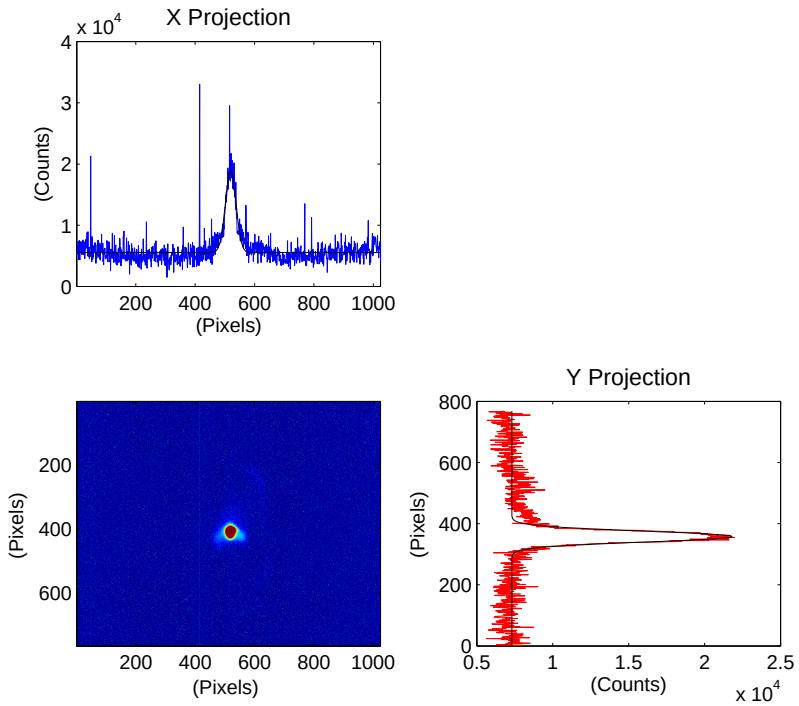



Figure 8.14: Image of the beam taken at the LTB1 screen. Gaussian profiles are fitted to the beam projections. The energy spread is given by the width of the horizontal projection, while the beam energy is determined from the mean of the horizontal projection.

#### Independent energy spread optimisation

Figure 8.16 shows the result obtained for the minimisation of the energy spread. The boundaries were set to ±10 ° from the nominal phase settings, $\varphi_{PBU} = 107°$ and $\varphi_{FBU} = 128°$. This latter point is denoted by a large cyan diamond in Fig. 8.16 and corresponds to an energy spread of 1.030% ± 0.035%.

The search started from the arbitrary point $\varphi_{PBU} = 113°$ and $\varphi_{FBU} = 134°$, represented by a large brown square in Fig. 8.16. The different colors in this figure help identify the way the search was operated. For example, the evolution in the search space can easily be followed by looking at the left plot in Fig. 8.18. At some point the density of trials (represented by the group of black dots) exceeded the set threshold and the search was relocated to $\varphi_{PBU} = 99°$ and $\varphi_{FBU} = 133°$ (in the region represented by the group of blue dots). When the density of trials exceeds a pre-determined threshold, the search was relocated to $\varphi_{PBU} = 114°$ and $\varphi_{FBU} = 133°$, represented by the group of yellow dots. Because the direction of decreasing energy spread was a region that had already been explored (represented by the black dots), an-

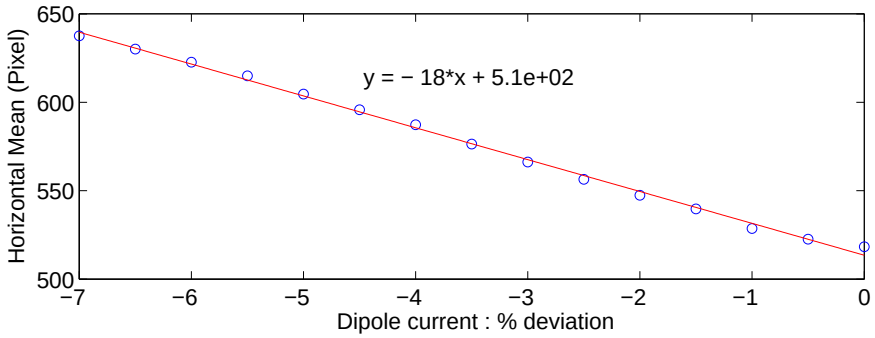



Figure 8.15: Calibration curve for the measurement of the energy spread. The current of the first dipole is scanned from its nominal value down to -7% from this value. The position of the center of the beam on the LTB1 screen is calculated from a Gaussian fit to the horizontal projection of the beam.

other jump was made to $\varphi_{PBU} = 110°$ and $\varphi_{FBU} = 121°$. After a few trials the optimisation agent identifies the direction of decreasing energy spread and pursues the search in that direction, eventually achieving an energy spread of 0.912% (green dots). Each point is the average of five measurements (i.e. five bunches), which results in a reduction in the error from 0.080% to ±0.035%. The experiment took about 30 minutes. This is mainly due to a waiting time of 3 sec/° imposed to stabilise the phases of the bunching sections after each adjustment, as well as the fact that five measurements are made for each point.

#### Independent transmission optimisation

Figure 8.17 shows the results obtained for the independent optimisation of the transmission. The search space was initially set to the same boundaries as for the energy spread (i.e. ±10° from the nominal phase settings). Since the transmission was not very sensitive in this region, the search was extended to ±20° from the nominal settings (i.e. $\varphi_{PBU} = 107°$ and $\varphi_{FBU} = 128°$). Each point in Fig. 8.17 is the average of 30 measurements. This results in a decrease of the rms error in the transmission measurement from 2.8% to 0.5%.

The search started in the region represented by the group of black dots, and proceeded until the density became too high, whence it was relocated to $\varphi_{PBU} = 90°$ and $\varphi_{FBU} = 135°$ (see the right plot in Fig. 8.18). The search then continued in the region represented by the group of blue dots, until the density of points became too high. The search was once again relocated,

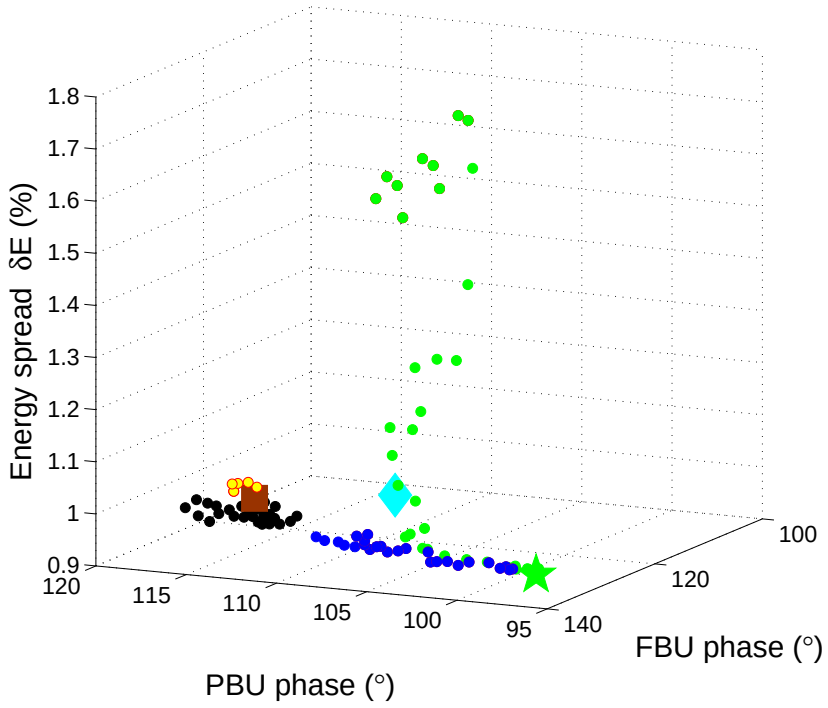



Figure 8.16: Independent optimisation of the energy spread $\delta E$. The initial energy spread of the machine (before optimisation) is represented by the cyan diamond. The large brown square indicates where the search started and the large green star indicates the energy spread that was achieved. The different colors correspond to the regions searched by the optimisation agent.

this time to the region represented by the group of green dots around $\varphi_{PBU} = 94°$ and $\varphi_{FBU} = 114°$. A few more points were taken until the maximum number of iterations was reached. The whole run took just under one hour, which is dictated by the waiting time imposed on each phase adjustment and the number of measurements taken for each point.

Figure 8.18 shows the 2D representation of the energy spread and transmission. Both plots have the same limits (±20° from the initial settings), although the search space for the energy spread (±10°) was half that of the transmission (±20°) on each axis. From this figure it is seen that both the energy spread and transmission are improved for lower values of the PBU phase. According to Fig. 8.16 increasing the FBU phase clearly reduces the energy spread while the transmission is not significantly affected (see Fig. 8.17). The simultaneous optimisation of the energy spread and the transmission should therefore require lower values of the PBU phase.

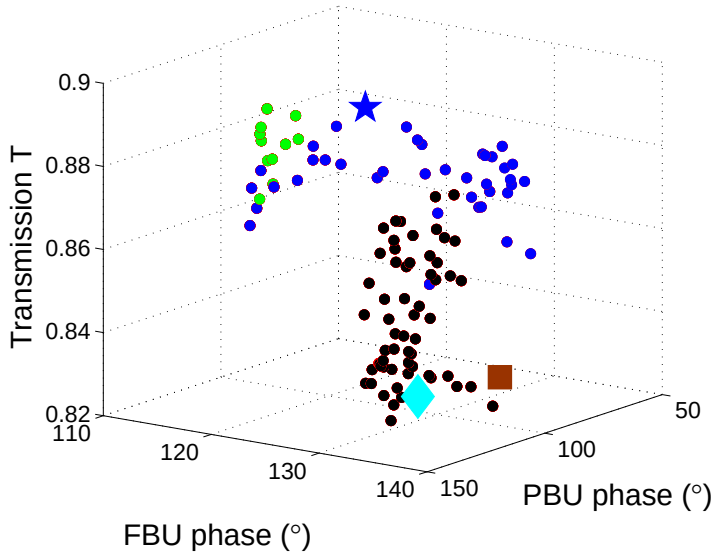



Figure 8.17: Independent optimisation of the transmission $T$. The large brown square shows the starting point, while the large blue star indicates the transmission that was achieved. The initial transmission of the machine (before optimisation) is represented by the cyan diamond.

The transmission was measured for the phases that optimised the energy spread and vice versa. Results are listed in Table 8.5. In the first case the energy spread optimisation also leads to an improvement in the transmission of about 3%. On the other hand, the energy spread was not affected by the changes in the phase that optimised the transmission.

Table 8.5: Results of the independent optimisation of the energy spread and transmission. A 0.12% decrease in the energy spread is achieved together with a 3% increase in transmission for the energy spread optimisation. A 5% transmission improvement was achieved for the independent transmission optimisation with no significant change in the energy spread.

| | Nominal Settings | $\delta E$ optim. | $T$ optim. |
|---|---|---|---|
| PBU Phase [°] | 107 | 98.5 | 89.9 |
| FBU Phase [°] | 128 | 130.2 | 121.2 |
| $\delta E$ [%] | 1.030±0.035 | 0.910 ±0.035 | 1.000 ±0.035 |
| $T$ [%] | 83.5 ±0.6 | 86.6±0.5 | 88.8 ±0.5 |

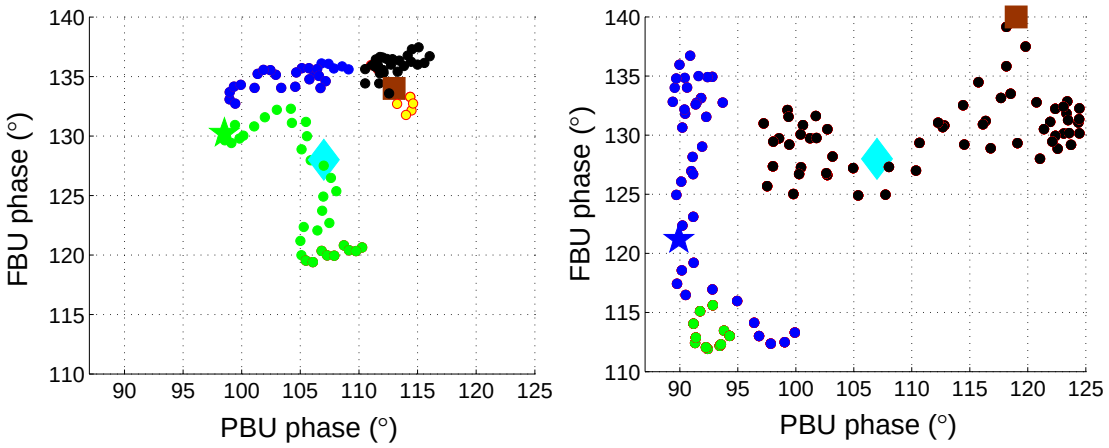



Figure 8.18: 2D representation of the optimisation of the energy spread (left) and transmission (right). The transmission search space was double the size of the energy spread.

#### Simultaneous energy spread and transmission optimisation

In order to optimise both the energy spread and transmission we must establish an objective function. Assuming that we give equal importance to the energy spread and the transmission, the objective function will contain two terms with identical weights of 1/2. Since the transmission $T \in [0,1]$, an objective function $H(T,\delta E) \in [0,1]$ will contain a term directly proportional to $T$. This is represented by the first term on the right hand side of Eq. (8.25). The energy spread is $\delta E \in [\delta E_{min}, \delta E_{max}]$ and in order to cover this range it is necessary to ensure that $\delta E_{min}$ and $\delta E_{max}$ are sufficiently low and high, respectively. Moreover, it is important that relative changes in the value of the energy spread and transmission contribute nearly equally to the value of the objective function. Considering that the smallest and largest energy spread observed were around 0.9% and 3.0%, respectively, we set limits of $\delta E_{min} = 0.5\%$ and $\delta E_{max} = 3.8\%$. Upon re-normalising the contribution to the energy spread, so that it ranges from 0 to 1, the objective function can be written as:

$$H(T,\delta E) = \frac{1}{2}T + \frac{1}{2}\left(1 - \frac{\delta E - \delta E_{min}}{\delta E_{max} - \delta E_{min}}\right) \tag{8.25}$$

Simultaneous optimisation was performed for a range of 50° in both the PBU and FBU with $\varphi_{PBU} \in [77°, 127°]$ and $\varphi_{FBU} \in [98°, 148°]$. In order to reduce the error on the transmission measurement, each point is the average of 30 measurements (for both the energy spread and the transmission). A total of 150 points were taken at 1 Hz with a run time of about 90 mins. This

duration is due to the number of measurements taken per point and the necessary waiting time imposed to stabilise the phases of the sections after each adjustment.

Results are shown in Fig. 8.19 and listed in Table 8.6. It is important to note that this optimisation was performed after a shutdown of the machine. Although the energy spread of the beam did not seem to be affected by the shutdown, we recorded a higher transmission. Indeed, while the nominal settings of the PBU and FBU gave 83.5% transmission before shutdown, the same settings resulted in 90.0% transmission after shutdown. The beam transmission is sensitive to the field imposed by focusing magnets. Since all magnets remained powered during the shutdown period the difference is most likely due to temperature variations in the cooling system of the focusing magnets between the two measurements.

The search can be divided into six parts that are represented by different colors in Fig. 8.19. It was started with the arbitrary phases $\varphi_{PBU} = 117°, \varphi_{FBU} = 138°$; this is denoted by a large brown square in Fig. 8.19. The progression through the search space can be followed from the projection in Fig. 8.19 (a). The search moved from the region represented by the black dots to the region represented by the green dots, corresponding to smaller values of the PBU phase with no major change in the FBU phase. The optimised settings were found at the $75^{th}$ trial in the region containing the green dots, with the optimal point given by the large green star. Because the density of trials exceeded the density threshold in this region, and the maximum number of steps had not yet been reached, the search was redirected to $\varphi_{PBU} = 107.4°, \varphi_{FBU} = 102.7°$, represented by the group of blue dots. Figure 8.19 (d) shows that this region corresponds to a local minimum of the energy spread, or a local maximum in the objective function (see Fig. 8.19 (b)), whereas the transmission does not vary significantly. Once the local maximum in the objective function is identified, the search is relocated to $\varphi_{PBU} = 83°, \varphi_{FBU} = 113.5°$, corresponding to the region containing the orange dots. Only a few points were taken before the density of trials exceeded the threshold, resulting in the search being re-directed to $\varphi_{PBU} = 103°, \varphi_{FBU} = 143.5°$, corresponding to the region containing the red dots. Again, after a few trials the density exceeded the threshold and the search was relocated to $\varphi_{PBU} = 82°, \varphi_{FBU} = 114°$ (region containing the pink dots). Finally, the optimisation relocated to $\phi_{PBU} = 79°, \varphi_{FBU} = 105°$, corresponding to the region containing the yellow dots; the search was stopped once the limit of 150 trials was reached. The optimisation resulted in an increase in the transmission from 90% to 97% and a decrease in the energy spread from 1.04% to 0.91%, demonstrating the efficacy of the technique.

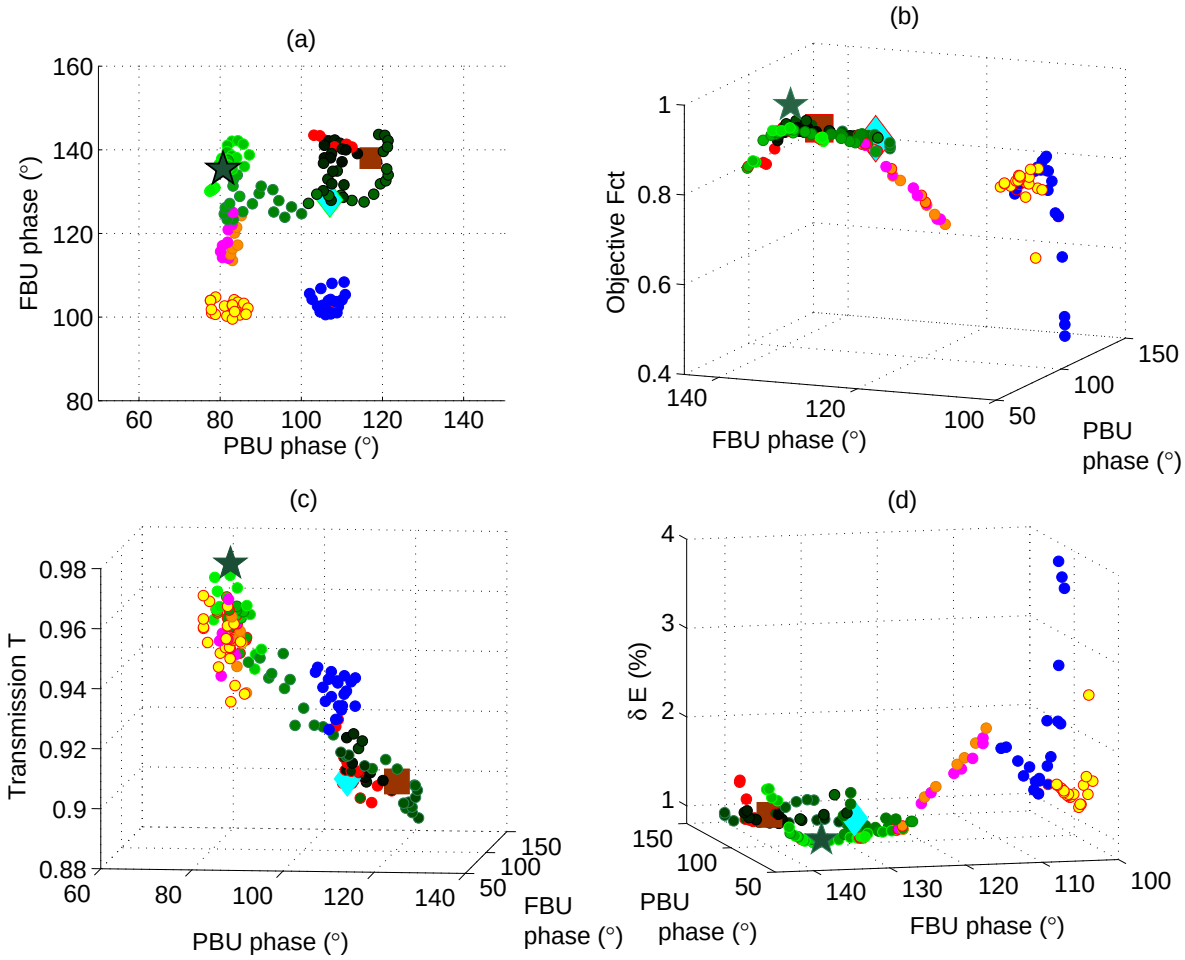



Figure 8.19: Simultaneous optimisation of the energy spread and transmission. (a) Points in the projected PBU-FBU search space. Different colors identify regions of the search space, which are arrived at by re-direction when the density of points exceeds a given threshold. (b) The objective function defined in Eq. (8.25) used by the NNet to perform the simultaneous optimisation. (c) Corresponding values of the transmission during the optimisation. (d) Corresponding values of the energy spread during the optimisation. Figure adapted from [120].

## 8.5 Optimisations at FERMI

### 8.5.1 Energy spread minimisation at the laser heater spectrometer

The compression of the beam depends critically on the energy spread $\delta E$ in the region of the laser heater; to achieve the compression factor required by the design of the FERMI machine, it is highly desirable to be able to set the energy spread to a chosen value. In the experiments at FERMI, we chose to search for a minimum energy spread at the laser heater (LH) spectrometer, while maintaining the energy $E$ of the beam within given limits, i.e. $E \in [E_{min}, E_{max}]$. Although we could have chosen to set the energy spread to a given value, we decided to investigate the lowest value that could be achieved. The photoinjector consists of the accelerating structure L0, which is composed of two identical accelerating structures L0A and L0B powered by klystron K02 as shown in Fig. 8.20. The phases of L0A and L0B can be controlled by the phase of K02 and the phase shifter of L0B. The phase of the K02 klystron and the phase of the shifter were chosen as adjustable parameters to minimise the energy spread. A Matlab script implemented for the commissioning of the FERMI machine was available for the measurement of the energy and energy spread at different locations, including the spectrometer line of the laser heater and the spectrometer line of the first bunch compressor. The energy is calculated using the dispersion in the bend, while the energy spread is calculated from the beam size on the screen located at the first spectrometer line (see Fig. 8.20).

To conduct our optimisation studies an objective function $H(E, \delta E)$ must be defined. We give equal importance to changes in the energy and the energy spread of the beam. Consequently, the objective function consisted of two terms, one for the energy and a second for the energy spread, with identical weights of 1/2. Both terms measure the deviation from the initial

Table 8.6: Results of the simultaneous optimisation of energy spread and transmission show a reduction in the energy spread of 0.13% and improvement of 7% in the transmission from the nominal settings.

| | Nominal Settings | Simultaneous optim. |
|---|---|---|
| PBU Phase [°] | 107 | 80.7 |
| FBU Phase [°] | 128 | 135.4 |
| $\delta E$ [%] | 1.04 ± 0.009 | 0.91 ± 0.009 |
| T [%] | 90.0 ± 0.6 | 97.0 ± 0.6 |

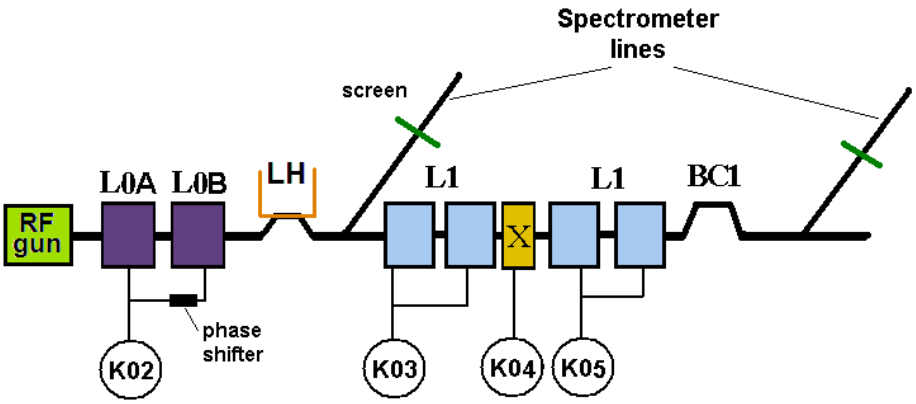



Figure 8.20: FERMI RF layout up to BC1. The Linac 0 is powered by klystron K02 equipped with a phase shifter. Klystrons K03 and K05 power the first two and last two sections of Linac 1, while K04 powers the X-band cavity. Two spectrometer lines at the laser heater and the bunch compressor allow for measurements of the beam energy and the energy spread.

settings, expressed as a percentage. Deviations in the energy and energy spread are measured and normalised against the maximum values of the deviations. The maximum deviation in the energy is specified by the limits $E_{min}$ and $E_{max}$, while the maximum variation of the energy spread $\delta E_{max}$ was set to 1% (based on experimental observations). The search is stopped when the energy falls outside the set limits, at which the value of the objective function is set to 2. This value was arbitrarily chosen to ensure wide coverage of the search space. The objective function is given by:

$$H(E,\delta E) = \begin{cases} \frac{1}{2}\frac{1}{|E_b - E_0|}|E - E_0| + \frac{1}{2}\frac{\delta E}{\delta E_{max}} & \text{if } \; E \in [E_{min}, E_{max}] \\ 2 & \text{otherwise,} \end{cases} \tag{8.26}$$

where $E_0$ refers to the nominal energy of the beam, $E$ is the measured energy, $\delta E$ is the energy spread, $\delta E_{max}$ is the maximum energy spread we can tolerate, which is determined by the actuators, and:

$$E_b = \begin{cases} E_{min} & \text{if } \; E < E_0 \\ E_{max} & \text{if } \; E > E_0. \end{cases} \tag{8.27}$$

In our first experiment the nominal energy was $E_0 = 85$ MeV, with $\phi_{K02} = 320^\circ$ and $\phi_{K02,shifter} = 218^\circ$, corresponding to the values of the phase and phase shifter of klystron K02, respectively. The initial absolute energy spread was measured to be 0.4 MeV and the run was operated over a range of $\pm 4^\circ$

from the nominal settings of both phases. Figure 8.21 shows the results of the optimisation. The NNet successfully processes the information, allowing the optimisation agent to navigate through the search space and optimise the objective function. However, when using the settings found by the optimisation agent, the beam exhibited a large spread on the camera. Figure 8.22 illustrates this, where the left image shows the beam before the optimisation and the right image shows the beam after optimisation.

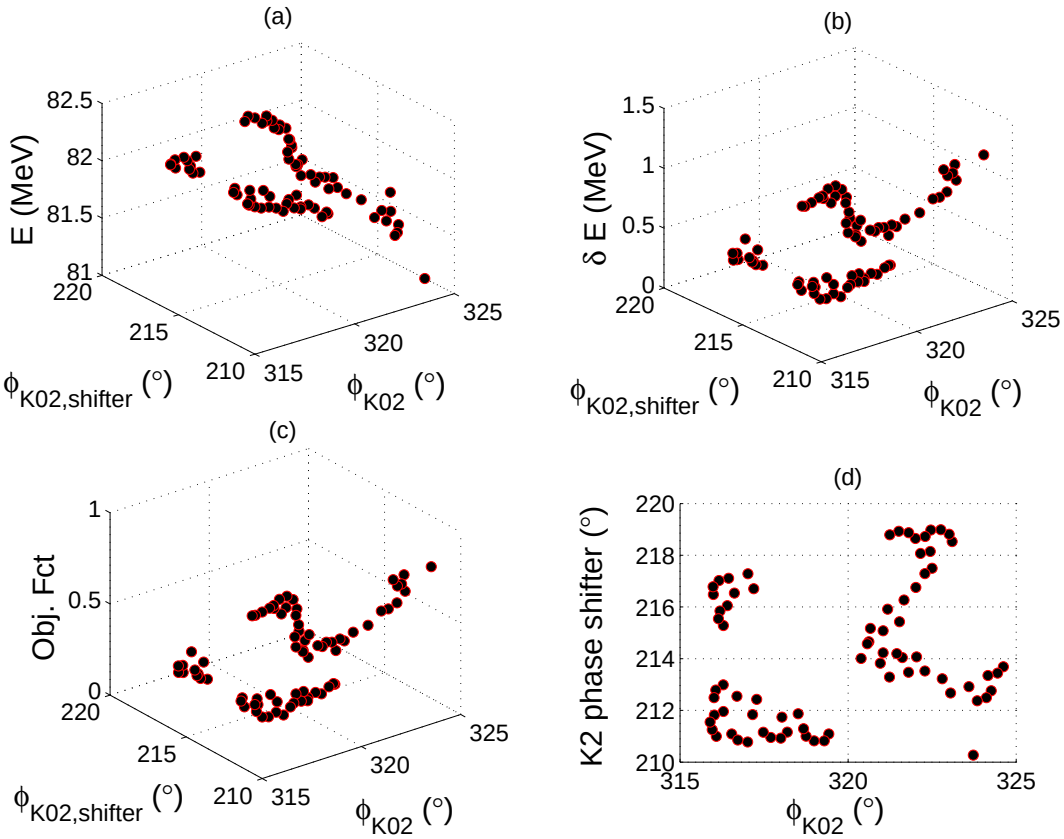



Figure 8.21: Minimisation of the energy spread at the laser heater of the FERMI machine. (a) Energy of the beam. (b) Corresponding energy spread of the beam. (c) The objective function defined in Eq. (8.26), which is used by the NNet to perform the optimisation. (d) Points in the K02-K02 shifter search space. The search started at $(\phi_{K02}, \phi_{K02,shifter}) = (323°, 213°)$ and moved to $(322°, 219°)$, where the density of points caused the search to relocate to $(317°, 217°)$ and then to $(319°, 211°)$.

The spread in the beam image was identified to be directly related to the measurement script. Because the beam has neither a Gaussian or a parabolic profile, the energy spread cannot be obtained from a direct fit based on these functions. To establish the beam size on the screen, an algorithm is used to select the region of interest from the image that contains

the beam spot. The image is first resized and a low pass filter is used to eliminate noise and smooth the image. To identify the region of interest (ROI) that contains the beam, the image is divided into equal sections and the mean intensity is calculated for each section. The intensity scale is divided into equal intervals, and the number of sections falling into each interval is plotted against the intensity intervals. The algorithm defines the transition between the beam and the background as the point where the intensity derivative is 40% of the maximum derivative of the plot. The corresponding transition in terms of the intensity threshold is then calculated. The algorithm[9] identifies the region of interest (from the center of the beam) corresponding to those sections with intensity values above the threshold.

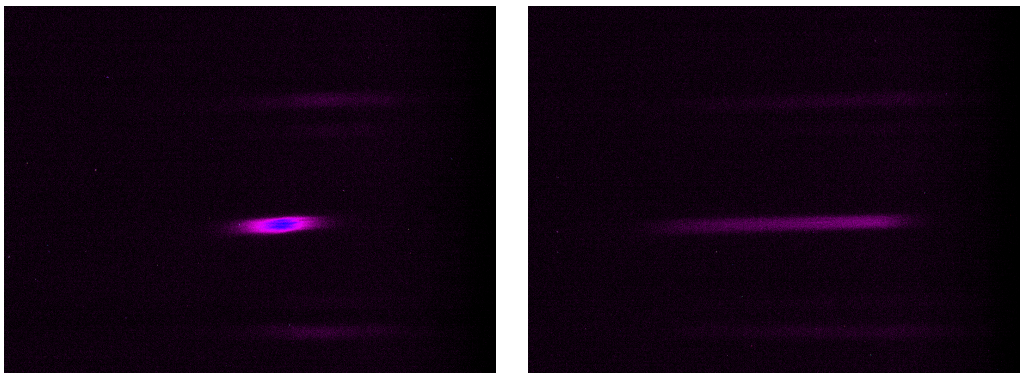


Figure 8.22: Images of the FERMI beam at the laser heater spectrometer. Left: image of the beam taken with the initial settings. Right: image of the beam for the settings found by the NNet. The beam appears smeared out because of invalid energy spread measurements.

The experiment was repeated on a different day in order to compare the results when the ROI was ON and OFF. The beam energy for these settings was 92 MeV, with initial phases $\phi_{K02} = 350^\circ$ and $\phi_{K02,shifter} = 265^\circ$. The phase range was $\pm 5^\circ$ from the nominal settings of both phases, but only a small portion of that search space was required. Figures 8.23 and 8.24 show the results with the ROI turned ON and OFF.

When the ROI option is turned OFF, the measurement of the energy spread is carried out by subtracting the background from the beam image. In each of the two runs the optimisation agent takes a slightly different path in order to minimise the objective function. In both cases the values of $\phi_{K02}$ and $\phi_{K02,shifter}$ were decreased from their initial settings. Checking the image of the beam while the optimisation tool was running revealed that the

[9] Although this algorithm has been employed at FERMI, it is not appropriate for delineating the region of interest containing the beam. A more sophisticated method would utilise differential edge detection or morphological edge detection.

problem persisted when the whole image was used for measurement. Turning the ROI option OFF did not resolve the problem and it is expected that the algorithm selecting the ROI must be revised. It should be emphasised that this algorithm was not pivotal to the work reported in the thesis. However, since the same script is used to control the energy spread at the bunch compressor it is imperative to review the algorithm that measures the energy spread.

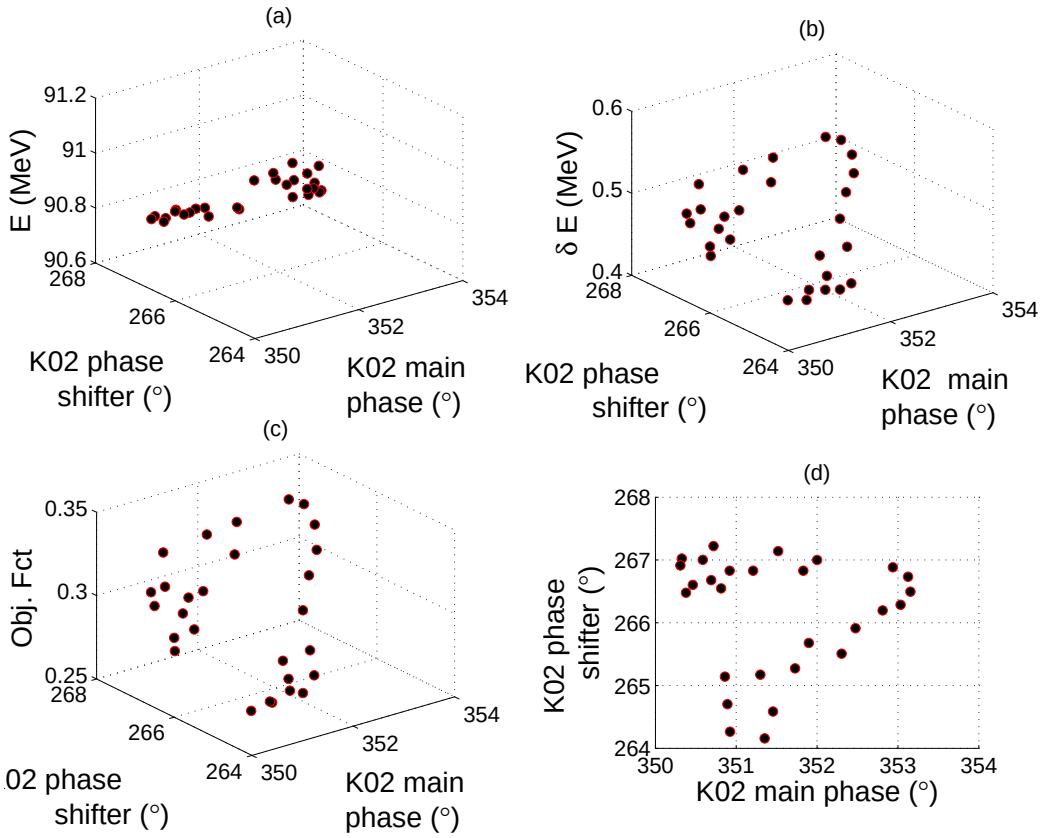



Figure 8.23: Minimisation of the energy spread with the automatic ROI selection turned ON. (a) Energy of the beam. (b) Corresponding energy spread of the beam. (c) The objective function defined in Eq. (8.26), which is used by the NNet to perform the optimisation. (d) Points in the K02-K02 shifter search space. Settings that minimise the energy spread tend to decrease both the phase and the phase of the shifter of klystron K02.

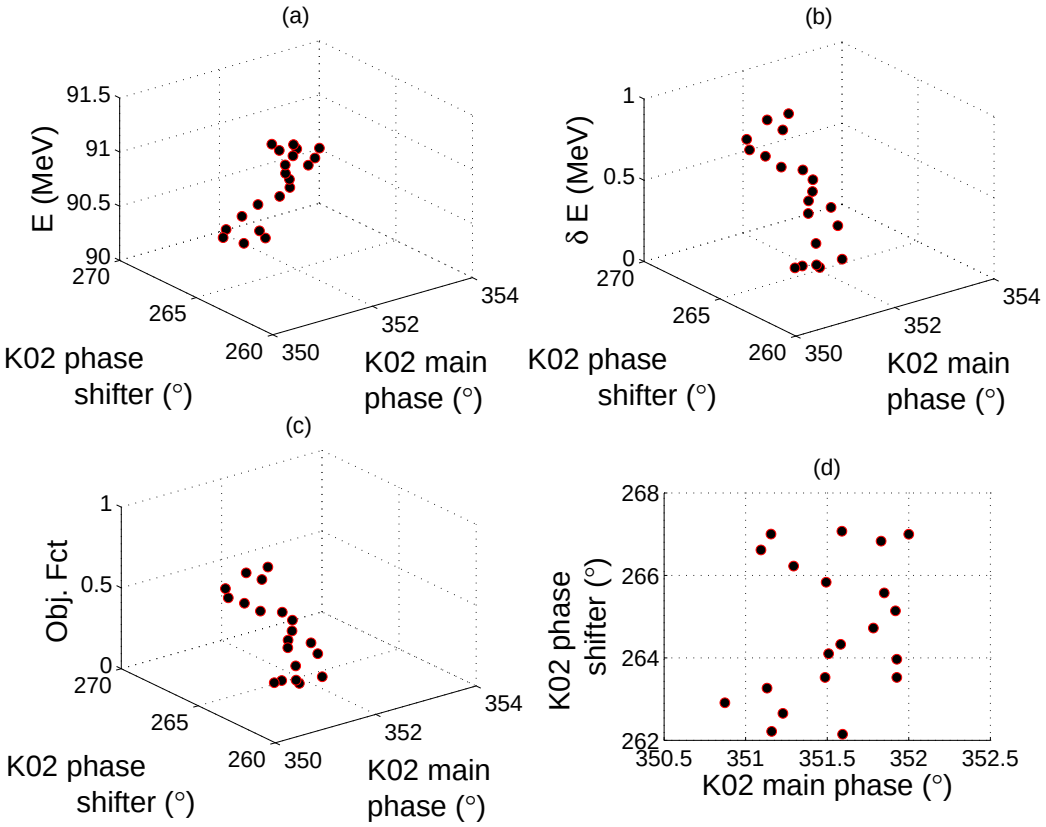



Figure 8.24: Minimisation of the energy spread with the automatic ROI option turned OFF. (a) Energy of the beam. (b) Corresponding energy spread of the beam. (c) The objective function defined in Eq. (8.26), which is used by the NNet to perform the optimisation. (d) Points in the K02-K02 shifter search space. These results show the same trends as Fig. 8.23, suggesting that the region of the image containing the beam is not selected regardless of whether the ROI is turned ON or OFF.

### 8.5.2 Beam transverse size sensitivity

The optimisation tool was also used as a diagnostic to identify potential correlations between jitter in the energy of the beam and the settings of the focusing magnets. During commissioning an abnormally high energy jitter was observed in the transverse size of the beam, as measured on a screen located immediately after the L0 section. Determination of the transverse size of the beam is needed at this location, since it used to measure the emittance (by employing quadrupole scans). Jitter at this location is important because the upstream beam trajectory is corrected to compensate for a misalignment of the accelerating cavities. The optimisation agent was used to determine whether it is possible to attenuate the jitter by adjusting the current in the gun solenoid and the field of one of the vertical correc-

tors located before the L0 structure. To investigate correlations between the magnets and the energy jitter, we need to define an objective function for the measured energy jitter; the optimisation agent should then aim to minimise this objective function. Since the energy deviation can be expressed as a percentage of the beam energy, the objective function is taken to be the ratio of the average deviation of the transverse beam size and its mean transverse size:

$$H(\delta\sigma_X) = \frac{\delta\sigma_X}{\overline{\sigma_X}}, \tag{8.28}$$

where $\overline{\sigma_X}$ is the mean transverse size of the beam taken from a sample of 25 bunches and $\delta\sigma_X$ is the average noise for the same sample. The optimisation agent was run with changes of ±5% in the actuators from the initial settings of the magnets. The results are shown in Fig. 8.25, where there is no obvious trend, which could be exploited to attenuate the jitter; even with averaging the noise remains significant.

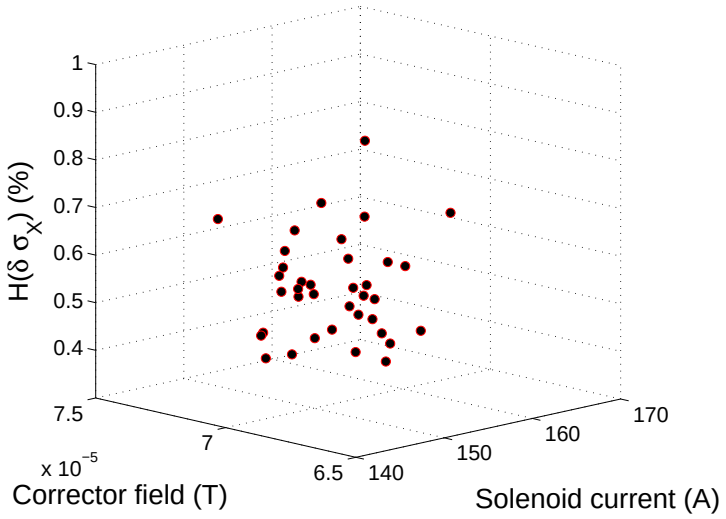



Figure 8.25: Relative deviation of the transverse beam size, measured at the end of the L0 accelerating structure, as a function of the gun solenoid current and the field of a vertical corrector, located before the L0 structure. The optimisation agent searched for a minimum in the relative deviation with a range of ±5% from the initial settings of the magnets. The plot shows a scatter of points without any particular trend.

### 8.5.3 Beam energy control test

The optimisation agent can be used as a feedback controller for the correction of a static perturbation. In particular, we consider a perturbation in the form of a step function. Such a perturbation can originate from a jitter in the machine parameters (such as the phase and voltage of the klystrons), or from a desired change in the beam requirements (for example, the beam energy).

In what follows, we consider the stabilisation of the beam energy at the bunch compressor when a 10° jump is imposed on the phase of K02. The phases of the L1 section are used as correcting actuators. This section is powered by two klystrons K03 and K05, feeding the first two and the last two cavities, respectively (see Fig. 8.20). The phases of these klystrons will serve as correctors. We note that each of the klystrons provides about 100 MeV to the beam, which reaches about 320 MeV at the bunch compressor. To use the optimisation agent as a controller, we need to define an objective function. As with the controller, the optimisation agent should aim to minimise the difference between the desired value and the measured value of the energy. This difference can be used as the objective function:

$$H(E) = E - E_0, \tag{8.29}$$

where $E_0$ is the desired energy and $E$ is the measured energy. The goal set for this experiment was to obtain a residual deviation lower than 0.05 MeV. The nominal phases of K03 and K05 were 281° and 325°, respectively. Figure 8.26 shows the results for two runs with different sized search spaces. In Fig. 8.26 (a) the optimisation agent was allowed to search in a region of ±8° from the nominal settings of K03 and K05. In Fig. 8.26 (b) the optimisation agent was only allowed to search within a region of ±5°. One can see from these two plots that the initial overshoot has an amplitude that increases with the size of the search space. This is due to the fact that the optimisation agent takes steps of sizes expressed as a percentage of the dimension of the search space. In Fig. 8.26 (a), because the search space is larger, the steps taken are larger as well. The optimisation agent must sample a few points before identifying that a particular direction taken in the search space is leading to the overshoot. The larger the step size, the larger the resulting overshoot caused by the optimisation agent as it navigates the search space. This behavior can easily be modified so that the optimisation agent takes steps that are independent of the dimensions of the search space. In both cases the energy of the beam is successfully brought back to its initial setting.

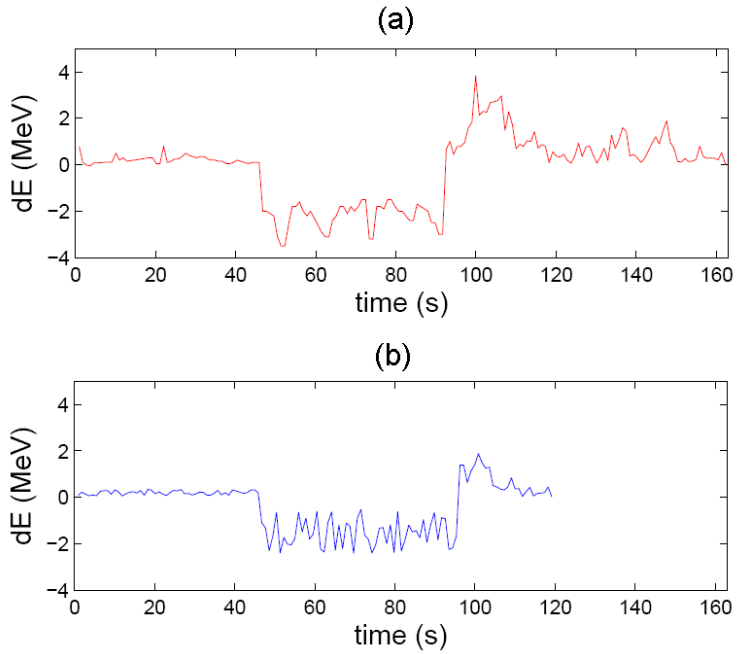



Figure 8.26: Beam energy control tests. A jump is imposed on the phase of klystron K02 at $t \simeq 50s$. At $t \simeq 95s$ the optimisation agent starts taking corrective action by adjusting the phases of K03 and K05. (a) The search space is set to $\pm 8°$ from the initial phase settings. (b) The search space is set to $\pm 5°$ from the initial phase settings. Figure adapted from [120].

This beam energy control experiment is simple because a jump imposes a fixed perturbation. That is, the correction is operated after the jump has occurred and the response of the actuators to the objective function is not time dependent. However, in the case of a periodic variation, the optimisation agent must be able to recognise time correlations within the data. In Chapter 9 we will discuss how it is possible to adapt the optimisation agent to learn to correct for periodic perturbations.

## 8.6 Discussion

The experiments carried out at the Australian Synchrotron showed that it is possible to build an optimisation agent capable of optimising the accelerator parameters. First, the optimisation agent performed independent optimisations of the transmission and energy spread (see Secs. 8.4.2 and 8.4.2, respectively). This resulted in a 5.3% increase in the transmission (from 83.5% to 88.8%) and a 1.12% decrease in the energy spread (from 1.03% to 0.91%). Second, a simultaneous optimisation of the transmission and energy spread was implemented using an objective function that assigned

equal weights to both parameters (see Sec.8.4.2). A 7.0% increase was obtained for the transmission (from 90% to 97%) and a 0.13% decrease in the energy spread (from 1.04 to 0.91%). These results demonstrate the efficacy of the method.

Experiments carried out on the FERMI accelerator showed that the optimisation agent can be used as a controller (see Sec. 8.5.3), which is capable of restoring the energy of the beam to its initial setting when a static jump is applied to the phase of one of the accelerating sections. Since a controller attempts to minimise the amplitude of a perturbation, in this case the objective function was the difference between the desired value of the beam energy and its measured value. The optimisation agent acts as a controller, and experimental results demonstrated the successful suppression of the energy perturbation.

Experiments carried out at FERMI also showed that the optimisation agent can be used as a diagnostic tool to find potential correlations between variables. Section 8.5.2 reports an experiment that was conducted to minimise the energy jitter after the L0 section, by trying to adjust the strength of the gun solenoid and a vertical corrector located between the gun and the L0 section. However, no correlations were observed between the settings of the magnets and the observed energy jitter, i.e. the optimisation agent could not find trends during the search.

In the present work the optimisation agent is designed to operate with time independent parameters and is limited to 2D search spaces. Indeed, the perturbation applied to the phase of the L0 section was a step function. However, if a cyclic (harmonic) perturbation is applied, the control system would not work, because for fixed settings of the machine (i.e. phase of the accelerating section), the perturbation would have a different effect on the beam, depending on the phase of the sinusoid at the time of the measurements. In Chapter 9 we propose two approaches to extend the optimisation tool to handle time dependent parameters and work in N-dimensional search spaces.

# 9 Future directions and conclusions

## 9.1 General considerations

In Chapter 7 we demonstrated the use of NNets to predict a deviation in the energy and bunch length of the beam, thereby allowing for corrective actions. In Chapter 8 NNets were used to build an optimisation tool, which was employed to maximise beam transmission and minimise the energy spread of the beam. This tool was tested for static perturbations applied to the phase of one of the accelerating structures. In the present chapter we extend the control system in order to build a controller that can adapt to changes in jitter conditions and machine settings. Building on the ideas developed in Chapters 7 and 8, we propose a controller that does not require the use of a response matrix. Such a system combines the following properties of the NNet predictor and the optimisation agent, developed in Chapters 7 and 8. Specifically, we require a controller that will:

- Overcome the problem of limited bandwidth and poor response at high frequencies;
- Have a limited range of action and is bounded, thereby ensuring a response that avoids instability in the beam parameters;
- Have a selective memory that allows it to only use data relevant to the current state of the optimisation agent, thereby making it computationally inexpensive;
- Be able to learn online from its interaction with the accelerator and through time.

In addition to the above requirements, the controller should address the limitations of the NNet predictor and the optimisation tool. In the feedforward - feedback control scheme developed in Chapter 7, the system required knowledge of a response matrix. As discussed in Sec. 7.5.4, the elements of this matrix should be updated dynamically, as they are functions

of the accelerator parameters (i.e. phase and voltage of the accelerating structures). Measurements of, and computations with, this matrix bring unwanted errors that degrade the accuracy of the controller (see Sec. 7.5.4); it would therefore be advantageous to construct a system that does not require the use of the response matrix.

In the experiments reported in Chapter 7, the energy and the bunch length were controlled at a single location along the accelerator (i.e. either at a dog leg or at a bunch compressor, but not both simultaneously). However, as described in Sec. 5.1, the energy and the bunch length must be stabilised at different locations (stages) of the machine. It is therefore necessary to construct a controller that can operate the control at multiple stages, since accounting for upstream corrections will influence the outcome of downstream corrections.

In Chapter 8, the system was designed to operate in a 2D search space. This limitation must also be addressed so that the optimisation agent can account for any number of parameters of the accelerator. The controller should have the following additional properties; namely, it should:

- Not require prior training, nor tuning and avoid using a response matrix;
- Be able to operate multi-stage control through a multi-step learning process;
- Not be limited in the number of adjustable parameters that can be accommodated.

In what follows we propose two possible schemes to address the above requirements. The first scheme consists of a NNet that receives lagged values of the actuators, in a similar way to the system described in Chapter 7. In that scheme, the NNet models the relation between the actuators and the residual deviations in the beam parameters as a function of time. New settings of the actuators are then chosen using inverse modeling (as discussed in Sec. 8.3.2). The second approach is based on rtNEAT, the real-time adaptation of NEAT (see Sec. 6.4). Here, the NNet's output correspond directly to new settings of the actuators. A major difference with the first approach is that the training must be operated with reinforcement learning (see Sec. 6.2.3).

The present chapter is organised as follows. Section 9.2 describes how to augment the optimisation agent to include searches in an N-dimensional space. Sections 9.3 and 9.4 describe the substructures of the system, which accommodates the desired additional properties listed above.

## 9.2 Generalisation of the optimisation agent to an N-dimensional search space

An important extension of the system consists in generalising the technique from a 2D search space to a N-dimensional search space, without imposing limitations on $N$. To accomplish this generalisation, all of the three substructures described in Sec. 8.3 need to be revisited.

### 9.2.1 Boundary-based navigation substructure

Let us first consider the boundary-based navigation substructure. Because this substructure was initially built for a 2D search space it requires significant adaptation. Previously the NNet had two outputs $r_k$ and $\phi_k$, corresponding to the "move forward" and "turn" actions taken at step $k$. The choice of polar coordinates was related to the knowledge-based structure that encodes instructions such as "if boundary in front then turn" (see Sec. 6.5). Now let us consider what happens when we try to generalise this scheme to a 3D case with spherical coordinates. The situation is illustrated in Fig. 9.1. The position of the optimisation agent in the search space at step $k$ is given by $\vec{P}_k = (x_k, y_k, z_k)$, where $x_k$, $y_k$ and $z_k$ are the corresponding values of the actuators X, Y and Z, respectively. The vector $\overrightarrow{r}_{dir.,k}$ gives the orientation of the optimisation agent at step $k$ and is defined by $\overrightarrow{r}_{dir.,k} = \vec{P}_{k-1} - \vec{P}_k$. In spherical coordinates the outputs of the NNet are $r_k$, $\phi_k$ and $\psi_k$. We also denote by $\overrightarrow{u}_{r,k}$, $\overrightarrow{u}_{\phi,k}$ and $\overrightarrow{u}_{\psi,k}$ the unit vectors associated with the variables $r_k$, $\phi_k$ and $\psi_k$, respectively (see Fig. 9.1).

In the 2D scheme the NNet was fed with 5 inputs whose values were inversely proportional to the distance to a boundary (see Sec. 8.3.1). These inputs were chosen by considering that the optimisation agent could only move through a maximum angle of $\pm\frac{\pi}{2}$ from the direction it is facing (see Fig. 8.2). This choice was arbitrary and if we were to choose an angle $\pm\pi$, the structure would include the eight inputs numbered in Fig. 9.1 (i.e. 0, $\pm\frac{\pi}{2}$, $\pm\frac{\pi}{4}$, $\pm\frac{3\pi}{4}$ and $\pi$, with reference to the optimisation agent's direction vector $\overrightarrow{r}_{dir,k}$), contained in the plane defined by the unit vectors $\overrightarrow{u}_{r,k}$ and $\overrightarrow{u}_{\phi,k}$ (shown in gray in Fig. 9.1).

With the third variable $\psi_k$, we define four planes with angles $\psi_k = 0, \pm\frac{\pi}{4}, \frac{\pi}{2}$, where $\psi_k = 0$ corresponds to the plane defined by the unit vectors $\overrightarrow{u}_{r,k}$ and $\overrightarrow{u}_{\phi,k}$ (see Fig. 9.1). We also denote by $\overrightarrow{u}_{37,k}$, the axis containing the input vectors 3 and 7 (see Fig. 9.1). When the plane is rotated around this axis (by an angle $\psi$), all four planes have the input vectors 3 and 7 (shown in blue) in common. Therefore there will be 8 distinct inputs for the first plane and

6 distinct inputs for the three other planes. That makes a total of 26 inputs to the NNet. Without rotating the plane around a particular axis $\vec{u}_{37,k}$ we would have to provide the NNet with 32 inputs, as we would have 8 distinct input vectors for each plane. Adopting this technique, in 1D we would require 2 inputs (one in each direction), in 2D we would require 8 inputs and 32 in 3D. The number of inputs consequently increases very rapidly with the dimension $N$ of the search space, according to:

$$n_{inputs} = 2^{2N-1}. \tag{9.1}$$

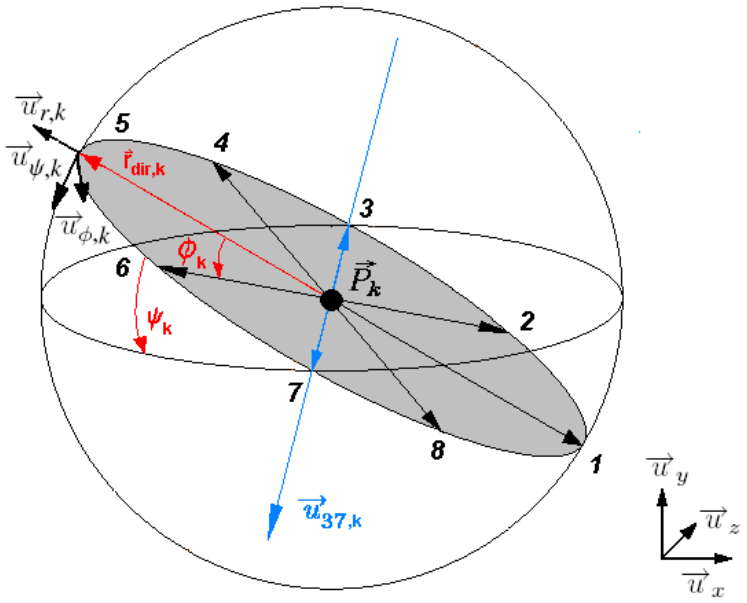



Figure 9.1: Spherical coordinates for the boundary-based navigation substructure of the optimisation agent. At step $k$, the optimisation agent is located at $\vec{P}_k = (x_k, y_k, z_k)$. In the plane (shown in gray) containing the vectors $\vec{u}_{r,k}$ and $\vec{u}_{\phi,k}$, we define eight inputs. A total of four planes are defined with angles $\psi = 0, \pm\pi/4, \pi/2$ from the plane defined by $\vec{u}_{r,k}$ and $\vec{u}_{\phi,k}$, corresponding to a total of 26 independent inputs.

Both the system of inputs described for the 2D search space (with polar coordinates) and the 3D search space (with spherical coordinates) are defined from the direction $\vec{r}_{dir,k}$ faced by the optimisation agent at step $k$. Because $\vec{r}_{dir,k}$ changes at each step, the associated input vectors will also change direction. This is useful for a game agent in 2D that needs to detect "obstacles" as illustrated in Fig. 9.2. This figure compares a game agent receiving inputs with fixed directions in cartesian coordinates (left panel) and a game agent receiving inputs in polar coordinates, defined from the

direction $\overrightarrow{u}_{dir,k}$ faced by the agent. When a the agent (in blue) goes towards the obstacle (in grey), the agent with fixed input vectors (Fig. 9.2 (a)) cannot detect the obstacle, whereas the agent receiving inputs in polar coordinates (Fig. 9.2 (b)) can identify the obstacle. However, when there are only fixed upper and lower boundaries, it is sufficient to use input vectors with fixed directions. This is illustrated in Fig. 9.2, where both agents (in brown) with fixed (Fig. 9.2 (a)) and dynamical (Fig. 9.2 (b)) input vectors can identify the upper and lower boundaries.

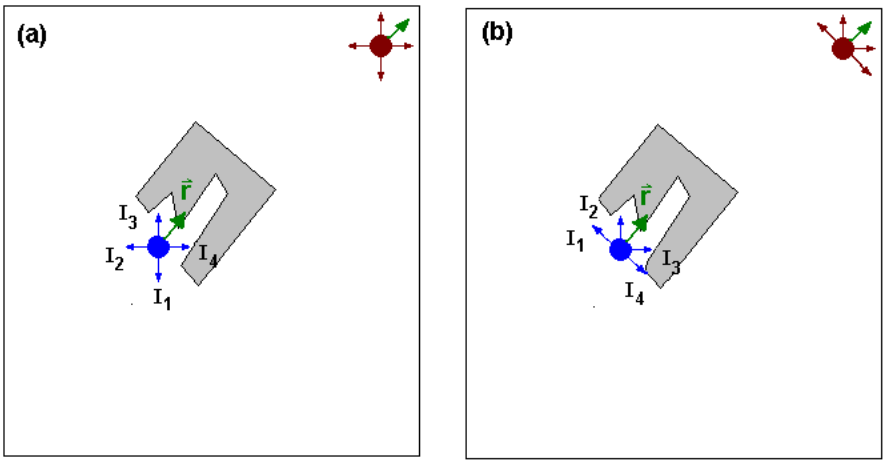



Figure 9.2: Comparison of navigation schemes. (a) The NNet receives inputs in cartesian coordinates, independent of the state of the optimisation agent (i.e. the direction it faces). (b) The NNet receives inputs depending on $\overrightarrow{r}$ in polar coordinates with reference to the position of the optimisation agent in the search space.

For optimisation purposes we are only concerned with remaining within upper and lower limits of the actuators. We can therefore use cartesian coordinates in order to reduce the number of inputs. By doing so, the outputs of the NNet correspond to some displacement in $x$ and $y$ in the search space. As illustrated in Fig. 9.2, the NNet only needs two inputs for each dimension; one to detect an upper limit and a second to detect a lower limit. The number of inputs thus increases as:

$$n_{inputs} = 2N. \tag{9.2}$$

For $N = 8$ dimensions the NNet only requires 16 inputs, whereas according to Eq. (9.1) 32,768 inputs would be required with the previous scheme. Figure 9.3 shows the minimal structure for the proposed navigation substructure in cartesian coordinates. With this substructure the NNet has $2N$ inputs (two inputs for each dimension of the search space), and $N$ outputs

(one output for each dimension). Each pair of inputs $(D_i^-, D_i^+)$ detects the presence of an upper or lower boundary, respectively, in the $i^{th}$ dimension of the search space. The output $D_i$ gives the size of the step to take in the $i^{th}$ dimension. Because the input vectors are either perpendicular or parallel to the boundaries (i.e. each input vector can detect one boundary only), there is *a priori* no need to connect inputs and outputs related to different dimensions (i.e. inputs $(D_i^-, D_i^+)$ only need to be connected to output $D_i$). The scheme shown in Fig. 9.3 is therefore the minimal structure and it is expected that only the weights of this structure should need to be evolved by NEAT. However, future studies need to be carried out to confirm whether this structure is sufficient, or if more than just the weights need to be evolved.

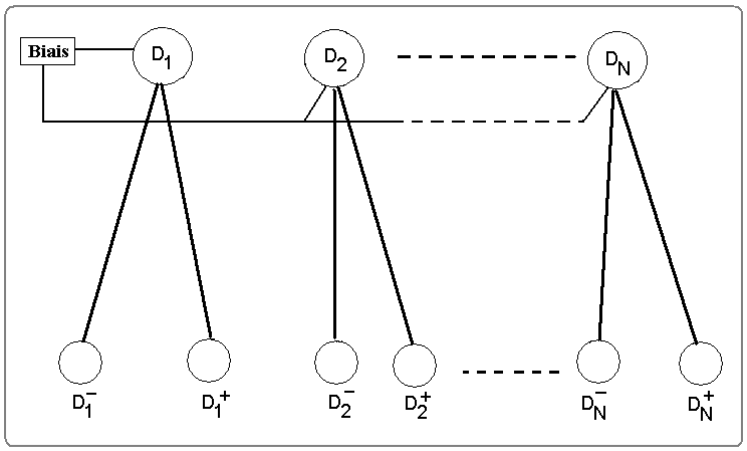



Figure 9.3: Boundary-based navigation structure of the NNet using cartesian coordinates. The NNet has $2N$ inputs providing information on the presence of an upper or lower boundary and has $N$ outputs, each corresponding to the step in each dimension $D_i$ with $i = 1, 2, ..., N$.

### 9.2.2 Performance-based navigation and local maxima avoidance substructures

The performance-based navigation substructure depicted in Fig. 8.12 must be augmented to accommodate $N$ input vectors, where each input vector corresponds to one of the search space variables. However, there is still a single output, corresponding to the value of the objective function. As with the 2D case, past trials used for training this substructure are selected if they are within a certain radius $R$ of the optimisation agent in the search space. In an N-dimensional search space, the distance between two points $\vec{x}_1, \vec{x}_2 \in \mathbb{R}^N$ is given by the Euclidean metric:

$$d = \sqrt{(x_1^1 - x_2^1)^2 + (x_1^2 - x_2^2)^2 + ..... + (x_1^N - x_2^N)^2}, \tag{9.3}$$

where the superscript is the dimension index. To summarise, the only changes to this substructure consist of increasing the number of inputs to the NNet and selecting the points according to their distance $R$ from the optimisation agent using Eq. (9.3).

Equation (9.3) is also required in order to generalise the local maxima avoidance substructure to N-dimensions. As for the 2D case, when the density of trials reaches a given threshold $D_{thres.}$, the local minimum avoidance substructure must relocate the optimisation agent to a new region in the search space as described in Chapter 8. In N-dimensions, the distance between a past trial and the optimisation agent is calculated using Eq. (9.3). If the distance is less than a given radius $R$ from the optimisation agent, the trial is taken into account to calculate the local density $D_k$. The substructure of the NNet itself (i.e. number of inputs and outputs) does not need to be modified, as the only change required to adapt the local maximum avoidance substructure to N-dimensions consists of calculating distances using Eq. (9.3). The generalisation of the performance navigation and avoidance substructures are therefore straightforward. Consequently, the optimisation agent can be adapted to a search space with an arbitrary number of dimensions.

## 9.3 Building a time-dependent optimisation agent for control

Now that we have described how to generalise the optimisation agent to N-dimensional search spaces, we need to consider how to adapt its structure to correct for time-dependent perturbations. In Sec 8.5.3 it was shown that the optimisation agent can be used as a controller to correct a static perturbation. However, for a periodic (harmonic) perturbation the NNet must be provided with inputs that contain the necessary information on the time-dependent perturbation in order to produce the appropriate response of the actuators. In Chapter 7 the NNet predictor was provided with lagged values of the accelerator parameters (phase and voltage) to predict a future deviation in the beam parameters (energy and bunch length). In what follows we show how the structure of the NNet predictor described in Chapter 7 can be combined with the optimisation agent in Chapter 8 to produce a real-time optimisation agent.

In Chapter 8 the performance-based substructure of the NNet built a local map of an objective function $H$ for a 2D search space, using past trials within a given radius $R$ of the optimisation agent's location. This substructure received two input vectors, $\overrightarrow{X}^1$ and $\overrightarrow{X}^2$, containing the coordinates of the past trials in the search space (see Sec. 8.3.4):

$$Inputs = \begin{cases} \overrightarrow{X}^1 = [x_1^1, x_2^1..x_l^1] \\ \overrightarrow{X}^2 = [x_1^2, x_2^2..x_l^2], \end{cases} \tag{9.4}$$

where $x_j^i$ (with $j = 1, ..., l$ and $i = 1, 2$) is the coordinate of the $j^{th}$ trial in the $i^{th}$ dimension and $l$ is the total number of past trials in the local neighborhood (within a given distance $R$ from the optimisation agent). In Chapter 8 we assumed that the relation between the beam parameters (and consequently the objective function) and the variables $X^1$ and $X^2$ (i.e. the settings of the actuators) was time invariant. However, to correct a time-dependent perturbation, it is necessary to include the time dependence in our model. We can rewrite the inputs to the NNet given in Eq. (9.4) as a function of the step $k$ of a time series and generalise the number of inputs to N-dimensional search spaces, spanned by $X^1, X^2, ..., X^N$. At step $k$ the inputs to the NNet are given by the following vectors:

$$Inputs = \begin{cases} \overrightarrow{X}^1(k) = [x_1^1(k), x_2^1(k)..x_l^1(k)] \\ \overrightarrow{X}^2(k) = [x_1^2(k), x_2^2(k)..x_l^2(k)] \\ ... \\ \overrightarrow{X}^N(k) = [x_1^N(k), x_2^N(k)..x_l^N(k)]. \end{cases} \tag{9.5}$$

Now let us consider what information a time-dependent optimisation agent should receive. In Chapter 7 we used NNets to predict a deviation in the energy of the beam, which required $n$ lagged values of the voltage $V$ and $m$ lagged values of the phase $\phi$ of an accelerating section (see Sec. 7.2). The NNet was thus fed with $n$ inputs for the voltage lags, $V(k), V(k-1), ..., V(k-n-1)$ and $m$ inputs for the phase lags, $\phi(k), \phi(k-1), ..., \phi(k-m-1)$. To be consistent with the notation used in Eq. (9.5), we denote the voltage variable $V$ and the phase variable $\phi$ by $X^1$ and $X^2$, respectively. Using this notation the NNet is fed with the following inputs:

$$Inputs = \begin{cases} X^1(k) \\ X^1(k-1) \\ ... \\ X^1(k-n-1) \\ X^2(k) \\ X^2(k-1) \\ ... \\ X^2(k-m-1), \end{cases} \tag{9.6}$$

Note that each of these inputs corresponds to a single value of the phase and voltage of the accelerator, i.e. they are not records of past trials. We can generalise these inputs to $N$ variables $X^1$,$X^2$...$X^N$, denoting by $n^i$ the number of lags used for the variable $X^i$:

$$Inputs = \begin{cases} X^1(k) \\ X^1(k-1) \\ ... \\ X^1(k-n^1-1) \\ X^2(k) \\ X^2(k-1) \\ ... \\ X^2(k-n^2-1) \\ ... \\ ... \\ X^N(k) \\ X^N(k-1) \\ ... \\ X^N(k-n^N-1), \end{cases} \tag{9.7}$$

where $X^i$ (with $i = 1, 2, ..., N$) is the value of the $i^{th}$ parameters of the accelerator (i.e. phase or voltage of the accelerating sections). To merge the structure of the NNet predictor with the NNet used for the optimisation agent, each input in Eq. (9.7) must contain the coordinates (values) of all the past trials for each lag. The inputs in Eq. (9.7) must therefore be vectors containing the coordinates of the past trials in the local neighborhood. Every time the optimisation agent takes a step in the search space, the past trials within the local neighborhood change. At step $k$, the corresponding number of past trials is $l(k)$. Replacing each of the values $X^i(L)$ (where $L = k-1-n, k-n, ..., k$ and $n$ is the number of lags) with the corresponding vectors $\overrightarrow{X}^i(L)$ containing the coordinates of the past trials (9.5), we obtain:

$$
Inputs = \begin{cases}
\overrightarrow{X^1}(k) = [x_1^1(k), x_2^1(k)...x_{l(k)}^1(k)X] \\
\overrightarrow{X^1}(k-1) = [x_1^1(k-1), x_2^1(k-1)...x_{l(k)}^1(k-1)] \\
... \\
\overrightarrow{X^1}(k-n-1) = [x_1^1(k-n-1), x_2^1(k-n-1)...x_{l(k)}^1(k-n-1)] \\
\overrightarrow{X^2}(k) = [x_1^2(k), x_2^2(k)...x_{l(k)}^2(k)] \\
\overrightarrow{X^2}(k-1) = [x_1^2(k-1), x_2^2(k-1)...x_{l(k)}^2(k-1)] \\
... \\
\overrightarrow{X^2}(k-n-1) = [x_1^2(k-n-1), x_2^2(k-n-1)...x_{l(k)}^2(k-n-1)]... \\
... \\
... \\
\overrightarrow{X^N}(k) = [x_1^N(k), x_2^N(k)...x_{l(k)}^N(k)] \\
\overrightarrow{X^N}(k-1) = [x_1^N(k-1), x_2^N(k-1)...x_{l(k)}^N(k-1)] \\
... \\
\overrightarrow{X^N}(k-n-1) = [x_1^N(k-n-1), x_2^N(k-n-1)...x_{l(k)}^N(k-n-1)],
\end{cases}
\tag{9.8}
$$

In Eq. (9.8) all the variables have the same number of lags $n$. This is because it is necessary to provide all the coordinates of past trials to the NNet in order to model the objective function. According to Eq. (9.8), the inputs consist of $n$ vectors (lags) for each variable of the N-dimensional search space, which contains the coordinates of the past trials for a given lag.

Figure 9.4 shows a schematic of the proposed NNet for a 2D scheme. The NNet models the relation between the value of the objective function $H(k)$ at step $k$, given the vectors containing the coordinates of the past trials for a series of $n$ lags. Unlike the NNet predictor developed in Chapter 7, here the NNet does not produce a prediction of $H(k+1)$, given past lagged inputs. Predicting $H(k+1)$ given the inputs in Eq. (9.6) would not be of interest, since it would require knowledge of the response matrix to calculate the appropriate correction. Instead, we need to directly predict what settings of the actuators $x^1(k+1)$ and $x^2(k+1)$ would best attenuate the perturbation. This is done by evaluating the value of the objective function for a series of points $P = [p_1, p_2...p_q]$, with coordinates $p_i = (x_i^1(k), x_i^2(k))$ chosen within a given radius of the optimisation agent. The new settings $(x^1(k+1), x^2(k+1) = (x_s^1, x_s^2))$ for the actuators are then chosen such that $H(x_s^1, x_s^2) = min(H(P))$, where $(x_s^1, x_s^2) \in P$.

This scheme has the advantage of calculating new values for the settings of the actuators without requiring knowledge of the response matrix. This avoids errors related to the measurement of the matrix, particularly since it must be re-measured whenever there is a change in the settings (phase and voltage) of the accelerator (see Sec. 7.5.4).

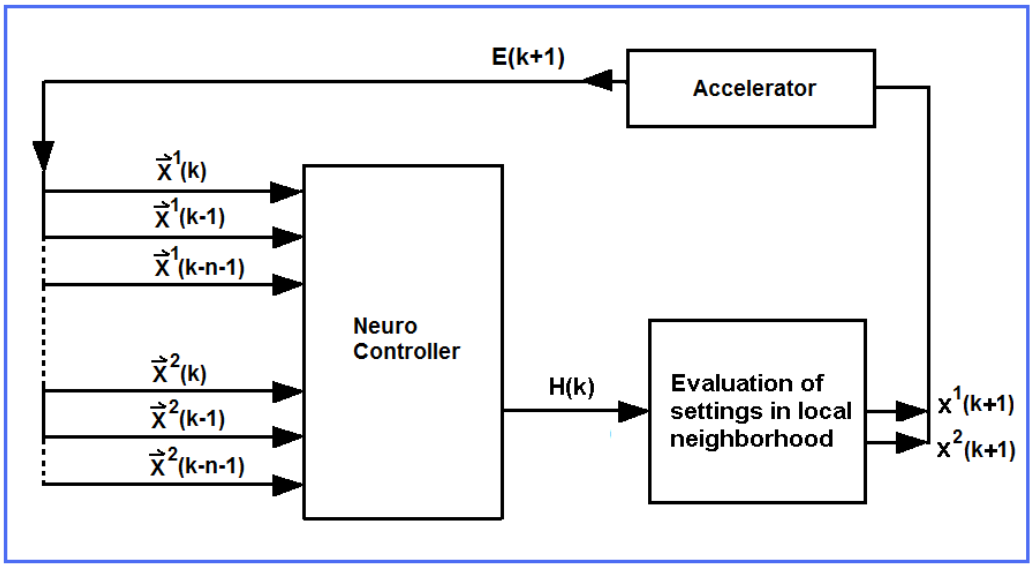



Figure 9.4: Control scheme using the time-dependent optimisation agent. The NNet is fed with the vectors containing the coordinates of past trials for $n$ lags and models the corresponding response of the objective function. Points in the local neighborhood of the NNet (current location in the search space) are then evaluated and the configuration that is most likely to increase the value of the objective function is applied to the machine. The effect of the new settings on the beam (e.g., the energy $E(k+1)$) are recorded and used to further train the NNet.

Another advantage of the proposed scheme is that no training is required before bringing the system online, unlike the system discussed in Chapter 7, where data had first to be recorded in order to train the NNet. The real-time optimisation agent will try different configurations of the actuators and learn from these trials. Also, the number of actuators is not limited, since the optimisation agent can be generalised to $N$ dimensions. In order to develop the time-dependent optimisation agent, the following steps need to be implemented:

1. Adapt the number of inputs to the agent. Inputs should include $n$ lagged vectors containing the coordinates of past trials for each of the $N$ dimensions of the search space.
2. Adapt the optimisation algorithm. The time-dependent optimisation agent should not be able to relocate in the search space when acting as a controller. However, relocation should be permitted when it is used to re-tune the machine parameters (i.e. when the requirements on the beam parameters are changed).
3. Carry out real-time training. This training is essentially the same as for the static optimisation; at every step new data are collected and the NNet is trained with past trials contained within the local neighborhood of the time-dependent optimisation agent. The controller can be built using Matlab, by adapting the code already implemented for the optimisation agent described in Chapter 8.

## 9.4 Using the rtNEAT technology for control

We present a more sophisticated approach to building a NNet controller with the properties listed in Sec. 9.1. The intention is to combine the characteristics of the NNet predictor and the optimisation agent, so that the NNet's outputs directly correspond to the new settings of the actuators (i.e. without requiring the inverse modeling used in Sec. 9.3). In the previous section, the NNet was used to model the objective function $H(k)$ as a function of past trials for a series of $n$ lags:

$$\begin{aligned} H(k) = H(&\overrightarrow{X}^1(k), \overrightarrow{X}^1(k-1), ..., \overrightarrow{X}^1(k-n-1), \\ &\overrightarrow{X}^2(k), \overrightarrow{X}^2(k-1), ..., \overrightarrow{X}^2(k-n-1), \\ &..., \\ &\overrightarrow{X}^N(k), \overrightarrow{X}^N(k-1), ..., \overrightarrow{X}^N(k-n-1)), \end{aligned} \tag{9.9}$$

where the vectors $\overrightarrow{X}^i(L)$ (with $i = 1, 2, ..., N$ and $L = k-n-1, k-n, ..., k$) constitute the inputs to the NNet defined by Eq. (9.8). In order to avoid calculating the new settings of the actuators by inverting the model of the objective function built by the NNet, the present scheme uses a NNet to directly model the optimised values of the actuators' settings, i.e., $(x^1_{opt.}(k+1)$, $x^2_{opt.}(k+1)$, ..., $x^N_{opt.}(k+1))$. To do this the NNet must correlate the values obtained for the objective function with past trials. At step $k$ the value of the

objective function for the $j^{th}$ past trial ($j = 1, 2, ..., l(k)$) can be written as a function of the coordinates of that trial:

$$H_j(k) = H(x_j^1(k), x_j^2(k), ..., x_j^N(k)). \tag{9.10}$$

The vector containing the values for the $l(k)$ past trials in the local neighborhood at step $k$ can be written as:

$$\overrightarrow{H}(k) = [H_1(k), H_2(k), ..., H_{l(k)}(k)]. \tag{9.11}$$

To correlate past trials in the search space with the corresponding values of the objective function, the NNet is fed with the coordinates of the past trials and the corresponding values of the objective function. Since the location of the optimisation agent in the search space changes at every step, we must provide the NNet with lags for the past trials and objective function. For each lag, the NNet is provided with the coordinate vectors of the past trials and the vector containing the corresponding values of the objective function. We can write the coordinates of the optimised values of the actuators ($x_{opt.}^1(k+1)$, $x_{opt.}^2(k+1)$, ..., $x_{opt.}^N(k+1)$) as a function of the past trials $F_{NNet}$ and the corresponding values of the objective function (for a series of lags) as:

$$\begin{aligned}(x_{opt.}^1(k+1), x_{opt.}^2(k+1)...x_{opt.}^N(k+1)) = F_{NNet}(&\overrightarrow{X}^1(k), \overrightarrow{X}^1(k-1)...\overrightarrow{X}^1(k-n-1),\\ &\overrightarrow{X}^2(k), \overrightarrow{X}^2(k-1)...\overrightarrow{X}^2(k-n-1),\\ &...,\\ &\overrightarrow{X}^N(k), \overrightarrow{X}^N(k-1)...\overrightarrow{X}^N(k-n-1),\\ &\overrightarrow{H}(k), \overrightarrow{H}(k-1)...\overrightarrow{H}(k-n-1)),\end{aligned} \tag{9.12}$$

where each input vector to the NNet is given by:

$$
Inputs = \left\{ \begin{array}{l}
\overrightarrow{X^1}(k) = [x_1^1(k), x_2^1(k)...x_l^1(k)] \\
\overrightarrow{X^1}(k-1) = [x_1^1(k-1), x_2^1(k-1)...x_l^1(k-1)] \\
... \\
\overrightarrow{X^1}(k-n-1) = [x_1^1(k-n-1), x_2^1(k-n-1)...x_l^1(k-n-1)] \\
\overrightarrow{X^2}(k) = [x_1^2(k), x_2^2(k)...x_l^2(k)] \\
\overrightarrow{X^2}(k-1) = [x_1^2(k-1), x_2^2(k-1)...x_l^2(k-1)] \\
... \\
\overrightarrow{X^2}(k-n-1) = [x_1^2(k-n-1), x_2^2(k-n-1)...x_l^2(k-n-1)]... \\
... \\
... \\
\overrightarrow{X^N}(k) = [x_1^N(k), x_2^N(k)...x_l^N(k)] \\
\overrightarrow{X^N}(k-1) = [x_1^N(k-1), x_2^N(k-1)...x_l^N(k-1)] \\
... \\
\overrightarrow{X^N}(k-n-1) = [x_1^N(k-n-1), x_2^N(k-n-1)...x_l^N(k-n-1)] \\
\overrightarrow{H}(k) = [H_1(k), H_2(k)...H_l(k)] \\
\overrightarrow{H}(k-1) = [H_1(k-1), H_2(k-1)...H_l(k-1)] \\
... \\
\overrightarrow{H}(k-n-1) = [H_1(k-n-1), H_2(k-n-1)...H_l(k-n-1)].
\end{array} \right. \tag{9.13}
$$

These inputs correspond to those given in Eq. (9.8), augmented with the vectors containing the values of the objective function. Because we do not know in advance what the optimised settings of the actuators should be, we cannot use supervised training (such as the backpropagation algorithm discussed in Sec. 6.2.1). Indeed, we do not know the relation between the objective function and the response matrix, or the jitter conditions. The optimisation agent must try to adjust the accelerator to correct a deviation in the beam parameters. We need a way to tell it how well it has performed. This corresponds to reinforcement learning discussed in Sec. 6.2.3, where the NNet is not told what the outputs should be, but where an appreciation is given. When the time-dependent optimisation agent takes an action, it is rewarded or penalised according to the outcomes of its actions (see Sec. 6.2.3). For example, let us assume that we want to stabilise the energy of the beam and train the NNet with reinforcement learning. At step $k$ the optimisation adjusts the actuators in order to reduce the energy deviation. The outcome is either an increase or a decrease in the residual deviation of the beam energy from its desired value. The residual deviation can be used as a reinforcement signal, with the NNet rewarded if the deviation

decreases and penalised if it increases. In this case the objective function $H(k)$ can be written as a function of the residual energy deviation and used to train the NNet.

Providing information on both the actuators' settings and the corresponding values of the objective function is important to ensure that the NNet is always provided with information on the perturbation. This is illustrated with the following simple example. Let us assume that we want to correct an energy deviation from a reference energy $E_0$ with actuator $X$ only. We use the objective function in Sec. 8.5.3 for the FERMI experiment:

$$H(E(k)) = E(k) - E_0. \tag{9.14}$$

The measured energy is considered to have a harmonic perturbation of the form:

$$E(k) = E_0 + dE\,sin(2\pi fkT), \tag{9.15}$$

where $f$ is the jitter frequency, $dE$ is the jitter amplitude and $T$ the time interval separating bunches (i.e. $1/T$ is the bunch repetition rate). The objective function (9.14) now becomes:

$$H(E(k)) = dE\,sin(2\pi fkT), \tag{9.16}$$

which has the same sinusoidal time dependence as the jitter. When the perturbation is entirely canceled $H(E(k)) = 0$, for all $k$. Since we know the response of the actuators is linear in the region needed to operate the correction, one can write $dE(k) = AdX(k)$, where $A$ is the proportionality constant. To achieve $H(E(k)) = 0$ (for all $k$), the response of the actuator $X(k)$ must be:

$$X(k) = -\frac{dE(k)}{A} sin(2\pi fkT). \tag{9.17}$$

This means that if the NNet operates its task correctly, the information on the jitter is contained within the data of past values of the actuator $X$. Now let us consider the case where the time-dependent optimisation agent starts the control and the NNet is untrained. The information on the jitter is no longer contained in the inputs; however, it is still contained in the lagged values of the objective function $H(E)$. By feeding the NNet with lagged values of $H(E)$, we allow it to make correlations between the settings of the actuators and the corresponding values of the objective function. When the time-dependent optimisation agent is able to perform the control task, the values of the lags $H(k-n-1)$,$H(k-n)$..$H(k)$ will be zero and the information on the jitter will be contained in past values of the correctors only. Thus, lagged values of the actuators and lagged values of the objective function

provide complementary information and must both be fed to the NNet. Alternatively, lagged values of the remaining perturbations could be fed to the NNet when the control is operated at different stages of the accelerator. This is particularly important for multi-stage control, where the NNet must distinguish between the effects of the actuators at different stages of the machine.

Reinforcement training can be applied to a NNet with fixed topology (i.e. fixed number of neurons and connections). However, the appropriate topology of the NNet depends on many parameters, including the number of inputs and the desired level of precision. Since these parameters can change over time, NEAT is a good candidate as it evolves the whole NNet structure. However, NEAT is not adapted to real-time applications, but its variant rtNEAT can be used. As discussed in Chapter 6, rtNEAT was developed for real-time evolution of agents in video games. The major difference is that in rtNEAT individuals are tested one at a time and for a restricted period (i.e. the interaction of the game agent with its environment is restricted to a fraction of the time taken for the whole game). Although one could argue that testing a whole population can be very time consuming, in an accelerator the bunch repetition rate is very high (10 Hz to 120 Hz depending on the machine), and the sampling frequency can match these rates if the communication with the actuators and diagnostics permits it. Consequently, the interaction time can be extremely short, since the NNet agent interacts with the machine at the beam rate. The LCLS is a particularly good candidate for future experiments to explore this strategy, since it can operate at 120 Hz. The limit is determined by the lowest frequency the NNet is able to recognise - the lower the frequency, the longer the training time.

At each step $k$ of a time series, the rtNEAT controller applies some change to the accelerator settings (i.e. changes to the phase or voltage of the accelerator) and the corresponding effects on the beam parameters are recorded (e.g., the energy or bunch length). The vectors containing these records are then fed to the NNet (see Fig. 9.5). The NNet computes output values that correspond to the next settings of the actuators. The effects of that action are measured and the value of the objective function is calculated to tell the NNet how well it has performed. If the NNet has decreased the deviation it will be rewarded, otherwise it is penalised according to the fitness function. Input vectors are then updated to take into account the results of the last step and the cycle continues until the NNet has finished its evaluation phase. A new NNet is then loaded for evaluation and rtNEAT selects the fittest individuals to create offspring. The process continues until a NNet reaches the level of fitness required to act as the operating controller.

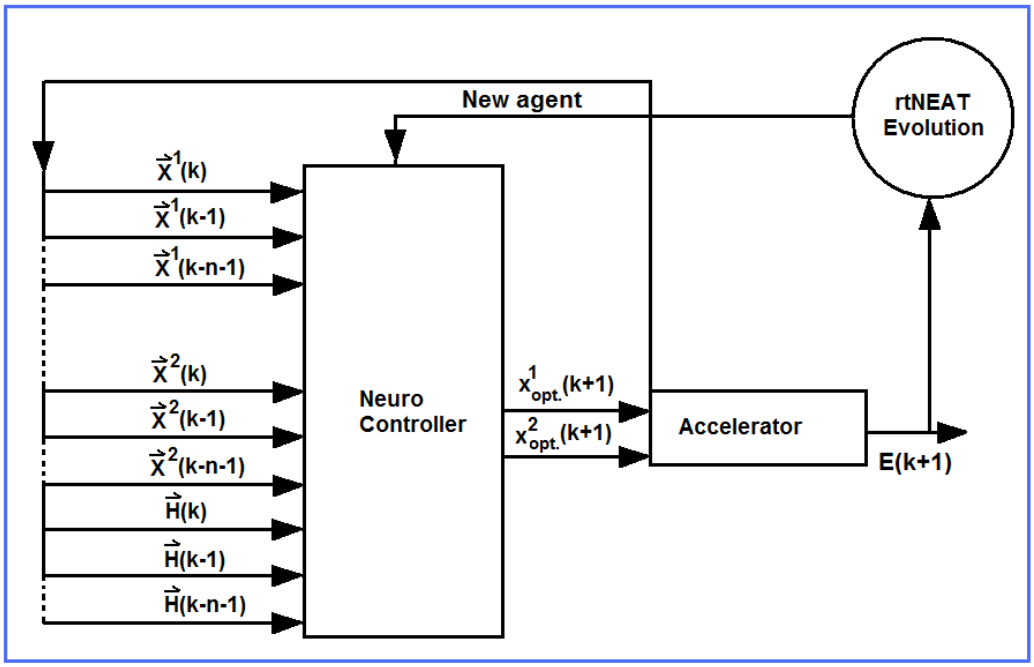



Figure 9.5: Structure of a rtNEAT-based neuro controller. The NNet is provided with lagged values of the actuators and lagged values of the objective function. Its output gives the values of new settings of the actuators, which are applied to the accelerator. The effect is measured on the beam (for example the energy $E(k)$) and used as a reinforcement signal to train the NNet with rtNEAT.

An important advantage of this method, compared to the approach developed in Chapter 7, is that the system does not require prior knowledge of the jitter source. Indeed, in Chapter 7 the NNet was trained with data containing the jitter (i.e. klystron phase and voltage). The occurrence of jitter in any other component would have invalidated the control. However, the system learns the relation between actions taken by the actuators and their effects on the beam parameters through the objective function.

In the NERO video game, agents learn to take different actions in stages [122]. For example, they are first trained to go around obstacles, then learn how to shoot, to jump, etc. This is because developing one attribute at a time is easier than developing all of them at once. This is relevant here since the controller needs to be operated at several stages along the machine (for example at a dog leg or a bunch compressor), as discussed in Chapter 7. Because of this, upstream corrections will affect the outcome of downstream corrections. Since the same NNet must take this effect into account, it is sensible that it should learn to operate the control at the first stage, then the second stage, and so on.

Last, we must consider the controller's response to a change in the machine RF components. Here the response of the system, or response matrix, is integrated into the response of the NNet, which operates the control directly. This means that a NNet trained for some machine settings will not be successful at operating the control at different settings. The optimisation agent developed in Chapter 8 can be used to bring the machine from one configuration to another, since it is a static step. At the new set point the rtNEAT controller should then take over. We then have two options: first, when the performance of the NNet is not sufficient for the new set point (which will be the case if the NNet is not trained for unseen machine settings), rtNEAT should restart the training, since the inputs of the NNet in Fig. 9.5 do not have a memory of past states of the machine. A second possibility consists of feeding the NNet with the values of settings that can affect the Observable that should be controlled. For the beam energy this includes the voltage and the phases of accelerating sections, other than the actuators. To summarise the following steps should be taken in order to develop the time-dependent optimisation agent based on rtNEAT:

1. Adapt the number of inputs to the NNet. Inputs should include $n$ lagged values of the actuators and the perturbed variables. The number of lags is determined by the lowest frequency that we want to correct.
2. Implement the optimisation agent (e.g., in C++) and integrate it with rtNEAT.
3. Test the structure of the system, in particular the type of inputs that should be fed to the NNet. For example, it might be more relevant to feed the NNet with lagged values of the observable errors rather than the objective function. This is particularly relevant when the control is operated at multiple stages of the machine (i.e. bunch compressors, dog legs, etc).
4. Test the use of inputs that identify machine settings and determine the time required to retrain the NNet when settings are changed.

## 9.5 Conclusions

In Chapter 7, NNets were used in a feedforward-feedback scheme to compensate for limitations of the PID algorithm, which is currently used to control the longitudinal parameters (energy and bunch length) of the electron beam in Free Electron Lasers. These limitations include the poor response for high frequency jitter and limited bandwidth (frequency range for which

the PID gains do not need re-tuning). The proposed stabilisation scheme based on a NNet was first tested on the Australian Synchrotron Linac for the control of a single variable - the beam energy. Further experiments were carried out at the Linac Coherent Light Source (at SLAC), demonstrating the adaptability of the NNet controller to multiple variables, i.e. the simultaneous control of the energy and the bunch length.

We identified the limitations of the NNet-based feedforward-feedback scheme. These include the use of a response matrix to compute the correction and the need to re-train the NNet when changes in jitter conditions and machine parameters occur. To rectify these deficiencies we considered the control problem as an optimisation problem and built a NNet-based optimisation tool inspired by video game technology (see Chapter 8). This tool was successfully tested for the optimisation of the energy spread of the beam and beam transmission at the Australian Synchrotron Linac. Experiments were also carried out at the new FERMI@Elettra facility, using the optimisation tool to correct a static deviation in the energy of the electron beam. These experiments demonstrated the ability of the system to learn online and from its interaction with the machine, by making small adjustments to the parameters of the accelerator. Moreover, the experiments carried out on the FERMI machine showed the suitability of the system for control purposes.

In order for the optimisation tool developed in Chapter 8 to compensate for time-dependent perturbations (e.g. periodic jitter), we discussed briefly how to merge the structures of the NNet feedforward-feedback stabilisation scheme with the structure of the NNet used for optimisation. The resulting system is expected to form a NNet-based optimisation tool with the following important features:

1. It can learn online, from its interaction with the accelerator and through time.
2. It has a limited range of actions and is bounded, which ensures a smooth response of the beam parameters.
3. It has a selective memory, that allows it to use data relevant to its current state, which makes it computationally inexpensive.
4. It does not require prior training, nor tuning and avoids the necessity for a dynamic response matrix.
5. It overcomes the problem of limited bandwidth and the concomitant poor response at high frequencies.
6. It does not require knowledge of what parameters caused the jitter.
7. It is not limited in the number of adjustable parameters it can accommodate.

8. It can operate multi-stage control from multi-step learning.
9. Its operation does not depend on the beam repetition rate.

Further work is necessary to complete this research project and provide an operational system, as well as to demonstrate its robustness and adaptability. The expected outcomes of this system are a novel adaptive control and optimisation tool, which does not require tuning and can accommodate any number of adjustable parameters. It is anticipated that the system can also be used for any kind of machine (i.e. it is not limited to Linac applications), and can be used to optimise and control any variable (i.e. not limited to the control and optimisation of longitudinal parameters of the electron beam). For example, this tool could be employed for beam tuning on Synchrotrons and beamlines.

In summary, this thesis has demonstrated the feasibility of NNets for the control and optimisation of beam parameters, using both simulations and realistic operational accelerator environments. The work has shown that NNets can be usefully employed in accelerator physics, and the outcomes have the potential to improve the performance of Synchrotrons and Free Electron Lasers.

# Bibliography


[1] C. W. Roberson. Free-electron laser beam quality. *IEEE Journal of Quantum Electronics*, QE-21:860–866, 1985.

[2] FERMI@Elettra conceptual design report, Sincrotrone Trieste. `http://www.elettra.trieste.it/FERMI/index.php?n=Main.CDRdocument`, January 2007.

[3] E. Allaria, G. De Ninno, and M. Trovó. Start-to-end time-dependent study of FEL output sensitivity to electron beam jitters for the FERMI@Elettra project. In *Proceedings of the Free Electron Laser Conference*. pages 178-181, 2006.

[4] J. Wu, P. Emma, and L. Hendrickson. Linac Coherent Light Source longitudinal feedback model. In *Proceedings of the Particle Accelerator Conference*. pages 1156–1158, 2005.

[5] D. P. Atherton and S. Majhi. Limitations of PID controllers. In *Proceedings of the American Control Conference*. pages 3843–3847, 1999.

[6] W. Y. Han, J. W. Han, and C. G. Lee. Development of a self-tuning PID controller based on neural network for nonlinear systems. In *Proceedings of the Control and Automation Conference*. pages 979 - 988, 1999.

[7] A. Ciaramella, C. Donalek, A. Staiano, M. Ambrosio, C. Aramo, P. Benvenuti, G. Longo, L. Milano, G. Raiconi, R. Tagliaferri, and A. Volpicelli. Application of radial basis function neural network model for short-term load forecasting. In *Recent Res. Devel. Astrophys.: ISBN: 81-7736-295-X*. 2005.

[8] S. Sklenak, V. Kvasnicka, and J. Pospichal. Application of recurrent neural networks in chemistry. Prediction and classification of carbon-13 NMR chemical shifts in a series of monosubstituted benzenes. *Journal of Chemical Information and Modeling*, 32:742–747, 1992.

[9] D. T. Manallack, D. J. Livingstone, M. A-Razzak, and R. C. Glen. Neural networks and expert systems in molecular design. In *Methods and Principles in Medicinal Chemistry*, chapter 5, pages 293–331. John Wiley and Sons, Inc., 2008.

[10] C. T. Harston. *Application of neural networks to robotics*. Academic Press Professional, Inc., 1990.

[11] C. Man-Chung, W. Chi-Cheong, and L. Chi-Chung. Financial time series forecasting by neural network using conjugate gradient learning algorithm and multiple linear regression weight initialization. Computing in Economics and Finance 2000 61, Society for Computational Economics, July 2000.

[12] C. N. W. Tan. An artificial neural networks primer with financial applications examples in financial distress predictions and foreign exchange hybrid trading system. `http://www.smartquant.com/references/NeuralNetworks/neural28.pdf`, 2005.

[13] I. B. Ciocoiu. Hybrid feedforward neural networks for solving classification problems. *Neural Processing Letters*, 16:81–91, 2002.

[14] Y. Chen, B. Yang, J. Dong, and A. Abraham. Time-series forecasting using flexible neural tree model. *Journal of Information Sciences*, 174:219–235, 2005.

[15] P. Melin, J. Urias, D. Solano, M. Soto, M. Lopez, and O. Castillo. Voice recognition with neural networks, type-2 fuzzy logic and genetic algorithms. *Engineering Letters*, 13:108–116, 2006.

[16] J. A. K. Suykens, J. P. L. Vandewalle, and B. L. R. De Moor. *Artificial neural networks for modelling and control of non-linear systems*. Kluwer Academic Publishers, 1997.

[17] X. Ren, F. L. Lewis, S. S. Ge, and J. Zhang. Neural network feedforward control for mechanical systems with external disturbances. In *Proceedings of the Decision and Control Conference*. pages 4687–4692, 2007.

[18] A. Blazina and N. Bolf. Neural network-based feedforward control of two-stage heat exchange process. In *Proceedings of the Computational Cybernetics and Simulation Conference*. pages 25–29, 1997.

[19] T. Higo, H. Shoaee, and J. E. Spencer. Some applications of AI to the problems of accelerator physics. In *Proceedings of the Particle Accelerator Conference*. pages 701-703, 1987.

[20] D. P. Weygand. Artificial intelligence and accelerator control. In *Proceedings of the Particle Accelerator Conference*. pages 564-566, 1987.

[21] D. Nguyen, M. Lee, R. Sass, and H. Shoaee. Accelerator and feedback control simulation using neural networks. In *Proceedings of the Particle Accelerator Conference*. pages 1437-1439, 1991.

[22] Y. Kijima, K. Yoshida, and M. Mizota. A beam diagnostic system for accelerator using neural networks. In *Proceedings of the European Particle Accelerator Conference*. pages 1155-1157, 1992.

[23] E. Bozoki and A. Friedman. Neural network technique for orbit correction in accelerators/storage rings. In *Proceedings of thw American Institute of Physics Conference*. pages 103-110, 1994.

[24] E. Bozoki and A. Friedman. Neural networks and orbit control in accelerators. In *Proceedings of the European Particle Accelerator Conference*. pages 1589–1591, 1994.

[25] K. H. Kim, K. Shim, J. Choi, M. H. Cho, W. Namkung, and I. S. Ko. Simulation of the global orbit feedback system for Pohang Light Source. In *Proceedings of the European Particle Accelerator Conference*. pages 1906-1908, 2000.

[26] Y. Hitaka, M. Shirakata, H. Sato, M. Yokomichi, and M. Kono. Numerical methods for the orbit control at the KEK 12 GEV-PS. In *Proceedings of the European Particle Accelerator Conference*. pages 2664-2666, 2004.

[27] 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 Instruments and Methods in Physics Research Section A*, 293:517–522, 1990.

[28] 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*, 293:271–289, 1992.

[29] J. M. Bouché, G. Daems, V. Filimonov, V. Khomoutnikov, F. Periollat, and Y. Ryabov. *Representation and usage of knowlegde for initialization of accelerator control equipment*. CERN Report Number PS-95-34 CO, 1995.

[30] N. R. Jennings, E. H. Mamdani, J. M Corera, I. Laresgoiti, F. Perriollat, P. Skarek, and L. Z. Varga. Using ARCHON to develop real-world DAI applications, part 1. *IEEE Expert: Intelligent systems and their applications*, 11:64–70, 1996.

[31] F. Perriollat, P. Skarek, L. Z. Varga, and N. R. Jennings. Using AR-

CHON, part 3: Particle acceleration control. *IEEE Expert: Intelligent systems and their applications*, 11:80–86, 1996.

[32] J. M. Corera, I. Laresgoiti, and N. R. Jennings. Using ARCHON, part 2: Electricity transportation management. *IEEE Expert: Intelligent systems and their applications*, 11:71–79, 1996.

[33] W. B. Klein and R. T. Westervelt. A general purpose intelligent control system for particle accelerators. *Journal of Intelligent and Fuzzy Systems*, 7:1–12, 1995.

[34] C. Stern, E. Olsson, M. Kroupa, R. T. Westervelt, G. Luger, and W. B. Klein. A control system for accelerator tuning combining adaptive plan execution with online learning. In *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*. pages 124-126, 1997.

[35] W. B. Klein, C. Stern, G. Luger, and E. Olsson. An intelligent control architecture for accelerator beamline tuning. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference*. pages 1019-1025, 1997.

[36] W. B. Klein, C. Stern, G. Luger, and E. Olsson. Designing a portable architecture for intelligent particle accelerator control. In *Proceedings of the European Particle Accelerator Conference*. pages 2422-2424, 1997.

[37] W. B. Klein, C. Stern, M. Kroupa, R. T. Westervelt, G. Luger, and E. Olsson. Tuning and optimization at Brookhaven and Argonne: results of recent experiments. pages 2535-2537, 1997.

[38] T. Atanasova and J. Zaprianov. Performance of the RBF neural controller for transient stability enhancement of the power system. In *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*. pages 281-283, 1999.

[39] L. Fortuna, A. Gallo, A. Rizzo, and M. G. Xibilia. An innovative intelligent system for fault detection in Tokamak machines. In *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, pages 25-27, 1999.

[40] J. C. Ribes, G. Delauney, J. Delvaux, E. Merle, and M. Mouillet. RBF neural net based classifier for the AIRIX accelerator fault diagnosis. In *Proceedings of the Linac Conference*. pages e-proc. TUC01, 2000.

[41] G. Buceti, L. Fortuna, A. Rizzo, and M.G. Xibilia. An automatic validation system for interferometry density measurements in the ENEA-FTU Tokamak based on soft-computing. In *Proceedings of the International Conference on Accelerator and Large Experimental Physics*

*Control Systems*. pages 343-346, 2001.

[42] B. Sayyar-Rodsari, C. Schweiger, and E. Hartman. Development of PUNDA, final report, 2007. Unpublished.

[43] B. Sayyar-Rodsari, C. Schweiger, E. Hartman, M. Lee, and Y. T Yan. Efficient modeling of nonlinear beam optics using parametric model independent analysis. In *Proceedings of the Particle Accelerator Conference*, pages 1-3, 2005.

[44] B. Sayyar-Rodsari, C. Schweiger, E. Hartman, J. Corbett, M. Lee, P. Lui, and E. Paterson. Parametric modeling of electron beam loss in Synchrotron Light Sources. In *Proceedings of the Particle Accelerator Conference*. pages 3853-3855, 2007.

[45] B. Sayyar-Rodsari, C. Schweiger, E. Hartman, J. Schmerge, M. Lee, P. Lui, and E. Paterson. Parametric modeling of transverse phase space of an RF photoinjector. In *Proceedings of the Particle Accelerator Conference*. pages 3462-3464, 2007.

[46] S. 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 *Proceedings of the Particle Accelerator Conference*. pages 2379-2381, 2007.

[47] J. J. D'Azzo, C. H. Houpis, and S. N. Sheldon. *Linear control system analysis and design with Matlab*. Marcel Dekker, Inc., 2003.

[48] M. Gopal. *Control Systems, Principles and Design*. McGraw Hill, 2008.

[49] H. Kwakernaak and R. Sivan. *Linear optimal control systems*. Wiley Interscience, 1972.

[50] R. N. Bateson. *Introduction to control system technology*. Macmillan Publishing, 1992.

[51] J. Stokes and G. R. L. Sophie. Implementation of PID controllers on the Motorola DSP56000/DSP56001. `http://research.microsoft.com/en-us/um/people/jstokes/implementationofPIDcontrollersAPR5.pdf`.

[52] J. R. Leigh. *Control theory*. The Institution Of Electrical Engineering And Technology, 2004.

[53] C. L. Phillips and H. T. Nagle. *Digital control system design and analysis*. Prentice Hall, 1994.

[54] J. Love. *Process Automation Handbook*. Springer - Verlag. pages 183-189, 2007.

[55] E. H. Anderson and J. P. How. Adaptive feedforward control for actively isolated spacecraft platforms. In *Proceedings of the American*

*Institute of Aeronautics and Astronautics.* pages 1200-1210, 1997.

[56] S. Haykin and B. Widrow. *Least mean square adaptive filters.* Wiley, 2003.

[57] Y. Gong, Y. Song, and S. Liu. Performance analysis of the unconstrained FxLMS algorithm for active noise control. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing.* pages 569-572, 2003.

[58] J. De Cuyper, M. Verhaegen, and J. Swevers. Off-line feed-forward and $H_\infty$ feedback control on a vibration rig. *Control Engineering Practice*, 11:129–140, 2002.

[59] O. S. Alvarez-Salazar and R. Goullioud. Combined feed-forward and feedback control for dim star fringe trecking on the sim test-bed 3. In *Proceedings of the Aerospace Conference*, volume 5. pages 2191-2203, 2003.

[60] K. Dunkel. 100 MeV electron linear accelerator for the Australian Synchrotron Project. Technical report, ACCEL-Linac Design Report Number 1440-BP-1893-B, 2004.

[61] LCLS conceptual design report. `http://ssrl.slac.stanford.edu/lcls/cdr/`. Chapter 7, April 2002.

[62] K. Wittenburg. Beam halo and bunch purity monitoring. In *Proceedings of the CERN Accelerator School Course on Beam Diagnostics.* pages 557-580, 2008.

[63] 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*, 609:79–88, 2008.

[64] H. Winick. *Synchrotron radiation sources.* World Scientific, 1994.

[65] C. D. Chan. Beam position monitor test stand. `http://www.lepp.cornell.edu/~hoff/papers/04reu\_chan.pdf`, September 2004.

[66] Z. Huang, P. Emma M. Borland and, J. Wu, C. Limborg, G. Stupakov, and J. Welch. Suppression of microbunching instability in the Linac Coherent Light Source. In *Proceedings of the European Particle Accelerator Conference.* pages 2206–2208, 2004.

[67] S. Spampinati, S. Di Mitri, and B. Diviacco. A laser heater for FERMI@Elettra. In *Proceedings of the Free Electron Laser Conference.* pages 362–365, 2007.

[68] C. Limborg-Deprey, S. Gierman, and D. Dowell. Modifications of the LCLS photoinjector beamline. In *Proceedings of the European Particle*

*Accelerator Conference*. pages 521-523, 2004.

[69] P. Emma. X-band RF harmonic compensation scheme for linear bunch compression in the LCLS. SLAC technical report LCLS-TN-01-1, 2001.

[70] J. Wu and P. Emma. The Linac Coherent Light Source (LCLS) accerator. SLAC-PUB-12123, 2006.

[71] S. Di Mitri. How to obtain high quality electron bunches in presence of normal conducting linac wakefields. In *Proceedings of the Free Electron Laser Conference*. pages 537–544, 2006.

[72] J. Wu, P. Emma, and Z. Huang. Coherent synchrotron radiation as a diagnostic tool for the LCLS longitudinal feedback system. In *Proceedings of the Particle Accelerator Conference*. pages 428-430, 2005.

[73] J. Frisch, R. Akre, F.-J. Decker, Y. Ding, D. Dowell, P. Emma, S. Gilevich, G. Hays, Ph. Hering, Z. Huang, R. Iverson, R. Johnson, C. Limborg-Deprey, H. Loos, E. Medvedko, A. Miahnahri, H.-D. Nuhn, D. Ratner, S. Smith, J. Turner, J. Welch, W. White, and J. Wu. Beam measurements at the LCLS. In *Proceedings of Beam Instrumentation Workshop*, pages 17-26, 2008.

[74] M. Cornacchia, S. Di Mitri, G. Penco, and A. Zholents. Formation of electron bunches for harmonic cascade X-ray free electron lasers. *Physical Review Special Topics - Accelerators and Beams*, 9:120701, 2006.

[75] M. Cornacchia, S. Di Mitri, and G. Penco. Formation of electron bunches for harmonic cascade X-ray free electron lasers. In *Proceedings of the European Particle Accelerator Conference*. pages 2738-2740, 2006.

[76] L. Badano, M. Ferianis, M. Trovò, and M. Veronese. The beam diagnostic system for the FERMI@Elettra photoinjector. In *Proceedings of Beam Diagnostics and Instrumentation for Particle Accelerators Conference*. pages 171-173, 2007.

[77] J. Wu, P. Emma, and Z. Huang. Coherent synchrotron radiation as a diagnostic tool for the LCLS longitudinal feedback system. In *Proceedings of the Particle Accelerator Conference*. pages 428–430, 2005.

[78] M. Lonza, S. Cleva, S. Di Mitri, O. Ferrando, G. Gaio, A. Lutman, G. Penco, L. Pivetta, and G. Scalamera. Beam-based feedbacks for the FERMI@Elettra free electron laser. In *Proceedings of the International Particle Accelerator Conference*. pages 2758-2760, 2010.

[79] K. L. F. Bane and P. Emma. LiTrack: A fast longitudinal phase space

tracking code with graphical user interface. In *Proceedings of the Particle Accelerator Conference*. pages 4266–4268, 2005.

[80] M. Borland. *A flexible* SDDS *Compliant Code for Acelerator Simulation*. Report Number APS LS-287, September 2000.

[81] R. J. England, J. B. Rosenzweig, G. Andonian, P. Musumeci, G. Travish, and R. Yoder. Sextupole correction of the longitudinal transport of relativistic beams in dispersionless translating sections. *Physical Review Special Topics-Accelerators and Beams*, 8:012801, 2005.

[82] K. Bane. Wakefields of sub-picosecond electron bunches. Technical report, SLAC-PUB-11829, 2006.

[83] K. Flottmann. Astra. `http://tesla.desy.de/~lfroehli/astra/download/Astra%20user%20guide.pdf`, 2008.

[84] K. Ogata. *Modern Control Engineering*. Prentice-Hall, Inc., 2002.

[85] J. Bechhoefer. Feedback for physicists: A tutorial essay on control. *Reviews of modern physics*, 77:783–836, 2005.

[86] K. O. Stanley and R. Miikkulainen. Continual coevolution through complexification. In *Proceedings of the Genetic and Evolutionary Computation Conference*. pages 113–120, 2002.

[87] H. Demuth, M. Beale, and M. Hagan. Neural network toolbox 6 user's guide. `http://www.mathworks.com/help/pdf_doc/nnet/nnet.pdf`, 2008.

[88] L. Ma, K. Xin, , and S. Liu. Using radial basis function neural networks to calibrate water quality model. *World Academy of Science, Engineering and Technology*, 38:385–393, 2008.

[89] M. Awad, H. Pomares, I. Rojas, O. Salameh, and M. Hamdon. Prediction of time series using RBF neural networks: A new approach of clustering. *The International Arab Journal of Information Technology*, 6:138–144, 2009.

[90] D. K. Ranaweera, N. E. Hubele, and A. D. Papalexopoulos. Application of radial basis function neural network model for short-term load forecasting. In *IEE Proceedings: Generation, Transmission and Distribution*. pages 45–50, 1995.

[91] G. W. NG. *Application of neural networks to adaptive control of nonlinear systems*. Research Studies Press, 1997.

[92] L. Pack Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[93] T. Kohonen. The self-organizing map. In *Proceedings of the IEEE*,

volume 78. pages 1464 - 1480, 1990.

[94] A. Zaknick. *Neural networks for intelligent signal processing*. World Scientific, 2003.

[95] N. B. Karayiannis and A. N. Venetsanopoulos. *Artificial Neural Networks*. Kluwer Academic, 1993.

[96] S. Hui and S. H. Żak. The Widrow-Hoff algorithm for McCulloch-Pitts type neurons. *IEEE Transactions on Neural Networks*, 5(6):924–929, 1994.

[97] C. F. N. Cowan S. Chen and P. M. Grant. Othogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309, 1991.

[98] S. Chen, C. F. N. Cowan, and P. M. Grant. Orthogonal least squares learning algorithm for training multioutput radial basis function networks. *IEEE Transactions on Neural Networks*, 2:378 – 384, 1996.

[99] K. L. Du and M. N. S. Swamy. *Neural networks in a softcomputing framework*. Springer, 2006.

[100] D. Baras. Direct policy search in reinforcement learning and synaptic platicity in biological neural networks. `http://webee.technion.ac.il/~rmeir/Theses/DoritThesis.pdf`. Research thesis, 2006.

[101] K. O. Stanley, B. Bryant, and R. Miikkulainen. Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation*, 9:653–668, 2005.

[102] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10:99–127, 2002.

[103] K. O. Stanley. Evolving neural network agents in the NERO video game. In *In Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games*. pages 182–189, 2005.

[104] K. O. Stanley, B. D. Bryant, I. Karpov, and R. Miikkulainen. Real-time evolution of neural networks in the NERO video game. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. pages 1671-1674, 2006.

[105] J. Reeder, R. Miguez, J. Sparks, M. Georgiopoulos, and G. Anagnostopoulos. Interactively evolved modular neural networks for game agent control. In *IEEE Symposium on Computational Intelligence and Games*, pages 167–174, 2008.

[106] K. O. Stanley and R. Miikkulainen. Evolving a roving eye for Go. *Journal of Artificial Intelligence Research*, pages 1226–1238, 2004.

[107] K. O. Stanley and R. Miikkulainen. Competitive coevolution through

evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

[108] K. O. Stanley and R. Miikkulainen. Achieving high-level functionality through complexification. Association for the Advancement of Artificial Intelligence, Technical Report Number SS-03-02, 2003.

[109] K. O. Stanley and R. Miikkulainen. Efficient reinforcement learning through evolving neural network topologies. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 569-577, 2002.

[110] O. Boz. Knowledge integration and rule extraction in neural networks. Technical report, Lehigh University, EECS Department, 1995.

[111] C. H. Yong, K. O. Stanley, and R. Miikkulainen. Incorporating advice into evolution of neural networks. In *Proceedings of the Genetics and Evolutionary Computation Conference*, 2005.

[112] O. Sørensen. Additive feed forward control with neural networks. `http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=250963BE78837C81F327CF410363536B?doi=10.1.1.33.677&rep=rep1&type=pdf`. In *Proceedings of the International Federation of Automatic Control World Congress*, July 1999.

[113] K. J. Hunt and D. Sbarbaro. Neural networks for nonlinear internal model control. In *Proceedings of Institution of Electrical Engineers Conference*, volume 138, September. pages 80-90, 1991.

[114] K. Gurney. *An introduction to neural networks.* John Wiley and Sons, Inc., 1997.

[115] S. A. Billings, H. B. Jamaluddin, and S. Chen. Properties of neural networks with application to modelling non-linear dynamical systems. *International Journal of Control*, 55:1:193–224, 1992.

[116] S. Chen, S. A. Billings, and P. M. Grant. Non-linear system identification using neural networks. *International Journal of Control*, 51:6:1191–1214, 1990.

[117] S. A. Billings and W. S. F. Voon. Correlation based model validity tests for non linear models. *International Journal of Control*, 44:1:235–244, 1986.

[118] M. J. L. Orr. Introduction to radial basis function networks. `http://anc.ed.ac.uk/rbf/papers/intro.ps`, April 1996.

[119] 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*, 610:629–635, 2009.

[120] 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*, 632:1–6, 2011.

[121] R. Miikkulainen, B. D. Bryant, R. Cornelius, I. V. Karpov, K. O. Stanley, and C. H. Yong. Computational intelligence in games. In *Computational Intelligence: Principles and Practice. Piscataway, NJ: IEEE Computational Intelligence Society*. pages 155–191, 2006.

[122] T. D'Silva, M. Chrien R. Janik, O. Stanley, and R. Miikkulainen. Retaining learned behavior during real-time neuroevolution. `http://nn.cs.utexas.edu/downloads/papers/dsilva.aiide05.pdf`, 2005.